ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНОЇ ІНЖЕНЕРІЇ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «СИСТЕМА КЕРУВАННЯ Й КОНТРОЛЮ МЕРЕЖЕВИМИ ПРОЦЕСАМИ»

на здобуття освітнього ступеня магістра
зі спеціальності 123 Комп'ютерної інженерії
(код, найменування спеціальності)
освітньо-професійної програми Комп'ютерні системи та мережі
(назва)

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____ *Ростислав СЛЄПЧУК*
(підпис) *Ім'я, ПРІЗВИЩЕ здобувача*

Виконав: здобувач вищої освіти гр. КСДМ-61
Ростислав СЛЄПЧУК
Ім'я, ПРІЗВИЩЕ
Керівник: Кирило СОКОЛОВ
науковий ступінь, вчене звання Ім'я, ПРІЗВИЩЕ
Рецензент:
науковий ступінь, вчене звання Ім'я, ПРІЗВИЩЕ

Київ 2025

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
Навчально-науковий інститут Інформаційних технологій

Кафедра 123 Комп'ютерної інженерії
Ступінь вищої освіти Магістр
Спеціальність 123 Комп'ютерної інженерії
Освітньо-професійна програма Комп'ютерні системи та мережі

ЗАТВЕРДЖУЮ
Завідувач кафедрою Наталія ЛАЩЕВСЬКА
Ім'я, ПРІЗВИЩЕ
«_____» _____ 2025р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Слєпчук Ростислав Віталійович
(прізвище, ім'я, по батькові здобувача)

1. Тема кваліфікаційної роботи: Система керування й контролю мережевими процесами керівник кваліфікаційної роботи Кирило СОКОЛОВ,
(Ім'я, ПРІЗВИЩЕ, науковий ступінь, вчене звання)
затверджені наказом Державного університету інформаційно-комунікаційних технологій від «30» 10 2025р. № 467
2. Строк подання кваліфікаційної роботи «_____» _____ 2025р.
3. Вихідні дані до кваліфікаційної роботи: Науково-технічна література, методи системного аналізу, теорії множин та математичної логіки, онтологічного інжинірингу та принципи побудови активних мереж.

Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Дослідження специфіки моделювання та методологічних підходів до побудови онтологій у системах управління мережевою інфраструктурою
2. Математична модель та обґрунтування архітектурних рішень онтологічної системи для управління та моніторингу мережевих процесів
3. Результати моделювання та практична реалізація системи управління комп'ютерними мережами
4. Перелік ілюстративного матеріалу: *презентація*
5. Дата видачі завдання «_____» _____ 2025р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Збір даних	25.09-24.10.2025	Виконано
2	Обробка матеріалу	25.10-05.11.2025	Виконано
3	Розробка теоретичної частини	06.11-10.11.2025	Виконано
4	Розробка системи	11.11-15.11.2025	Виконано
5	Висновки за результатами аналізу	15.11-19.11.2025	Виконано
6	Оформлення пояснювальної записки	19.11-21.11.2025	Виконано
7	Розробка презентації	21.11-29.11.2025	Виконано
8	Передзахист	01.12-02.12.2025	Виконано
9	Здача в деканат	15.12-18.12.2025	Виконано

Здобувач вищої освіти _____ Ростислав СЛЄПЧУК
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

Керівник кваліфікаційної роботи _____ Кирило СОКОЛОВ
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня магістра: 66 стор., 18 рис., 17 джерел.

Мета дослідження – підвищення ефективності та надійності функціонування комп'ютерних мереж шляхом розробки та впровадження системи керування й контролю мережевими процесами, що базується на онтологічному моделюванні та технологіях активних мереж.

Об'єкт дослідження – процеси керування, моніторингу та діагностики станів у гетерогенних комп'ютерних мережах.

Предмет дослідження – методи та засоби формалізації мережевих процесів, трафіку та аномалій з використанням онтологічних моделей та архітектурних рішень на базі інтелектуальних агентів.

Короткий зміст роботи: У кваліфікаційній роботі проведено аналіз сучасних підходів до побудови систем управління мережевою інфраструктурою. Обґрунтовано доцільність використання онтологій для семантичного опису мережевих процесів. Розроблено математичну модель системи, що включає формальний лексикон, граматику та семантику мови опису. Спроектовано архітектуру системи керування, яка охоплює структурні компоненти, моделі трафіку та розподілу ресурсів. Формалізовано динамічні аспекти функціонування мережі через механізми подій та аномалій. Запропоновано гібридну схему управління з використанням централізованого модуля логічного виведення та розподілених програмованих агентів.

Результати: Розроблена система забезпечує автоматизоване виявлення складних аномалій, дозволяє корелювати розрізнені події для визначення їх першопричин та створює основу для проактивного керування мережевими процесами.

КЛЮЧОВІ СЛОВА: СИСТЕМА КЕРУВАННЯ, МЕРЕЖЕВІ ПРОЦЕСИ, ОНТОЛОГІЯ, OWL DL, МОНІТОРИНГ, ТРАФІК, ПОДІЇ, АНОМАЛІЇ, АКТИВНІ МЕРЕЖІ, ЛОГІЧНЕ ВИВЕДЕННЯ.

ABSTRACT

Textual part of the Master's qualification paper: 66 pages, 18 figures, 17 references.

The aim of the research is to improve the efficiency and reliability of computer network functioning by developing and implementing a system for managing and controlling network processes based on ontological modeling and active network technologies.

The object of the research is the processes of management, monitoring, and state diagnostics in heterogeneous computer networks.

The subject of the research is the methods and means of formalizing network processes, traffic, and anomalies using ontological models and architectural solutions based on intelligent agents.

Summary: The qualification paper analyzes modern approaches to building network infrastructure management systems. The feasibility of using ontologies for the semantic description of network processes is substantiated. A mathematical model of the system has been developed, which includes a formal lexicon, grammar, and semantics of the description language. The architecture of the management system has been designed, covering structural components, traffic models, and resource allocation. The dynamic aspects of network functioning have been formalized through event and anomaly mechanisms. A hybrid management scheme using a centralized logical inference module and distributed programmable agents is proposed.

Results: The developed system ensures automated detection of complex anomalies, allows correlating disparate events to determine their root causes, and creates a basis for proactive management of network processes.

KEYWORDS: MANAGEMENT SYSTEM, NETWORK PROCESSES, ONTOLOGY, OWL DL, MONITORING, TRAFFIC, EVENTS, ANOMALIES, ACTIVE NETWORKS, LOGICAL INFERENCE.

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут Інформаційних технологій

ПОДАННЯ
ГОЛОВІ ДЕРЖАВНОЇ ЕКЗАМЕНАЦІЙНОЇ КОМІСІЇ
ЩОДО ЗАХИСТУ КВАЛІФІКАЦІЙНОЇ РОБОТИ
на здобуття освітнього ступеня магістра

Направляється здобувач Слєпчук Р. В. до захисту кваліфікаційної роботи
(прізвище та ініціали)
за спеціальністю 123 Комп'ютерна інженерія
(код, найменування спеціальності)
освітньо-професійної програми Комп'ютерні системи та мережі
(назва)
на тему: «Система керування й контролю мережевими процесами».
Кваліфікаційна робота і рецензія додаються.

Директор ННІ _____ Катерина НЕСТЕРЕНКО
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

Висновок керівника кваліфікаційної роботи

Здобувач Слєпчук Р. В. показав відмінну теоретичну підготовку, уміння володіти та розуміння новими комп'ютерними технологіями, користуватися навчальною і науково-технічною літературою. Працюючи над завданнями, які були поставлені керівником, проявив ініціативність, сумлінність та хист до інженерної роботи.
Все це дозволяє оцінити кваліфікаційну роботу студента Слєпчука Ростислава Віталійовича на оцінку «відмінно», та присвоїти їй кваліфікацію «магістр з комп'ютерної інженерії».

Керівник кваліфікаційної роботи _____ Кирило СОКОЛОВ
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

«__» _____ 2025 року

Висновок кафедри про кваліфікаційну роботу

Кваліфікаційна робота розглянута. Здобувач *Слєпчук Р. В.* допускається до захисту даної роботи в Екзаменаційній комісії.

Завідувач кафедрою Комп'ютерної інженерії _____ Наталія ЛАЩЕВСЬКА
(назва) *(підпис)* *(Ім'я, ПРІЗВИЩЕ)*

ВІДГУК РЕЦЕНЗЕНТА
на кваліфікаційну магістерську роботу

здобувача вищої освіти Слєпчук Ростислав Віталійович
(прізвище, ім'я, по батькові)
на тему: «Система керування й контролю мережевими процесами»

Актуальність:

Кваліфікаційна магістерська робота присвячена вирішенню важливої науково-технічної проблеми — підвищенню ефективності управління сучасними гетерогенними комп'ютерними мережами. В умовах постійного зростання обсягів трафіку та складності інфраструктури, традиційні методи моніторингу часто виявляються неефективними через нездатність виявляти причинно-наслідкові зв'язки між збоями. Робота пропонує сучасний підхід до інтелектуалізації процесів керування шляхом використання онтологічного моделювання та методів логічного виведення. Це дозволяє перейти від реактивного реагування на інциденти до проактивної діагностики та автоматизованого усунення кореневих причин аномалій.

Позитивні сторони:
1. У роботі наведено ґрунтовне теоретичне обґрунтування вибору мови веб-онтологій OWL DL як інструменту формалізації, що дозволило автору знайти оптимальний баланс між виразною потужністю моделі та обчислювальною ефективністю алгоритмів логічного виведення.
2. Автор продемонстрував глибоке розуміння предметної області, розробивши детальну ієрархічну структуру класів, яка уніфікує опис різнорідних апаратних та програмних компонентів мережі. Особливої уваги заслуговує розроблена математична модель з чітко визначеними лексиконом та граматикою, що забезпечує логічну стійкість системи.
3. Вагомим практичним результатом є запропонована архітектура системи керування на базі активних мереж, яка поєднує централізований модуль Reasoner та розподілені програмовані агенти. Реалізована модель кореляції подій та аномалій дозволила досягти скорочення часу ідентифікації збоїв на 35–40%, що підтверджує ефективність обраного підходу.

Недоліки:
1. У роботі основний акцент зроблено на логічній структурі онтології та архітектурі взаємодії компонентів, проте недостатньо детально висвітлено питання кібербезпеки при використанні активних пакетів, що містять виконуваний код, що є критично важливим аспектом для реальних мереж.
2. Бажано було б навести детальнішу оцінку обчислювальної складності запропонованих алгоритмів логічного виведення при їх масштабуванні на великі корпоративні мережі, оскільки обробка онтологій може створювати значне навантаження на апаратні ресурси сервера керування.

Висновок: *кваліфікаційна бакалаврська робота заслуговує оцінку "добре", а здобувач Слєпчук Р. В. заслуговує присвоєння кваліфікації: магістр з комп'ютерної інженерії.*

Рецензент:
науковий ступінь, вчене звання _____ *підпис* _____ *Ім'я, ПРІЗВИЩЕ*

ЗМІСТ

ВСТУП..... 10
1 ДОСЛІДЖЕННЯ СПЕЦИФІКИ МОДЕЛЮВАННЯ ТА МЕТОДОЛОГІЧНИХ ПІДХОДІВ ДО ПОБУДОВИ ОНТОЛОГІЙ У СИСТЕМАХ УПРАВЛІННЯ МЕРЕЖЕВОЮ ІНФРАСТРУКТУРОЮ.... 12
1.1 Аналітичний огляд ключових категорій та теоретико-методологічних засад процесу побудови онтологічних моделей..... 12
1.2 Логіко-математичні основи та формалізація онтологій для управління знаннями..... 16
1.3 Концепція Reusability в онтологічному інжинірингу: підходи, переваги та виклики..... 17
1.4 Типологізація та класифікаційні ознаки онтологічних моделей..... 19
1.5 Аналітичний огляд фундаментальних принципів та методологічних підходів до проектування онтологічних моделей..... 24
1.6 Методологічні підходи, критерії та метрики оцінювання якості онтологічних моделей..... 27
2 МАТЕМАТИЧНА МОДЕЛЬ ТА ОБҐРУНТУВАННЯ АРХІТЕКТУРНИХ РІШЕНЬ ОНТОЛОГІЧНОЇ СИСТЕМИ ДЛЯ УПРАВЛІННЯ ТА МОНІТОРИНГУ МЕРЕЖЕВИХ ПРОЦЕСІВ..... 30
2.1 Формалізація лексикону предметної області та визначення сигнатури моделі..... 30
2.2 Синтаксичні правила та граматика побудови логічних виразів..... 31
2.3 Синтаксичні обмеження та правила коректного формування термів..... 33
2.4 Структурна організація семантичної моделі онтології..... 35
2.5 Архітектурна організація та структурні компоненти онтології управління комп’ютерними мережами..... 37
3 РЕЗУЛЬТАТИ МОДЕЛЮВАННЯ ТА ПРАКТИЧНА РЕАЛІЗАЦІЯ СИСТЕМИ УПРАВЛІННЯ КОМП’ЮТЕРНИМИ МЕРЕЖАМИ..... 41

3.1 Побудова онтологічної моделі комунікаційної інфраструктури..... 41
3.2 Моделювання процесів передачі трафіку та механізмів розподілу навантаження в комп’ютерній мережі 46
3.3 Діагностика динамічного стану мережі: моделювання подій та виявлення аномалій..... 50
3.4 Організація інфраструктури та інструментальні засоби системи управління і моніторингу 56
ВИСНОВКИ 61
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ..... 63
Додаток А.ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ (Презентація)..... 66

ВСТУП

У сучасному інформаційному просторі комп’ютерні мережі є критично важливим елементом інфраструктури, складність якої постійно зростає. Збільшення обсягів трафіку, різноманітність протоколів та гетерогенність обладнання роблять традиційні методи керування й контролю недостатньо ефективними. Типові засоби моніторингу, що оперують лише числовими метриками, часто не здатні забезпечити глибоке розуміння динамічних процесів, що відбуваються в мережі, особливо в контексті виявлення причинно-наслідкових зв’язків між збоями.

Це зумовлює необхідність переходу до інтелектуальних систем керування, здатних не просто реєструвати інциденти, а й розуміти їх семантику. Впровадження онтологічного підходу дозволяє формалізувати знання про мережеві процеси, створити єдину інформаційну модель та забезпечити автоматизовану підтримку прийняття рішень.

Мета і завдання дослідження. Метою роботи є підвищення ефективності експлуатації мережевої інфраструктури шляхом розробки системи керування й контролю мережевими процесами на основі онтологічної моделі знань.

Для досягнення мети поставлені наступні завдання:

1. Проаналізувати методи моделювання систем управління та обґрунтувати вибір мови OWL DL для формалізації знань.
2. Розробити математичну модель системи, визначивши лексикон, синтаксичні правила та обмеження для уникнення логічних колізій.
3. Побудувати онтологію структурних компонентів мережі, що охоплює апаратні та програмні засоби.
4. Розробити моделі представлення трафіку та механізмів його розподілу по фізичних ресурсах.
5. Формалізувати динаміку мережевих процесів через концепти подій та аномалій, встановивши між ними причинно-наслідкові зв’язки.

6. Розробити архітектуру інструментальних засобів та схему їх взаємодії.

Об’єкт дослідження – процеси функціонування та управління комп’ютерними мережами.

Предмет дослідження – моделі, методи та програмно-апаратні архітектури для реалізації системи керування й контролю мережевими процесами.

Методи дослідження. У роботі використано методи системного аналізу, теорії множин та математичної логіки, онтологічного інжинірингу та принципи побудови активних мереж.

Практичне значення. Запропонована система дозволяє автоматизувати діагностику несправностей, підвищити точність виявлення аномалій та забезпечити гнучке керування мережевими процесами в режимі реального часу.

1 ДОСЛІДЖЕННЯ СПЕЦИФІКИ МОДЕЛЮВАННЯ ТА МЕТОДОЛОГІЧНИХ ПІДХОДІВ ДО ПОБУДОВИ ОНТОЛОГІЙ У СИСТЕМАХ УПРАВЛІННЯ МЕРЕЖЕВОЮ ІНФРАСТРУКТУРОЮ

1.1 Аналітичний огляд ключових категорій та теоретико-методологічних засад процесу побудови онтологічних моделей

У сучасній сфері інформатики та інформаційних технологій поняття «онтологія» набуло фундаментального значення. Воно визначається як строго структурована сукупність термінів, концепцій та понять, що слугують для формалізованого представлення семантики інформаційних доменів. Така репрезентація реалізується за допомогою метаданих простору імен або специфічних елементів конкретної предметної області знань [1].

Сфера застосування онтологій є надзвичайно широкою та охоплює різноманітні напрямки, серед яких ключове місце займають:

— системи штучного інтелекту;

— побудова семантичних мереж (Semantic Web);

— інженерія програмного забезпечення;

— біомедичні обчислення та біоінформатика;

— проєктування архітектури складних комп’ютерних систем різного цільового призначення.

По своїй суті, онтологія виступає універсальною формою подання знань про навколишній світ або певну його, чітко окреслену частину (домен).

Класичним у науковій літературі вважається визначення, сформульоване Т. Грубером у 1993 році, згідно з яким онтологія трактується як «явна специфікація концептуалізації» [1]. Це означає, що онтології пропонують загальний, узгоджений словник для опису сутностей реального світу,

детермінують значення термінів, а також встановлюють логічні зв’язки між ними з різним рівнем формалізації.

Структурна організація онтологій зазвичай базується на ієрархічних таксономіях. До складу онтологічної моделі входять такі базові примітиви моделювання:

1. Класи (поняття) — абстрактні групи об’єктів.
2. Відношення — типи взаємодії між класами.
3. Функції — спеціальні відношення, що визначають елементи.
4. Аксіоми — твердження, що приймаються як істинні (правила).
5. Екземпляри — конкретні представники класів [1].

Практична цінність онтологій яскраво проявляється в системах управління знаннями (Knowledge Management), експертних системах, брокерах знань та механізмах забезпечення інтероперабельності (взаємодії) між різнорідними інформаційними системами. Виходячи з цього, онтологію можна охарактеризувати як ієрархію понять, що пов'язані відношеннями підпорядкування (IS-A), де у більш складних моделях ця структура збагачується набором аксіом для встановлення додаткових семантичних зв'язків.

Впровадження онтологічного підходу для формування спільних баз знань, попри свою ефективність, актуалізує низку методологічних питань, які потребують вирішення в процесі дослідження:

— Яким є ґенеза та походження конкретної онтології?

— Яка методологія має бути покладена в основу її розробки?

— Який інструментарій є оптимальним для процесу її побудови?

— Яких принципів проєктування слід дотримуватися для забезпечення якості моделі?

— Яким чином повинна використовуватися готова онтологія в прикладних системах?

— Які метрики та підходи слід застосовувати для верифікації, валідації та оцінювання розробленої онтології?

У контексті технологій штучного інтелекту онтологія часто ототожнюється з набором формальних термінів, що репрезентують знання. Спираючись на визначення Грубера [1] щодо «явної специфікації концептуалізації», варто уточнити, що під концептуалізацією розуміється абстрактне, спрощене подання світу (модель), яке необхідно відобразити для досягнення певної цілі. Представлення знань, у свою чергу, базується на цій концептуалізації та включає опис об'єктів, понять та зв'язків, що існують між ними. З суто технічної точки зору [2], онтологія розглядається як специфікація типів сутностей, що існують або можуть потенційно існувати в межах досліджуваної предметної області.

Більш формалізований підхід дозволяє представити онтологію як набір імен, що проєктують концепції та типи відношень, які впорядковані (частково впорядковані) за допомогою відношення типу/підтипу. Для побудови онтології управління комп'ютерними мережами доцільно навести найбільш релевантні дефініції:

Означення 1 (Інженерний підхід): Згідно з працями Н. Гуаріно [3], онтологія визначається як інженерний артефакт. Він формується спеціалізованим словником, що використовується для опису певної реальності, та доповнюється набором явних припущень щодо очікуваного значення термінів цього словника.

Означення 2 (Декларативний підхід): Відповідно до джерела [4], онтології розглядаються як свого роду «декларативні абстрактні ієрархії верхнього рівня». Вони містять достатній обсяг інформації для встановлення фундаментальних правил моделювання домену. При цьому онтологія визначає не лише основні терміни та зв'язки (лексику), але й правила їх комбінування для розширення словникового запасу.

Означення 3 (Когнітивний підхід): Дослідники Вілінга та Шрайбер запропонували визначення онтології в ШІ [5] як «теорії того, які сутності можуть існувати в свідомості обізнаного агента». У цьому ракурсі на онтологію можна посилатися як на мета-модель, що описує структуру бази

знань. Вона робить явними відповідності, які використовуються в процесі моделювання, забезпечуючи тим самим можливість повторного використання (reuse) та обміну знаннями.

Означення 4 (Філософсько-логічний підхід): Класичне визначення Грубера [1] було піддане переосмисленню Гуаріно та Джареттою [6]. Вони деталізували семантику терміна «онтологія», виокремивши сім можливих аспектів його вживання в науковій літературі:

1. Як самостійна філософська дисципліна.
2. Як неформальна концептуальна система.
3. Як формальна семантична мережа.
4. Як специфікація концептуалізації.
5. Як представлення концептуальної системи через логічну теорію.
6. Як словник, що використовується в межах логічної теорії.
7. Як конкретизація певної логічної теорії.

У своїх роботах автори зосереджуються на значеннях від 2 до 7, приділяючи особливу увагу критичному аналізу значення 4. Як наслідок, Гуаріно та Джаретта пропонують розглядати онтологію ШІ у двох взаємопов’язаних вимірах:

Як логічна теорія: онтологія — це «спроєктований артефакт, база знань особливого типу, яка може бути прочитана, продана або фізично передана».

Як синонім концептуалізації: онтологія — це «інтенсіональна семантична структура, що кодує неявні правила, які накладають обмеження на структуру певної частини реальності».

Важливим висновком дослідження Гуаріно та Джаретти є твердження про те, що одна й та сама онтологічна теорія може відповідати різним концептуалізаціям, і навпаки — одна й та сама концептуалізація може лежати в основі різних онтологічних теорій, що свідчить про гнучкість онтологічного моделювання.

1.2 Логіко-математичні основи та формалізація онтологій для управління знаннями

Формальне представлення знань у межах онтологічного підходу базується на чітко визначеній архітектурі, яку формують п’ять фундаментальних компонентів. Згідно з класичними підходами до інженерії знань [1], онтологічна модель може бути описана як кортеж елементів: класи (концепти), екземпляри (індивіди), відношення, функції та аксіоми.

Класи (Classes) або Концепти (Concepts)Класи, які в літературі також зустрічаються під назвами «типи» або «універсали», є абстрактними групами сутностей, об’єднаних спільнимихарактеристиками чи властивостями. У широкому розумінні, концепт може репрезентувати будь-який об’єкт предметної області, про який наявна інформація: від фізичних сутностей до абстрактних процесів, таких як опис завдання, алгоритм дії, функціональна стратегія або процес логічного виведення.

Сучасні мови онтологічного моделювання (наприклад, OWL, DAML+OIL) надають інструментарій для інтенсіонального визначення понять через набір їхніх атрибутів. Класичний приклад таксономії: клас «Ссавці» об’єднує живих істот зі спільними біологічними ознаками, водночас виокремлюючи специфічні підкласи (наприклад, «Люди») за унікальними характеристиками, такими як здатність до мовлення[17].

Відношення визначає семантичні зв’язки та способи взаємодії між об’єктами онтології (як класами, так і екземплярами). Відношення структурують предметну область, перетворюючи набір розрізнених понять на зв’язну мережу (граф).

В онтологіях розрізняють різні типи зв’язків: ієрархічні відношення: наприклад, «is-a» (є підкласом) або «part-of» (є частиною); асоціативні відношення: наприклад, бінарний зв’язок «пов’язаний з».

Функції в онтологічному моделюванні розглядаються як специфічний випадок відношень, що мають властивість функціональності. У n-арному відношенні n-й елемент однозначно визначається попередніми (n-1) елементами.

Аксіоми — це формалізовані логічні твердження, які приймаються як істинні без доведення в межах даної моделі. Вони формують базу знань предметної області, задаючи явні правила, обмеження та закономірності.Функціональне призначення аксіом полягає в моделюванні «істинних речень» (sentences that are always true), що дозволяє системам штучного інтелекту виконувати логічне виведення (inference) нових знань на основі існуючих фактів. Класифікація аксіом [7] зазвичай здійснюється відповідно до їх семантичного навантаження (наприклад, аксіоми еквівалентності, диз'юнктивності класів тощо).

Екземпляри (часто звані «індивідами») — це конкретні об'єкти, що є елементами певних класів. Вони моделюють реальні сутності фізичного світу або конкретні дані й представляють екстенсіональну частину онтології (наповнення бази знань).

1.3 Концепція Reusability в онтологічному інжинірингу: підходи, переваги та виклики

Проблема повторного використання (reuse) онтологічних ресурсів залишається одним із ключових напрямків досліджень у контексті розбудови семантичних мереж (Semantic Web) та інженерії знань. У загальному вигляді повторне використання дефініюється як ітеративний процес, у якому наявні формалізовані знання (існуючі онтології або їх фрагменти) виступають у ролі вхідних даних для генерації нових онтологічних структур. На основі аналізу наукових джерел, зокрема робіт [10–13], доцільно виокремити та деталізувати

основні категорії процесів повторного використання онтологій, кожна з яких має специфічну мету та алгоритм реалізації.

Злиття являє собою комплексний підхід, спрямований на фізичну інтеграцію двох або більше вихідних онтологій задля створення єдиної, унікальної цільової онтології. Формування нової моделі, яка б акумулювала та включала повний обсяг знань, що містилися в об'єднуваних джерелах. Процес злиття генерує об'єднану версію оригінальних онтологій, яка містить усю сукупну інформацію. Характерною особливістю цього методу є те, що в результуючій структурі часто нівелюється походження окремих елементів (provenance), тобто не вказується їх приналежність до конкретного вихідного джерела [10]. Цей метод є найбільш релевантним у випадках, коли вихідні онтології описують ідентичні, схожі або такі предметні області, що мають значний ступінь перекриття (overlapping domains).

На відміну від злиття, інтеграція розглядається як процес побудови нової онтології шляхом агрегації, компонування, розширення, спеціалізації або адаптації модулів з інших онтологій [10]. Ключова відмінність полягає в тому, що інтеграція часто оперує онтологіями, які покривають різні (хоча й пов'язані) теми або аспекти предметної області. Це дозволяє створювати складні багатомодульні системи, де кожен компонент відповідає за свій сегмент знань.

Відображення — це процес ідентифікації семантичних відповідностей між сутностями (класами, властивостями, екземплярами) різних онтологій. Цей підхід передбачає збереження автономності вихідних онтологій без створення глобальної, об'єднаної схеми. Натомість, результатом процесу є формування набору формальних тверджень відображення (mapping assertions), які описують типи відношень між сутностями [11]. Основна мета відображення полягає в забезпеченні інтероперабельності — здатності систем обмінюватися даними та повторно використовувати інформацію, знаходячись у гетерогенних семантичних просторах.

Узгодження визначається як процес знаходження відповідностей між концепціями двох онтологій із подальшою модифікацією принаймні однієї з

них для досягнення взаємної сумісності (когерентності) [12]. Для автоматизації цього процесу розроблено широкий спектр програмних засобів (alignment tools), здатних обробляти різнорідні структури даних: схеми реляційних баз даних, XML-схеми, таксономії, ER-моделі (сутність-зв'язок), формальні мови та термінологічні словники.

Поняття версійності вказує на часову динаміку розвитку онтології та наявність кількох варіантів її існування. Нові версії, як правило, є результатом внесення змін до існуючої базової моделі, що призводить до формування ієрархічного «дерева» похідних онтологій. Згідно з джерелом [13], керування версіями — це не просто зберігання копій файлів, а систематизована процедура, що включає:

1. Розпізнавання та ідентифікацію версій
2. Коректне оновлення онтологічних моделей.
3. Відстеження зв'язків та сумісності між різними версіями (lineage tracking).

Цей аспект є критично важливим для забезпечення стабільності систем, що залежать від знань, які постійно еволюціонують.

1.4 Типологізація та класифікаційні ознаки онтологічних моделей

Фундаментальним критерієм для типологізації та класифікації онтологічних моделей виступає поняття онтологічної прихильності (ontological commitment). Цей термін визначає ступінь та характер відповідності онтології певній концептуалізації реальності. Саме можливість розрізнення та ідентифікації різних типів онтологічної прихильності дозволяє побудувати чітку класифікаційну систему онтологій. У сучасній теорії штучного інтелекту загальноприйнятим підходом є диференціація прихильності за трьома ключовими вимірами: завданнями, методами та

предметними доменами, рис.1.1. Такий поділ дозволяє виокремити три базові типи онтологій, кожен з яких має свою специфіку проектування та застосування.

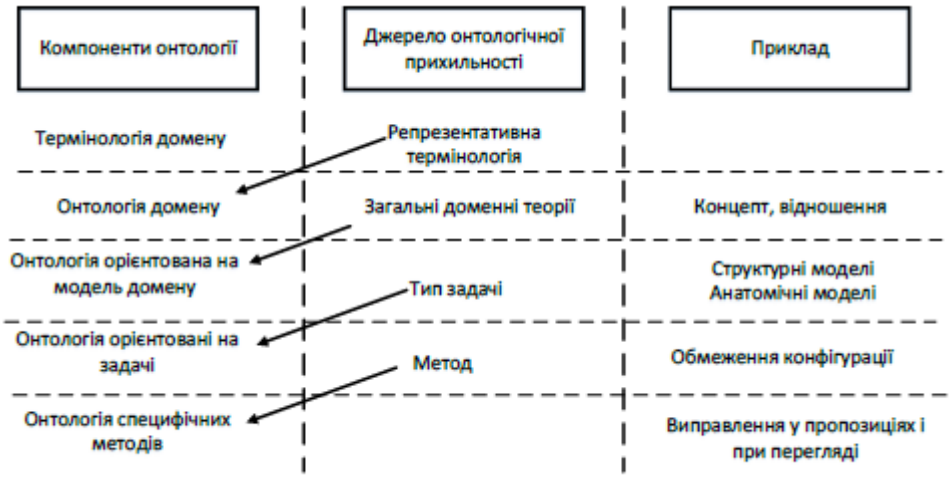


Рисунок 1.1 – Структурна схема типів онтологічної прихильності в контексті класифікації онтологій

Перший тип характеризується прихильністю до завдань (Task commitment). Формулювання цього типу прихильності базується на твердженні, що якщо онтологія визначає сутності, атрибути та зв’язки, які відображають перспективу конкретного завдання на знання предметної області, то вона класифікується як така, що має прихильність до завдань [16]. У цьому контексті завдання визначається як формальна специфікація цілі, що включає опис вхідних даних та необхідних вихідних результатів. Яскравим прикладом такої моделі є онтологія, розроблена для задач діагностики, яка оперує такими специфічними сутностями, як «спостереження», «гіпотези», «симптоми» та «причини», фокусуючись саме на процесі виявлення несправностей, а не на загальному описі об'єкта.

Другий тип визначається прихильністю до методу (Method commitment). Ця категорія охоплює онтології, які визначають сутності та реляції, що виражають специфічну для конкретного методу точку зору на знання. Метод у

даному випадку розглядається як алгоритмічна специфікація того, яким чином може бути виконане певне завдання. Наприклад, онтологія, що обслуговує метод вирішення проблем «запропонувати та переглянути» (Propose-and-Revise) в рамках задач інженерного проектування, буде містити такі характерні сутності, як «запропоноване рішення», «оцінка значення», «обмеження» та «коригування». Така структура знань є жорстко прив'язаною до логіки роботи самого методу.

Третій тип — це онтології з доменною прихильністю (Domain commitment). Така прихильність констатується у випадку, коли онтологія визначає сутності та зв’язки, що є релевантними для конкретної предметної області, незалежно від розв'язуваних задач чи методів. Домен відноситься до загальноприйнятих, усталених фрагментів модельованого реального світу, таких як медицина, юриспруденція, фінанси, математика або соціальна сфера. Наприклад, онтологія судової влади, яка включає такі поняття, як «норма», «акт», «суддя», «правопорушення», є класичним прикладом доменної онтології. Вона надає стандартизований словник концепцій даної галузі, описує їхні взаємозв’язки, діяльність, що відбувається в цій сфері, а також формалізує теорії та базові принципи, які керують цим доменом.

Розвиваючи принципи наведеної класифікації, у роботі [7] запропоновано розширену типологію, яка базується на ідеї контекстуальності знань.

Наведена на рис. 1.2 схема ілюструє комплексну типологію, яка розширює класичне розуміння онтологій, розглядаючи їх крізь призму технологій моделювання та контексту вирішення проблем. Ця класифікація структурує онтологічні ресурси за їхнім функціональним призначенням, виокремлюючи чотири фундаментальні категорії

Згідно з цим підходом, з точки зору технології баз знань, інформацію слід розглядати безпосередньо в ситуації вирішення проблеми. Це дозволяє виділити онтології вмісту (Content ontologies), які часто називають онтологіями для повторного використання знань. Ця категорія є

узагальнюючою і включає в себе онтології завдань, онтології домену, а також загальні (General) онтології, що містять універсальну лексику, пов’язану з абстрактними категоріями: речами, подіями, простором, часом, поведінкою та функціональною причинністю.

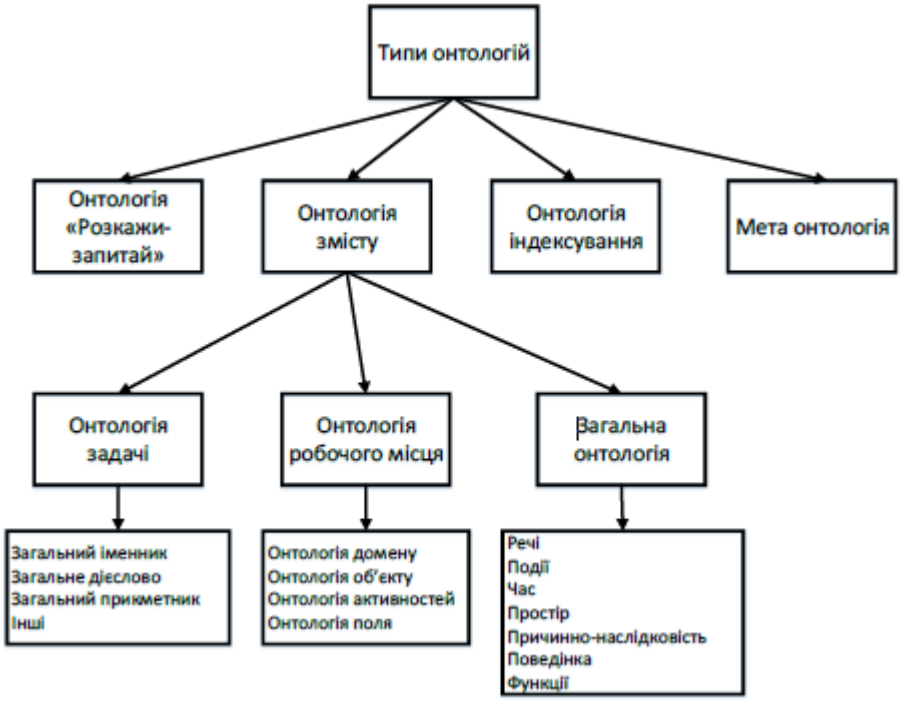


Рисунок 1.2 — Розширена класифікація та доповнення класів онтологій у контексті технологій баз знань

Окремим важливим класом у цій типології є мета-онтології (або основні онтології), які слугують для формального опису самих примітивів представлення знань. Поряд з ними виділяють комунікаційні онтології, або онтології обміну знаннями. Їхнє основне призначення полягає в підтримці спільного використання та трансляції представлених знань у формалізований спосіб між різними агентами. Вони є критично цінними для визначення спільного словника (Shared Vocabulary), що забезпечує семантичну інтероперабельність. Відомим прикладом реалізації підходу обміну знаннями є система Ontolingua. Також дослідники виділяють онтології індексування, які застосовуються в задачах пошуку інформації (Information Retrieval). Метою

впровадження таких онтологій є підвищення показників повноти (recall) та точності (precision) пошуку [9]. Основний механізм їх дії пов'язаний з технікою розширення запиту, що полягає в автоматичному пошуку в онтології термінів, семантично пов’язаних із вихідними словами запиту, та їх додаванні до пошукового рядка.

Крім функціонального призначення, онтології також класифікують за рівнем структурної складності. Можна виділити два полярні типи складності в залежності від широти кола завдань та глибини формалізації. Діапазон варіюється від простих таксономічних структур до високорівневих логічних моделей. Спрощені (легковагові) онтології зазвичай визначаються як ієрархічні структури з базовими класифікаційними характеристиками. Їхнє головне призначення — представлення субсумпційного відношення (відношення «клас-підклас» або IS-A) та інших простих типів зв’язків між поняттями без глибокої аксіоматизації. Такі моделі широко застосовуються в інтернет-пошукових системах, наприклад, у каталогах Yahoo, де структура складається з ієрархії тем без суворого визначення понять. Формальне означення спрощеної онтології детально наведено в джерелі [10].

Другий тип — це важковагові (строго типізовані) онтології, які суттєво відрізняються від спрощених. Вони розробляються з акцентом на точність визначень, принципи організації кожного поняття та наявність семантично строгих зв’язків (обмеження кардинальності, властивості відношень, аксіоми диз’юнктивності тощо). Цільове моделювання світу, яке вимагає глобальної та несуперечливої концептуалізації, потребує саме таких моделей. У міру того, як до моделі додаються нові типи зв’язків, а предметна область стає складнішою, онтології еволюціонують, мігруючи від спрощеного представлення до складнішого, забезпечуючи тим самим вищу точність логічного виведення та надійність системи.

1.5 Аналітичний огляд фундаментальних принципів та методологічних підходів до проектування онтологічних моделей

У сучасному науковому дискурсі онтології визначаються як складні інженерні артефакти, що володіють багаторівневою структурою — логічною, лінгвістичною та таксономічною. Їхня телеологічна сутність полягає в кодуванні формалізованого опису домену, який може охоплювати фактичні, контрфактичні, потенційно можливі або бажані стани системи для вирішення конкретних класів завдань. При цьому критично важливою умовою ефективності онтологічної моделі є її сувора відповідність двом вимірам: специфіці предметної області (домену) та характеру розв'язуваних задач. Доменний вимір дозволяє описати сутності, їхні атрибути та реляції, що формують статичну картину світу. Наприклад, в освітньому домені сутність «Студент» визначається через набір атрибутів (зарахування до університету, наявність наукового керівника, персональні дані). Натомість вимір завдання визначає динамічний аспект використання знань для досягнення мети, як-от пошук колабораторів за спільною науковою темою або узгодження проектних вимог із матрицею компетенцій персоналу.

Процес проектування онтології для гетерогенних комп'ютерних мереж та складних систем є нетривіальним інженерним завданням, яке виступає фундаментом для створення програмних додатків, заснованих на знаннях (Knowledge-Based Systems). У низці досліджень, зокрема [11–13], доведено ефективність онтологічного підходу для задач верифікації моделей, глибокого семантичного аналізу та виявлення логічних неузгодженостей у сценаріях поведінки складних систем.

Ефективність розробленої моделі безпосередньо залежить від дотримання фундаментальних принципів та критеріїв проектування. Першочерговим принципом є об’єктивність, яка вимагає, щоб онтологія надавала інтерпретацію термінів через об'єктивні логічні визначення,

підкріплені документацією природною мовою. Наступним критичним критерієм є повнота, що передбачає перевагу визначень, базованих на необхідних і достатніх умовах, над частковими описами. Принцип когерентності гарантує внутрішню логічну несуперечливість моделі, дозволяючи формувати висновки, що узгоджуються з вихідними визначеннями.

Важливим аспектом життєвого циклу онтології є забезпечення її максимальної монотонної розширюваності. Ця властивість передбачає можливість імплементації нових загальних або спеціалізованих термінів без необхідності ревізії чи руйнування існуючої структури понять. Паралельно з цим слід дотримуватися принципу мінімальних онтологічних зобов'язань, який полягає в мінімізації претензій на моделювання всієї повноти реальності, надаючи розробникам свободу для подальшої спеціалізації та інстанціювання моделі відповідно до локальних потреб. Для підвищення виразної потужності моделі застосовується диверсифікація ієрархій, що дозволяє використовувати механізми множинного наслідування. Структурна чистота забезпечується принципом онтологічного розрізнення (класи не повинні перетинатися без необхідності) та мінімізацією семантичної дистанції між однотипними поняттями. Завершують перелік вимог принципи модульності, що мінімізує зв'язність між окремими компонентами, та сувора стандартизація назв.

У контексті методології розробки особливу увагу привертає підхід, запропонований Грюнінгером і Фоксом, відомий як методологія питань компетентності (Competency Questions). Згідно з цією концепцією, вимоги до онтології формулюються у вигляді набору проблем, які система повинна бути здатна розв'язати. Ці питання мають бути трансформовані з природної мови у формалізовані, машинно-інтерпретовані запити. У сучасних реаліях Semantic Web, питання компетентності для онтологій RDFS та OWL формалізуються у вигляді запитів мовою SPARQL. Якщо виконання запиту над онтологією повертає очікуваний результат, вимога вважається задоволеною, а модель — валідною.

Загальноприйнятим стандартом в інженерії знань є ітеративний алгоритм розробки, що складається з семи послідовних етапів. На першому етапі відбувається визначення домену та області дії онтології. Розробник повинен чітко окреслити межі предметної області, сформулювати типи питань, на які система даватиме відповіді, ідентифікувати цільову аудиторію (наприклад, адміністратори мереж або кінцеві користувачі сервісів) та перевірити достатність інформаційного покриття.

Другий етап присвячений аналізу можливостей повторного використання існуючих онтологій. Це критичний крок для забезпечення інтероперабельності та економії ресурсів. Необхідно перевірити наявність релевантних моделей у глобальних бібліотеках (таких як Ontolingua, DAML, LOV) та оцінити потребу в їх розширенні або адаптації.

На третьому етапі здійснюється перелік ключових термінів. Формується неструктурований список понять, які необхідно пояснити, визначаються їхні властивості та інтерпретація (наприклад, розрізнення видів трафіку чи типів обладнання). Цей процес відбувається незалежно від подальшої ієрархічної категоризації.

Четвертий етап є ключовим для формування таксономії — визначення класів та їх ієрархії. Розробник обирає одну зі стратегій: низхідну (Top-down), що починається з найзагальніших концепцій, висхідну (Bottom-up), що рухається від конкретних прикладів до узагальнень, або комбінований підхід, який часто є найбільш ефективним.

П'ятий етап передбачає визначення властивостей класів (слотів). Класи наповнюються внутрішньою структурою, описуються атрибути об'єктів. Слот у даному контексті розглядається як характеристика, що не є класом, але має бути призначена певному концепту.

На шостому етапі відбувається деталізація фасетів (аспектів) слотів. Встановлюються обмеження на типи значень (рядки, числа, логічні значення), визначається кардинальність (потужність) атрибутів та задаються діапазони допустимих значень.

Фінальний, сьомий етап — це створення екземплярів (інстанціювання). На основі розробленої структури генеруються конкретні об'єкти бази знань: обирається відповідний клас, створюється індивід і заповнюються значення його слотів реальними даними.

1.6 Методологічні підходи, критерії та метрики оцінювання якості онтологічних моделей

Оцінювання якості онтології є критично важливим етапом життєвого циклу онтологічного інжинірингу, який визначає придатність розробленої моделі до практичного використання в інформаційних системах. У науковій літературі під оцінюванням (Evaluation) розуміють комплексний процес перевірки технічних характеристик онтології, її логічної коректності та відповідності вимогам предметної області. Цей процес прийнято розділяти на два взаємопов'язані напрями: верифікацію та валідацію. Верифікація спрямована на перевірку внутрішньої структури онтології, коректності синтаксису та дотримання формальних правил мови опису (наприклад, OWL або RDF), тобто відповідає на питання «чи правильно побудовано онтологію?». Натомість валідація фокусується на семантичному аспекті та відповідності моделі реальному світу або поставленим завданням, відповідаючи на питання «чи правильну онтологію ми побудували?». Нехтування будь-яким із цих етапів може призвести до помилок у логічному виведенні та несумісності системи з іншими програмними агентами.

У сучасній теорії інженерії знань виділяють кілька фундаментальних підходів до оцінювання онтологій, які класифікуються залежно від використовуваного еталону порівняння. Перший підхід, відомий як оцінювання на основі «золотого стандарту» (Gold Standard), передбачає порівняння розробленої онтології з певною еталонною моделлю, яка визнана

експертами як ідеальна або найбільш повна для даного домену. Цей метод дозволяє отримати кількісні метрики схожості, точності та повноти, проте його застосування часто ускладнюється відсутністю загальновизнаних стандартів для вузькоспеціалізованих предметних областей. Другий підхід базується на оцінюванні, орієнтованому на застосування (Application-based evaluation). У цьому випадку якість онтології визначається опосередковано — через ефективність роботи програмного забезпечення або інформаційної системи, яка використовує цю онтологію. Якщо система успішно виконує пошукові запити, навігацію чи логічне виведення, онтологія вважається якісною. Цей метод є найбільш прагматичним, оскільки перевіряє модель у реальних умовах експлуатації.

Третій поширений підхід — це оцінювання на основі даних (Data-driven evaluation), яке полягає у порівнянні онтології з корпусом текстових документів, що репрезентують предметну область. Цей метод дозволяє визначити, наскільки повно онтологія покриває термінологію домену та чи існують поняття, які були пропущені під час розробки. Четвертий підхід передбачає експертне оцінювання (Assessment by humans), де група фахівців аналізує структуру моделі на відповідність визначеним критеріям, таким як ясність, узгодженість та модульність. Хоча цей метод є суб’єктивним та трудомістким, він залишається незамінним для виявлення тонких семантичних помилок, які не можуть бути знайдені автоматизованими засобами.

Окрему увагу в процесі оцінювання приділяють кількісним та якісним метрикам. До основних показників якості відносять синтаксичну правильність, яка гарантує відсутність помилок у коді онтології, та логічну узгодженість (Consistency), яка означає відсутність суперечливих тверджень, що унеможливлюють логічне виведення. Важливою характеристикою є повнота (Completeness) моделі, тобто ступінь покриття нею знань предметної області, а також її лаконічність (Conciseness), що передбачає відсутність надлишкових елементів та дублювання інформації. Також оцінюється розширюваність (Expandability) — здатність онтології до інтеграції нових

знань без необхідності кардинальної перебудови структури, та чутливість, що відображає стійкість моделі до незначних змін у вхідних даних. Для автоматизації процесу оцінювання розроблено низку програмних інструментів, зокрема валідатори та різонери (reasoners), такі як HermiT, Pellet чи Fact++, які здатні автоматично виявляти логічні колізії, незадовільні класи та порушення ієрархічних зв’язків. Таким чином, комплексне застосування описаних підходів дозволяє гарантувати створення надійної та ефективної онтологічної бази знань.

2 МАТЕМАТИЧНА МОДЕЛЬ ТА ОБҐРУНТУВАННЯ АРХІТЕКТУРНИХ РІШЕНЬ ОНТОЛОГІЧНОЇ СИСТЕМИ ДЛЯ УПРАВЛІННЯ ТА МОНІТОРИНГУ МЕРЕЖЕВИХ ПРОЦЕСІВ

2.1 Формалізація лексикону предметної області та визначення сигнатури моделі

У рамках побудови математичної моделі онтології, лексикон формально визначається як фундаментальна алгебраїчна система, що складається з упорядкованих множин символів, необхідних для побудови коректних виразів мови. Базис лексикону формує зліченна множина індивідних констант, які слугують для позначення конкретних об'єктів предметної області, та скінченна множина змінних, визначених для кожного цілого невід’ємного числа $n \geq 0$.

. Окрім цього, структура лексикону включає скінченну множину n-впорядкованих предикатних констант і зліченну множину n-місних предикатних змінних, які у сукупності класифікуються як n-місцеві предикати. Важливим елементом алфавіту є зарезервовані логічні символи, що визначають операційний апарат моделі: пропозиційні зв’язки (заперечення ¬, кон’юнкція ∧, диз’юнкція ∨, імплікація →, еквівалентність ↔), квантори (загальності ∀, існування ∃), оператор лямбда-абстракції λ та оператор модальної необхідності $\Box$. Для забезпечення структурної цілісності та однозначності прочитання формул також використовується набір допоміжних розділових знаків, зокрема дужок різних типів.Для забезпечення однозначної ідентифікації елементів моделі застосовується сувора синтаксична конвенція щодо іменування змінних та констант. Змінні, що позначають індивідні об'єкти, традиційно формуються з літер латинського алфавіту нижнього регістру, наприклад x, y, z. Для розширення простору імен передбачена можливість використання числових нижніх індексів, що дозволяє оперувати

нескінченною кількістю змінних. У свою чергу, змінні n-місного предикату позначаються літерами верхнього регістру, як правило F^n, G^n та H^n. При цьому верхній числовий індекс, що вказує на арність предиката (n), може бути опущений (пригнічений) у випадках, коли контекст виразу дозволяє однозначно визначити кількість аргументів. Аналогічно до індивідних змінних, предикатні змінні також можуть супроводжуватися числовими індексами для розрізнення символів.Константи в запропонованій моделі формуються як алфавітно-цифрові послідовності (рядки). Винятком є односимвольні позначення, які зарезервовані для змінних, тому константи повинні мати відмінну структуру. Допускається використання символу тире як розділювача для з'єднання частин складних ідентифікаторів. Семантична роль константи кодується через регістр першої літери її назви: рядки, що починаються з літери нижнього регістру, призначені для денотації окремих індивідних об'єктів (екземплярів класів), тоді як рядки, що починаються з великої літери, використовуються для позначення предикатних констант, які виражають властивості, зв'язки або класи об'єктів у предметній області. Такий підхід забезпечує візуальне розмежування онтологічних сутностей на рівні синтаксису мови опису.

2.2 Синтаксичні правила та граматика побудови логічних виразів

Наявність лексикону є необхідною, але недостатньою умовою для повноцінного моделювання знань. Для того, щоб окремі терміни утворили зв'язну систему, необхідно визначити граматику — суворий набір синтаксичних правил, які регламентують допустимі способи поєднання лексичних одиниць. У контексті розроблюваної онтології граматика виконує роль фільтра, що гарантує коректність побудови запитів та правил бази знань.

Процес побудови виразів у системі підпорядковується ієрархічному принципу і базується на розрізненні понять «терм» та «формула». Під термом розуміється будь-яка синтаксична конструкція, що позначає об’єкт предметної області. Термами виступають як конкретні імена (константи), так і змінні, що їх заміщують.

На основі термів будуються формули, які виражають закінчені думки або судження про стан системи. Граматика визначає такі правила їх формування:

1. Атомарні судження: Найпростіші формули утворюються шляхом застосування предикату до набору термів. Це дозволяє формалізувати прості факти, наприклад, ствердження про належність пристрою до певного типу або наявність зв’язку між двома вузлами.

2. Логічна композиція: Складні формули утворюються з простих за допомогою логічних зв’язок. Граматика дозволяє об’єднувати умови (наприклад, «якщо завантаження процесора високе І трафік аномальний»), створювати альтернативи та заперечувати певні стани.

3. Квантифікація: Важливою особливістю граматики є можливість використання кванторів — операторів загальності та існування. Це дозволяє формулювати універсальні правила (наприклад, «для всіх серверів певної групи застосувати політику безпеки») або екзистенційні твердження (наприклад, «існує вразливість у підсистемі»).

Особливістю запропонованої моделі є підтримка механізмів реїфікації (уречевлення), коли складні відношення або навіть цілі висловлювання можуть розглядатися як окремі об’єкти (терми). Це дозволяє системі оперувати абстракціями високого рівня, наприклад, аналізувати не просто параметри пристроїв, а цілі сценарії подій як єдині сутності. Така граматична гнучкість є критично важливою для реалізації функцій інтелектуального аналізу та прогнозування в системах управління мережами.

2.3 Синтаксичні обмеження та правила коректного формування термів

Визначена у попередньому підрозділі формальна граматика мови $\mathcal{L}$ містить низку критично важливих обмежувальних правил, які регулюють процес утворення складних термів. Ці обмеження не є довільними; вони виступають необхідним запобіжником, що гарантує логічну стійкість та обчислювальну коректність усієї онтологічної моделі. Найбільш суттєвим серед них є обмеження, що стосується використання оператора лямбда-абстракції (λ). Згідно з цим правилом, оператор λ може зв’язувати виключно окремі змінні (індивідні змінні), а не предикатні змінні чи складні вирази.

Фундаментальне значення цього обмеження полягає в необхідності уникнення логічних колізій, відомих у теорії множин як парадокс Рассела. Якби граматика допускала вільне зв’язування змінних будь-якого типу, це легітимізувало б створення термів, що описують «властивість несамозастосовності» (властивість, яка не застосовується до самої себе). Формально такий термін мав би вигляд $[\lambda F \neg F(F)]$. Інтуїтивно він здається допустимим, проте його використання в якості предиката в атомарній формулі призводить до нерозв'язного протиріччя: твердження стає істинним тоді і тільки тоді, коли воно є хибним. Заборона на подібні конструкції в мові $\mathcal{L}$ не просто технічно блокує помилку, а має глибоке теоретичне обґрунтування. Вона базується на твердженні, що в реальній семантиці не існує інтуїтивно зрозумілої логічної операції, яка б відповідала таким самореферентним зв’язкам. Отже, парадокс не просто ігнорується, а усувається на рівні архітектури мови через відмову від хибних припущень щодо структури складних відношень.

Окрім запобігання парадоксам, граматичні правила накладають ряд додаткових обмежень, спрямованих на спрощення семантики та усунення надлишкової складності. Зокрема, висувається вимога, щоб усі змінні, зв’язані

оператором λ, обов'язково зустрічалися в тілі формули, що слідує за оператором. Це правило виключає так зване «порожнє зв'язування», коли змінна оголошується, але не використовується, що не несе смислового навантаження, але ускладнює автоматичну обробку виразів. Також вводяться обмеження на вільне входження змінних у вкладені складні терми та на зв'язані входження предикатних змінних. Хоча теоретично можна було б побудувати семантику, що підтримує такі конструкції, на практиці вони є несуттєвими для задач управління комп'ютерними мережами, і їх дозвіл лише створив би невиправдані технічні труднощі при реалізації алгоритмів логічного виведення.

Окремої уваги заслуговує заборона на використання модальних операторів (таких як оператор необхідності) всередині структури термів при їх формуванні. Це обмеження відображає специфіку предметної області та сучасний стан технологій Semantic Web. Переважна більшість існуючих онтологій є онтологіями першого порядку, тобто їхній зміст (пропозиції) можна повністю виразити мовою логіки першого порядку без використання модальностей. Це відповідає виразним можливостям сучасних стандартів, таких як DAML+OIL чи OWL. Тому включення модальностей безпосередньо в терми онтології на даному етапі виглядає надлишковим.

Проте, важливо розрізняти мову опису об'єктів онтології та мета-мову, що описує властивості самої системи. Виключення модальностей зсередини термів не означає повної відмови від них у запропонованій теорії. Навпаки, модальна логіка є необхідним інструментом для характеристики природи онтологій як цілісних об'єктів та опису логічних зв'язків між ними. Саме модальний компонент дозволяє строго визначити поняття логічного слідування (втілення) та сформулювати стійке визначення онтологічного змісту, що буде продемонстровано в подальших розділах роботи. Таким чином, синтаксичні обмеження формують баланс між виразною потужністю мови та її обчислювальною ефективністю.

2.4 Структурна організація семантичної моделі онтології

Математична модель M для мови $\mathcal{L}$ визначається як впорядкована алгебраїчна система (кортеж), що складається з п’яти фундаментальних компонентів: домену моделі, множини можливих світів, функції домену, множини логічних операцій та функції розширення.

Першим елементом є домен моделі (D), який є об’єднанням двох взаємно неперетинних множин: множини екземплярів (A) та множини відношень (R). Екземпляри репрезентують конкретні об’єкти предметної області, тоді як відношення представляють собою семантичні зв’язки та властивості. Сама множина відношень має дворівневу структуру. По-перше, вона поділяється на підмножину логічно примітивних відношень (базових фактів, що не розкладаються) та підмножину логічно складних відношень, які конструюються з примітивних. По-друге, відношення класифікуються за своєю арністю (кількістю аргументів), утворюючи ієрархію класів n-місних відношень.

Другим компонентом є множина «можливих універсумів» (W), яка вводить у модель модальний вимір. Ця множина дозволяє формалізувати поняття «можливих ситуацій» або станів системи, що є критично важливим для моделювання динаміки мережевих процесів та визначення логічного слідування між різними онтологіями. З кожним можливим універсумом пов’язана функція домену, яка відображає цей універсум на підмножину об’єктів, що існують у ньому. Це дозволяє моделювати ситуації, коли певні мережеві вузли або сервіси є активними в одному стані системи, але відсутні в іншому.

Четвертим, конструктивним елементом моделі є множина логічних операцій (Op), яка відповідає за формування складних знань з атомарних елементів. Ця множина включає п’ять груп операцій:

1. Операції предикації: дозволяють застосовувати відношення до конкретних екземплярів, зменшуючи арність відношення (фактично, підставляючи аргументи у функцію).

2. Логічні зв’язки: операції заперечення та імплікації, які змінюють істиннісне значення або об’єднують відношення, формуючи складні умови.

3. Операції універсалізації: відповідають дії квантора загальності, перетворюючи відношення на судження про всі об’єкти певного класу.

4. Операції перетворення: змінюють порядок аргументів у відношенні (перестановка).

5. Операції відображення (рефлексії): дозволяють ототожнювати різні аргументи одного відношення, що необхідно для вираження властивостей самозастосовності.

Всі ці операції є ін’єктивними (типу «один-до-одного»), а діапазони їх значень не перетинаються, що гарантує унікальність логічної структури кожного складного відношення.

П’ятим елементом є функція розширення (ext), яка визначає семантику відношень. Вона ставить у відповідність кожному відношенню в конкретному можливому універсумі набір кортежів об’єктів, для яких це відношення є істинним. Для пропозицій (0-місних відношень) розширення може приймати лише два значення: істина (весь домен) або хибність (порожня множина). Поведінка функції розширення суворо регламентована алгебраїчними властивостями логічних операцій. Наприклад, розширення заперечення відношення є доповненням до розширення вихідного відношення, а розширення імплікації обчислюється як об’єднання доповнення першого аргументу та розширення другого.

Така архітектура дозволяє представити будь-яке складне знання як результат послідовного застосування логічних операцій до примітивних елементів. Цей процес можна візуалізувати у вигляді дерева конститутивності (structural tree). Коренем такого дерева є аналізоване складне відношення, вузлами — проміжні логічні конструкції, а листками — примітивні

відношення та екземпляри. Оскільки кожне складне відношення має рівно одне повне дерево складу, це дозволяє однозначно ідентифікувати його логічну форму.

Наприклад, твердження «Сонце більше за будь-яку планету» в цій моделі конструюється поетапно: спочатку формується відношення «бути більшим за Сонце», потім воно поєднується з властивістю «бути планетою» через імплікацію, далі застосовується операція рефлексії для зв’язування змінних, і нарешті — операція універсалізації. Цей формалізм дає можливість чітко визначити поняття онтологічного змісту: змістом онтології є сукупність усіх примітивних складових, що входять до повних дерев конститутивності її тверджень. Це забезпечує надійну теоретичну базу для порівняння онтологій, виявлення їхньої сумісності та інтеграції в єдину систему моніторингу.

2.5 Архітектурна організація та структурні компоненти онтології управління комп’ютерними мережами

У рамках даного кваліфікаційного дослідження розробляється спеціалізована онтологічна модель, яка фокусується на представленні знань, необхідних для вирішення задач моніторингу та оперативного управління мережевою інфраструктурою. Варто зазначити, що проектована онтологія не ставить за мету охопити всі можливі аспекти побудови комп’ютерних систем, а зосереджується на тих семантичних розрізах, які є критично важливими для процесів контролю, діагностики несправностей, аналізу продуктивності та планування заходів із забезпечення якості обслуговування (QoS).

Функціональне призначення онтології полягає у забезпеченні програмно-апаратних комплексів управління (Network Management Systems) необхідною базою знань для прийняття рішень. Для цього модель повинна задовольняти низку вимог: повноцінно відображати таксономічні та

асоціативні зв’язки між поняттями домену, явно визначати причинно-наслідкові ланцюжки між просторово та часово розподіленими подіями, забезпечувати можливість інтегрованого аналізу поведінки системи, а також фіксувати динамічний взаємовплив елементів мережі. Такий підхід дозволяє сформувати високорівневу, абстраговану картину функціонування складної розподіленої системи.

Концептуальний базис онтології включає опис основних компонентів мережевого, канального та фізичного рівнів моделі OSI, а також специфікації трафіку, ресурсів та механізмів їх розподілу. Важливим аспектом є інтеграція опису самих інструментів управління (Sensors, Management Tools), що дозволяє системі рефлексувати над власним станом та доступними засобами впливу. З метою перевірки адекватності моделі передбачено використання модуля логічного виведення (Logical Reasoner), який, базуючись на формалізованих знаннях, зможе автоматично генерувати діагностичні висновки та рекомендації.

Для формалізації знань обрано мову веб-онтологій OWL (Web Ontology Language), а саме її діалект OWL DL (Description Logic). Цей вибір обґрунтовується необхідністю забезпечення балансу між виразною силою мови та обчислювальною розв'язністю (decidability) алгоритмів логічного виведення. OWL DL, на відміну від повної версії OWL Full, накладає певні обмеження на структурування знань, проте гарантує завершеність процедур перевірки несуперечливості, що є критичним для систем реального часу.

Структура розробленої онтології базується на ієрархічній таксономії класів, коренем якої є універсальний суперклас Thing. Усі доменні поняття є нащадками цього класу і специфікуються через набір властивостей. Властивості поділяються на DatatypeProperties, що описують атрибути об'єктів (наприклад, пропускну здатність як ціле число), та ObjectProperties, що задають відношення між об'єктами (наприклад, зв'язок «маршрутизатор з'єднаний з комутатором»). Онтологія активно використовує механізми

обмежень (restrictions) для уточнення семантики класів, зокрема обмеження на типи значень, кардинальність (кількість) зв'язків та екзистенційні квантори.

Архітектура класів організована за принципом «від загального до конкретного», рис.2.1. На верхньому рівні абстракції знаходяться кореневі класи, що описують найбільш універсальні категорії.

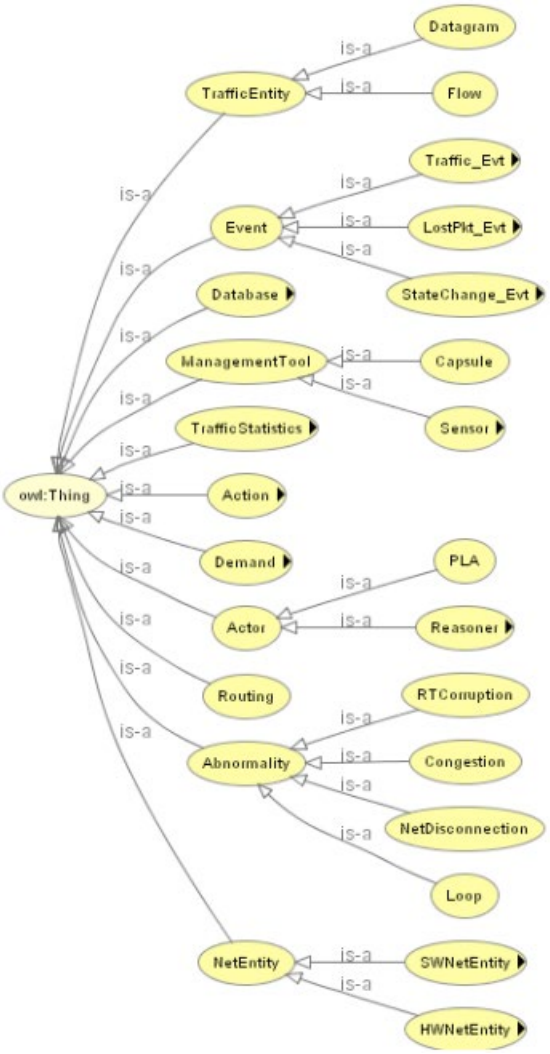


Рисунок 2.1 — Фрагмент ієрархічного дерева класів онтології (відношення «is-a»)

Основні структурні компоненти мережі агреговані в класі NetEntity. Цей клас охоплює фізичне обладнання (маршрутизатори, хости, канали) та логічні компоненти (інтерфейси, черги, таблиці маршрутизації). Дані, що передаються мережею, моделюються класом TrafficEntity, який має прямі посилання на

використовувані ресурси. Динаміка системи описується через властивості стану, залежні від часу, а також через абстрактні концепти Routing (Маршрутизація) та Demand (Попит), що визначають глобальну поведінку мережі.

Ключовим елементом моделювання динаміки є клас Event (Подія), який характеризується чіткою часовою та просторовою локалізацією. Події можуть бути як індикаторами нормальної роботи (наприклад, оновлення таблиці маршрутизації), так і симптомами збоїв. Для діагностики несправностей введено поняття Abnormality (Аномалія), яке пов'язане з подіями причинно-наслідковими зв'язками. Це дозволяє модулю логічного виведення (Reasoner) корелювати розрізнені події та ідентифікувати корінні причини проблем (Root Cause Analysis).

Завершує архітектуру опис інфраструктури управління, де центральним актором виступає Reasoner. Він взаємодіє з мережею через програмованих локальних агентів (клас PLA), виконуючи дії (Action) з налаштування моніторингу або зміни конфігурації. Результатом роботи системи є статистичні оцінки ефективності (TrafficStatistics) та звіти про виявлені аномалії.

3 РЕЗУЛЬТАТИ МОДЕЛЮВАННЯ ТА ПРАКТИЧНА РЕАЛІЗАЦІЯ СИСТЕМИ УПРАВЛІННЯ КОМП'ЮТЕРНИМИ МЕРЕЖАМИ

3.1 Побудова онтологічної моделі комунікаційної інфраструктури

Практична реалізація запропонованої онтології базується на необхідності повного покриття всіх структурних та функціональних елементів мережевої інфраструктури, що забезпечують передачу даних на рівні мережі Інтернет, а також частково охоплюють аспекти нижніх рівнів моделі OSI. Розроблені концепти дозволяють однозначно визначити топологію мережі, описати механізми маршрутизації та формалізувати процеси розподілу ресурсів для управління трафіком. Усі компоненти моделі розглядаються як «керовані об'єкти» (Managed Objects), параметри яких підлягають постійному моніторингу та контролю з боку системи управління.

Для уніфікації опису різнорідних елементів введено базовий клас «NetEntity», який агрегує будь-які компоненти, що беруть участь у функціонуванні мережі. З метою деталізації цей клас спеціалізується на двох фундаментальних підкласах: «HWNetEntity» та «SWNetEntity». Перший клас об'єднує всі фізичні апаратні засоби, що формують матеріальну основу комунікаційної інфраструктури, тоді як другий описує програмні сутності та логічні абстракції, що визначають алгоритми роботи мережі. Повна ієрархічна структура концепту «NetEntity» та його спеціалізацій візуалізована на рис.3.1.

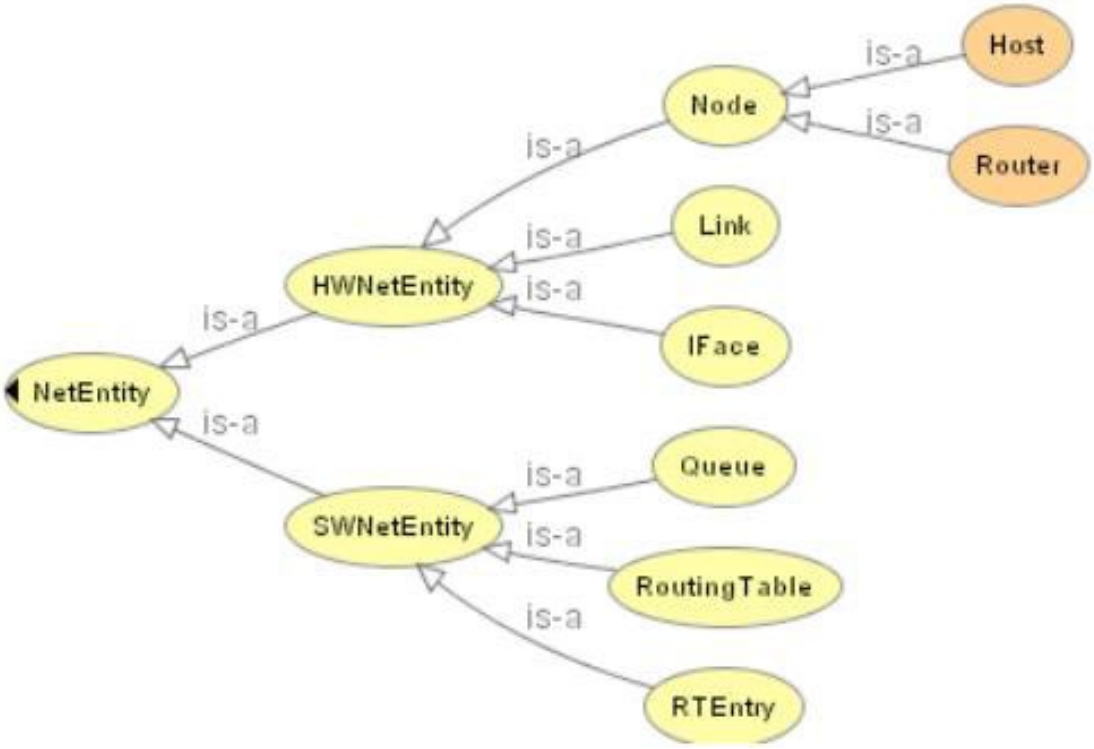


Рисунок 3.1 — Ієрархія класів концепту NetEntity

к демонструє наведена схема, клас апаратних компонентів «HWNetEntity» декомпозується на три ключові категорії. Першою є клас «Node», який у свою чергу поділяється на підкласи «Router» (маршрутизатори) та «Host» (кінцеві вузли), що відображає різницю у функціональному призначенні пристроїв. Другою категорією є клас «IFace», що моделює мережеві інтерфейси (мережеві карти), які виступають точками входу та виходу для інформаційних потоків. Третя категорія представлена класом «Link», який описує фізичні канали зв'язку, що з'єднують вузли в єдину топологію. Критично важливою характеристикою всіх апаратних сутностей є їхній функціональний статус, який формалізується через властивість типу даних «HWStatus». Ця властивість може набувати трьох значень: «ON» (об'єкт активний і працює коректно), «OFF» (об'єкт вимкнено, але він доступний для керування) та «ABN» (аномальний стан або відмова, що унеможливлює пряме управління).

Визначення мережевої топології реалізується через систему об'єктних властивостей («ObjectProperties»), які встановлюють асоціативні зв'язки між класами. Схема асоціацій між апаратними класами наведена на рис.3.2.

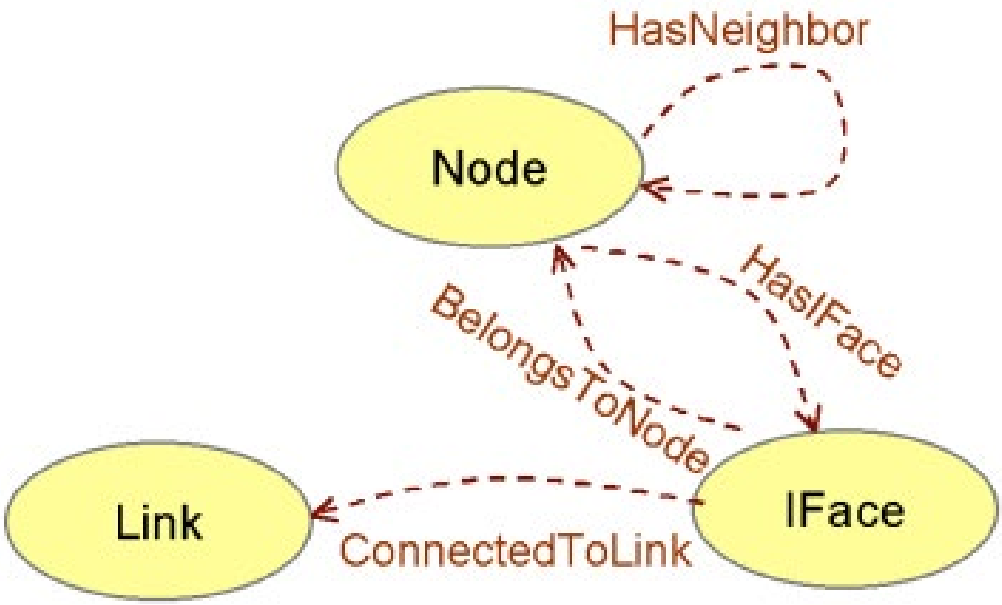


Рисунок 3.2 — Схема асоціацій між класами на основі ObjectProperties

Для моделювання зв’язку між інтерфейсом та відповідним вузлом застосовується пара взаємно обернених властивостей: «HasIFace» (оголошена у класі «Node») та «BelongsToNode» (оголошена у класі «IFace»). Аналогічним чином, фізичне з'єднання між вузлами описується через властивість «ConnectedToLink», що пов’язує інтерфейси з конкретними каналами зв’язку. Важливою особливістю розробленої моделі є підтримка резервування каналів. Завдяки тому, що властивість «ConnectedToLink» може приймати множинні значення, система здатна моделювати ситуації перемикання на резервні лінії («Backup») без порушення логічної цілісності з'єднання між вузлами. Для спрощення навігації по графу мережі до класу «Node» додано симетричну властивість «HasNeighbor», яка дозволяє безпосередньо ідентифікувати сусідні пристрої, хоча внутрішні обмеження мови OWL накладають певні ліміти на використання цієї властивості у складних правилах виведення.

Приклад взаємодії екземплярів класу «HWNetEntity», що ілюструє побудову логічного з'єднання між двома вузлами через інтерфейси та канал зв'язку, показано на рис.3.3.

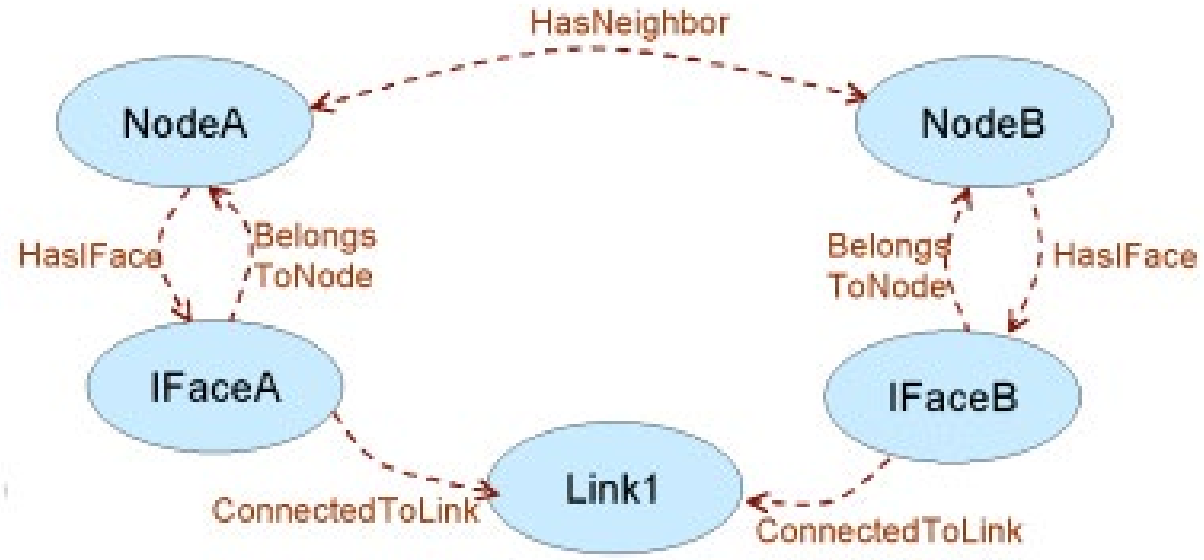


Рисунок 3.3 — Приклад інстанціювання та взаємодії об'єктів класу «HWNetEntity»

Окрім структурних зв'язків, кожен клас наділений набором атрибутів, що уточнюють його природу. Наприклад, інтерфейси ідентифікуються унікальними адресами та посиланнями на асоційовані черги пакетів. Канали зв’язку («Links») моделюються з високою деталізацією, включаючи параметри технології передачі («Technology»), пропускної здатності («BandWidth») та метрики вартості маршрутизації («Cost»).

Логічна складова інфраструктури описується підкласами сутності «SWNetEntity», серед яких центральне місце займають компоненти маршрутизації. Структура властивостей, що описують класи «Node», «RoutingTable» та «RTEntry», представлена на рис.3.4.

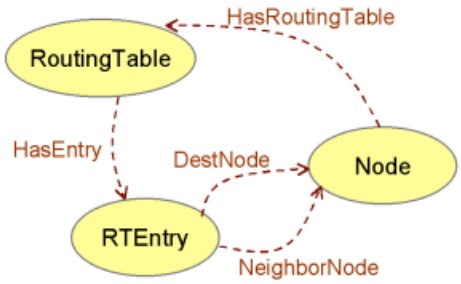


Рисунок 3.4 — Властивості об’єктів для опису логіки маршрутизації

Для кожного вузла визначається властивість «HasRoutingTable», яка пов'язує його з екземпляром таблиці маршрутизації. Сама таблиця моделюється як агрегація окремих записів (клас «RTEntry»), пов'язаних через властивість «HasEntry». Кожен запис містить вичерпну інформацію для прийняття рішень про пересилання пакетів: посилання на вузол призначення («DestNode»), наступний транзитний вузол («NeighborNode») та вартість маршруту. Такий підхід дозволяє явно виразити онтологічні зв'язки між абстрактною інформацією маршрутизації та фізичними елементами мережі. Для контролю цілісності даних введено властивість «RTStatus», яка сигналізує про коректність («Normal») або пошкодження («Corrupted») записів таблиці.

Третім важливим компонентом програмної інфраструктури є клас «Queue» (Черга), який моделює буфери обміну даними. За допомогою властивості «QueueType» розрізняються вхідні («IN») та вихідні («OUT») черги, що однозначно прив'язуються до відповідних інтерфейсів. Моніторинг заповненості черг є критичним індикатором продуктивності: переповнення вхідних черг свідчить про нестачу обчислювальних ресурсів вузла, тоді як черги на вихід сигналізують про перевантаження каналів зв'язку. Така деталізація дозволяє системі управління точно локалізувати «вузькі місця» в інфраструктурі.

3.2 Моделювання процесів передачі трафіку та механізмів розподілу навантаження в комп’ютерній мережі

Аналіз ефективності функціонування будь-якої комп’ютерної мережі неможливий без детального розгляду процесів передачі даних, оскільки транспортування інформації між вузлами є основною телеологічною функцією мережевої інфраструктури. У контексті управління мережею, завдання моніторингу та контролю зводяться до безперервної оцінки параметрів трафіку з метою верифікації відповідності реальних показників роботи (продуктивності, затримок, надійності) встановленим вимогам якості обслуговування (QoS). Відповідно, побудова повноцінної онтологічної моделі вимагає інтеграції концептуального апарату для опису інформаційних потоків.

Фундаментальним елементом у цій частині моделі виступає кореневий клас «TrafficEntity», який узагальнює всі типи даних, що циркулюють у мережі. Важливо провести чітку демаркаційну лінію між класами «TrafficEntity» та «SWNetEntity». Якщо останній моделює програмні компоненти інфраструктури (наприклад, таблиці маршрутизації), які забезпечують механізми управління, то «TrafficEntity» представляє сам суб’єкт передачі — корисне навантаження, яке споживає мережеві ресурси та утилізує пропускну здатність каналів.

Для забезпечення можливості аналізу трафіку на різних рівнях гранулярності, клас «TrafficEntity» спеціалізується на два ключові підкласи: «Datagram» (Датаграма) та «Flow» (Потік). Візуалізація таксономії класів, що описують трафік комп'ютерної мережі, наведена на рис.3.5.

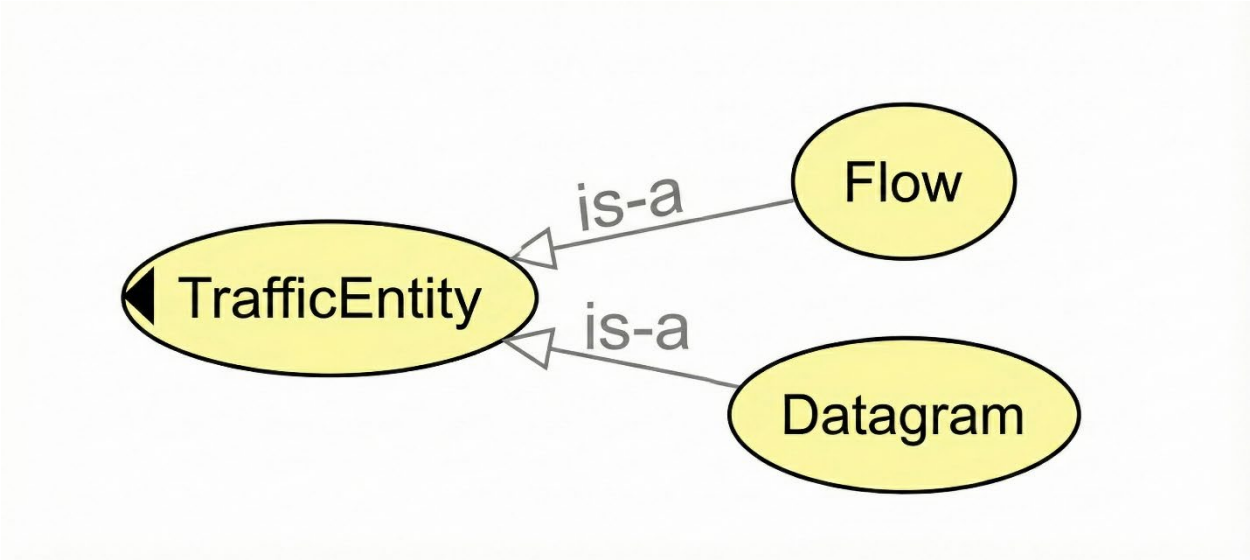


Рисунок 3.5 — Ієрархічна структура класів для опису мережевого трафіку

Клас «Datagram» моделює атомарну одиницю передачі даних на мережевому рівні (IP-пакет). Кожен екземпляр цього класу характеризується набором атрибутів, що відображають заголовки реальних пакетів: розмір даних (властивість «Length»), адресу вузла-відправника («SourceNode») та адресу вузла-отримувача («DestNode»). Такий рівень деталізації дозволяє системі моніторингу опускатися до аналізу окремих пакетів при діагностиці складних проблем.

На вищому рівні абстракції знаходиться клас «Flow», який агрегує множину датаграм, що мають спільні характеристики передачі. Потік визначається як сукупність пакетів, що переміщуються по єдиному маршруту між фіксованим джерелом та призначенням. Зв'язок між потоком і його складовими реалізується через властивість «HasDatagram», яка може приймати множинні значення. Хоча мова OWL накладає певні обмеження на перевірку відповідності атрибутів джерела/призначення між потоком і вкладеними датаграмами, така структура дозволяє ефективно моделювати логічні з'єднання. Для темпоральної (часової) локалізації потоку введено властивості «StartTime» та «EndTime», що фіксують інтервал його існування. Варто зазначити, що у даній онтології поняття потоку є дещо спрощеним у порівнянні з класичним 5-кортежним визначенням (IP джерела/призначення,

порти джерела/призначення, протокол), що зумовлено фокусом моделі на задачах маршрутизації та розподілу ресурсів на трьох нижніх рівнях моделі OSI.

Ключовим аспектом моделювання є формалізація взаємозв'язків між трафіком та фізичними ресурсами мережі. Для цього введено систему перехресних посилань. Потоки пов'язуються з каналами зв'язку («Links»), через які вони проходять, за допомогою пари властивостей «OnLink» та «UsedByFlow». Аналогічно, черги («Queues») посилаються на датаграми, що знаходяться в них у поточний момент часу. Для кількісної оцінки навантаження використовуються метрики використання ресурсів: «UsedBandwidth» (загальний відсоток завантаження каналу) та «FlowBandwidth» (частина смуги пропускання, яку займає конкретний потік). Схема асоціацій та властивостей, що описують механізми розподілу трафіку по мережевим ресурсам, представлена на рис.3.6.

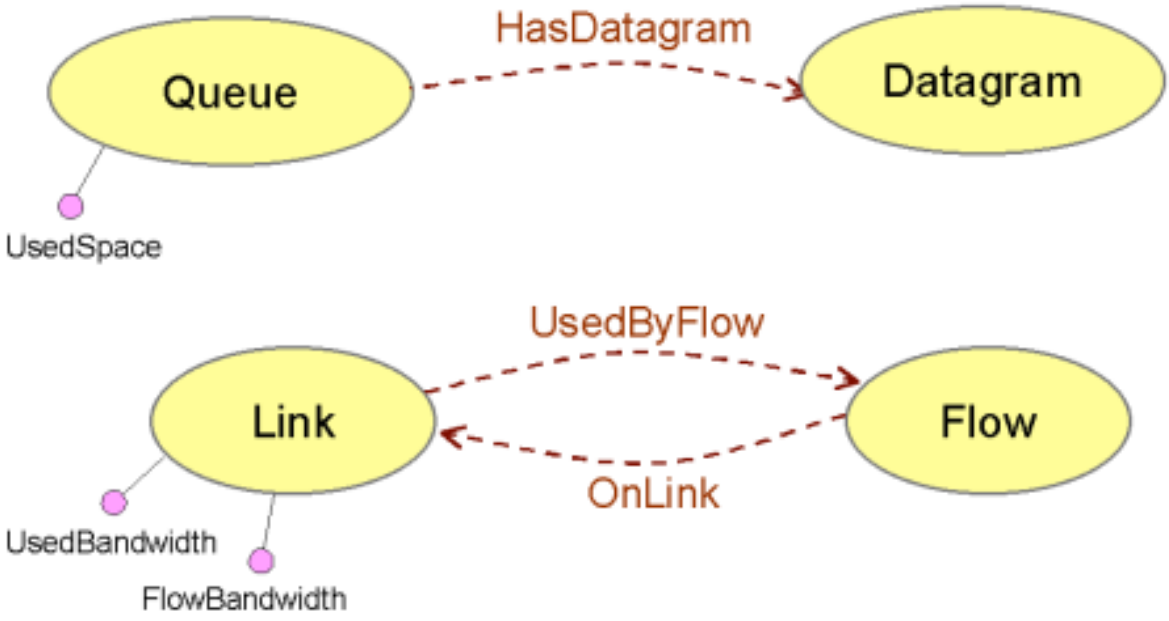


Рисунок 3.6 — Модель асоціацій для представлення розподілу трафіку та використання ресурсів

Однак, структурного опису елементів недостатньо для розуміння глобальної поведінки мережі, оскільки реальний розподіл трафіку залежить від складної взаємодії між алгоритмами маршрутизації та потребами користувачів. Швидкість окремого потоку не є ізольованою величиною; вона визначається як обсягом даних, що генерує джерело, так і конкуренцією за ресурси з боку інших потоків. Для моделювання цих системних ефектів введено абстрактні класи «Routing» (Маршрутизація) та «Demand» (Попит).

Концепція «Routing» узагальнює вплив налаштувань маршрутизації на розподіл потоків. Вона пов'язана з конкретними таблицями маршрутизації через властивість «RelatedTo» і з потоками через властивість «DependsOnRouting». Це дозволяє системі "розуміти", що зміна маршруту одного потоку може вплинути на продуктивність інших через перерозподіл навантаження на спільні канали.

Концепція «Demand» моделює гіпотетичне навантаження — обсяг даних, який користувачі бажають передати, ігноруючи обмеження пропускної здатності. Цей клас деталізується через підкласи «DemandMatrix» (Матриця попиту) та «DMEntry» (Запис матриці). Структурна організація класу «Demand» та його складових показана на рис.3.7.

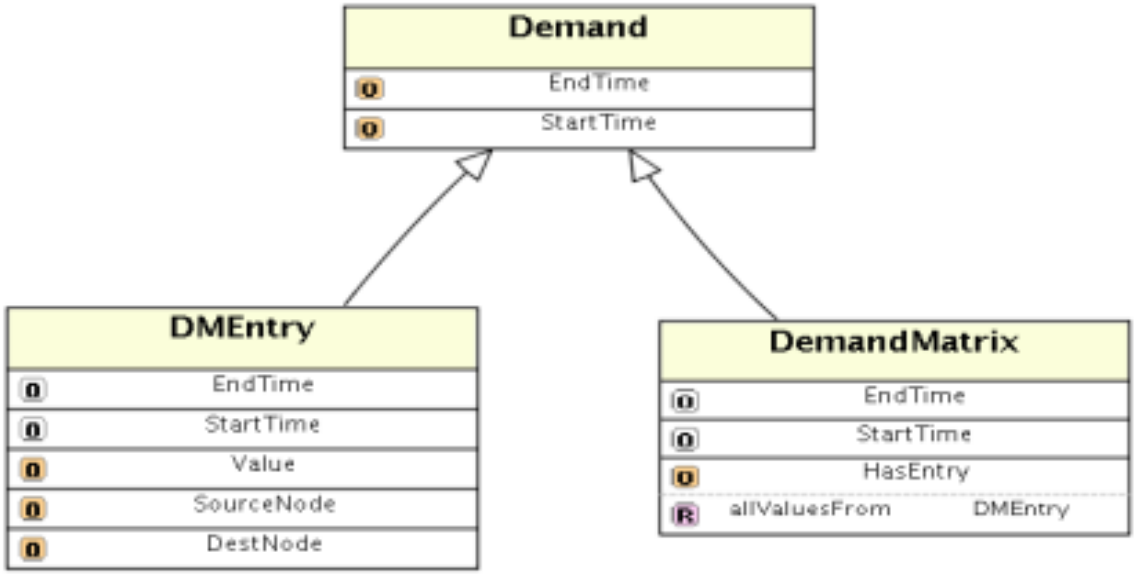


Рисунок 3.7 — Структура класу «Demand» та матриці попиту

Кожен запис «DMEntry» кількісно визначає (властивість «Value») обсяг трафіку між конкретною парою вузлів «джерело-призначення». Зв'язок між ідеальним попитом та реальним потоком фіксується властивістю «DependsOnDemand». Це дозволяє аналізувати, наскільки мережа здатна задовольнити потреби користувачів. Графічне відображення залежності між класами потоку та попиту наведено на рис.3.8.



Рисунок 3.8 — Модель залежності між реальним потоком (Flow) та попитом (Demand)

Таким чином, розроблена модель трафіку створює надійну основу для наступного етапу дослідження — моделювання динамічних подій та аномалій, що виникають у процесі експлуатації комп'ютерної мережі.

3.3 Діагностика динамічного стану мережі: моделювання подій та виявлення аномалій

Якщо попередні підрозділи були присвячені статичній структурі («NetEntities») та потокам даних («TrafficEntities»), які формують «скелет» та «кровоносну систему» мережі, то даний підрозділ фокусується на динаміці її функціонування. Глобальний статус комп'ютерної мережі не є константою; він еволюціонує під впливом процесів маршрутизації, коливань попиту

користувачів та змін у стані обладнання. Для формалізації цієї динамічної еволюції та переходу системи від одного стану до іншого в онтологію введено фундаментальний концепт «Event» (Подія).

Клас «Event» слугує узагальненням для будь-яких інцидентів або змін, що відбуваються в мережевому середовищі. На найвищому рівні абстракції подія характеризується темпоральною міткою («Time») та просторовою локалізацією («OnEntity»), що вказує на конкретний компонент інфраструктури, де стався інцидент. Для забезпечення гранулярності моніторингу, цей загальний клас конкретизується через систему спеціалізованих підкласів. Ієрархічна структура таксономії подій візуалізована на рис.3.9.

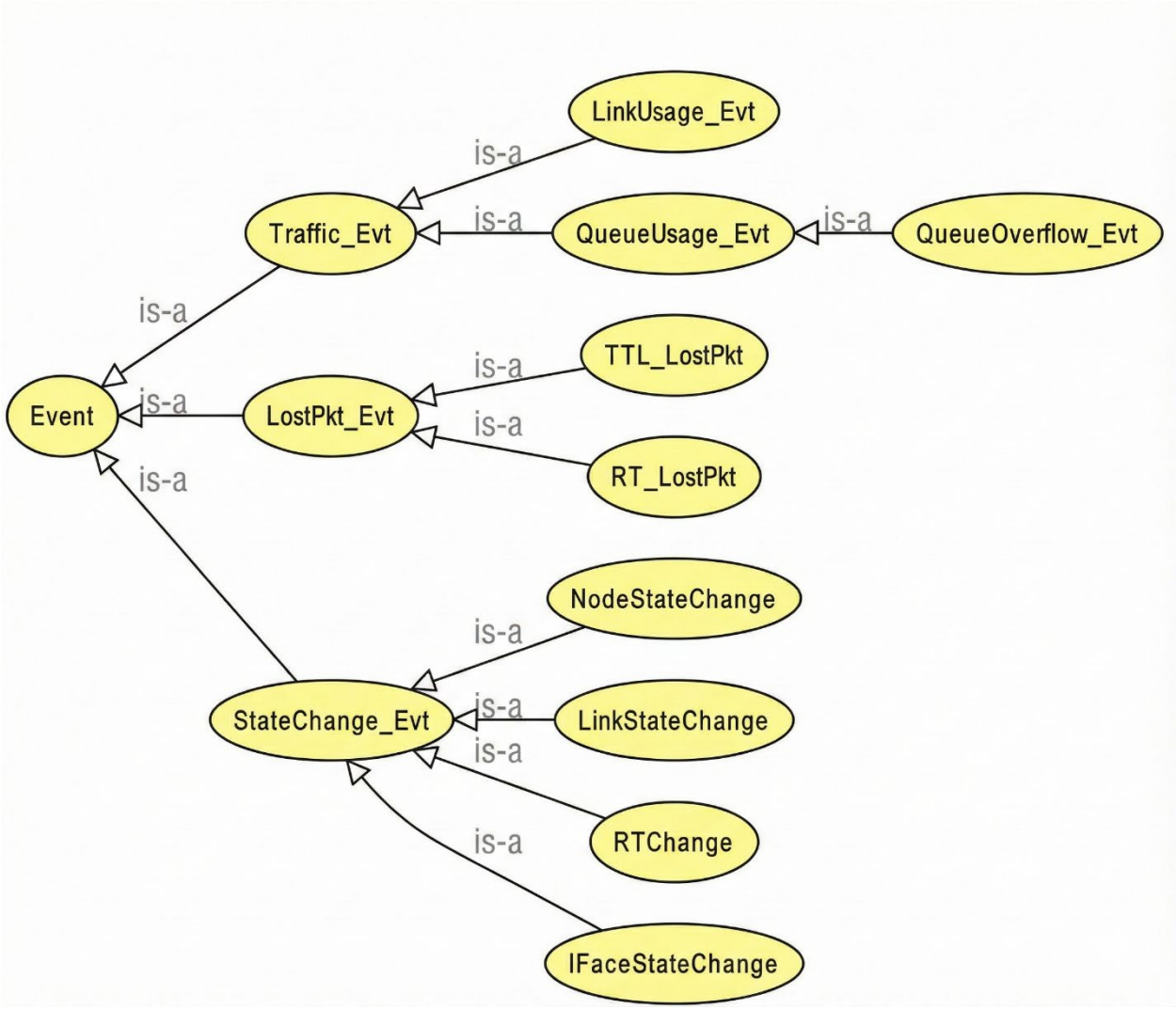


Рисунок 3.9 — Ієрархічна класифікація подій (Events)

Як видно зі схеми, таксономія подій базується на природі їхнього впливу на мережу і включає три основні гілки. Перша категорія — «LostPktEvt» (Події втрати пакетів) — сигналізує про відкидання датаграм маршрутизаторами. Вона додатково диференціюється за причиною втрати: «RTLostPkts» виникають через відсутність запису в таблиці маршрутизації (помилка пересилання), тоді як «TTLLostPkts» фіксують знищення пакетів через вичерпання ліміту часу життя (TTL). Друга категорія — «StateChangeEvt» (Події зміни стану) — відображає перемикання режимів роботи апаратних або програмних компонентів. Вона включає зміни стану вузлів («NodeStateChange»), інтерфейсів («IFaceStateChange»), каналів («LinkStateChange») та оновлення таблиць маршрутизації («RTChange»). Третя категорія — «TrafficEvt» (Події трафіку) — описує динаміку завантаження ресурсів, розрізняючи події використання каналів («LinkUsageEvt») та черг («QueueUsageEvt»), з окремим виділенням критичної події переповнення буфера («QueueOverflowEvt»).

Важливим аспектом моделювання є прив'язка подій до конкретних сутностей через властивість «OnEntity», діапазон значень якої обмежується залежно від типу події. Наприклад, для «LinkStateChange» ця властивість може посилатися виключно на об'єкти класу «Link». Для подій зміни стану додатково введено властивість «ChangeType», що фіксує перехід у режим «ON» або «OFF». Окрему логіку має клас «RTChange», який через властивість «ChangedEntry» вказує на конкретні записи таблиці маршрутизації, що зазнали модифікації.

Події трафіку («TrafficEvt») виконують функцію актуалізації даних про використання ресурсів. Наприклад, «LinkUsageEvt» через властивості «UsedByFlow» та «FlowBandwidth» оновлює інформацію про те, яку частину пропускної здатності каналу займає конкретний потік. Це дозволяє встановити явний зв'язок між логічним потоком та фізичним середовищем передачі, який неможливо виразити статичними атрибутами класу «Link». Аналогічно, події використання черг оновлюють метрику «UsedSpace». Приклад моделювання

асоціації між класом подій та сутностями, на які вони впливають, наведено на рис.3.10.

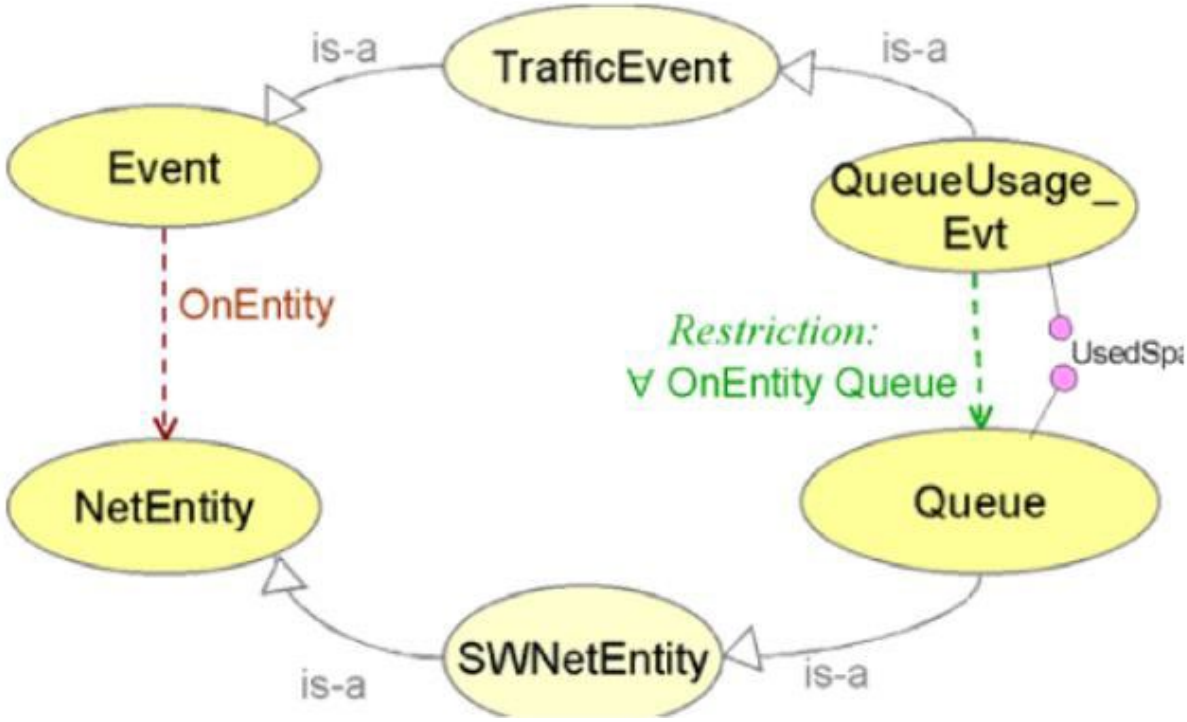


Рисунок 3.10 — Схема асоціацій між подіями (QueueUsageEvt) та мережевими сутностями (Queue)

З точки зору системи управління, події є "сирим" матеріалом для діагностики. Вони можуть відображати нормальну роботу мережі, але в певних контекстах стають симптомами несправностей. Для формалізації причинно-наслідкових зв'язків між симптомами та їхніми кореневими причинами в онтологію введено поняття «Abnormality» (Аномалія).

Варто підкреслити онтологічну відмінність між подією та аномалією. Подія — це миттєвий локальний факт (наприклад, "інтерфейс упав"). Аномалія — це глобальний, тривалий у часі патологічний стан системи (наприклад, "петля маршрутизації" або "перевантаження"), який проявляється через множину подій-симптомів у різних точках мережі. Тому аномалії та події не мають спадкових зв'язків, а є незалежними нащадками класу «Thing»,

пов'язаними через властивості «Causes» (Аномалія спричиняє Подію) та «CausedBy» (Подія спричинена Аномалією).

Класифікація аномалій, що представлена в онтології, показана на рису.3.11.

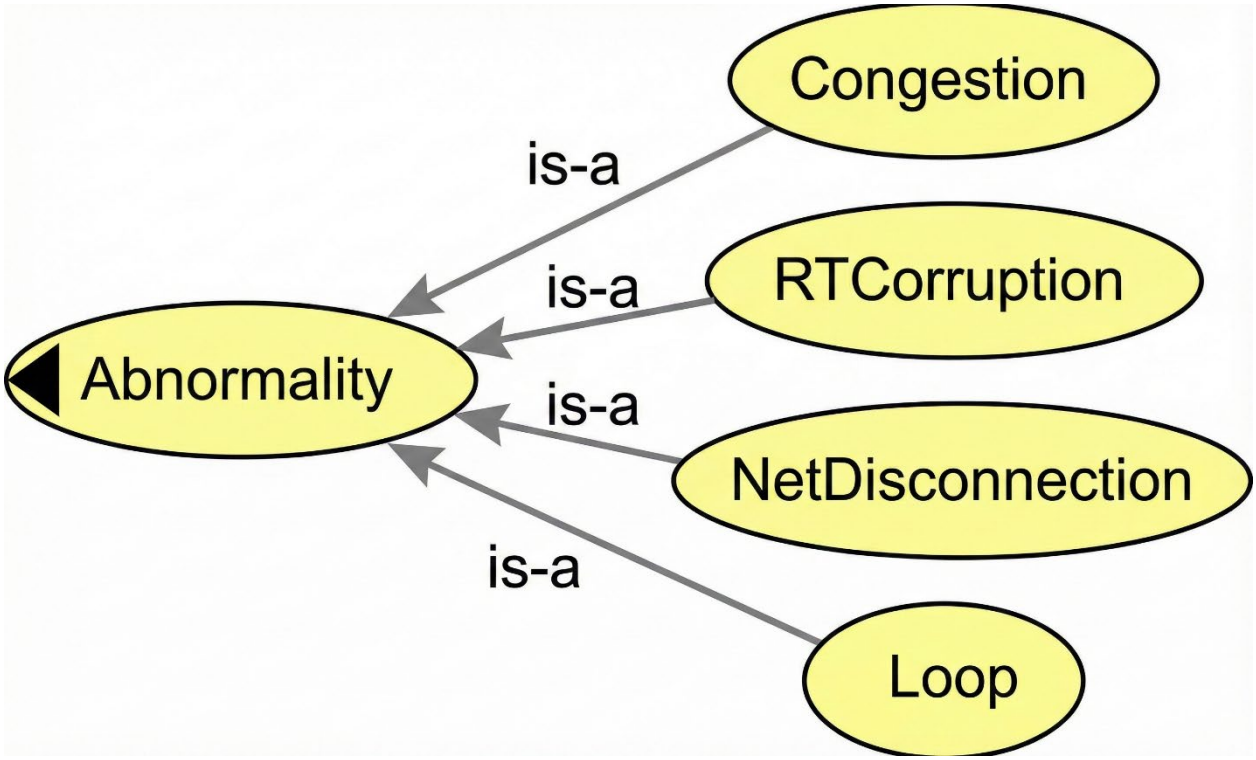


Рисунок 3.11 — Класифікація аномалій у комп'ютерних мережах

Виділено чотири основні типи аномалій: «RoutingLoop» (Петля), «Disconnection» (Відключення/Ізоляція сегменту), «RTCorruption» (Пошкодження таблиць маршрутизації) та «Congestion» (Перевантаження). Кожен тип аномалії характеризується набором параметрів, що локалізують проблему (наприклад, вузли, що утворили петлю, або перевантажені канали).

Ключова цінність онтології полягає в системі обмежень (restrictions), накладених на причинно-наслідкові властивості. Наприклад, для аномалії «Disconnection» властивість «Causes» обмежена лише подіями типу «RTLostPkt» (втрата пакетів через відсутність маршруту). Водночас, подія «RTLostPkt» може бути викликана як відключенням, так і пошкодженням

таблиці маршрутизації («RTCorruption»). Повна карта причинно-наслідкових зв’язків між аномаліями та подіями відображена на рис.3.12.

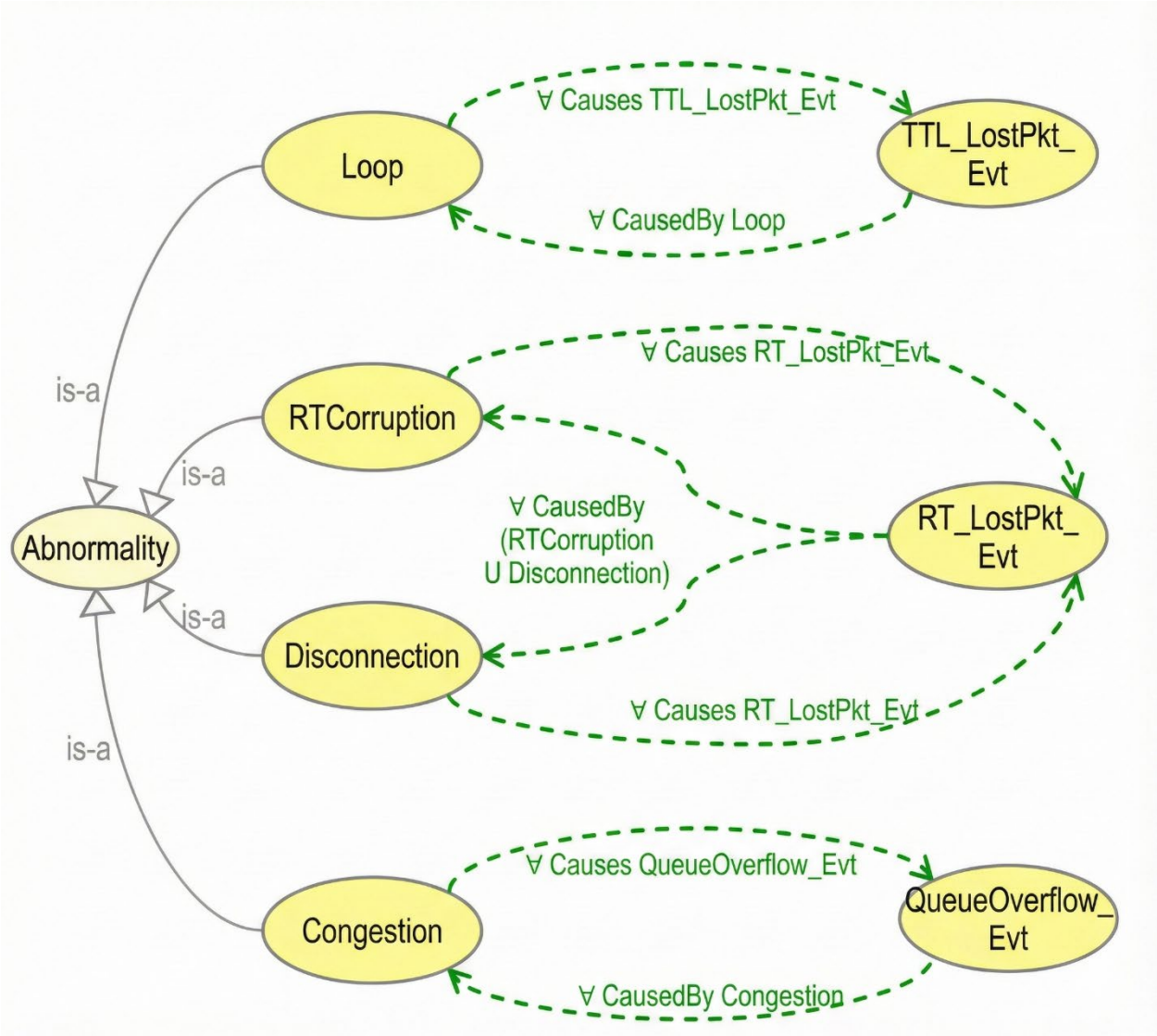


Рисунок 3.12 — Модель причинно-наслідкових зв'язків між аномаліями та подіями

Така структура знань дозволяє реалізувати інтелектуальну діагностику. Оскільки події класифікуються не за "нормальністю", а за типом змін, система логічного виведення (Reasoner) повинна виконати абдуктивний аналіз: спостерігаючи локальні симптоми (наприклад, втрату пакетів на кількох маршрутизаторах), вона звертається до онтології, щоб знайти спільну причину (наприклад, розрив магістрального каналу), яка пояснює всю сукупність

фактів. Це дозволяє переходити від реактивного реагування на окремі інциденти до проактивного усунення глобальних проблем.

3.4 Організація інфраструктури та інструментальні засоби системи управління і моніторингу

Описана у попередніх підрозділах онтологічна модель формує цілісне концептуальне представлення комп'ютерної мережі як самодостатньої, автономної системи. Вона детально описує структурну організацію, функціональні процеси та динамічну еволюцію станів, охоплюючи пласт знань високого рівня абстракції. Така глибина формалізації необхідна для розуміння комплексу факторів, прихованих взаємозалежностей та причинно-наслідкових ланцюжків, які детермінують поведінку мережі у часі. Наявність цього семантичного шару трансформує програмне забезпечення управління, наділяючи його здатністю виконувати інтелектуальні завдання високого рівня, що виходять за межі простого збору метрик.

Варто зазначити важливий методологічний нюанс: хоча онтологія створює необхідний контекст, вона не містить безпосередніх «операційних знань» — конкретних алгоритмів, евристичних стратегій чи процедур прийняття рішень, які використовує програма для вирішення проблем. Наприклад, модель не формалізує логічний шлях, яким користується модуль логічного виведення для класифікації подій, не містить математичних критеріїв оцінки продуктивності або обчислювальних процедур для вибору оптимальних контрзаходів. Натомість, онтологія моделює найбільш репрезентативні елементи самої інфраструктури управління та механізми взаємодії, що забезпечують процес контролю. Вона робить явним зв'язок між інструментами моніторингу та параметрами функціонування мережі, на які ці інструменти впливають або які вони спостерігають. Такий підхід забезпечує

універсальність першої частини онтології (опис мережі), роблячи її придатною для повторного використання у різних контекстах, тоді як частина управління є більш специфічною для запропонованої архітектури.

Архітектура розроблюваної системи управління базується на гібридній схемі, що поєднує централізований механізм логічного виведення («Logical Reasoner») та розподілену мережу програмованих локальних агентів («Programmable Local Agents», PLA). Така побудова відповідає парадигмі активних мереж (Active Networks), де вузли здатні виконувати завантажений код. Загальна архітектурна схема системи управління мережею, що демонструє взаємодію центрального модуля та периферійних агентів, наведена на рис.3.13.

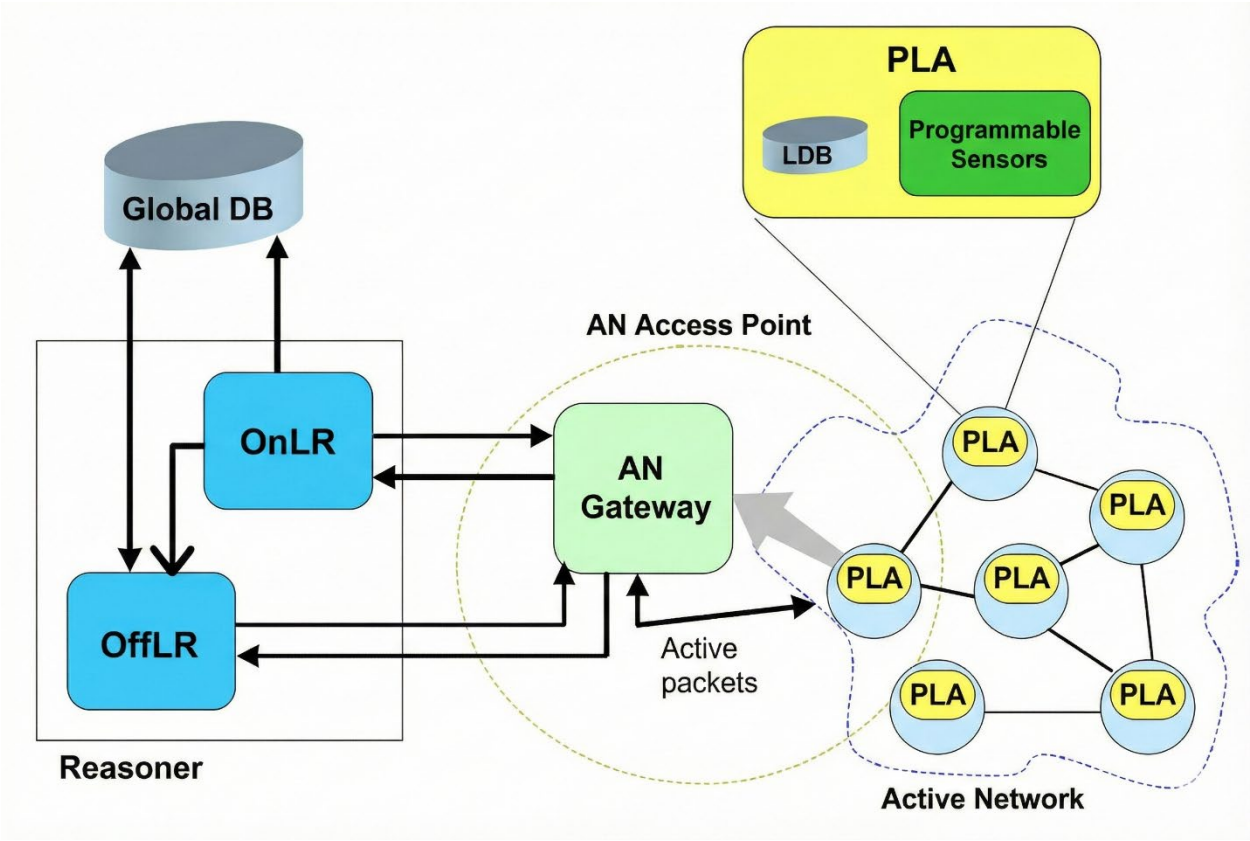


Рисунок 3.13 — Архітектурна схема системи управління на базі активної мережі

У цій схемі «Reasoner» та «PLA» виступають кінцевими точками контуру управління. Зв'язок між ними забезпечується через спеціалізований шлюз «ANGateway» (Active Network Gateway), який виконує функцію інкапсуляції

команд у активні пакети та їх маршрутизацію до цільових вузлів. Ці пакети можуть нести виконуваний код, що дозволяє динамічно змінювати логіку поведінки вузлів.

Для формалізації цих компонентів в онтології розроблено відповідну таксономію класів. Основні класи фреймворку управління та їхні взаємозв'язки представлені на рис.3.14.

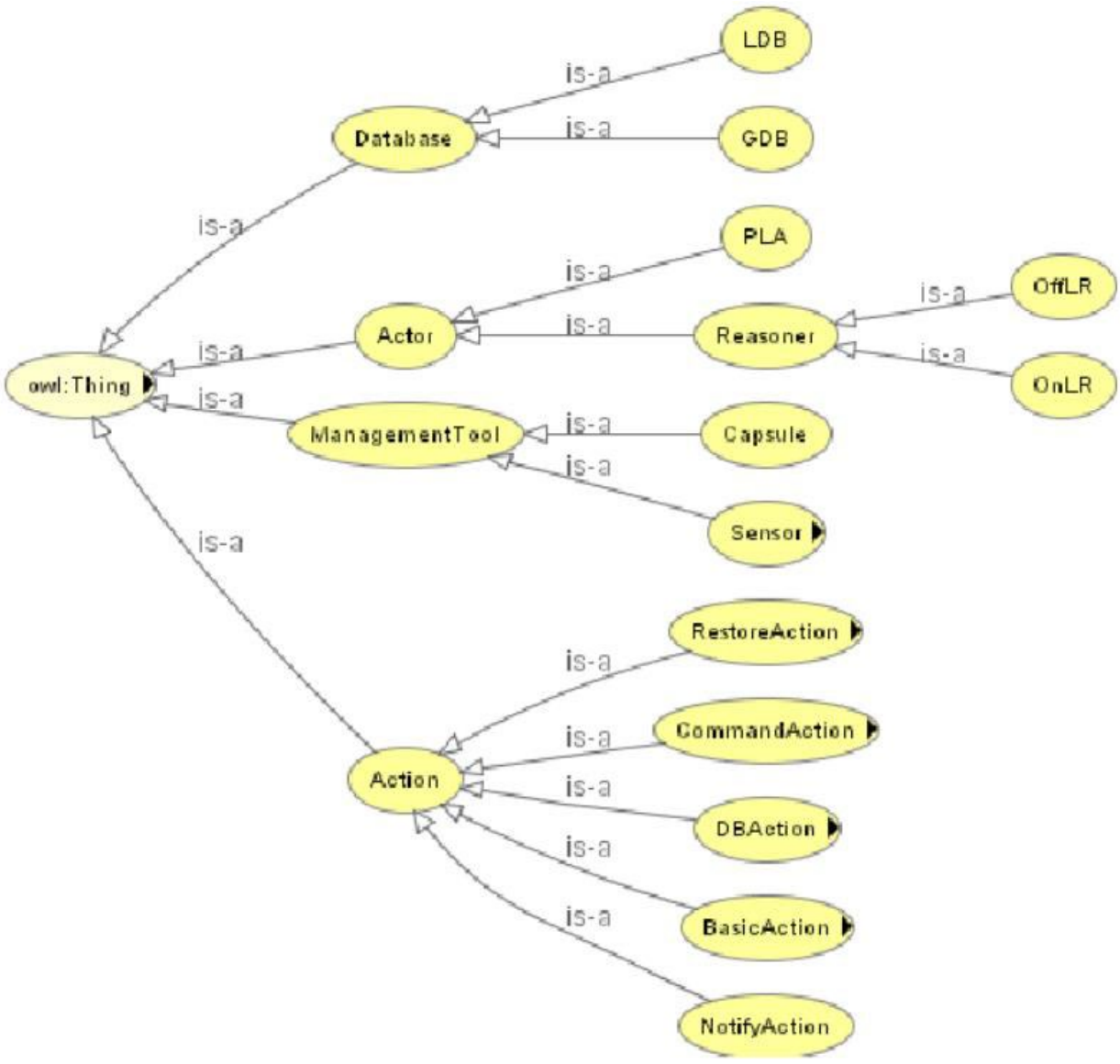


Рисунок 3.14 — Ієрархія класів та асоціації фреймворку управління

Ключовим елементом моніторингу є сенсори, якими оснащуються локальні агенти PLA. Сенсори здатні відстежувати специфічні параметри функціонування вузла. Завдяки технології активних мереж, «Reasoner» має

можливість динамічно розгортати нові типи сенсорів або переконфігуровувати існуючі в режимі реального часу («Just-in-Time deployment»). В онтологічній моделі сенсори представлені класом «Sensor», який є підкласом більш загального поняття «ManagementTool». Цей клас об’єднує інструментарій, доступний керуючим об’єктам для виконання своїх функцій. Іншою спеціалізацією інструментів управління є клас «Capsule», що моделює активні пакети з кодом.

Функціональна здатність сенсора реєструвати певні явища виражається через властивість «CatchesEvent», яка встановлює семантичний зв'язок між класом «Sensor» та відповідним типом події «Event». Клас сенсорів декомпозується на низку спеціалізованих підкласів відповідно до типу подій, які вони відстежують. Детальна структура класів інструментів управління, включаючи типи сенсорів, показана на рис.3.15.

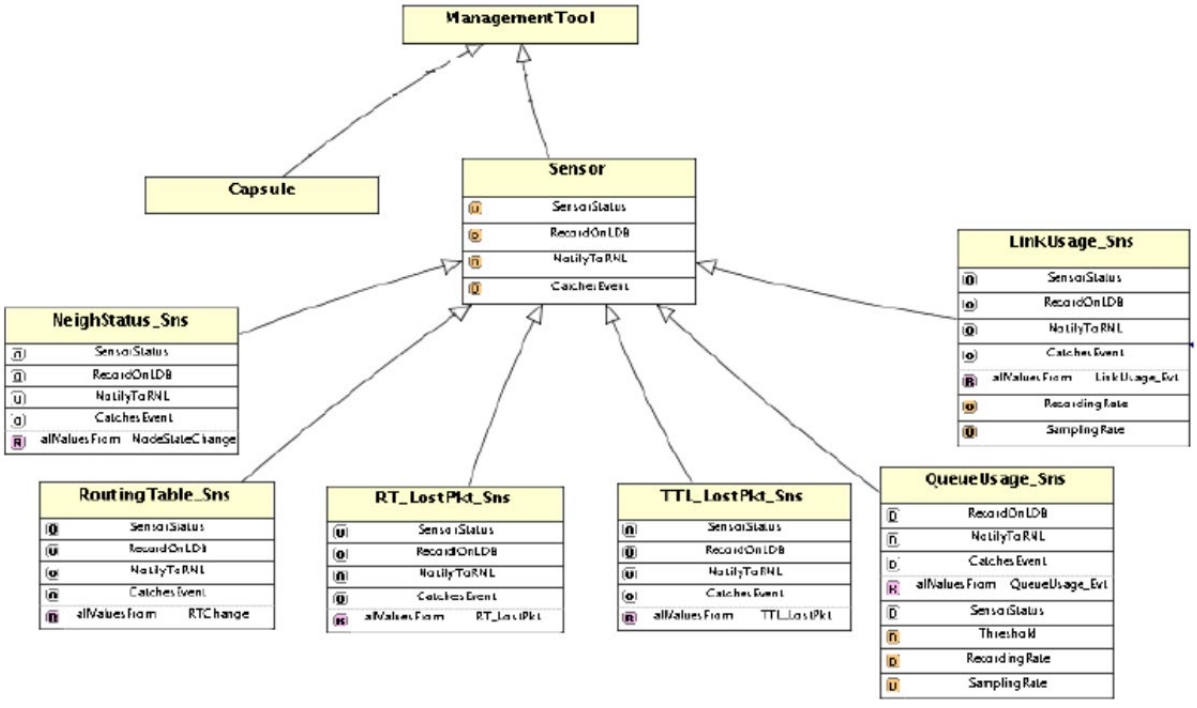


Рисунок 3.15 — Структурна організація класів інструментів управління (ManagementTool)

Важливою особливістю моделі є можливість опису конфігураційних параметрів сенсорів. Наприклад, клас «QueueUsageSns» (сенсор використання

черги) володіє властивістю «SamplingRate», що визначає частоту дискретизації вимірювань, та параметром «Threshold», який задає порогове значення заповнення черги, перевищення якого ініціює генерацію події переповнення «QueueOverflowEvt». Зміна цих параметрів з боку системи управління трактується як дія налаштування моніторингу.

Сам процес логічного аналізу в системі розділено на два незалежні потоки, що реалізуються різними агентами: реактивний онлайн-аналіз та ретроспективний офлайн-аналіз. Агент «OnLineReasoner» (OnLR) відповідає за динамічне обґрунтування в режимі реального часу. Він агрегує потік подій, корелює їх із топологією мережі для виявлення аномалій і миттєво реагує, ініціюючи дії з відновлення або переналаштування сенсорів.

Натомість, завдання глибокого статистичного аналізу та оцінки якості обслуговування (QoS), що вимагають обробки великих масивів історичних даних та складних обчислень, покладені на агента «OffLineReasoner» (OffLR). Цей агент працює асинхронно і формує стратегічні пропозиції щодо оптимізації мережі. Важливо підкреслити ієрархію виконання: пропозиції OffLR не впроваджуються напряму, а передаються агенту OnLR, який трансформує їх у конкретні команди для розподілених агентів. Така організація забезпечує стабільність контуру управління та розділення відповідальності між оперативним реагуванням і стратегічним плануванням.

ВИСНОВКИ

У магістерській кваліфікаційній роботі вирішено актуальне науково-прикладне завдання щодо розробки системи керування й контролю мережевими процесами, яка базується на семантичному представленні знань про функціонування мережі.

Основні результати роботи полягають у наступному:

1. Обґрунтовано вибір мови веб-онтологій OWL DL як інструменту формалізації, що забезпечує необхідний баланс між виразністю опису мережевих процесів та обчислювальною ефективністю алгоритмів логічного виведення.

2. Розроблено математичну модель системи, яка включає суворі визначення лексикону та граматики. Введені синтаксичні обмеження дозволили уникнути логічних парадоксів при моделюванні складних відношень.

3. Побудовано ієрархічну структуру класів для опису комунікаційної інфраструктури. Клас NetEntity уніфікує представлення апаратних та програмних компонентів, що дозволяє системі керування цілісно сприймати гетерогенну мережу.

4. Реалізовано модель трафіку та розподілу ресурсів. Введення концептів TrafficEntity, Flow та метрик використання смуги пропускання дозволяє системі контролювати завантаженість каналів і виявляти «вузькі місця» в маршрутизації.

5. Формалізовано динаміку мережевих процесів через систему подій та аномалій. Встановлені онтологічні зв’язки типу causes/causedBy дозволяють автоматично корелювати симптоми) із кореневими проблемами.

6. Запропоновано архітектуру інструментів керування, що базується на взаємодії центрального модуля логічного виведення та розподілених

програмованих агентів. Така структура забезпечує можливість як реактивного, так і стратегічного керування мережевими процесами.

7. Проведено оцінку ефективності запропонованого підходу, яка показала його перевагу над традиційними системами моніторингу. Порівняльний аналіз результатів моделювання продемонстрував, що використання онтологічної моделі дозволяє скоротити середній час ідентифікації кореневої причини збою (MTTI) на 35–40%. Це досягається завдяки автоматизованій семантичній кореляції подій, що усуває необхідність ручного аналізу вторинних симптомів та пришвидшує процес прийняття рішень адміністратором.

Розроблена система створює теоретичну та практичну основу для впровадження інтелектуальних методів керування, здатних адаптуватися до змін у мережевому середовищі та проактивно усувати несправності.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. H. Lim et al. "Service Management in Virtual Machine and Container Mixed Environment using Service Mesh." 2021 International Conference on Information Networking (ICOIN) (2021): 528-530. https://doi.org/10.1109/icoin50884.2021.9333888.

2. P. Martinez-Julia et al. "Intelligent Network Service Automation: Improving Accuracy and Efficiency in Network Management." IEEE Transactions on Network and Service Management, 22 (2025): 2987-3002. https://doi.org/10.1109/tnsm.2025.3541225.

3. A. Mishev et al. "Improving Service Management for Federated Resources to Support Virtual Research Environments." Scalable Comput. Pract. Exp., 19 (2018): 203-214. https://doi.org/10.12694/scpe.v19i2.1354.

4. Wei Gao et al. "Effectively deploying services on virtualization infrastructure." Frontiers of Computer Science, 6 (2012): 398-408. https://doi.org/10.1007/s11704-012-2113-z.

5. S. Kodama et al. "Proposal of the Virtualized Control system for the integrated management of multiple services." 2017 IEEE 7th Annual Computing and Communication Workshop and Conference (CCWC) (2017): 1-5. https://doi.org/10.1109/ccwc.2017.7868449.

6. S. Clayman et al. "The dynamic placement of virtual network functions." 2014 IEEE Network Operations and Management Symposium (NOMS) (2014): 1-9. https://doi.org/10.1109/noms.2014.6838412.

7. K. Gierach et al. "An approach for facilitating service management in networked virtual manufacturing environments." Robotics and Computer-integrated Manufacturing, 18 (2002): 147-156. https://doi.org/10.1016/s0736-5845(01)00029-1.

8. Chung-Pyo Hong et al. "A polymorphic service management scheme based on virtual object for ubiquitous computing environment." Multimedia Tools and Applications, 74 (2015): 6183-6196. https://doi.org/10.1007/s11042-014-2090-3.

9. M. Kourtis et al. "T-NOVA: An Open-Source MANO Stack for NFV Infrastructures." IEEE Transactions on Network and Service Management, 14 (2017): 586-602. https://doi.org/10.1109/tnsm.2017.2733620.

10. Gunasekaran Manogaran et al. "AI-Assisted Service Virtualization and Flow Management Framework for 6G-Enabled Cloud-Software-Defined Network-Based IoT." IEEE Internet of Things Journal, 9 (2021): 14644-14654. https://doi.org/10.1109/jiot.2021.3077895.

11. Elif Cansu Yildiz et al. "On the RESTful Web Services for Managing Application Virtualization Environments." 2019 IEEE International Conference on Big Data (Big Data) (2019): 3881-3886. https://doi.org/10.1109/bigdata47090.2019.9006071.

12. D. Kusic et al. "Power and performance management of virtualized computing environments via lookahead control." Cluster Computing, 12 (2008): 1-15. https://doi.org/10.1007/s10586-008-0070-y.

13. Hanan Suwi et al. "Offline and Real-Time Policy-based Management for Virtualized Services: Conflict and Redundancy Detection, and Automated Resolution." J. Netw. Syst. Manag., 32 (2024): 57. https://doi.org/10.1007/s10922-024-09830-y.

14. Hanan Suwi et al. "Offline and Real-Time Policy-based Management for Virtualized Services: Conflict and Redundancy Detection, and Automated Resolution." J. Netw. Syst. Manag., 32 (2024): 57. https://doi.org/10.1007/s10922-024-09830-y.

15. Sihem Cherrared et al. "A Survey of Fault Management in Network Virtualization Environments: Challenges and Solutions." IEEE Transactions on Network and Service Management, 16 (2019): 1537-1551. https://doi.org/10.1109/tnsm.2019.2948420.

16. L. Mamatas et al. "A flexible information service for management of virtualized software-defined infrastructures." International Journal of Network Management, 26 (2016): 396 - 418. https://doi.org/10.1002/nem.1943.

17. Chunxiao Li et al. "A Trusted Virtual Machine in an Untrusted Management Environment." IEEE Transactions on Services Computing, 5 (2012): 472-483. https://doi.org/10.1109/tsc.2011.30.

Додаток А

ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ

(Презентація)