ДОДАТОК А
ГРАФІЧНИЙ МАТЕРІАЛ АТЕСТАЦІЙНОЇ РОБОТИ

Харківський національний університет радіоелектроніки

Факультет _____Комп’ютерних наук_____

Кафедра _____Інформаційних управляючих систем_____

АТЕСТАЦІНА РОБОТА
ГРАФІЧНИЙ МАТЕРІАЛ

_____Дослідження моделей і методів інформаційних технологій_____
_____обробки даних про автомобілі, які підлягають розмитненню_____
(тема роботи)

Студент гр.__ІУСТм-19-1__ _____ ____Асмоловський Б.І______
(шифр групи) (підпис) (прізвище, ініціали)

Керівник роботи _____ _____
(підпис) (посада, прізвище, ініціали)

2020 р.

Таблиця 1 – Мета і завдання роботи

Показник	Опис
Тема роботи	Дослідження моделей і методів адаптації інформаційного та програмного забезпечення інформаційної системи до нових вимог
Об’єкт дослідження	Процеси адаптації інформаційного та програмного забезпечення інформаційної системи до нових вимог
Предмет дослідження	Методи вирішення задачі адаптації інформаційного та програмного забезпечення інформаційної системи до нових вимог
Мета роботи	Дослідження моделей і методів адаптації інформаційного та програмного забезпечення інформаційної системи до нових вимог, які дозволили б зменшити часові та людські витрати на адаптацію інформаційного та програмного забезпечення інформаційної системи до нових вимог без значного зменшення її ефективності
Наукова новизна	Розроблено метод відображення онтологій під для вирішення задачі повторного використання програмного коду
Практична новизна	Проведено апробацію отриманих результатів під час проектування різних класів інтелектуальних систем, орієнтованих на спільне використання знань і забезпечення семантичної сумісності різних моделей подання та обробки даних і знань
Перечень задач	– аналіз проблеми повторного використання програмного коду; – огляд існуючих методів повторного використання програмного коду; – розробка моделей повторного використання програмного коду; – розробка методів повторного використання програмного коду; – практична апробація отриманих наукових результатів

Таблиця 1– Залежність ступеня автоматизації і методу створення зв'язку

Метод створення зв'язку	Ступінь автоматизації		
	Ручна	Автоматизована	Автоматична
Рівень брокерів	Відсутня	Дозволена	Бажана
Рівень даних	Дозволена	Дозволена	Бажана
Рівень сервісів	Дозволена	Бажана	Дозволена
Рівень інтерпретування метаінформації	Дозволена	Дозволена	Бажана

Рисунок 1– Класична чотирирівнева ієрархія моделей ІС

Таблиця 2 – Переваги та недоліки методів створення зв'язку в ІС

Метод створення зв'язку	Переваги	Недоліки
Рівень брокерів	Універсальність – можливість створити додатковий програмний модуль, який буде звертатися в обидві системи	Складність, трудомісткість, а отже, висока вартість розробки, впровадження та володіння
Рівень даних	Низька вартість інтеграції	Руйнування цілісності даних. Якщо БД не має обмежень цілісності, різні додатки можуть приводити дані в суперечливі стани, в іншому випадку в паралельно працюючих з однією БД додатках будуть дубльовані частини коду. При змінах структури БД необхідно переписувати код всіх додатків, які з нею працюють
Рівень сервісів	Низька вартість інтеграції, швидке об'єднання систем, що сильно розрізняються без їх модифікації і дописування додаткових модулів	Присутня фіксація; а якщо структури або процеси змінюються, то утворюються проблеми і вузькоспеціалізовані, приватні рішення. Необхідність застосування стандартизованих компонентів
Рівень інтерпретування метаінформації	Гнучкість підходу і спрощення модифікації систем, зниження необхідного рівня кваліфікації користувача при внесенні змін до структури і функцій системи	Складність створення програмного забезпечення з підвищеним рівнем абстракції, в реалізації мета-моделі і в зіставленні інформаційних ресурсів в різних системах метаданих

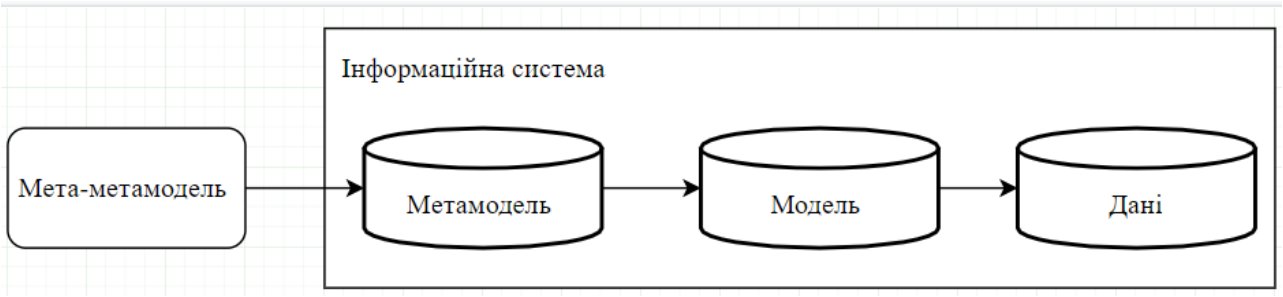

Рисунок 2 – Технологія DSM з інтерпретацією метаданих

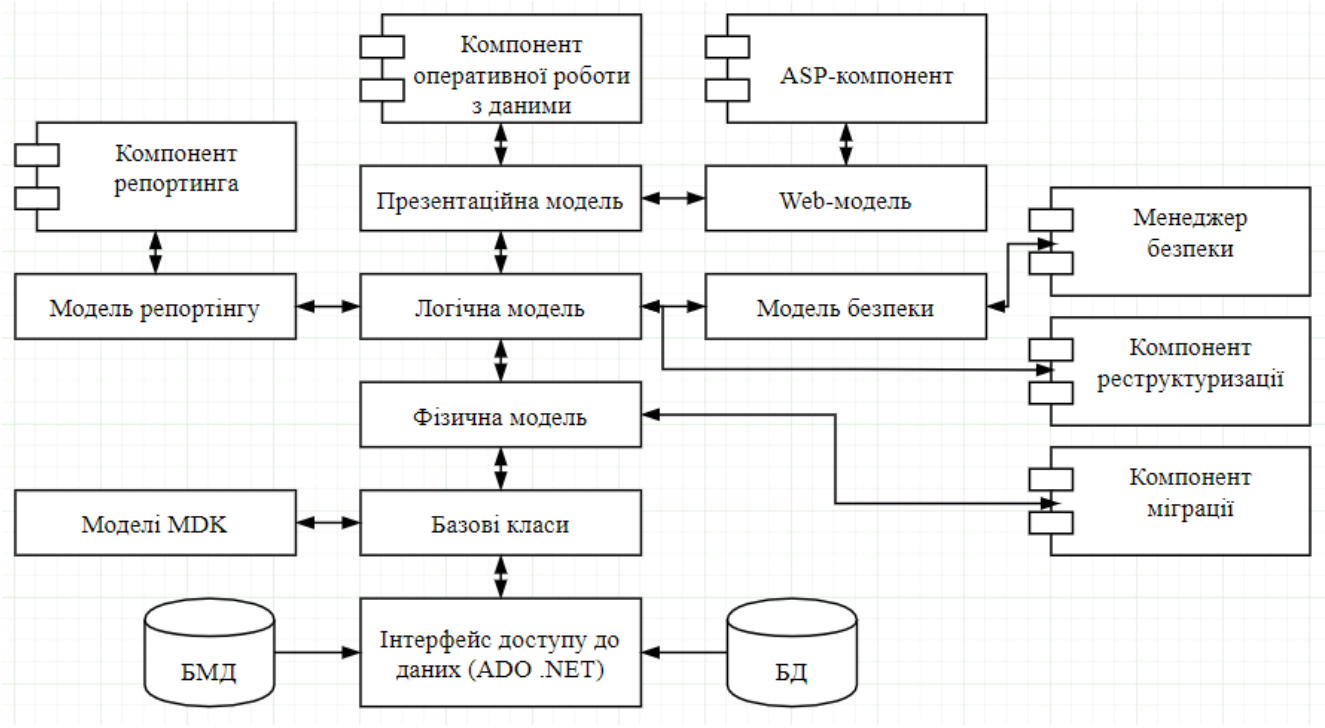

Рисунок 3 – Моделі метаданих та компоненти METAS

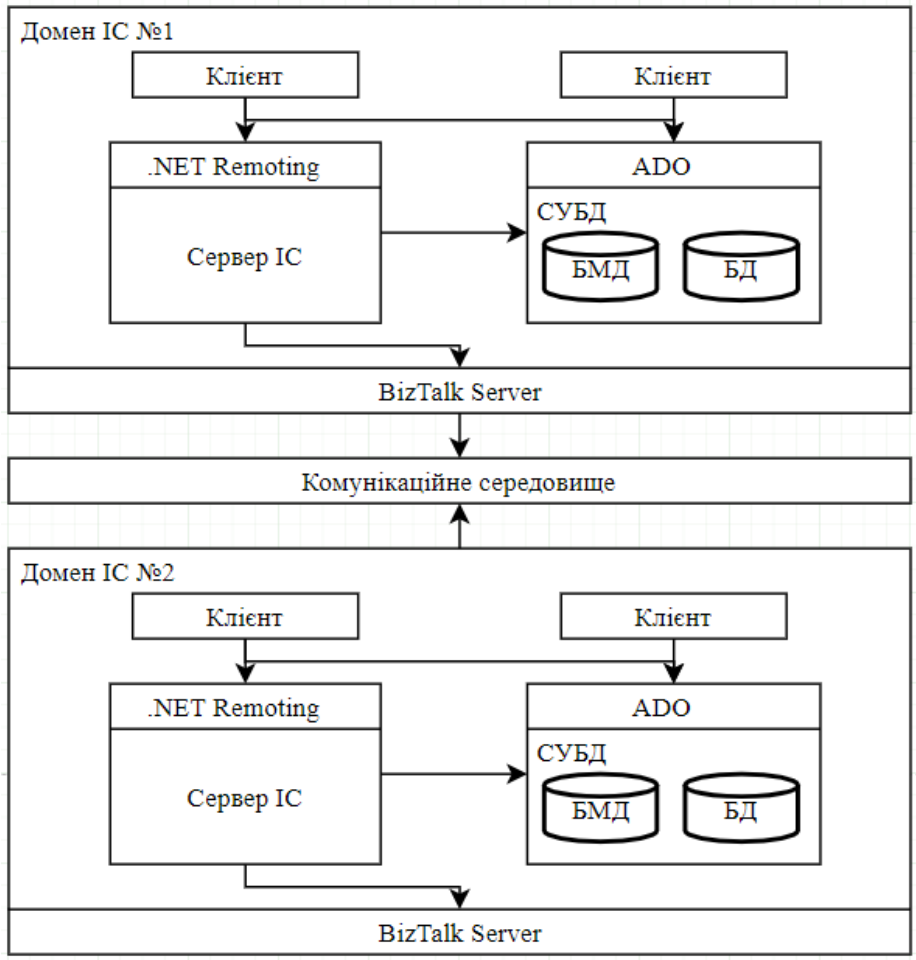

Рисунок 4 – Доменна архітектура ІС

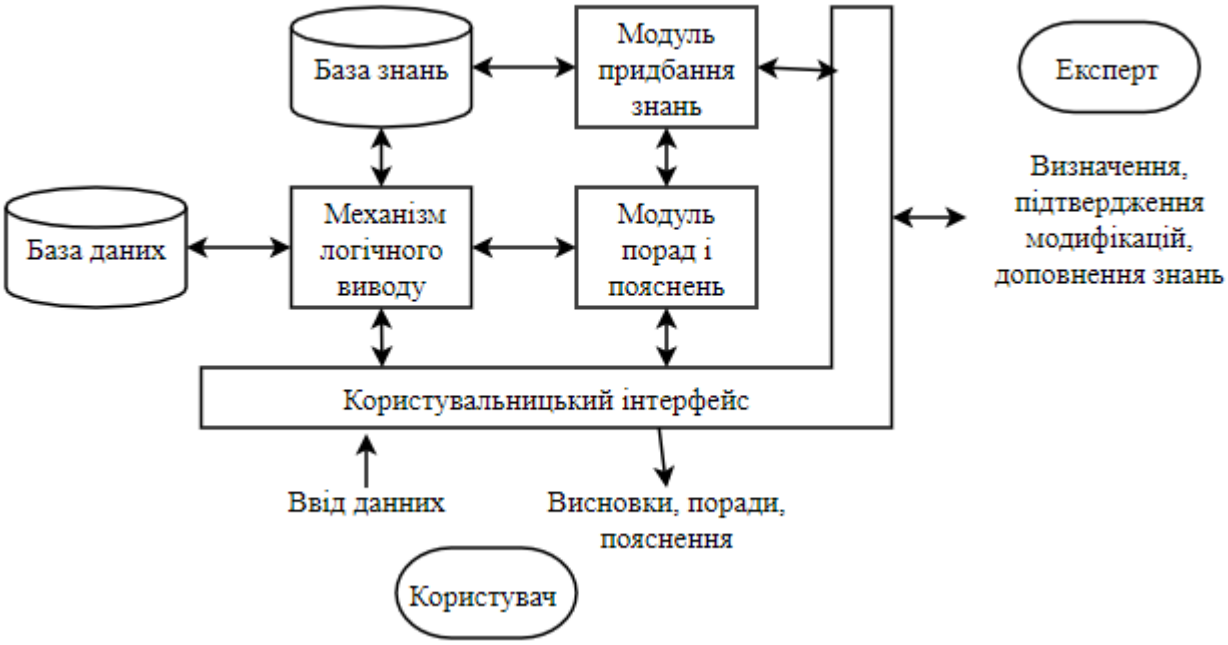

Рисунок 5 – Структура експертної системи

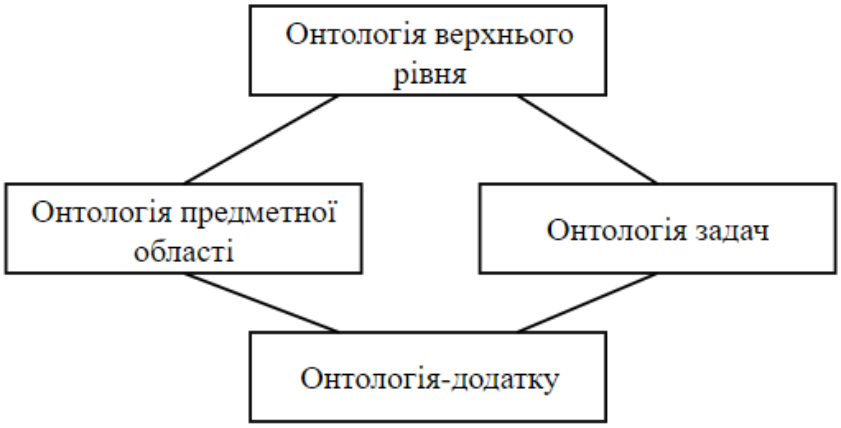

Рисунок 6 – Багаторівнева схема відносин між онтологіями

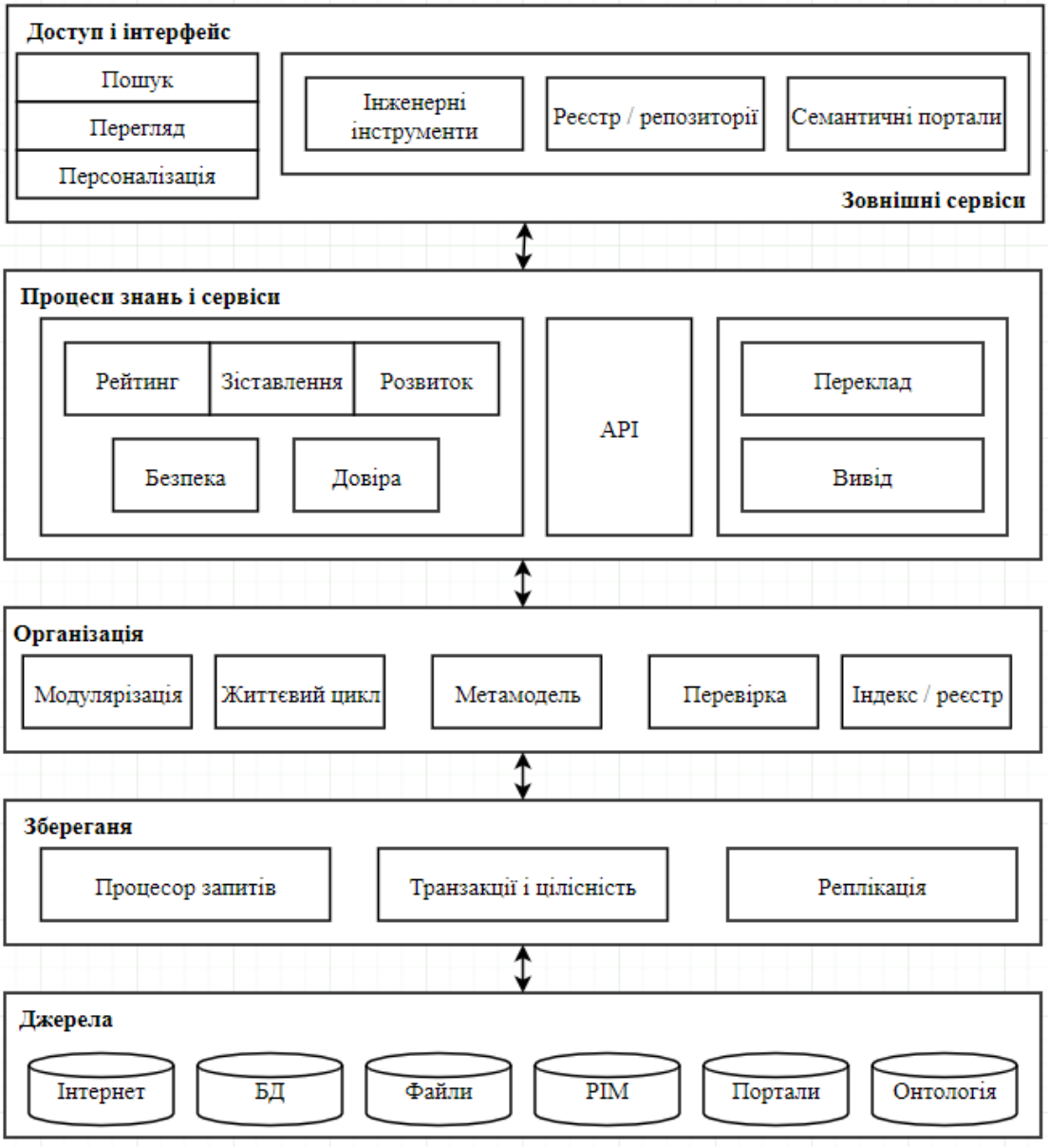

Рисунок 7– Концептуальна структура сховища онтологій

Таблиця 3 – Шаблон для елемента метаданих в OMV

Name	Ім'я елемента (entity) метаданих
Type	Тип онтологічного примітиву, використовуваного для подання елемента в OWL: Class, ObjectProperty або DatatypeProperty
Identifier	Унікальний ідентифікатор, який використовується для цього елемента
Occurrence Constraint	Один з наступних: required, optional або extensional
Category	Залежна від контенту категорія, до якої відноситься елемент
Definition	Коротке визначення мети, яка може бути детально описана в тегах коментарів
Domain	Предметна галузь елемента OMV (для властивостей OWL)
Range	Ранг елемента OMV entity (для властивостей OWL)
Cardinality	Потужність елемента OMV (MIN: MAX)
OMV version	Версія OMV, в якій представлений елемент
Comments	Детальний опис елемента

```
<owl:Class rdf:ID=”id1”>
        <rdfs:label>комп’ютерна мишка<label/>
</owl:Class>
<owl:Class rdf:ID=”id2”>
        <rdfs:label>комп’ютерна миша<label/>
</owl:Class>
```

Рисунок 8 – Приклад 1

```
<owl:Class rdf:ID="Регіон"/>

<owl:Class rdf:ID=»РегіонЦентральногоПобережжя»/>
```

Рисунок 9 – Приклад 2

```
<Регион rdf:ID="РегионГорыСантаКруз">
        <locatedIn rdf:resource="#РегионКалифорния"/> </Регион>
<Винодельня rdf:ID="ВиноградникГораСантаКруз"/>
<КабернеСовиньон
        rdf:ID="КабернеСовиньонВинодельняГораСантаКруз">
                <расположенВ rdf:resource="#РегионГорыСантаКруз"/>
        rdf:resource="#ВиноградникГораСантаКруз"/> </КабернеСовиньон>
<ВиноградКабернеСовиньон
        rdf:ID="ВиноградКабернеСовиньонВинодельняГораСантаКруз">
                <расположенВ rdf:resource="#РегионГорыСантаКруз"/>
        rdf:resource="#ВиноградникГораСантаКруз"/> </ВиноградКабернеСовиньон>
```

Рисунок 10 – Приклад 3

```
<owl:ObjectProperty rdf:ID="ЗробленоІзВінограду">
        rdfs:domain rdf:resource="#Вино"/>
        <rdfs:range rdf:resource="#Виноград"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="ОтриманоІзВінограду">
         <rdfs:domain rdf:resource="#Вино"/>
         <rdfs:range rdf:resource="#Виноград"/>
</owl:ObjectProperty>
```

Рисунок 11 – Приклад 4

```
<Owl: Class rdf: ID = "література" />
<Owl: Class rdf: ID = "книги">
        <Rdfs: subClassOf rdf: resource = "# література" /> </ Owl: Class>
<Owl: Class rdf: ID = "розумні люди">  <Owl: Restriction>
        <Owl: onProperty rdf: resource = "# читають" />
        <Owl: hasValue rdf: resource = "# книги" /> </ Owl: Restriction> </ Owl: Class>
<Owl: Thing rdf: ID = "Сергій"> <Rdfs: type rdf: resource = "# розумні люди" /> </ Owl: Thing>
```

Рисунок 12 – Приклад 5

```
<Owl: Class rdf: ID = "Напій"> <Rdfs: subClassOf rdf: resource = "#ПродуктХарчування" /> </ Owl: Class>
<Owl: Class rdf: ID = "Їжа"> <Rdfs: subClassOf rdf: resource = "#ПродуктХарчування" /> </ Owl: Class>
```

Рисунок 13 – Приклад 6

```
<Owl: Class rdf: ID = "Їжа" />
<Owl: Class rdf: ID = "Страва" />
<Owl: Class rdf: ID = "Хліб">
        <Rdfs: subClassOf rdf: resource = "#Їжа" />
        <Rdfs: subClassOf rdf: resource = "#Страва" /> </ Owl: Class>
```

Рисунок 14 – Приклад 7

```
<Owl: Class rdf: ID = "Автомобіль" /> <Owl: Class rdf: ID = "Машина" />
<Owl: Class rdf: ID = "Порш" />
        <Rdfs: subClassOf rdf: resource = "# Автомобіль" /> </ Owl: Class>
<Owl: Class rdf: ID = "Жигулі" />
        <Rdfs: subClassOf rdf: resource = "# Автомобіль" />
        <Rdfs: subClassOf rdf: resource = "# Машина" /> </ Owl: Class>
<Owl: Class rdf: ID = "Мерседес" />
        <Rdfs: subClassOf rdf: resource = "# Машина" /> </ Owl: Class>
```

Рисунок 15 – Приклад 8

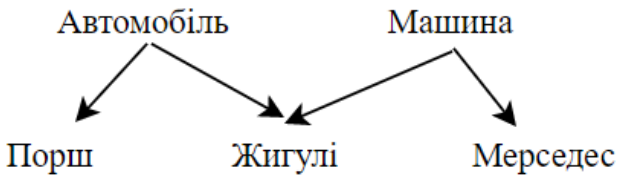

Рисунок 16 – Візуалізація прикладу 8

```
<Owl: Class rdf: ID = "ХарактеристикаВина" /> <Owl: Class rdf: ID = "КолірВина">
        <Rdfs: subClassOf rdf: resource = "# ХарактеристикаВина" /> </ Owl: Class>
<Owl: ObjectProperty rdf: ID = "МаєХарактеристикуВина">
        <Rdfs: domain rdf: resource = "# Вино" />
        <Rdfs: range rdf: resource = "#ХарактеристикаВина" /> </ Owl: ObjectProperty>
<Owl: ObjectProperty rdf: ID = "МаєКолір">
        <Rdfs: subPropertyOfrdf: resource = "#ВолодієХарактеристикоюВина" />
        <Rdfs: range rdf: resource = "# КолірВина" /> </ Owl: ObjectProperty>
<Owl: ObjectProperty rdf: ID = "МаєВідтінок">
        <Rdfs: subPropertyOf rdf: resource = "#ВолодієХарактеристикоюВина" />
        <Rdfs: range rdf: resource = "#КолірВина" /> </ Owl: ObjectProperty>
```

Рисунок 17 – Приклад 9

```
<Owl: Class rdf: ID = "Автомобіль" /> <Owl: Class rdf: ID = "Машина" />
<Owl: Thing rdf: ID = "ПоршСергія"> <Rdfs: type rdf: resource = "# Автомобіль" />
        <Rdfs: type rdf: resource = "# машина" /></ Owl: Thing>
<Owl: Thing rdf: ID = "МерседесСергія"> <Rdfs: type rdf: resource = "# Автомобіль" />
        <Rdfs: type rdf: resource = "# Машина" /></ Owl: Thing>
```

Рисунок 18 – Приклад 10

```
<Owl: Class rdf: ID = "Автомобіль" />
<Owl: Thing rdf: ID = "ПоршСергія">
<Rdfs: type rdf: resource = "# Автомобіль" /> </ Owl: Thing>
<Owl: Thing rdf: ID = "МерседесСергія">
<Rdfs: type rdf: resource = "# Автомобіль" /> </ Owl: Thing>
```

Рисунок 19 – Приклад 11

```
<Owl: Class rdf: ID = "Автомобіль" /> <owl: Class rdf: ID = "Машина" />
<Owl: Thing rdf: ID = "ПоршСергія">
        <Rdfs: type rdf: resource = "# Автомобіль" />
        <Rdfs: type rdf: resource = "# Машина" /> </ owl: Thing>
<Owl: Thing rdf: ID = "МерседесСергія">
        <Rdfs: type rdf: resource = "# Автомобіль" />
        <Rdfs: type rdf: resource = "# Машина" /> </ owl: Thing>
```

Рисунок 20 – Приклад 12

```
<Owl: ObjectProperty rdf: ID = "РозташованийВ">
        <Rdf: type rdf: resource = "& owl; TransitiveProperty" />
        <Rdfs: domain rdf: resource = "& owl; Thing" />
        <Rdfs: range rdf: resource = "# Регіон" /> </ Owl: ObjectProperty>
<Регіон rdf: ID = "РегіонГориСантаКруз"> <РасположенВ rdf: resource = "# РегіонСША" /> </ Регіон>
<Регіон rdf: ID = "РегіонКаліфорнія"> <РасположенВ rdf: resource = "# РегіонСША" /> </ Регіон>
```

Рисунок 21 – Приклад 13

```
<Owl: ObjectProperty rdf: ID = "РозташованийВ">
        <Rdf: type rdf: resource = "& owl; TransitiveProperty" />
        <Rdfs: domain rdf: resource = "& owl; Thing" />
        <Rdfs: range rdf: resource = "# Регіон" /> </ Owl: ObjectProperty>
<Owl: ObjectProperty rdf: ID = "ЗнаходитьсяВ">
        <Rdf: type rdf: resource = "& owl; TransitiveProperty" />
        <Rdfs: domain rdf: resource = "& owl; Thing" />
        <Rdfs: range rdf: resource = "# Регіон" /> </ Owl: ObjectProperty>
<Регіон rdf: ID = "РегіонГориСантаКруз"> <РозташованийВ rdf: resource = "# РегіонСША" /> </
Регіон>
<Регіон rdf: ID = "РегіонКаліфорнія"> <ЗнаходитьсяВ rdf: resource = "# РегіонСША" /> </ Регіон>
```

Рисунок 22 – Приклад 14

```
<Вино rdf: ID = "Улюблена Вино Михайла">
        <Owl: sameAs rdf: resource = "# StGenevieve Техаське Біле" /></ Вино>
```

Рисунок 23 – Приклад 12

```
<owl:Class rdf:ID="БілеВіно">
        <owl:intersectionOf rdf:parseType="Collection">
                <owl:Class rdf:about="#Віно"/> <owl:Restriction>
                        <owl:onProperty rdf:resource="#МаєКолір"/>
                        <owl:hasValue rdf:resource="#Біле"/>
                </owl:Restriction> </owl:intersectionOf> </owl:Class>
<owl:Class rdf:ID="Шампанське">
         <owl:intersectionOf rdf:parseType="Collection">
        <owl:Class rdf:about="#Віно"/> <owl:Restriction>
                        <owl:onProperty rdf:resource="#МаєКолір"/>
                        <owl:hasValue rdf:resource="#Біле"/>
                </owl:Restriction> </owl:intersectionOf> </owl:Class>
```

Рисунок 24 – Приклад 16

Таблиця 4 – Приклад роботи критерію

Правило А	Правило В	Слідство
Якщо – брат) і (– батько), то (– батько)	Якщо – сестра) і (– батько), то (– батько)	З А і В випливає, що відносини «брат» і «сестра» подібні

```
<ruleml:imp> <ruleml:_rlab ruleml:href="#Приклад1"/> <ruleml:_body>
 <swrlx:individualPropertyAtom swrlx:property="МаєБрата">
        <ruleml:var>x1</ruleml:var>
        <ruleml:var>x2</ruleml:var>  </swrlx:individualPropertyAtom>
 <swrlx:individualPropertyAtom swrlx:property="МаєОтця">
        <ruleml:var>x2</ruleml:var>
        <ruleml:var>x3</ruleml:var>  </swrlx:individualPropertyAtom> </ruleml:_body>
<ruleml:_head>  <swrlx:individualPropertyAtom swrlx:property=" МаєОтця ">
        <ruleml:var>x1</ruleml:var>
        <ruleml:var>x3</ruleml:var>  </swrlx:individualPropertyAtom> </ruleml:_head> </ruleml:imp>
<ruleml:imp> <ruleml:_rlab ruleml:href="#Приклад2"/> <ruleml:_body>
 <swrlx:individualPropertyAtom swrlx:property="МаєСестру">
        <ruleml:var>x1</ruleml:var>
        <ruleml:var>x2</ruleml:var>  </swrlx:individualPropertyAtom>
 <swrlx:individualPropertyAtom swrlx:property=" МаєОтця ">
        <ruleml:var>x2</ruleml:var>
        <ruleml:var>x3</ruleml:var>  </swrlx:individualPropertyAtom> </ruleml:_body> <ruleml:_head>
 <swrlx:individualPropertyAtom swrlx:property=" МаєОтця ">
        <ruleml:var>x1</ruleml:var>
        <ruleml:var>x3</ruleml:var>  </swrlx:individualPropertyAtom> </ruleml:_head> </ruleml:imp>
```

Рисунок 25 – Приклад 17

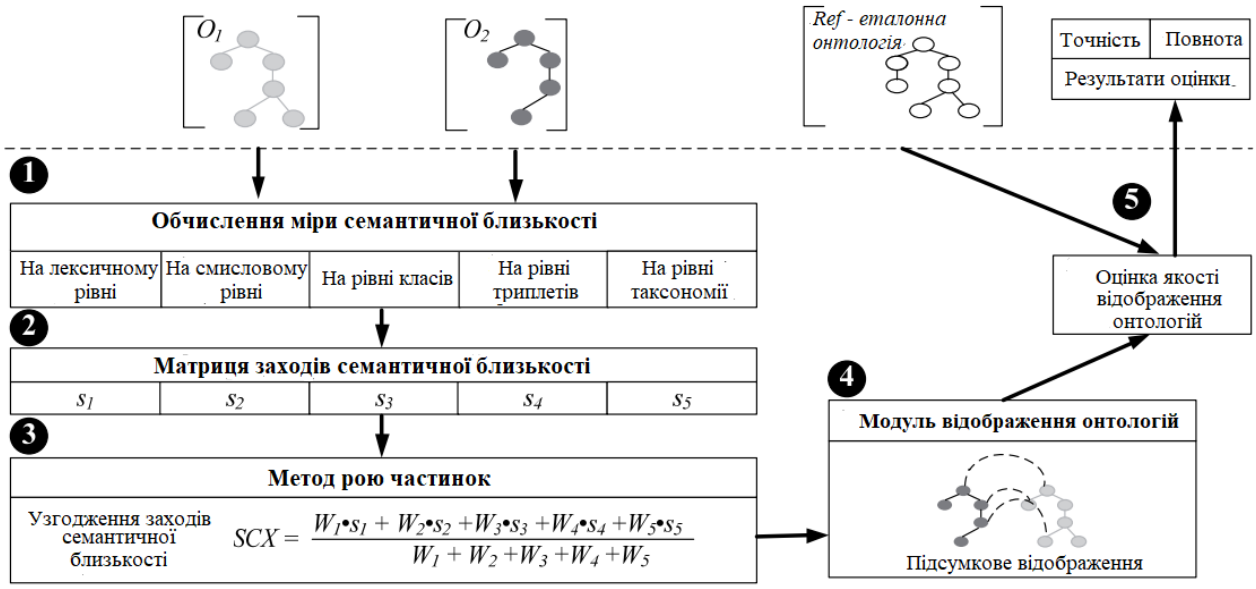

Рисунок 26 – Етапи відображення онтології з використання

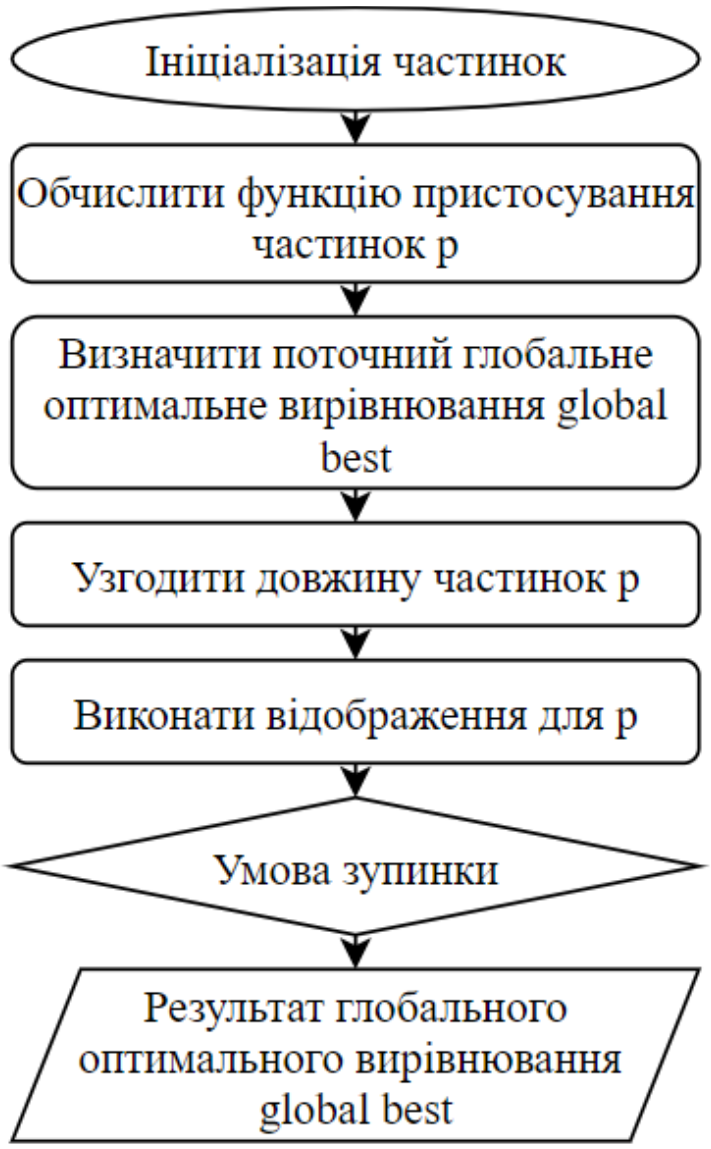

Рисунок 27 – Алгоритм оптимізації методом рою частинок

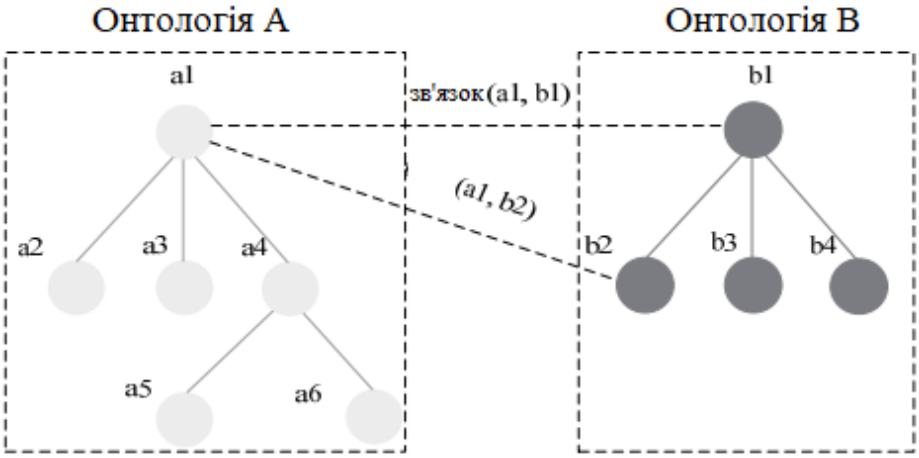

Рисунок 28 – Приклад відображення онтологій

Таблиця 5 – Результати одно і багатокритеріальної оптимізації

Підхід	Точність	Повнота пошуку
Багатокритеріальна оптимізація	0.81428	1.00
Однокритеріальна оптимізація (точність)	0.7142	0.86
Однокритеріальна оптимізація (повнота)	0.3333	1.00

Таблиця 6 – Зведення критеріїв подібності онтологій

Формулювання критерію	Критерій	Що?
Поняття подібні, якщо подібні <Що?>		мітки
		URI
		властивості
		батьківські поняття
		поняття того ж рівня
		дочірні поняття
		екземпляри
		вибірка примірників
		відношення «sameAs»
		обмеження
		правила
Атрибути подібні, якщо подібні <Що?>		мітки
		URI
		область і вектор
		батьківські атрибути
		дочірні атрибути
		пов'язані екземпляри
		відношення «sameAs»
		обмеження
		правила

Кінець таблиці 6

Формулювання критерію	Критерій	Що?
Примірники подібні, якщо подібні <Що?>		мітки
		URI
		батьківське поняття
		властивості і екземпляри
		відношення «sameAs»
		обмеження
		правила
		хеш-коди
		MIPE-type

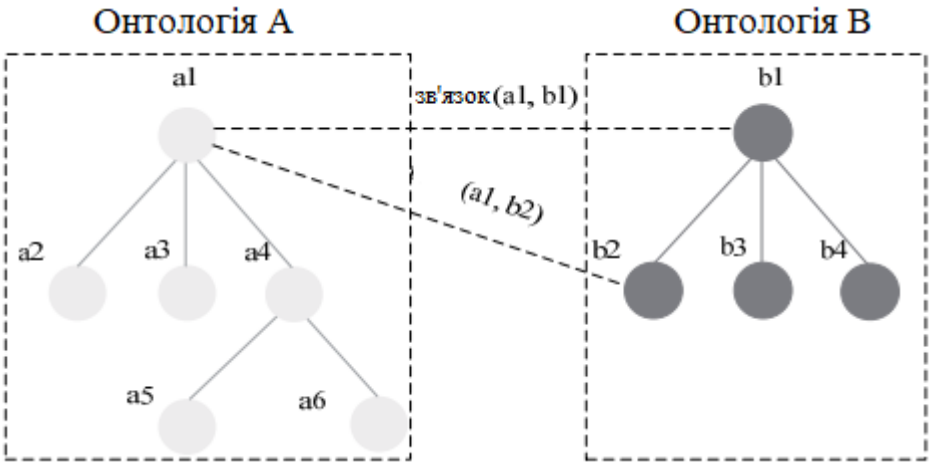

Рисунок 29 – Приклад відображення онтологій

Таблиця 7 – Результати одно і багатокритеріальної оптимізації

Підхід	Точність	Повнота пошуку
Багатокритеріальна оптимізація	0.81428	1.00
Однокритеріальна оптимізація (точність)	0.7142	0.86
Однокритеріальна оптимізація (повнота)	0.3333	1.00

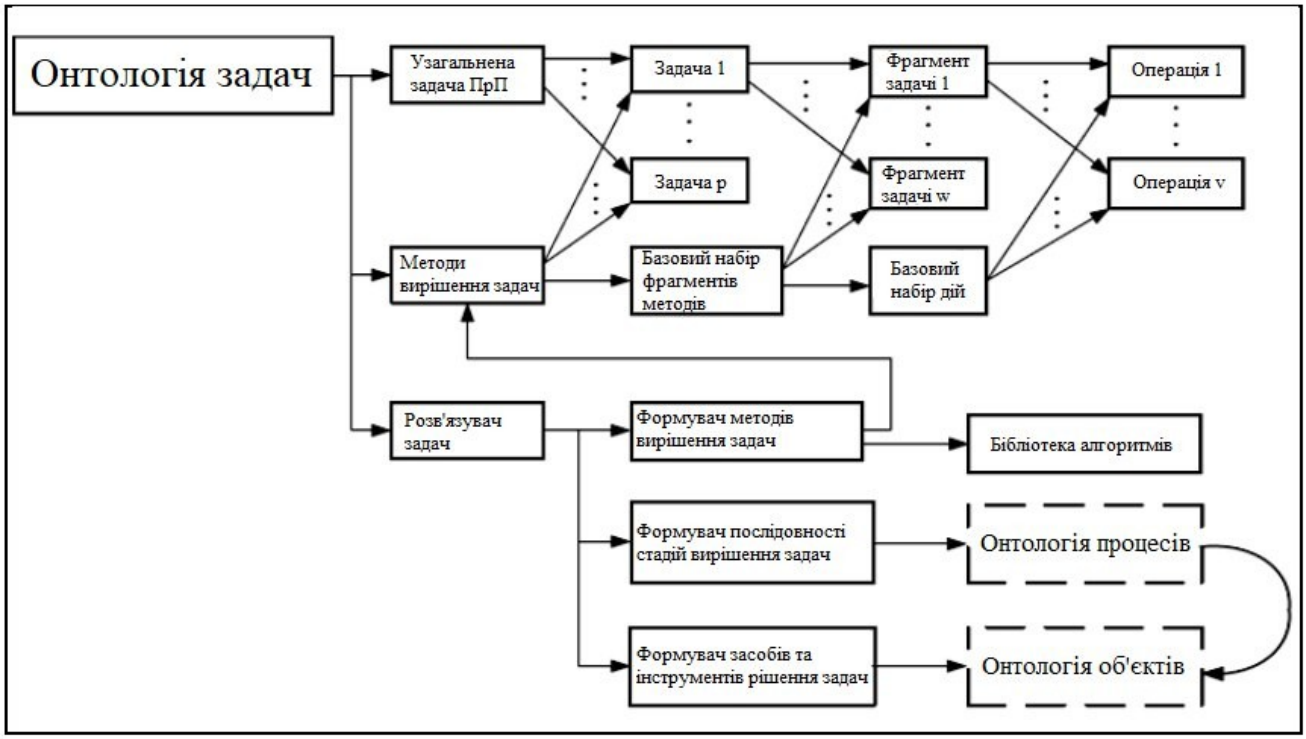

Рисунок 30 – Схема онтології задач

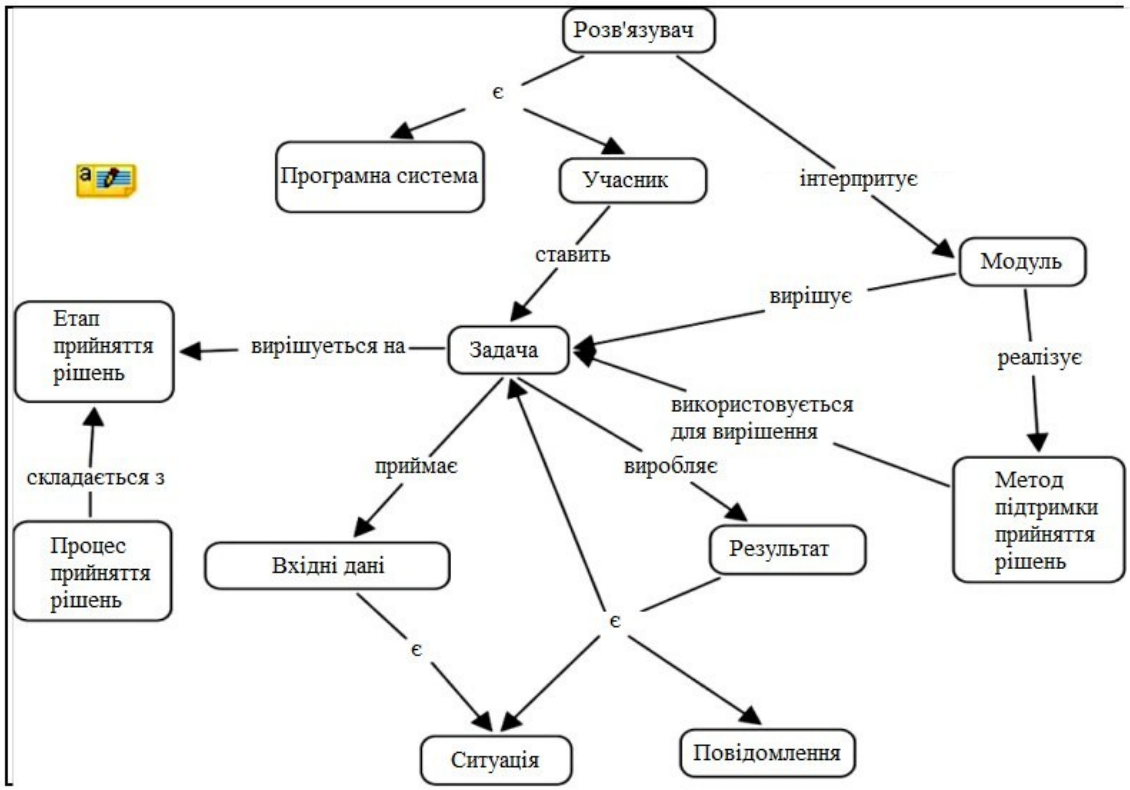

Рисунок 31 – Онтологія розв'язувача задач

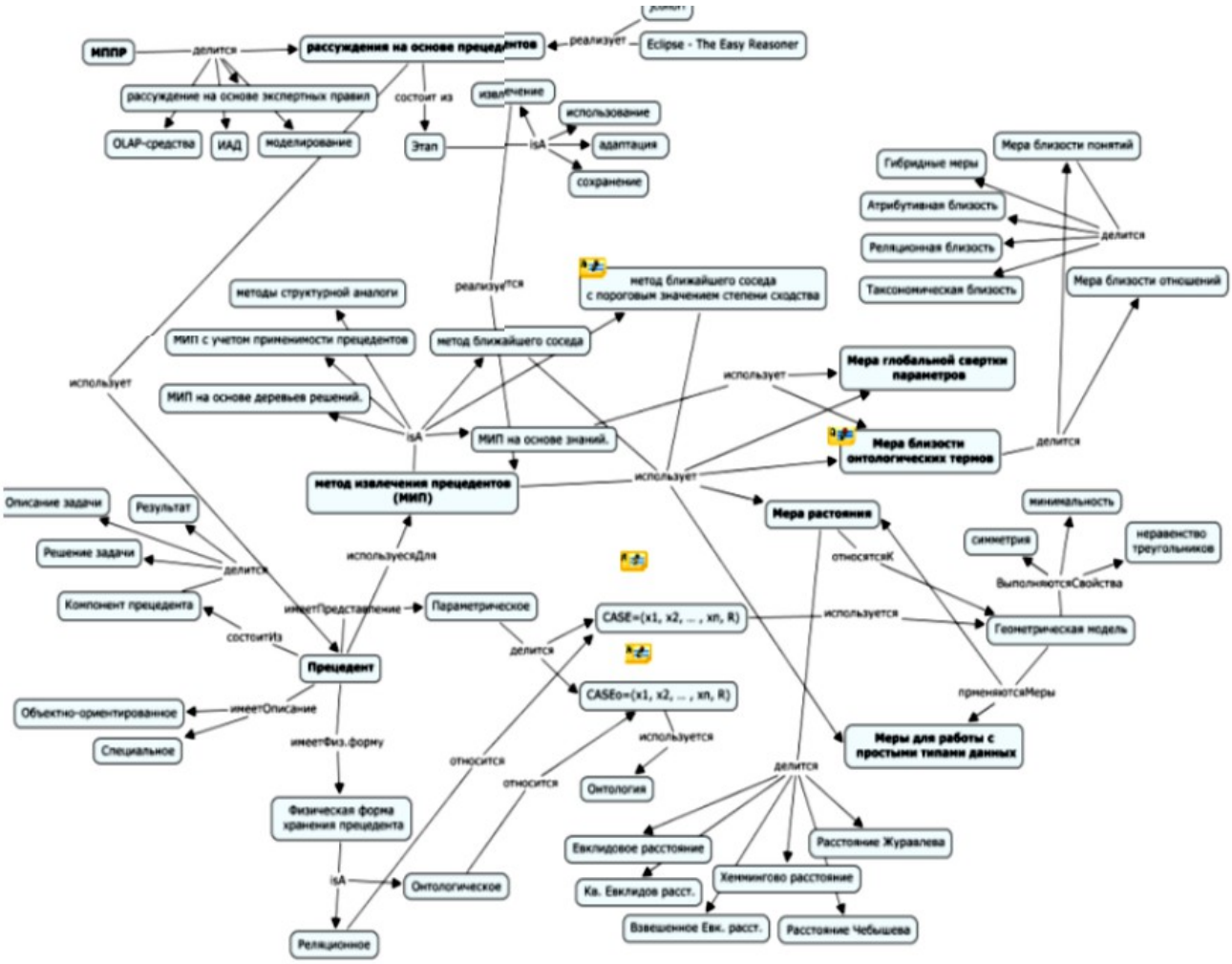

Рисунок 32 – Інтегруюча онтологія