КВАЛІФІКАЦІЙНА РОБОТА

Група МІПЗс-22

Габурак М.М.

2024

ЗВО УНІВЕРСИТЕТ КОРОЛЯ ДАНИЛА

Факультет суспільних та прикладних наук

Кафедра інформаційних технологій

на правах рукопису

Габурак Михайло Мирославович

УДК 004.4

Імплементація моделей покращення ефективності та оптимальності сервіс-орієнтованого проектування програмних рішень

Спеціальність 121 – «Інженерія програмного забезпечення»

Кваліфікаційна робота на здобуття кваліфікації магістра

Нормоконтроль

_____Стисло О.В.

(підпис, дата, розшифрування підпису)

Студент

_____ Габурак М.М.

(підпис, дата, розшифрування підпису)

Допускається до захисту

Завідувач кафедри

_____к.т.н., доц. Ващишак С.П.

(підпис, дата, розшифрування підпису)

Керівник роботи

_____ к.т.н., доц. Демчина М.М.

(підпис, дата, розшифрування підпису)

Івано-Франківськ – 2024

ЗВО УНІВЕРСИТЕТ КОРОЛЯ ДАНИЛА

Факультет суспільних та прикладних наук

Кафедра інформаційних технологій

Освітній ступінь: «магістр»

Спеціальність: 121 «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____

« 19 » лютого 2024 року

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Габураку Михайлу Мирославовичу

(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи

Імплементація моделей покращення ефективності та оптимальності сервіс-орієнтованого проектування програмних рішень

керівник роботи:

Демчина Микола Миколайович, кандидат технічних наук, доцент

затверджена наказом вищого навчального закладу від « 26 » червня 2023 року № 32/1 с

2. Термін подання студентом роботи 16.02.2024

3. Вихідні дані роботи: Формальні моделі, методи та алгоритми.

4. Зміст кваліфікаційної роботи (перелік питань, які потрібно розробити)

1. Аналіз сервіс-орієнтованої архітектури побудови програмних рішень.
2. Структуризація засобів сервіс-орієнтованого моделювання.
3. Розробка шаблонів аналізу для сервіс-орієнтованого проектування
4. Імплементація сервіс-орієнтованих моделей на основі шаблонів

5. Дата видачі завдання 29.06.2023

КОНСУЛЬТАНТИ РОЗДІЛІВ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Розділ	Консультант (прізвище, ініціали та посада)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Термін виконання етапів роботи	Примітка
1.	Аналіз сервіс-орієнтованої архітектури побудови програмних рішень.	26.09.2023	Виконано
2.	Дослідження та структуризація засобів сервіс-орієнтованого моделювання.	20.10.2023	Виконано
3.	Розробка шаблонів аналізу для сервіс-орієнтованого проектування	15.11.2023	Виконано
4.	Імплементація сервіс-орієнтованих моделей на основі шаблонів	30.11.2023	Виконано
5.	Формування висновків	09.12.2023	Виконано
6.	Оформлення пояснювальної записки	22.12.2023	Виконано
7.	Оформлення графічного матеріалу та підготовка до захисту роботи	11.01.2024	Виконано

Студент _____ Габурак М.М.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Демчина М.М.
(підпис) (прізвище та ініціали)

Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

Сторінка	Опис графічного матеріалу	Сторінка	Опис графічного матеріалу
23	Концепції, що стосуються архітектури підприємства	58	Залежності взаємодії та переходу
26	Гетерогенні архітектурні домени	61	Представлення архітектурної основи
29	Структура MDA і три рівні моделювання	65	Залежності акторів
32	Сервіс як цикл взаємодії	67	Приклад семантичних залежностей між акторами
36	Архітектурна основа для сервіс-орієнтованого моделювання	68	Позначення інформації, фізичних потоків і потоків рішень

АНОТАЦІЯ

Кваліфікаційна робота присвячена дослідженню методів та імплементації моделей покращення ефективності та оптимальності сервіс-орієнтованого проектування програмних рішень шляхом розробки методу сервіс-орієнтованого моделювання для аналізу та проектування інформаційних систем.

В першому розділі проаналізовано концепції сервіс-орієнтованої архітектури побудови програмних рішень, виконано дослідження предметної області проектування інформаційних систем на основі сервіс-орієнтованої архітектури. Описано процеси проектування архітектури інформаційних систем, наведена архітектура керована моделлю, рівні моделювання та послуги в сервіс-орієнтованому аналізі та проектуванні

В другому розділі виконано структуризацію засобів сервіс-орієнтованого проектування та моделювання програмних рішень, описано процеси сервіс-орієнтованого моделювання. Представлена семантична структура програмного рішення та сутність сервісно-орієнтованої мови моделювання.

В третьому розділі проведена імплементація сервіс-орієнтованих моделей на основі шаблонів для покращення ефективності проектування програмних рішень, наведена сутність пропонованого методу сервіс-орієнтованого моделювання. Здійснена розробка методу сервіс-орієнтованого моделювання для аналізу та проектування інформаційних систем, представлено семантичні конструкції сервіс-орієнтованого моделювання та виконана розробка шаблонів аналізу для покращення ефективності сервіс-орієнтованого аналізу та проектування.

КЛЮЧОВІ СЛОВА: СЕРВІС-ОРІЄНТОВАНА АРХІТЕКТУРА, МОДЕЛЮВАННЯ ПРОЦЕСІВ, ІНФОРМАЦІЙНА СИСТЕМА, КЕРУВАННЯ МОДЕЛЛЮ, ШАБЛОНИ ПРОЕКТУВАННЯ.

SUMMARY

The qualification work is devoted to the research of methods and implementation of models for improving the efficiency and optimality of service-oriented design of software solutions by developing a method of service-oriented modeling for the analysis and design of information systems.

In the first section, the concepts of service-oriented architecture for building software solutions are analyzed, the subject area of designing information systems based on service-oriented architecture is studied. Information systems architecture design processes are described, model-driven architecture, modeling levels and services in service-oriented analysis and design are given

In the second section, the structuring of the means of service-oriented design and modeling of software solutions is performed, the processes of service-oriented modeling are described. The semantic structure of the software solution and the essence of the service-oriented modeling language are presented.

In the third section, the implementation of service-oriented models based on templates is carried out to improve the efficiency of designing software solutions, the essence of the proposed method of service-oriented modeling is given. The service-oriented modeling method for the analysis and design of information systems was developed, the semantic constructions of service-oriented modeling were presented, and the analysis templates were developed to improve the efficiency of service-oriented analysis and design.

KEY WORDS: SERVICE-ORIENTED ARCHITECTURE, PROCESS MODELING, INFORMATION SYSTEM, MODEL MANAGEMENT, DESIGN PATTERNS.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

ІС - інформаційна система

ІТ - інформаційна технологія

EM - Enterprise Modelling

SOA - Service-Oriented Architecture

UML - Unified Modelling Language

OOD - Object-oriented design

OPM - Object Process Methodology

EA - Enterprise Architecture

MDA - Model Driven Architecture

CIM - Computation Independent Model

PIM - Platform Independent Model

PSM - Platform Specific Model

OMG - Object Management Group

ВСТУП

Актуальність дослідження. З точки зору бізнесу, послуга – це чітко визначена, інкапсульована, багаторазово використовувана, узгоджена з бізнесом сервісна здатність. Сервісна операція є елементарною частиною служби та вказує відповідні входи, мету (функція, обов'язок або зобов'язання) і результати (артефакти, продукти, результати). Послуга повністю визначається описом, а опублікований документ або артефакт – описує мету послуги та її внески, мету, результати, обсяг, відповідальність, управління, стійкість (термін надання, обслуговування) та якість надання послуг.

З точки зору інформаційних технологій (IT), служба — це доступний для виявлення ресурс програмного забезпечення, який має опис служби та інтерфейс і налаштовується за допомогою сценаріїв. Опис сервісу доступний для пошуку, прив'язки та виклику споживачем послуги. Реалізація опису імплементується через постачальника послуг, який надає якісні послуги (QoS). З появою SOA, принципи програмування що було наслідувано з ООП програма для інтерфейсів а не реалізації — було віднесено до рівня архітектури програмного забезпечення, в якому фактично логічний рівень використовується для опису послуг. Це часто випадок, до якого розвиваються концепції програмування, архітектурні конструкції, а потім методи.

Початкові спроби використовувати традиційні методи OOAD для підтримки веб-служб і SOA знайшли ці методи недостатніми для підтримки викликів і нюансів необхідних для нових першокласних структур обслуговування, а саме послуги, компоненти, що реалізують ці послуги, потоки бізнесу, процеси хореографії цих послуг, і інформаційні вимоги та пов’язані сценарії з послугами. Якщо уточнити, послуги будуть призначені як механізм логічного інтерфейсу для відокремлення споживача послуги від послуги постачальника через протокол на основі стандартів, який дозволяє викликати розподілені програмні ресурси на основі потреби.

Подальший опис того, як послуги взаємодіють із їхніми базовими реалізаціями, символізовані в компонентах архітектури програми так як вони будуть об'єднані та складені в нове поняття, названі терміном хореографія або оркеструванням бізнес-процесів потоків, що вимагає нового набору конструкцій методів. Це не означає, що це єдиний спосіб використання інструментів послуг через композицію та хореографію. Послуги також можна використовувати, на різноманітті рівнів: наприклад, послуги заміни застарілих функцій, прості служби надання нового функціоналу, використання послуг для збільшення повторного використання та рефакторингу, щоб зменшити вартість обслуговування системи, послуги, що використовуються для видобування бізнес-значущих узагальнень та інші комплексні послуги.

Мета і завдання дослідження. Метою кваліфікаційної роботи є аналіз, дослідження та проектування інформаційних програмних рішень з використанням архітектури керованої моделлю для покращення ефективності процесів сервіс-орієнтованого проектування програмних рішень.

Для досягнення поставленої мети необхідно розв’язати такі завдання: 1) виконати аналіз предметної області сервіс-орієнтованої архітектури побудови програмних рішень; 2) дослідити архітектуру керовану моделлю та описати рівні моделювання; 3) дослідити семантичну структуру програмного рішення під час проектування; 4) представити структуризацію та опис засобів сервіс-орієнтованого моделювання; 5) виконати розробку шаблонів аналізу для покращення ефективності сервіс-орієнтованого аналізу та проектування; виконати імплементацію сервіс-орієнтованих моделей на основі шаблонів.

Об’єктом дослідження є самі програмні рішення, що розробляються або вже існують на основі сервіс-орієнтованої архітектури. Це можуть бути системи з різними рівнями складності, від простих мікросервісів до складних розподілених систем.

Предметом дослідження є імплементація моделей покращення ефективності та оптимальності в контексті сервіс-орієнтованого проектування програмних рішень. Тобто, вивчення та застосування методів, які дозволяють покращити якість, ефективність та оптимальність програмних рішень, що базуються на сервіс-орієнтованій архітектурі.

Методи дослідження базуються на аналізі шаблонів проектування програмного забезпечення та використанні моделей семантики архітектурних конструкцій програм.

Наукова новизна одержаних результатів полягає у тому, що на основі ґрунтовного аналізу предметної області проектування інформаційних систем запропоновано метод сервіс-орієнтованого моделювання програмних рішень який містить прагматичну специфікація інформаційних систем, сервіс-орієнтовану основа моделювання для аналізу та проектування інформаційних систем та принципи переходу до конкретної реалізації на основі шаблонів.

Практичне значення одержаних результатів полягає в реалізації оптимальної моделі життєвого циклу сервіс-орієнтованого проектування програмних рішень, яка використовує послідовні ітерації для ефективного аналізу, проектування, реалізації та розгортки проектів сервіс-орієнтованої архітектури.

Апробація результатів дослідження. Матеріали дослідження було представлено у матеріалах Міжнародної наукової інтернет-конференції “Інформаційне суспільство: технологічні, економічні та технічні аспекти становлення” (випуск 82), у тезах доповіді “Застосування штучного інтелекту в сервісно-орієнтованому дизайні програмного забезпечення”.

Структура. Кількість розділів – 3. Загальний обсяг основної частини – 87 сторінок. Список використаних джерел містить – 50 позицій.

РОЗДІЛ 1. АНАЛІЗ КОНЦЕПЦІЇ СЕРВІС-ОРІЄНТОВАНОЇ АРХІТЕКТУРИ ПОБУДОВИ ПРОГРАМНИХ РІШЕНЬ

1.1 Дослідження предметної області проектування інформаційних систем на основі сервіс-орієнтованої архітектури

Для підтримки процесу аналізу та проектування інформаційних систем (ІС) використовуються різні моделі та методи, але після багатьох років практики залишається багато питань і невирішених проблем, які призводять до провалу проектів з розробки ІС. Однією з причин є те, що швидкі зміни в бізнес-середовищі викликають необхідність впровадження нових рішень для бізнес-дизайну, які повинні ефективно підтримуватися комп'ютеризованими інформаційними системами. Такі ситуації збільшують складність специфікацій інформаційних систем і створюють труднощі для організацій у досягненні бізнес-цілей. Безпрецедентні темпи змін у бізнес-процесах і технологічний прогрес ускладнили команді розробників інформаційних систем бути більш гнучким у реагуванні на мінливі вимоги [1]. Бізнес-експерти, а також розробники систем не забезпечені методами, які підтримують систематичний процес розробки системи через організаційні та технічні межі системи. Не вистачає методу, який би надавав систематичні керівні принципи для узгодження загального дизайну ІС з цілями підприємства [2]. Традиційні методи моделювання не дають вказівок щодо того, як прагматичні специфікації, що мотивують бізнес-дизайн, можна пов’язати зі структурними та динамічними аспектами специфікацій ІС, які представляють дані та бізнес-процеси [3]. Відсутність методу концептуального моделювання, який допомагає виявити проблеми семантичної цілісності специфікацій ІС, створює розрив у спілкуванні між бізнесом та експертами з інформаційних технологій (ІТ). Таким чином, ця

hj,jnf представляє новий концептуальний метод моделювання, який кидає виклик існуючим проблемам інтеграції специфікацій ІС.

Що стосується розробки нового методу моделювання ІС, необхідно представити деякі важливі теми в області розробки інформаційних систем. Це необхідно, щоб мотивувати, а також пояснити, що слід робити і як цей новий метод слід розробляти, щоб сприяти гнучкості реінжинірингу ІС та керувати складністю.

Найуспішнішими є ті організації, які планують необхідні зміни та здатні їх реалізувати [4]. Така ситуація потребує, перш за все, змін на організаційному рівні, які призводять до постійної реорганізації та редизайну бізнес-процесів. Спільним для різних типів комп'ютеризованих інформаційних систем є те, що всі вони вбудовані в організацію з метою підвищення ефективності та результативності організацій [5]. Інформаційні системи, як правило, охоплюють кордони навколишнього середовища, організації та комп'ютеризованої інформаційної системи. Щоб знати та розуміти інформаційні системи, необхідно брати до уваги організаційне середовище, яке є більш широкою системою, що складається з комп'ютеризованих та некомп'ютеризованих частин системи. Складність інформаційних систем викликає труднощі у узгодженні організаційних підсистем з IT-підсистемами. Це одна з найскладніших проблем, з якими стикається сьогоднішній процес аналізу та проектування інформаційних систем [5].

Аналіз і проектування інформаційних систем є двома різними видами діяльності в процесі розробки системи. Вони базуються на розумінні організаційних цілей, структури та процесів [6]. У цій діяльності беруть участь різні зацікавлені сторони з різними поглядами, цілями та досвідом: люди, яким потрібен продукт, ті, хто його проектує та створює, і ті, хто його розгортає. Розуміння та взаємна згода між зацікавленими сторонами має вирішальне значення для успішної комунікації, яка вимагає спільної мови. Діяльність з моделювання підприємства (EM - Enterprise Modelling) [7] іноді

використовується для підтримки ранніх фаз розробки ІС з точки зору цільових моделей [8]. ЕМ досить часто посилається на вираження корпоративних знань, які забезпечують цілісне уявлення про бізнес-цілі, процеси, правила, концепції, акторів, ресурси та відповідні специфічні аспекти впровадження. Поняття ЕМ часто використовується як взаємозамінне з бізнес-моделюванням, оскільки бізнес-моделювання також розглядає всі виміри ЕМ, такі як процеси, рішення та інформація [9]. Цілісне розуміння вимог підприємства є критично важливим для бізнес-експертів, які визначають організаційні стратегії. ЕМ важливий, оскільки він допомагає визначити, як компоненти інформаційної системи підтримують певну бізнес-діяльність і чому цей компонент корисний. Моделі підприємств також допомагають зрозуміти, чому виконуються бізнес-процеси та як вони сприяють досягненню цілей організації.

ЕМ та інтеграція є двома критичними кроками в аналізі та проектуванні ІС [10]. ЕМ є основною технікою для роботи зі складністю, яка притаманна дизайну та зміні бізнес-процесів. Цей процес моделювання важливий, оскільки він допомагає розробникам систем візуалізувати, специфікувати, конструювати та документувати статичні та динамічні аспекти специфікацій ІС [11]. Статичні аспекти представляють структурні характеристики системи, підкреслюючи частини, які складають систему. Динамічні аспекти визначають поведінкові та інтерактивні характеристики системи, наприклад, показуючи, як система поводиться у відповідь на зовнішні події. Моделі, які використовуються для представлення цих аспектів, можна розглядати як ключ до успішного процесу інтеграції, оскільки вони зосереджені на взаємодії між різними аспектами системи. Інтегровані моделі повинні сприяти процесу валідації та верифікації [12], процесу перевірки того, чи відповідає система специфікаціям і чи виконує вона заплановану мету. Цей процес дуже важливий для досягнення кращої якості специфікацій системи. Процес перевірки зазвичай повертається до потреб користувача та виражається запитанням «Чи правильно ви створюєте?» Для цього потрібні докази того,

що кінцевий продукт відповідає запланованим вимогам. Це передбачає перевірку відповідності специфікацій системи намірам людей, заявленим цілям. Перевірка часто є внутрішнім процесом, який виражається запитом «Чи правильно ви створюєте річ?» Щоб підтримати розробку систематичного та інтегрованого методу моделювання, моє дослідження зосереджено на трьох основних темах, перелічених нижче.

Сервіс-орієнтований метод моделювання повинен допомогти проаналізувати перші вимоги до ІС з точки зору цілей проектування [13]. У цій роботі ранні етапи аналізу підтримуються прагматичною специфікацією, яка є важливою для мотивації виміру «чому». Прагматичний метод моделювання, орієнтований на послуги, повинен забезпечити спосіб аналізу бізнес-намірів, які допомагають мотивувати статичні та динамічні аспекти специфікацій ІБ.

Традиційні методи розробки ІС проектують статичні та динамічні аспекти в різні типи графічних зображень. Це створює труднощі в досягненні семантичної цілісності між різними вимірами архітектури підприємства. Статичні та динамічні аспекти є додатковими. Вони описують один і той самий артефакт і тому не можуть аналізуватися ізольовано. Ізоляція архітектурних поглядів і розмірів інформаційних систем створює труднощі у виявленні неузгодженості та неповноти вимог ІС [14].

Сервісно-орієнтований спосіб моделювання має бути нейтральним щодо обчислень, оскільки на процес аналізу ІС не повинні впливати жодні рішення щодо реалізації. Ця властивість має вирішальне значення для подолання розриву в спілкуванні між бізнесом та ІТ-фахівцями. Концептуальні моделі розроблені для підтримки спілкування між зацікавленими сторонами. Моделі та методи повинні сприяти однозначному обміну проектними рішеннями між зацікавленими сторонами та забезпечувати відстежуваність вимог. Моделі повинні допомогти бізнес-експертам і розробникам ІС у перевірці та верифікації специфікацій системи. Для побудови відповідної ІС вкрай важливо, щоб термінологія та

уявлення, які використовуються як засіб комунікації, були однозначними та зрозумілими. Важливою якістю артефактів моделювання є їх зрозумілість. Якщо графічні представлення, які використовуються в процесі розробки, не відображають семантику намічених вимог, вони не можуть служити основою для комунікативних цілей).

Інтегрований спосіб моделювання системних послуг бізнесу та інформаційних технологій (ІТ) є незамінним у поточній практиці реінжинірингу [15]. Це є необхідною умовою для розвитку цілісного розуміння та планування впорядкованих процесів переходу від поточної до нової ситуації системи підприємства. Досягнення інтеграції є проблематичним через відмінності в методах архітектурного моделювання [16]. Їм бракує спільної мови для спільних знань, що важливо для подолання розривів у спілкуванні між зацікавленими сторонами за допомогою спільних бізнес-моделей і методів проектування. Поточна традиція розробки ІС має тенденцію відволікати увагу від стратегічних аспектів бізнес-моделювання та зосереджуватися на специфічних для реалізації артефактах. Така ситуація породжує дві проблеми. По-перше, ті самі методи моделювання, що залежать від реалізації, використовуються на етапах аналізу та проектування. Це збільшує розрив у спілкуванні між зацікавленими сторонами, які не є ІТ-фахівцями. По-друге, аналіз цілей та інших бізнес-орієнтованих аспектів залишається позаду.

Орієнтація на послуги є однією з найбільш швидкозростаючих парадигм розвитку ІС [17]. Це не просто нова парадигма розробки програмного забезпечення, а й ширша тема архітектури підприємства. Сервісно-орієнтована парадигма має потенціал для вирішення проблем інтеграції специфікації ІС, згаданих вище, оскільки феномен сервісу можна розглядати як зв’язок між різними параметрами моделювання підприємства. Метою цієї роботи є розробка методу моделювання для аналізу та проектування інформаційної системи, де феномен сервісу та принципи

орієнтації на сервіс можуть бути застосовані для аналізу та проектування ІС через організаційні та технічні межі системи.

Сервісно-орієнтований спосіб моделювання повинен базуватися на онтологічних принципах сервісів і загальному розумінні загальної структури сервісу, на який не впливають жодні рішення реалізації. Онтологічна природа концепції послуги [18] забезпечує новий спосіб мислення, який базується на взаємодії послуги між організаційними або технічними компонентами системи. Послугу можна визначити як набір упорядкованих і цілеспрямованих взаємодій між акторами, які можна розглядати як різні людські, організаційні або технічні компоненти. Це важливо, оскільки розуміння архітектури корпоративної системи залежить від знання того, як різні підсистеми взаємопов'язані. Систему підприємства можна розглядати як композицію взаємодіючих компонентів, які розглядаються як запитувачі послуг і постачальники послуг. Процес аналізу та проектування ІС повинен мати набагато ширший погляд на орієнтацію на сервіс, ніж Сервіс-орієнтована архітектура (SOA - Service-Oriented Architecture), яка є архітектурним підходом для побудови складних програмних систем із набору будівельних блоків, які називаються сервісами.

Сервісно-орієнтоване моделювання має підтримувати принцип поділу проблем [19], який є одним із керівних принципів управління складністю ІС. Здатність застосовувати принцип поділу інтересів допомагає досягти бажаної гнучкості для повторного використання та управління змінами. Сервісно-орієнтований спосіб моделювання дозволяє чітко моделювати потоки взаємодії, що має вирішальне значення для виявлення розривів у специфікаціях ІС та для розуміння деталей наскрізних проблем між різними підсистемами підприємства. Новий спосіб моделювання ІС має базуватися на сервіс-орієнтованій парадигмі, яка дозволяє інтегрувати структурні та динамічні аспекти концептуалізацій ІС.

Зміни в архітектурі сервісу необхідно постійно фіксувати, візуалізувати та погоджувати. Вони мають вирішальне значення для узгодження дизайну

бізнес-системи та технічної системи. Сервісна орієнтація — це архітектурний стиль, який допомагає зрозуміти бізнес-процеси як композиції послуг, які можна змінити шляхом заміни окремих послуг. Використовуючи сервіс-орієнтований спосіб моделювання, інформаційні системи можна структурно візуалізувати як розвиваються концептуалізації сервісних архітектур. Архітектура сервісу визначається бізнес-процесами, які є композиціями організаційних або технічних сервісів.

Новий спосіб ІС, заснований на сервісно-орієнтованій парадигмі, дозволяє інтегрувати структурні та динамічні аспекти концептуалізацій ІС, що є критично важливим для інтеграції різних архітектурних доменів і для досягнення цілісного уявлення про архітектуру. Щоб зберегти цілісне уявлення про архітектуру підприємства та зрозуміти функціональність інформаційних систем, підсистеми підприємства слід визначати відповідно до цілей проектування. Комплексний спосіб бізнес-моделювання не може відокремити бізнес-орієнтовані деталі послуг від структурних аспектів, аспектів взаємодії та поведінки. Метод моделювання повинен забезпечити мотивацію для впровадження компонентів програмного забезпечення та допомогти виявити проблеми проектування на ранніх етапах аналізу ІС.

Щоб досягти консенсусу та розуміння між зацікавленими сторонами, необхідно розробити метод, який забезпечує мову, процес моделювання та методи моделювання для систематичного аналізу та проектування ІБ. Процес і методи моделювання повинні допомогти виявити семантичні розбіжності специфікацій ІС на різних рівнях абстракції. Будучи прагматичним, метод моделювання повинен забезпечувати керівні принципи для аналізу неповноти специфікацій ІС щодо цілей. Новий метод має допомогти бізнес-експертам і системним дизайнерам визначати, візуалізувати та оцінювати різні організаційні зміни за допомогою повністю графічного підходу до реінжинірингу ІС. Це, у свою чергу, допоможе узгодити дизайн бізнес-процесів з організаційними та технічними компонентами. Нейтральні до обчислень представлення архітектур сервісів можна використовувати як

посібники, які підтримують спілкування між зацікавленими сторонами під час аналізу інформаційних систем і процесу проектування.

1.2 Аналіз та проектування архітектури інформаційних систем

Інформаційна система – це система, яка збирає, зберігає, обробляє та перетворює інформацію [20]. Кожна інформаційна система працює в контексті організації та має задовольняти її вимоги [21]. Інформаційні системи розроблені для підтримки обміну інформацією в організаціях з метою підвищення ефективності та ефективності організації. Роль будь-якої інформаційної системи в організації можна розглядати як посередника між бізнесом та інфраструктурою інформаційних технологій організації. Щоб визначити, як комп’ютеризована інформаційна система може підтримувати конкретну бізнес-діяльність, важливо зрозуміти, як ця діяльність виконується та як вона сприяє досягненню цілей організації. Щоб розробити та зрозуміти інформаційні системи, необхідно добре розуміти контекст організацій, які використовують ці системи. Розробка інформаційних систем — це спосіб розробки, аналізу, проектування та впровадження інформаційних систем.

Аналіз та проектування інформаційних систем є двома різними видами діяльності в процесі розробки інформаційних систем. Ці дві види діяльності базуються на розумінні цілей організації, її структури та процесів, а також того, як інформаційні технології (IT) можна використовувати на користь бізнесу. Необхідно, щоб специфікації допоміжних технічних систем були вмотивовані та обґрунтовані в контексті моделей організаційних процесів. Розробники систем повинні розуміти технічну архітектуру системи та організаційну інфраструктуру, де буде встановлено додаток. Ключовим питанням тут є визначення справжніх IT-потреб і того, як ці потреби інтегровані в загальну організаційну систему для їх підтримки.

Процеси аналізу та проектування інформаційних систем включають різні зацікавлені сторони з різними цілями та досвідом. Аналіз

зосереджується на кращому розумінні вимог, перш ніж намагатися знайти рішення. Це дослідження існуючої системи та визначення вимог до їх вирішення. Мета фази аналізу полягає в тому, щоб з'ясувати, що потрібно бізнесу, щоб охопити загальну картину, але уникнути деталей впровадження. Що важливо на етапі аналізу, так це можливість повністю визначити предметну область без упередження будь-якої конкретної реалізації.

Фаза системного аналізу відповідає на питання, хто використовуватиме систему, що вона робитиме, де і коли використовуватиметься. Фаза зосереджена на визначенні системних вимог, метою якої є визначення функціональних та інших вимог, які зацікавлені сторони очікують дотримуватися під час впровадження. Він визначає бізнес-вимоги до нової системи. Він визначає, що має бути зроблено, а не як це має бути зроблено. У процесі аналізу розробники розглядають доступні джерела інформації та намагаються вирішити можливі неясності. Аналіз — це процес створення моделей, які фіксують основні характеристики бажаної системи.

Проектування — це спосіб перевірки того, що заплановане рішення дійсно відповідає потребам ситуації, перш ніж воно буде втілене в життя. Проектування системи забезпечує специфікації рішення інформаційної системи, визначені на етапі аналізу. Дизайн іноді розглядається як прогресивне удосконалення від абстрактного до конкретного та конкретного. Він визначає, як система виконуватиме цілі, визначені на етапі аналізу. Він показує, як поєднуються технічні та організаційні компоненти. Процес проектування складний, оскільки це не суто аналітичне завдання.

Щоб контролювати або керувати необхідними змінами та розуміти систему підприємства в цілому, організація повинна мати архітектуру, яка керує проектуванням та еволюцією підприємства. Архітектура виражає характерні особливості, структуру, поведінку та взаємозв'язки конкретного артефакту [22].

Відповідно до стандарту IEEE, архітектура є фундаментальною організацією системи, що складається з компонентів, їхніх зв'язків один з

одним і з навколишнім середовищем, а також принципу, який керує її проектуванням та еволюцією. Процес проектування є важливим для подолання розриву між стратегічними проблемами організацій та їх реалізацією.

Архітектура в контексті організації називається «архітектурою підприємства». Поняття підприємства в контексті розвитку інформаційної системи позначає обмежену сферу діяльності в організації, яка становить інтерес для різних зацікавлених сторін. Згідно з [23], підприємство — це суб'єкт, що постійно розвивається, постійно змінюється у відповідь на зовнішні та внутрішні впливи. Він формується для виробництва продукту або надання послуги. Це цілеспрямована система, яка має створювати цінність, яка зазвичай виражається баченням, цілями або досяжними цілями. Підприємство – це ширше розуміння організації. Це складна та цілісна система, що охоплює організаційні, бізнес-процеси та технічні погляди, які потребують архітектури. У цій роботі концепція корпоративної системи розуміється як будь-яка проблемна область: це може бути організація або набір організацій, які підтримують свій бізнес за допомогою комп'ютеризованих інформаційних систем.

Архітектура підприємства (EA - Enterprise Architecture) — це набір стратегічних та архітектурних дисциплін, які охоплюють інформаційну, бізнес та технічну архітектуру [24]. Архітектура підприємства важлива як інструмент для вирішення проблеми інтеграції різних архітектурних доменів у масштабах підприємства. EA є важливою в контексті розвитку інформаційних систем, оскільки вона дає розуміння структури організації, бізнес-процесів і технологій, які підтримують бізнес і складають усе підприємство. Він описує узгоджене ціле принципів, методів і моделей, які використовуються під час розробки та реалізації організаційної структури підприємства, бізнес-процесів, інформаційних систем та інфраструктури.

Існує ряд цінних понять, які є значущими і відображають загальну архітектурну характеристику підприємства. Ці поняття необхідні для

розуміння розвитку інформаційної системи. Ключові поняття та зв'язки між ними, пов'язані з архітектурою підприємства (EA) (рис. 1.1).

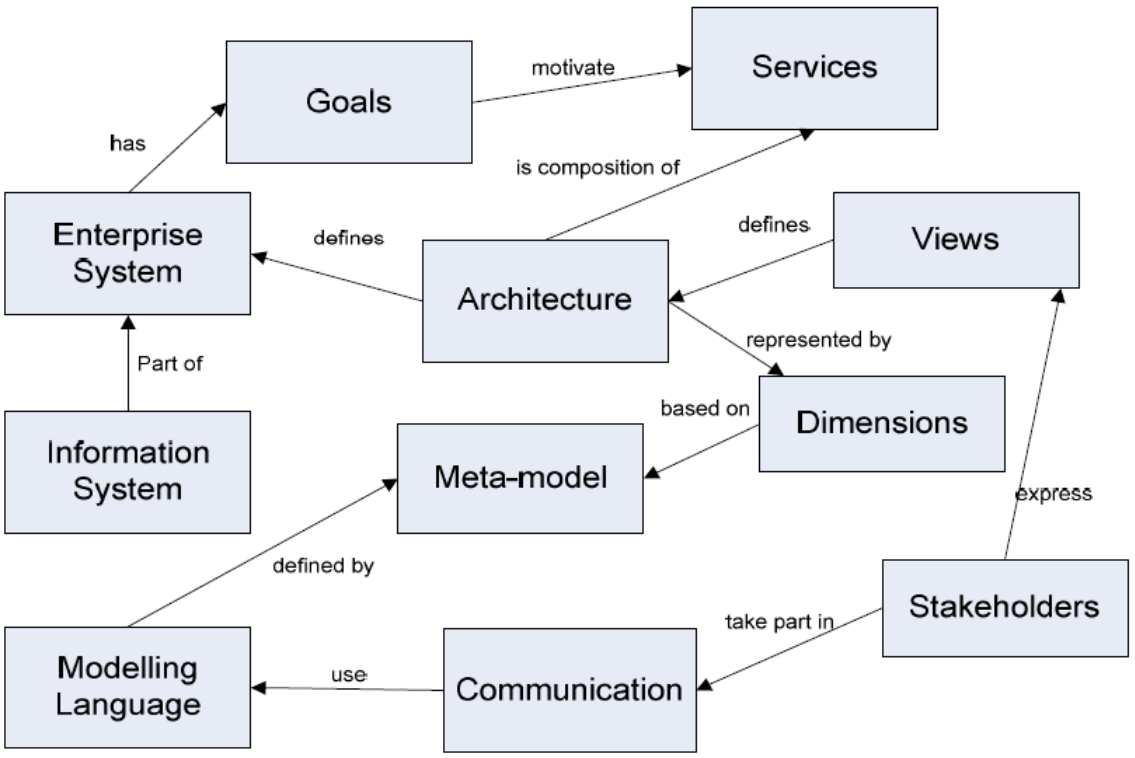


Рисунок 1.1 – Концепції, що стосуються архітектури підприємства

Інформаційна система є вбудованою в систему підприємства. Автономні послуги мотивовані різними цілями корпоративної системи. Архітектура визначає систему підприємства, яка виражається поглядами різних зацікавлених сторін і може бути представлена в різних вимірах. За кожним графічним зображенням лежить опис точки зору зацікавлених сторін, які разом забезпечують інтегрований архітектурний опис корпоративної системи. Розміри базуються на різних моделях, які визначаються різними мовами моделювання. Семантичні моделі, які розглядаються як метамоделі, визначають нотацію моделювання графічної мови. Мова моделювання необхідна для однозначного розуміння діаграм і для спілкування між зацікавленими сторонами. Метою спілкування є подолання розриву за допомогою спільних моделей і методів з метою розробки описів EA.

Конструкція ЕА надає способи доставки бізнес-рішень, які підтримуються технологією. ЕА характеризується використанням фреймворків, які підтримують аналіз від бізнес-орієнтованого рівня до рівня ІТ. Структуру можна розглядати як ключ до розуміння архітектури підприємства, динаміки організації. Фреймворк Zachman [26] є одним із фреймворків, які складають архітектуру підприємства. Це двовимірна схема, що представляє шість точок зору планувальника, власника, дизайнера, будівельника, субпідрядника та функціонуючої системи. Щоб зрозуміти важливість створення та конструювання нового артефакту, необхідно вивчити майбутнє створення артефакту з шести різних аспектів: мотивація, організація та місце розташування, матеріали, процес будівництва та координація часу. Структура представляє шість стовпців для визначення різних аспектів: речі, процеси, зв'язок, люди, час і мотивація. Як правило, це визначається з різних точок зору, наприклад «що», «як», «де», «хто», «коли» і «чому». Структура згідно з [27] наведена нижче в таблиці 1.1.

Таблиця 1.1

Матриця різних представлень архітектури підприємства

Aspects/ Views	**Data (What)**	**Function (How)**	**Network (Where)**	**People (Who)**	**Time (When)**	**Motivation (Why)**
Scope **Planner view**	List of concepts	List of processes	List of locations	List of organisational units	List of business events	List of business goals
Organisational system **Owner view**	Entity Relation-ship diagram	Business Process diagram	Diagram of logical network	Organisational Decomposition chart with roles	Schedule charts	Business Strategy Plan
Information System **Designer view**	Logical Data Architec-ture	Software Application Architecture	Distribut ed system Architec-ture	Human Interface Architecture	Control structure	Constraints and Rules
Technology **Builder view**	Physical Data Architec-ture	Deployment Architecture	Techno-logy Architec-ture	Presentation/ Layouts structure	Compo-nent control structure	Rule design
Representations **Subcontractor view**	Data definition	Process design	Network Architec-ture	Interface Architecture	Timing definition	Rule specification
Functioning System	Data	Components	Network	Organisation	Schedule	Strategy

Архітектура підприємства відноситься до різних типів графічних зображень, які визначають різні архітектурні домени. Він визначає, як бізнес, дані, технології та структури програмного забезпечення сприймаються з різних точок зору. «Що» представляє структурні аспекти, «як і коли» представляють поведінкові аспекти, а «де» і «хто» представляють інтерактивні аспекти архітектури підприємства.

Згідно з сервіс-орієнтованою структурою (див. рисунок 1.3), вимір «чому» відповідає прагматичному рівню. На цьому рівні проводиться бізнес-орієнтований аналіз. Аналіз призводить до прагматичних специфікацій, які мотивують і керують подальшими процесами аналізу та проектування на семантичному та синтаксичному рівнях. Структурні, інтерактивні та поведінкові аспекти інтегровані за допомогою сервіс-орієнтованого методу моделювання на семантичному рівні. Це концептуальний рівень, який відповідає погляду власника. Логічний дизайн представлений синтаксичним рівнем, де можуть бути заповнені орієнтовані на реалізацію проекції різних діаграм. Існують різні способи створення цінності для організацій, але створення цінності завжди відбувається в робочому процесі, який можна розглядати з різних точок зору. Інтеграція цих перспектив сприятиме розумінню та можливості створення та підвищення цінності бізнесу для підприємства.

Інший погляд на архітектуру підприємства складається з організаційної архітектури, що представляє організаційний дизайн, включаючи організаційних учасників, їхні ролі та відносини в організації. Архітектура процесу важлива для представлення бізнес-процесів. Архітектура програми представляє логіку програми. Він представляє різні компоненти програми та те, як вони пов’язані один з одним. Технічна архітектура представляє собою архітектуру технічної інфраструктури, необхідної для запуску додатків. На рисунку 1.2 представлені різні домени, необхідні для розуміння архітектури підприємства в цілому.

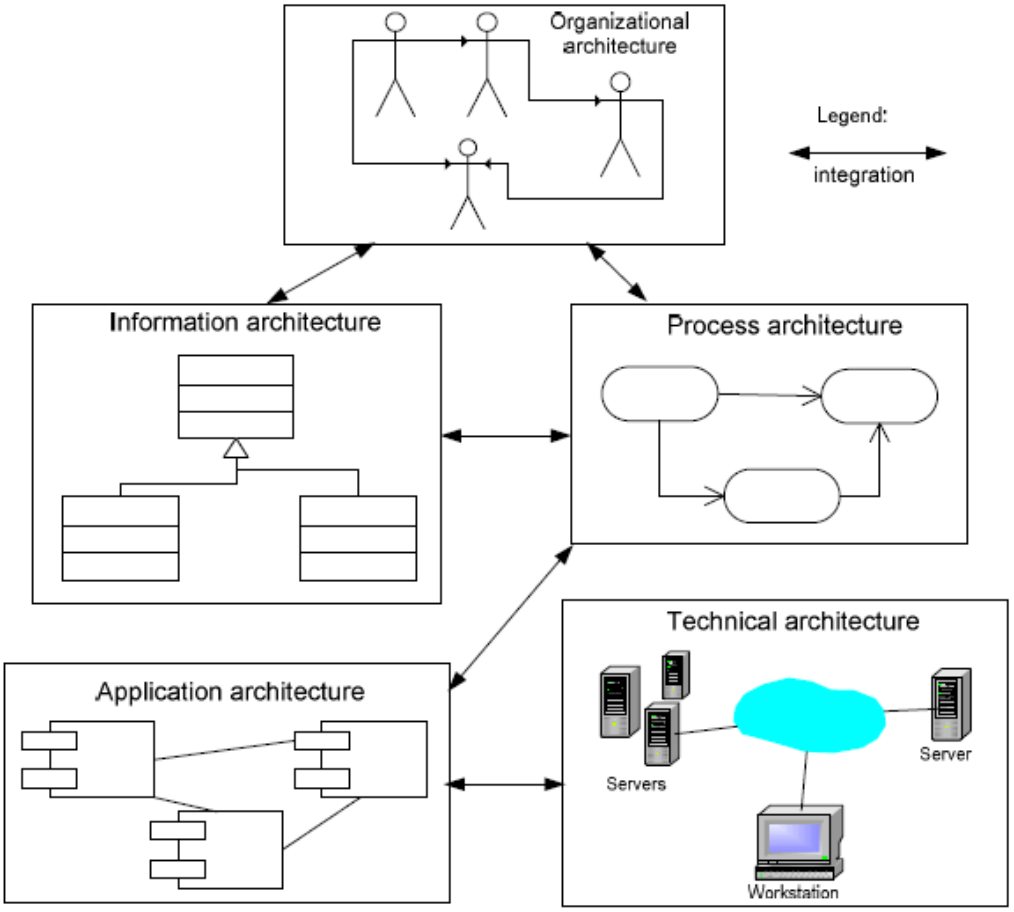


Рисунок 1.2 – Гетерогенні архітектурні домени

Семантична цілісність між різними архітектурними доменами має вирішальне значення для розуміння та чіткого спілкування архітектури підприємства. Щоб отримати інтегроване уявлення про підприємство, необхідно інтегрувати різні виміри для опису архітектури підприємства в цілісному поданні. Інтеграція різнорідних архітектурних областей має вирішальне значення, оскільки інтеграція усуває надмірність, розриви та відмінності всередині підприємства. Досягнення інтеграції є проблематичним через відмінності в методах і моделях архітектурного моделювання і відсутність спільної мови для спільного використання знань. Різні методи та моделі з різних архітектурних доменів ускладнюють інтеграцію особистих поглядів в єдину архітектуру підприємства. Іншим важливим питанням є те, як структурні та поведінкові аспекти пов’язані один з одним, а також як вони пов’язані з організаційною та технічною сферами.

Якість системних специфікацій залежить від досягнення семантичної цілісності графічних зображень. Семантична цілісність — це ступінь зрозумілості графічних зображень, незважаючи на їх складність. Проблеми семантичної цілісності в концептуальних уявленнях створюють труднощі в передачі семантичних деталей домену програми. Неоднозначні, неповні та неузгоджені специфікації ІС роблять перевірку семантичної цілісності між бізнес-процесом і даними особливо проблематичною. Семантичні відношення, що використовуються в концептуальному моделюванні, є одними з ключових конструкцій для фіксації того, як поняття пов’язані одне з одним. Семантична цілісність специфікацій ІС може бути досягнута шляхом усунення суперечностей, неповноти, неоднозначності та надмірностей у процесі моделювання. Семантичну цілісність типу діаграми можна перевірити за допомогою семантичних відносин між елементами моделей, що є критичним для досягнення правильного дизайну системи.

1.3 Архітектура керована моделлю та рівні моделювання

Архітектура, керована моделлю (MDA - Model Driven Architecture) [28] — це підхід до розробки програмних систем, керованої моделлю. Він надає набір вказівок щодо структурування специфікацій, які виражені у вигляді трьох моделей: незалежна від обчислень модель (CIM - Computation Independent Model), незалежна від платформи модель (PIM - Platform Independent Model) і специфічна для платформи модель (PSM - Platform Specific Model). MDA має на меті забезпечити відкритий, нейтральний до постачальників підхід до викликів, пов’язаних зі зміною бізнесу та технологій. Метою цього підходу є розділення функціональних специфікацій системи від питань реалізації на конкретній платформі. Цей каркас містить три рівні абстракції та відображення між ними. Незалежні від платформи моделі будуються з використанням UML та інших пов’язаних стандартів моделювання OMG. CIM охоплює бізнес-аспекти корпоративної системи. Цю

модель іноді називають доменною або бізнес-моделлю, яка описує ситуацію, в якій буде використовуватися ІС. На цьому рівні не розглядається жодна інформація про автоматизовану систему обробки даних. Аналізуючи та моделюючи корпоративні системи, не зосереджуючись на специфікації бажаної функціональності, розробники систем можуть краще керувати складністю розроблених інформаційних систем. Розділення між корпоративними моделями та моделями, пов'язаними з реалізацією, призводить до природного поділу інженерних продуктів ІС на два різних представлення. Цей поділ важливий, тому що розумніше концептуалізувати архітектуру підприємства до того, як буде визначена допоміжна програмна система.

PIM описує роботу системи, але приховує деталі конкретної платформи. PSM поєднує специфікації з рівня PIM і надає специфікацію того, як система використовує певний тип платформи. Важливою особливістю MDA є правила відображення та методи, які використовуються для модифікації однієї моделі в іншу. Тим не менш, питання про те, наскільки можлива автоматизація правил відображення між рівнями CIM і PIM, залишається серйозною дослідницькою діяльністю [31]. Щоб зрозуміти, як і чому компоненти технічної системи є корисними і як вони вписуються в загальну організаційну систему, необхідні принаймні три рівні моделювання специфікацій інформаційної системи; прагматичний рівень, семантичний рівень і синтаксичний рівень. Взаємодія між MDA і трьома рівнями моделювання графічно представлено на рисунку 1.3. На цьому рисунку пояснюється подібність між рівнями моделювання MDA та IS.

Прагматичні, семантичні та синтаксичні рівні представляють великий інтерес у контексті проектування інформаційних систем, оскільки вони мають справу з використанням, значеннями та структурами представлень. Ці три рівні можна розглядати як три виміри розробки вимог: узгодження, представлення та специфікація. Вимір згоди стосується прагматичних аспектів аналізу змін. Вимір представництва стосується семантичних аспектів

системи, представляючи статичні та динамічні аспекти бізнес-процесів через організаційні та технічні межі системи.

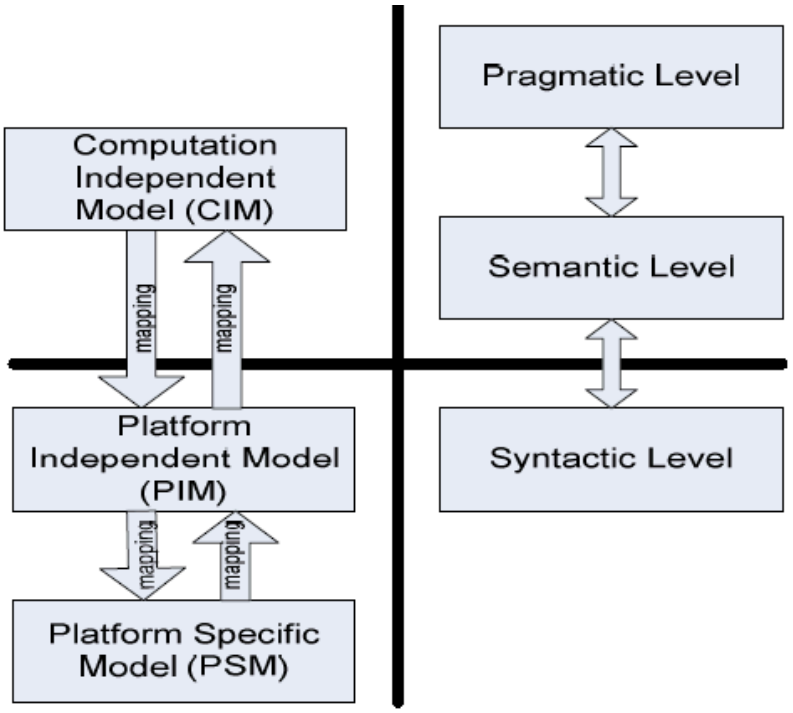


Рисунок 1.3 – Структура MDA і три рівні моделювання

Розмір специфікації можна порівняти з синтаксичним рівнем, на якому визначаються деталі, орієнтовані на реалізацію, які пояснюють потреби обробки даних конкретної програми або програмного компонента [32]. Прагматичний і семантичний рівні також можна порівняти з трьома вимірами для бізнес-моделювання: навмисним, концептуальним і організаційним. Прагматичний рівень можна порівняти з навмисним виміром, який описує стратегічний вимір з точки зору цілей, критичних факторів успіху, сильних і слабких сторін. Він дає відповіді на запитання «чому» щодо бізнесу. Семантичний рівень можна порівняти з виміром поняття та організаційним виміром. Концептуальний вимір описує бізнес-сутності домену та їхні відносини. Це стосується питання «що». Організаційний вимір дає відповідь на питання «як»: він описує учасників, дії та робочі процеси.

Трирівневу структуру, яка представлена на рисунку 1.3, можна використовувати для розуміння різних вимірів аналізу та проектування системи ІС. Чіткий метод моделювання того, як поєднати ці три рівні, має вирішальне значення для розробки зрозумілого та систематичного процесу моделювання. Моделі, що використовуються на кожному рівні, повинні сприяти загальній методології проектування в масштабах підприємства. Така структура забезпечує не тільки спосіб структурування бізнес- і технологічних рішень з точки зору «зверху вниз», але також забезпечує спосіб, за допомогою якого розробка послуг «знизу вгору» може відповідати вимогам бізнесу «зверху вниз».

Розробка, керована моделлю, повинна бути зосереджена на прагматичних і семантичних питаннях ІС. Прагматичний і семантичний рівні стосуються незалежного від обчислень моделювання. На цих рівнях інформаційні системи аналізуються так, як вони сприймаються бізнес-експертами та користувачами, нехтуючи тим, як система буде реалізована. Синтаксичний рівень має бути в основному зосереджений на специфічних для обчислень, але незалежних від платформи питаннях, які необхідні для розуміння того, як будуть реалізовані прагматичні та семантичні моделі.

Розглядаючи весь процес розробки інформаційних систем, представлені три рівні охоплюють як аналіз системи, так і фазу проектування. Двокінцеві стрілки вказують на потребу у фізичній формі та вирівнюванні між рівнями. Усі три рівні взаємопов’язані, оскільки забезпечують основу для розуміння системи підприємства в цілому. Така трирівнева архітектурна структура є основою моделювання, яка допомагає забезпечити взаємодію між потребами бізнесу та технічними рішеннями.

Прагматичний рівень наказує та мотивує специфікації інформаційних систем на семантичному рівні. Специфікації інформаційних систем неможливо зрозуміти, якщо не сформульовано цілі. Аналіз, проведений на семантичному рівні, повинен відповідати цілям зацікавлених сторін,

заявленим на прагматичному рівні. Узгодженість між цими рівнями має вирішальне значення для досягнення узгодженості між специфікаціями ІС.

1.4 Послуги в сервіс-орієнтованому аналізі та проектуванні

Пропонований метод базується на онтологічній перспективі поняття послуги. Розуміння послуги базується на аналізі самої природи поняття. Онтологічний аналіз і розуміння послуги також сприяють кращому розумінню організації або організаційної семіотики [33]. Такий аналіз визначає основні елементи, що складають концепцію, і пояснює, чому і як ці елементи пов’язані. Онтологічна основа сервісу не має прямого відношення до технологічних рішень. Завдання сервісу — надати опис того, що відбувається, без упередженості до питань реалізації. Це не «послуга», яка надається і має цінність для запитувача послуги, а цінність вмісту. Цей зміст не може бути досягнутий без динамічного процесу, без взаємодії між деякими акторами, результат якої створює цінність для акторів.

Зі згаданих вище визначень послуги можна стверджувати, що послуга – це, перш за все, динамічний акт виконання чогось комусь. Це означає, що є більше елементів, необхідних для побудови концепції послуги, ніж просто процес «роблення». Оскільки в процесі «виконання» завжди бере участь кілька акторів, це означає, що це комунікаційний акт або взаємодія між принаймні двома суб’єктами, людськими, організаційними чи технічними компонентами: один суб’єкт, який запитує про послугу, і той, хто надає це. Якщо хтось щось робить, це означає, що відбувається якась дія. Дія завжди є цілеспрямованою чи цілеспрямованою та передбачає обов’язки залучених учасників. Дія, будучи цілеспрямованою, завжди повинна мати певну цінність для актора; будь то фізичний, інформаційний чи емоційний.

З цього опису можна зробити висновок, що послуга є складним поняттям, яке має структуру. Елементи цієї структури діють разом для отримання бажаного результату. Щоб отримати результат, який забезпечує

певну цінність на вимогу, необхідні чотири ключові елементи: запитувач служби, запит на послугу, постачальник послуг і відповідь на послугу. Без одного з цих елементів поняття послуги втрачає семантичне значення. З точки зору аналізу ІС послугу можна розглядати як функцію, яка визначається кількістю потоків у протилежних напрямках між запитувачем послуг і постачальником послуг. Кожна відповідь служби (вихід) є функцією запиту служби (вхід). Постачальники послуг — це актори, які отримують запити на обслуговування та перетворюють їх у відповіді, які надсилаються запитувачам послуг. Це цикл взаємодії між запитувачем послуг і постачальником послуг. Ця ідея графічно представлена на рисунку 1.4.

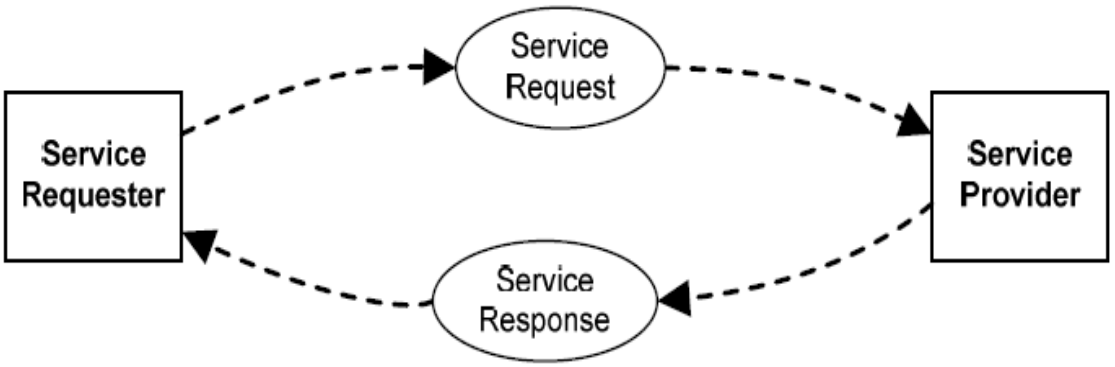


Рисунок 1.4 – Сервіс як цикл взаємодії

Послугу в сервісно-орієнтованому аналізі та проектуванні можна визначити як набір упорядкованих і цілеспрямованих взаємодій між різними учасниками підприємства. Взаємодії послуг можна концептуалізувати як різні типи дій, які відбуваються між суб'єктами підприємства. Архітектура сервісу — це композиція різних типів суб'єктів підприємства та набір циклів взаємодії між ними. Актори, яких можна розглядати як зовнішні сутності або активні елементи є організаційними та технічними підсистемами.

Основними характеристиками, важливими для розробки методу сервіс-орієнтованого моделювання, є:

- Підприємство є системою. Система підприємства — це композиція підсистем, які можна розглядати як взаємодіючі компоненти [34]. Кожна підсистема може грати роль запитувача послуг і постачальників послуг.
- Коли підсистеми взаємодіють, вони викликають зміни в об’єктах, які проявляються властивостями.
- Організаційно-технічні компоненти представлені слабозв’язаними підсистемами, які функціонально не перетинаються. Підсистеми можуть бути декомпозовані та пов’язані взаємодією між службами. Слабкий зв'язок є одним із основних принципів орієнтації на обслуговування.

Будь-який цикл взаємодії між акторами має бути мотивований результуючим потоком цінностей. Таким чином, систему підприємства можна визначити за допомогою архітектури сервісу, яка представлена як набір слабко пов’язаних взаємодіючих підсистем. Підсистеми можна розглядати як технічні або організаційні компоненти. Організаційними компонентами можуть бути особи, компанії, підрозділи або ролі, які позначають групи людей. Технічні компоненти - це підсистеми, які можуть бути представлені програмним і апаратним забезпеченням.

Висновки до розділу 1

В даному розділі подано підхід і методи дослідження, що представляє підхід до сервіс-орієнтованого проектування програмних рішень розробки. Також представлені пов’язані дослідження, концепції та інформація, що представляє важливі підходи та їх передумови, які мають найбільше значення для дослідження. В розділі описується концепція сервісу та характерні риси сервіс-орієнтованого аналізу та проектування і наведені характеристики для сервіс-орієнтованих представлень.

РОЗДІЛ 2. СТРУКТУРИЗАЦІЯ ЗАСОБІВ СЕРВІС-ОРІЄНТОВАНОГО ПРОЕКТУВАННЯ ТА МОДЕЛЮВАННЯ ПРОГРАМНИХ РІШЕНЬ

2.1 Опис процесу сервіс-орієнтованого моделювання

Процес моделювання є основою представленого сервіс-орієнтованого методу моделювання. Це розроблена та визначена послідовність дій та організаційних процедур, які покращать співпрацю в команді розробників і сприятимуть постачанню якісного продукту замовнику. Він включає в себе дії та техніки моделювання через три рівні абстракції та визначає інтегрований порядок поведінки протягом усього процесу моделювання. Процес моделювання сприяє узгодженому переходу від одного рівня до іншого. Кожен процес є цілеспрямованим і має початковий і кінцевий етапи.

Системні аналітики в області проектування підприємства спочатку визначають дуже загальний і неоднозначний бізнес-процес на прагматичному рівні. Цей сценарій схожий на процес концепції системи [36], метою якого є відкласти деталі та зрозуміти загальну картину майбутньої системи. Пізніше розробники систем поступово розширюють представлення аналізу, розміщуючи численні припущення в чітко визначені представлення інформаційної системи на семантичному рівні. Щоб досягти консенсусу, аналітики об’єднують різні частини специфікацій ІС і відтворюють сценарії з об’єктами різних типів, щоб перевірити та обґрунтувати компоненти технічної системи на синтаксичному рівні. Низхідний аналіз сприяє кращій семантичній якості проектування підприємства [37]. Процес моделювання включає набір правил, які визначають, як загалом має здійснюватися процес. Дизайн і архітектура системи втілюють найважливіші рішення про те, як система побудована та реалізована.

Успіх процесу моделювання залежить від двох речей: від мови моделювання, яка використовується для формування моделей, і від процесу

використання цих моделей. Моделюючі конструкції повинні бути передані; тому мова моделювання має бути достатньо виразною, щоб передати задум. Він має бути однозначним і простим у використанні, щоб полегшити ідентифікацію протиріч, які викликають комунікативні проблеми. Зазвичай графічна мова використовується на основі діаграм, оскільки природна мова є надто неоднозначною та часто викликає непорозуміння. Недостатньо розробити мову моделювання для графічних зображень. Весь процес моделювання має бути забезпечений відповідним способом моделювання, який керує процесом моделювання систематично, що сприяє кращій якості спілкування між зацікавленими сторонами.

Запропонований у роботі процес сервіс-орієнтованого моделювання має наступні переваги:

- він керує послідовністю дій через три рівні абстракції та забезпечує спосіб переходу від прагматичних до семантичних уявлень системи, а також відображення від обчислювально-нейтрального моделювання до проектування, специфічного для реалізації;
- сервіс-орієнтований аналіз допомагає графічно представити взаємодію між інтерактивними, поведінковими та структурними аспектами специфікацій ІС;
- семантичні залежності сервіс-орієнтованих діаграм допомагають виявити небажані характеристики, такі як неузгодженість, неповнота, надмірність і неоднозначність моделювання, що залежить від реалізації.

Успіх сервіс-орієнтованого аналізу та проектування значною мірою залежить від пошуку відповідної відповідності між трьома рівнями корпоративної структури, яка представлена на рисунку 2.1.

Відповідність специфікацій системи між цими рівнями має вирішальне значення для успіху кінцевого результату. Відповідність і узгодженість між рівнями значною мірою залежить від наявності технік уточнення прагматичних сутностей (цілі, проблеми та можливості), які представляють їхні структурні та динамічні аспекти на семантичному рівні. Методи повинні

надавати керівництво для підтримки розробки та переходу від початкових вимог до проектування системи. Концептуальні моделі потребують значних зусиль, щоб відобразити артефакти програмного забезпечення.

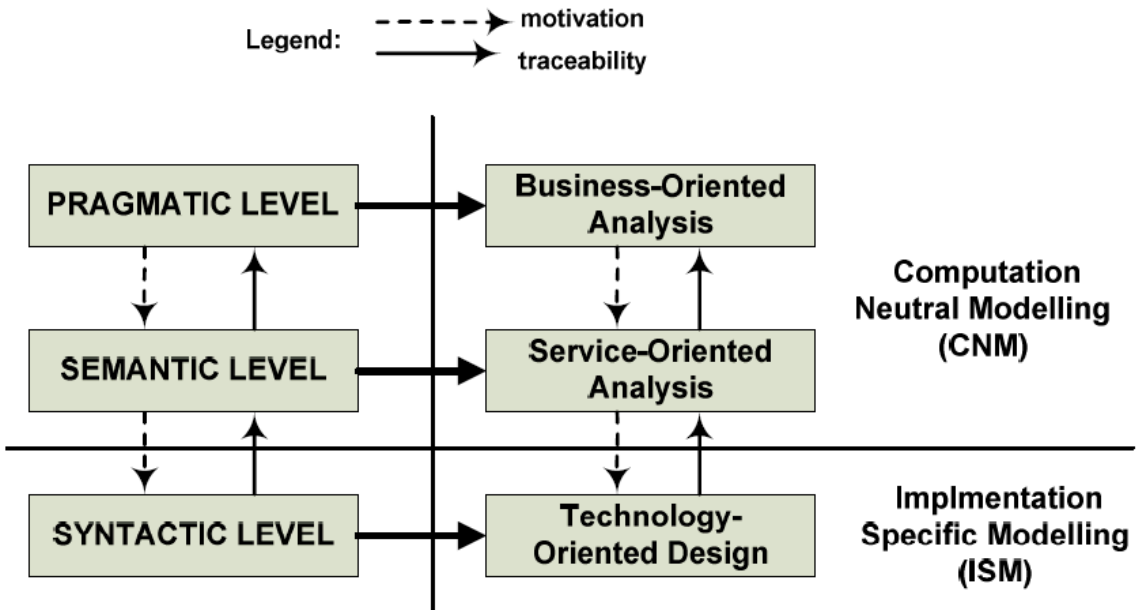


Рисунок 2.1 – Архітектурна основа для сервіс-орієнтованого моделювання

Інтегрований спосіб моделювання забезпечує вказівки для переходу між рівнями. Він також надає спосіб відображення від обчислювально-нейтрального моделювання до конкретного впровадження. Можливість переходу дозволяє перевіряти узгодженість між рівнями. Це також підкреслює можливості для вимог простежуваності, яка є важливою частиною для управління змінами [37], верифікації та підтвердження. Верифікація стосується процесу забезпечення відповідності проекту або впровадження вимогам. Валідація відноситься до діяльності, яка відповідає вимогам, узгодженим на етапі аналізу і забезпечує її якість. Це відноситься до діяльності, яка дає змогу розробникам системи перевірити, чи справді аналіз і дизайн задовольняють критерію, встановленому зовнішнім користувачем служби. Семантичні залежності, які використовуються в процесі

моделювання, дають можливість контролювати якість семантичного моделювання. Він надає нові способи контролю семантичної неповноти.

Архітектурна структура методу сервіс-орієнтованого моделювання схожа на структуру MDA, оскільки вона також забезпечує нейтральне до обчислень і залежне від реалізації моделювання, яке відповідає незалежної від обчислень моделі MDA та незалежної від платформи моделі. Різниця полягає в призначенні фреймворку, побудованого на мові та техніці моделювання. Метою інфраструктури MDA є підтримка керованої моделлю розробки програмних систем. Ця структура використовує UML та інші стандарти моделювання OMG. Метод сервіс-орієнтованого моделювання розроблено для аналізу та проектування інформаційних систем з акцентом на концептуальне моделювання специфікацій ІС. Метод сервіс-орієнтованого моделювання базується на мові моделювання та наборі методів моделювання для специфікацій ранніх вимог, де прагматичні та семантичні аспекти архітектур сервісу можна аналізувати разом. Він використовує мову, яка базується на принципах взаємодії та використовує єдиний тип діаграми для представлення структурних і поведінкових аспектів. Крім того, метод забезпечує процес моделювання, який підтримує відстежуваний спосіб уточнення прагматичних специфікацій у концептуальні представлення ІС. Він також надає принципи переходу від моделювання, що не залежить від обчислень, до моделювання, що залежить від реалізації.

У цьому дослідженні основна увага приділяється нейтральним до обчислень представленням корпоративних архітектур [38], які можуть використовувати нетехніки. Моделі на рівні обчислювально-нейтрального моделювання відіграють ключову роль для інтеграції семантичної системи на етапі бізнес-орієнтованого (підтримує вигляд планувальника) та сервіс-орієнтованого аналізу (підтримує перегляд власника). Нейтральні до обчислень моделі визначають систему з точки зору, незалежної від обчислень, що необхідно для подолання розриву між бізнес-аналітиками та експертами з проектування систем. Вони повинні бути встановлені до прийняття будь-яких

рішень щодо реалізації. Визнається, що підтримка UML для такого завдання досить розпливчаста [39], тому що досі відсутні семантичні принципи інтеграції різних типів діаграм.

У методі сервіс-орієнтованого моделювання моделювання, виконане на прагматичному та семантичному рівнях, є нейтральним щодо обчислень. Семантичні та прагматичні моделі визначають лише істотні аспекти архітектури сервісу та побудовані на інтегрованому наборі семантичних залежностей. Семантичний рівень керується прагматичними принципами, які розміщені на вищому рівні абстракції. Він визначає обмеження для синтаксичного рівня, а також надає рішення для прагматичного рівня. Узгодженість між семантичним і синтаксичним рівнями є вирішальними для успіху кінцевого реалізованого продукту. Моделі, створені на кожному рівні, необхідні для виявлення невідповідностей процесу аналізу. Вони забезпечують виявлення невідповідностей на ранній стадії аналізу, що сприяє можливості своєчасного реагування. Узгодженість моделей, що використовуються на різних рівнях, необхідна для прийняття обґрунтованих бізнес-рішень і створення цінності для підприємства [40]. Впроваджений продукт повинен узгоджуватися з цілями, заявленими бізнес-аналітиками, і це залежить від досягнення узгодженості між нейтральним обчисленням і моделюванням, пов’язаним із реалізацією.

Сервісно-орієнтовані діаграми забезпечують основу для логічного проектування на синтаксичному рівні, рівні, де враховуються деталі реалізації. Узгодження концептуальних уявлень із технологічно-орієнтованим дизайном є необхідним, оскільки кожна ціль має узгоджуватися з кінцевим реалізованим продуктом. Спеціальне для реалізації моделювання можна продемонструвати за допомогою UML. Мотивацією до цього є мій досвід викладання курсів об’єктно-орієнтованого аналізу та дизайну. Робота зі студентами, які використовують об’єктно-орієнтований підхід і UML як мову моделювання у своїй практичній роботі, і побачивши певні труднощі, додали мені впевненості в моєму дослідженні. Використання різних типів діаграм

для представлення різних аспектів ІБ створює труднощі в досягненні узгодженості та простежуваності між діаграмами.

Сервісно-орієнтоване моделювання враховує подвійну природу аналізу ІС та процесу проектування: між різними аспектами ЕА та різними поглядами зацікавлених сторін. Такий процес моделювання є особливо гнучким для впровадження еволюційних змін специфікацій ІС. Цей інтегрований і систематичний спосіб сервіс-орієнтованого аналізу сприяє і забезпечує керівні принципи від цільового моделювання до специфікацій, орієнтованих на сервіси, і до проектування, орієнтованого на реалізацію. Перевага такого процесу полягає в тому, що він може допомогти розробникам ІС зосередитися на нейтральних до обчислень аспектах системи ІС, а не лише на технологічних рішеннях. Цей метод побудовано на сервісно-орієнтованому процесі моделювання, який є прагматичним і нейтральним до обчислень. Метою такого методу моделювання є забезпечення чіткої системи принципів декомпозиції, а також принципів поділу наскрізних проблем. Ці принципи є критично важливими для управління складністю ІС та проблемами інтеграції, а також для мінімізації неузгодженостей і розривів бізнес-процесів, які можуть перетинати межі підприємства, коли відбуваються деякі зміни [41].

2.1.1. Прагматичні залежності

Бізнес-орієнтований аналіз, проведений на прагматичному рівні, забезпечує мотивацію нових бізнес-рішень. Бізнес-стратегія організації визначає напрямок і масштаби можливих змін в організації та визначає позицію організації, що є важливим для конкурентної переваги [42]. Це перший і найбільш абстрактний крок, на якому формулюються та обговорюються цілі. Стратегічний і оперативний вибір, прийнятий на цьому рівні, є ключовим у визначенні того, як буде підходити до зміни робочого процесу. Цей рівень зосереджується на стратегічному описі, який має дати

визначення «чому» - довгостроковий намір або бачення підприємства, що розвивається.

Будь-який бізнес-процес можна розглядати як набір послуг. З прагматичної точки зору функціональність сервісу можна розглядати як проблему, можливість або ціль. Функціональність сервісу може бути бажаною або небажаною. Прагматичні аспекти обумовлені цілями, проблемами та можливостями розробників інформаційних систем. Аналіз послуг може допомогти бізнес-експертам проаналізувати можливі зміни для створення додаткової вартості бізнесу. На цьому рівні послуга характеризується директивним способом [42], де встановлюються «правила гри». Тут аналізуються типи дій, що входять до складу послуги, актори та їхня відповідальність. Обов'язки зазвичай розподіляються відповідно до конкретних структурних і поведінкових обмежень в організації.

Прагматичні залежності використовуються для визначення намірів акторів, залучених до бізнес-процесів. Їх також можна розглядати як основу моделювання, щоб міркувати про наміри дизайнерів щодо нової архітектури системи. Графічне позначення прагматичних залежностей представлено на рисунку 2.2.

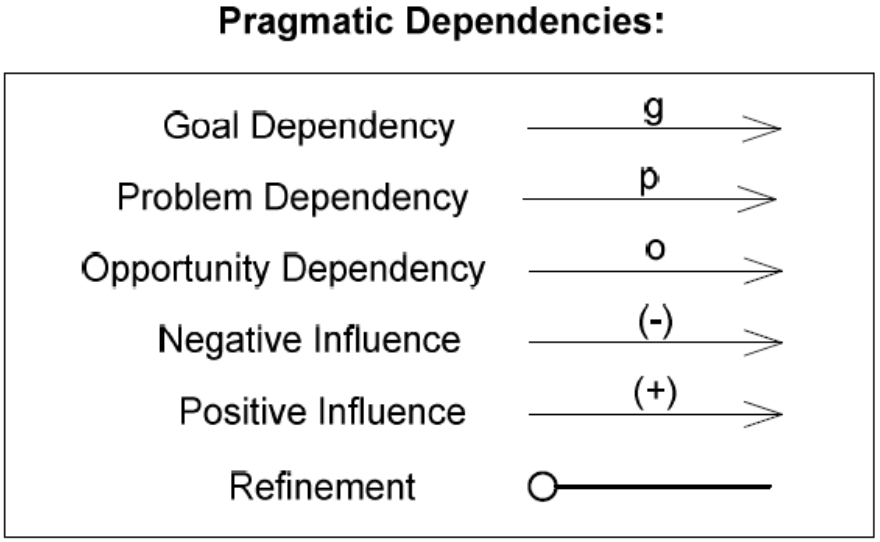


Рисунок 2.2 – Легенда прагматичних залежностей

Цільова залежність використовується для визначення бажаних ситуацій або бажаних структурних властивостей у системі. Він пов’язує суб’єктів підприємства з такими навмисними поняттями, як цілі чи завдання.

Проблемна залежність використовується для визначення небажаних ситуацій. Обмежувальні поняття, такі як проблеми або причини проблем, виражають або небажані властивості системи, або небажані ситуації, яких учасники підприємства мають намір уникати. Проблема зазвичай заважає досягненню мети. Залежність від можливості можна використовувати для позначення майбутніх ситуацій, якими можна скористатися для покращення поточної ситуації. Можливості також можна розглядати як переваги існуючої системи або бізнес-процесу. Можливість - це бажаний фрагмент існуючої семантичної специфікації, який призначений для підтримки в новій системі. Бажані особливості проблемної ситуації можна вказати як можливості за допомогою уточнюючих посилань.

Залежність від негативного впливу (-) і залежність від позитивного впливу (+) використовуються для позначення впливу серед цілей, проблем і можливостей. Залежність негативного впливу від A до B вказує на те, що A можна розглядати як проблему, оскільки вона перешкоджає досягненню цілі B. Позитивна залежність впливу від A до B означатиме, що A можна розглядати як можливість для досягнення цілі B [43].

Залежність уточнення використовується як засіб прагматичної декомпозиції цілей, проблем і можливостей. Прагматичні сутності зазвичай уточнюються шляхом посилання на інші більш конкретні ситуації.

2.1.2. Процес декомпозиції цілей

Цілі різних організаційних компонентів стимулюють взаємодію між акторами, що призводить до подальших взаємодій [43]. Аналіз цілей має суттєві перспективи в допомозі у виявленні та розробці вимог. У той же час аналіз цілей забезпечує контроль зобов’язань, які різні суб’єкти мають

виконати. Будь-який актор може досягти мети, уникаючи проблеми. Аналіз цілей є корисним інструментом для вирішення різного роду конфліктів між прагматичними суб'єктами, оскільки досягнення однієї мети може заважати досягненню інших цілей. Слід зазначити, що тлумачення вимоги як проблеми, можливості чи цілі є відносним. Досягнення мети одним актором може розглядатися як проблема для іншого актора. Ціль, можливість і залежність проблеми можуть використовуватися для позначення бажаних або небажаних ситуацій. Цілі або проблеми можна розкласти за допомогою посилання на уточнення.

На рисунку 2.2 представлено приклад декомпозиції цілей.

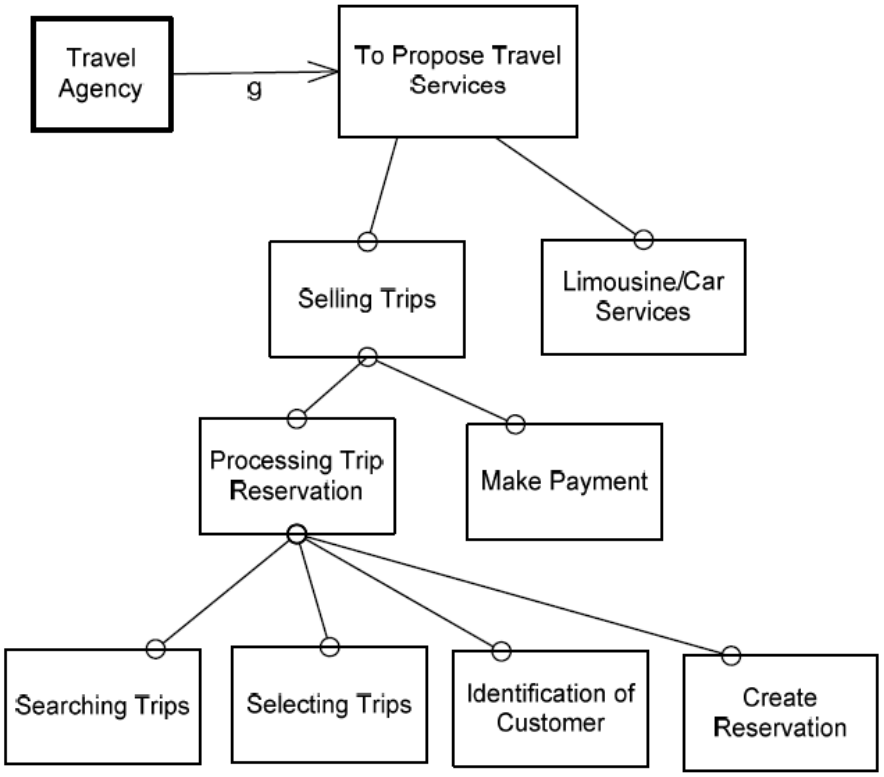


Рисунок 2.2 – Приклад декомпозиції цілей

У наведеному вище прикладі ціль туристичної агенції «Пропонувати туристичні послуги» уточнюється на дві підцілі «Продаж поїздок» і «Послуги лімузинів/автомобілів» (неповна декомпозиція). Підціль «Продаж подорожей» розкладається на дві підцілі «Обробка бронювання поїздки» та

«Здійснення оплати» (повна декомпозиція). Підціль «Обробка бронювання поїздки» далі розкладається на чотири підцілі (повна декомпозиція), необхідні для досягнення підцілі «Обробка бронювання поїздки».

Впливи між прагматичними суб'єктами можна аналізувати за допомогою негативних або позитивних залежностей впливу. Проблеми зазвичай негативно впливають на досягнення цілей. Один із способів усунути цю проблемну ситуацію - послабити проблему шляхом введення нової можливості, яка негативно впливає на проблему та позитивно впливає на ціль. Ця ситуація представлена на рисунку 2.3.

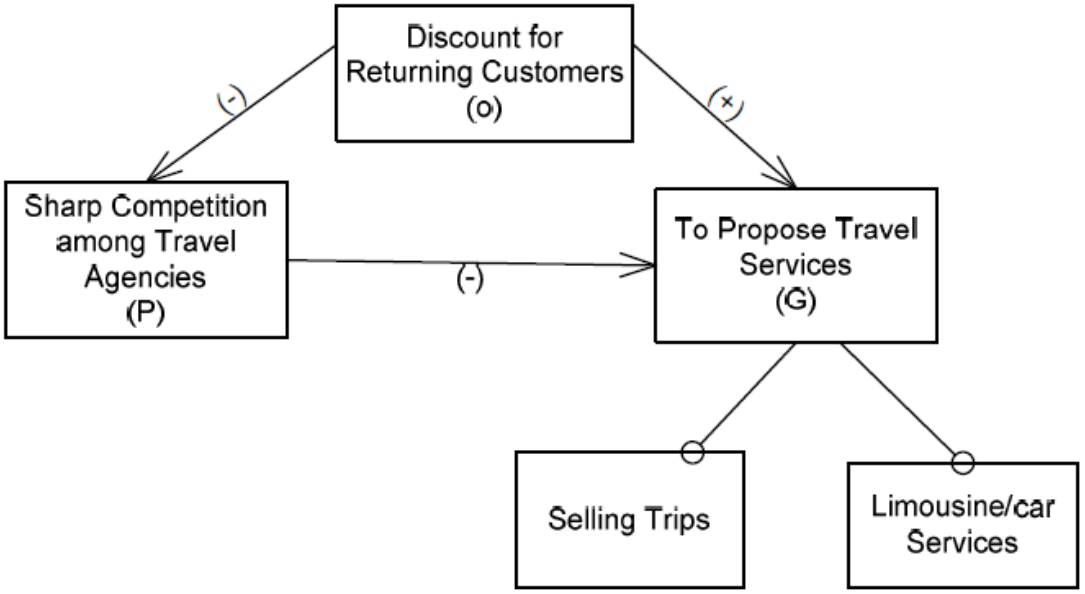


Рисунок 2.3 – Введення нової можливості

Проблема «Гостра конкуренція між туристичними агентствами» негативно впливає на мету «Пропонувати туристичні послуги». Нова запроваджена можливість «Знижка для постійних клієнтів» має негативно вплинути на існуючу проблему та позитивно вплинути на мету «Пропонувати туристичні послуги».

Прагматичні специфікації мають на меті забезпечити мотивацію для концептуальних представлень компонентів підприємства, які визначаються в термінах взаємодії між запитувачами послуг і постачальниками послуг на

семантичному рівні. Аналіз цілей має вирішальне значення для кінцевого результату, оскільки основне припущення полягає в тому, що системні послуги варті уваги, якщо вони відповідають цілям організації. Прагматичні специфікації можуть бути використані як рушійна сила для аналізу семантичної структури представлень сервісу.

2.2 Семантична структура програмного рішення

Семантичний (концептуальний) рівень повинен мати здатність описувати статичні та динамічні аспекти бізнес-процесів за межами організаційної та технічної системи [44]. Моделі, визначені на цьому рівні, показують діяльність, яку виконують різні актори. Їх діяльність, як правило, визначається цілями. Семантичний рівень може забезпечувати опис взаємодії сервісу з класами попередньої та післяумови. Він визначає семантику використовуваних понять і відносин, які визначають основний зміст бізнес-процесу (послуги або складу послуги). Моделювання на цьому рівні також повинно мати здатність показувати процес прийняття рішень, що призводить до вибору курсу дій з кількох альтернатив. Якщо вказана загальна поведінка системи, також можна виразити альтернативи, послідовності та дії синхронізації.

Семантичні специфікації не можна аналізувати окремо від прагматичних специфікацій. Цей спосіб моделювання забезпечує основу для перевірки узгодженості між рівнями. На нижчому рівні абстракції необхідно уточнити прагматичні сутності в сервіси, які можна розглядати як основи сервіс-орієнтований спосіб моделювання. Після формулювання цілей та аналізу можливостей і рішень на прагматичному рівні важливо показати, як ці цілі уточнюються далі на семантичному рівні. Уточнення прагматичної сутності (підцілі) на цикли взаємодії представлено на рисунку 2.4.

Цей рисунок представляє лише одне уточнення підцілі та лише один цикл взаємодії. Тут не показано, як ця підціль інтегрована в повний опис

послуги «Пошукові поїздки». Усі чотири проміжні цілі, наведені в прикладі, є підцілями цілі «Продаж поїздок», і всі вони повинні бути уточнені в послуги та далі інтегровані в єдину діаграму, що представляє весь бізнес-процес послуги «Продаж поїздок», включаючи статичний і динамічний аспекти.

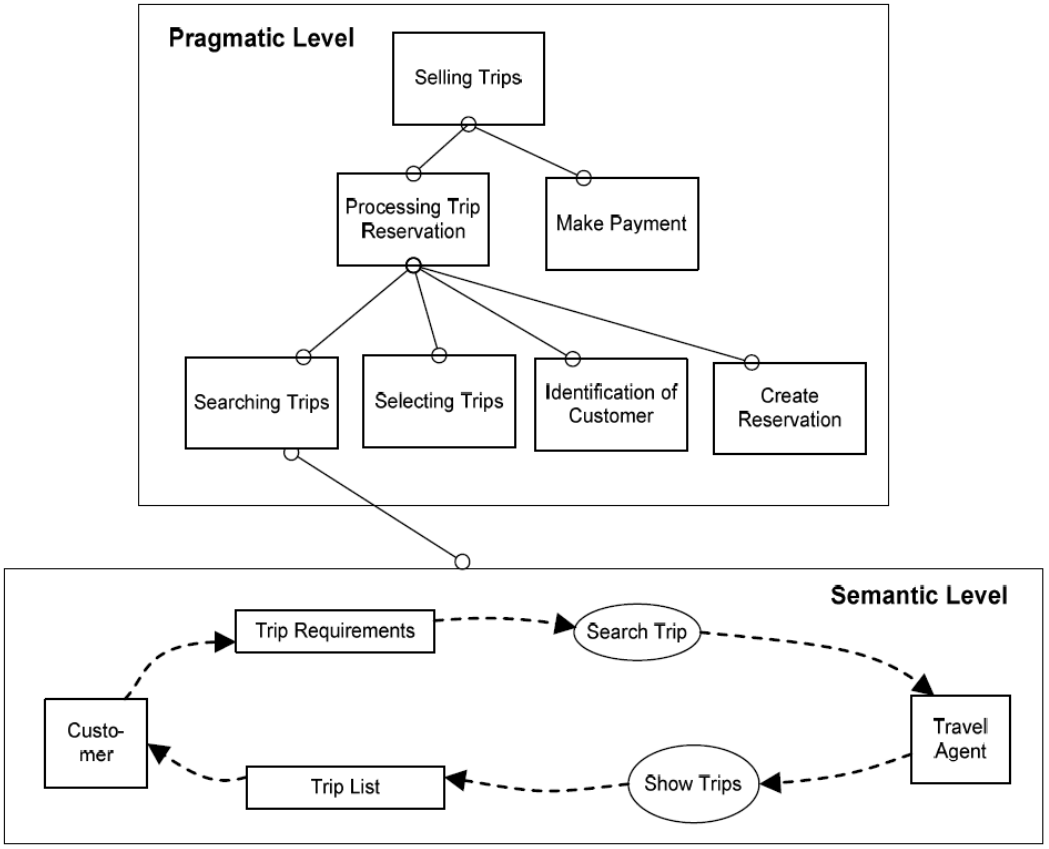


Рисунок 2.4 – Уточнення прагматичних сутностей

Існує два основних класи семантичних залежностей, які використовуються на семантичному рівні: статичні та динамічні. У моделюванні підприємства можна визначити всі види традиційних статичних відносин між такими поняттями, як класи, актори, стани та потоки [45]. Семантика статичних залежностей може бути визначена як обмеження кількості.

З точки зору розробника програмного забезпечення, недостатньо представити архітектуру системи з прагматичної та семантичної точки зору. Розробники систем повинні розуміти технічну архітектуру системи, де буде

встановлено додаток. Мова моделювання повинна мати можливість графічно відображати узгодження між організаційними та технічними межами системи. Синтаксичний рівень повинен визначати специфічні для реалізації деталі, які пояснюють потреби обробки даних конкретного додатка або програмного компонента. Впровадження розуміється як призначення технологічних засобів елементам у моделі реалізації, щоб система могла працювати [47]. Актори підприємства, які на семантичному рівні представлені квадратними прямокутниками на синтаксичному рівні, можна розглядати як людей або технічні компоненти. Архітектура корпоративної системи може бути розроблена як декомпозиція слабко пов'язаних файлів, програмного забезпечення або апаратних компонентів. На рисунку 2.5 представлено позначення компонентів, які можуть замінити акторів, що використовуються в різних проектах.

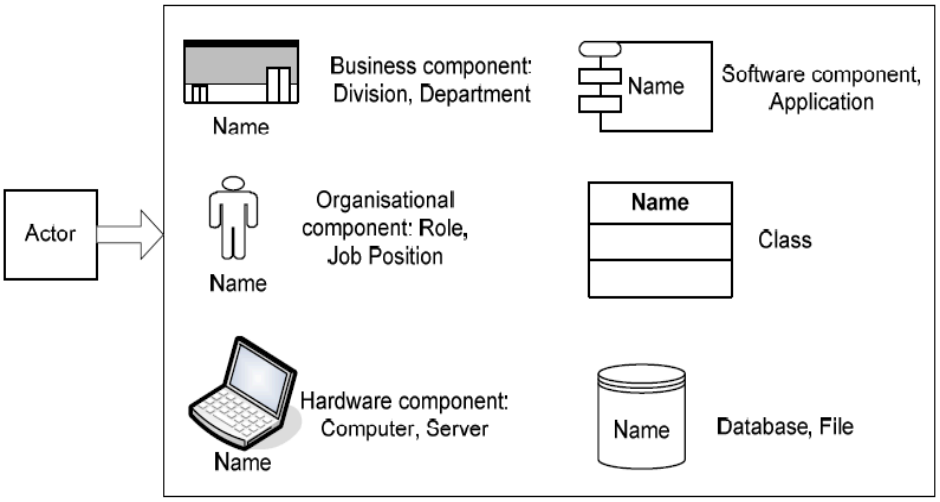


Рисунок 2.5 – Позначення компонентів

Акторів можна розкласти на організаційну та технічну складові. Організаційні підсистеми можуть бути представлені як люди або бізнес-компоненти. Класи та файли даних можуть представляти бази даних, сховища даних або інші типи зберігання даних. Як правило, узгоджений набір взаємодій делегується одному незалежному компоненту. Взаємодії служби

можуть бути визначені в термінах компонентів та їх інтерфейсів із структурним визначенням макетів роздруківок, повідомлень, екранів або файлів. Вони мають визначати існуючі або очікувані потоки взаємодії для підтримки намірів різних акторів [49]. Форми екрану та роздруківки повинні бути налаштовані для кожного потоку взаємодії між організаційними та програмними компонентами. Форми повідомлень призначені для реалізації потоків взаємодії між програмними або апаратними компонентами. Позначення для інтерфейсів між технічними та організаційними компонентами представлено на рисунку 2.6.

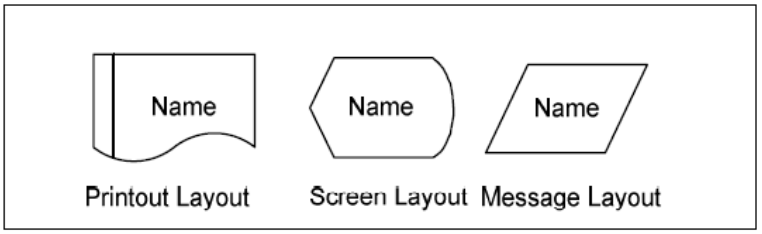


Рисунок 2.6 – Позначення інтерфейсів

Сервісний цикл можна визначити з точки зору технічних компонентів та організаційних компонентів та інтерфейсів між ними. Коротка ілюстрація представлена на рисунку 2.7.

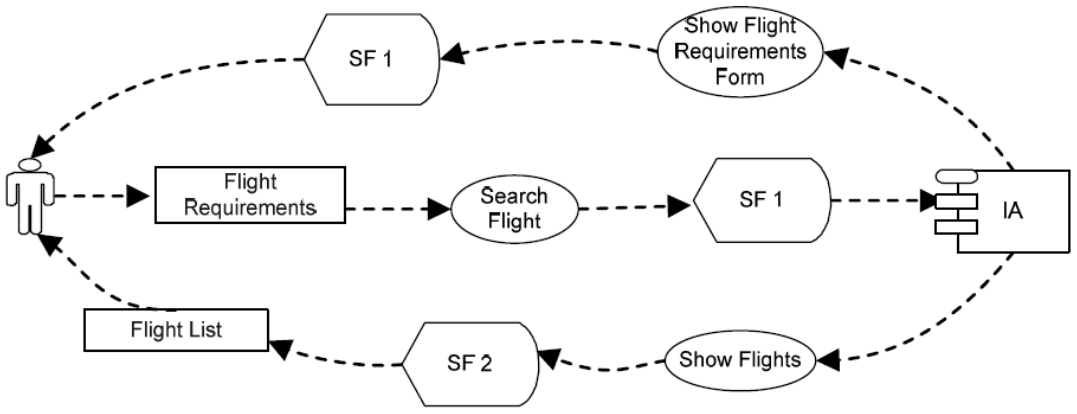


Рисунок 2.7 – Інтерфейси між технічними та організаційними компонентами

SF1 — це екранна форма вимог до польотів, а SF2 — це екранна форма списку польотів. Графік чітко визначає інфраструктуру компонента Інтернет-агента (ІА). Він показує інтерфейс користувача з програмним компонентом. Коли користувач починає процес, програмний компонент ІА викликає дію Show Flight Requirements Form, і користувачеві відображається екранна форма SF1. Коли користувач заповнює вимоги до польоту та натискає кнопку «Пошук польоту», запускається операція «Пошук польоту» і компонент ІА шукає польоти. Якщо відповідні рейси знайдено, ІА запускає операцію «Показати рейси», і користувачеві відображається екранна форма «Список рейсів» SF2.

Усі узгоджені взаємодії, які поєднуються разом для досягнення мети запитувача послуг, можуть бути реалізовані як автономний програмно-апаратний компонент. Таким чином, ідентифікація потоків взаємодії має вирішальне значення для розбиття корпоративної системи на узгоджені служби, що не перетинаються. Кожне підприємство може визначити власний набір символів, які використовуються для представлення різних типів компонентів. Синтаксичні елементи розглядаються як залежність від перспективи реалізації або інструменту CASE. Наприклад, відносини бази даних є типовими представниками синтаксичного рівня. Макети файлів або баз даних можна визначити за допомогою звичайних мов програмування або традиційних мов визначення баз даних. Майже той самий набір синтаксичних елементів можна використовувати для об'єктно-орієнтованого підходу. У цьому випадку поняття макета файлу слід замінити поняттям макета класу, який також міститиме операції. Узагальнення та агрегація ієрархій акторів з точки зору технічних компонентів, які використовуються в різних комунікаційних петлях, визначатиме відповідну архітектуру апаратної чи програмної системи. Інші типи піктограм, як-от факсимільні апарати, телефони та комп’ютерні мережі, можуть бути введені на вимогу. Синтаксичні елементи можна розглядати як будівельні блоки для представлення технічної архітектури розгортання, інфраструктури

програмних компонентів або організаційної структури залежності [50]. Залежний від реалізації файл, повідомлення, роздруківка або макет екрана мають визначати технічну архітектуру. Усі три рівні фреймворку взаємопов’язані, оскільки вони визначають один і той самий артефакт. Діаграми на синтаксичному рівні обмежені залежностями на семантичному та прагматичному рівнях. Процес моделювання та проектування на трьох архітектурних рівнях забезпечує прийоми та вказівки для переходу від одного рівня до іншого. Це відкриває можливості для перевірки цілей, сформульованих на прагматичному рівні, і перевірки логічного дизайну, зробленого на синтаксичному рівні, до концептуального моделювання, виконаного на семантичному рівні. Він надає можливість семантичної відстежуваності на всіх трьох рівнях і забезпечує взаємодію між бізнес-вимогами та технічними рішеннями, що забезпечує природне уявлення для розуміння артефакту моделювання в цілому.

2.3 Сутність сервісно-орієнтованої мови моделювання

У цьому розділі представлено основу мови моделювання, що використовується для методу сервіс-орієнтованого моделювання, і пояснює ідею гнучкої інтерпретації концепцій і представляє нотацію, яка використовується для сервіс-орієнтованого аналізу та проектування. Він представляє мета-модель, яка визначає елементи мови моделювання. Елементи — це актори, концепції, дії та залежності, які є як статичними, так і динамічними. Метод моделювання та мова моделювання є двома важливими внесками цієї роботи. Метод — це явний спосіб моделювання, що складається з нотації мови, методів моделювання та вказівок. Це важливо, тому що воно визначає набір дій моделювання, розповідаючи, що робити і чому. Для побудови сервіс-орієнтованих діаграм потрібна мова моделювання. Елементи моделювання та позначення є важливими методами роботи для зацікавлених сторін, залучених до процесу розробки інформаційної системи.

Вони служать основою для розуміння та управління бізнес-процесами та узгодження бізнесу з архітектурою інформаційних технологій. Нотація моделювання забезпечує основу для визначення внутрішньої та зовнішньої поведінки сервісу графічним способом.

2.3.1. Інтерпретація понять моделювання

Інтерпретація концепцій моделювання для представлення проблемної області є дуже важливою, оскільки концепції визначають семантичну коректність специфікації вимоги. Процес специфікації вимог стосується моделювання вимог зацікавлених сторін, визначених на етапі визначення вимог. Зміст формулювань вимоги відіграє важливу роль для вираження статичних і динамічних зв'язків між поняттями в проблемній області. Аналіз концептів є важливим, оскільки вони є об'єктами та носіями інформації проблемної області, які становлять головний інтерес. Інтерпретація Концепції, які використовуються в сервіс-орієнтованому моделюванні, є відносними. Мотивація важливості концепції відносності полягає в тому, щоб уникнути структурно різних, але семантично еквівалентних репрезентацій. Суворе тлумачення ролей є одним з головних джерел труднощів у сфері інтеграції зору. Семантично еквівалентні, але структурно різні репрезентації порушують принцип незалежності - головний принцип концептуалізації. Рисунок 2.8 демонструє, що те саме поняття можна інтерпретувати багатьма способами.

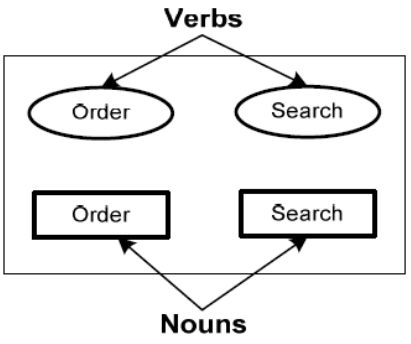


Рисунок 2.8 – Відносна інтерпретація понять

Будь-який концепт можна розглядати як актора, атрибут або клас, залежно від комбінації семантичних відносин, які пов’язують його з іншими концептами. Те, чи розглядається концепція як екземпляр, клас, атрибут, відношення, потік або актор, залежить від типів статичних і динамічних залежностей, через які вони пов’язані з іншими концепціями. Можливі інтерпретації понять представлені на рисунку 2.9. Семантичні залежності визначають, як можна класифікувати поняття.

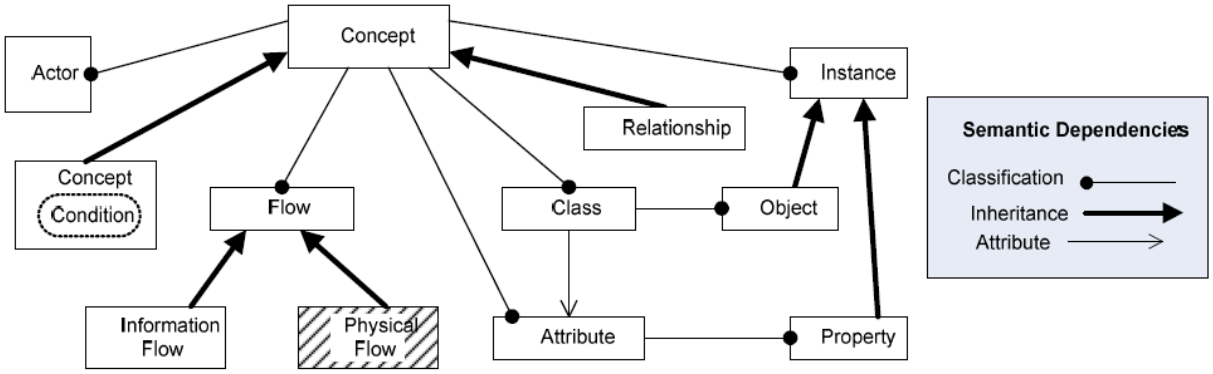


Рисунок 2.9 – Концепція метамоделі

Метамодель — це модель моделей або метамодель — це модель концептуальної основи мови, що складається з набору основних понять і правил, які визначають моделі в цій мові. Це відноситься до доменно-незалежних абстракцій і демонструє, що будь-яке поняття можна тлумачити по-різному. Він містить типи, екземпляри яких також є типами [32]. Цю діаграму слід аналізувати разом із метамоделлю зв’язків. Метамодель різних типів відносин представлена на рисунку 2.10.

Відношення є однією з ключових конструкцій, що використовуються в концептуальному моделюванні. Це найважливіший елемент у більшості моделей, які використовуються для визначення того, як концепції пов’язані між собою. Зв'язки між поняттями бувають двох типів: статичні та динамічні. «Дія» та «Правило» представляють дві підмножини «Динамічних зв’язків». Дії визначаються в термінах взаємодії між різними типами акторів. Залежно

від природи елемента потоку в дії, комунікаційні потоки можуть бути інформацією, рішенням або фізичним потоком. Потоки являють собою повідомлення або фізичні об’єкти, які мають бути передані від агентів до одержувачів.

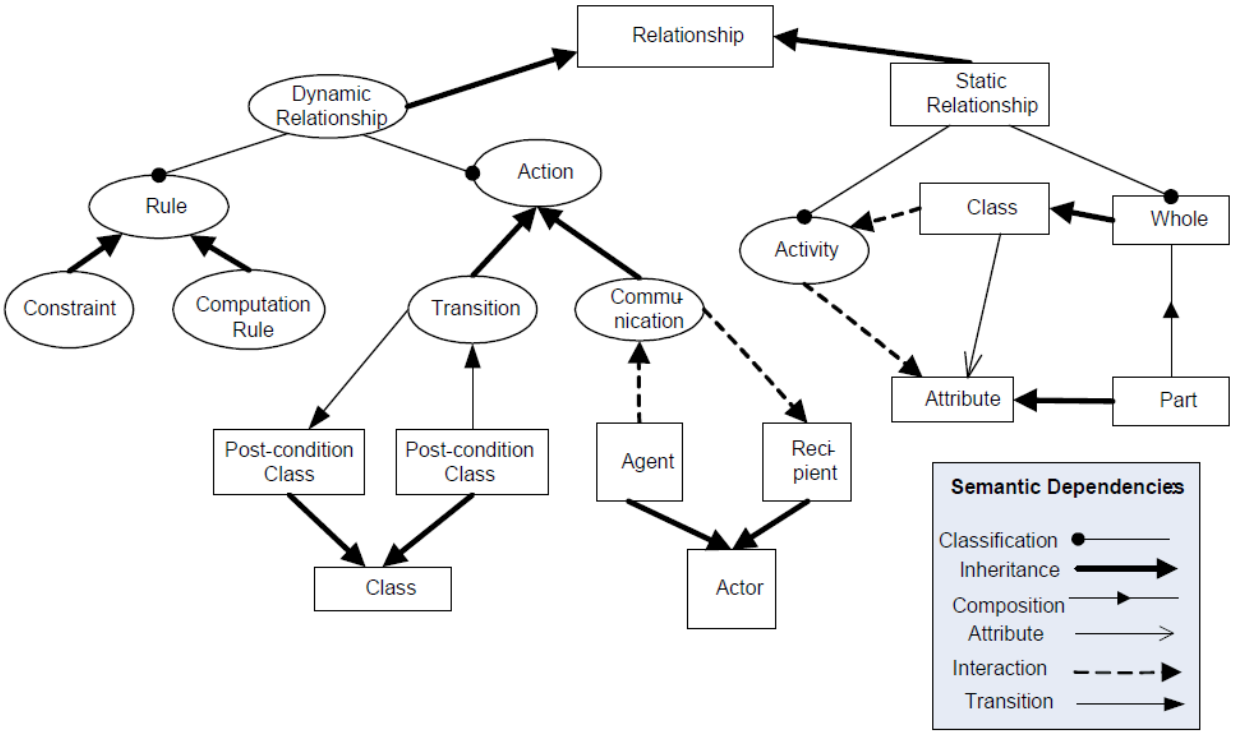


Рисунок 2.10 – Метамодель відношень

Слід зазначити, що в представленому сервіс-орієнтованому методі моделювання уникають таких понять, як відносини або кінці асоціацій, оскільки моделювання таким чином обмежує перспективу еволюції специфікації. Це означає, що під час процесу моделювання може знадобитися зафіксувати та вказати властивості, які представлені цими ролями асоціації, які є нестабільними. Феномен відносності семантичної ролі стверджує, що неможливо провести сувору класифікацію семантичних ролей, які виконує концепт. Інтерпретація понять багато в чому залежить від ситуації моделювання. Традиційні підходи до моделювання не підтримують відносність семантичної ролі, і це є причиною різноманітних проблем інтеграції поглядів.

2.3.2. Позначення статичних залежностей

Семантичні залежності, що використовуються на семантичному рівні, можуть бути двох видів; статичні залежності та динамічні залежності. Статичні залежності складають основу для розуміння поведінки об'єктів інформаційної системи. Вони традиційно використовуються для представлення лише структурних аспектів понять. Повна нотація залежностей, яка використовується в сервісно-орієнтованому моделюванні, визначена в наступному розділі.

Класифікація є статичною залежністю. Його можна використовувати для визначення об’єктів, які є екземплярами понять. Екземпляр можна розглядати як елемент набору, який визначається концептом, до якого він належить. Іноді залежність класифікації називають інстанціюванням, що є протилежністю класифікації. В об’єктно-орієнтованих підходах він представляє семантичне відношення між об’єктом і класом. Примірники понять поширюються на більш загальні поняття в ієрархії успадкування. Графічне позначення класифікаційної залежності представлено на рисунку 2.11.

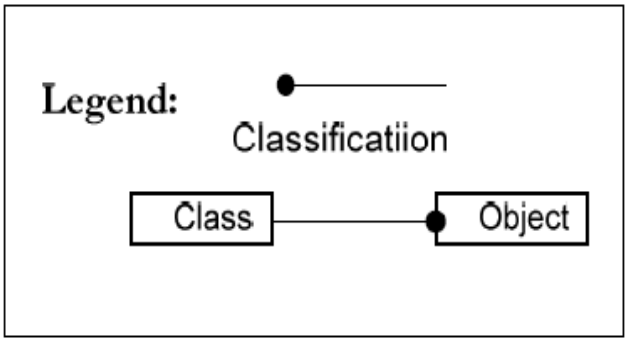


Рисунок 2.11 – Класифікаційна залежність

Успадкування — це статична залежність, яка використовується для спільного використання подібностей із більш загальних концепцій. Він попередньо визначає успадкування як статичних, так і динамічних посилань

залежностей для більш конкретних концепцій. Залежність від успадкування є дуже корисним відношенням для міркувань про семантичну цілісність концептуалізацій. Слід зазначити, що в об'єктно-орієнтованому підході успадкування застосовується лише для атрибутів і операцій. У представленому сервіс-орієнтованому методі моделювання залежності атрибутів, композиції, взаємодії та переходу також успадковуються більш конкретними поняттями від більш загальних. Спадкування використовується для специфікації суб-концептів або супер-концептів. Більш специфічні класи успадковують обов’язкові атрибути та взаємодії більш конкретних класів. Графічне позначення залежності успадкування представлено на рисунку 2.12.

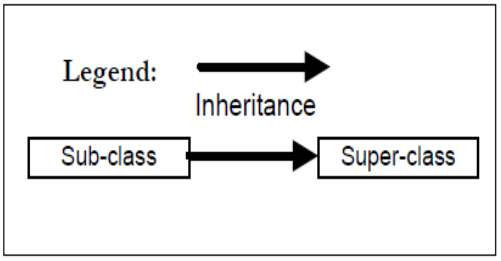


Рисунок 2.12 – Залежність: успадкування

У сервіс-орієнтованому способі моделювання набір усіх статичних різновидів асоціацій використовується для розрізнення атрибутів і не-атрибутів. Лише п’ять основних асоціацій з обов’язковими обмеженнями рекомендовані для кінцевого сервіс-орієнтованого аналізу та етапу проектування, оскільки асоціації з необов’язковими обмеженнями породжують семантичні діри, які спричиняють неповноту та неоднозначність концептуальних моделей. Відповідно до методу сервіс-орієнтованого моделювання концепти, які пов’язані факультативними асоціаціями, мають недостатньо визначену семантику. Оскільки необов’язкова потужність використовується в об’єктно-орієнтованому аналізі та проектуванні, її не заборонено використовувати на ранніх стадіях сервіс-орієнтованого аналізу та проектування. Графічне позначення залежностей атрибутів та їх потужностей представлено на рисунку 2.13.

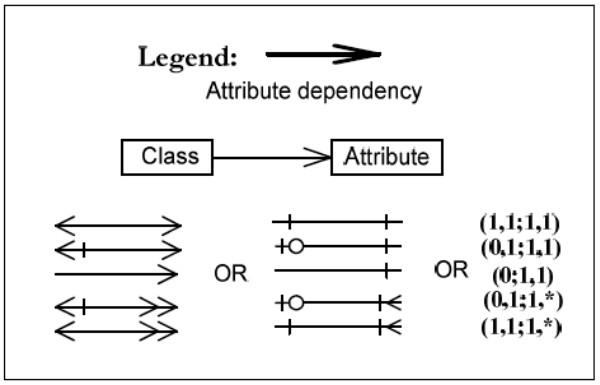


Рисунок 2.13 – Графічне представлення залежності атрибутів

Необов’язкові властивості не слід використовувати, якщо діаграми призначені для однозначного передачі семантичних деталей специфікацій ІС. Недолік такого обмеження полягає в тому, що діаграми збільшаться в розмірі, але перевага полягає в тому, що діаграми будуть керовані і семантика домену буде передана однозначно. Набір повністю придатних залежностей можна вважати хорошою основою для однозначного аналізу та проектування системи.

2.3.3.Позначення динамічних залежностей

Динамічні залежності в методі сервіс-орієнтованого моделювання використовуються для визначення відносин між різними акторами, їхніми діями та комунікаційними потоками. Описи організаційної діяльності, а також акторів, залучених до цієї діяльності, базуються на динамічних залежностях. Таким чином, динамічна частина моделі підприємства може бути представлена діями, які використовують і створюють різноманітні комунікаційні потоки, а також акторами, відповідальними за ініціювання цих дій. Динамічні зв'язки - це залежності стану та залежності зв'язку. Комунікаційні залежності між суб’єктами підприємства мають значення для опису точки зору «хто». Якщо концепція А пов’язана з В комунікаційною залежністю, тоді А є агентом, а В є одержувачем. Залежно від того, чи є в дії

елементи потоку чи ні, комунікаційні потоки можуть бути інформаційними, матеріальними або рішеннями.

Будь-яку залежність потоку можна описати більш детально за допомогою комунікативної дії. Залежність від актора вважається як дією, так і комунікаційним потоком. Згуртованість дії та комунікаційного потоку призводить до більш складної абстракції, яка називається комунікативною дією [43]. У такому випадку зв'язок залежності потоку між двома акторами вказує, що одержувач залежить від агента не лише конкретним потоком, але й дією. Дії будуть представлені еліпсом. Графічне позначення динамічних складових комунікаційної дії представлено на рисунку 2.14.

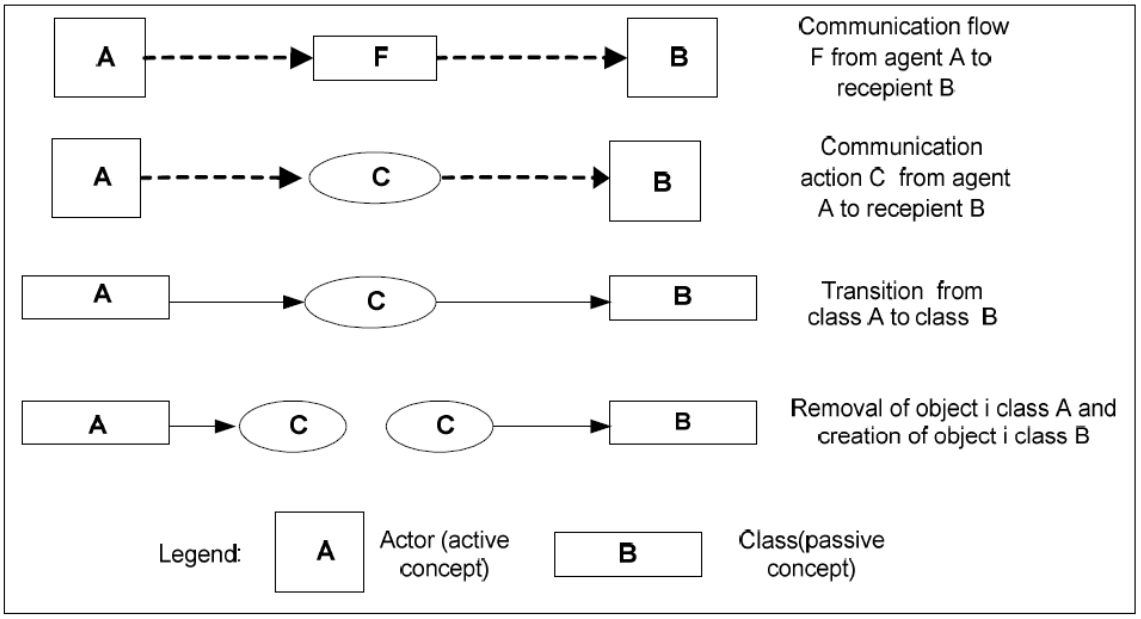


Рисунок 2.14 – Графічне позначення динамічних залежностей

Взаємодія є динамічним відношенням (--->). Він визначається комунікаційною дією між двома активними акторами. Це означає, що один актор залежить від іншого актора. Взаємодію також можна розглядати як дію та комунікаційний потік. У той же час взаємодія є стратегічним потоком залежністю між агентом і реципієнтом. Дії можуть бути ініційовані організаційними учасниками або технічними компонентами. Взаємодіючі компоненти можна сприймати як запитувачів послуг і постачальників послуг.

Залежність взаємодії важлива, оскільки вона визначає обов’язки учасників, які беруть участь у взаємодії. Ініціація дій залежить від цілей акторів. Міжсуб’єктивна перспектива, яка виражається залежностями взаємодії між різними компонентами підприємства, такими як актори, є відправною точкою в сервіс-орієнтованому моделюванні.

Перехід (→) — це динамічне відношення, яке в сервіс-орієнтованому моделюванні використовується для представлення фундаментальних змін об’єктів, що відбуваються під час подій створення, припинення та перекласифікації. У сервіс-орієнтованому моделюванні він представляє об'єктивну перспективу обслуговування. Зазвичай вони розглядаються як перспективи «як». Залежності переходів використовуються для представлення внутрішніх змін об’єктів, коли відбувається дія. Дії вказують на допустимі способи спричинення переходів шляхом маніпулювання властивостями об’єктів у різних класах. Зміни можна відстежити, перевіривши зміни в атрибутах, які визначають властивості об’єктів. Ці зміни мотивують дії, які відбуваються під час взаємодії. Якщо це не зміни значень атрибутів, то корисність деяких конкретних дій повинна бути піддана сумніву та перевірена.

Об’єкти класів попередньої та післяумови характеризуються атрибутами, яких має бути достатньо для розуміння деталей ефектів взаємодії. Це один із аспектів, які відрізняються підходом до сервісно-орієнтованого моделювання. В об'єктно-орієнтованих підходах переходи ніколи не асоціюються з класами, а лише зі станами. Приклад із служби зайнятості на рисунку 2.15 демонструє взаємодію та залежності переходів, а також семантичні відмінності, які характеризуються атрибутами.

Основну семантичну відмінність понять «Претендент» і «Працівник» можна виявити в атрибутах. Коли апліканта перекласифікують до працівника, до нього додаються два нові атрибути: контракт і посада.

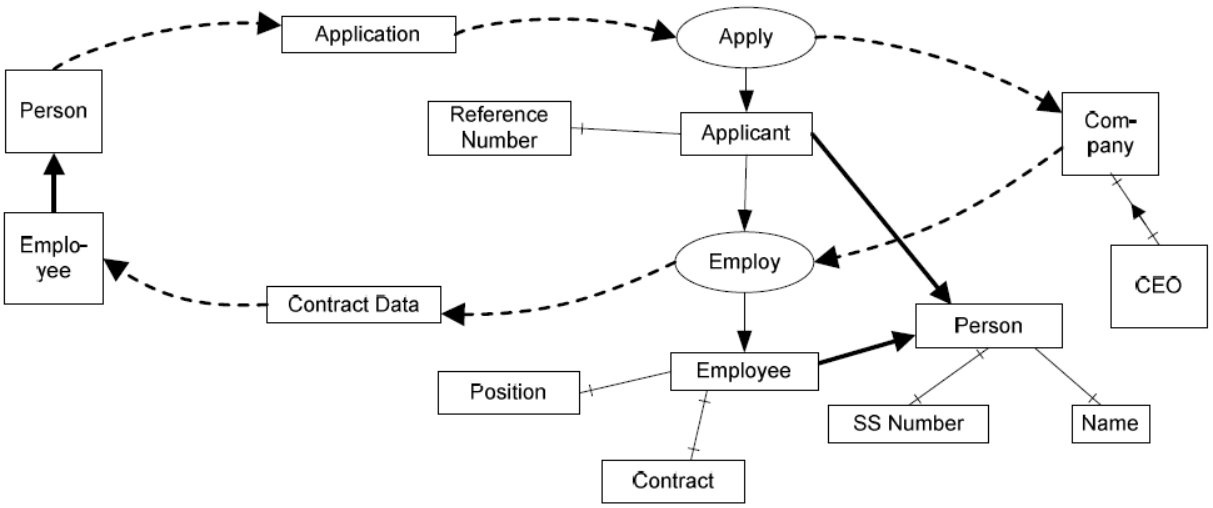


Рисунок 2.15 – Залежності взаємодії та переходу

Експерти з аналізу повинні узгодити узгоджений набір взаємодій сервісів, які визначаються за допомогою статичних і динамічних залежностей. Однією з цілей семантичних залежностей, які використовуються в концептуальному моделюванні, є з'ясування відмінностей між поняттями. Представлені семантичні залежності можна використовувати для визначення семантичних деталей процесу та даних на різних рівнях абстракції.

Висновки до розділу 2

В даному розділі представлено метод сервіс-орієнтованого проектування та моделювання: процес моделювання та мова представляє два фундаментальні основи методу. Описано процес моделювання, який проходить через три рівні і використовувані залежності. Основи сервіс-орієнтованої мови моделювання представлено мета-моделлю, пояснюючи, як концепції інтерпретуються в методі сервіс-орієнтованого моделювання. Також подано та пояснено позначення статичних і динамічних залежностей, які використовуються для сервіс-орієнтованого моделювання.

РОЗДІЛ 3. ІМПЛЕМЕНТАЦІЯ СЕРВІС-ОРІЄНТОВАНИХ МОДЕЛЕЙ НА ОСНОВІ ШАБЛОНІВ ДЛЯ ПОКРАЩЕННЯ ЕФЕКТИВНОСТІ ПРОЕКТУВАННЯ ПРОГРАМНИХ РІШЕНЬ

3.1 Сутність пропонованого методу сервіс-орієнтованого моделювання

В цьому розділі демонструється, як застосовувався новий метод. Він демонструє, як мова моделювання та процес моделювання були застосовані для мотивації та обґрунтування нового сервіс-орієнтованого методу моделювання. Результати дослідження у формі застосування нового методу будуть продемонстровані за трьома темами дослідження:

1) Прагматична специфікація інформаційних систем,

2) Сервіс-орієнтована основа моделювання для аналізу та проектування інформаційних систем (включає сервіс-орієнтовані конструкції, шаблони аналізу та метод інтеграції між статичними та динамічними аспектами),

3) Принципи переходу до проектування, специфічного для реалізації, які показують різні дослідження переходу від моделі, орієнтованої на послуги, до проектування, специфічного для реалізації.

Ці три теми дослідження пов’язані з трьома рівнями архітектурної основи для процесу сервіс-орієнтованого моделювання. Результати дослідження цих трьох тем сприяли розробці системного та інтегрованого сервіс-орієнтованого методу моделювання, який має декілька нових особливостей. Основними відмінностями від інших відомих методів моделювання є наступні:

- Він забезпечує систематичне моделювання через три рівні абстракції, що необхідно для поєднання вимог дизайну та технічного дизайну. Він контролює відповідність між питаннями бізнесу та інформаційних технологій і може використовуватися для покращення

спілкування між двома типами зацікавлених сторін: експертами з бізнесу та дизайну. Цей метод моделювання забезпечує мову моделювання, процес моделювання та спосіб переходу від одного рівня до іншого. У ньому наведено вказівки та методи для перемикання між рівнями. Сервісно-орієнтований метод моделювання забезпечує контрольований спосіб моделювання, який допомагає однозначно передавати семантику домену.

- Сервіс-орієнтований метод моделювання дозволяє інтегрувати статичні та динамічні аспекти в єдину нотацію моделювання. Узгодженість обох аспектів має вирішальне значення для полегшення міркувань і визначення цілісного розуміння архітектури підприємства. Семантична цілісність статичного та динамічного аспектів може бути досягнута шляхом комбінування зовнішнього та внутрішнього вигляду послуг. Взаємодія між інтерсуб'єктивною та об'єктивною перспективами, використовуючи єдину сервіс-орієнтовану структуру, сприяє цій інтеграції. Різноманітні статичні та динамічні залежності дають можливість виявити та усунути такі небажані характеристики специфікацій системи, як неоднозначність, надмірність, неповнота щодо цілей, суперечливість. Ці критерії є критичними для досягнення кращої якості специфікацій Ic.
- Метод показує, як поєднати моделі, нейтральні до обчислень, і дизайн, що залежить від реалізації. Сервісно-орієнтовані діаграми заздалегідь визначають семантичні деталі, які можна використовувати для створення об'єктно-орієнтованих діаграм. Сервісно-орієнтовані діаграми не мають упередженості щодо реалізації, тому їх можна використовувати для подолання комунікаційного розриву між системними дизайнерами та бізнес-експертами.

На рисунку 3.1 показано взаємозв'язок між архітектурною структурою та темами дослідження.

Перша тема дослідження, Прагматична специфікація інформаційних систем, сприяє методу систематичного моделювання для структурування

прагматичних знань про різні послуги з використанням прагматичних залежностей і визначає аспект «чому» проблемної області. Наміри бізнес-експертів представлені в термінах набору прагматичних залежностей, які керують загальним процесом проектування системи. Прагматичні специфікації мають на меті забезпечити обґрунтування концептуальних представлень компонентів підприємства на семантичному рівні.

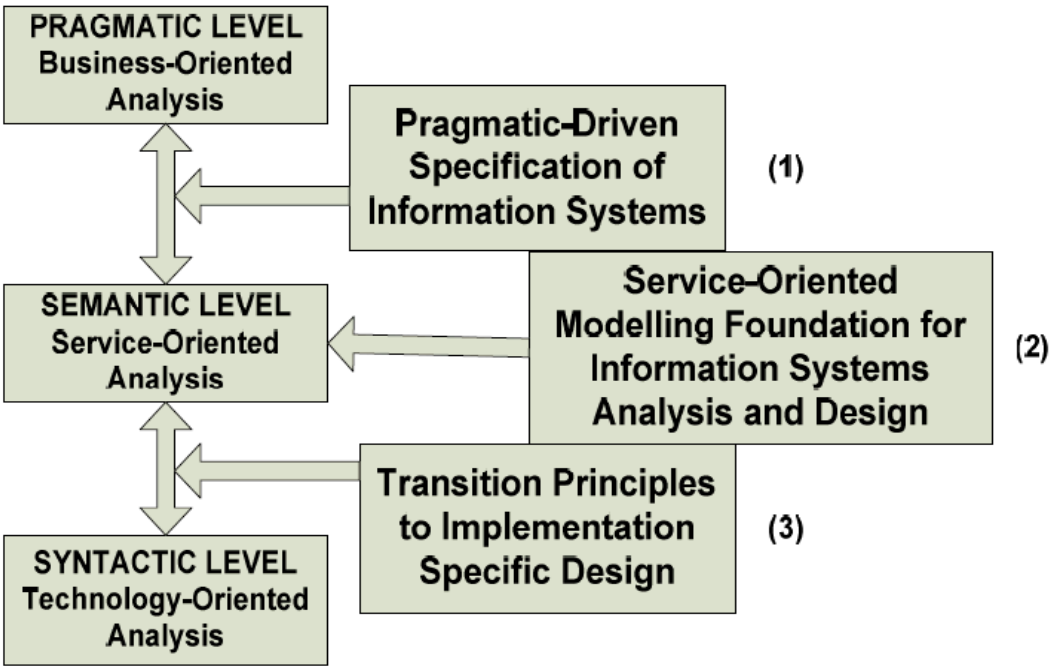


Рисунок 3.1 – Представлення архітектурної основи

Друга тема дослідження, Сервіс-орієнтована основа моделювання для аналізу та проектування інформаційних систем, сприяє семантичній інтеграції статичних і динамічних аспектів. Негласна структура сервісів, включаючи структурні та динамічні аспекти, важлива для систематичного аналізу архітектур сервісів. Це має вирішальне значення для візуалізації та підтримки цілісного представлення зовнішніх і внутрішніх поглядів однієї системи. Ця тема також сприяє створенню сервіс-орієнтованих конструкцій, етапів моделювання та шаблонів аналізу.

Третя тема дослідження, «Принципи переходу до проектування, специфічного для впровадження», сприяє отриманню знань, які показують перехід від нейтрального обчислення до дизайну, орієнтованого на реалізацію. Це дослідження сприяє розробці технічних компонентів. Показано, як сервіс-орієнтовані конструкції можуть бути зіставлені з технічними компонентами та об’єктно-орієнтованими діаграмами.

3.2 Розробка методу сервіс-орієнтованого моделювання для аналізу та проектування інформаційних систем

У цьому розділі представлено основи нового сервіс-орієнтованого методу моделювання для аналізу та проектування інформаційних систем. Метою цього нового сервіс-орієнтованого методу моделювання є надання нотації мови графічного моделювання та набору принципів проектування, які дають змогу аналізувати корпоративну архітектуру з точки зору послуг. Аналіз виконано на високому рівні модульності, визначаючи послуги як бізнес-одиниці. В основу цього методу покладено онтологічні принципи поняття послуги. Він базується на передумові, що концепція сервісу може бути застосована для концептуалізації архітектури системи підприємства у формі організаційних і технічних компонентів системи. Цю архітектуру можна визначити як набір слабко пов’язаних компонентів, які взаємодіють один з одним.

Основними онтологічними принципами, покладеними в основу методу, є наступні:

- світ складається з речей і все змінюється. Речі мають властивості. Ми вивчаємо їх і модифікуємо, щоб зрозуміти зміни, які є результатом певних дій (об'єктивна перспектива);
- речі згруповані в системи. Незалежних підсистем немає. Кожна система взаємодіє з іншими системами (інтерсуб'єктивна перспектива).

Відправна точка онтологічного визначення корпоративної системи в представленій сервіс-орієнтованій основі досить подібна до онтологічного розуміння системи та підприємства як онтології системи та підприємства. Кожне підприємство можна розглядати як сукупність організаційно-технічних компонентів. Представлений сервіс-орієнтований метод базується на припущенні, що моделі бізнес-процесів складаються з слабо пов’язаних компонентів, які розглядаються як запитувачі послуг і постачальники послуг. Взаємодіючи, вони викликають зміни в об'єктах, які визначаються властивостями. Організаційні та технічні актори можна розглядати як слабко пов’язані компоненти, які можна розкласти за допомогою різних залежностей. Щоб охопити цілісну структуру проблемної області, необхідно розуміти статичні та динамічні аспекти компонентів підприємства та те, як різні компоненти взаємопов’язані. На підприємстві взаємодія будується навколо довгострокових відносин, які відображають життя підприємства.

Концепція орієнтації на послуги використовується в різних контекстах і з різними цілями, але є один аспект, який є незмінним у своєму існуванні, а саме те, що вона представляє окремий підхід до розділення проблем. Це означає, що організаційно-технічні рішення проблем можна конструювати та управляти, розкладаючи їх на менші підсистеми. Декомпозиція та вдосконалення послуг на прагматичному рівні та їх аналіз за допомогою прагматичних залежностей допомагають бізнес-аналітикам побачити та зрозуміти структуру цілей. Зосередившись на архітектурі сервісу, з точки зору того, як бізнесмени її розуміють, можна зробити сервіс-орієнтований аналіз і проектування більш прибутковими.

Архітектура сервісу базується на припущенні, що моделі бізнес-процесів можуть складатися з слабо пов’язаних компонентів, які розглядаються як запитувачі послуг і постачальники послуг. Відповідно до філософії сервіс-орієнтованої основи, кожне підприємство можна розглядати як сукупність організаційних і технічних компонентів, які розглядаються як

різні типи акторів підприємства. Поняття підсистеми є фундаментальним у представленому сервіс-орієнтованому методі, оскільки кожен компонент можна розглядати як підсистему. Аналіз, заснований на акторах як компонентах і підсистемах, допомагає запровадити принцип поділу інтересів, що важливо для досягнення незалежності вимог.

Будь-які два актори, організаційні чи технічні, можуть бути пов’язані різними семантичними залежностями. Принципи сервіс-орієнтованого методу базуються на проектуванні взаємодії між різними акторами підприємства. Дві залежності взаємодії в протилежних напрямках між акторами утворюють петлю взаємодії. Розрив взаємодії між послугами спричиняє збої в специфікаціях бізнес-процесів, що знижує якість специфікацій системи. Поломки свідчать про незавершеність взаємодії сервісів.

3.2.1. Основні елементи та перспективи сервіс-орієнтованого моделювання

Аналіз системи означає створення загального набору понять; щоб повідомити ці концепції в процесі розробки, вони повинні бути однозначними та зрозумілими для всіх зацікавлених сторін. Взаємозв'язки домену складають значну частину його неявної структури. Глибоке розуміння домену залежить від знання того, як усі об’єкти проблемного домену взаємопов’язані. Дуже часто неоднозначні або неповні вимоги є результатом недостатнього аналізу даних на рівні аналізу та неможливості визначити повний набір взаємозв’язків між різними елементами домену. Це призводить до того, що неповні специфікації ІС передаються на етап проектування, що спричиняє неточний зв’язок між етапами аналізу та проектування.

Сервіс-орієнтований аналіз базується на моделюванні міжсуб’єктивних і об’єктивних перспектив системи підприємства через межі організаційної та технічної системи. Інтерсуб'єктивна перспектива визначає, як суб'єкти

підприємства, які можуть бути організаційними або технічними компонентами, пов'язані один з одним. Цих учасників можна розглядати як запитувачів послуг і постачальників послуг. Ця перспектива означає певні зобов’язання та відповідальність між суб’єктами підприємства, які мотивуються на прагматичному рівні. Об'єктивна перспектива визначає, як різні об'єкти змінюються, коли відбуваються дії під час процесу взаємодії між акторами. Зміни стану об'єктів визначають їхню поведінку. Предметна перспектива вказує на внутрішні зміни предметів. Дії викликають переходи, які вносять зміни в атрибути об'єктів.

Одним із основних елементів сервіс-орієнтованої основи є актори підприємства. Актори — це підсистеми, які представлені окремими особами, організаціями та їхніми підрозділами або ролями, які позначають групи людей. Технічні суб’єкти — це такі підсистеми, як машини, програмні та апаратні компоненти тощо. Будь-які два суб’єкти можуть бути пов’язані залежностями успадкування, композиції, класифікації чи взаємодії, які графічно представлені на рисунку 3.2.

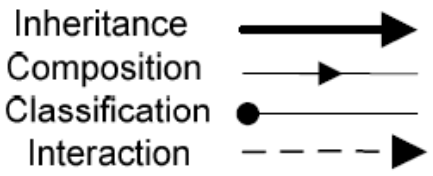


Рисунок 3.2 – Залежності акторів

Залежність успадкування між акторами використовується для спільного використання статичних і динамічних подібностей. Більш конкретні актори успадковують склад і залежності взаємодії від більш загальних акторів. Залежності представляють додаткові властивості внутрішньої взаємодії акторів і структурні властивості, які призначаються корпоративною системою.

Композиція — це концептуальна залежність, яка використовується для зв’язку цілого з іншими поняттями, які розглядаються як частини. Це більш суворе семантичне відношення порівняно з агрегацією та композицією, яка визначається в об’єктно-орієнтованих підходах.Композиція характеризується такими властивостями:

а) частина не може одночасно належати більш ніж одному цілому того самого поняття;

б) якщо вона належить більш ніж одному цілому, то це має бути ціле, яке є екземпляром інше поняття;

в) частина і ціле створюються одночасно.

Коли частина створюється, вона припиняється одночасно з припиненням цілого. Класифікаційний зв’язок між двома акторами використовується для визначення їхніх екземплярів. У концептуальному моделюванні екземпляр можна розглядати як елемент набору, який визначається концепцією, до якої він належить.

Залежності взаємодії використовуються для концептуалізації послуг між різними учасниками корпоративної системи. Оскільки актори реалізовані як організаційно-технічні компоненти системи, вони можуть використовувати один одного, відповідно до заданих шаблонів, для досягнення своїх цілей. Дві залежності взаємодії в протилежних напрямках між запитувачем послуг і постачальником послуг визначають типовий цикл робочого процесу дій [44].

Постачальники послуг — це актори, які зазвичай отримують запити на обслуговування, над якими вони не мають прямого контролю, і ініціюють відповіді на обслуговування, які надсилаються запитувачам послуг. Систему можна визначити як набір слабко пов’язаних взаємодіючих компонентів, які можуть виконувати певні послуги за запитом. Приклади залежностей акторів представлені на рисунку 3.3.

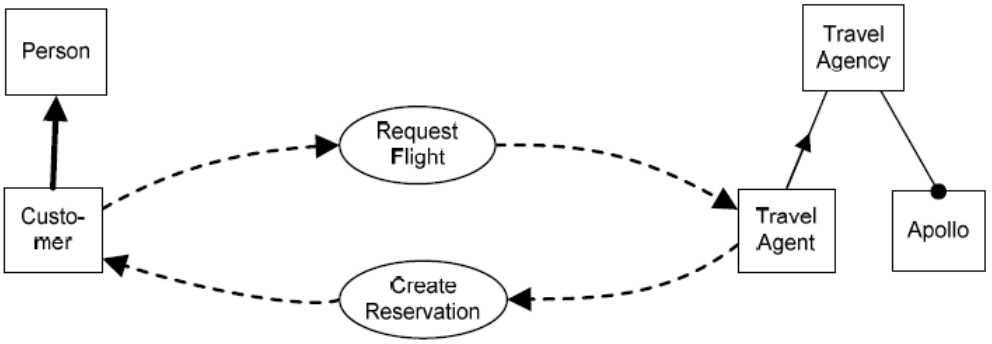


Рисунок 3.3 – Приклад семантичних залежностей між акторами

Сервісні запити та сервісні відповіді розуміються як комунікаційні дії. Ми використовуємо поняття «комунікаційна дія» в загальному сенсі, щоб описати співпрацю або акт людської поведінки з певним наміром або суб'єктивним значенням. У сервісно-орієнтованому методі моделювання розрізняють спілкування та взаємодію.

3.2.2. Інтерсуб'єктивна перспектива

Інтерсуб'єктивна перспектива дії вказує на те, що один актор залежить від іншого актора. Ця перспектива підкреслює завдання акторів та організаційних підрозділів, які здійснюють процес. Він представлений нотацією залежності взаємодії (--□). Значення стрілки взаємодії подібне до зв'язку стратегічної залежності, який був представлений підходом i* [38], де актори пов'язані один з одним для досягнення цілей, виконання завдань і надання ресурсів. Інтерсуб'єктивна перспектива зазвичай розрізняється шляхом визначення фізичного, інформаційного потоку або потоку рішень між двома залученими акторами. Ця перспектива є важливою, оскільки вона представляє акторів як незалежних суб'єктів, які додають цінність, виконуючи діяльність. У представленому сервіс-орієнтованому методі моделювання стратегічна залежність розглядається водночас як потік дії та взаємодії. Дії здійснюють актори, яких називають агентами. Ці агенти можуть розглядатися як запитувачі послуг або постачальники послуг. Залежність

взаємодії між цими акторами можна розглядати як канал зв'язку для передачі інформації, фізичного потоку або потоку рішень від агента до одержувача. Як правило, агент надсилає потік одержувачу, щоб досягти своєї мети. Досягнення мети залежатиме від постачальника послуг, який повинен доставити потік послуг у зворотному напрямку. Цикл взаємодії між запитувачем послуг і постачальником послуг повинен бути повним і замкнутим циклом, щоб досягти мети.

Потоки — це концепції, які представлені прямокутниками. Суцільні рамки використовуються для представлення фізичних потоків, а світлові коробки представляють потоки інформації. Комунікаційна дія без інформаційного або фізичного компонента потоку визначає рішення. Графічні позначення трьох різних потоків зображені на рисунку 3.4.

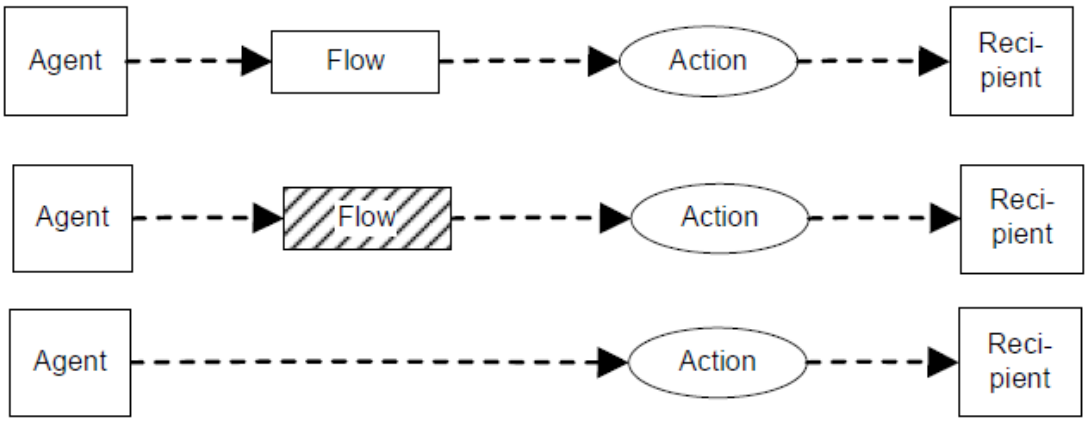


Рисунок 3.4 – Позначення інформації, фізичних потоків і потоків рішень

Інтерсуб'єктивна перспектива допомагає візуалізувати, хто використовує дані, і відслідковувати використані дані. Відстежуваність даних важлива для вирішення проблем із якістю даних. Вона може свідчити про розриви або порушення бізнес-процесів, які необхідно виявити на ранній стадії, оскільки це може спричинити проблеми на наступних етапах.

Для концептуалізації організаційної та технічної частин інформаційних систем було виділено інтерсуб'єктивну та об'єктивну точки зору. Об'єктивну перспективу можна застосувати для представлення внутрішньої поведінки

об’єктів. Він являє собою дані, які аналізуються в контексті взаємодії між організаційними та технічними компонентами системи. Взаємодії між різними суб’єктами можуть бути використані для виявлення змін властивостей об’єкта, які важливі для виявлення семантичного значення предметної області. Рисунок 3.5 представляє два ракурси циклу взаємодії.

Інтеграція внутрішньої та зовнішньої поведінки, яка інкапсульована в концепції послуги, надає можливість виразити динамічні аспекти, які включають взаємодію між зовнішніми акторами (інтерсуб’єктивна перспектива) та поведінку, яка визначає стани об’єктів (об’єктивна перспектива).

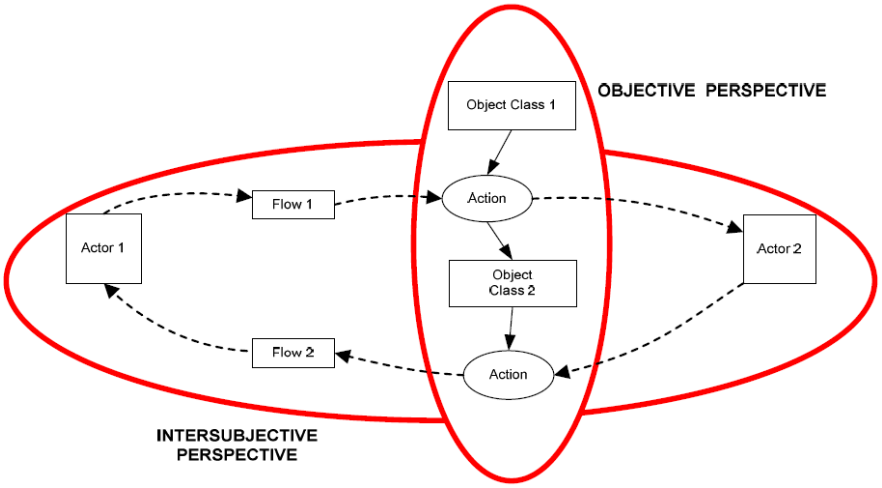


Рисунок 3.5 – Дві перспективи одного циклу взаємодії

Він також виражає структурні аспекти, які визначають внутрішні дані. Такий спосіб моделювання послуг дозволяє інтегрувати процес і дані. Об’єднання цих точок зору в одну нову абстракцію дає змогу представити статичні та динамічні аспекти за допомогою єдиної нотації моделювання. Перевагою єдиної нотації моделювання є те, що вона забезпечує контроль семантичної узгодженості статичних і динамічних аспектів. Семантична цілісність статичних і динамічних залежностей у різних циклах взаємодії є однією з переваг сервіс-орієнтованого аналізу та проектування. Традиційні нотації моделювання інформаційних систем не поєднують статичний і

динамічний аспекти, представляючи їх натомість у абсолютно різних типах діаграм.

3.3. Представлення семантичних конструкцій сервіс-орієнтованого моделювання

Базові конструкції є фундаментальними семантичними конструкціями для методу сервіс-орієнтованого моделювання, представленого в роботі. Ці конструкції базуються на двох базових подіях, які є основними подіями, що використовуються для представлень сервісу. Представлення послуг будуються шляхом концептуалізації взаємодії між організаційними та технічними компонентами, які можна розглядати як різні типи акторів підприємства.

Метою представлених семантичних конструкцій є те, що їх можна використовувати для сервіс-орієнтованого аналізу та семантичної інтеграції різних розмірів моделювання. Є дві основні події, фундаментальні для семантичних сервіс-орієнтованих конструкцій: події створення та завершення. Вони є основними для визначення рекласифікації, навіть якщо її можна зрозуміти як комунікаційну дію. По суті, під час будь-якого переходу можливі три види змін: дія або завершує, або створює об’єкт, або вона може виконувати завершення та створення одночасно. Створення позначається вихідною стрілкою переходу до класу постумови. Графічне позначення дії створення представлено на рисунку 3.6.

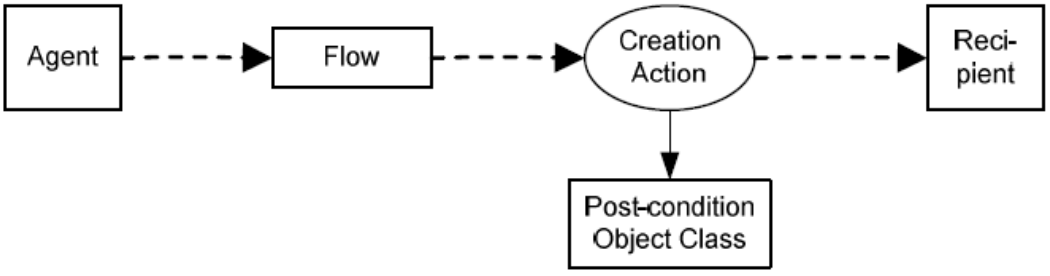


Рисунок 3.6 – Графічне представлення дії створення

Дія завершення представлена залежністю переходу, спрямованою з класу об’єкта передумови. Перш ніж припинити дію об’єкта, його необхідно створити. Оскільки майбутній стан не має сенсу для події завершення, він не включається в специфікацію дії. Клас попередньої умови в дії завершення можна розуміти як остаточний у часі життя об’єкта. Графічне позначення дії завершення представлено на рисунку 3.7.

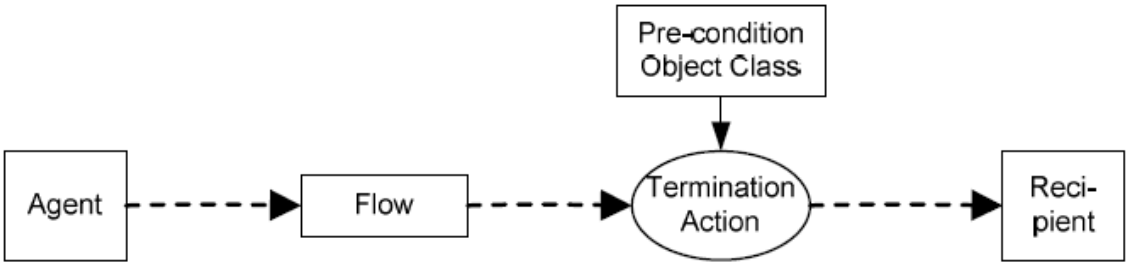


Рисунок 3.7 – Графічне представлення дії завершення

Перекласифікація об’єкта може бути визначена в термінах комунікаційної дії, яка припиняє об’єкт в одному класі та створює його в інший клас. Іноді об'єкти проходять кілька класів, а потім їх видаляють. Графічне позначення дії перекласифікації представлено на рисунку 3.8.

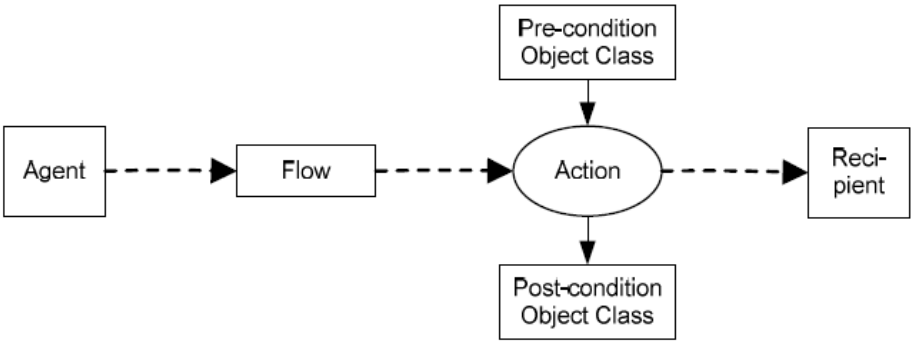


Рисунок 3.8 – Графічне представлення дії перекласифікації

Класи попередньої та післяумови зазвичай визначають обмеження на об’єкти, які обмежують надсилання та отримання потоків зв’язку між технічними або бізнес-компонентами. Дія перекласифікації в

комп’ютеризованій системі може бути реалізована як послідовність однієї чи кількох операцій створення та завершення об’єкта (або читання й оновлення). Потоки запитів і відповідей разом із створеними та завершеними класами об’єктів є вирішальними для розуміння семантичних аспектів послуг. Стан об’єкта попередньої умови та вхідний потік повинні бути достатніми для визначення потоку вихідних даних служби та стану об’єкта після умови.

Архітектура сервісу може складатися з різних циклів взаємодії. Семантику такої композиції можна визначити за допомогою двох або більше конструктів основних дій. Композиція цих трьох типів базових конструкцій використовується для концептуалізації безперервного або кінцевого життєвого циклу для одного або кількох об’єктів у циклі обслуговування. Приклад взаємодії трьох основних подій в одному сервісному циклі представлено на рисунку 3.10.

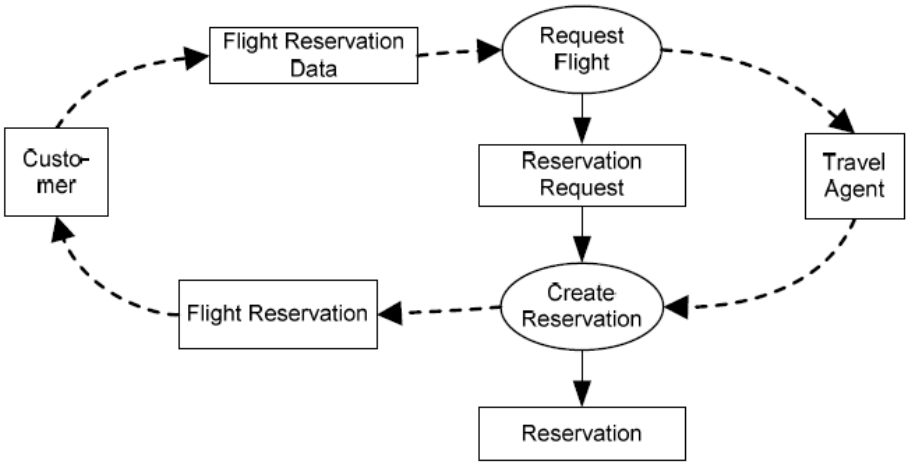


Рисунок 3.10 – Базовий цикл обслуговування

У цьому прикладі послуга бронювання рейсів складається зі створення, припинення та перекласифікації. Дію «Запит на рейс» може ініціювати Клієнт, який має право надсилати дані бронювання рейсу Турагенту. Запит на бронювання можна переглядати як збережений запис у веб-системі бронювання квитків. Якщо збережені дані задовольняють певну умову, тоді турагент викликає дію «Створити бронювання», яка може бути реалізована як

програмний компонент. Згідно з представленою схемою, дія перекласифікації створить об’єкт Reservation, і в той же час об’єкт Reservation Request необхідно видалити. Клієнта буде проінформовано про його бронювання, надіславши йому інформацію про потік бронювання рейсу.

Семантика змін об'єкта виражається за допомогою трьох типів дій: створення, припинення та перекласифікація. Життєвий цикл об'єкта зазвичай представлений початковим, проміжним і кінцевим класом. Подія створення відповідає початковій точці, а дія видалення – кінцевій точці життєвого циклу об’єкта. Найбільш критичним питанням у моделюванні деталей взаємодії є семантична цілісність статичного та динамічного аспектів. Недостатньо уявити, який тип створених об’єктів і припинено. Наприклад, Запит на бронювання та Дані бронювання авіаквитків у прикладі можуть відповідати представленню бази даних або вони можуть бути реалізовані як незалежні класи об’єктів. Сервісно-орієнтовані моделі та правила повинні чітко представляти семантичні деталі значень атрибутів, які мають бути видалені або збережені під час будь-якої дії створення, припинення та перекласифікації.

3.4 Розробка шаблонів аналізу для покращення ефективності сервіс-орієнтованого аналізу та проектування

Базові сервіс-орієнтовані конструкції можна використовувати для побудови шаблонів системного аналізу. Вони подібні до шаблонів робочого процесу, які були створені з метою окреслення вимог, що виникають під час моделювання бізнес-процесів на регулярній основі, і представлення їх у напівформальному описі. Одним із головних внесків цієї роботи є представлення таких моделей. Було продемонстровано, що сервіс-орієнтована мова моделювання достатня для визначення основних системного аналізу та шаблонів проектування, таких як послідовність, синхронізація, ітерація, вибір і пошук. Шаблони робочих процесів зазвичай визначаються за

допомогою нотації моделювання бізнес-процесів (BPMN) для діаграми бізнес-процесів і діаграми активності UML від групи керування об’єктами (OMG - Object Management Group). Обидві нотації можуть виражати поведінку процесу, але не враховують статичну частину бізнес-процесу. Він явно не показує, що відбувається з об’єктами, коли відбувається дія.

У цій роботі представлено декілька основних шаблонів аналізу для сервіс-орієнтованого аналізу та проектування. Також наведено приклади відповідної поведінки. На відміну від традиційних шаблонів робочого процесу, шаблони аналізу бізнес-процесу, представлені в роботі, побудовані шляхом поєднання статичних і динамічних залежностей. Сервісно-орієнтовані конструкції, що використовуються для представлення шаблонів, визначаються як цикл взаємодії між запитувачем послуг і постачальником послуг. Виразна сила статичних і динамічних залежностей достатня для визначення основних шаблонів робочого процесу, таких як: послідовність, вибір (вибір і злиття), синхронізація (розділення і об'єднання) і ітерація. Семантику архітектури сервісу можна визначити за допомогою одного або комбінації кількох циклів взаємодії. Кожен цикл взаємодії складається з дій створення, завершення або перекласифікації. Зіставляючи залежності взаємодії від агентів до одержувачів, можна досліджувати можливості, доступні для акторів. Статичні залежності визначають додаткові семантичні деталі взаємодії, які є важливими для міркування про шаблони архітектури сервісу.

Шаблони моделювання для сервіс-орієнтованого аналізу та проектування важливі з двох основних причин. По-перше, їх можна використовувати для демонстрації взаємодії фундаментальних конструкцій, які використовуються в системному аналізі та процесі проектування. По-друге, шаблони важливі для оцінки виразної сили мов семантичного моделювання. Розуміння та візуальне розпізнавання основних патернів необхідні для побудови більш конкретних варіацій патернів шляхом їх компонування різними способами.

Шаблон взаємодії служби: послідовність. Шаблон послідовності визначається впорядкованою серією дій. Одна дія починається після завершення попередньої. Шаблон послідовності можна визначити за допомогою композиції двох або більше дій перекласифікації. Оскільки дія створення та припинення є окремим випадком перекласифікації, її можна використовувати замість дії перекласифікації. Приклад шаблону послідовності представлено на рисунку 3.11.

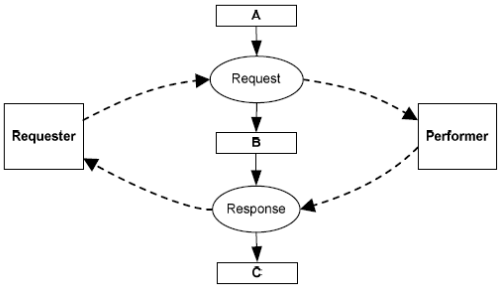


Рисунок 3.11 – Шаблон послідовності

Приклад цього шаблону послідовності представлено на рисунку 3.12.

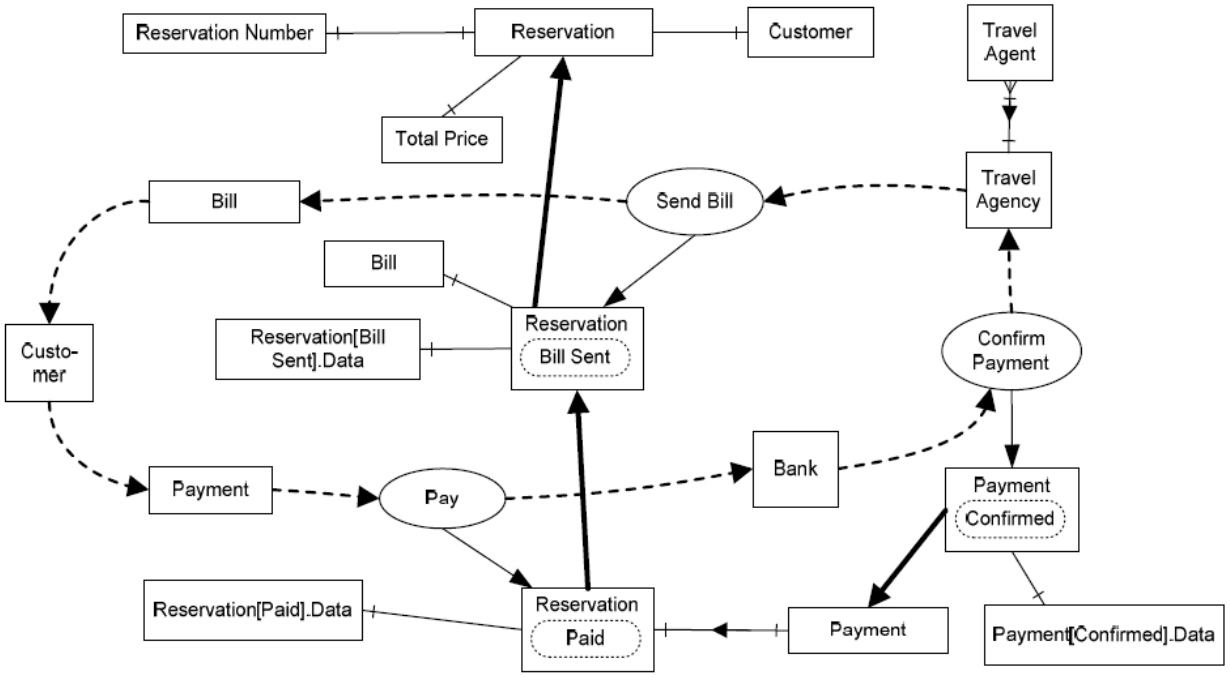


Рисунок 3.12 – Приклад застосування шаблону послідовності

У представленому прикладі послідовність із трьох дій створення використовується для вираження шаблону послідовності: «Надіслати рахунок», «Оплатити» та «Підтвердити платіж». Дію «Оплатити» можна виконати, лише якщо виконано дію «Надіслати рахунок», а дію «Підтвердити платіж» можна обробити, лише якщо процес оплати завершено.

Зміни, які відбуваються з об'єктами, представляють статичні аспекти системи (об'єктивна перспектива). Коли туристичне агентство надсилає рахунок Клієнту, дія Надіслати рахунок створює новий об’єкт Reservation у стані [Bill Sent], який є спеціалізацією об’єкта Reservation, створеного в попередньому циклі взаємодії. Об’єкт має два специфічні атрибути: Reservation[Bill Sent].Data та Bill. Коли Клієнт отримує рахунок і запускає процес оплати, дія Pay створює новий об’єкт Reservation[Paid] із певними атрибутами Payment і Reservation[Paid].Data.

Кожна дія відповідає за видалення зв’язків об’єкта атрибута, які пов’язані з класом передумови, і за створення зв’язків об’єкта атрибута з класом постумови. Він не несе відповідальності за будь-які зміни зв'язків об'єктів більш загальних класів або за зв'язки об'єктів атрибутів. Створення, перекласифікація та видалення об’єктів у більш загальних класах і в атрибутах, які розглядаються як класи з власними атрибутами, має відбуватися в більш ранній послідовності. Наприклад, для запуску дії «Надіслати рахунок» об’єкт «Бронювання» має бути заздалегідь створений іншою службою.

Шаблон взаємодії служби: Синхронізація. Іноді деякі дії потрібно виконувати одночасно, а не послідовно. Шаблон синхронізації поєднує шлях цих дій. Важливо, щоб останній набір заходів було завершено до того, як можна буде продовжити наступний процес. Схема синхронізації представлена на рисунку 3.13.

Цей шаблон ілюструє, що дія відповідає за видалення об’єкта A та всіх його частин B. Створення D вимагає створення принаймні одного об’єкта E. Об’єкти композиційних атрибутів повинні бути створені, перекласифіковані

або завершені тією ж дією, тому що частина і ціле мають однакові життєві цикли. Якщо створюється об'єкт, то створюються і зв'язки з композиційною частиною.

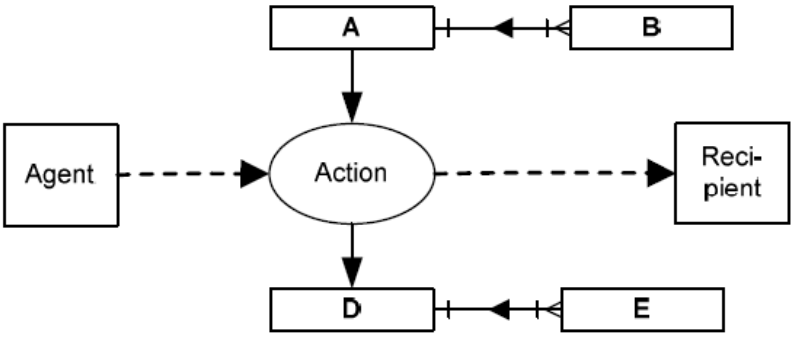


Рисунок 3.13 – Шаблон синхронізації

Якщо об'єкт видалено, тоді посилання видаляються/від'єднуються одночасно. Ось чому дія поширюється відповідно до класової композиції зв'язків від цілого до частини і навпаки. Розповсюдження дій є корисною властивістю, оскільки воно дозволяє моделювати синхронізацію природним шляхом. Приклад шаблону синхронізації показано на рисунку 3.14.

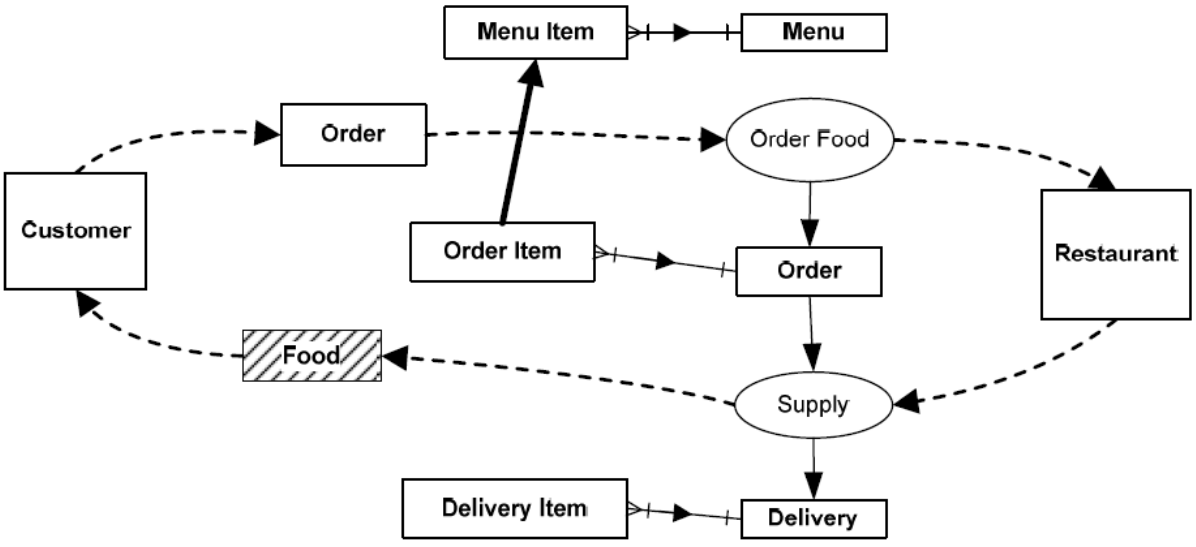


Рисунок 3.14 – Приклад шаблону синхронізації

Дія постачання поширюється на частини: припинення замовлення та елементів замовлення та створення доставки та елементів доставки. Харчування (матеріальний потік) означає доставку набору предметів, які були предметами замовлення.

Шаблон взаємодії служби: ітерація. Шаблон ітерації — це окремий випадок послідовності, де клас постумови дії відповіді відіграє роль класу попередньої умови для запиту на послугу. Шаблон зображено графічно на рисунку 3.15.

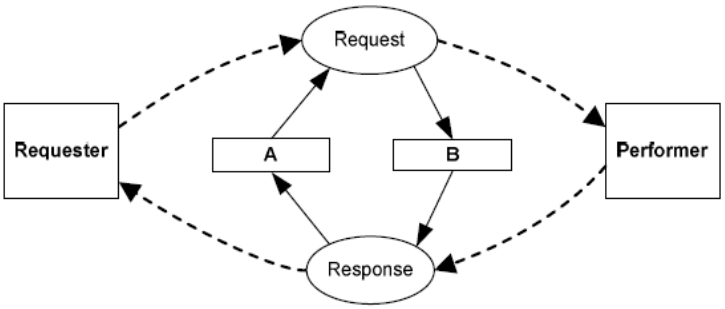


Рисунок 3.15 – Шаблон ітерації

‘Приклад ітераційного шаблону демонструється з циклом взаємодії між двома акторами: клієнтом і книжковим магазином (можна розуміти як комп’ютеризовану систему), який представлений на рисунку 3.16

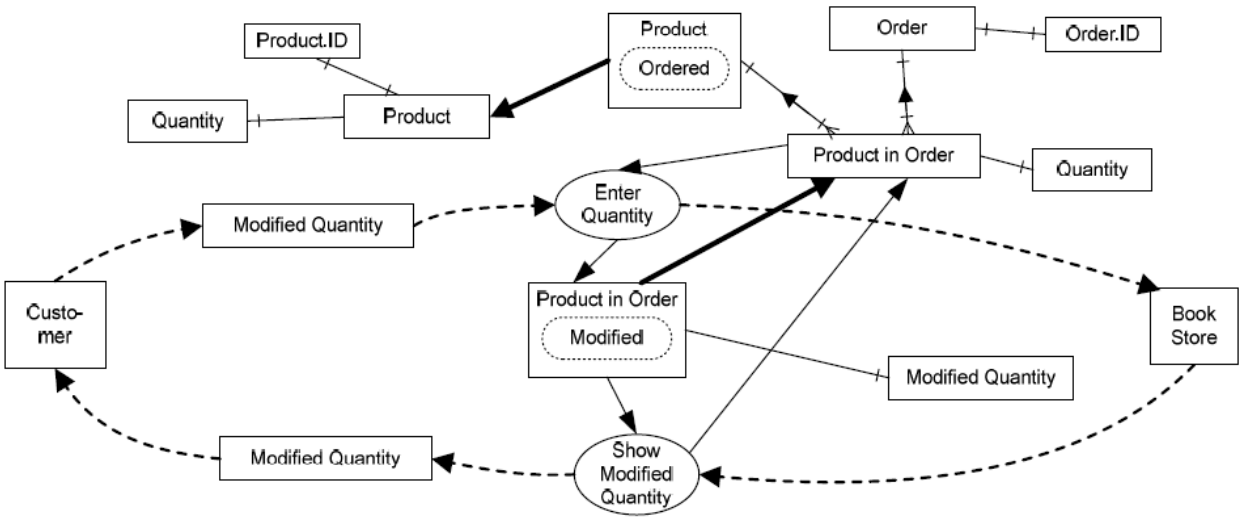


Рисунок 3.16 – Приклад застосування шаблону ітерації

Якщо Клієнт хоче купити кілька книг, він вводить кількість і запускає дію «Введіть кількість». Дія створює об’єкт Товар у замовленні [змінено], який має один введений номер кількості (див. атрибут Змінена кількість). Коли книжковий магазин отримує повідомлення, він викликає дію «Показати змінену кількість» і надсилає потік інформації про змінену кількість клієнту. Атрибут Modified Quantity використовується потоком Modified Quantity у цій дії. Дія «Показати змінену кількість» припиняє дію об’єкта «Продукт у замовленні» [змінено] та створює об’єкт «Продукт у замовленні», який буде введено для дії «Введіть кількість» під час нової взаємодії служби.

Шаблон взаємодії служби: вибір. Шаблон вибору може бути виражений за допомогою композиції двох різних послідовностей між тими самими двома акторами. Вибір представляє два альтернативні результати запиту на послугу, які може вибрати постачальник послуг. Два способи відтворення постачальником послуг є взаємовиключними. Як правило, лише один тип відповіді бажаний для запитувача. Схема вибору графічно представлена на рисунку 3.17.

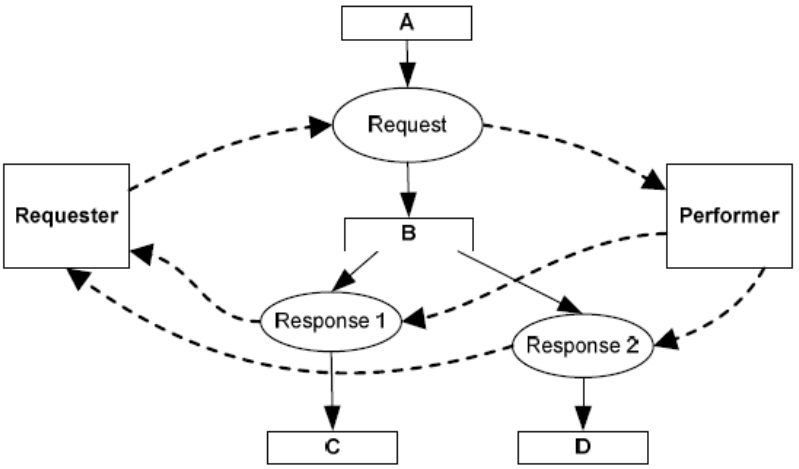


Рисунок 3.17 – Шаблон вибору

Зверніть увагу, що Відповідь 1 і Відповідь 2 є ексклюзивними. Якщо ініціюється відповідь 2, то об’єкт класу B попередньої умови видаляється, і

відповідь 1 не може бути ініційована, і навпаки. Приклад шаблону вибору представлено на рисунку 3.18.

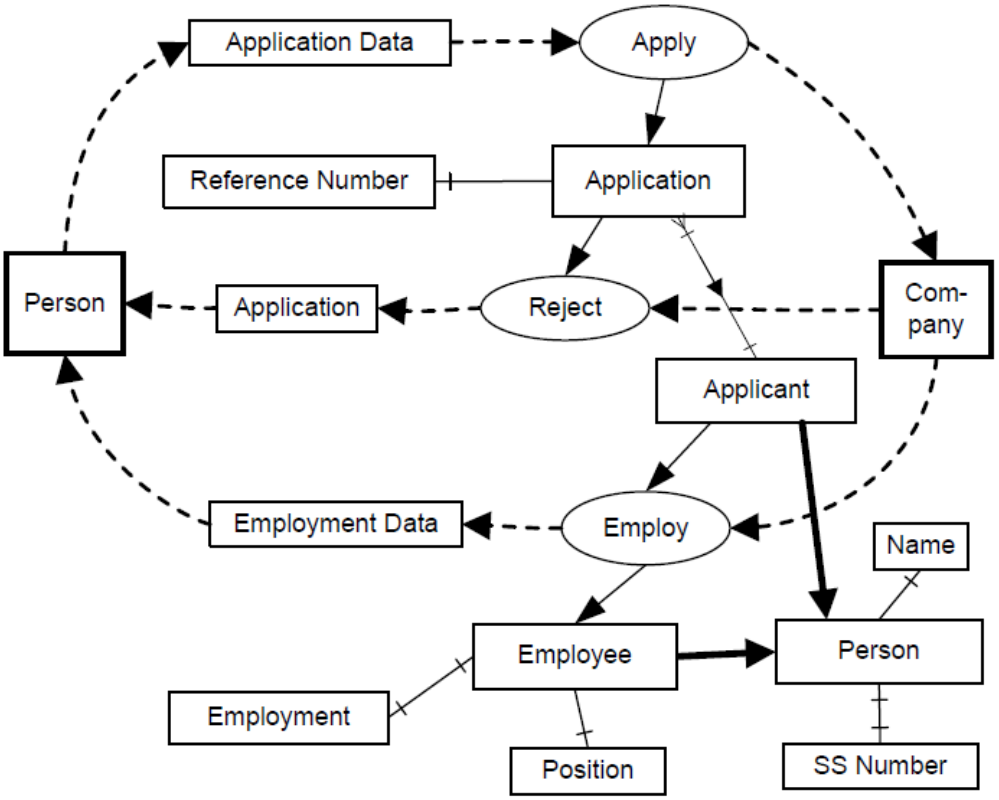


Рисунок 3.18 – Приклад застосування шаблону вибору

Наприклад, якщо Особа (заявник послуг) подає заявку (Подати заявку) на вакансію, надіславши заявку (Потік даних заявки) до Компанії (постачальника послуг), тоді Компанія має два варіанти вибору: найняти Особу або відхилити Заявку. Дія «Застосувати» має створити об’єкт «Програма», пов’язаний з одним Заявником. Кожен об’єкт Applicant може складатися з однієї або кількох програм. У разі дії Employ об’єкт Applicant слід перекласифікувати до Employee. В іншому випадку дія «Відхилити» має припинити роботу об’єкта Application.

Різні комбінації статичних і динамічних залежностей здатні виражати основні шаблони керування робочим процесом. Традиційно в аналізі патернів використовуються лише динамічні моделі. Поділ статичних і динамічних

деталей представлених патернів створює фундаментальні труднощі з двох основних причин:

1. Оскільки статичні деталі повинні бути якимось чином компенсовані за допомогою динамічних конструкцій, кількість основних шаблонів стає більшою, ніж це дійсно необхідно. Іноді їх невеликі відмінності важко зрозуміти і візуально не розпізнати бізнес-експерти.

2. Якщо не враховувати статичні аспекти, то шаблони стануть більш складними для використання з метою моделювання та еволюції бізнес-процесів, що важливо для управління змінами.

Висновки до розділу 3

Отже, в цьому розділі представлено застосування методу сервіс-орієнтованого моделювання, а також застосування пропонованого методу для покращення ефективності процесу сервіс-орієнтованого проектування програмних рішень. Подано результати досліджень, які сприяють систематичному процесу моделювання від прагматичних специфікацій до конкретного дизайну реалізації. Процес сервіс-орієнтованого моделювання для аналізу та проектування інформаційних систем, представлено основними елементами сервіс-орієнтованого методу, а також розроблено основні семантичні конструкції та визначені етапи процесу моделювання. У цьому розділі також розроблено шаблони аналізу: послідовність, синхронізація, ітерація та вибір які були створені з використанням основи сервіс-орієнтованого моделювання.

ВИСНОВКИ

В кваліфікаційній роботі виконано процес імплементація моделей покращення ефективності та оптимальності сервіс-орієнтованого проектування програмних рішень. Розроблено метод моделювання для аналізу та проектування інформаційних систем за межами організаційних і технічних систем, який базується на орієнтації на послуги. Метод сервіс-орієнтованого моделювання для аналізу та проектування інформаційних систем полегшує семантичну інтеграцію статичних і динамічних аспектів специфікацій ІС. Він надає методи графічного моделювання для поєднання прагматичних, семантичних і синтаксичних специфікацій сервісних архітектур. Цей метод відноситься до методів концептуального моделювання ІС інформаційних систем. Пропонований метод сервіс-орієнтованого моделювання складається з трьох частин:

1) прагматична специфікація інформаційних систем;

2) сервіс-орієнтована основа моделювання для аналізу та проектування інформаційних систем;

3) принципи переходу до конкретної реалізації.

Перша частина методу передбачає прагматичний аналіз з точки зору цілей, проблем і можливостей, що пропонує використовувати новий спосіб удосконалення прагматичних сутностей у взаємодії сервісів. Друга частина методу визначає нову основу сервіс-орієнтованого моделювання і забезпечує нотацію моделювання та процес моделювання на рівні моделювання, нейтральному щодо обчислень. Третя частина методу представляє принципи переходу до конкретної реалізації проекту. Представлений метод сервіс-орієнтованого моделювання є еволюційним методом, який забезпечує новий спосіб моделювання та інтеграції підприємства. Це метод графічного дизайну, який дає змогу міркувати про архітектуру системи через організаційні та технічні межі системи.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. D. L. Parnas, ‘‘On the Criteria to be Used in Decomposing Systems into Modules,’’ Communications of the ACM 15,No. 12, 1972.

2. G. Booch, J. Rumbaugh, and I. Jacobson, The UnifiedModeling Language User Guide, Addison-Wesley, Boston 1999.

3. E. Gamma, R. Helm, R. Johnson, and J. Vlissides, DesignPatterns: Elements of Reusable Object-Oriented Software Architecture, Addison-Wesley Professional, Boston 1995.

4. A. Arsanjani, L.-J. Zhang, M. Ellis, A. Allam, and K. Channabasavaiah, ‘‘S3: A Service-Oriented Reference Architecture,’’ IEEE IT Professional 9, No. 3, 2007.

5. A. Arsanjani, L.-J. Zhang, M. Ellis, A. Allam, and K.Channabasavaiah, ‘‘Design an SOA Solution Using a Reference Architecture,’’ IBM eveloperWorks, IBM IBM SYSTEMS JOURNAL, VOL 47, NO 3, 2008.

6. O. Zimmerman, P. Krogdahl, and C. Gee, ‘‘Elements ofService-Oriented Analysis and Design,’’ IBM developerWorks, IBM Corporation (2004).

7. A. Arsanjani and K. Holley, ‘‘The Service Integration Maturity Model: Achieving Flexibility in the Transformation to SOA,’’ Proceedings of the IEEE International Conference on Services Computing, Chicago 2006.

8. A. Arsanjani, ‘‘Service-oriented Modeling and Architecture: How to Identify, Specify and Realize Services for Your SOA,’’ IBM developerWorks, IBM Corporation 2004.

9. R. Varadan, K. Channabasavaiah, S. Simpson, K. Holley,and A. Allam, ‘‘Increasing Business Flexibility and SOAAdoption Through Effective SOA Governance,’’ IBM Systems Journal 47, No. 3, 473–488 2008.

10. B. W. Boehm, ‘‘A Spiral Model of Software Developmentand Enhancement,’’ IEEE Computer 21, No. 5, 61–72, 1988.

11. L. Bass, P. Clements, and R. Kazman, Software Architecture in Practice, Addison-Wesley Professional, Boston 1998.

12. L.-J. Zhang, A. Arsanjani, A. Allam, D. Lu, and Y.-M.Chee, Proceedings of the IEEE International Conference on Services Computing,‘‘Variation-Oriented Analysis for SOA Solution Design,’’ Salt Lake City, UT, 2007.

13. K. Levi and A. Arsanjani, ‘‘A Goal-driven Approach to Enterprise Component Identification and Specification,’’ Communications of the ACM 45, No. 10, 45–52 2002.

14. G. Sauter, B. Mathews, M. Selvage, and E. Lane, ‘‘Information Service Patterns, Part 1: Data Federation Pattern,’’ IBM developerWorks, IBM Corporation 2006.

15. M. Keen, O. Adinolfi, S. Hemmings, A. Humphreys, H. Kanthi, and A. Nottingham, ‘‘Patterns: SOA with an Enterprise Service Bus in WebSphere Application Server V6,’’ IBM Redbooks 2005.

16. A. Arsanjani, ‘‘Rule Object 2001: A Pattern Language for Adaptive and Scalable Business Rule Construction,’’ Proceedings of the 8th Conference on Pattern Languages of Programs, Monticello, IL 2001.

17. D. F. Bacon, P. Cheng, and V. T. Rajan. ‘‘A Real-Time Garbage Collector with Low Overhead and Consistent Utilization,’’ ACM SIGPLAN Notices, 38(1):285--298, Jan. 2003.

18. S. Bodhare, ‘‘Optimizing Service Infrastructures’’, ttp://blogs.ittoolbox.com/eai/
optimization/archives/optimizing-serviceinfrastructures-3928

19. D. Chappel, Enterprise Service Bus: Theory in Practice, O’reilly Media, 2004.

20. Y. Chen and W.T. Tsai, Introduction to Programming Languages: Programming in C, C++, Scheme, Prolog, C#, and SOA, second edition, Kendall/Hunt Publishing, 2006.

21. X. Fu, T. Bultan, and J. Su, “Formal Verification of E-Services and Workflows,” Proc. Workshop on Web Services, E-Business, and the Semantic Web, LNCS 2512, Springer-Verlag, 2002.

22. O. Goh, Y.-H. Lee, Z. Kaakani, and E. Rachlin. ‘‘A Schedulable Garbage Collection for Embedded Applications in CLI,’’ The 11th International Conference on Real-Time and Embedded Computing Systems and Applications (RTCSA05), July 2005.

23. J. Helander and S. B. Sigurdsson, “Self-Tuning Planned Actions Time to Make Real-Time SOAP Real,” The 8th IEEE International Symposium on Object-Oriented Real-Time Distributed Computing 2001.

24. N. Kavantzas, D. Burdett, T. Fletcher, and Y. Lafon, “Web Services Choreography Description Language”, Version 1.0, W3C Working Draft 17 Dec. 2004.

25. D. Kim, Y. H, Lee, M. F. Younis, “SPIRITμKernel for Strongly Partitioned Real-Time Systems,” The 6th International Conference on Real-Time and

26. Embedded Computing Systems and Applications (RTCSA 2000), pp. 73-80

27. C. Kohlhoff and R. Steele, "Evaluating SOAP for High Performance Business Applications: Real-Time Trading Systems", Proc. of WWW'03, 2003.

28. Frank Leymann, Web Services Flow Language, Version 1.0, May http://www-306.ibm.com/ software/ solutions/webservices/pdf/WSFL.pdf, 2001.

29. R. Paul, “DoD Towards Software Services”, Tenth IEEE International Workshop on Objectoriented Real-time Dependable Systems (WORDS 05), February 2005.

30. M. P. Singh, M. N. Huhns, Service-Oriented Computing, John Wiley & Sons, 2005.

31. W.T. Tsai, Ray A. Paul, Bingnan Xiao, Zhibin Cao, Yinong Chen, "PSML-S: A Process Specification and Modeling Language for Service Oriented

Computing", The 9th IASTED International Conference on Software Engineering and Applications (SEA), Phoenix, November 2005.

32. W.T. Tsai, C. Fan, Y. Chen, and R. Paul, “DDSOS: A Dynamic Distributed Service-Oriented Simulation Framework”, in Proceedings of 39th Annual Simulation Symposium (ANSS), Huntsville, AL, April 2006.

33. O. Zimmermann, S. Milinski, M. Craes, F. and Oellermann, "Second eneration Web ServicesOriented Architecture in Production in the Finance Industry", OOPSLA’04, Oct. Vancouver, 2004.

34. I. Nadareishvili, R. Mitra, M. Mclarty, and M. Amundsen, Microservice Architecture. O’Reilly Media, 2016.

35. P. Di Francesco, I. Malavolta, and P. Lago, “Research on architecting microservices: Trends, focus, and potential for industrial adoption,” in Proc. of the Int. Conf. on Software Architecture (ICSA). IEEE, 2017.

36. T. Erl, Service-Oriented Architecture (SOA) Concepts, Technology and Design. Prentice Hall, 2005.

37. C. Pahl and P. Jamshidi, “Microservices: A systematic mapping study,” in Proc. of the 6th Int. Conf. on Cloud Computing and Services Science (CLOSER), 2016.

38. M. P. Papazoglou and W.-J. Van Den Heuvel, “Service oriented architectures: approaches, technologies and research issues,” The VLDB journal, vol. 16, no. 3, pp. 389–415, 2007.

39. A. Rodrigues Da Silva, “Model-driven engineering: A survey supportedby the unified conceptual model,” Computer Languages, Systems andStructures, vol. 43, pp. 139–155, 2015.

40. D. Ameller, X. Burgués, O. Collell, D. Costal, X. Franch, and M. P. apazoglou, “Development of service-oriented architectures using modeldriven evelopment: A mapping study,” Information and Software Technology, vol. 2, no. 1, pp. 42–66, 2015.

41. F. Rademacher, S. Sachweh, and A. Zündorf, “Differences betweenmodel-driven development of service-oriented and microservice

architecture,” in Proc. of the Int. Workshop on Architecting with MicroServices (AMS) co-located with ICSA. IEEE, 2017.

42. Object Management Group, Model Driven Architecture (MDA) Guide,OMG Std., Rev. 2.0, 2014.

43. B. Combemale, R. B. France, J.-M. Jézéquel, B. Rumpe, J. Steel, andD. Vojtisek, Engineering Modeling Languages. CRC Press, 2017.

44. R. France and B. Rumpe, “Model-driven development of complex software: A research roadmap,” Proc. of the 2007 Workshop on Future of Software Engineering (FOSE), 2007.

45. J. Whittle, J. Hutchinson, and M. Rouncefield, “The state of practicein model-driven engineering,” IEEE software, vol. 31, no. 3, pp. 79–85,2014.

46. S. Sendall and W. Kozaczynski, “Model transformation: The heart andsoul of model-driven software development,” IEEE Software, vol. 20,no. 5, pp. 42–45, 2003.

47. H. Kreger and J. Estefan, “Navigating the soa open standards landscapearound architecture,” OASIS, OMG and The Open Group, 2009.

48. J. Hutchinson, J. Whittle, M. Rouncefield, and S. Kristoffersen, “Empirical assessment of mde in industry,” in Proc. of the 33rd Int. Conf.on Software Engineering (ICSE). IEEE, 2011.

49. A. Balalaie, A. Heydarnoori, and P. Jamshidi, “Migrating to cloud-native architectures using microservices: an experience report,” in Workshop Proc. of the 4th Europ. Conf. on Service-Oriented and Cloud Computing (ESOCC). Springer, 2015.

50. M. Richards, Microservices vs. Service-Oriented Architecture. O’Reilly Media, 2015.

метадані

Заголовок

Імплементація моделей покращення ефективності та оптимальності сервіс-орієнтованого проектування програмних рішень

Автор

Габурак М.М. Науковий керівник / Експерт

підрозділ

King Danylo University

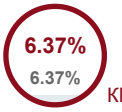
Тривога

У цьому розділі ви знайдете інформацію щодо текстових спотворень. Ці спотворення в тексті можуть говорити про МОЖЛИВІ маніпуляції в тексті. Спотворення в тексті можуть мати навмисний характер, але частіше характер технічних помилок при конвертації документа та його збереженні, тому ми рекомендуємо вам підходити до аналізу цього модуля відповідально. У разі виникнення запитань, просимо звертатися до нашої служби підтримки.

Заміна букв	ß	20
Інтервали	A→	0
Мікропробіли	·	0
Білі знаки	ß	0
Парафрази (SmartMarks)	a	63

Обсяг знайдених подібностей

Коефіцієнт подібності визначає, який відсоток тексту по відношенню до загального обсягу тексту було знайдено в різних джерелах. Зверніть увагу, що високі значення коефіцієнта не автоматично означають плагіат. Звіт має аналізувати компетентна / уповноважена особа.



КП 1

КП 2

25
Довжина фрази для коефіцієнта подібності 2

16427
Кількість слів

130045
Кількість символів

Подібності за списком джерел

Нижче наведений список джерел. В цьому списку є джерела із різних баз даних. Колір тексту означає в якому джерелі він був знайдений. Ці джерела і значення Коефіцієнту Подібності не відображають прямого плагіату. Необхідно відкрити кожне джерело і проаналізувати зміст і правильність оформлення джерела.

10 найдовших фраз — Колір тексту

ПОРЯДКОВИЙ НОМЕР	НАЗВА ТА АДРЕСА ДЖЕРЕЛА URL (НАЗВА БАЗИ)	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)	
1	Analysis of Service-oriented Modeling Approaches for Viewpoint-specific Model-driven Development of Microservice Architecture Sabine Sachweh,Florian Rademacher,Albert Zündorf;	125	0.76 %
2	https://dl.acm.org/doi/10.1147/sj.473.0377	47	0.29 %
3	https://dl.acm.org/doi/10.1147/sj.473.0377	42	0.26 %
4	http://repository.ukd.edu.ua/bitstream/handle/123456789/391/%D0%9F%D0%B0%D1%85%D0%BE%D0%BB%D1%8C%D1%87%D1%83%D0%BA%20%D0%9E.%D0%A0.%20%D0%B4%D0%B8%D0%BF%D0%BB%D0%BE%D0%BC%D0%BD%D0%B0.pdf?sequence=1	35	0.21 %