КВАЛІФІКАЦІЙНА РОБОТА

Група ІПЗс-20

Сергієвич Н.Р.

2024

ЗВО УНІВЕРСИТЕТ КОРОЛЯ ДАНИЛА

Факультет суспільних та прикладних наук

Кафедра інформаційних технологій

на правах рукопису

Сергієвич Назар Романович

УДК 004.4

Впровадження адаптивних моделей у задачах децентралізованого навчання

Спеціальність 121 – «Інженерія програмного забезпечення»

Кваліфікаційна робота на здобуття кваліфікації бакалавр

Нормоконтроль

_____Стисло О.В.

(підпис, дата, розшифрування підпису)

Студент

_____ Сергієвич Н.Р.

(підпис, дата, розшифрування підпису)

Допускається до захисту

Завідувач кафедри

_____к.т.н., доц. Ващишак С.П.

(підпис, дата, розшифрування підпису)

Керівник роботи

_____ к.ф-м.н., доц. Бойчук А.М.

(підпис, дата, розшифрування підпису)

Івано-Франківськ – 2024

ЗВО УНІВЕРСИТЕТ КОРОЛЯ ДАНИЛА

Факультет суспільних та прикладних наук

Кафедра інформаційних технологій

Освітній ступінь: «бакалавр»

Спеціальність: 121 «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____

«____» _____ 2024 року

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Сергієвичу Назару Романовичу

(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи
Впровадження адаптивних моделей у задачах децентралізованого навчання

керівник роботи:
Бойчук Андрій Михайлович, кандидат фізико-математичних наук, доцент

затверджена наказом вищого навчального закладу від « 12 » березня 2024 року
№ 19/1

2. Термін подання студентом роботи 05.06.2024

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

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

1. Аналіз складових адаптивності інформаційних систем та рішень
2. Дослідження методів виведення та концепції адаптивних сервісів
3. Моделювання архітектур керування

5. Дата видачі завдання 14.03.2024

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 складових адаптивності інформаційних систем та рішень	20.03.2024	Виконано
2.	Дослідження методів виведення та концепції адаптивних сервісів на основі даних	26.03.2024	Виконано
3.	Моделювання архітектур керування	09.04.2024	Виконано
4.	Структуровані моделі адаптивного навчання на основі даних	19.04.2024	Виконано
5.	Формування висновків	30.04.2024	Виконано
6.	Оформлення пояснювальної записки	10.05.2024	Виконано
7.	Оформлення графічного матеріалу та підготовка до захисту роботи	20.05.2024	Виконано

Студент _____ Сергієвич Н.Р.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Бойчук А.М.
(підпис) (прізвище та ініціали)

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

37	Проектний розмір – контроль	64	Інваріанти exactlyOneMaster
38	Проектний розмір – ідентифікація	56	DECOR reasoning framework
39	Проектний розмір – адаптація виконання	64	Приклад шаблонів MAPE-K
45	Значення для Hierarchical Agglomerative Clustering	66	Середовище спільного моделювання
45	Bootstrap аналіз	68	Представлення передбачуваної архітектури
47	Розміри проекту SAS на основі навчання		

АНОТАЦІЯ

Кваліфікаційна робота присвячена дослідженню та застосуванню моделей адаптивності в задачах децентралізованого навчання шляхом розробки децентралізованого самоадаптивного підходу до проектування сервісу, основними функціями якого є: самоадаптація, онлайн-навчання, усвідомлення QoS, масштабованість та стійкість.

В першому розділі виконано аналіз складових адаптивності інформаційних систем та рішень, описано моделі процесів адаптивності інформаційних систем. Проведено аналіз підходів та методів реалізації децентралізованої адаптації, розглянуто методи навчання в самоадаптивних системах та інженерні самоадаптивні системи.

В другому розділі наведені методи виведення та концепції адаптивних сервісів на основі даних, досліджено парадигму “Everything as a Service”, здійснено проектування адаптивних навчальних систем на основі даних. Наведені вимоги до проектування адаптивної системи, здійснено проектування та оцінка архітектур керування, представлені методології, керовані даними для децентралізованої системи.

В третьому розділі виконаний аналіз та моделювання децентралізованих адаптивних систем на основі даних. Наведена структура міркування на основі моделі, проведено моделювання архітектур керування і наведено архітектурне керування для децентралізованого навчання.

КЛЮЧОВІ СЛОВА: АДАПТИВНІ СИСТЕМИ, КОМПОНЕНТИ СЕРВІСУ, АРХІТЕКТУРА ДЕЦЕНТРАЛІЗОВАНОГО КЕРУВАННЯ, ВІДМОВОСТІЙКІСТЬ, МАШИННЕ НАВЧАННЯ, СТРУКТУРА МІРКУВАННЯ.

SUMMARY

The qualification work is dedicated to research and implementation of adaptability models in decentralized tasks by developing a decentralized self-adaptive approach to service design, the main functions of which are: self-adaptation, online learning, QoS awareness, scalability and sustainability

The first section analyzes the components of the adaptability of information systems and solutions, describes the models of the processes of the adaptability of information systems. An analysis of the approaches and methods of implementation of decentralized adaptation was carried out, methods of learning in self-adaptive systems and engineering self-adaptive systems were considered.

In the second chapter, the methods of derivation and concepts of adaptive services based on data are presented, the paradigm "Everything as a Service" is studied, and the design of adaptive educational systems based on data is carried out. The requirements for the design of an adaptive system are presented, the design and evaluation of control architectures are performed, and data-driven methodologies for a decentralized system are presented.

In the third section, the structured model of decentralized adaptive learning is implemented on the basis of data, the learning process in decentralized adaptive systems is investigated. The structure of reasoning based on the model is presented, the simulation of control architectures is carried out, and the control architecture for decentralized learning is given.

KEY WORDS: ADAPTIVE SYSTEMS, SERVICE COMPONENTS, DECENTRALIZED CONTROL ARCHITECTURE, FAILURE TOLERANCE, MACHINE LEARNING, REASONING STRUCTURE.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

QoS – Quality of Service

SAS – Self-adaptive systems

ML – Machine Learning

CAS – Collective Adaptive Systems

MAS – Multi-Agent Systems

SOS – Self-Organizing Systems

ASPLe – Autonomic Software Product Lines engineering

XaaS – everything-as-a-service

RL – Reinforcement Learning

DWL – Distributed W-Learning

MARL – Multi-Agent Reinforcement Learning

eARF – extended Architectural Reasoning Framework

dQAS – domain Quality Attribute Scenarios

DEVS – Discrete Event System Specification

CAME – Control Architecture Modeling Environment

CoSE – Co-Simulation Environment

dRS – domain Responsibility Structure

OCL – Object Constraint Language

DEVS – Discrete EVent System

HAC – Hierarchical Agglomerative Clustering

ВСТУП

Актуальність теми. Більшість версій індуктивного узагальнення представляють один із двох підходів. Перші фокусуються на відносно загальні, незалежні від знань статистичні механізми висновку, засновані на подібності: асоціації, кореляції або інші статистичні показники. Такий підхід призвів до успішної мат моделі узагальнення в лабораторних завданнях, але не враховує багато важливих явищ навчання та міркування в складних сферах реального світу. Другий підхід має на меті охопити більшу сферу висновків, звертаючись до складних предметно-спеціальних представлень знань, або інтуїтивних теорій. Інтуїтивна теорія може мислиться як система пов'язаних понять разом з набором причинно-наслідкових правил, структурних обмежень, принципів пояснення, які керують індуктивним висновком в певному домені. Проте теоретичні підходи до індукції, як відомо, важко формалізувати, особливо в термінах, які роблять кількісні прогнози про поведінку або які можна зрозуміти з точки зору раціонального статистичного висновку.

Ми будемо аргументувати альтернативний підхід, де структуровані знання та статистичні висновки взаємодіють а не конкурують, дозволяючи нам спиратися на знання обох доменів. Ми розглядаємо індукцію як форму Байєса для статистичного висновку на основі структурованих імовірнісних моделей світу. Ці моделі можна розглядати як імовірнісні версії інтуїтивних теорій або схем, фіксуючи знання про домен, який дозволяє індуктивне узагальнення з розріджених даних. Це підхід став можливим лише в останні роки, як досягнення в області штучного інтелекту і статистики надали необхідні інструменти для формалізації інтуїтивних теорій та статистичних висновків на основі теорії. Такий вплив є двохнаправленим, оскільки ці байєсівські когнітивні моделі призвели до нових алгоритмів машинного навчання. Поведінкові дослідження індуктивного узагальнення можливо, розпочалися з вивчення категорійного навчання. Базове експериментальне завдання представляє набір об’єктів або

візуальних стимулів, а також вербальний ярлик що застосовується до підмножини об’єктів. Ці завдання штучного навчання категорій абстрагують суть проблеми, з якою стикаються при вивченні ключових слів для виділених речей і формальних моделей категорійного навчання. Вони зазвичай покладаються на висхідну статистику загального призначення механізмів, що є або явно ймовірнісні, або оформлені в термінах подібності або асоціації.

Ці моделі передбачають відносно прості поняття категорій та як мітки пов’язані з категоріями: наприклад, кожен об'єкт належить до однієї категорії, і кожен label вибирає унікальну категорію, тому кожен об’єкт отримує рівно одну мітку.

Мета і завдання дослідження. **Метою класифікаційної роботи** є дослідження підходів та методів реалізації децентралізованої адаптації, методів навчання в самоадаптивних системах.

Для досягнення поставленої мети необхідно розв’язати такі задачі: 1) виконати аналіз складових адаптивності інформаційних систем та рішень; 2) представити методи виведення та концепції адаптивних сервісів на основі даних; 3) виконати проектування адаптивних навчальних систем на основі даних; 4) здійснити моделювання архітектур керування для адаптивних систем.

Об'єктом дослідження є процеси децентралізованого навчання. Це включає методи та технології, що використовуються для навчання моделей штучного інтелекту та машинного навчання в розподілених системах, де дані та обчислювальні ресурси розміщені на різних вузлах мережі.

Предметом дослідження є адаптивні моделі та їх впровадження у децентралізоване навчання. Це передбачає вивчення та реалізацію підходів для динамічного налаштування моделей на різних вузлах мережі з урахуванням змінних умов, таких як обсяг даних, пропускна здатність мережі, обчислювальні потужності та інші фактори.

Методи дослідження базуються на використанні методів моделювання індуктивного виведення на множині знань, методів та засобів адаптивного навчання, методів та інструментів теорії категорій.

Практичне значення одержаних результатів полягає в розробці мотиваційного сценарію, що описує клас адаптивних додатків, опрацьовано і виявлено набір вимог, які повинні бути адресовані розробці цих програм, а також зіставлено визначені вимоги з простором дизайну SAS, з метою дослідження розміру дизайну, що пов’язано з процесами самоадаптації.

Структура. Кількість розділів – 3. Обсяг основної частини – 63 сторінки. Список використаних джерел містить – 23 позиції.

РОЗДІЛ 1. АНАЛІЗ СКЛАДОВИХ АДАПТИВНОСТІ ІНФОРМАЦІЙНИХ СИСТЕМ ТА РІШЕНЬ

1.1 Опис досліджуваної наукової проблеми в області побудови адаптивних інформаційних систем

Передбачається, що комп'ютерні середовища майбутнього будуть наповнені безліччю розподілених, повсюдних і пов'язаних речей реального світу, які співпрацюють одна з одною, щоб надати безмежні можливості промисловості та суспільству – наприклад, розумні міста, інтелектуальні транспортні системи та промислове виробництво. У цьому випадку додаток можна розглядати як мережеву систему, де велика відкрита колекція автономних і гетерогенних служб динамічно взаємодіє одна з одною, щоб надати користувачам багаті функціональні можливості. Дійсно, додатки динамічно з'являються як збірки цікавих послуг, доступних у будь-який момент часу.

Цей клас програм, ймовірно, характеризується проблемами якості обслуговування (QoS - Quality of Service), що стосуються таких атрибутів, як час відгуку, надійність, доступність і вартість. Насправді висока динамічність середовища, що характеризується приєднанням і виходом послуг із мережі та зміною їхніх атрибутів якості, вносить невизначеності, які, у свою чергу, можуть як порушити правильну функціональність, так і завдати шкоди QoS системи. У цьому контексті головна критична проблема полягає в тому, щоб надати системі здатність бути стійкою до цих невизначеностей з мінімальним втручанням людини або без нього, щоб полегшити збірку програмних додатків.

Самоадаптація була запропонована як життєздатний механізм для підтримки стійкості шляхом усунення невизначеностей під час виконання та підтримки цілей системи. Самоадаптивні системи (SAS - Self-adaptive systems) концептуально організовані як керована система, яка реалізує основну функціональність і керуюча система, яка реалізує логіку адаптації за допомогою

архітектури керування, складеної відповідно до добре встановленої моделі MAPE-K [2] з компонентами Monitor (M), Analyze (A), Plan (P) і Execute (E), а також Knowledge (K) який зберігає відповідну інформацію для інших компонентів.

Важлива проблема в самоадаптивних системах стосується реалізації логіки адаптації, яка встановлює, «коли» і «як» система повинна адаптуватися. Передбачення всіх потенційних змін навколишнього середовища, з якими система може зіткнутися під час виконання та має самоадаптуватися у відповідь, у більшості випадків є нездійсненним. У цьому контексті методи машинного навчання (ML - Machine Learning) надають можливість самостійно навчатися, вдосконалюватися та приймати рішення для досягнення стійкості [3]. Однією з ключових особливостей, які сприяють застосуванню машинного навчання в самоадаптивних програмних системах, є те, що ці методи в основному мають справу з невизначеністю, тоді як традиційне програмне забезпечення по суті приховує невизначеність [4]. Однак, незважаючи на дедалі більше впровадження методів машинного навчання, немає жодної сукупності знань про найкращі практики, які можна застосувати для розроблення логіки адаптації на основі навчання, що залишає дослідників і практиків без вказівок щодо того, як пом’якшити цю постійну проблему.

По-друге, коли системи великі та розподілені, централізації архітектури керування може бути недостатньо для реалізації логіки адаптації. Підтримка узгодженості глобальних знань, своєчасний їх аналіз і розподіл рішень щодо адаптації через централізований компонент у розподілених системах є неможливим. Децентралізація архітектури керування та розгортання кількох скоординованих циклів зворотного зв’язку MAPE-K є інтуїтивно зрозумілим рішенням [5]. Однак розробка та оцінка децентралізованих архітектур керування є грандіозним викликом: інженери повинні розглянути багато різних і взаємозалежних параметрів дизайну, щоб розробити архітектуру керування, здатну вчасно виконувати правильні адаптації та змусити систему відповідати поставленим цілям. Щоб знайти рішення цієї проблеми, потрібен підхід, який

підтримує діяльність з проектування та реалізації з можливостями оцінки архітектурних рішень, аргументації та прийняття обґрунтованого вибору дизайну.

І нарешті, хоча децентралізована самоадаптація, заснована на навчанні, є ключовим чинником стійких сервісних вузлів, ефективне проектування цих систем, безсумнівно, є надзвичайно складним. З одного боку, відсутність найкращих практик для вибору моделей і технік навчання перешкоджає конкретному застосуванню машинного навчання для реалізації логіки адаптації. З іншого боку, невід'ємна складність проектування та оцінки самоадаптивних систем бентежить інженерів і вимагає системного підходу, який підтримує міркування щодо функціональних і нефункціональних аспектів системи, яка буде створена.

В даній роботі розглядаються ці критичні проблеми в три етапи. По-перше, вивчаються, класифікуються та агрегуються знання з досліджень, пов’язаних із самоадаптивними системами, використовуючи методи навчання як засоби вирішення невизначеності. Потім представляється структура аргументації, яка підтримує діяльність з проектування та впровадження з можливостями оцінки архітектурних рішень.

Нарешті, використовуючи рішення, розроблені на основі вирішення вищезгаданих проблем, пропонується підхід до побудови та підтримки протягом тривалого часу стійкого набору послуг, які разом здатні надавати якісні послуги.

1.2 Моделі процесів адаптивності інформаційних систем

Самоадаптація дозволяє сучасним великомасштабним програмним системам - наприклад, сервіс-орієнтованим системам, кіберфізичним системам мати можливість автономно пом'якшувати невизначеність життєвого циклу системи [5]. Прикладами таких невизначеностей є динамічна доступність ресурсів програмного забезпечення та динамічна якість (наприклад, QoS), яку забезпечують такі ресурси.

У цій роботі ми спираємося на класичний підхід до реалізації самоадаптації, де система розділена на керовану систему, тобто частину системи, що підлягає адаптації, та керуючу систему, яка містить логіку адаптації. Система керування відповідає за реалізацію циклу зворотного зв'язку, який під час виконання контролює керовану систему та адаптує її відповідно до умов, що виникають, таким чином зменшуючи невизначеності.

Далі ми обговорюємо еталонну модель, яку ми використовуємо в нашому дослідженні, тобто еталонну модель MAPE-K.

1.2.1. Еталонна модель MAPE-K

SAS концептуально організовані як керована система, яка реалізує основні функціональні можливості, і система керування, яка реалізує логіку адаптації через архітектуру керування. У еталонній моделі MAPE-K, представленій IBM, система управління складається з 5 внутрішніх компонентів: моніторинг, аналіз, планування, виконання та знання, організовані як цикл зворотного зв'язку MAPE-K [6].

У цій моделі (рис. 1.1) середовище представляє частину зовнішнього світу, де знаходиться SAS, і забезпечує контекст для керованої системи. Керована та керуюча системи з'єднані через датчики та виконавчі механізми для сприйняття та адаптації, відповідно, керованої системи, коли це необхідно. Компоненти MAPE-K системи керування виконують наступні завдання:

1. Monitor: компонент монітора відповідає за збір і агрегування даних від датчиків.
2. Analyze: компонент аналізу відповідає за аналіз даних, зібраних із компонента монітора, щоб визначити, коли потрібна адаптація.
3. Plan: компонент плану відповідає за планування дій, необхідних для адаптації керованої системи.

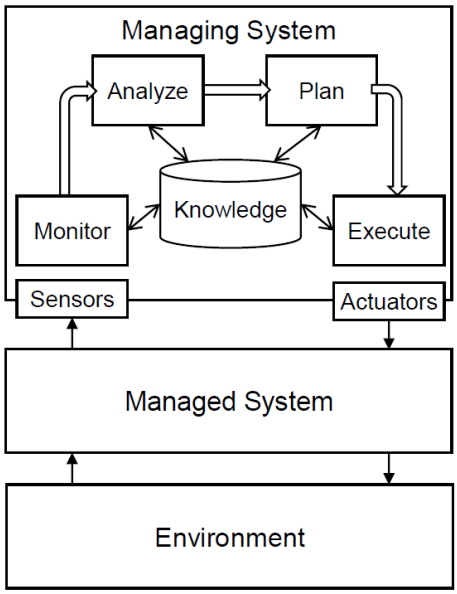


Рисунок 1.1 – Еталонна модель MAPE-K для самоадаптивних систем

4. Execute: компонент відповідає за виконання плану адаптації керованої системи.

5. Knowledge: компонент знань підтримує відповідні знання (наприклад, історичні дані керованої системи), які використовуються іншими компонентами для виконання своїх завдань.

1.2.2. Децентралізовані самоадаптивні системи

Здійсненність ефективної самоадаптації в розподілених системах за допомогою централізованого керування є сумнівною [6]. Дійсно, використання єдиного автономного менеджера, тобто єдиного замкнутого циклу MAPE-K, для моніторингу розподіленої системи та дії на системні ресурси зазвичай залежить від нереалістичних припущень. Ці припущення включають, але не обмежуються цим, здатність автономного менеджера отримувати послідовні та повні глобальні знання, а також його здатність обробляти цю інформацію, аргументувати та розподіляти рішення вчасно.

Надання можливостей самоадаптації кожному компоненту розподіленої системи може вирішити проблеми, з якими стикається централізоване управління [7].

Дійсно, адаптація має бути колективною: кілька розподілених компонентів повинні адаптуватися таким чином, щоб усунути критичну невизначеність під час виконання, зберігаючи при цьому переваги спільної взаємозалежності.

У літературі описуються процеси які мають справу з адаптацією в децентралізованих системах, тобто самоадаптивних системах [8], колективних адаптивних системах (CAS - collective adaptive systems) [9], мультиагентних системах [10] (MAS - multi-agent systems) і системах, що самоорганізуються [11] (SOS - self-organizing systems). Загальні припущення в цих спільнотах полягають у тому, що різні компоненти (агенти):

– можуть взаємодіяти з іншими компонентами/агентами прямо або опосередковано;

– індивідуально не володіє загальносистемними знаннями;

– може приймати рішення на основі колективних або сукупних знань деяких схожих систем.

Абстрагуючись від конкретних спільнот, ми зосереджуємося на цих властивостях і досліджуємо системи, які за своєю суттю є розподіленими, автономними та спрямовані на роботу в дуже динамічних середовищах.

Тому в цій кваліфікаційній роботі, щоб уникнути двозначності, ми будемо використовувати термін «децентралізована SAS» для позначення класу систем, який нас цікавить.

1.3 Аналіз підходів та методів реалізації децентралізованої адаптації

У цьому розділі ми обговорюємо кілька ключових підходів, які тісно пов’язані з даним дослідженням. Спочатку ми коротко підсумуємо існуючі опитування щодо методів машинного навчання для реалізації децентралізованої адаптації. Потім ми досліджуємо існуючі роботи, які вирішують проблему створення SAS. Нарешті, ми розглянемо існуючі підходи до складання послуг, запропоновані в літературі.

1.3.1. Методи навчання в самоадаптивних системах

Методи машинного навчання були визнані ключовими засобами для реалізації та вдосконалення логіки адаптації в SAS, наприклад, шляхом вивчення нових правил [12]. Таким чином, протягом останніх років ML все більше використовувався для вирішення проблеми прогнозування умов виконання та вирішення невизначеностей у SAS [13].

Однак, незважаючи на те, що навчання було досліджено для реалізації логіки адаптації в багатьох самоадаптивних програмах, наприклад, [14], немає жодного обсягу знань про найкращі практики в децентралізованому SAS на основі навчання.

Фактично, що стосується робіт, пов’язаних з навчанням з само адаптивних системах, існуючі опитування щодо навчання обмежуються переглядом альтернативних методів, запропонованих у цьому контексті [15]. Зокрема, переваги використання самоорганізації для вирішення складних проблем дослідження в MAS вивчаються в [16]. Кооперативні та конкурентоспроможні багатоагентні методи досліджуються в [17], тоді як у [18] аналізуються відмінності між цими методами. Навпаки, у цій роботі ми зосереджуємось на розумінні стану практики розробки децентралізованих SAS на основі навчання.

Методи машинного навчання можуть доповнювати контур керування та процес розробки в SAS. Однак кожна техніка машинного навчання має свої особливості, що базуються на потребі в знаннях предметної області та її перевагах. Кожну техніку можна використовувати по-різному залежно від її характеристик. На жаль, техніка машинного навчання зазвичай вибирається завдяки її можливостям. Наприклад, машина опорних векторів стає найбільш використовуваною технікою через її високу точність результатів за допомогою її функцій ядра. Зазвичай він забезпечує найкращий прогноз порівняно з іншими класифікаторами машинного навчання. Потрібні рекомендації щодо впровадження техніки машинного навчання для покращення процесу адаптації в SAS.

Рисунок 1.2 – Запропонована початкова таксономія для вибору техніки машинного навчання в SAS

Таким чином, наша перша рекомендація щодо напрямку дослідження полягає в необхідності таксономії, яка повинна забезпечувати механізм вибору техніки на основі проблем предметної області та характеристик машинного навчання. На основі елементів даних ми запропонували початкову таксономію для вибору техніки ML у SAS, як показано на рисунку 1.2.

1.3.2. Інженерні самоадаптивні системи

Встановлення шляхів від простору проектування до впровадження системи за допомогою архітектурних шаблонів, фреймворків і проміжного програмного забезпечення визнано одним із головних дослідницьких прогалин для SAS [19]. З цією метою нещодавно було запропоновано кілька різних підходів.

В [3] пропонують інженерну методологію під назвою Autonomic Software Product Lines engineering (ASPLe). ASPLe надає розробникам підтримку процесів для реалізації ліній продуктів SAS із повторним використанням на рівні системи керування. Ця теза використовує ASPLe і містить міркування щодо децентралізованої самоадаптації.

Архітектурна структура Rainbow [12] використовує архітектурні моделі для обґрунтування динамічної поведінки системи та дозволяє чітко вказувати багаторазові стратегії адаптації для багатьох проблем у централізованій SAS. Зокрема, використовуючи підхід зовнішнього керування, структура спрямована на моніторинг властивостей часу виконання системи, оцінює порушення архітектурної моделі та, якщо виникає проблема, виконує адаптацію. Автори визнають проблеми координації та інших розподілених обчислень проблемою для майбутніх досліджень. З іншого боку, ми націлені на те, щоб підтримати інженерів у рамках аргументації для управління та керованого проектування та оцінки систем.

Однією з найбільш повних робіт є FESAS [19], структура, зосереджена на аспектах повторного використання компонентів, що визначають архітектуру керування SAS. Структура підтримує децентралізацію логіки адаптації через концепцію менеджерів логіки адаптації. У FESAS зв’язок між розподіленими елементами MAPE реалізується через публікацію/підписку системи, реалізована через проміжне програмне забезпечення для всеосяжних обчислень. З іншого боку, ми прагнемо дослідити, як архітектури керування, а також невизначеності в мережевому середовищі впливають на вимоги до системи.

В [20] пропонують фреймворк на основі архітектури програмного забезпечення, який забезпечує підтримку розробки мобільних програмних систем. Під час розробки структура використовується для оцінки атрибутів якості початкової архітектури системи. Під час виконання фреймворк постійно відстежує та розраховує найбільш прийнятну архітектуру. Подібно до цієї дисертації, запропонована структура пропонує повну інженерну підтримку життєвого циклу та передбачає для неї інтегрований набір інструментів від початку до кінця. Однак сфера роботи обмежена мобільними програмними системами, тоді як ми прагнемо охопити більш загальний клас децентралізованих SAS.

FUSION [21] спрямований на аналіз та самоналаштування адаптивної поведінки системи за наявності непередбачуваних змін. Основна увага FUSION зосереджена на вивченні впливу рішень щодо адаптації на цілі системи під час

виконання для автоматичного онлайн-налаштування логіки адаптації. FUSION керує архітектурою середовища виконання через централізований контур керування. З іншого боку, ми прагнемо надати інженерам загальну структуру для явного проектування та оцінки децентралізованих архітектур керування.

FUSION використовує цикл навчання (зображений на рисунку 1.3), щоб дізнатися про вплив рішень щодо адаптації з точки зору вибору функцій на цілі системи. Перше виконання циклу навчання відбувається до початкового розгортання системи. Система або моделюється, або виконується в автономному режимі і збираються показники, що відповідають кожному вибору функції. Ці дані використовуються для навчання FUSION для створення попередньої моделі поведінки системи.

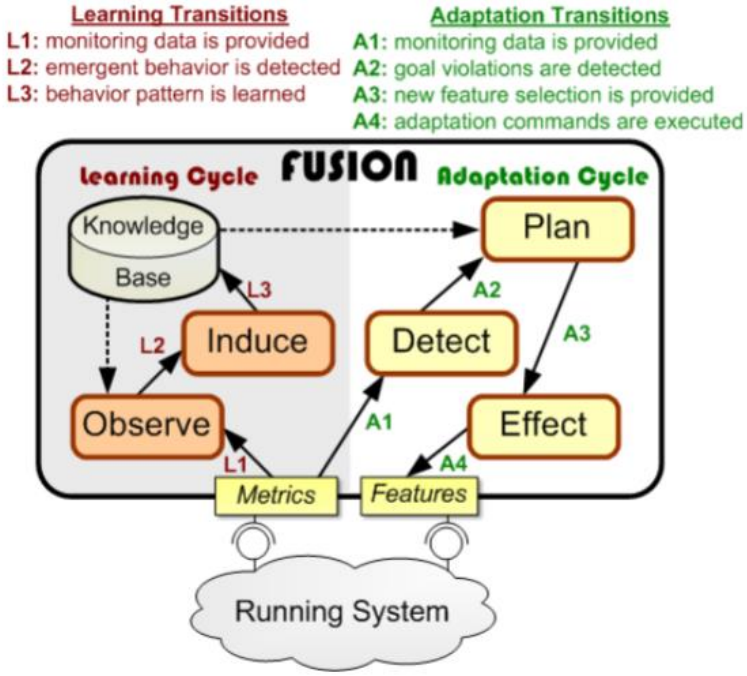


Рисунок 1.3 – Огляд фреймворку FUSION

В [23] створюють основу для архітектурної основи для представлення та проектування SAS. Запропонована структура допомагає розробнику у визначенні характерних елементів, для яких система демонструватиме самоадаптивні властивості. Їхня модель пропонує сувору підтримку ідентифікації та управління аспектами, які характеризують контекст і

самосвідомість. З іншого боку, ми прагнемо забезпечити підтримку дизайну децентралізованої самоадаптації системи.

В [20] виступають за використання абстракцій на основі компонентів для розробки SAS. Запропонована структура дозволяє моделювати розгортання в реальному часі шляхом оцінки поведінки системи за різних мережевих конфігурацій і налаштувань. Автори надають абстракції архітектури автономних компонентів і ансамблів компонентів, на основі яких можуть бути розгорнуті різні методи адаптації. З іншого боку, ми прагнемо забезпечити механізми для проектування та оцінки децентралізованих архітектур керування.

Arcaini та ін. [12] пропонують формальну структуру для специфікації архітектур керування SAS через абстрактні кінцеві автомати. Фреймворк дозволяє описувати складні цикли MAPE-K відповідно до шаблонів і використовує методи перевірки та перевірки для забезпечення правильності поведінки та взаємодії компонентів. Незважаючи на подібність із [12], ця теза не підтримує специфікацію та формальну перевірку поведінки компонентів MAPE. Однак ми зосереджені на забезпеченні допоміжного інструменту, який дозволяє оцінювання архітектур керування та керованих систем у поєднанні та визначення бажаних атрибутів якості.

В [19] використовують цикли керування зворотним зв’язком як абстракцію моделювання, щоб запропонувати керований моделлю підхід, використовуючи модель потоку даних [14], для проектування розподілених SAS. Подібно до [18], ця теза використовує петлі зворотного зв’язку як першокласну абстракцію моделювання для розробки архітектур керування. Однак ми прагнемо створити структуру, яка підтримує не лише розробку архітектури керування, але й її оцінку.

EUREMA [20] — це керований моделлю інженерний підхід, що забезпечує предметно-спеціальну мову моделювання та інтерпретатор часу виконання для адаптації на основі циклів зворотного зв’язку. Цей підхід базується на концепціях виконання мегамоделей, які зберігаються під час виконання, і шляхом їх інтерпретації вони безпосередньо виконуються для запуску циклів

зворотного зв’язку. DEUREMA [8] розширює попередній підхід, підтримуючи координацію розподілених циклів зворотного зв’язку. Обидві ці роботи забезпечують мову моделювання, яка підтримує явний дизайн циклів зворотного зв’язку. З іншого боку, ми прагнемо забезпечити підтримку моделювання для міркувань про властивості архітектури керування та керованої системи.

Нарешті, ми зазначимо, що література про підходи до розробки SAS величезна. У цьому розділі ми розглянули лише ті підходи, які тісно пов’язані з цією тезою. Підсумовуючи, огляд пов’язаних робіт з інженерної підтримки проектування та оцінки SAS вказує на фокус на моделюванні архітектури системи керування. У цьому відношенні підходи не залежать від керованої системи та забезпечують інтерфейси для підключення до даної керованої системи. Однак, жодна пов’язана робота не підтримує інтегровану оцінку можливостей під час архітектурних міркувань. Це являє собою дослідницьку прогалину, яку ми розглядаємо в цій роботі. Ми пропонуємо фреймворк із комплексною підтримкою життєвого циклу на рівні архітектури. Підтримка моделювання керованих і керуючих систем для проектування та оцінки дає можливість повносистемної аргументації, що, наскільки нам відомо, є новим внеском у галузі інженерних SAS.

1.3.3. Підходи до складання сервісу

Фокус у цій роботі лежить у загальній області робіт, що стосуються проблеми вибору та композиції послуг у розподіленому середовищі [4]. У цьому відношенні ми поділяємо з більшістю цих робіт загальне припущення, що набір залежностей служби відомий, коли вона випущена в робоче середовище. Це справедливо для залежностей, що стосуються інфраструктурних послуг, таких як обчислювальна потужність або сховище. Це також стосується залежностей вищого рівня, які були визначені під час розробки служби або в результаті попередньої діяльності з планування, яка розклала якусь глобальну ціль на набір підцілей, делегованих різним службам, які повинні інтегрувати одна одну.

Було запропоновано різні підходи до вирішення проблеми вибору та складу послуг. Зокрема, кілька статей сформулювали це як задачу оптимізації, яка централізовано вирішується виділеним брокером, припускаючи заздалегідь відомий набір послуг-кандидатів (наприклад [4]). Однак у цьому розділі ми не розглядаємо величезну кількість літератури щодо вибору та складання служби QoS, яка не має прямого відношення до даного дослідження.

В [18] динамічний набір агентів співпрацює, щоб зберегти деякі архітектурні обмеження. У цій роботі адаптаційні дії виконуються не автономно, а глобально скоординовано з використанням повністю впорядкованої трансляції, яка реалізує схему розподіленого блокування. Цей механізм глобальної координації вимагає чіткої взаємодії між усіма агентами. Отримані накладні витрати, таким чином, обмежують масштабованість архітектури та її можливе застосування в поширених середовищах. У роботі використовується адаптивна децентралізована процедура самоскладання на основі припущень. Однак система FlashMob вимагає, щоб кожен одноранговий пристрій підтримував і поширював інформацію про глобальний стан, яка складається з усього набору пропонованих і необхідних послуг. FlashMob також явно не розглядає глобальні цілі QoS і залежні від навантаження атрибути, які, з іншого боку, є об’єктом інтересу в цій роботі.

В [19] пропонують підхід для автономного збирання програмних компонентів. Подібно до цієї роботи, автори прагнуть вирішувати динаміку навколишнього середовища за допомогою онлайнового RL, але на відміну від нас припускають централізовану збірку та навчальний модуль. Їхня робота експериментує через прототип програмного забезпечення з часом відгуку конкретної програми, наосліп досліджуючи з цією метою різні збірки. Навпаки, у цій дисертації ми прагнемо використовувати моделювання для врахування масштабованості та аспектів невизначеності. Крім того, ми розглядаємо відому кореляцію між QoS компонентів, які збираються, і результатом збирання на основі загальної моделі, що залежить від навантаження, яка керує вибором служб, які збираються.

Дві роботи використовують методи децентралізованого навчання, щоб задовольнити вимоги QoS і перевірити масштабованість підходу, але не мають справу з атрибутами, що залежать від навантаження [21]. У [11] запропонована методика використовує RL для вирішення великомасштабних динамічних сервісних середовищ. Однак залежить від навантаження, а атрибути QoS явно не враховуються, хоча шаблон робочого процесу відомий службам апріорі під час розробки. Крім того, хоча автори [22] досліджують три специфічні атрибути QoS (доступність, час відгуку та надійність), у цій роботі ми пропонуємо загальну модель QoS, що залежить від навантаження, і показуємо її можливу реалізацію з різними атрибутами QoS. У [13] представики підхід до великомасштабної адаптивної композиції послуг. Їхня модель інтегрує багатоагентний RL для досягнення адаптації та теорію ігор, щоб дозволити агентам працювати над загальним завданням композиції. Подібно до цієї тези, їхній підхід спрямований на коригування процесу вибору за допомогою техніки RL для покращення QoS. Однак їхні агенти є лише виконавцями послуг у системі, які повинні максимізувати спільну мету, вивчаючи різні схеми співпраці. Насправді ця робота стосується кількох сценаріїв, виділених у їхній майбутній роботі. Зокрема, в [14] зазначено, що слід досліджувати поведінку розроблених підходів, коли агенти можуть здаватися конкурентоспроможними, якщо вони забирають ресурс на шкоду іншим агентам. У цьому відношенні ця теза зосереджена на тому, як можна покращити глобальний QoS у випадку атрибутів, що залежать від навантаження. Крім того, на додаток до обмежень QoS, [15] передбачає врахування довіри з огляду на існування небажаних служб і, з цією метою, включення механізму репутації в їх підхід. У зв’язку з цим ця теза спрямована на використання механізму репутації, який за допомогою моделі довіри врівноважує рекламоване QoS послуги з фактичним QoS.

В [16] пропонують повністю децентралізоване проміжне програмне забезпечення під назвою GoPrime для адаптивної самоскладання розподілених сервісів. Він заснований на використанні протоколу пліток для досягнення децентралізованого поширення інформації та прийняття рішень. За допомогою

GoPrime можна створювати та підтримувати набір служб, які, окрім функціональних вимог, також задовольняють глобальні QoS та структурні вимоги. З іншого боку, ця теза спрямована на врахування залежних від навантаження атрибутів якості та використання методів машинного навчання для досягнення стійкості.

В дослідженні [17] вивчають процес багатоагентного RL в контексті балансування навантаження в розподіленій системі без використання центральної координації чи явного зв'язку. Вони розглядають систему, що складається з певної кількості агентів, які використовують кінцевий набір ресурсів, кожен з яких має залежну від часу потужність. Це дослідження є найбільш схожим на наше, оскільки в ньому використовується залежна від навантаження модель для процедури вибору послуги з урахуванням QoS. Представлене експериментальне дослідження має справу з відносно невеликою системою з 100 агентів. У статті розглядається сценарій із одним типом залежності, де агенти вже знають повний набір доступних ресурсів. Навпаки, ця теза зосереджується на більш реалістичному припущенні, що кожен вузол не знає заздалегідь про інші вузли (і послуги, які вони пропонують) у середовищі, а виявляє їх динамічно.

Дослідники в [18] пропонують підхід, заснований на багатоцільовому RL, щоб полегшити проблему композиції сервісу з урахуванням QoS. У цьому дослідженні розроблено два алгоритми для обробки різних сценаріїв композиції на основі уподобань користувача. У [19] пропонується адаптивна техніка композиції сервісу з використанням ієрархічного навчання з підкріпленням (HRL). Подібно до цієї тези, автори розглядають кілька і, можливо, конфліктуючих атрибутів QoS, і зважують їх, щоб отримати єдине скалярне значення винагороди. Цікавим внеском статті є інтеграція автоматичної декомпозиції завдання та HRL для вирішення проблеми прокляття розмірності. Відходячи від цих двох робіт, ця дисертація спрямована на роботу з динамічними сценаріями та підтримку сталості надання послуг у разі неочікуваних змін.

На закінчення, розглянуті пов’язані підходи, не зовсім мають справу з залежними від навантаження атрибутами QoS, а вимоги до стійкості та масштабованості рідко беруться до уваги.

Ми пропонуємо стійкий підхід, який використовує машинне навчання та децентралізоване самоадаптування для підтримки масштабованої та децентралізованої збірки послуг з урахуванням QoS.

Висновки до розділу 1

В даному розділі наведені передумови дослідження та пов’язані роботи, які лежать в основі нашого дослідження. Виконано аналіз представлень принципів самоадаптації, висвітлено еталонну модель MAPE-K і визначено основну роль децентралізованої SAS. Також описано ряд підходів, які стосуються даної кваліфікаційної роботи. Досліджено сучасні фреймворки для розробки SAS і виконано аналіз літератури пов’язаної з підходами до додатків складання сервісів.

РОЗДІЛ 2. МЕТОДИ ВИВЕДЕННЯ ТА КОНЦЕПЦІЇ АДАПТИВНИХ СЕРВІСІВ НА ОСНОВІ ДАНИХ

2.1 Дослідження парадигми “Everything as a Service”

В майбутніх обчислювальних середовищах програму можна розглядати як мережеву систему, де велика відкрита колекція автономних і поширених речей динамічно взаємодіє одна з одною, щоб надати користувачам багаті функціональні можливості. Дійсно, програми динамічно з’являються як збірки цікавих речей, доступних у будь-який момент часу. Щоб досягти цього бачення, ключовими цілями є:

– сприяння розробці, реалізації та складанню речей, незалежно від їхньої специфічної природи;

– сприяння відкриттю та випадковому збиранню речей, що представляють інтерес.

З цією метою найкращі практики програмної інженерії пропонують використання так званої парадигми «все як послуга» (XaaS - everything-as-a-service), яка дозволяє однаково представляти фізичні речі, апаратні ресурси та програмні програми як незалежні та ізольовані об’єкти, що пропонують послуги. Крім того, XaaS полегшує створення нових комплексних сервісів як набору незалежних сервісів, доступних у середовищі [15].

Однак, щоб бути прийнятою в майбутньому обчислювальному середовищі, парадигма XaaS потребує ефективних рішень для проблем, які включають:

– взаємодію (як зробити гетерогенні служби здатними з’єднуватися та взаємодіяти один з одним);

– управління (як створити та підтримувати з часом відповідний набір служб, які разом здатні виконати задане завдання).

Обидві проблеми самі по собі не є новими в контексті XaaS [15], але вони значно посилюються специфічними характеристиками майбутніх обчислювальних

середовищ: перша — безпрецедентною гетерогенністю та різноманітністю сервісів, які мають бути пов’язані між собою, а друга — надзвичайною відкритістю, варіативністю. і непередбачуваність цього середовища.

2.1.1. Сценарій складання сервісу

Як сценарій мотивації розглянемо програму соціального сприйняття, що показана на рисунку 2.1. Його мета — зробити висновок про соціальну ситуацію людини (відвідування зустрічі, їзда на велосипеді, танці на танцмайданчику тощо) за даними, що надходять від різних датчиків, і зробити це доступним для її друзів через соціальні мережі.

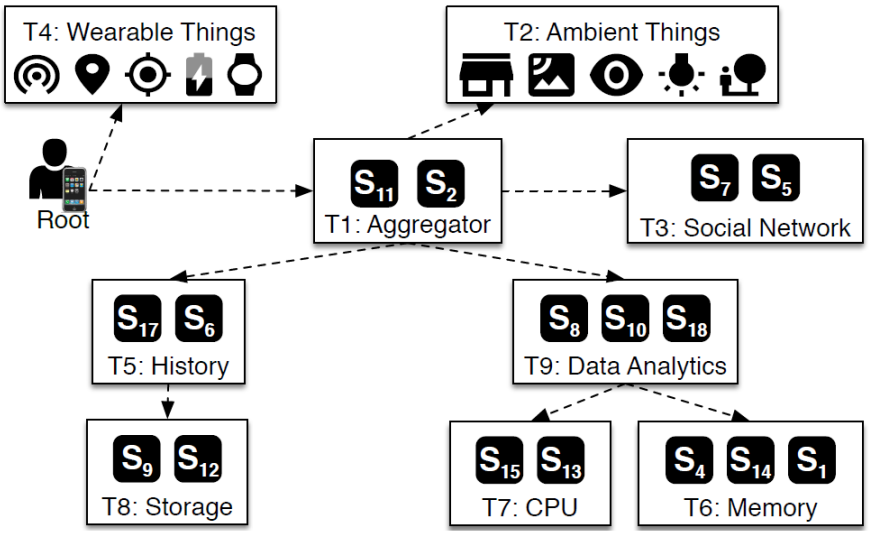


Рисунок 2.1 – Сценарій складання сервісу – програма соціального визначення

Цю програму можна створити з агрегації деяких логічно окремих блоків, які, ймовірно, включають:

– «кореневий» модуль, що реалізує логіку оркестровки програми;

– набір різних датчиків, що надають дані про контекст людини (наприклад, предмети і речі які можна носити);

– один або кілька модулів із інтенсивним обчисленням, які агрегують дані з обох цих датчиків, щоб отримати відповідні функції та зробити висновки про поточну ситуацію (тобто агрегатор);

– модуль соціальної мережі для обміну поточною соціальною ситуацією;

– модуль історії для збереження даних історії для пізньої (автономної) обробки.

Крім того, ці модулі потребують основних обчислювальних ресурсів, ресурсів зберігання та пам’яті для виконання своїх потреб у обробці, сховищі та пам’яті, відповідно. З точки зору XaaS, кожен із цих модулів програмного забезпечення та базові ресурси інфраструктури можна розглядати як організації, які пропонують послуги, і які можуть вимагати для цього послуги, які пропонують інші.

Ми можемо визначити сценарій розгортання цієї програми, де всі залежності, виражені її модулями, вирішуються службами, розміщеними на одному вузлі (зазвичай, якийсь персональний пристрій, як-от смартфон). Однак у передбачуваній мережевій системі з автономними та всеохоплюючими ресурсами, ймовірно, є кілька альтернатив. Наприклад, залежність даних датчиків може бути, принаймні частково, виконана датчиками, розміщеними на інших персональних пристроях, або розосередженими в середовищі.

Міркування про продуктивність або енергоспоживання можуть запропонувати вирішити залежність від обчислювальної потужності модулів аналітики даних за допомогою послуг обробки, які пропонують навколишні вузли крайової обчислювальної інфраструктури. Для цього можуть бути доступні різні пропозиції з різними характеристиками щодо продуктивності, енергоспоживання, вартості. Ці характеристики не можуть бути стабільними: наприклад, обчислювальна послуга, яка зараз кваліфікується як «хороша», насправді може не виявитися такою, якщо на неї зараз націлені запити, що надходять від інших програм, що працюють у тому самому середовищі, які перевантажують її. Крім того, доступність послуг може змінюватися (наприклад,

через мобільність або самостійні рішення їх власника); це вимагатиме реструктуризації попередньо визначеної збірки.

Якщо тепер уявімо натовп людей, які використовують свій власний екземпляр програми соціального визначення, який ми розглядаємо, кожен із цих екземплярів у кінцевому підсумку призведе до набору сервісів, яким знадобляться інші сервіси для спільного вирішення їхніх залежностей. Кожна така служба, ймовірно, буде зацікавлена в отриманні «хорошої» якості від послуг, які будуть вибрані для цього. Однак інтереси кожної послуги мають узгоджуватися з інтересами інших, щоб уникнути глобального незбалансованого вибору, який може фактично погіршити якість наданої послуги.

Крім того, зроблений вибір слід підтримувати протягом тривалого часу, адаптуючи його до мінливих умов, щоб гарантувати надання послуг і відповідати очікуванням щодо якості.

2.1.2. Вимоги до додатків складання служби

Розробка ефективного рішення для додатків складання послуг вимагає задоволення кількох вимог.

R1 – Стійкість: система має бути здатна працювати з динамічними сценаріями та підтримувати постійність надання послуг у разі неочікуваних змін. Внутрішній динамізм цього середовища, коли служби приєднуються/виходять із системи або змінюють свою поведінку, робить кожну збірку потенційно «нестабільною», яка тоді повинна бути стійкою до мінливого середовища.

R2 – масштабованість: система повинна працювати з великою кількістю об’єктів. Робота з безліччю різнорідних і автономних сервісів, як і в майбутніх обчислювальних середовищах, вимагає розробки процедур вибору сервісів і збирання які масштабуються зі збільшенням розміру системи.

R3 – Децентралізація: система повинна мати справу з розподіленим набором автономних об’єктів, де не існує глобальних централізованих знань. Відсутність глобальних знань, яких важко досягти та підтримувати у великій розподіленій

системі автономних сервісів, робить централізовані політики збирання та адаптації сервісів малопридатними. Дійсно, служби повинні координувати одна одну для децентралізації контролю [22].

R4 - QoS-обізнаність: система повинна враховувати атрибути QoS. Додатки, ймовірно, характеризуються глобальними вимогами до QoS щодо таких атрибутів, як доступність або продуктивність, які, таким чином, слід враховувати при створенні збірки.

R5 – Залежність від навантаження: система повинна враховувати, що деякі атрибути QoS залежать від фактичного навантаження під час роботи. Наявність функціонально еквівалентних послуг з різними значеннями їхніх атрибутів QoS може зробити нетривіальним визначення «найкращого» вибору пропонованих послуг, які будуть прив’язані до інших послуг, які їх потребують.

Дійсно, можлива залежність від навантаження характеру деяких атрибутів (наприклад, ті, що стосуються QoS), можуть виключати просту політику жадібного вибору послуг, яка завжди вибирає те, що наразі виглядає як найкраща послуга, оскільки вони можуть легко призвести до перевантаження служби та подальшого погіршення соціального добробуту системи.

Ми покладаємося на самоадаптацію як на ключову основу для досягнення вищевказаних вимог. Насправді SAS було запропоновано як рішення для створення стійких автономних систем [18]. Отже, у наступному підрозділі ми досліджуємо простір проектування нашої проблеми, аналізуючи розміри проектування, типові для самоадаптивних програмних систем. Зокрема, відповідаючи на запитання щодо дизайну, ми відобразимо визначені вимоги до попереднього простору вирішення нашої проблеми.

2.2 Простір проектування адаптивних навчальних систем на основі даних

Простір дизайну — це набір відкритих рішень щодо артефакту разом із обґрунтуванням кожного рішення. В [16] визначають п’ять кластерів вимірів для простору проектування SAS:

– спостереження;
– репрезентація (представлення);
– контроль;
– ідентифікація;
– адаптація виконання.

Визначені 5 вимірів підходять для проектування адаптивної системи програмного забезпечення на основі набору вимог та даних. Тому ми дотримуємося цієї таксономії, щоб дослідити простір проектування сценарію складання служби.

2.2.1. Спостереження

Кластер спостереження (рис. 2.2) включає виміри, що охоплюють проектні рішення щодо того, яку інформацію спостерігає система керування.

Ключове проектне рішення щодо SAS стосується «яку інформацію про зовнішнє середовище та саму систему слід вимірювати та представляти всередині?». Система керування повинна нести відповідальність за спостереження за доступністю послуг у середовищі та їх QoS (тобто R4).

Враховуючи набір спостережень, ще одне важливе проектне рішення стосується того, «як система визначатиме цю інформацію?». Через вимоги до масштабованості та децентралізації (тобто R2 і R3) мати єдиний компонент, відповідальний за моніторинг служб і їх QoS, є неможливим. З цією метою система повинна прийняти стратегію розподіленого моніторингу, де кожен вузол відповідає за моніторинг своїх власних розміщених послуг і їх QoS. Насправді, розподіл частини функціональних можливостей для кожного компонента системи є відомим методом сприяння масштабованості.

Design Question	Solution Space	R link
What information will the system observe?	Available services and their QoS	R4
How will the system determine that information?	Distributed Monitoring	R2,R3
What does trigger observation?	Periodic or reactive observation	R2,R3
What does trigger adaptation?	Existence of a "better" service	R4
How is uncertainty in the observations handled?	Machine Learning	R1,R5

Рисунок 2.2 – Проектний розмір – спостереження

Коли ми знаємо, що спостерігати і як це спостерігати, наступний вимір стосується «коли» спостерігати. Насправді є два питання щодо часу: "що викликає спостереження?" і "що викликає адаптацію?". Спостереження повинно проводитися або проактивно, наприклад, кожні t одиниць часу або реактивно, наприклад, коли виявлено зміни у значенні контрольованих змінних. Адаптація, з іншого боку, викликається існуванням служби з «кращим» QoS (тобто R4).
Остаточний набір рішень, що стосуються спостереження, стосується «як обробляється невизначеність у спостереженнях?». Цей розмір тісно пов'язаний із вимогами до стійкості та залежності від навантаження (тобто R1 і R5). Насправді індекси якості можуть залежати від навантаження, і служба, що рекламує найкращу якість, може бути перевантажена та не надавати її. У цій роботі ми прагнемо пом'якшити цей тип невизначеності за допомогою машинного навчання, оскільки воно надає можливість навчатися, вдосконалюватися та приймати рішення самостійно.

2.2.2. Представлення

Кластер представлення (рис. 2.3), пов'язаний із проектними рішеннями щодо того, як система та спостережувана інформація представлені під час виконання.

Design Question	Solution Space	R link
What information is made available to the components of the self-adaptive system?	Current and historical QoS	R4
How is this information represented?	Through internal data structures (might depend on the adopted learning technique)	R4,R5
When and how is the information updated?	Event-based messaging	R2,R3,R4

Рисунок 2.3 – Проектний розмір – представлення

Важливе проектне рішення стосується того, яка інформація має бути доступною для компонентів SAS (рис. 2.2). Оскільки система повинна вивчати минулий досвід, як поточний так і минулий QoS (тобто R4), отриманий кожною службою шляхом взаємодії з іншими службами, повинні бути доступні. Другий вимір дизайну стосується того, як представлена доступна інформація. Ми прагнемо розробити систему таким чином, щоб компоненти програмного забезпечення мали доступ до спостережуваної інформації через внутрішні структури даних.

Ми не вказуємо далі цей параметр, оскільки додаткова інформація, така як додаткові показники (наприклад, процентилі) або конкретний спосіб реалізації структур даних, може залежати від прийнятої техніки навчання.

Нарешті, останній вимір визначає, «коли» і «як» інформація оновлюється. Оскільки спостережувана інформація (тобто R4) є динамічною, необхідно застосувати підхід до оновлення подання. Зокрема, система повинна бути розроблена для виконання розподіленого моніторингу (рис. 2.2) і оновлення представленої інформації за наявністю нових даних, доступних для неї. З цією метою обмін повідомленнями на основі подій є добре відомим шаблоном проектування для управління інформацією у великомасштабних розподілених системах, що складаються з кількох компонентів [16]. Ідея використання повідомлень на основі подій полягає в тому, що дії (тобто оновлення інформації) мають ініціюватись подіями, які визначені як чітко визначені умови стану системи (тобто отримання повідомлення).

Обмін повідомленнями на основі подій дозволяє відокремити виробників і споживачів інформації, а потім досягти асинхронного обміну повідомленнями. Крім того, обмін повідомленнями на основі подій сприяє масштабованості та децентралізації (тобто R2 і R3).

2.2.3. Контроль

Цей кластер (рис. 2.4) пов'язаний з механізмами, що реалізують архітектуру керування на рівні керуючої системи, приводячи керовану систему у відповідність до цілей адаптації. Ми припускаємо, що вибір, зроблений у кластері спостереження (рис. 2.2), дозволяє системі мати достатньо параметрів для досягнення бажаного контролю.

Design Question	Solution Space	R link
Does the system provide enough parameters to achieve the desired control?	Yes	R4
How will the control loops be orchestrated?	Decentralized	R2,R3
What to adapt?	Service bindings	R4
How to adapt?	Machine Learning	R1,R5
When to adapt?	With the existence of a "better" service	R4

Рисунок 2.4 – Проектний розмір – контроль

Критичний вимір цього кластера визначає, як буде організований контур керування в системі. Вимога децентралізації (тобто R2) накладає прийняття кількох контурів керування для адаптації програмної системи, тобто децентралізована адаптація керування. З цією метою для проектування системи можуть бути використані різні архітектури децентралізованого керування. Однак перед прийняттям рішення слід ретельно оцінити якість архітектури. Важливим виміром цього кластера є рішення про те, «що» змінити. Основною метою системи є надання стійкого набору послуг за наявності залежних від

навантаження атрибутів QoS (R4 і R5). З цією метою механізм адаптації, якщо необхідно, адаптує прив'язки служби. Вимір, що описує «як» адаптуватися, відноситься до конкретної техніки навчання, яка реалізує самоадаптацію. Ми помітили, що вибір правильної методики, заснованої на навчанні, яка адаптує поведінку системи при зустрічі зі сценаріями, непередбаченими під час проектування, вимагає спочатку глибокого аналізу того, як машинне навчання використовується для реалізації децентралізованої адаптації.

Для розміру, що вказує «коли» адаптуватися, відповідно до тригера адаптації на рисунку 2.2, кожен вузол повинен виконувати децентралізованою процедурою перезв'язування служби з існуванням «кращої» служби.

2.2.4. Ідентифікація

Ідентифікаційний кластер (рис 2.5) відповідає за визначення характеристик екземплярів самоадаптації рішення.

Design Question	Solution Space	R link
What are possible solutions for a given set of adaptation targets?	Available services	R1
What are the relevant domain assumptions and context for each solution?	Stateless services	R1,R2
What are the required observed and control parameter for each solution?	Available services and their QoS	R4

Рисунок 2.5 – Проектний розмір – ідентифікація

Можливими рішеннями для набору цілей адаптації (щодо R1) є доступні служби, з якими кожен вузол може встановити прив'язку.

Третій вимір визначає відповідні припущення предметної області та контекст для кожного рішення. Ми припускаємо, що доступні служби не мають статусу. Сервіс без ідентифікації є принципом дизайну, який застосовується в рамках

сервіс-орієнтованої парадигми проектування, щоб розробляти масштабовані (R2) служби, відокремлюючи їх від даних стану, коли це можливо. Служба без збереження стану наразі є нормою для хмарних і периферійних програм. Це припущення дозволяє техніці адаптації оновлювати прив'язку служби (щодо R1) без урахування можливої історії транзакцій.

Нарешті, важливо визначити інформацію, яку слід брати до уваги, щоб увімкнути цикл зворотного зв'язку для переходу від одного рішення до іншого [16]. Ця інформація повинна відповідати кластеру спостереження (рис. 2.2) і складатися з доступності послуг у середовищі та їх QoS (тобто R4).

2.2.5. Адаптація виконання

Останній кластер (рис. 2.6) пов'язаний з визначенням механізмів адаптації, як вони запускаються, як вони підтримуються та як обробляється невдача [16].

Design Question	Solution Space	R link
Is the adaptation mechanism represented explicitly or implicitly in the architecture?	Explicitly	-
How is adaptability supported?	API	-
How will failures of the adaptation mechanism be handled	Rollback to fail-safe state	R1
What is the cause of adaptation	Functional and Non-functional changes	R1,R4,R5

Рисунок 2.6 – Проектний розмір – адаптація виконання

Ми прагнемо чітко представити децентралізовану архітектуру циклу зворотного зв'язку. З цією метою ми покладаємося на модель зворотного зв'язку MAPE-K для розробки та впровадження заходів, необхідних для реалізації самоадаптації. Адаптивність має підтримуватися керованою системою за допомогою набору API, які дозволяють прив'язувати службу. Нарешті, збій механізму адаптації переведе систему в безвідмовний стан (R1), де самоадаптація не підтримується,

але система продовжує працювати. Зауважте, однак, що це занепокоєння виходить за межі цієї кваліфікаційної роботи і не буде досліджуватися.

Нарешті, причиною самоадаптації є як функціональні, так і нефункціональні зміни в системі. З одного боку, нові служби можуть приєднатися до середовища під час виконання або існуючі служби можуть зникнути, створюючи кінцеву збірку нестабільною (тобто R1). З іншого боку, система повинна самоадаптуватися у відповідь на зміну QoS (тобто R4 і R5).

2.3 Вимоги до проектування адаптивної системи

Простір проектування, представлений у попередньому розділі, допомагає краще зрозуміти майбутню систему, відображаючи вимоги класу програм, які нас цікавлять, у початкове проектне рішення. Зокрема, кластер вимірів, описаний вище, ставить запитання, на які необхідно відповісти, щоб відповісти на наше дослідницьке запитання: «Як розробити стійкі програми як динамічну збірку послуг?» .

Результат такого попереднього дослідження простору дизайну вказує на три важливі проблеми, які необхідно додатково проаналізувати та розглянути в решті цієї дисертації, щоб відповісти на це запитання:

1. Як вибрати відповідні методи навчання?
2. Як проектувати та оцінювати архітектури керування?
3. Як розробити підхід до стійкої збірки сервісу?

Нижче ми детально описуємо ці виклики, щоб краще зрозуміти їх -актуальність.

2.3.1. Вибір відповідних методів навчання

Перше завдання проектування, яке необхідно вирішити, це вибір відповідної методики навчання для нашого сценарію. Дослідження дизайну, яке ми виконали, підкреслює потребу в техніці навчання, яка здатна працювати з невизначеними середовищами (рис. 2.2).

Однак, незважаючи на те, що навчання досліджувалося в багатьох амоадаптивних програмах, використання різних типів моделей навчання в SAS все ще є ручним процесом, який значною мірою покладається на знання предметної області. Немає жодних знань про найкращі практики SAS з підтримкою навчання, що залишає дослідників і практиків без вказівок щодо того, як пом’якшити цю періодичну проблему. Отже, перша мета цієї роботи полягає в наступному – проаналізувати, як вибрати відповідні методи навчання: щоб досягти цієї мети, ми прагнемо розглянути найкращі практики, прийняті в літературі, і отримати рекомендації щодо проектування, які дозволять нам приймати обґрунтовані рішення щодо методики навчання, яку слід застосовувати в нашому сценарії.

2.3.2. Проектування та оцінка архітектур керування

Дослідження дизайну, яке ми виконали, підтримує необхідність адаптації, реалізованої через децентралізовані контури керування (рис. 2.4), але як саме ці контури керування мають бути організовані? Ця пролема підкреслює потребу в систематичному способі оцінки рішень для архітектурних рішень.

Розробка, впровадження та оцінка великомасштабних SAS є складною справою, і проблема зборки сервісу, яку ми вирішуємо, є конкретним прикладом. Для цього потрібен підхід, який підтримує діяльність з проектування та впровадження з можливостями оцінки архітектур керування та прийняття обґрунтованих проектних рішень. Друга мета цієї роботи спрямована на заповнення наступної проблеми – реалізувати процес розробки та оцінки архітектури керування: щоб досягти цієї мети, ми прагнемо чітко представити архітектуру керування, покладаючись на модель зворотного зв’язку MAPE-K для розробки та реалізації дій, необхідних для реалізації самоадаптації (рис. 2.6). Отже, цілями для досягнення вищезазначеного завдання є:

– забезпечення підтримки специфікації та міркувань щодо децентралізованих архітектур керування;

– використання MAPE-K для першокласної абстракції моделювання;

– інтеграція платформи моделювання для підтримки оцінка SAS для прийняття обґрунтованих рішень на основі результатів моделювання.

2.3.3. Розробка підходу

Інженерні додатки, які динамічно з’являються як збірка сервісів, вимагають підходу, основною властивістю якого є стійкість. Отже, третьою і останньою метою цієї роботи є – розробка підходу до стійкої збірки сервісу: для досягнення цієї мети ми прагнемо використати результати, отримані за попередніми цілями. З одного боку, аналіз літератури допоможе прийняти проектні рішення щодо техніки, яку слід прийняти. З іншого боку, методологія, розроблена з O2, дозволить розробити та оцінити децентралізовану SAS та обґрунтувати її цікаву властивість.

2.4 Методології, керовані даними для децентралізованої системи

Розробка децентралізованої SAS є складною через недетерміноване та дуже динамічне робоче середовище, яке виникає внаслідок одночасної взаємодії кількох автономних об’єктів. Крім того, загальносистемні знання розподіляються між агентами, що передбачає використання передових механізмів для ефективного отримання знань. Наше дослідження підкреслює, що використання навчання додає ще один рівень складності, що залежить від наявності даних, вибору техніки, створення екземпляра моделі та оновлення моделі. Таким чином, численні впливові дизайнерські рішення, що виникають із такого багатовимірного та складного простору проектування спантеличують вибір інженерів при проектуванні децентралізованої SAS на основі навчання.

Із зростаючою доступністю даних програмної інженерії методи, керовані даними, з’явилися як ефективна методологія для забезпечення спеціалістів програмного забезпечення актуальною та актуальною інформацією, що

підтримує процес прийняття рішень. Методології, керовані даними, підтримують зменшення розмірності шляхом розпізнавання кореляцій у даних і широко застосовуються в літературі, наприклад, для розуміння еволюції програмного забезпечення [15] або виявлення зразків шаблонів проектування з вихідного коду системи [16].

Збір даних про відповідний минулий досвід, вилучення з них знань і надання знань у спосіб, який можна обґрунтувати, є першим кроком до підтримки інженерів SAS у навігації у великому просторі проектування таких систем і прийнятті економічно ефективного вибору [11]. Насправді, гарне розуміння наслідків, пов’язаних із конкретними розмірами конструкції, дає змогу надавати навчальні рішення, застосовні до класу SAS із подібними характеристиками.

Наприклад, якщо доступ до знань мінімальний, усі спостереження з навколишнього середовища здійснюються самим агентом і, таким чином, агент повинен досліджувати його середовище достатньо для навчання. Наслідки для навчання також можуть бути пов’язані з кількома вимірами. Наприклад, у SAS, що складається з автономних агентів з мінімальним доступом до знань, прийняті алгоритми навчання повинні швидко сходитися, оскільки інформація не ділиться, і агенти хочуть максимізувати свою власну корисність.

Використовуючи методології, що керуються даними, можна визначити кореляції в даних, зібраних нашим систематичним оглядом літератури, на основі яких ми представляємо відповідні знання про міркування під час розробки.

Ми використовуємо результати нашого систематичного огляду літератури щодо децентралізованих SAS на основі навчання як вхідні дані для наших методологій на основі даних, тобто ієрархічної агломеративної кластеризації (HAC - Hierarchical Agglomerative Clustering) і аналізу множинної відповідності (MCA - Multiple Correspondence Analysis).

Кластеризація дозволяє нам охопити набір варіантів дизайну, застосованих до SAS, на основі минулого досвіду (тобто сучасного рівня). Обґрунтування полягає в тому, що аналіз кластеризації можна використовувати для виявлення шаблонів проектування та зменшення складності простору проектування. Аналіз

відповідності, з іншого боку, зосереджується на виявленні кореляції між розмірами проекту. Це дозволяє фіксувати системні обмеження, які накладають певні варіанти дизайну. У результаті інженер отримує краще уявлення про взаємодію та взаємодію між різними розмірами проекту. Актуальність та ефективність цих методів для рішень щодо проектування системи були показані в [16].

В якості відправної точки HAC [18] розглядає кожне спостереження (тобто кожну рецензовану статтю в нашому випадку) як окремий кластер. Потім HAC поступово ідентифікує та об’єднує два найбільш схожі кластери. Поняття подібності між паперами стосується подібності між їхніми конструктивними розмірами. Оскільки вимірювання, визначені в нашому систематичному огляді літератури, є категоричними, ми використовуємо вимірювання відстані Гауера [19], яке є простим, але широко застосовуваним показником відстані, придатним для категоріальних даних. Типово застосовуваний метод для ідентифікації бажаної кількості кінцевих кластерів у HAC (заданий K) передбачає використання критеріїв перевірки кластеризації [15]. У нашому аналізі ми покладаємося на силуетні значення [18] і метод початкового завантаження [17]. Значення силуету є мірою узгодженості даних, де вищі значення представляють більшу узгодженість між точками даних у кластері. На рисунку 2.7 показано значення силуету для HAC до 10 кластерів. Сюжет припускає, що через п'ять кластерів узгодженість точок даних у кожному кластері падає та досягає локального максимуму з K = 9.

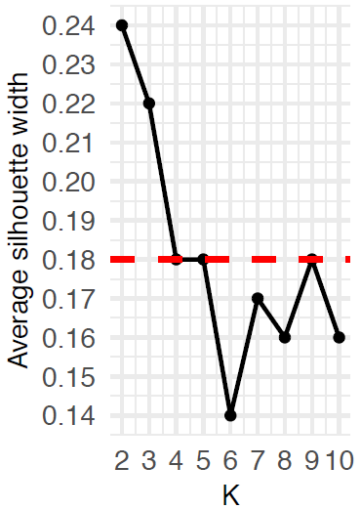


Рисунок 2.7 – Значення для Hierarchical Agglomerative Clustering

Бутстрап-аналіз дає змогу оцінити стабільність розглянутої кількості кластерів шляхом дослідження того, наскільки легко кластери розчиняються. Для проведення початкового аналізу ми вибираємо кластери із середньою шириною силуету, що перевищує або дорівнює 0,18, тобто середньою шириною силуету з K = 9 (див. пунктирну лінію на рисунку 2.7). Нижчі значення силуету призводять до слабких або штучних структур. Результати початкового завантаження представлені на рисунку 2.8.

K	Vector of clusters stabilities	Times each cluster is dissolved in 100 re-sampling
2	⟨0.90,0.85⟩	⟨0,0⟩
3	⟨0.82,0.86,0.50⟩	⟨2,0,59⟩
4	⟨0.87,0.86,0.60,0.61⟩	⟨0,1,48,9⟩
5	⟨0.84,0.77,0.64,0.69, 0.27⟩	⟨0,8,43,29,92⟩
9	⟨0.70,0.69,0.61,0.64,0.84,0.74,0.47,0.51,0.33⟩	⟨18,32,40,36,21,29,67,69,93⟩

Рисунок 2.8 – Bootstrap аналіз

Для кожного числа розглянутих кластерів K ми повідомляємо: вектор стабільності кластерів (значення, близькі до 1, вказують на стабільні кластери) і кількість разів, коли кожен кластер розчиняється після 100 повторних вибірок (кластери, які розчиняються, часто є нестабільними). Результати показують, що

K = 2 утворює два стабільні кластери, які ніколи не розчиняються. K = 3 і K = 4 вводять легкий ступінь нестабільності, тоді як K = 5 і K = 9 призводять до високої нестабільності.

Висновки до розділу 2

В даному розділі описано бачення майбутніх обчислювальних середовищ і додатків, що з’являються як збірки сервісів. Представлено мотиваційний сценарій, з якого виводимо набір вимог, які необхідно задовольнити для розробки цього класу програм. Потім, використано простір дизайну, щоб допомогти перейти від аналізу вимог до передбаченого рішення. Також в цьому розділі представлено дослідницькі завдання, які необхідно вирішити для розробки цих програм.

РОЗДІЛ 3. АНАЛІЗ І МОДЕЛЮВАННЯ ДЕЦЕНТРАЛІЗОВАНИХ АДАПТИВНИХ СИСТЕМ

3.1 Дослідження процесу навчання в децентралізованих адаптивних системах

Метод дослідження, прийнятий у цьому розділі, відповідає стандартній практиці систематичних оглядів літератури. Оскільки наше дослідження зосереджено на SAS на основі навчання, сфера огляду обмежена рішеннями SAS, у яких агенти всередині розширені здібностями до навчання.

Досліджувані документи класифіковано на основі їх вибору для дев'яти розмірів дизайну, передбачених для SAS на основі навчання (рис. 3.1).

Dimension	Description	Q
Application Domain	The domain for which a SAS is developed	Q1
Autonomy	The agents' ability to act autonomously or in need of a supervised entity	Q1
Knowledge Access	The amount of information available to an agent from its peers or the environment	Q1
Behavior	Agent's comportment toward self-goals and global goals	Q1
Emergent Behavior	Whether the collective demonstrates a behavior different than the one of single agents	Q2
Cooperative Agent	Whether agents are cooperating	Q2
Learning Technique	The technique used by agents to exhibit learning	Q3
Trigger First	The initial knowledge used to instantiate learning models	Q4
Trigger Update	Criteria for updating the learning models	Q4

Рисунок 3.1 – Розміри проекту SAS на основі навчання

Ці розміри доповнюють простір проектування, представлений у попередньому розділі, зосереджуючись на конкретних аспектах, що характеризують рішення, засноване на навчанні.

Зокрема, введені параметри надають детальний погляд на рішення, засновані на навчанні, що стосуються проблеми «як адаптуватися», «коли адаптуватися» і «що викликає адаптацію».

Автономія означає здатність агента мати повний контроль над власним станом і поведінкою. Коли автономний агент навчається, він має повну автономію в процесі навчання, тобто він відповідає за створення моделей навчання, коригування параметрів тощо. З іншого боку, агенти без автономії оновлюють свою модель навчання ззовні. Це може зробити одноранговий або наглядовий агент, який виконує діяльність з оновлення моделі, і агент використовує оновлену модель (після її отримання), щоб керувати своїми діями. Хоча це компрометує стандартне визначення агента, його можна використовувати в окремих ситуаціях, наприклад, для побудови моделі навчання з більшою кількістю даних, ніж той, до якого агент може отримати доступ. Обмежена автономія відповідає екземплярам SAS, у яких зовнішні об’єкти мають лише частковий прямий вплив на процес агента для оновлення його моделі навчання.

Більшість проаналізованих досліджень передбачає повну автономію для всіх агентів, тобто агенти не контролюються, і всі їхні рішення виконуються в їхньому середовищі. Це означає, що всі агенти несуть відповідальність за свої дії і не можуть бути перезаписані ієрархічно вищою сутністю.

Наш аналіз показує, що деякі моделі на основі агентів явно обмінюються між собою навчальною інформацією. Ми називаємо цю концепцію доступом до знань.

Хоча агенти SAS знаходяться в середовищі та можуть сприймати його, ступінь сприйняття може коливатися від обмеженої частини середовища, де знаходиться агент, до всього середовища. Крім того, агенти можуть співпрацювати, що означає, що агент може мати доступ не тільки до інформації, яку він сприймає з навколишнього середовища, але також до знань, наданих іншими агентами, включаючи їхню внутрішню інформацію. Отже, наше поняття доступу до знань також включає ступінь обміну інформацією між агентами. Однак ми зосереджуємося на цінності інформації, доступної для агента, а не на засобах, які

використовуються для отримання інформації. Таким чином, доступ до знань коливається від мінімальних знань, у яких агенти не обмінюються інформацією, до максимальних знань, у яких інформація повністю розподіляється між агентами, що забезпечує повне сприйняття середовища.

Мета SAS визначається її навчальними завданнями, які зазвичай відповідають глобальним цілям. Окрім завдань навчання, взаємодія між агентами (навмисна чи ненавмисна) може призвести до емерджентної поведінки для колективу. Емерджентну поведінку не можна передбачити апріорі, оскільки вона не є простою сукупністю поведінки окремих агентів. Щоб відповісти на друге дослідницьке запитання, ми аналізуємо дослідження на основі їхніх навчальних завдань (тобто навчальної мети) та можливих форм поведінки. Аналіз взаємозв'язку між цими двома аспектами виявив наступні спостереження.

Ключовою відмінністю між дослідженнями є навчальне завдання та поведінка, що викликає інтерес. Декілька досліджень не повідомляють про будь-яку конкретну невідкладну поведінку, але пов’язують невідкладну поведінку з очікуваним навчальним завданням [13; 16]. У підмножині цих досліджень завдання навчання та пов’язана з нею поведінка виконуються, не вимагаючи від агентів активної співпраці.

Ми визначили дві основні групи щодо зв’язку між навчальними завданнями та поведінкою, що виникає. Одна група стосується досліджень, які пов’язують емерджентну поведінку з точним очікуваним навчальним завданням колективу. Навчальне завдання в цих дослідженнях є або сукупністю окремих неспівпрацюючих завдань або загальносистемної співпраці між агентами.

Навчання з підкріпленням (RL - reinforcement learning) є широко використовуваним методом у SAS. Посилаючись на рисунок 3.2, результати нашого огляду підкреслюють, що біля 60% відібраних досліджень використовують RL.

Learning Technique	# Studies	Percentage
Reinforcement Learning (RL)	35	59,32%
Supervised Learning	6	10.17%
Game Theory	5	8,47%
Probabilistic	4	6.78%
Statistics	2	3.39%
Swarm System	2	3,39%
Applied Logic	1	1.69%
Evolutionary Process	1	1.69%
Game Theory and RL	1	1.69%
Gradiend Descend	1	1.69%
Genetic Algorithm + RL	1	1.69%

Рисунок 3.2 – Класифікація на основі техніки навчання

У [4] і [12] навчання використовується для самоорганізації мережі. Q-навчання обрано через його простоту та придатність для представлення поведінки запропонованого механізму в термінах дій та винагород. У [13] автори мають справу з проблемами оптимізації кількох політик у SAS, використовуючи розподілене W-навчання (DWL - Distributed W-Learning), підхід RL, на основі W-навчання [15], де кожна політика реалізується одним процесом Q-навчання. Автори стверджують, що Q-навчання є підходящою технікою навчання та оптимізації в ситуаціях, коли немає попередньо визначеної моделі середовища. Так само RL використовується в задачах децентралізованого планування, оскільки агентам не потрібно знати модель апріорі, але вони можуть вивчати політики через повторювану взаємодію з середовищем і один з одним.

Багатоагентне підсилювальне навчання (MARL - Multi-agent reinforcement learning) використовується в [13] для вирішення проблем розподіленим способом у нестаціонарних середовищах, коли централізоване керування стає неможливим. MARL також використовується для подолання складності, що виникає в доменах MAS, оскільки він дозволяє адаптивним і автономним агентам покращувати свої моделі навчання на основі досвіду.

Канонічний погляд на процес навчання агента SAS. Агент SAS на основі навчання розроблено для покращення його здатності виконувати завдання

навчання. Незалежно від характеристик SAS, тобто мети навчання, методів навчання та тригерів, що використовуються для вивчення та вдосконалення моделі, кожен агент у межах свого циклу самоадаптації (наприклад, циклу MAPE-K) дотримується стандартного процесу навчання.

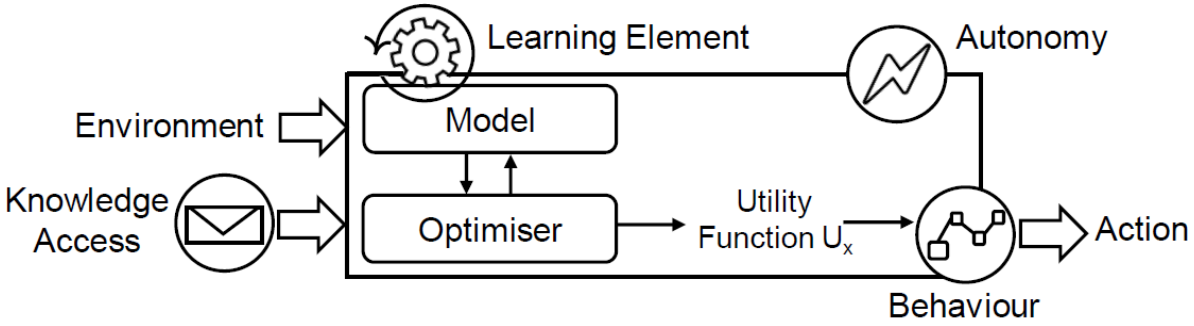


Рисунок 3.3 – Канонічний вигляд процесу навчання агента SAS

Рисунок 3.3 ілюструє канонічний вигляд процесу навчання в такій мережі самоадаптації. Елемент навчання приймає навколишні та внутрішні спостереження як вхідні дані та генерує дію (що веде до реконфігурації SAS) як вихідні дані. Всередині, функція корисності U_x використовується для характеристики бажаності поведінки агента. Крім того, навчальний елемент містить модель (наприклад, класифікатор) і компонент оптимізації (або контролер) який приймає рішення про дії, що оптимізують використану функцію корисності.

Залежно від рівня автономії, рішення агента можуть бути скасовані, змінені або заблоковані зовнішнім втручанням; Залежно від рівня доступу до знань, інформація від інших агентів може бути доступна на додаток до локального перегляду внутрішньої інформації та інформації про середовище; Залежно від типу поведінки, визначення функції корисності U_x може відрізнятися між перевагою агента та/або загальносистемною корисністю.

Агент SAS може отримати можливість виконувати завдання навчання під час проектування, під час виконання або гібридним способом.

Навчання під час проектування вимагає великої кількості вибіркових даних для навчання класифікатора або регресійної моделі з використанням контрольованих або неконтрольованих методів навчання. Вироблені знання (тобто попередньо навчена модель) використовується під час виконання для керування SAS. Для навчання під час виконання, допоміжна функція постійно оцінює переваги SAS на основі отриманих значень корисності. Застосовувана модель навчання поступово вдосконалюється шляхом використання даних часу виконання. Такі дані збираються за допомогою прийнятої методики навчання (наприклад RL) та знань, що надаються колегами (залежно від рівня доступу до знань). Схема гібридного навчання використовує попередньо зібрані зразки даних для навчання моделі навчання під час проектування та використовує дані під час виконання для вдосконалення моделі.

Удосконалення навчальної моделі (з урахуванням особливостей навчального завдання) входять до однієї з наступних комбінацій (перелічених у порядку збільшення складності):

1. Навчальне завдання і навчальна модель мають статичні властивості. Це призводить до вибору попередньо визначеної конфігурації під час розробки. Такі характеристики не дозволяють агенту SAS реагувати на непередбачені події та не дозволяють навчатися під час виконання. У цьому налаштуванні агент SAS навчається лише під час розробки, використовуючи лише зразки даних, виключаючи будь-яку інформацію під час виконання [7].

2. Модель навчання є динамічною, а навчальна задача має статичні ознаки. Це дозволяє оновлювати модель навчання під час виконання у відповідь на знання, зібрані (у несподіваний час) із середовища або агента (наприклад, межа рішення в класифікаторі). Техніки навчання (наприклад, RL) використовуються під час виконання для оновлення моделі. Це найпоширеніший метод навчання в проаналізованих дослідженнях.

3. Навчальна задача має динамічні ознаки, а модель статична. Цей параметр може бути результатом різної доступності вхідної інформації (наприклад, зміна шаблонів датчика). Цей параметр вимагає або набору попередньо навчених

моделей [14], швидкого перемикання між ними під час виконання або оптимізаційних методів інтерпретації моделі з використанням обмеженого введення.

4. Як навчальна задача, так і модель мають динамічні особливості. Ці функції можуть бути результатом активації під час виконання, додавання, дезактивації або видалення датчиків. У цьому параметрі динамічна модель відноситься до розгляду інформації під час виконання (наприклад, через RL).

3.2 Структура міркування на основі моделі

Підходи до впровадження для SAS класифікуються відповідно до типу інтеграції між логікою адаптації та керованою системою. У внутрішніх підходах логіка адаптації переплітається з керованими ресурсами, тоді як зовнішні підходи чітко відокремлюють логіку адаптації від керованої системи, узгоджуючи їх за допомогою датчиків і виконавчих механізмів. Це робить зовнішні підходи добре придатними для впровадження децентралізованих SAS, оскільки вони сприяють поділу інтересів і модульності, що призводить до масштабованості та повторного використання.

Як було введено раніше, у великомасштабних розподілених системах прийнято децентралізувати управління та розгортати кілька скоординованих елементів адаптації. Дійсно, композиція таких елементів адаптації у відповідній архітектурі керування є важливою. З цією метою важливо оцінити та порівняти різні архітектури керування та вибрати найкращий кандидат з огляду на цікаві атрибути якості для даної керованої системи. Дійсно, мета полягає в тому, щоб отримати уявлення про якість SAS для розробки: враховуючи конкретну керовану систему, правильна архітектура керування визначається в ітеративному процесі міркування на основі моделі. Це підтримує системних інженерів у процесі прийняття обґрунтованих архітектурних рішень щодо системи, що досліджується, потенційно вирізаючи або детально досліджуючи ті архітектури керування, які показують поганий або хороший рівень якості відповідно.

Як показано на рисунку 3.4, процес починається зі звичайного аналізу вимог, який визначає набір бажаних показників якості для майбутньої системи.

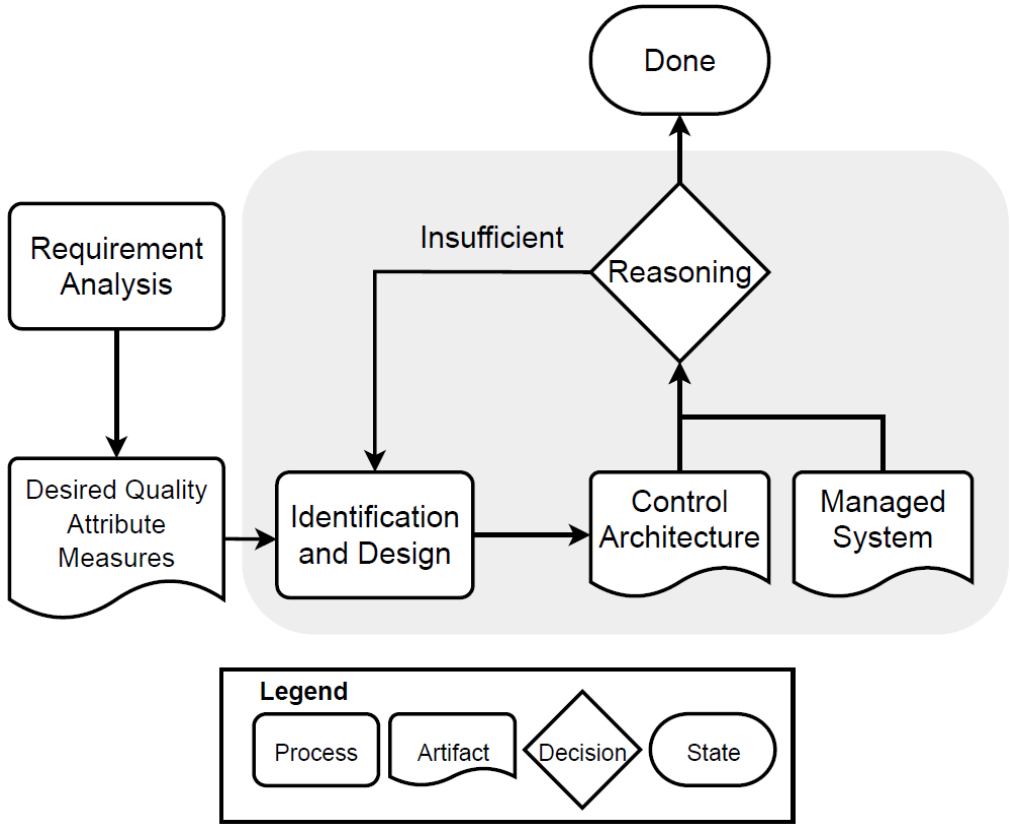


Рисунок 3.4 – Процес прийняття рішення

Потім ідентифікація та проектування присвячені розробці потенційної архітектури керування для даної керованої системи. Зауважте, що розробник вибирає потенційну початкову архітектуру, використовуючи знання домену (якщо такі є), які він має про адресований домен. Окрім знання предметної області, іншими факторами вибору архітектури керування можуть бути конфіденційність даних, дизайн алгоритмів і обмеження домену. Потім обґрунтовується SAS (тобто керована система та архітектура керування-кандидата) шляхом оцінки параметрів бажаної якості. Якщо система не задовольняє бажану якість, процес повертається до діяльності ідентифікації та проектування шляхом створення циклу оцінки дизайну (див. рис. 3.2). Коли архітектура-кандидат задовольняє бажану якість, проектування SAS завершується. Дійсно, міркування повинні враховувати два різні питання:

– якість архітектури керування;

– вплив адаптації на якість самої керованої системи.

Щоб полегшити такий процес прийняття рішень, нам потрібен систематичний і інтегрований підхід, заснований на моделях, щоб обґрунтувати архітектури керування та їх вплив на керовану систему.

Рамки міркування дозволяють оцінювати конкретні атрибути якості архітектури, що розробляється, за допомогою чітко визначених аналітичних теорій. Відповідно до [17] структура обґрунтування повинна містити такі елементи:

1. Опис проблеми: набір показників якості, які можна розрахувати за допомогою Reasoning Framework.
2. Аналітична теорія: теоретичні основи, на яких базується аналіз.
3. Аналітичні обмеження: набір обмежень, накладених аналітичною теорією.
4. Представлення моделі: модель системи у формі, придатній для використання з процедурою оцінювання.
5. Інтерпретація: процедура, яка використовується для генерації уявлень моделі з архітектурних описів.
6. Процедура оцінювання: методи, які використовуються для розрахунку показників якості на основі представлення моделі.

З цією метою DECOR поєднує усталену аналітичну теорію та системну інженерію на основі моделей, щоб реалізувати такі ключові елементи (рис. 3.5) і підтримувати діяльність міркування.

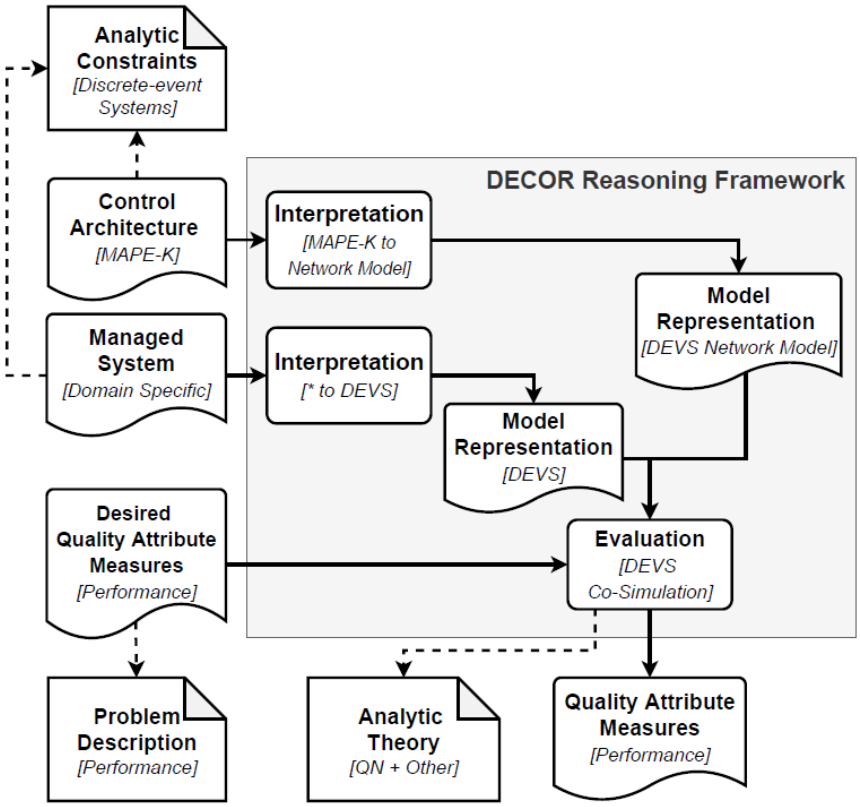


Рисунок 3.5 – DECOR reasoning framework

Опис проблеми. Опис проблеми визначає атрибути якості, для яких використовується система аргументації. Зокрема, DECOR робить акцент на якості показників продуктивності (наприклад, час адаптації, масштабованість тощо) в архітектурі керування, а також різні вимірювання якості в керованій системі. З цією метою DECOR використовує розширену структуру архітектурного обґрунтування (eARF - extended Architectural Reasoning Framework) [3], щоб визначити вимоги як сценарії атрибутів якості домену (dQAS - domain Quality Attribute Scenarios), зв'язавши їх з Architecture Tactics і Architecture Patterns (рис. 3.6). Крім того, dQAS дозволяє вказати, як система повинна реагувати в конкретних ситуаціях.

Аналітична теорія (обмеження). Аналітична теорія представляє основу структури міркувань, оскільки дає змогу систематично аналізувати бажані міри атрибутів якості. Дійсно, він надає інженерам конкретні значення для оцінки та порівняння різних адаптацій, а також їх впливу на керовану систему. Враховуючи неоднорідність децентралізованих SAS (наприклад, сервіс-

орієнтовані системи, системи трафіку та інтелектуальні мережі), DECOR використовує мультипарадигмальну аналітичну теорію, зокрема, починаючи з теорії мереж масового обслуговування широко визнаний як потужний і універсальний інструмент для оцінки та прогнозування продуктивності системи [19], він використовується на рівні архітектури керування для оцінки показників її якості. З іншого боку, інші теорії, придатні для адресованого домену і конкретних показників якості, що представляють інтерес, використовуються на рівні керованої системи.

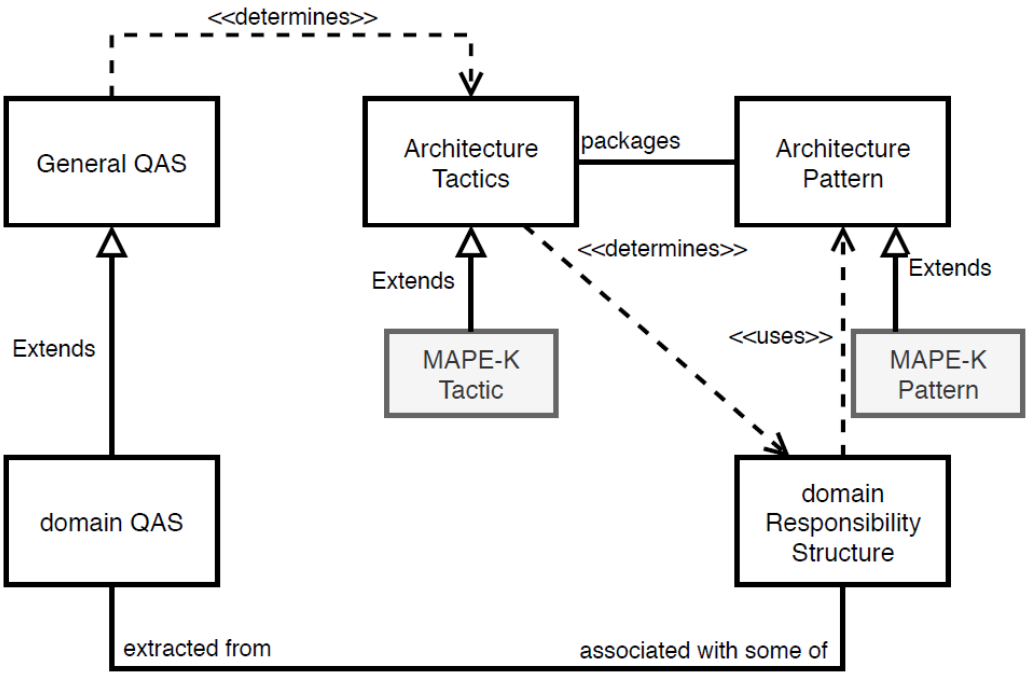


Рисунок 3.6 – Розширена структура архітектурних міркувань

Застосування теорії мереж масового обслуговування на рівні архітектури керування накладає деякі аналітичні обмеження на тип керування, який можна аналізувати. Дійсно, DECOR можна використовувати для аналізу та оцінки лише тих архітектур керування, визначених як система дискретних подій. З іншого боку, керована система може бути як дискретною, так і безперервною, залежно від конкретного адресованого домену.

Представлення моделі та інтерпретація. Представлення моделі пов’язане з конкретною аналітичною теорією, яка використовується в основі міркувань.

Щоб адаптувати мультипарадигмальну аналітичну теорію, DECOR використовує багатопарадигмальну модель представлення. Зокрема, архітектура керування моделюється як модель мережі, тоді як керована система моделюється відповідно до відповідного формалізму, залежно від конкретної адресованої області. Такі представлення моделі генеруються Інтерпретацією з інформації, що міститься в Архітектурі керування та Керованій системі відповідно.

Процедура оцінювання. Процедура оцінювання — це застосування аналітичної теорії для розрахунку конкретних показників якості. Методи оцінки можуть бути формальними, напівформальними або заснованими на моделюванні. Формальні методи (наприклад, перевірка моделі, мережа Петрі, часові автомати) широко використовуються в літературі для оцінки SAS [13]. Однак ці методи, незважаючи на властиву їм здатність надавати гарантії, погано масштабуються для великих проектів. Ефективне використання формальних методів вимагає скорочення обчислювальної задачі різноманітністю таких методів, як декомпозиція/композиція моделі або спрощення припущень. Однак, навіть із спрощеними припущеннями та декомпозицією, результуючі аналітичні моделі часто не піддаються математичній обробці.

Таким чином, життєздатною альтернативою для оцінки неіснуючої системи є моделювання. Зокрема, DECOR розраховує бажані показники атрибутів якості за допомогою механізму спільного моделювання, здатного виконувати представлення моделі з кількома парадигмами і забезпечуючи повну системну оцінку контролю адаптації. Варто зауважити, що для виконання спільного моделювання представлення моделі повинні відповідати специфікації системи дискретних подій (DEVS - Discrete Event System Specification) [16]. Це означає, що буде можливо імітувати безперервні керовані системи як системи з дискретними подіями.

DECOR полегшує діяльність міркування та дозволяє замкнути цикл у процесі прийняття рішень (рисунок 3.2): інженери можуть оцінити правильність і придатність архітектур керування та прийняти рішення щодо dQAS для всієї системи керування в контексті керованої системи.

З цією метою DECOR надає допоміжний інструмент для проектування архітектури керування, виконання оцінки за допомогою спільного моделювання (рис. 3.7). Подвійний процес дозволяє різним зацікавленим сторонам використовувати основні функції DECOR у середовищі моделювання архітектури управління (CAME - Control Architecture Modeling Environment) і середовищі спільного моделювання (CoSE - Co-Simulation Environment) які зображені заштрихованими прямокутниками на рисунку 3.7 (див. 2 і 3). Варто зауважити, що два актори повинні спілкуватися один з одним, щоб забезпечити розуміння та координацію [3].

Інженер керованої системи відповідає за моделювання функціональних можливостей керованої системи за допомогою інструменту моделювання (див. 1), придатного для конкретної адресованої області (наприклад, мережева модель, модель трафіку мікросимуляції).

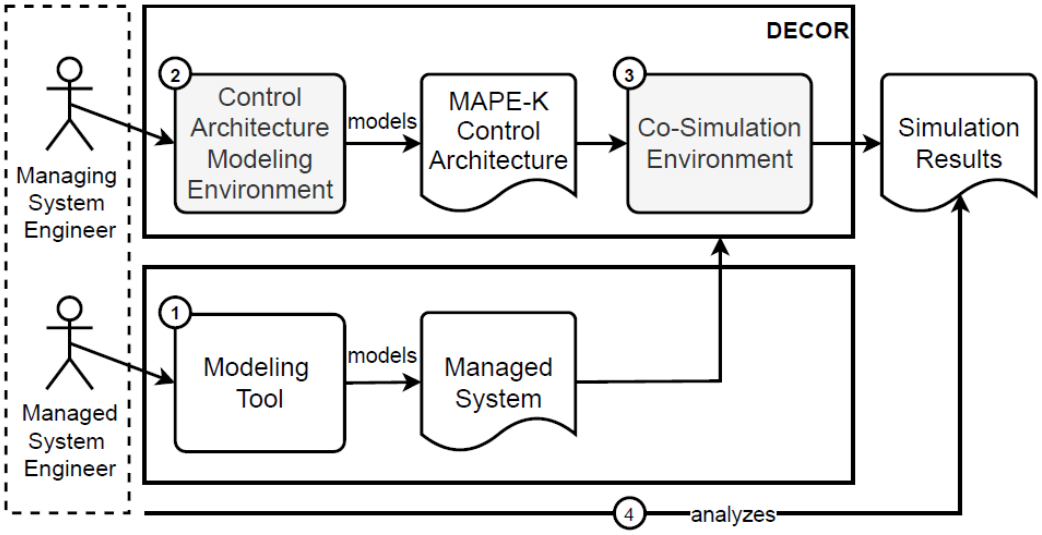


Рисунок 3.7 – Міркування за допомогою інструментів

З іншого боку, керуючий системний інженер відповідає за моделювання архітектури керування, реалізуючи логіку адаптації. По-перше, інженер імпортує модель керованої системи в DECOR. Завдяки цьому основні властивості моделі керованої системи (тобто те, що можна спостерігати та контролювати) автоматично стають доступними в CAME. Потім інженер розробляє архітектуру

керування MAPE-K (див. 2) для системи керування та моделює дві моделі в CoSE (див. 3).

DECOR передбачає різні стратегії оцінювання: архітектуру керування та моделі керованої системи можна моделювати окремо для перевірки їхніх власних властивостей окремо або разом для оцінки загальної поведінки системи. На основі аналізу результатів моделювання (див. 4) інженери можуть розпочати нову ітерацію, щоб удосконалити модель керування та керованих систем.

DECOR дозволяє досліджувати показники атрибутів якості, що стосуються як керованої системи (наприклад, середній QoS вузла служби), так і системи керування (наприклад, час адаптації, кількість обмінених повідомлень). Ці атрибути якості значною мірою залежать від конкретної прийнятої архітектури керування, оскільки різні шаблони та стратегії реконфігурації можуть передбачати різні показники атрибутів якості. Дійсно, головна мета DECOR полягає в тому, щоб уможливити оцінку та порівняння різних архітектур керування, що потім призводить до вибору архітектури керування, яка досягає бажаного рівня якості.

3.3 Моделювання архітектур керування

Як було зазначено вище, DECOR використовує eARF для визначення вимог до якості (тобто dQAS) і зв’язування їх з тактикою архітектури та шаблонами архітектури (рис. 3.6). З цією метою eARF визначає структуру відповідальності домену (dRS - domain Responsibility Structure) як архітектуру керування на основі компонентів, що реалізує стратегію самоадаптації, де обов’язки компонентів безпосередньо витягуються з dQAS (рис. 3.6). Потім вибирається архітектурна тактика для реалізації dQAS і призначаються обов’язки (відображення на елементи dRS).

3.3.1. Метамодель архітектури керування MAPE-K

DECOR використовує тактику MAPE-K, яка широко визнана ефективною для реалізації самоадаптації. Зокрема, DECOR покладається на децентралізовану архітектуру керування MAPE-K, яка дозволяє структурувати та координувати декілька взаємодіючих компонентів MAPE-K (рис. 3.6). В роботі [22] автори вводять просту нотацію для опису інтерактивних циклів MAPE-K. DECOR формалізує таку нотацію та надає загальну метамодель для визначення взаємодії MAPE-K.

Мета-модель архітектури керування MAPE-K (додаток А) походить від FORMS, яка визначає уніфікуючу еталонну мета-модель для формальної специфікації розподілених SAS [20]. Метамодель FORMS складається з кількох елементів моделювання, які визначають ключові аспекти SAS, і набір зв’язків, що керують його створенням. У цьому контексті треба розглянути додаток Б, мета-модель архітектури керування MAPE-K походить від FORMS, надаючи конкретну перспективу аспектів координації, які стосуються компонентів MAPE-K, що реалізують архітектуру децентралізованого керування. Посилаючись на FORMS, координація являє собою набір правил, що регулюють взаємодію між обчисленнями, що беруть участь.

Посилаючись в додатку А, архітектура керування MAPE-K складається з набору архітектурних елементів, а саме підсистеми та взаємодії. Ці два елементи визначають відповідно компоненти та з’єднувачі архітектури децентралізованого керування.

Підсистема спеціалізується на самоадаптивному пристрої та локальній керованій системі. Локальна керована система моделює керовану систему та визначає набір атрибутів, що представляють її спостережувані/контрольовані аспекти. Self-Adaptive Unit моделює систему керування з точки зору п’яти різних типів компонентів MAPE-K, а саме моніторингу, аналізу, планування, виконання та знань. Зауважимо, що компоненти моніторингу та виконання можуть бути пов’язані з підсистемою для визначення складних ієрархічних архітектур керування, де цикл MAPE-K контролює інші цикли MAPE-K.

Взаємодія моделює зв’язок між двома компонентами MAPE-K. Абстрактна взаємодія визначає контекст і ціль. Перший вказує на компонент, який ініціює взаємодію, тоді як другий вказує на компонент призначення. Відповідно до позначення, використаного в [22], Interaction спеціалізується на InterComponentInteraction, IntraComponentInteraction і ReadWriteInteraction. У свою чергу, IntraComponentInteraction додатково спеціалізується на DelegationInteranction і CoordinationInteranction. Зокрема, InterComponent Interaction моделює взаємодію між різними типами компонентів MAPE-K, тоді як IntraComponentInteraction моделює взаємодію між компонентами MAPE-K одного типу, CoordinationInteraction моделює двонаправлену взаємодію, яка використовується, коли два компоненти MAPE-K повинні координувати кожен інший, DelegationInteranction моделює односпрямовану взаємодію, а ReadWriteInteraction представляє взаємодію між компонентом MAPE і Knowledge.Для вираження обмежень на моделі ми використовуємо мову обмежень об’єктів (OCL - Object Constraint Language) [14], декларативну мову для опису правил, що застосовуються до моделей UML. Зокрема (додаток А), мета-модель архітектури керування MAPE-K визначає п’ять правил OCL або, використовуючи термінологію OCL, інваріантів. Щоб модель була дійсною, усі її інваріанти OCL повинні виконуватись .

Наприклад, на рисунку 3.8 показано інваріант OCL hasAtLeastOneMapeCompenent (додаток А). Зокрема, інваріант визначає, що самоадаптивний блок повинен містити принаймні один компонент MAPE-K.

```
1  invariant hasAtLeastOneMapeKComponent:
2          self.monitor <> null or
3          self.analyze <> null or
4          self.plan <> null or
5          self.execute <> null or
6          self.knowledge <> null;
```

Рисунок 3.8 – Інваріант hasAtLeastOneMapeKComponent

3.3.2. Середовище моделювання архітектури керування

DECOR надає графічне середовище моделювання архітектури керування (CAME), засноване на мета-моделі архітектури керування MAPE-K, що дозволяє вказувати архітектури керування як набір елементів, кожен з яких реалізує певний компонент MAPE-K.

CAME реалізовано як плагін Eclipse і базується на Eclipse Modeling Framework і Graphical Modeling Project, які забезпечують відповідні механізми для розробки інструментів розробки на основі моделі.

Враховуючи керовану систему для керування, CAME дозволяє інженерам розробляти архітектуру керування, вказуючи набір компонентів MAPE-K та їх взаємодію або, як альтернативу, використовуючи готові до використання шаблони керування [22], доступні в каталозі CAME. Зауважимо, що каталог CAME можна легко розширити шляхом визначення нових шаблонів проектування.

Варто зауважити, що шаблони керування навмисно виключають компонент знань, щоб спростити дизайн і уникнути залежності елемента знань від системних доменів та базової інфраструктури.

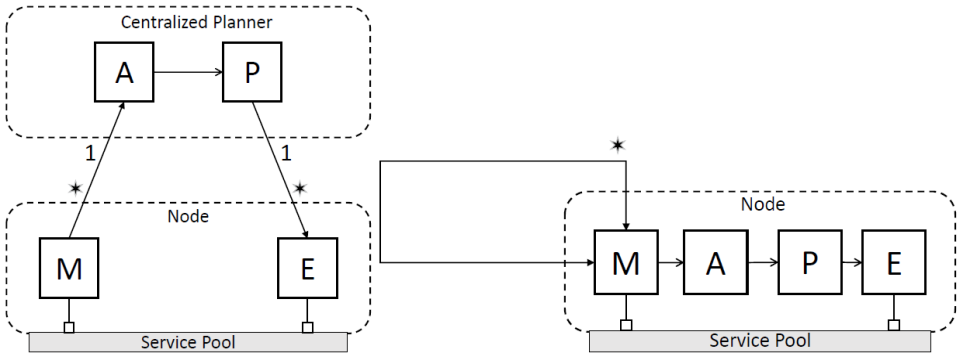
Натомість кінцеві користувачі можуть вільно створювати екземпляри компонента знань, оскільки він найкраще підходить для конкретної області та/або інфраструктури.

Щоб перевірити, чи конкретна архітектура керування MAPE-K сумісна з уже визначеним шаблоном, CAME використовує специфічні для шаблону правила OCL.

Наприклад, на рисунку 3.10 показано інваріант OCL exactlyOneMaster шаблону Master/Slave (рис. 3.10a). Цей інваріант обмежує наявність лише одного головного компонента в архітектурі керування. Зокрема, інваріант отримує головні компоненти з набору підсистем (див. рядки 3 і 4) і обмежує існування точно одного головного.

```
1  invariant exactlyOneMaster:
2   if(type = PatternType::MasterSlave)
3      then self.subsystems->
4           select(oclIsKindOf(Master))->size() = 1
5   else true
6   endif;
```

Рисунок 3.9 – Інваріанти exactlyOneMaster



(a) Master/Slave **(b)** Information Sharing

Рисунок 3.10 – Приклад шаблонів MAPE-K

Потрібно звернути увагу, що наразі CAME не надає інженерам можливості моделювати поведінку компонентів MAPE (тобто, як здійснювати моніторинг, як аналізувати дані тощо). Фактично, хоча мета-модель архітектури керування MAPE-K дозволяє визначати компоненти та з'єднувачі, що реалізують архітектуру керування, конкретну поведінку змодельованих компонентів має вказувати розробник. Майбутнє розширення DECOR дозволить формально вказати поведінку компонентів MAPE.

Далі ми покажемо два можливих шаблони для сценарію складання служби, а саме шаблон Master/Slave і шаблон обміну інформацією.

Шаблон Master/Slave (рис. 3.10a) моделює централізоване управління, де єдиний централізований об'єкт відповідає за аналіз даних і планування стратегії адаптації для всієї системи. У цьому сценарії вузол відповідає за моніторинг розміщених служб і виконання запланованої стратегії адаптації.

Архітектура обміну інформацією (рис. 3.10b) моделює децентралізоване керування (також показано на рис. 3.8). У цій архітектурі керування взаємодія

передбачає лише вузли (тобто без зовнішнього планувальника). Зокрема, інформація спільного використання моделює випадок, коли зібрана інформація про статус послуг розподіляється між різними циклами MAPE-K, що дозволяє здійснювати локальний аналіз, планування та виконання.

3.4 Оцінка архітектур керування

Як зазначено в розділі 3.1, оцінка архітектури керування здійснюється шляхом спільного моделювання представлення моделі з кількома парадигмами. Мультимодель — це модель, що складається з кількох взаємодіючих моделей [14] (тобто архітектури керування та керованих систем у нашому випадку), кожна з яких використовує свій власний формалізм для визначення об’єктів та операцій у різних просторових і часових масштабах. Таким чином, спільне моделювання стосується здатності моделювати багатопарадигмальні моделі за допомогою різних механізмів моделювання, по одному для кожної моделі [17]. Спільне моделювання одночасно виконує ці моделі, дозволяючи обмінюватися інформацією між ними.

Середовище спільного моделювання DECOR (CoSE) включає механізм спільного моделювання для спільного моделювання моделі керованої системи та архітектури керування, розроблений за допомогою інструменту моделювання керованої системи та DECOR CAME відповідно (рис. 3.11). CoSE реалізує різні стратегії моделювання, дозволяючи оцінювати різні показники атрибутів якості: архітектуру керування та моделі керованої системи можна моделювати окремо, щоб перевірити їх власні властивості окремо, або разом, щоб оцінити загальну поведінку системи.

Моделі спільно моделюються шляхом використання MECSYCO [16], механізму спільного моделювання, що обробляє неоднорідність моделей і забезпечує механізми для синхронізації та зв’язку між моделями. MECSYCO використовує агенти та артефакти для опису та моделювання складних систем як MAS.

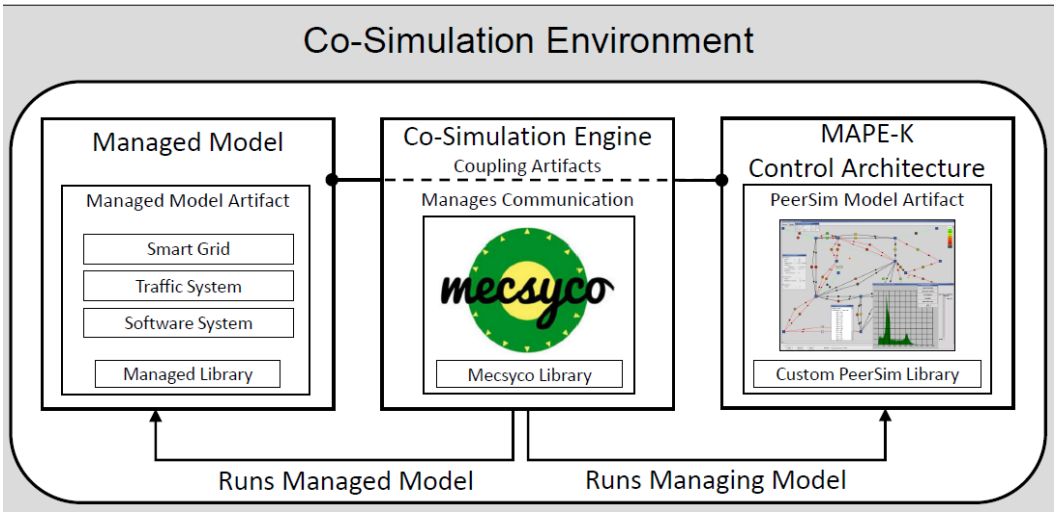


Рисунок 3.11 – Середовище спільного моделювання

Як описано в розділі 3.2, у DECOR усі моделі, що складають мультипарадигмальну модель, повинні відповідати формалізму системи дискретних подій (DEVS - Discrete EVent System). Формалізм DEVS накладає на моделі такі припущення:

– вхідні дані моделей не можуть негайно (з точки зору часу моделювання) генерувати вихідні дані;

– моделі мають відповідати деяким примітивам (тобто так званому артефакту інтерфейсу).

Останнє припущення передбачає, що симулятори для систем повинні бути сумісні з API артефакту інтерфейсу, що надається механізмом, який використовується співсимулятором для забезпечення зв’язку моделі та синхронізації .

У SAS моделювання керованої системи означає роботу з фізичними або програмними ресурсами (наприклад, системи дорожнього руху, електромережі, інфраструктура хмарних обчислень). Дійсно, середовище спільного моделювання може імітувати будь-яку загальну модель, яка інтерпретується відповідно до формалізму DEVS. Прикладами є фізичні моделі, експортовані зовнішніми інструментами моделювання (наприклад, OpenModelica) як FMU (Functional Mock-up Unit) через FMI (Functional Mock-up Interface) [21],

інструментальний незалежний стандарт для обміну динамічними моделями та виконання спільного моделювання. У цьому випадку обгортка DEVS для компонентів FMU дозволяє правильно інтерпретувати модель, яка буде спільно моделюватися двигуном.

З іншого боку, моделювання архітектури керування стосується розробки логіки адаптації як композиції децентралізованих компонентів MAPE-K, які взаємодіють один з одним через канали зв'язку (тобто з'єднувачі).

У дослідженнях зв'язку та комп'ютерних мереж ці моделі зазвичай створюють через мережеве моделювання. Симулятор мережі є незамінним елементом середовища моделювання для децентралізованої SAS, оскільки він дає змогу: а) моделювати топологію мережі, вказуючи самоадаптивні блоки в мережі та зв'язки між ними; б) моделювати потік додатків (тобто трафік) між самоадаптивними блоками; в) для надання показників продуктивності мережі як результат.

Представлення моделі архітектури керування, визначене CoSE, спирається на PeerSim, масштабований імітатор однорангової мережі. Щоб інтегрувати PeerSim у CoSE, ми повністю оновили частину симулятора PeerSim. Зокрема, ми розширили PeerSim, визначивши API, який сумісний із примітивами, які вимагаються для Co-Simulation Engine.

PeerSim визначає архітектуру керування MAPE-K як набір мережевих самоадаптивних блоків. Інтерпретація моделі керування реалізується за допомогою перетворень моделі в код, де кожен самоадаптивний блок зіставляється з вузлом мережі, а кожен компонент MAPE генерує заглушку мережевого протоколу, яку розробник повинен заповнити.

3.5 Архітектурне керування для децентралізованого навчання

У цьому розділі ми розглянемо архітектуру керування, яку слід застосувати в сценарії складання служби. Узагальнюючи приклад сценарію, наведений у 2 розділі, таким чином абстрагуючись від конкретних характеристик окремого

додатка, ми уявляємо собі розподілену керовану систему, що складається з ряду сервісів, які можуть бути з'єднані один з одним за допомогою відповідного набору зв'язків збірки, з'єднуючи необхідні доступні функції. Ці послуги включають як програмно реалізовані послуги (незалежні та ізольовані обчислювальні блоки, кожен з яких володіє своїми даними), так і віртуалізовані апаратні засоби (наприклад, сховище та обчислювальні ресурси), які пропонуються набором підключених до мережі вузлів. Отже, розглянуті зв'язки збірки представляють абстрактне та уніфіковане уявлення як про функціональні зв'язки між програмними службами, так і про зв'язки розгортання між програмними службами та віртуалізованими ресурсами базової інфраструктури. Зв'язка збірки може бути встановлена між запропонованою та необхідною послугами, лише якщо перша відповідає вимогам, визначеним останньою, що стосуються загалом як типу наданої послуги (наприклад, обчислювальної послуги), так і контексту, де вона має бути надана (наприклад, поблизу певного місця).Для ілюстрації нижня частина рисунка 3.12 зображує можливий екземпляр загального сценарію, який ми передбачаємо, що складається з чотирьох програмно реалізованих послуг S2, S6, S8 і S20 і трьох вузлів Node 1, Node 2 і Node 3. Хоча S2 прив'язаний до S6 і S8 для вирішення деяких своїх функціональних залежностей (як показано пунктирними стрілками), він також має деякі залежності, які ще не розв'язані.

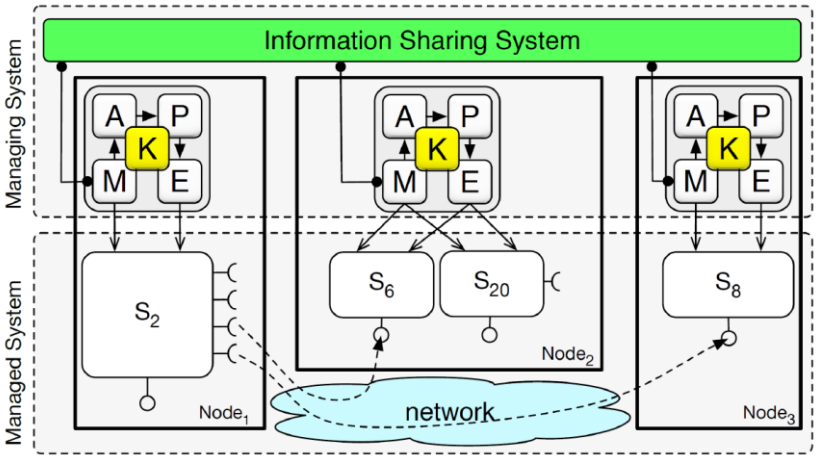


Рисунок 3.12 – Представлення передбачуваної архітектури

Наша мета полягає в тому, щоб такі системи могли самостійно керувати як початковою конструкцією збірки, так і її самоадаптацією відповідно до можливих змін, які можуть відбутися (тобто додавання чи видалення служб чи вузлів або зміна їх QoS атрибутів). Отже dQAS для сценарію складання служби можна сформулювати таким чином: «система повинна мати можливість створювати сервісну збірку, що забезпечує хороший рівень QoS, і підтримувати його з часом, незважаючи на мінливість середовища».

Щоб виконати цю вимогу в динамічному сценарії, який ми розглядаємо, де служби можуть мати кілька залежностей, які потрібно розв’язати, і характеризуються декількома залежними від навантаження атрибутами QoS, ми змушуємо вузли самостійно вивчати правило вибору служби, яке потрібно застосувати, використовуючи RL підхід. Відповідно до цього підходу, користувачц не повідомляють, які дії виконувати, як у більшості форм машинного навчання, а замість цього він повинен визначити, які дії приносять найбільшу винагороду. Ці функції добре відповідають сценарію, на якому ми зосереджуємося, де вузли не знають один одного (і послуги, які вони пропонують) заздалегідь, але виявляють себе динамічно.

Використання розподіленого підходу RL вимагає архітектури, де аналіз і планування виконуються локально на кожному вузлі. Крім того, необхідність розповсюдження локально відстежуваної інформації через систему вимагає архітектури керування, здатної підключати дії локального моніторингу. З цією метою, щоб реалізувати архітектуру керування, ми дотримуємося шаблону обміну інформацією [22], де кожен вузол самостійно адаптується локально, реалізуючи власний цикл MAPE-K, але вимагає інформації про стан від інших вузлів у системі, щоб керувати ним. Крім обміну інформацією про стан, вузли не зобов’язані координувати інші дії з адаптації, що робить шаблон оптимальним для впровадження систем з окремими особами, які приймають рішення. Таким чином, шаблон обміну інформацією є відповідною архітектурою децентралізованого керування для сценарію складання сервісу, який складається з дуже великого набору слабко пов’язаних компонентів, які вимагають адаптації

для підтримки певного атрибута якості [22]. Цей шаблон підтримує рішення щодо автономної адаптації на кожному вузлі та забезпечує масштабованість завдяки необхідній слабкій координації, обмеженій обміном інформацією про стан.

Інформація, яку передають компоненти Monitor, головним чином стосується, у нашому випадку, служб, розміщених на кожному вузлі, а також їхніх функціональних і нефункціональних властивостей. Відповідно до нашої точки зору децентралізації, ми припускаємо, що компоненти Monitor використовують для цієї мети якийсь відповідний децентралізований протокол для публікації та виявлення послуг у поширених обчислювальних середовищах. Потрібно звернути увагу, що в сценарії складання служби вибір методики навчання передбачає прийняття певної архітектури керування. Тобто на діяльність з ідентифікації та проектування (рис. 3.2) впливає алгоритм адаптації та дизайн. Це відповідає тенденції, яка стверджує, що проектування даних і алгоритмів є важливими загальносистемними архітектурними проблемами майбутніх програмних систем. Таким чином, не потрібно експериментувати з різними архітектурами керування, а треба використовувати DECOR для отримання показників якості атрибутів для вибраної архітектури. Отже, розробка та оцінка архітектур керування — це грандіозний виклик: інженери повинні розглянути багато різних і взаємозалежних параметрів проекту, щоб розробити архітектуру керування, здатну вчасно виконувати правильні адаптації та змусити систему відповідати поставленим цілям. Щоб знайти рішення цієї проблеми, потрібен підхід, який підтримує діяльність з проектування та реалізації з можливостями оцінки архітектурних рішень, аргументації та прийняття обґрунтованого вибору дизайну.

Висновки до розділу 3

Отже, в цьому розділі запропоновані архітектурні рішення для розробки SAS: як має бути структурована архітектура керування і як архітектура керування, прямо

чи опосередковано, впливає на якість системи. З цією метою ми представляємо DECOR, структуру міркувань, що стосується розробки процесу проектування та оцінки архітектур керування. Ми розробляємо допоміжний інструмент, що об’єднує середовище моделювання та платформу моделювання, який можна використовувати для проектування сервісних вузлів. З цією метою в цьому розділі ми представили DECOR, засновану на моделі структуру міркувань для моделювання та оцінки децентралізованих архітектур керування. Зокрема, DECOR надає підтримку моделювання для обґрунтування властивостей як в архітектурі керування, так і в керованій системі та середовище моделювання для визначення архітектур керування як кількох і взаємодіючих циклів MAPE-K.

ВИСНОВКИ

В кваліфікаційній роботі розглянуто моделі та методи міркувань і навчання на основі даних для побудови самоадаптивних інформаційних децентралізованих систем. В роботі було проведено систематичний огляд літератури про навчання в децентралізованих SAS, який надає комплексне уявлення про використання моделей навчання та методів, що використовуються для розробки адаптації в децентралізованих системах. На основі цього аналізу ми запровадили системний підхід до обґрунтування вибору дизайну та шаблонів SAS на основі навчання. Обґрунтування виконується за допомогою нового застосування методологій, керованих даними, таких як кластеризація, аналіз множинної відповідності та дерева рішень. Ми застосували міркування до сценарію складання сервісу, щоб витягти рекомендації щодо проектування.

Досліджено структуру фрейморку DECOR: структуру міркування для моделювання та оцінки децентралізованих архітектур керування. DECOR забезпечує підтримку моделювання для обґрунтування цікавих властивостей як в архітектурі керування так і в керованій системі, середовище моделювання для визначення архітектур керування як кількох взаємодіючих циклів MAPE-K, а також середовище спільного моделювання для моделювання як децентралізованої архітектури керування, так і керована система та оцінка цікавих атрибутів якості. Ми використовували фреймворк для розробки та оцінки додатків складання сервісів.

Представлено повністю децентралізований самоадаптивний підхід до збирання сервісу, основними функціями якого є: самоадаптація, тобто здатність працювати автономно, онлайн-навчання (тобто здатність динамічно навчатися на досвіді), усвідомлення QoS (тобто включення вимог до QoS як рушійних сил для самостійного збирання), масштабованість (тобто здатність впоратися з великою кількістю послуг) та стійкість (тобто здатність підтримувати безперервність надання послуг у разі неочікуваних змін у кількості та/або якості

послуг). Також представлено мотиваційний сценарій, що описує клас додатків, опрацьовано і виявлено набір вимог, які повинні бути адресовані розробці цих програм. Ми зіставили визначені вимоги з простором дизайну SAS, щоб дослідити розміри дизайну, пов’язані з самоадаптацією. Аналіз підкреслює необхідність самоадаптації та децентралізованого керування на основі ML як основних блоків системи для забезпечення стійкості та врахування масштабованості та залежних від навантаження атрибутів QoS.

Основним елементом запропонованого підходу є комбіноване використання керованого QoS RL для подолання нестачі глобальних знань і прийняття децентралізованої архітектури керування обміном інформацією MAPE-K, яка дозволяє масштабоване та гнучке поширення інформації.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang. Deep learning with differential privacy. In Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, CCS ’16, page 308–318, New York, NY, USA, 2016.

2. N. Abbas and J. Andersson. Architectural reasoning for dynamic software product lines. In Proceedings of the 17th International Software Product Line Conference Co-Located Workshops, SPLC ’13 Workshops, page 117–124, 2013.

3. N. Abbas, J. Andersson, and D. Weyns. Asple: A methodology to develop self-adaptive software systems with systematic reuse. Journal of Systems and Software, 167:110626, 2020.

4. S. Abdallah and V. Lesser. Multiagent reinforcement learning and self-organization in a network of agents. In AAMAS’07, pages 39:1–39:8. ACM, 2007.

5. S. Aghabozorgi, A. S. Shirkhorshidi, and T. Y. Wah. Time-series clustering – a decade review. Information Systems, 53:16 – 38, 2015.

A. Ali-Eldin, J. Tordsson, and E. Elmroth. An adaptive hybrid elasticity controller for cloud infrastructures. In 2012 IEEE Network Operations and Management Symposium, pages 204–212, April 2012.

6. N. K. Altshuler and D. Sarne. Modeling assistant’s autonomy constraints as a means for improving autonomous assistant-agent design. In AAMAS’18, pages 1468–1476, 2018.

7. F. Alvares, E. Rutten, and L. Seinturier. A domain-specific language for the control of selfadaptive component-based architecture. Journal of Systems and Software, 130:94–112, 2017.

8. G. Anders, A. Schiendorfer, F. Siefert, J.-P. Steghöfer, and W. Reif. Cooperative resource allocation in open systems of systems. ACM Trans. Auton. Adapt. Syst., 10(2):11:1–11:44, June 2015.

9. C. Apté and S. Weiss. Data mining with decision trees and decision rules. Future Generation Computer Systems, 13(2):197 – 210, 1997.

10. P. Arcaini, R. Mirandola, E. Riccobene, and P. Scandurra. Model-based testing for mape-k adaptation control loops. In 2020 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW), pages 43–51, 2020.

11. P. Arcaini, R. Mirandola, E. Riccobene, and P. Scandurra. Msl: A pattern language for engineering self-adaptive systems. Journal of Systems and Software, 164:110558, 2020.

12. P. Arcaini, E. Riccobene, and P. Scandurra. Formal design and verification of self-adaptive systems with decentralized control. ACM Trans. Auton. Adapt. Syst., 11(4), Jan. 2017.

13. B. Babcock, S. Babu, M. Datar, R. Motwani, and J. Widom. Models and issues in data stream systems. In Proceedings of the Twenty-First ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS ’02, page 1–16, New York, NY, USA, 2002.

14. T. Ball, J.-M. Kim, A. A. Porter, and H. P. Siy. If your version control system could talk. In ICSE Workshop on Process Modelling and Empirical Studies of Software Engineering, volume 11, 1997.

15. M. C. Ballandies, M. M. Dapp, and E. Pournaras. Decrypting distributed ledger designtaxonomy, classification and blockchain community evaluation. arXiv preprint arXiv:1811.03419, 2018.

16. L. Bass, J. Ivers, M. Klein, and P. Merson. Reasoning frameworks. Technical Report CMU/SEI2005-TR-007, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA, 2005.

17. M. Beetz, A. Kirsch, and A. Muller. Rpllearn: Extending an autonomous robot control language to perform. In AAMAS’04, pages 1022–1029. IEEE, 2004.

18. M. Bernardo and J. Hillston. Formal Methods for Performance Evaluation, volume 4486. Springer, 2007.

19. N. Bicocchi, M. Mamei, A. Prati, R. Cucchiara, and F. Zambonelli. Pervasive self-learning with multi-modal distributed sensors. In SASO'08, pages 61–66, 2008.

20. T. Blochwitz, M. Otter, J. Åkesson, M. Arnold, C. Clauss, H. Elmqvist, M. Friedrich, A. Junghanns, J. Mauss, D. Neumerkel, H. Olsson, and A. Viel. Functional mockup interface 2.0: The standard for tool independent exchange of simulation models. In Proceedings of the 9th International Modelica Conference, pages 173–184. The Modelica Association, 2012.

21. P. Bock. The emergence of artificial intelligence: Learning to learn. AI Magazine, 6(3):180, Sep. 1985.

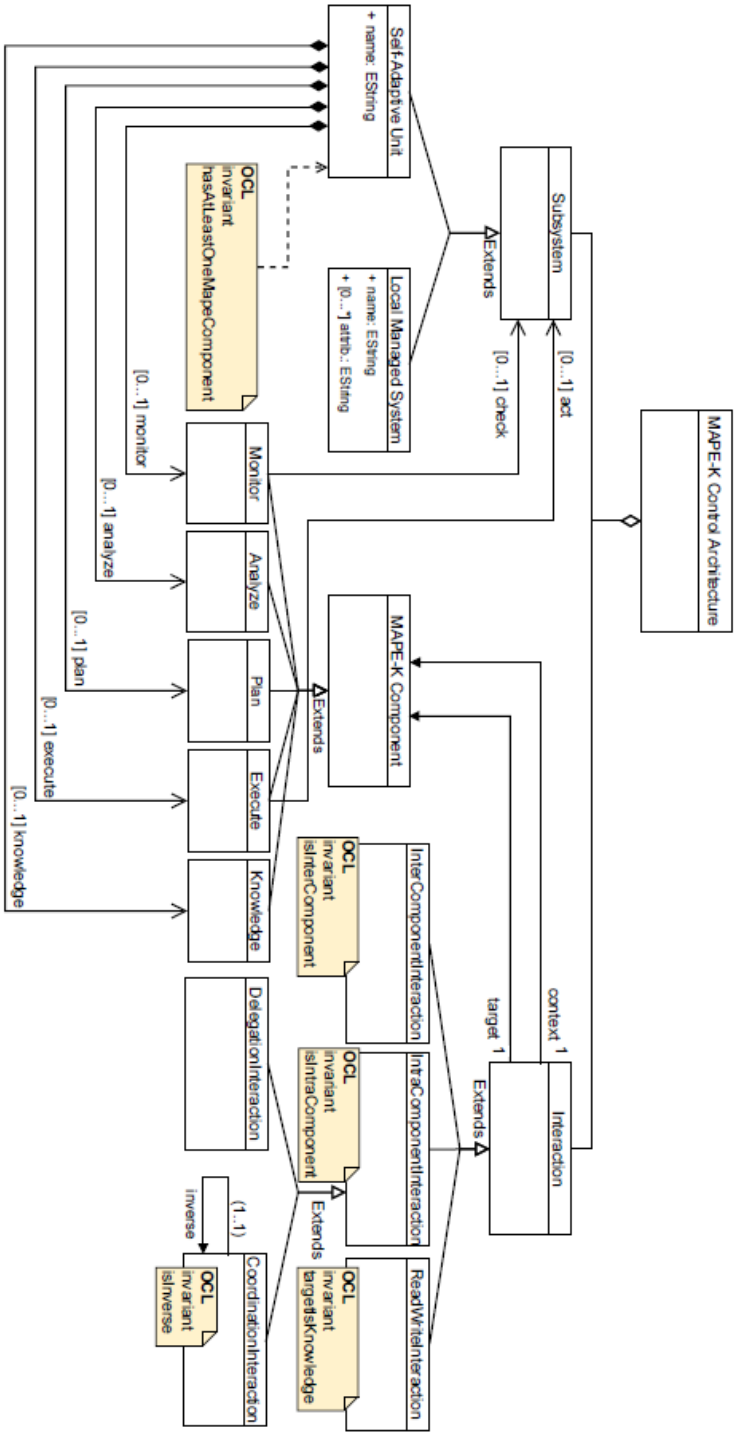
22. C. Bolchini, M. Carminati, A. Miele, and E. Quintarelli. A framework to model self-adaptive computing systems. In 2013 NASA/ESA Conference on Adaptive Hardware and Systems (AHS-2013), pages 71–78, 2013.

23. Boronat, A. Knapp, J. Meseguer, and M. Wirsing. What is a multi-modeling language? In International Workshop on Algebraic Development Techniques, pages 71–87. Springer, 2008.

ДОДАТОК

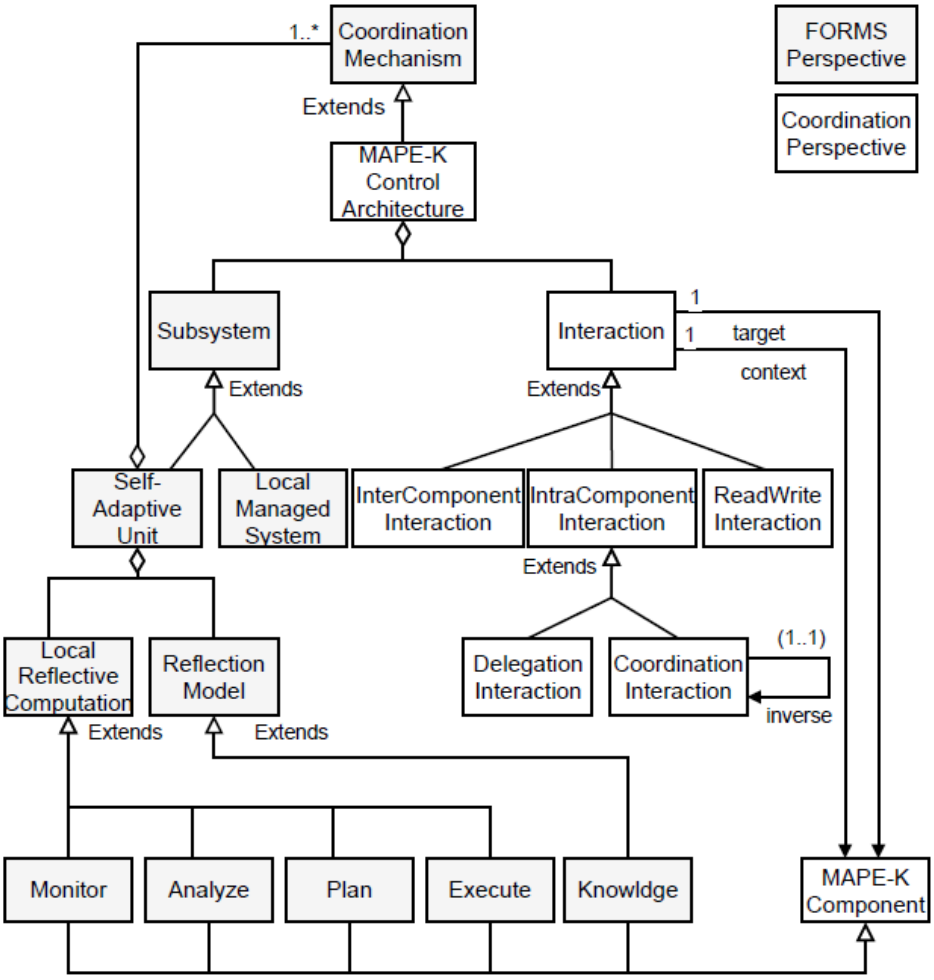
Додаток А

Метамодель архітектури керування MAPE-K



Додаток Б

Зв’язок між FORMS і мета-моделлю архітектури керування MAPE-K



метадані

Заголовок

Впровадження адаптивних моделей у задачах децентралізованого навчання

Автор

Сергієвич Н.Р. Науковий керівник / Експерт

підрозділ

King Danylo University

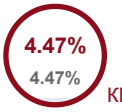
Тривога

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.

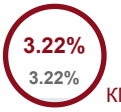
Заміна букв	ß	0
Інтервали	A→	0
Мікропробіли	·	0
Білі знаки	ß	0
Парафрази (SmartMarks)	a	15

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

7554
Кількість слів

57586
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	http://repository.ukd.edu.ua/bitstream/handle/123456789/388/%D0%94%D0%B8%D0%BF%D0%BB%D0%BE%D0%BC%D0%BD%D0%B0%20%D1%80%D0%BE%D0%B1%D0%BE%D1%82%D0%B0%20%D0%9B%D0%B8%D1%82%D0%B2%D0%B0%D0%BA.pdf?sequence=1	72	0.95 %
2	Імплементація моделей адаптивності в задачах децентралізованого навчання [redacted] 1/7/2024 King Danylo University (King Danylo University)	55	0.73 %
3	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	31	0.41 %