ДИПЛОМНА РОБОТА
БАКАЛАВРА

ДРБ. ІП - 55.00.00.000 ПЗ

Група ІП-19-1К

Ткачук Олексій

2022

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

Інститут інформаційних технологій

Кафедра інженерії програмного забезпечення

УДК 004.4

ДИПЛОМНА РОБОТА БАКАЛАВРА

Тема: Імплементація онтологічних моделей в модальних та діалогових формах

(назва згідно з наказом ректора)

Спеціальність – 121 "Інженерія програмного забезпечення"

ПОЯСНЮВАЛЬНА ЗАПИСКА

ДРБ.ІП - 55.00.00.000 ПЗ

(позначення)

Студент

ІП-19-1К /О.М.Ткачук/
(шифр групи) (підпис) (дата) (розшифровка підпису)

Керівник роботи

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

Консультанти:

з питань нормоконтролю

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

Допускається до захисту
Завідувач кафедри

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

2022
(рік)

Івано-Франківський національний технічний університет нафти і газу

Інститут, факультет інформаційних технологій

Кафедра інженерії програмного забезпечення

Ступінь вищої освіти бакалавр

Спеціальність 121 – Інженерія програмного забезпечення

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

ЗАВДАННЯ

НА ДИПЛОМНУ РОБОТУ БАКАЛАВРА СТУДЕНТОВІ

Ткачуку Олексію Михайловичу

(прізвище, ім’я, по-батькові)

1. Тема проекту (роботи) Імплементація онтологічних моделей в модальних та діалогових формах

керівник проекту (роботи) Шекета Василь Іванович, проф.

затверджені наказом закладу вищої освіти від **“** _ __ ” __ 2022 р. № _____

2. Строк подання студентом проекту (роботи) 17 червня 2022 р.

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

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

1. Огляд існуючих концепцій, рішень та сервісів в даній області
2. Побудова моделі або алгоритму власного рішення
3. Документування реалізації власного оригінального рішення вибраними засобами

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

1. Структура модифікованого DM на основі запропонованого способу (рис. 2.2, ст.28)
2. Процес управління діалогом у модифікованому DM (рис. 2.3, ст.37)
3. Діалоговий інтерфейс діалогової системи (рис. 2.8, ст.47)
4. Примірники Завдання одиниці та їх властивості (рис. 3.3, ст.51)

6. Консультанти розділів проекту (роботи)

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання **25 травня 2022 р.**

Керівник _____
(підпис)

Завдання прийняв до виконання _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів проекту	Примітка
1	Визначення та обґрунтування теми роботи	26.05.2022	виконано
2	Огляд існуючих концепцій, рішень та сервісів в даній області	29.05.2022	виконано
3	Побудова моделі або алгоритму власного рішення	30.05.2022	виконано
4	Документування реалізації власного оригінального рішення вибраними засобами	04.06.2022	виконано
5	Тестування ефективності розробленого програмного додатку	10.06.2022	виконано
6	Оформлення пояснювальної записки кваліфікаційної роботи	17.06.2022	виконано

Студент-дипломник _____
(підпис)

Керівник роботи _____
(підпис)

АННОТАЦІЯ

В дипломній роботі представлено підхід задання управління діалогом на основі онтології для досягнення незалежних від предметної системи діалогових систем. Створюється модель онтології завдання, що поєднує знання та методи з онтології та моделювання завдань. Знання про певну задачу в предметній області моделюються в онтології її задачі за допомогою онтології екземпляра моделі. Структура менеджера діалогів також модифікована, щоб містити компонент міркування онтології, який витягує конкретні знання задачі з онтології завдання. Таким чином завдання або знання предметної області відокремлюються від розробки діалогової системи.

КЛЮЧОВІ СЛОВА: ОНТОЛОГІЯ, МОДЕЛЬ, МОДЕЛЮВАННЯ, КОМПОНЕНТ, ПРЕДМЕТНА ОБЛАСТЬ, АЛГОРИТМ, ДІАЛОГОВА СИСТЕМА.

ANNOTATION

This paper presents a new approach to the task of ontology-based dialogue management to achieve systems independent of the subject dialog system. A task ontology model is created, which combines knowledge and a model of ontology and task methodology. Knowledge of a particular problem in the subject area is modeled in the ontology of its problem using the ontology of the model instance. The structure of the dialog manager has also been modified to obtain an ontology reasoning component that draws specific knowledge of tasks from the task ontology. Thus, the tasks or knowledge of the subject area are separated from the development of a dialog system.

KEY WORDS: ONTOLOGY, MODEL, SIMULATION, COMPONENT, SUBJECT FIELD, ALGORITHM, DIALOGUE SYSTEM.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ 7
ВСТУП 8
1 ОГЛЯД ІСНУЮЧИХ ПРОГРАМНИХ РІШЕНЬ ТА СЕРВІСІВ ДЛЯ ПОБУДОВИ ДІАЛОГОВИХ СИСТЕМ 12
1.1 Діалогічні системи 12
1.2 Підходи до управління діалогом 13
1.3 Архітектури для управління незалежним діалогом 15
1.4 Онтологічна модель та її представлення 18
1.5 Моделі завдання 22
2 АЛГОРИТМІЧНІ ТА МАТЕМАТИЧНІ РІШЕННЯ ДЛЯ ЗАДАЧ СТВОРЕННЯ ДІАЛОГОВИХ СИСТЕМ 26
2.1 Опис задачі побудови моделі 26
2.2 Пропонована модель онтології 28
2.3 Розробка менеджера незалежного діалогу домену 36
2.4 Розробка класів онтології завдання 40
2.5 Доменно-незалежна діалогова система 43
3 ПРОГРАМНА РЕАЛІЗАЦІЯ ОНТОЛОГІЧНИХ МОДЕЛЕЙ В МОДАЛЬНИХ ТА ДІАЛОГОВИХ ФОРМАХ 49
3.1 Діалогова система для послуги позичання книг 49
3.2 Опис процесу діалогу для послуги позичання книг 53
3.3 Послуга онлайн бронювання квитків на поїзд 55
ВИСНОВКИ 64
СПИСОК ПОСИЛАНЬ НА ДЖЕРЕЛА 66
БІБЛІОГРАФІЧНА ДОВІДКА

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

МО – модель онтології

БД – База даних

СУБД - Система управління базами даних

ДФ – діалогова форма

ВСТУП

Комп'ютерна система, яка здатна вести діалог з людьми, була темою дослідження штучного інтелекту протягом багатьох років. Однак лише за останні два десятиліття або близько того, з розвитком мовленнєвих технологій, обробки мови та моделювання діалогів, системи розмовного діалогу були розроблені та введені в комерційне використання. В останні роки дослідження діалогових систем було розширено на мультимодальні діалогові системи, які підтримують не тільки мовлення, але й текст, графіку, жести та інші режими введення користувача, а також виведення мультимедійної системи. Мультимодальна система діалогу зазвичай складається з компонентів, які обробляють вхідну інформацію та генерують вихідне повідомлення, а також центрального елемента управління діалогом, який координує всю систему та контролює процес діалогу.

З 1990-х років діалогові системи швидко розвиваються як у наукових дослідженнях, так і в комерційному застосуванні. Їх можна застосовувати в різних сферах, таких як бізнес, охорона здоров'я, освіта тощо. Через складність проектування розробка діалогової системи займає багато часу та витрат. Це дуже бажано для загальної системи діалогу, особливо управління діалогом, яке не залежить від конкретних доменів. Методи або архітектура для систем діалогу, що не залежать від предметної області, були запропоновані попередніми дослідженнями в літературі, проте кожен з них має свої обмеження, і жодне з них не набуло широкого поширення. У цій роботі представлено новий підхід, модель онтології задач для управління незалежним діалогом предметної області. Розроблено модель онтології абстрактної задачі і на її основі створено загальний менеджер діалогів. Знання про конкретне завдання моделюються в його онтології задачі та витягуються компонентом міркування онтології, розташованим в менеджері діалогів. Таким чином, менеджер діалогів не залежить від завдання чи домену. На основі запропонованої методики розроблено діалогову систему, яка експериментує з двома різними завданнями: взяти книгу в борг та онлайн-

бронювати квитки на потяг. Результати експерименту показують, що діалогову систему можна легко застосувати до завдань з різних доменів без будь-яких модифікацій. Ця робота має значення для майбутніх досліджень і розробок систем діалогу, незалежних від предметної області. Це також сприяє отриманню знань і повторному використанню системи діалогу і матиме вплив на застосування систем діалогу в більш широкому діапазоні областей.

Системи діалогу мають широкий спектр застосувань, такі як обслуговування клієнтів, продажі, технічна підтримка, персоналізовані послуги та навчання в різних галузях, включаючи охорону здоров'я, розваги, уряд, освіту та бізнес тощо . Вона має великий потенціал також у багатьох інших сферах. З іншого боку, для проектування та розробки діалогової системи потрібна співпраця як експертів із знаннями в області, так і комп’ютерних спеціалістів із систем управління діалогом. Розробка працездатної системи розмовного або мультимодального діалогу часто займає багато часу. Дуже бажано створити загальну систему діалогу, особливо модуль керування діалогом, який не залежить від завдань із різних доменів.

Більшість пов’язаних досліджень у літературі зосереджені на компонентах, відмінних від центрального управління діалогом діалогової системи. Лише деякі з них звернулися до управління незалежним діалогом домену. Перший основний підхід — це архітектура на основі агентів, в якій агенти розмовного діалогу використовуються для обробки різних доменів. Таким чином, додавання або застосування до нового домену означає створення нового агента розмовного діалогу для цього домену замість того, щоб перебудовувати всю систему. Другий підхід полягає в тому, щоб створити експертів з предметної області для контролю діалогу в кожній області. Центральний модуль системи вибирає домен і керує експертами домену. По-третє, для кожного конкретного завдання встановити менеджера сервісу. Усі ці методи відокремлюють те, що використовується для вирішення конкретних доменів або завдань, від інших складових модуля керування діалогами. Але створення агента діалогу, експерта домену чи

менеджера сервісу для певного завдання все ще вимагає залучення та співпраці як розробників діалогової системи, так і експертів для конкретних доменів. Найновішою спробою є структура управління діалогами на основі дерева завдань. У цьому контексті завдання або знання предметної області моделюються в дереві завдань, а потім на основі структури дерева завдань формується план діалогу. Це дозволяє фахівцям предметної області зосередитися на побудові дерева завдань, а системним фахівцям — на створенні діалогових систем. Однак різні завдання представлені різними деревами завдань, і для будь-якого нового завдання буде створено новий план діалогу. Тому розробка менеджера діалогів все ще залежить від завдання чи домену.

У цій роботі пропонується модель *онтології задач* для управління незалежним діалогом предметної області. Він має на меті застосувати онтологію та модель завдань у розробці діалогової системи, щоб здійснити незалежне від домену управління діалогом. Онтологія була прийнята в різних діалогових системах. Окрім обробки природної мови, нещодавні роботи щодо застосування онтології в діалогових системах в основному використовують онтологію як джерело знань, наприклад, систему діалогу, керовану онтологією, для навчання агентів , базу на онтологію для систем діалогу з консультуванням щодо здоров’я, діалогова система, в якій онтологія для кулінарної області може бути інтегрована в систему діалогу , і модель вимог на основі онтології для інтерактивного виявлення вимог . Однак кожен з них зосереджений на певній області, і жодна не підтримує незалежні від домену системи діалогу.

У запропонованій моделі поєднані також знання з моделювання завдань. Метою системи управління діалогом є досягнення завдання. Завдання, включаючи його діяльність та їх взаємозв'язки, можна представити моделлю завдання . Було запропоновано ряд представлень моделі завдань, включаючи ієрархічний аналіз завдань, вибір методів операторів цілей, аналіз групових завдань і тощо. Ці моделі в основному призначені для аналізу завдань, а не інтеграція в систему діалогу.

У запропонованій моделі знання конкретної задачі моделюється в незалежній від діалогового контролю онтології завдання. Створюється онтологія абстрактної задачі та розробляється загальний менеджер діалогів, у якому абстрактна онтологія використовується для напряму взаємодії системи з користувачами. Деякі прийоми і концепції моделювання завдань прийняті в онтології разом з тими новими класами і відносинами, створеними в моделі. Абстрактну онтологію задач можна створити в онтологіях завдань для завдань у різних областях. Завдяки компоненту аргументації, який з'єднує онтології завдань, загальний менеджер діалогів може виконати конкретне завдання шляхом взаємодії з користувачами без будь-яких змін.

Таким чином, проектування та розробка діалогового менеджменту та діалогових систем ізольованих від предметних знань для конкретного завдання; і систему діалогу можна застосувати безпосередньо до нових областей. Для перевірки та демонстрації ефективності запропонованого методу проведено два тематичні дослідження з різних завдань із різних областей.

1 ОГЛЯД ІСНУЮЧИХ ПРОГРАМНИХ РІШЕНЬ ТА СЕРВІСІВ ДЛЯ ПОБУДОВИ ДІАЛОГОВИХ СИСТЕМ

1.1 Діалогічні системи

Діалог – це розмова двох або більше людей або машин. [2] Система діалогу - це комп'ютерна система, яка спілкується з людьми-користувачами природним чином. Системи діалогу беруть участь у діалозі з людьми-користувачами для досягнення конкретних завдань, тому вони в основному орієнтовані на завдання.

Більшість досліджень було зосереджено на системах розмовного діалогу, які підтримують розмовну природну мову. В останні роки він був поширений на мультимодальні діалогові системи, які підтримують текстові, аудіо, відео та інші режими як для введення користувача, так і для системного виводу. [2] Згідно з Буй [2], типова мультимодальна діалогова система складається з таких модулів: Input, Fusion,

Менеджер діалогів, база знань, поділ та вихід. Серед цих компонентів Менеджер діалогів відіграє ключову роль у системі діалогу. Рисунок1.1 ілюструє архітектуру діалогової системи.

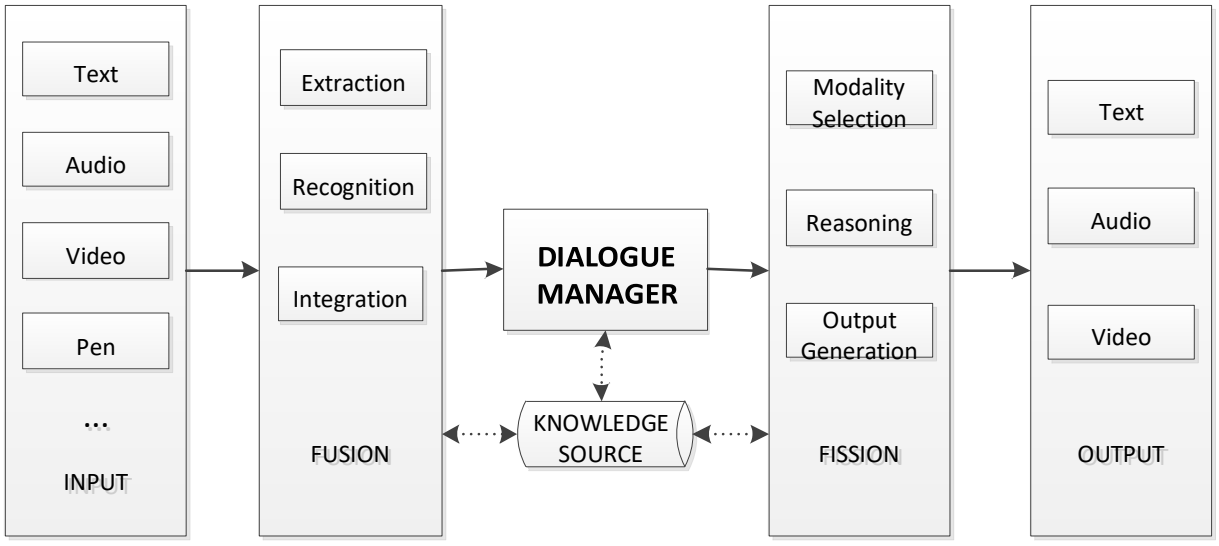


Рисунок 1.1 - Архітектура діалогової системи

У мультимодальній діалоговій системі модуль введення може обробляти різні режими введення, такі як текст, мова, перо, сенсорний екран, жести тощо.

Модуль Fusion витягує, розпізнає та інтегрує різні режими інформації з модуля введення, а потім надсилає до Dialogue Manager. Він також може консультуватися з Базою знань.

База знань використовується модулями Dialogue Manager, Fusion і Fission. Він керує багатьма типами знань, такими як історія діалогів, загальні знання діалогів, знання конкретної області та інформація про завдання та користувачів.

Модуль Fission відповідає за вибір модальності, міркування та координацію результатів. Він генерує вихідну інформацію в різних режимах.

Модуль виведення має представляти інформацію з модуля Fission у різних видах виводу.

Менеджер діалогів є центральним модулем управління як для систем розмовного діалогу, так і для мультимодальних діалогових систем. Він координується з іншими компонентами, такими як Fusion, Fission і Knowledge Base. Він також визначає намір користувача допомогти системі зрозуміти, чого користувач насправді хоче. Він контролює потік діалогів, вирішуючи, що реагувати на введення користувача.

1.2 Підходи до управління діалогом

За останні два десятиліття було запропоновано ряд підходів до управління діалогом. Кожен підхід має свої сильні та недоліки. За методами, які використовуються для контролю діалогу з користувачами, ці підходи можна розділити на такі чотири категорії: [1-2]

1) Підхід на основі кінцевих станів

Цей підхід є найпоширенішою та найпростішою стратегією управління діалогом. У системі діалогу на основі стану стани діалогу та системні висловлювання заздалегідь визначені, і діалог спрямовується системою через набір заздалегідь визначеної послідовності питань. Він підходить для реалізації

простої діалогової системи з добре структурованими завданнями, але не має гнучкості, природності та застосовності до інших областей.

2) Рамковий підхід

Підхід на основі фреймів є вдосконаленням управління діалогом на основі кінцевих станів.

Система працює як завдання заповнення слотів для збору заздалегідь визначеного набору інформації. Управління діалогами на основі фреймів допускає певний ступінь змішаної ініціативи та заповнення кількох слотів, що є більш гнучким, ефективним і природним у порівнянні з підходом на основі кінцевих станів. Він здатний обробляти помилки розпізнавання та непорозуміння під час діалогу з користувачами.

3) Імовірнісний і інформаційний підходи

У керуванні діалогом на основі стану інформації набір станів попередньо визначено, і стан діалогу змінюється відповідно до правил оновлення та стратегій діалогу. Як розширення підходів, заснованих на стані інформації, імовірнісне управління діалогом використовує імовірнісні методи, щоб спробувати покращити ефективність управління діалогом. [18] У цьому типі підходів діалог моделюється як процес прийняття рішень Маркова (MDP) [19-20] або частково спостережуваний процес прийняття рішень Маркова (POMDP) для вирішення невизначеності взаємодії людини з комп'ютером. Інші методи, такі як байєсівські мережі, мережі прийняття рішень, також вивчалися дослідниками.

4) Підходи на основі плану або *агента*

Підходи, засновані на плані або агенти, базуються на планових теоріях мовленнєвого акту та діалогу. Цей тип управління діалогом підтримує більш складні завдання діалогу, ніж три вищенаведені підходи. Відповідно, це передбачає більш високий рівень складності.

Перші два типи підходів є найбільш поширеними методами в існуючих діалогових системах. На їх основі побудовано ряд практичних систем. Інші

підходи, включаючи підходи на основі стану інформації та ймовірності, а також підходи на основі планів або агентів є відносно складними і потребують удосконалення для практичного використання. У цій дипломній роботі реалізована система діалогу на основі фрейму, щоб продемонструвати, як за пропонованим методом може бути здійснено незалежне від домену управління діалогом.

1.3 Архітектури для управління незалежним діалогом

Ряд діалогових систем було реалізовано з використанням різних стратегій керування діалогом, таких як автоматична банківська система Nuance [1], автоматична інформаційна система розкладу поїздів Philip , система Circuit-Fix-It Shop , система E -формувати оголошення про вживані автомобілі, систему Carnegie Mellon Communicator , систему MATCH для мультимодального доступу до міської допомоги , Immersive Virtual Worlds , Virtual Music Center], громадські телефонні будки Smartkom тощо. Ці системи були розгорнуті в різних сферах застосування, включаючи обслуговування клієнтів, технічну підтримку, персоналізовані послуги, планування, вирішення проблем і навчання в сферах охорони здоров’я, розваг, уряду, освіти та бізнесу тощо Очікується, що це очікується. що діалогові системи матимуть набагато ширше застосування та визнання в практичному використанні в майбутньому.

З іншого боку, для проектування та розробки працездатної діалогової системи через складність системи потрібно багато зусиль. Для цього часто потрібна співпраця як експертів із знаннями в області, так і комп’ютерних спеціалістів із систем керування діалогами, що є дорогим і трудомістким. Для мультимодальної діалогової системи процес побудови стає ще складнішим. Дуже бажано створити загальну систему діалогу, особливо модуль керування діалогами, який можна легко застосувати до завдань із різних доменів.

За останнє десятиліття низка дослідницьких проектів була спрямована на незалежні від предметної області діалогові системи. Багато з них зосереджені на модулях, пов’язаних з обробкою природної мови, таких як розпізнавання мовлення або семантичне представлення, а не на керування діалогами. Приклади цих проектів включають незалежну від домену платформу розмовного діалогу, яка використовує техніку визначення ключових фраз, засновану на комбінованій мовній моделі, двошарову архітектуру семантичної інтерпретації та аргументації домену для багатодоменної діалогової системи, надійну вибір домену проти висловлювань, що не відповідають граматиці, для багатодоменної системи розмовного діалогу та багатодоменної системи розмовного діалогу, яка здатна відповідати на запити користувачів у кількох доменах проти помилок розпізнавання мовлення тощо.

Була запропонована серія загальних архітектур, спрямованих на проектування систем діалогу, незалежних від домену. У проекті TRIPS Аллен та ін. запропонував загальну діалогову оболонку – загальну діалогову інфраструктуру для системи розмовного діалогу. У цій структурі основні компоненти системи розмовного діалогу, включаючи розпізнавання мовлення, обробку природної мови, генерацію відповідей та керування діалогами, будуються таким чином, що можна адаптувати до нових додатків до інших завдань або доменів, вказавши домен і завдання. операції. На основі цієї основи Allen et al. далі представили архітектуру, яка відокремлює лінгвістичне та дискурсивне знання від інформації про завдання та область. Це уточнює відповідальність окремих компонентів і покращує переносимість системи до нових завдань і доменів. Буй проілюстрував модульну загальну архітектуру мультимодальної діалогової системи. Він складається з модулів для обробки вхідних даних, генерації вихідних даних та керування діалогами. Кожен модуль має власну функцію і працює разом з іншими компонентами під координацією менеджера діалогів. Однак в архітектурі не йдеться про те, як створити незалежний від домену модуль керування діалогами. Це скоріше концептуальна

основа, ніж практична модель. У рамках цієї структури необхідно більше працювати, щоб створити систему діалогу, незалежну від домену.

Лише кілька досліджень були зосереджені на багатодоменному або незалежному від домену діалозі

управління. У літературі є чотири основні типи підходів. Перший – це агентна архітектура. Лін та ін. [5] запропонував багатоагентну архітектуру для багатодоменних систем розмовного діалогу. Ця архітектура застосовує різні агенти мовного діалогу (SDA) для обробки різних доменів і агент інтерфейсу користувача (UIA), щоб користувачі отримували доступ до правильного SDA через протокол перемикання домену. SDA розробляються незалежно, а потім співпрацюють один з одним, щоб допомогти досягти цілей користувача. Багатоагентна архітектура була прийнята рядом багатодоменних систем

Інший підхід — створити менеджер сервісів для кожного конкретного завдання. Запропонована архітектура діалогової системи складається з менеджера введення/виводу, менеджера діалогів і менеджера служб. Менеджер послуг обробляє всю інформацію, специфічну для домену.

Третій метод використовує експертів домену для підтримки управління діалогами в кількох доменах. Коматані та ін. використовують експертів домену для керування діалогами в кожному домені та центральний модуль для вибору належного домену та керування експертами домену в багатодоменній діалоговій системі. Лі та ін. представляє багатодоменну діалогову систему з використанням платформи моделювання діалогів на основі прикладів і техніки визначення домену.

Загальною проблемою вищезгаданих підходів є те, що всі вони потребують розробки нового компонента, такого як новий агент, новий менеджер служб або новий експерт домену під час додавання або адаптації до нового завдання чи домену в системі діалогу.

Останнім підходом до управління незалежним діалогом від предметної області є загальне моделювання діалогу на основі порядку денного або дерева

завдань. У цьому підході великі завдання розкладаються на менші та легше оброблені підзадачі. Структура керування діалогом RavenClaw є останньою, заснованою на цьому підході. Він містить дворівневу архітектуру на основі плану, в якій специфічні для домену аспекти логіки керування діалогом відокремлені від незалежних від домену. У цій структурі завдання або знання моделюються в дереві завдань; а потім на основі дерева завдань формується план діалогу для конкретного завдання. Різні завдання будуть мати різні плани діалогу, тому управління діалогом все ще залежить від конкретного завдання.

1.4 Онтологічна модель та її представлення

За Грубером, онтологія є формальною, явною специфікацією спільної концептуалізації. Концептуалізація — це абстрактна модель світу, який репрезентується. Іншими словами, онтологія надає спільний словник для представлення знань про задачу або область. Він визначає поняття, властивості кожного поняття та обмеження щодо їх використання в межах області.

Люди можуть мати різні цілі для розвитку онтології, наприклад:

1) Спільне розуміння структури інформації між людьми або програмним забезпеченням

2) Щоб дозволити повторне використання знань предметної області

3) Зробити явні припущення щодо домену

4) Відокремити предметні знання від операційних

5) Проаналізувати загальні знання в межах області

Основною метою онтології є підтримка обміну знаннями та їх повторного використання. Його можна використовувати в різних областях, таких як спілкування людей, сумісність системи та системна інженерія.

Онтологія зазвичай складається з таких компонентів: поняття в межах області, властивості кожного поняття, а також відносини або обмеження між поняттями або їх властивостями тощо. Згідно з Ной і МакГіннесом, існує ряд кроків, типових для побудови онтології, як наступне:

1) Визначення класів, що представляють поняття області, що описується

2) Упорядкування визначених класів у таксономії або зв'язку суперклас-підклас 3) Визначення властивостей класів і екземплярів та обмежень на їх значення

4) Заповнення екземплярів, їх властивостей та значень для кожної властивості.

Після того, як усі класи, властивості, екземпляри та їхні відносини визначені належним чином, розробляється онтологія, або онтологія завдання, або онтологія домену.

Мови онтології

Онтологія будується мовами онтологій. Мова онтологій зазвичай вводить поняття та їх властивості, відносини та додаткові обмеження. Мови онтологій можуть бути простими, заснованими на фреймах або логічними.Прості мови онтологій мають лише поняття та таксономії. Онтологічні мови на основі фреймів більш структурні порівняно з простими та містять поняття та властивості. Онтологічні мови, засновані на логіці, є більш складними та виразними. Він включає два поширених типи. Одним з них є мова онтології на основі логіки першого порядку, наприклад Common Logic (Common Logic Standard, http://common-logic.org/), Cycl (синтаксис Cycl, http://www.cyc.com/cycdoc/ ref/cycl-syntax.html) та KIF (формат обміну знаннями, http://www.ksl.stanford.edu/knowledge-sharing/kif/). Цей тип мови допускає загальні предикати. Іншим типом є мова онтологій на основі логіки опису, яка описує знання в термінах понять та їхніх відносин. Цей тип мов онтологій має можливість висновку. Іншими словами, можна міркувати про знання в мовах онтології на основі логіки опису, таких як висновок про неявне членство в класі, висновок про неявний підклас і еквівалентність класів, а також перевірити узгодженість за допомогою аргументу. Взагалі кажучи, чим багатша мова, тим неефективною вона підтримує міркування.

Мова веб-онтології (OWL), яку рекомендує W3C, є одним із

Опис Логічні мови. Це веб-стандарт для обробки інформації в Інтернеті. OWL написаний у XML і призначений для інтерпретації комп'ютерами. Існує три підмови OWL: OWL Lite, OWL DL і OWL Full. Серед них OWL Full є найбільш виразною мовою, але жоден існуючий аргумент не підтримує його висновок. OWL Lite і OWL

DL є вирішальними, але їх виразністю приноситься в жертву. [48] Таким чином, як доповнення до OWL DL та OWL Lite було запропоновано мову з можливістю вирішення, DL-безпечну мову правил семантичної мережі (SWRL).

Можна зробити висновок про неявні знання в онтології за допомогою логіки опису. Вже існує багато стабільних аргументів, доступних для міркування OWL DL, наприкладприйнято для побудови онтології, а Pellet — для онтологічних міркувань.

Застосування онтології в діалогічних системах

Онтологія використовувалася роками для інтерпретації та генерації мови в системах природних мов. Гаціус і Гонсалес представили систему діалогу, яка використовувала онтологію для покращення взаємодії між системою та користувачем, обробки втручань користувача та генерації системних повідомлень. Ця стаття присвячена обробці природної мови. Онтологія використовується в модулях обробки введення і генерації виводу, а не в модулі керування діалогами в системі діалогів. Зонтаг і Романеллі представили онтологічну синтаксичну структуру результатів мультимодальних відповідей на запитання для систем запитань і відповідей. У цій роботі онтологія служить комунікаційним інтерфейсом між компонентами обробки природної мови. Дослідження Мілварда і Беверіджа показало, що онтологічну інформацію можна використовувати для роз’яснення діалогу. У цьому дослідженні онтологія використовується для дослідження відмінностей між монологом і діалогом, а не для розробки діалогової системи.

У системах діалогу новітні роботи використовують онтологію для охоплення предметних знань і забезпечення джерелом знань для системи. Фліхт-

Еріксон і Йонссон запропонували архітектуру систем діалогу, в якій онтології застосовуються для різних завдань, включаючи аналіз запитань, діалогову взаємодію та вилучення інформації. В архітектурі онтології надають спільні джерела знань для співпраці двох основних частин діалогових систем: компонента взаємодії та компонента обробки інформації.

Ван Ойєн та ін. представили систему діалогу, орієнтовану на онтологію, для навчання агентів. У системі онтології використовуються для отримання знань про предметну область та знання ролі агента. Вони забезпечують спільний набір даних для концепцій та їхніх атрибутів, якими обмінюються агенти. Використання онтології забезпечує тісний зв'язок між агентами та моделюванням і дозволяє повторно використовувати можливості агента.

Бікмор та ін. запропонували базу на онтології для систем діалогу з консультуванням щодо здоров'я. Фреймворк реалізує онтологію концепцій зміни поведінки та модель завдання для втручання системи та користувачів. Онтологія охоплює знання щодо консультування щодо зміни поведінки у здоров'ї. У цьому підході онтологія і модель завдання роз'єднані, і онтологія служить лише онтологією предметної області.

Пардал і Мамед мали намір розширити систему діалогу на нові області знань за допомогою онтологій. Вони створили систему діалогу Cookcoach на базі OLYMPUS, яка включає в себе управління діалогами RAVENCLAW . Для кулінарної області була розроблена онтологія OntoChef. OntoChef містить модулі спеціально для сфери приготування їжі, включаючи дії, їжу, рецепти, посуд, одиниці та міру. Cookcoach підтримує різні рецепти; однак модель підтримує лише домен приготування.

Чжан запропонував модель вимог на основі онтології для інтерактивного виявлення вимог. У цій моделі концепції, включаючи функцію, якість і програмну мету, а також відносини між цими поняттями і правилами встановлюються, зокрема, для виявлення вимог. Навряд чи можна застосувати

цю модель до завдань в інших областях, наприклад, послуги з бронювання або бронювання квитків на потяг тощо.

Щоб зробити менеджер діалогів на основі POMDP здатним до практичних систем діалогу, Янг та ін. запропонував модель стану прихованої інформації (HIS). Правила онтології прийняті в моделі HIS як модель завдання діалогового сценарію і представляють групи подібних станів переконань у вигляді розділів. Модель використовує деревовидну структуру для представлення цілі користувача. Деревовидна структура залежить від конкретного завдання. Так само діють правила онтології, засновані на цьому.

Heinroth et al. створили адаптивний менеджер розмовних діалогів OwlSpeaker, який підтримує багатозадачність у системі діалогу. У OwlSpeaker, онтології розмовного діалогу (SDO) розроблені для моделювання бази знань для різних доменів. Кожен SDO містить підкласи мовлення та стан та їх окремі елементи розташовані у вигляді дерева. OwlSpeak дозволяє перемикатися між різними взаємопов'язаними завданнями. Однак метою статті є підтримка багатозадачності в системі діалогу, а не створення доменного незалежної діалогової системи.

Жодна із вищезгаданих робіт не підтримує незалежне від домену управління діалогом. На відміну від попередніх підходів, ця дипломна робота розробляє модель онтології задач, яка може відокремити знання предметної області від управління діалогом у системі діалогу. Таким чином, управління діалогом не залежить від знання предметної області, а система діалогу, заснована на моделі, не залежить від предметної області.

1.5 Моделі завдання

Моделі завдань описують можливі види діяльності та їх взаємозв'язки в завданнях. Вони створюються шляхом аналізу завдань, який є процесом збору даних про те, як люди виконують завдання, і отримання глибокого розуміння цих даних.

Моделі завдань були використані для підтримки проектування інтерфейсів користувача та інтерактивних систем. Основні цілі моделей завдань включають:

1) Щоб краще зрозуміти домен програми

2) Записати результат міждисциплінарного обговорення

3) Для підтримки ефективного дизайну та оцінки зручності використання

4) Надавати документацію та підтримку користувачів. Аналіз і моделювання завдань вперше було введено в психологію. Ієрархічний аналіз завдань (HTA) Аннет і Дункан був першим важливим методом, запропонованим для аналізу та моделювання завдань. З тих пір моделі завдань стали більш формальними, і було розроблено ряд уявлень, включаючи HTA, Goals Operators Methods Selectors (GOMS) , User Action Notation (UAN) , Groupware Task Analysis (GTA).) , онтологія світу завдань та ConcurTaskTrees (CTT) тощо.

Серед методів представлення для моделей завдань HTA є основою для багатьох інших методів. У HTA завдання описуються в термінах операцій і планів. Операції - це дії для досягнення мети, а плани - це їх умови. Завдання ієрархічно розкладаються на серію підзадач, таким чином моделюється ієрархія завдань. У HTA ієрархія цілей не представлена явно.

GOMS представляє завдання, а також їх цілі. З тих пір GOMS-аналіз був однією з найбільш відомих методик аналізу завдань, і було розроблено багато варіантів. У GOMS діяльність у завданні описується в термінах цілей, операторів, методів і правил вибору. Цілі – це стани, які користувачі мають переслідувати; а методи – це можливі шляхи досягнення мети. Оператори описують етапи методу. Правила відбору є критеріями для методу. Однак GOMS переважно текстовий. Він не враховує помилки користувача та можливість перерв. GOMS представляє лише послідовні завдання.

UAN — це нотація, орієнтована на користувача та завдання, що описує поведінку користувача для дизайну інтерфейсу користувача. UAN визначив ряд тимчасових зв’язків між завданнями, зокрема: послідовність, очікування,

повторне роз’єднання, незалежність від порядку, переривання, одностороння або взаємна взаємозамінність і паралельність. Ці відносини можуть бути чітко і точно представлені в описах завдань UAN. В UAN завдання, які неможливо розкласти, визначаються як базові завдання. Основні завдання визначаються діями користувача, системним зворотним зв'язком і станом системи. Основним недоліком UAN є те, що він використовує текстовий синтаксис і відсутність підтримки інструментів.

GTA зосереджується на групових користувачів або організаціях та їх діяльності, а не на окремих користувачах. У GTA модель завдання містить три різні аспекти світу завдань, включаючи агентів, роботу та ситуацію. *Агенти* – це люди або системи, які виконують різні ролі. Робота виконується за завданнями; і завдання можуть бути визначені на різних рівнях складності, наприклад, завдання на одиницю та основне завдання. Завдання підрозділу та основні завдання можна далі розкласти на дії системи або дії користувача. Ситуація світу завдань включає середовище для виконання певного завдання та опис об'єктів середовища. Опис об'єкта включає його структуру та атрибути.

Велі [2] розробив онтологію світу завдань, яка описує концепції та їх взаємозв'язки на основі GTA. Онтологія включає поняття завдання, мети, ролі, агента та події.

Завдання може бути одиничним завданням або основним завданням. Завдання на одиницю повинно складатися з одного або кількох основних завдань. Ці поняття мають різні взаємозв’язки між собою, такі як використання, тригери, відтворення, perform_by, has, subtask/submeal, subrole, influence, answer, used_by. Він був використаний для дизайну інтерфейсу користувача. Обмеженням GTA разом із онтологією світу завдань та іншими варіантами GTA є те, що вони підтримують лише послідовні завдання. Часовий зв’язок між завданнями не може бути точно визначений у цьому типі моделі завдань.

ConcurTaskTrees (CTT) вирішує проблему специфікації відношення часу в GTA. CTT — це графічне позначення для опису завдань та їх взаємозв’язків. Він

має ієрархічну структуру та графічний синтаксис. CTT має багатий набір операторів для визначення тимчасових відносин між завданнями, включаючи включення, вимкнення, вибір, паралельність, переривання, необов'язковість та ітерацію тощо. Інструмент, пов'язаний з CTT, ConcurTaskTrees Environment (CTTE) надає методи для опису окремих користувацькі, а також кооперативні моделі завдань. Таким чином, CTT також є вдосконаленням GOMS.

Усі вищезгадані уявлення використовуються в основному як інструменти для аналізу завдань. Вони, як правило, прийняті для підтримки процесу проектування системи і не містять достатньої та необхідної інформації для автоматичного створення інтерактивних систем. У цій роботі пропонується модель онтології задач для інтеграції в систему управління діалогом. Онтологія завдання використовується для автоматичного керування процесом діалогу.

2 АЛГОРИТМІЧНІ ТА МАТЕМАТИЧНІ РІШЕННЯ ДЛЯ ЗАДАЧ СТВОРЕННЯ ДІАЛОГОВИХ СИСТЕМ

2.1 Опис задачі побудови моделі

У цьому розділі представлена модель задач на основі онтології для систем управління діалогами, які не залежать від домену. Як було зазначено, мета системи діалогу, незалежної від предметної області, полягає в тому, щоб відокремити розвиток діалогових систем від знань, специфічних для завдання в певній області. Для досягнення мети в цій роботі пропонується новий підхід – модель онтології задач для управління незалежним діалогом у предметній області. У цьому підході розробляється модель онтології абстрактної задачі. На основі цієї моделі модифікується менеджер діалогів. Знання про конкретну задачу моделюються в онтології її задачі за допомогою онтології екземпляра абстрактної моделі. Інформацію в онтології завдання можна отримати за допомогою компонента обґрунтування онтології, розташованого в менеджері діалогів.

Таким чином конкретне завдання або знання предметної області ізольовано від управління діалогом у системі діалогу. Модифікований менеджер діалогів не залежить від завдань; і діалогова система, заснована на цьому методі, не залежить від домену і може бути легко застосована до нових завдань або нових доменів.

Запропонована модель онтології завдання відрізняється від попередніх підходів до моделювання завдань. По-перше, вона поєднує знання моделювання задач та онтології. У цій моделі онтологія використовується для представлення понять та їхніх відносин у моделі завдання; тоді як графічні або текстові уявлення зазвичай використовуються в попередніх підходах моделювання завдань. По-друге, запропоновану онтологію завдання можна отримати безпосередньо за допомогою діалогових систем. Використовується для

автоматичного керування взаємодією з користувачем у системі діалогу; тоді як попередні методи моделювання завдань в основному використовуються як інструменти для аналізу завдань.

Ця модель розширена з онтології світу завдань. У цій моделі прийняті поняття агента, об'єкта і завдання та їх зв'язки в онтології світу завдань. Однак онтологія світу завдань не деталізує декомпозицію завдання та можливі зв’язки між його підзадачами.

Щоб подолати цю проблему, ця модель використовує метод ієрархічного аналізу завдань для декомпозиції завдання на набір підзадач. Він приймає концепції основного завдання, завдання одиниці та їх специфікації дій користувача та зворотного зв'язку системи з UAN та CTT. Деякі тимчасові зв'язки між завданнями, представлені в UAN і CTT , такі як послідовність і вибір, адаптуються та моделюються у відносини в моделі онтології завдання. Ця модель також розробляє концепції, зв’язки та властивості, які не можуть бути представлені в попередніх моделях завдань, але підходять для онтології, наприклад, непересікаючі зв’язки та властивості даних.

Як модель онтології завдань буде використана в менеджері діалогів, щоб допомогти досягти незалежності домену ілюструє структуру універсального менеджера діалогів на основі запропонованої моделі онтології задач. Він включає в себе загальний компонент управління діалогами (DM) і компонент міркування онтологій.

Компонент міркування онтології викликається загальним DM. Він отримує знання про завдання або предметну область з онтології завдання та причини для збору інформації з діалогу з користувачами. Знання про конкретне завдання моделюються в онтології завдання. Таким чином, проектування та розробка загальних DM та компонента аргументації онтології не залежить від конкретних завдань або доменів.

2.2 Пропонована модель онтології

На рисунку 2.1 представлена запропонована модель онтології завдання побудови діалогових систем.

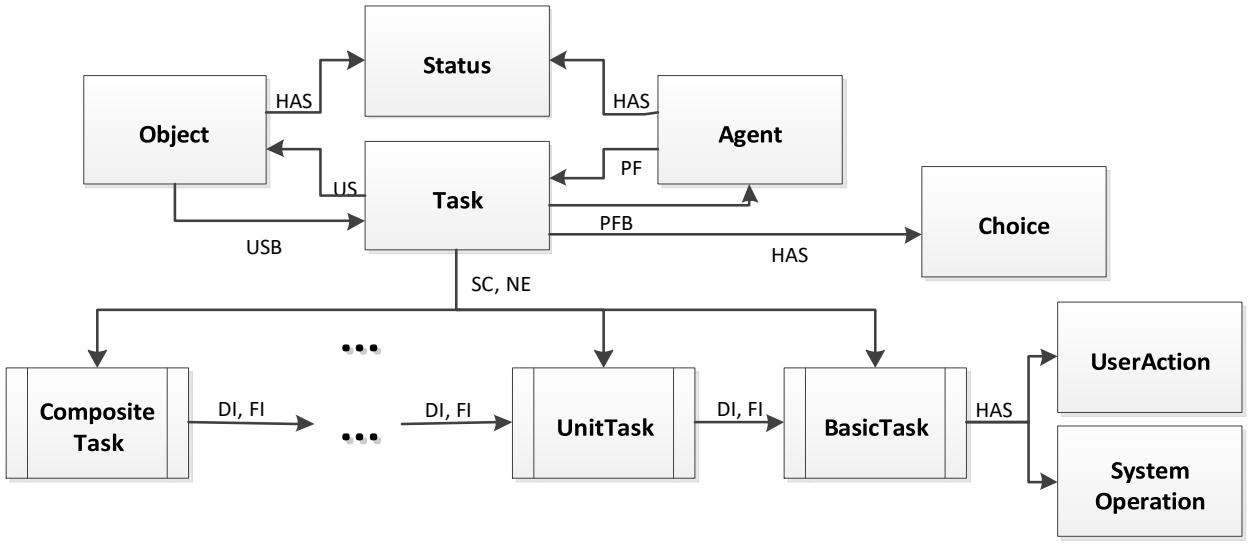


Рисунок 2.1. - Запропонована модель онтології

Поняття в моделі онтології цієї задачі з позначеннями на діаграмі класів і підкласів. Об'єкт і агент представляють контекст, в якому виконується завдання. Об'єкт використовується завданням; і агент виконує завдання. І об’єкт, і агент мають статус. Завдання — це дії, які допомагають користувачам досягти своєї мети. Існує три типи завдань: CompositeTask, UnitTask і BasicTask. BasicTask має UserAction і SystemOperation. UserAction моделює дії, які користувачі можуть виконати під час виконання завдання; тоді як SystemOperation — це операція, яку мала б виконувати система. Деякі завдання можуть мати вибір для користувачів, і вибір, який робить користувач, буде спрямовувати систему рухатися вперед. Усі концепції, створені в моделі онтології завдання, описуються таким чином:

1) Завдання: це дії для досягнення певної мети. Тут визначено три типи завдань модель: CompositeTask, UnitTask і BasicTask, що представляють різні рівень декомпозиції в ієрархії завдань.

a) Складене завдання: це завдання, яке користувачі хочуть досягти своєї мети. Складене завдання: знаходиться на верхньому рівні розкладання завдань. Далі він розкладається на одну або кілька Завдання одиниці.

b) Завдання одиниці: знаходиться в середині ієрархії розкладання завдань. Може бути один або кілька рівнів Завдання одиниці. Завдання одиниці складає Складене завдання:. Він також розкладається на одну або кілька Основні завдання.

в) Основні завдання.: знаходиться в нижній частині ієрархії завдань. Основні завдання. складає Завдання одиниці. Він є атомарним і включає Дія користувача і Робота системи.

2) Агент: можуть бути люди або машини. Агент виконує завдання.

3) Об'єкт: відноситься до фізичних або нефізичних об'єктів, які використовуються в Task.

4) Статус: і Об'єкт, і Агент мають статус. Значення Status може бути доступним або недоступним.

5) Вибір: це параметри, які користувачі можуть мати у завданні. Те, що користувачі виберуть, керуватиме наступним кроком системи.

6) Дія користувача: відноситься до дій, які користувачі повинні виконати для виконання завдання. Це стосується лише Основні завдання.

7) Робота системи: це дія, що виконується системою в завданні. Подібно до Дія користувача:, він має значення лише для Основні завдання..

Відносини

Завдання зазвичай представлено у вигляді деревоподібної ієрархії завдань. У дереві завдань найважливішими зв’язками між завданнями та їх підзадачами є:

1) взаємозв’язок надзавдання та підзадачі: завдання може розкластися на кілька підзадач. Це моделюється як DecomposeInto;

2) тимчасові відносини: порядок виконання набору завдань або підзадач. Це представлено як Перший і Наступний;

3) альтернативне співвідношення: для деяких завдань лише одне з них буде виконано для виконання суперзавдання. Цей тип відносин представлений за допомогою has Choice.

Крім has Choice, в моделі онтології задачі побудовано два інших зв'язку «has». Одним з них є hasUserAction. Він використовується для зберігання інформації про те, які дії користувачі повинні виконати, щоб виконати завдання. Це стосується лише BasicTask. Іншим є hasSystemOperation. Це схоже на відношення hasUserAction, яке застосовується лише до BasicTask. HasSystemOperation використовується для зберігання інформації про операції, які система буде виконувати завдання.

Запропонована модель також враховує контекст завдання. Поняття «Об'єкт» і «Агент» визначаються для представлення об'єктів чи інструментів, які потрібні задачі, і людей чи машин, які виконують завдання відповідно. Їх відносини з Task представлені за допомогою Uses, UsedBy, Performs і PerformedBy. Відношення hasStatus розроблено для представлення статусу об'єкта абоАгент.

SC: Підклас (x, y), x { Task }, y { Composite Task , Unit Task, BasicTask }

DI: розкласти на (x, y), x {складова задача}, y {задача на одиницю}

DI: розкласти на (x, y), x {Одиниця завдання}, y { Основне завдання}

FI: First(x, y), x {Composite Task}, y {Unit Task}

FI: First(x, y), x {Unit Task}, y {Basic Task}

NE: Next(x, y), x {Task}, y {Task}

USA: використовує(x, y), x {Task}, y {Object}

USB: використовується By(x, y), x {Object}, y {Task}

PF: виконує(x, y), x {Агент}, y {Task}

PFB: виконується (x, y), x {Task}, y {Agent}

HAS: має вибір (x, y), x {Task}, y {Choice}

HAS: має дію користувача(x, y), x {Основне завдання}, y {Дію користувача}

HAS: має операційну систему (x, y), x {Основне завдання}, y {Операцію системи}

HAS: має статус(x, y), x {об'єкт, агент}, y {Status}

У OWL концепції будуються як класи, а їхні відносини представлені властивостями. Існує два основних типи властивостей: властивості об’єкта і властивості даних. У запропонованій моделі онтології задачі відносини DI, FI, NE, US, USB, PF і PFB будуються як властивості об'єкта, а відносини HAS розвиваються як властивості даних, у тому числі має вибір, має статус, має дію користувача, має Системну роботу.

1) SC: це відношення суперклас-підклас. Завдання — суперклас. Його підкласи, складена задача, одиничне завдання та базова задача представляють різні рівні декомпозиції завдання.

2) DI: DecomposeInto. Складене завдання знаходиться на вершині ієрархії завдань. Його можна розкласти на одну або кілька одиничних завдань. Задача підрозділу знаходиться в середині ієрархії завдань, яку можна далі розкласти на одне або кілька основних завдань.

3) FI: це перше завдання на одиницю, що міститься в декомпозиції складної задачі, або перша основна задача в декомпозиції одиничної задачі. Він використовується для визначення порядку виконання в складеній задачі або одиничному завдання.

4) NE: Далі визначає, яке наступне завдання чи дію буде виконувати система після завершення поточного завдання. Далі разом із First представляють послідовні відносини між завданнями та підзадачами.

5) USA: завдання використовує об'єкт. Виконання певного завдання може залежати від стану відповідних об'єктів.

6) USB: об'єкт використовується завданням.

7) PF: Агент виконує завдання. Статус агента може вплинути на виконання певного завдання.

8) PFB: завдання виконує агент.

9) HAS: існує чотири типи відносин HAS, у тому числі має статус, має вибір, має дію користувача та має операційну систему. Вони будуються як властивості даних у запропонованій моделі. Властивість data дозволяє онтології завдання мати значення, що представляють статус агента або об’єкта, параметри, які мають користувачі, дії, які користувачі повинні виконувати, або операції на системній стороні завдання.

а) має статус: представляє статус агента або об'єкта. Значення може бути Available або Unavailable.

б) має вибір: завдання може мати вибір. Він представляє параметри, які користувачі можуть мати у завданні, і те, що користувачі вибирають, спрямовує наступні дії, які виконає система. Значенням може бути список параметрів, які представлені користувачам.

с) має дію користувача: представляє дії, які користувачі мають виконати у завданні. Це стосується лише BasicTask. Значення — це рядок пояснення користувачам. Залежно від розміру інформації значенням може бути файл.

d) Операція системи: представляє операції, які система має в завданні. Це має сенс лише в базовому завданні. Значення — це рядок пояснення користувачам.

Залежно від розміру інформації значенням може бути файл.

ПРАВИЛА

Для запропонованої моделі онтології завдання визначено наступні правила SWRL:

1. DecomposeInto(x, x) недійсний

Відношення DecomposeInto є нерефлексивним.

2. DecomposeInto(x, y) → DecomposeInto(y, x) є недійсним

Відношення DecomposeInto є асиметричним.

3. First(x, x) недійсний

Відносини First є нерефлексивними.

2. First(x, y) First (y, x) недійсний

Відношення First є асиметричним.

5. Next(x, y) Next (y, x) недійсний

Відношення Next є асиметричним.

6. Наступний (x, x) недійсний *The relationship Next is irreflexive.*

1. Складене завдання (x) Одиничне завдання (x) недійсне

Підкласи завдань: складена задача і одиничне завдання не перетинаються один з одним.

2. Складене завдання (x) Основне завдання (x) недійсне

Підкласи задач: складена задача і базова задача не перетинаються один з одним.

3. Одиничне завдання (x) Основне завдання (x) недійсне

Підкласи завдань: Unit Task і basicTask не перетинаються один з одним.

4. Завдання (x) Агент (x) недійсний

Класи Task і Agent не перетинаються один з одним.

5. Завдання (x) Об'єкт (x) недійсний

Класи Task і Object не перетинаються один з одним.

6. Агент (x) Об'єкт (x) недійсний

Класи Agent і Object не перетинаються один з одним.

7. Використовує (x, y) → використовується (y, x)

Uses і used By є зворотними властивостями об'єкта.

8. Виконує (x, y) →виконує (y, x)

Виконує та виконує By є зворотними властивостями об’єкта.

9. Використання (x, y) → Використання (y, x) недійсне

Відносини «Використання» асиметричні.

10. Використання (x, x) недійсне

Відношення Використання є нерефлексивним.

11. Виконує (x, y) →Виконує (y, x) недійсне Відношення Виконує асиметричне.

12. Виконує (x, x) недійсний

Відносини Performs є нерефлексивними.

Після того, як онтологія завдання побудована, ці правила можуть бути використані обґрунтувальником онтологій для виведення інформації, яка явно не описана в онтології. Система може допомогти провести перевірку узгодженості, а також отримати неявні знання. Якщо в онтології вбудовано недопустимий зв'язок, процес міркування допоможе виявити та виправити проблему.

Крім правил, в запропонованій моделі онтології завдання також визначаються інші обмеження, включаючи екзистенційні, універсальні та потужності. Вони також можуть бути використані резонатором для онтологічного висновку та перевірки. В онтології завдання визначено такі обмеження:

1) Одиничне завдання еквівалентно:

Завдання і ((Розкласти на деяку основну задачу) і (Розкласти на хв. 1 основне завдання) і

(Перша хвилина 1 основне завдання))

Unit Task — це завдання, яке можна розкласти принаймні на одну основну задачу і має принаймні одну основну задачу як властивість першого об'єкта.

2) CompositeTask еквівалентний:

Завдання і ((Розкласти на деяку одиничну задачу) і (Розкласти на хв. 1 одиничне завдання) і (Перша хвилина 1 одиничне завдання))

Складена задача – це завдання, яке можна розкласти принаймні на одну одиничну задачу; і має принаймні одну UnitTask як властивість першого об'єкта.

3) Доменом для використання є Task, а діапазоном є Object

Завдання використовує тільки Object.

4) Домен, який використовується By — це Object, а діапазон — Task

Об'єкт використовується тільки Task.

5) Домен для Performs — агент, а діапазон — Task

Агент виконує тільки завдання.

6) Домен для виконуваного By — це завдання, а діапазон — агент

Завдання виконує тільки Агент.

7) має вибір мінімум 2 літери

Властивість даних має вибір, оскільки завдання має принаймні 2 потужності як значення.

8) Домен для властивостей даних має дію користувача та базове завдання

Лише базове завдання має властивості даних, які мають дію користувача та дію системи.

9) Домен для властивості даних має Вибір – Задача

Лише завдання має властивість даних має вибір.

10) Домен для властивості даних має статус Агент або тільки об'єкт Агент або Об'єкт має властивість даних має статус.

Інстанціація онтології

У цій дисертації екземпляр онтології є процесом моделювання задач для конкретних завдань з використанням запропонованої моделі онтології завдання. Конкретне завдання зазвичай представляється у вигляді деревоподібної ієрархії завдань. Спочатку завдання та його підзадачі ідентифікуються та відображаються як екземпляри або члени в підкласи

CompositeTask, UnitTask і BasicTask відповідно до рівня їх декомпозиції в дереві завдань. Складені завдання – це найскладніші завдання, які знаходяться на верхньому рівні дерева завдань. Він розкладається на набір одиничних завдань. Для складних завдань можна мати більше одного рівня одиничних завдань. Зв'язки між різними рівнями одиничних завдань однакові з зв'язками між складеним і одиничним завданням. Завдання підрозділу далі розкладаються на основні завдання, які є листками дерева завдань. Потім зв’язки між цими завданнями та підзадачами визначаються та відображаються у відповідні властивості в онтології завдання. У запропонованій моделі онтології задачі прийнято два основних типи властивостей: один — властивість об'єкта, яка зв'язує індивіда з індивідом; а інший — властивість даних, яка пов’язує індивіда із значенням даних. На цьому кроці зв’язки між екземплярами та належним значенням даних мають бути ідентифіковані та додані в онтологію. DecomposeInto, Next і First зазвичай

використовуються для порядку виконання завдання. Відношення hasChoice використовується для представлення різних варіантів, які користувачі можуть вибрати, і вибір користувачів визначатиме наступну дію, яку виконає система. Таким чином, різні типи потоку завдань, що включають послідовні завдання або альтернативні завдання, можуть бути представлені в моделі онтології завдання.

Для деяких типів завдань також слід враховувати контекст. Клас Object використовується для моделювання інструментів для завдання або його підзадач. Наявність інструментів відображається як статус в онтології завдання. Агент використовується для людей, машин або програмних додатків, які виконують завдання або його підзадачі. Їх статус також представлений як hasStatus.

Після того, як вінтологія для конкретного завдання буде створена, ми можемо запустити обґрунтування онтологій, щоб класифікувати онтологію, щоб отримати виведені класи. Тоді онтологія матиме виведену ієрархію, а також вставлену ієрархію. Процес міркування також перевірить узгодженість і вкаже на будь-яку невідповідність між особами або їхніми властивостями. Таким чином, можна визначити будь-які проблеми, які виникають під час процесу. Після того, як усі помилки виправлені, встановлюється дійсна онтологія для конкретного завдання. Ця онтологія завдання буде використовуватися незалежним від домену менеджером діалогів, описаним у наступному розділі, для керування процесом діалогу між системою та користувачами.

2.3 Розробка менеджера незалежного діалогу домену

Відповідно, попередні підходи до управління діалогом, що не залежать від домену, розділяли менеджер діалогів на загальний компонент керування діалогами та компонент керування діалогом, специфічний для домену.

Компонент онтологічного обґрунтування додається до DM і замінює компонент управління діалогом, специфічний для домену, у попередніх підходах. Він здатний отримувати специфічні для предметної області знання з онтології завдання ілюструє структуру модифікованого DM. Він включає в себе загальне

керування діалогами та компонент обґрунтування онтологій. Компонент обґрунтування онтології викликається загальним DM. На рисунку лише онтологія завдання є специфічною для певного завдання; а всі інші компоненти не залежать від конкретних завдань або доменів.

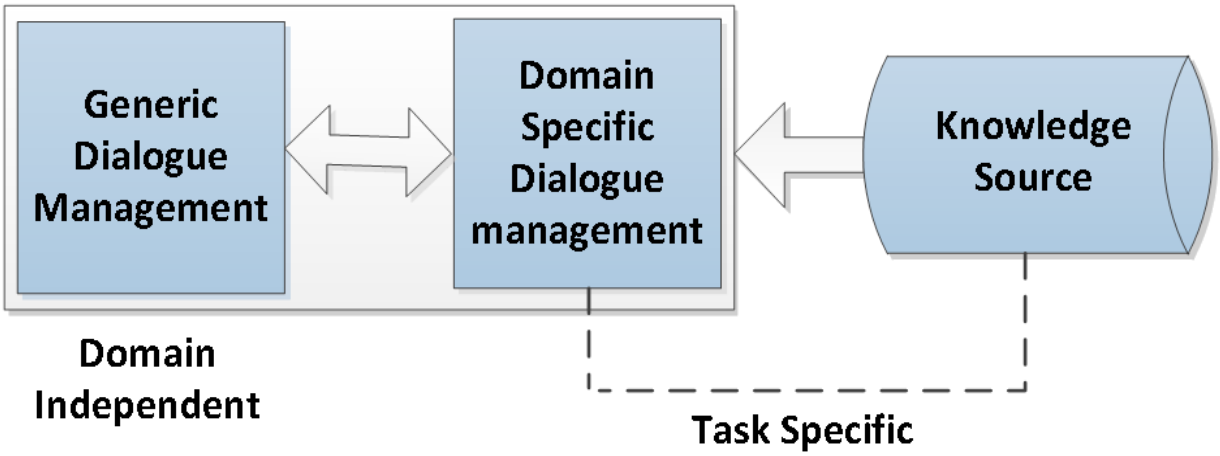


Рисунок 2.1 - Структура DM на основі попереднього доменного незалежного діалогу підходи до управління

На рисунку 2.2 подана структура модифікованого DM на основі запропонованого способу.

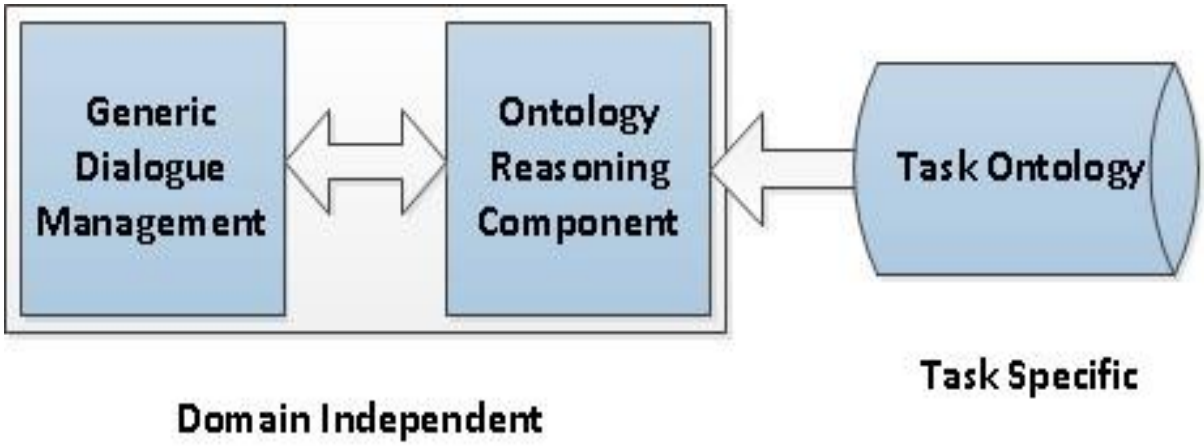


Рисунок 2.2 - Структура модифікованого DM на основі запропонованого способу

Рисунок 2.3 ілюструє процес керування діалогами діалогової системи на основі модифікованого менеджера діалогів. На початку діалогу користувач може вибрати зі списку завдань, доступних у системі діалогу. Після того, як користувач вибере конкретне завдання, система діалогів знайде корінь завдання, складене завдання. Потім система почне переглядати дерево завдань, поки не буде виконано все завдання.

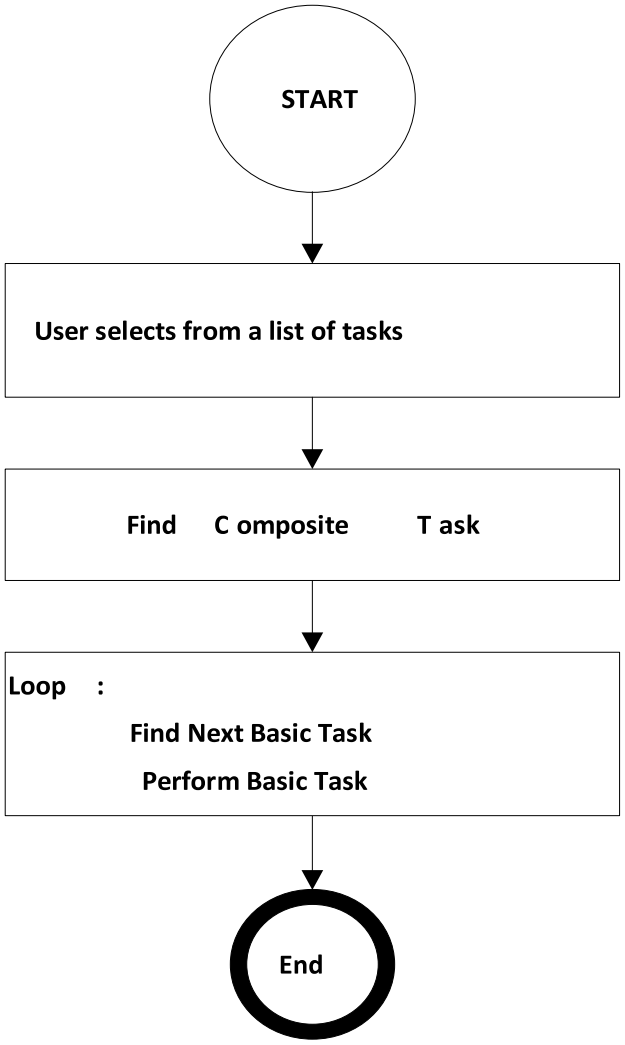


Рисунок 2.3 - Процес управління діалогом у модифікованому DM

На рис. 2.4 наведено псевдокод циклу пошуку наступного основного завдання та виконання основного завдання, наведеного на рис. 2.3.

```
findBasicTask(currentTask){
        if (currentTask is not a basic task) {
                if (currentTask has > 1 First properties) {
                        find the hasChoice property;
                        ask user to select one;
                        find the user selected task;
                        set current task to the user selected task;
                } else {
                        find First task;
                        set current task to the First task;
                }
        }
        if (current task is a basic task) {
                perform the task;
                if (current task has Next property) {
                        if (current task has >1 Next properties) {
                                find the hasChoice property;
                                ask user to select one;
                                find the user selected task;
                                set current task to the user selected task;
                        } else {
                                find the next task;
                                set current task to the next task;
                        }
                        findBasicTask(currentTask);
                } else {
                        entire task completed;
                }
        } else {
                findBasicTask(currentTask);
        }
}
```

Рисунок 2.4 - Псевдокод процесу управління діалогом

Дерево завдань будується таким чином, що підзадача буде відвідана не більше одного разу. Якщо підзавдання є базовим завданням, то його властивість об’єкта Next буде відвідано один раз, і кожне з двох його властивостей даних – hasUseraction і hasSystemOperation – буде розгорнуто один раз. Відвідати властивість об’єкта означає знайти підзавдання, яке вказано як значення властивості. Розширити властивість даних означає відкрити значення та доставити користувачеві в режимі виведення. Якщо підзавдання є одиничним завданням або складеним завданням, то властивість об’єкта First буде відвідано один раз. Тоді ми можемо оцінити процес управління діалогом наступним чином:

Нехай : номер одиниці завдання

: номер основного завдання

: витрати на відвідування майна об'єкта

: витрати на розширення властивості даних

Тоді загальний обсяг виконаної роботи:

Цикл перед: постійна кількість

Після циклу: постійна кількість Кожен раз проходьте цикл:

для основного завдання: для одиничного завдання:

У гіршому випадку всі підзадачі будуть відвідані один раз у циклі, тому:

Загальна кількість часу: (+))

Часова складність: O ( )

Дозволяє

Тоді Часова складність: O ( )

Таким чином, часова складність дорівнює O ( ), у якому є максимум із загальної кількості основних завдань або одиничних завдань у рамках завдання.

Реалізація системи включає створення онтології абстрактної задачі та розробку діалогової системи на основі запропонованого методу. Онтологія завдання може бути створена за допомогою екземпляра онтології цієї абстрактної онтології, коли система діалогів застосовується до певного завдання.

2.4 Розробка класів онтології завдання

На основі запропонованої моделі онтології завдання, описаної в попередньому розділі, за допомогою Protégé 2.1 створюється абстрактна онтологія завдання. При реалізації розглядаються лише 3 рівні декомпозиції завдання. Для більш складних завдань, онтологію абстрактних завдань можна легко розширити, щоб містити кілька рівнів одиничних завдань, таких як UnitTask1, UnitTask2 тощо. Відносини між різними рівнями одиничних завдань та між іншими класами такі самі, як і між складовими завданнями та одиничні завдання. Усі правила та обмеження в онтології також залишаються незмінними.

На рисунку 2.5 представлені всі концепції або класи, розроблені в онтології завдання, включаючи Агент, Вибір, Об'єкт, Статус, SystemOperation, Task і

UserAction. Підкласи Task, такі як CompositeTask, UnitTask і BasicTask також включені в модель.

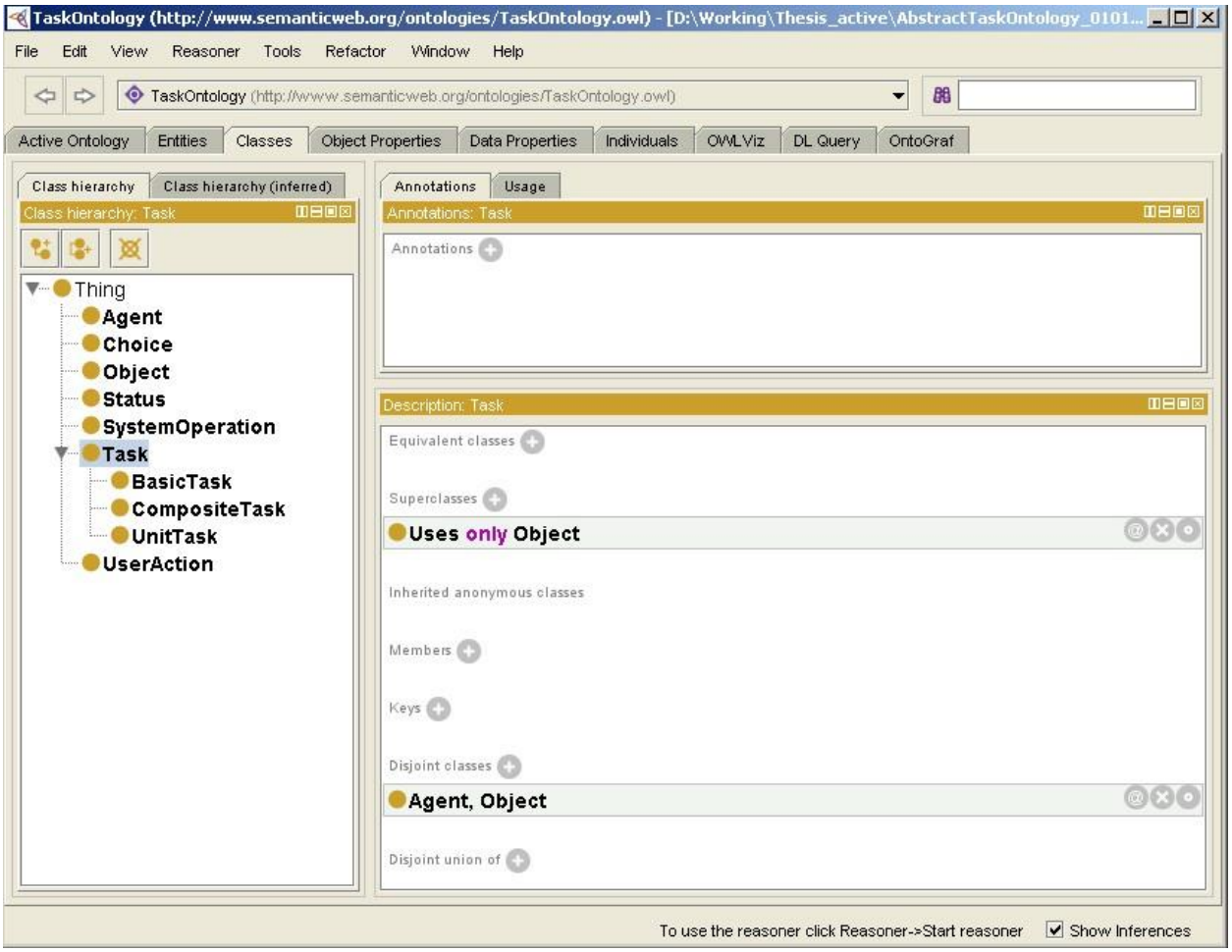


Рисунок 2.5 - Класи, розроблені в моделі онтології задач

Усі властивості об'єкта, які представляють різні відносини між поняттями, створеними в моделі онтології завдання, показані на рисунку. До них належать Спочатку, Далі, Розкласти на, Виконує, Виконує, Користувач та Використовує. Характеристики для кожної властивості також будуються в моделі, наприклад, асиметричні, іррефлексивні тощо.

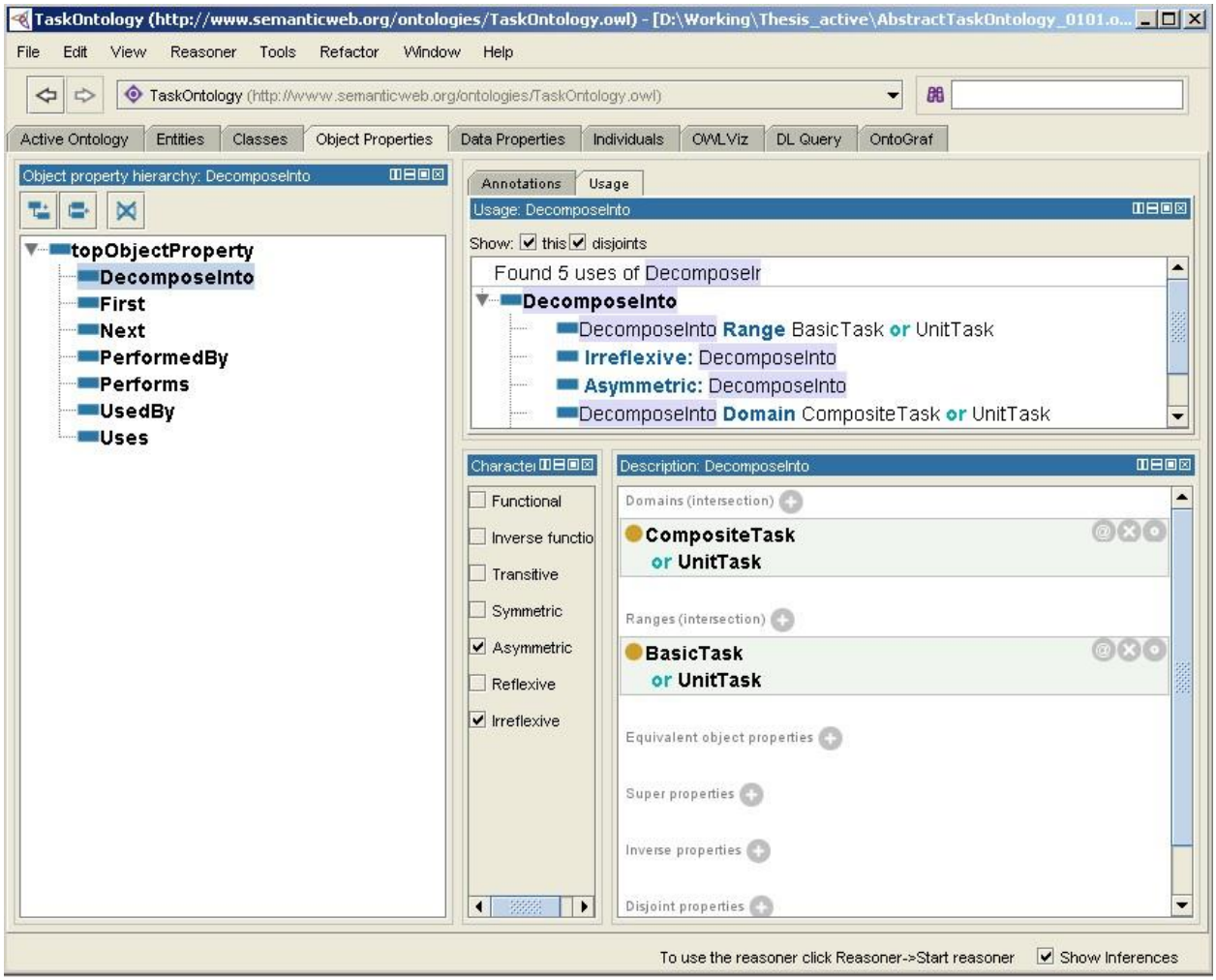


Рисунок 2.6 - Властивості об'єкта, розроблені в моделі онтології завдання

Властивості типу даних також прийняті в моделі онтології завдання, включаючи має вибір, має статус, має дію системи і дію користувача. Обмеження на цей тип властивостей встановлюються в моделі. Наприклад, доменом має дію користувача є Основне завдання.

Властивості даних, розроблені в моделі онтології завдання. Цю абстрактну онтологію завдання можна створити в онтології завдання для конкретних завдань. У цій роботі на основі онтології абстрактної задачі будуються онтології для двох завдань з різних доменів. Одне із завдань – послуга запозичень книг; а інший — онлайн-бронювання квитків на потяг.

2.5 Доменно-незалежна діалогова система

Незалежна від домену діалогова система розроблена на основі запропонованого методу, описаного вище. Система складається із запропонованого менеджера діалогів, контролера введення/виводу та діалогового інтерфейсу. Менеджер діалогів підключається до діалогового інтерфейсу через контролер введення/виводу та ілюструє архітектуру діалогової системи, реалізованої в цій роботі.

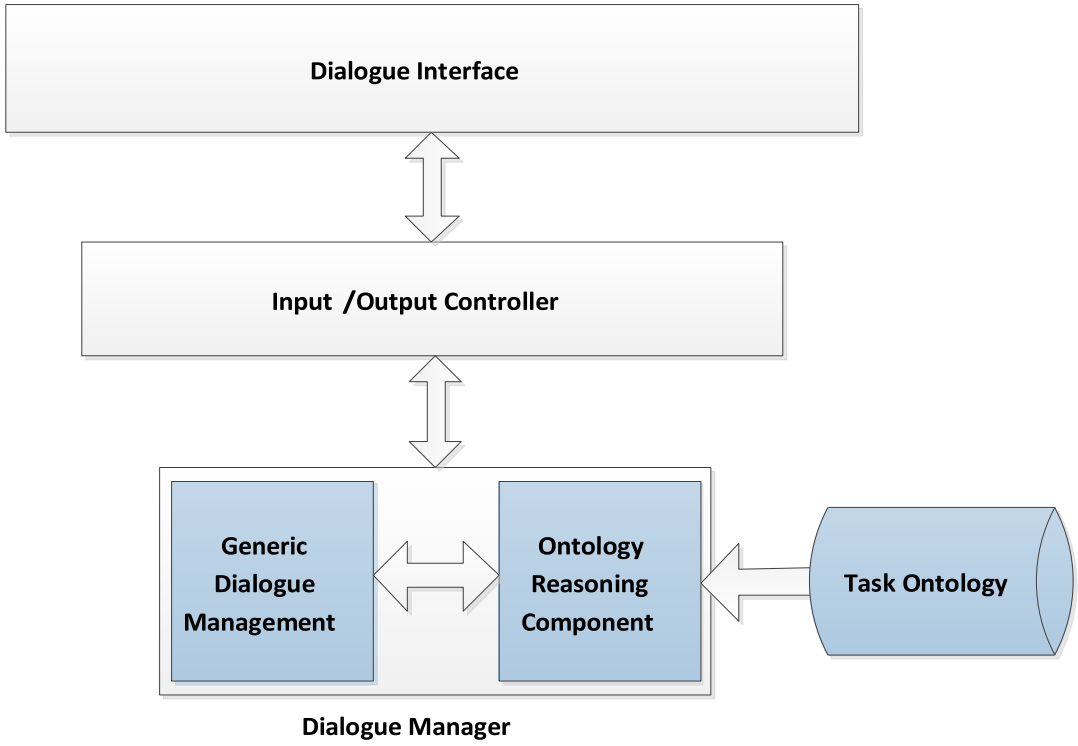


Рисунок 2.7 - Архітектура діалогової системи

Менеджер діалогів використовує підхід до управління діалогом на основі фрейму. Він використовується для керування взаємодією з людьми-користувачами, щоб інструктувати користувачів щодо досягнення своєї мети. Процес діалогу моделюється як набір завдань із заповнення слотів. Менеджер діалогів збирає заздалегідь визначену інформацію з відповідей користувачів, щоб заповнити прорізи. Наперед задана інформація зберігається в онтології завдання. Висловлювання, згенеровані менеджером діалогів, слоти та параметри їх значень

– усі витягуються з онтології завдання. Менеджер діалогів керує процесом діалогу на основі інформації, зібраної з відповідей користувачів, і онтологічних міркувань, щоб спрямовувати користувачів виконувати конкретне завдання.

Інтерфейс діалогу заснований на тексті, який відображає висловлювання, згенеровані системою діалогів, і надає один слот, який користувачі можуть заповнити. Висловлювання – це інформація про те, що користувачі повинні робити і що збирається робити інструмент, з яким він працює. Інструментом може бути машина або програма, яка допомагає користувачам виконати конкретне завдання, наприклад, придбати квиток на поїзд або позичити книгу. У задачі купівлі квитків на потяг мета – купити квиток, а програма онлайн-бронювання квитків на поїзд є інструментом для користувачів для досягнення цієї мети. Типові системні висловлювання включатимуть наступні дві частини: перша – дія, яку очікують користувачі, щоб виконати конкретне підзавдання; а ще одна операція, яку має інструмент – у цьому випадку програма онлайн-бронювання залізничних квитків матиме. Наприклад, система діалогу може відображати інформацію на кшталт «Ви повинні ввести інформацію про подорож, включаючи станцію відправлення, станцію прибуття, дату відправлення та дату повернення до форми бронювання», щоб вказати, що користувачі очікують робити на певному кроці. Тим часом інформація про поведінку програми бронювання також відображається, щоб інформувати користувачів про те, що відбувається в програмі бронювання. У цьому випадку це буде «Система шукатиме розклад поїздів, щоб знайти поїзди, які відповідають вашому плану подорожі. Після цього будуть відображені всі доступні потяги, які ви зможете вибрати». Після того, як користувач виконає конкретне підзавдання згідно з інструкціями системи діалогу, він може заповнити «Далі», щоб рухатися вперед, щоб отримати інструкції щодо наступного кроку до своєї мети.

Якщо є варіанти для користувачів, усі можливі варіанти будуть відображатися в інтерфейсі, щоб користувачі могли вибрати з. Очікується, що користувачі виберуть одного з них, щоб заповнити слот. Відповідно до вибору користувачів,

система діалогу підбере відповідний маршрут, щоб спрямовувати користувачів до виконання завдання. У прикладі бронювання квитків на потяг діалогова система відображатиме параметри «Подорож в обидва боки» та «В одну сторону», щоб користувачі могли вибрати тип квитка, який він хоче купити. Таким чином, інтерфейс відобразить вислів «Який квиток ви хочете купити? Будь ласка, виберіть один із наступних варіантів: «Подорож в обидва боки» або «В одну сторону»». Очікується, що користувачі введуть або «Подорож в обидва боки» або «В одну сторону», щоб заповнити слот. Потім відповідь користувачів передається контролеру введення/виводу, який надсилає інформацію до DM. DM визначить наступний крок на основі відповіді користувача, а також інформації, отриманої з онтології завдання.

Інформація про всі висловлювання, слоти та параметри для користувачів надходить з онтології завдання для конкретного завдання. Компонент аргументації менеджера діалогів отримуватиме відповідну інформацію з онтології завдання та керуватиме процесом діалогу з користувачами, поки завдання не буде завершено. Інформація, згенерована менеджером діалогів, буде надіслана до

Контролер введення/виводу, який дозволяє відображати текстове повідомлення в діалоговому інтерфейсі. Різні завдання матимуть різну онтологію завдань. Таким чином, система діалогу може спрямовувати користувачів виконувати різноманітні завдання з різних областей шляхом взаємодії з людьми-користувачами.

Система діалогів розроблена на Java 6.0 з використанням eclipse IDE 3.7. У реалізації також використовуються OWL API для OWL 2 і Pellet - аргумент OWL 2 для Java. 2.7 - блок-схема діалогової системи. На даний момент декомпозиція завдання в запропонованій моделі онтології задач містить три рівні. Для складніших завдань можна ввести додаткові рівні для декомпозиції завдання. У цьому випадку модель можна легко розширити до більш ніж трьох підкласів класу завдань. Крім того, система діалогу легко підтримує необхідну кількість рівнів.

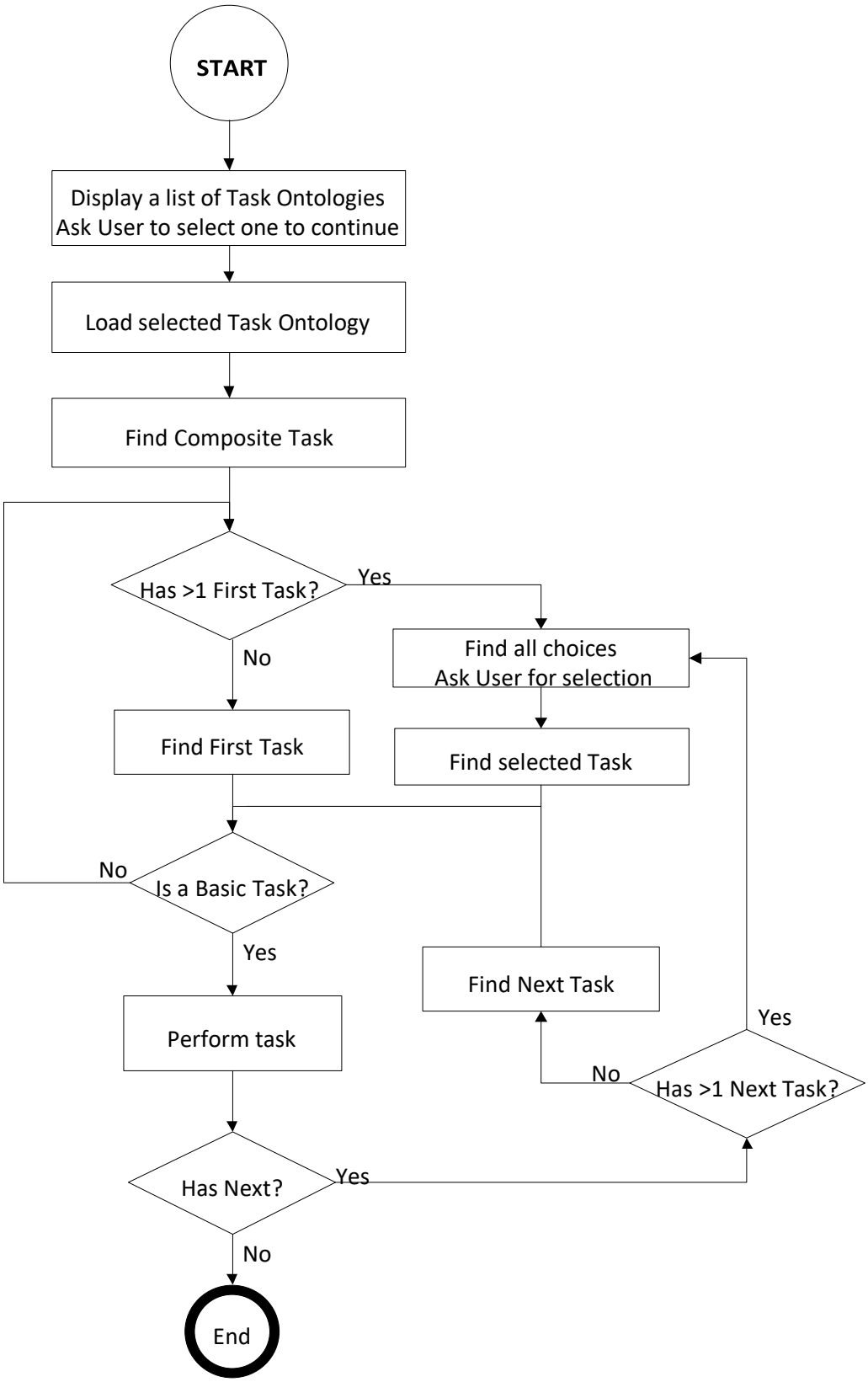


Рисунок 2.8 - Блок-схема діалогової системи

Для створення діалогового інтерфейсу з користувачами використовується інструмент Java GUI, Window Builder.

На рис. 2.9 показаний інтерфейс діалогової системи.

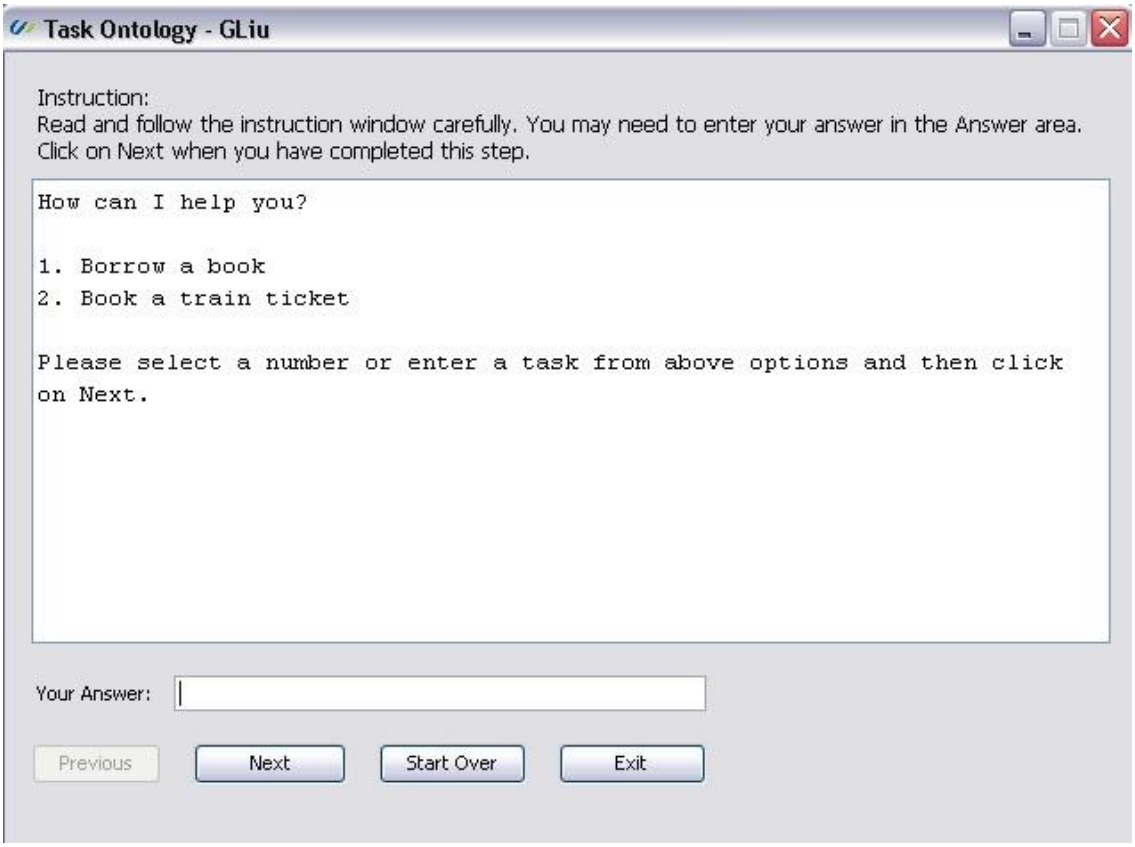


Рисунок 2.9 - Діалоговий інтерфейс діалогової системи

Логотип установи та назву проекту можна додати та відобразити у верхній частині вікна.

Середина інтерфейсу — це вікно, яке використовується для відображення системних висловлювань користувачам. Повідомлення в основному витягується з онтології завдання для певного завдання. Якщо розмір повідомлення завеликий для відображення в одному вікні, з правого боку з'явиться смуга прокрутки, що дозволить користувачам прокручувати вниз і вгору.

Під цим вікном є менше поле для користувачів, щоб заповнити свою відповідь. Кожного разу, коли система діалогів відображає параметри користувачам, їх можна буде редагувати. І в центрі уваги інтерфейс також налаштований на цьому полі. Тут передбачено два способи: за номером або за назвою завдання. Введений текст встановлюється без урахування регістру, що

зручно для користувачів. Якщо користувачі введуть неправильну відповідь, поряд з полем відобразиться повідомлення про помилку, щоб нагадати користувачам повторно ввести дійсну відповідь.

У нижній частині інтерфейсу є чотири кнопки: «Назад», «Далі», «Почати знову» та «Вийти», щоб користувачі могли вибрати, що вони хочуть.

a) Кнопка «Назад» дозволяє користувачам повернутися до одного або кількох кроків за бажанням.

b) Далі – це кнопка, за допомогою якої користувачі можуть продовжити виконання завдання.

в) Почати спочатку: дозволяє користувачам починати з початку в будь-який час.

d) Вийти: користувачі можуть вийти в будь-який час.

При необхідності деякі кнопки відключаються. Наприклад, кнопка «Назад» у цьому діалоговому вікні вимкнена, оскільки для поточної взаємодії немає попереднього кроку. Ці функції полегшують користувачам ефективну взаємодію з системою діалогу.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ОНТОЛОГІЧНИХ МОДЕЛЕЙ В МОДАЛЬНИХ ТА ДІАЛОГОВИХ ФОРМАХ

3.1 Діалогова система для послуги позичання книг

Діалогова система, реалізована в роботі імплементується з двома завданнями з різних областей, одне — послуга запозичення книг; а інший — онлайн-бронювання квитків на потяг. Для них створюються дві онтології завдань відповідно шляхом створення екземплярів онтології абстрактної онтології завдання. Онтології завдань надають діалоговій системі знання про кожне завдання, тому систему можна використовувати для двох завдань без будь-яких змін.

Іншими словами, система діалогу не залежить від завдання чи домену. У наступному докладно описуються два випадки відповідно.

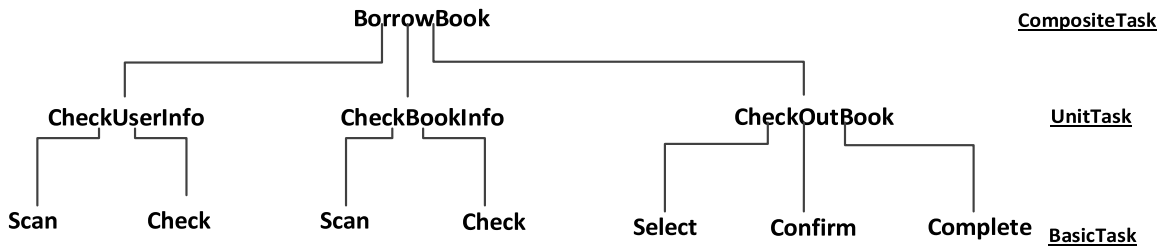


Рисунок 3.1 - Дерево декомпозиції завдань для служби запозичень книг

Завдання послуги книжкового запозичення можна розкласти на набір підзадач різного ієрархічного рівня. Рисунок ілюструє ієрархію декомпозиції завдань служби запозичень книг у бібліотеці. На верхньому рівні «Книга позики» — це складне завдання. В онтології завдань для служби запозичення книг, Borrow Book є екземпляром (або членом Protégé) підкласу Composite Task. Його можна розкласти на три одиничні завдання, включаючи перевірку інформації про користувача,

Перевірте Book Info та Check Out Book, які є екземплярами підкласу Unit Task. Серед цих Завдання блоку, Перевірка інформації про користувача далі розкладається на дві основні завдання: сканувати картку користувача та

Перевірити статус користувача; Інформація про чекову книгу розкладається на сканування штрих-коду та статус чекової книги.

Книга виписки розкладається на Вибрати тип позики, Підтвердити інформацію про позику та завершити транзакцію. Усі сім базових завдань знаходяться в нижній частині ієрархії завдань і створені як екземпляри основного завдання в онтології завдань.

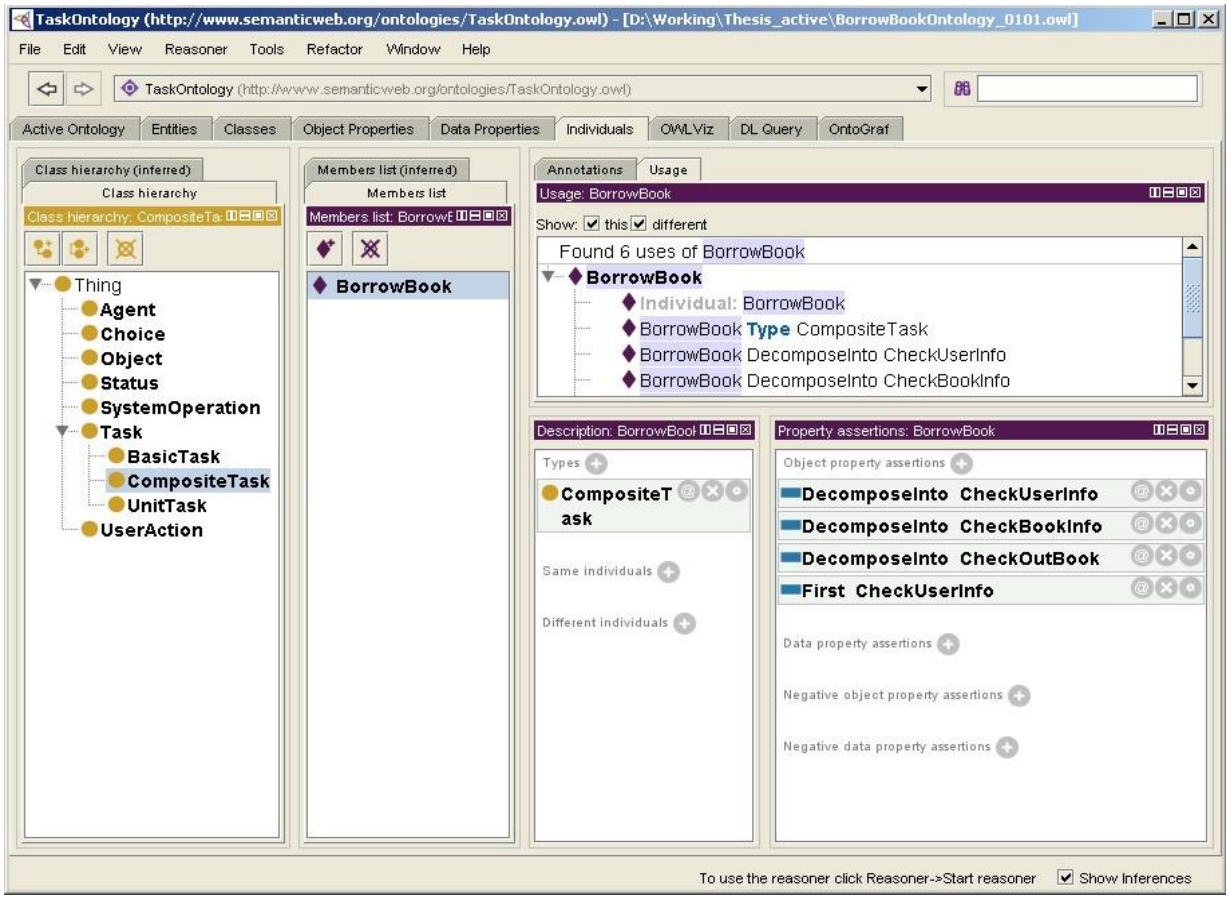


Рисунок 3.2 - Примірник складеної задачі та її властивості

На рис. 3.2 представлено екземпляр Складене завдання, Позичи книгу та його зв’язки з іншими екземплярами. Зв'язки представлені як властивості об'єкта в онтології для цього конкретного завдання, включаючи:

- Розкласти на перевірку інформації про користувача
- Розкласти на інформацію чекової книги
- Розкласти на виписну книгу
- Спочатку перевірте інформацію про користувача

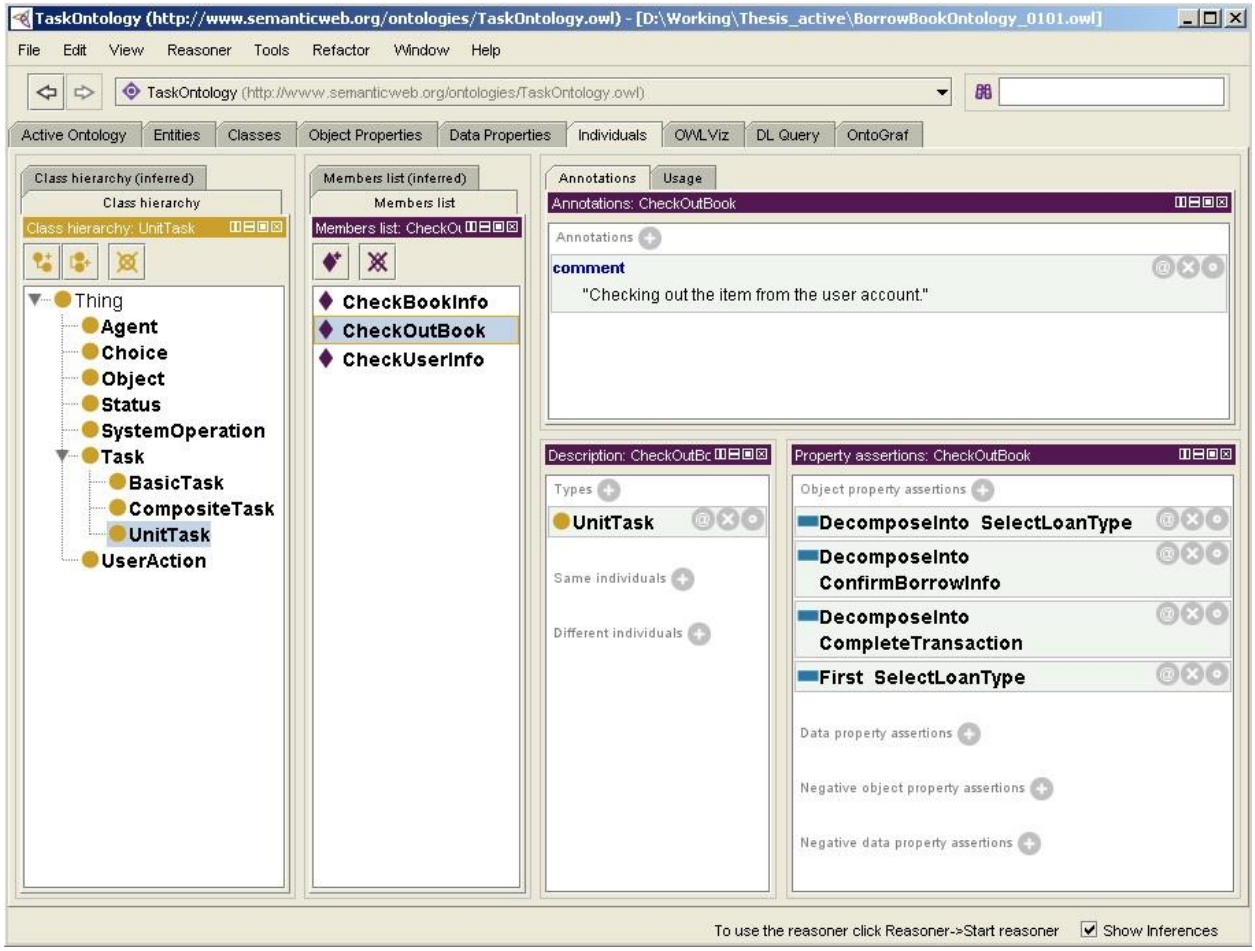


Рисунок 3.3 - Примірники Завдання одиниці та їх властивості

На рис. 3.3 показані екземпляри задачі одиниці, у тому числі Перевірити інформацію про користувача, Перевірити інформацію про книгу та Перевірити книгу. Вони мають власні властивості об’єкта. Наприклад,

Отримати книгу має такі властивості об’єкта:

- Розкласти на вибір типу кредиту
- Розкласти на підтвердження інформації про позику
- Розкласти на повну транзакцію

- Спочатку виберіть тип кредиту

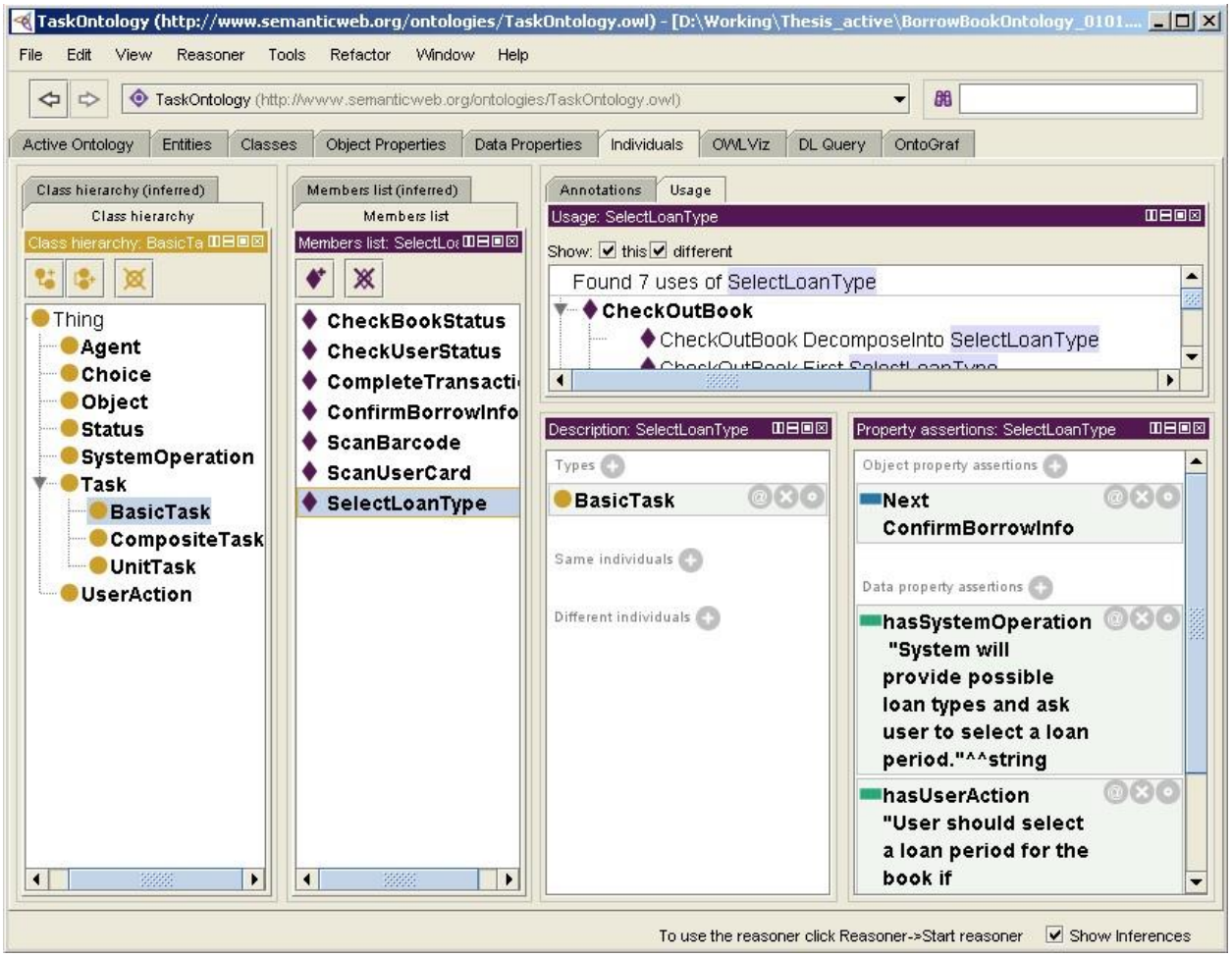


Рисунок 3.4 - Основне завдання та їх властивості

Екземпляри Основне завдання та їх властивості відображені на рисунку 3.4.

Виберіть тип кредиту як приклад. Він містить властивості об’єктів і даних. Властивість об'єкта: Далі Підтвердити інформацію про позику.

Властивості даних включають: має дію користувача «Користувач повинен отримати термін для книги, якщо є» має дію системи «Система надасть можливі типи позики та попросить користувачів вибрати термін позики»

Ці правила та обмеження, створені в моделі абстрактної онтології, автоматично застосовуються до цієї онтології. Створена онтологія для служби запозичення книги використовується в системі діалогів, створеній у розділі, щоб

інструктувати користувачів виконати завдання запозичення книги за допомогою взаємодії між системою діалогу та користувачами.

3.2 Опис процесу діалогу для послуги позичання книг

Після того, як користувачі вибирають послугу запозичення книг, система діалогу починає отримувати інформацію з онтології завдання для цього завдання та збирати інформацію від користувачів шляхом взаємодії з ними. На рисунку показано інтерфейс діалогу для підзавдання «Сканувати картку користувача».

Це перше основне завдання, яке вказує користувачам на дії та пояснює роботу системи.

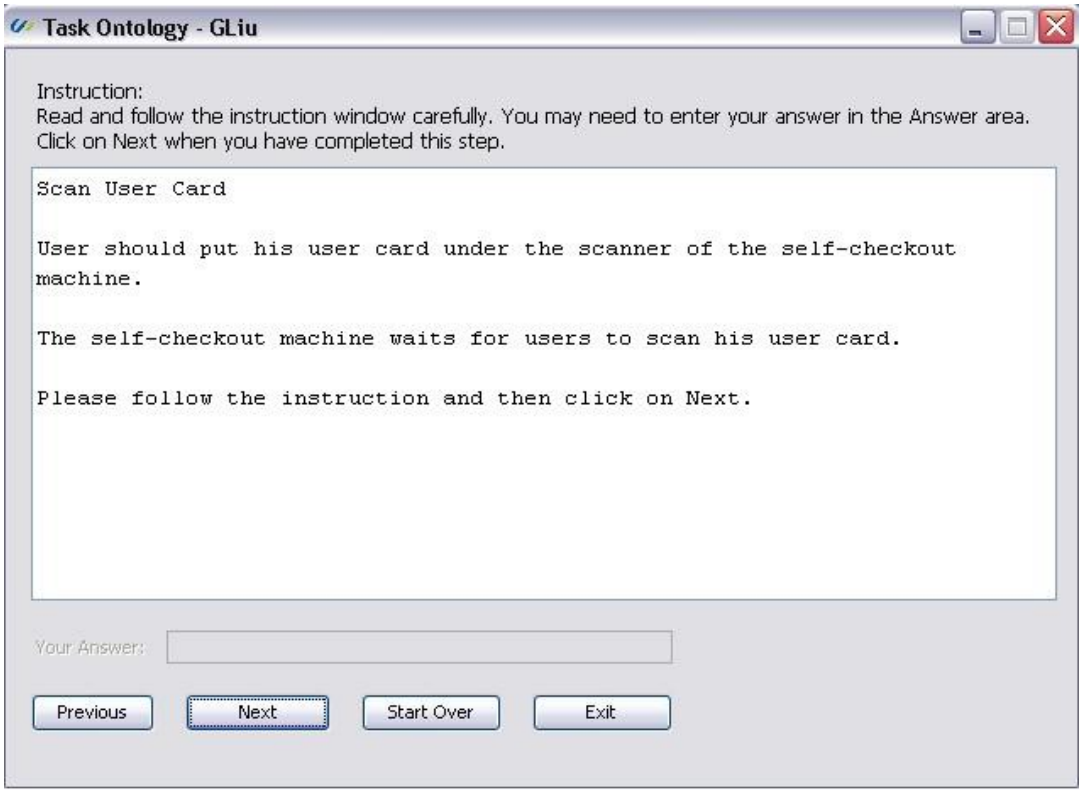


Рисунок 3.5 - Інтерфейс діалогу для підзадачі «Сканувати картку користувача»

У великому вікні відображення назва завдання «Сканувати картку користувача» походить від назви поточної підзадачі, встановленої в онтології

завдання запозичення книги. Інструкція для користувачів: «Користувач повинен покласти свою картку під сканер касового апарату» та пояснення системних операцій «Автомат чекає, поки користувач відскануює свою картку» – це значення даних. властивості для цієї підзадачі зберігаються в онтології завдання запозичення книги. При необхідності бібліотека, яка використовує діалогову систему, може змінити значення властивості даних безпосередньо через редактор онтології. Крім того, якщо є зміна політики або інші зміни, що впливають на підзадачі для цієї служби, бібліотека може сама змінити онтологію завдання. Немає необхідності просити розробників діалогової системи змінити код.

Оскільки користувачі не очікують вводити відповідь на цьому кроці, поле для користувачів, які заповнюють свою відповідь, вимкнено. Натомість фокусується кнопка «Далі», яка зазвичай вибирається користувачем на цьому кроці. Інші кнопки ввімкнені. Користувачі можуть повернутися до останнього кроку або більше попередніх кроків, натиснувши кнопки. Поточне завдання може бути перервано з якоїсь причини, тому користувачі можуть у будь-який момент натиснути «Почати спочатку» або «Вийти».

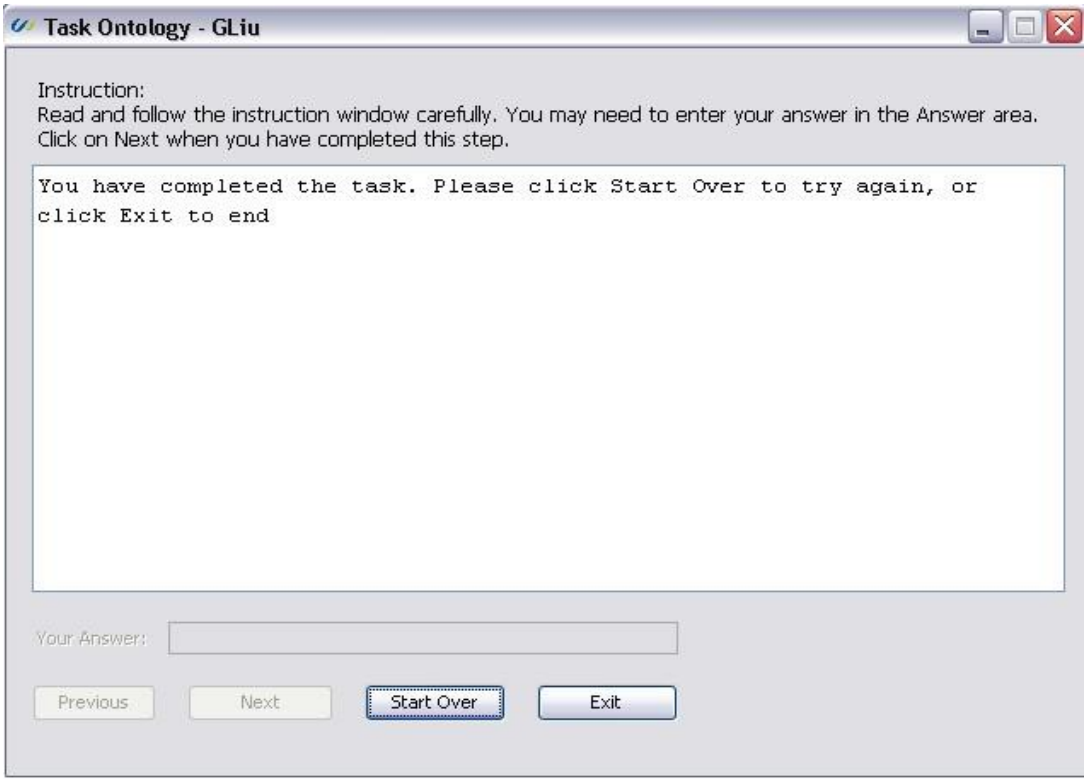


Рисунок 3.3 - Інтерфейс діалогу після виконання завдання

Дотримуючись інструкцій діалогової системи, будуть виконані всі основні підзавдання для цієї послуги, включаючи Перевірку статусу користувача, Сканування штрих-коду книги, Перевірку стану книги, Вибір типу позики, підтвердження інформації про позику та завершення транзакції. Коли все завдання буде виконано, користувачі мають можливість почати знову або вийти з системи.

3.3 Послуга онлайн бронювання квитків на поїзд

Порівняно із завданням запозичення книги, завдання онлайн-бронювання залізничних квитків є складнішим. Він містить не лише послідовні підзадачі, а й альтернативні підзадачі. Для альтернативних підзадач користувачі можуть зробити вибір серед усіх наданих системою варіантів. Вибір користувачами визначить наступне підзавдання. Запропонована модель онтології завдань підтримує ці типи підзадач за допомогою класу Вибір. і властивість даних має Вибір.

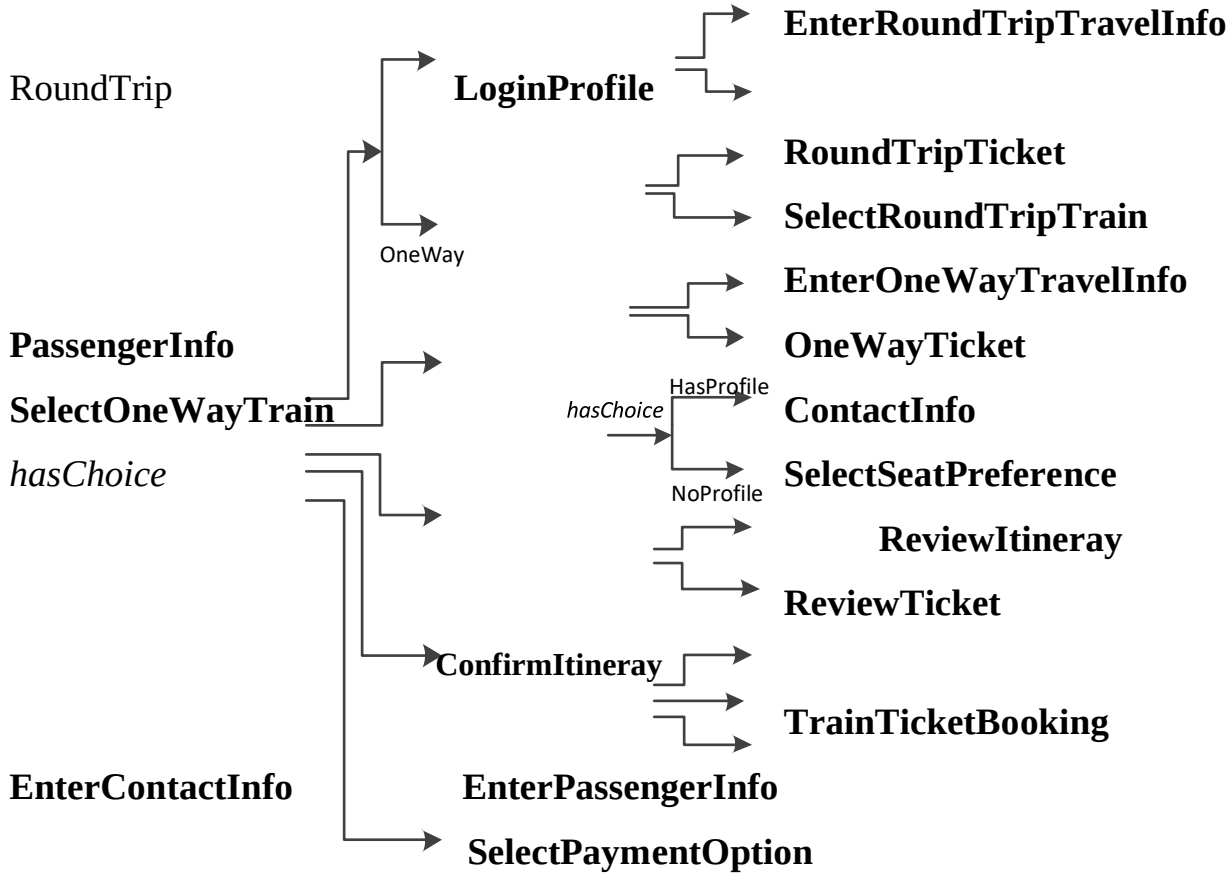


Рисунок 3.4 - Розклад завдання бронювання залізничних квитків

Рис. 3.4 ілюструє розкладання завдання для онлайн-завдання бронювання залізничних квитків наступним чином:

1) Один екземпляр складеного завдання: бронювання залізничних квитків

2) Шість екземплярів одиничне завдання:

- Туди й назад
- Квиток в один бік
- Перегляд квитка
- Контактна інформація
- Інформація про пасажира
- Здійсніть оплату.

Серед них квиток в обидва боки та квиток в один кінець є альтернативними завданнями, які мають виконуватися залежно від вибору користувачів. 3) Тринадцять екземплярів базових завдань:

- Введіть інформацію про подорожі туди й назад
- Виберіть Потяг туди й назад
- Введіть інформацію про подорож в одну сторону
- Виберіть потяг в одну сторону
- Огляд Маршрут
- Підтвердьте маршрут
- Профіль входу
- Введіть контактну інформацію
- Введіть інформацію про пасажира
- Виберіть Налаштування місця
- Виберіть Варіант оплати
- Введіть інформацію про картку
- Підтвердьте бронювання квитків

Ці екземпляри складають різні завдання одиниці. Серед них «Профіль входу» та «Введіть контактну інформацію» є альтернативними завданнями, які будуть виконуватися залежно від вибору користувачів.

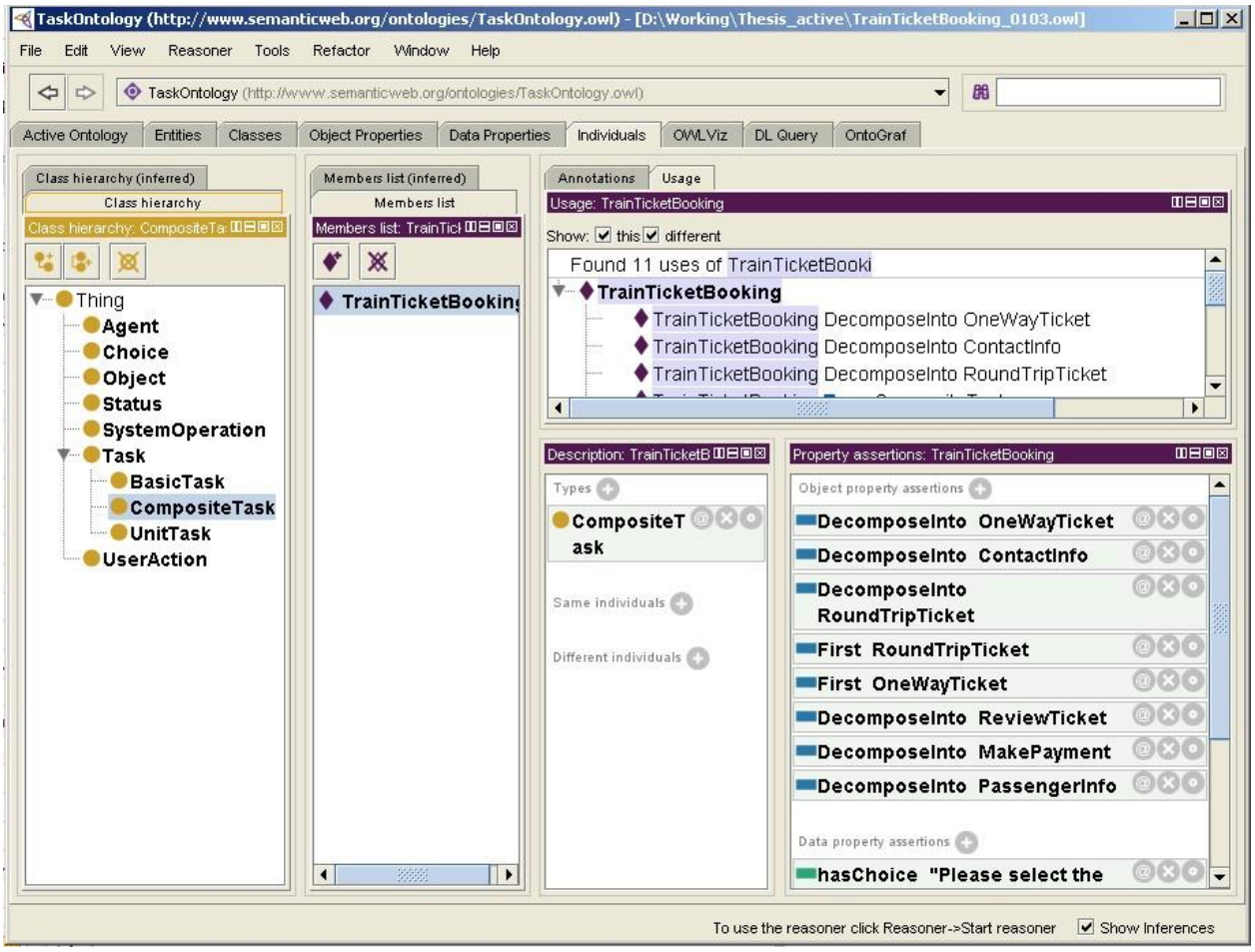


Рисунок 3.5 - Складена задача в онтології бронювання залізничних квитків

На рис. 3.5 представлений екземпляр Складене завдання та його властивості об'єкта та властивість даних таким чином:

- Розкласти на квиток туди й назад
- Розкласти на квиток в один кінець
- Розкласти на оглядовий квиток
- Розкласти на контактну інформацію
- Розкласти на інформацію про пасажира
- Розкласти на Здійснити платіж

- Перший квиток туди й назад
- Перший квиток в один бік має вибір «Оберіть тип квитка, який ви хочете забронювати:
- Квиток в обидва боки
- Квиток в один бік"

У заявах є дві властивості Спочатку. Якщо для екземпляра є кілька властивостей Спочатку., менеджер діалогів, а точніше компонент міркування онтології, перевірить властивість має вибір,щоб побачити значення параметрів, і попросить користувачів вибрати один із можливих варіантів. Цінність цих параметрів має бути точно однаковою з підзадачами (екземплярами), на які вони посилаються. Для інших властивостей об’єкта, наприклад далі,застосовується те саме правило. У прикладі, який ми обговорюємо, властивість даних має Вибір містить два параметри.

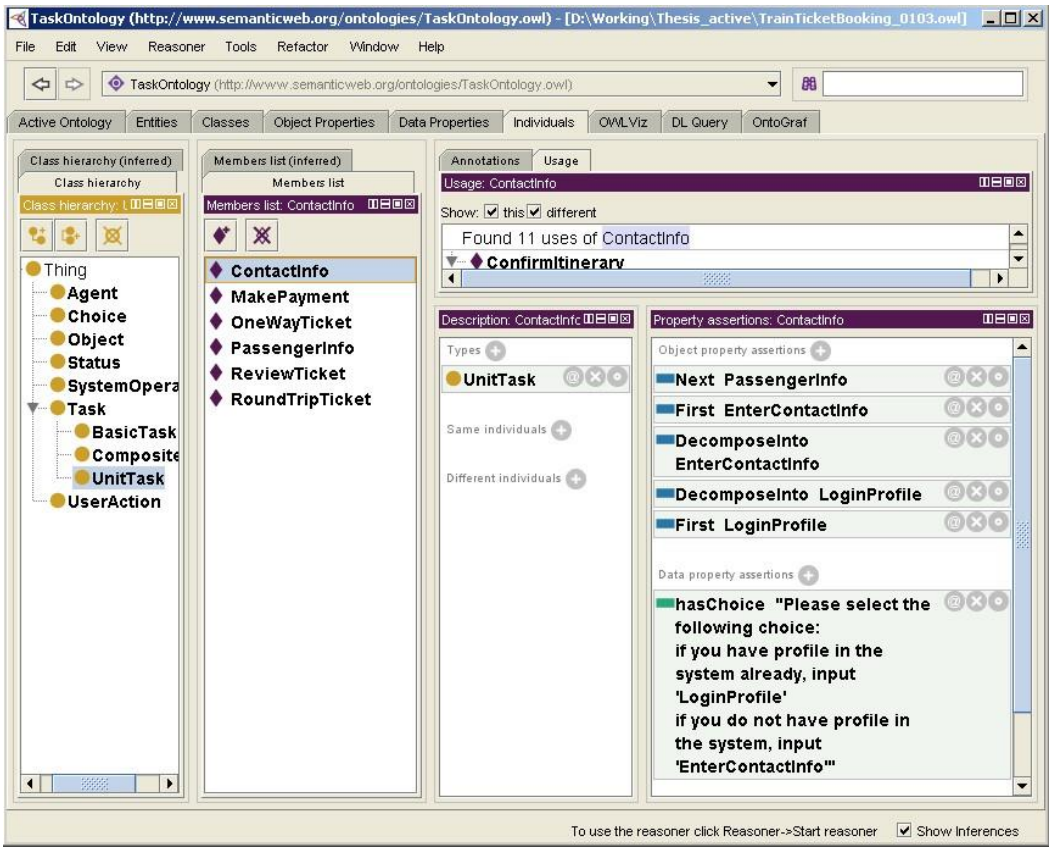


Рисунок 3.6 - Екземпляри Unit Task в онтології Бронювання залізничних квитків

«Квиток в обидва боки» та «Квиток в один кінець», які вказують на два екземпляри завдання одиниці. Будь-яка з них, вибрана користувачами, буде першою підзавданням у потоці завдань для завдання Бронювання залізничних квитків.

На рис. 3.6 показані екземпляри Unit Task та їх властивості. Деякі з них мають властивості даних, а також властивості об'єктів, наприклад, контактну інформацію; тоді як інші мають лише властивості об’єкта, такі як квиток туди й назад, квиток в одну сторону, квиток на огляд, інформація про пасажира та оплата. Подібно до властивості має Вибір в екземплярі складеної задачі

Бронювання квитків на потяг, екземпляр задачі підрозділу містить параметри, які користувачі можуть вибрати. Якщо у користувача вже є профіль у системі, він/вона може вибрати Профіль входу; в іншому випадку він/вона повинен вибрати Ввести контактну інформацію, щоб ввести свою контактну інформацію в систему бронювання.

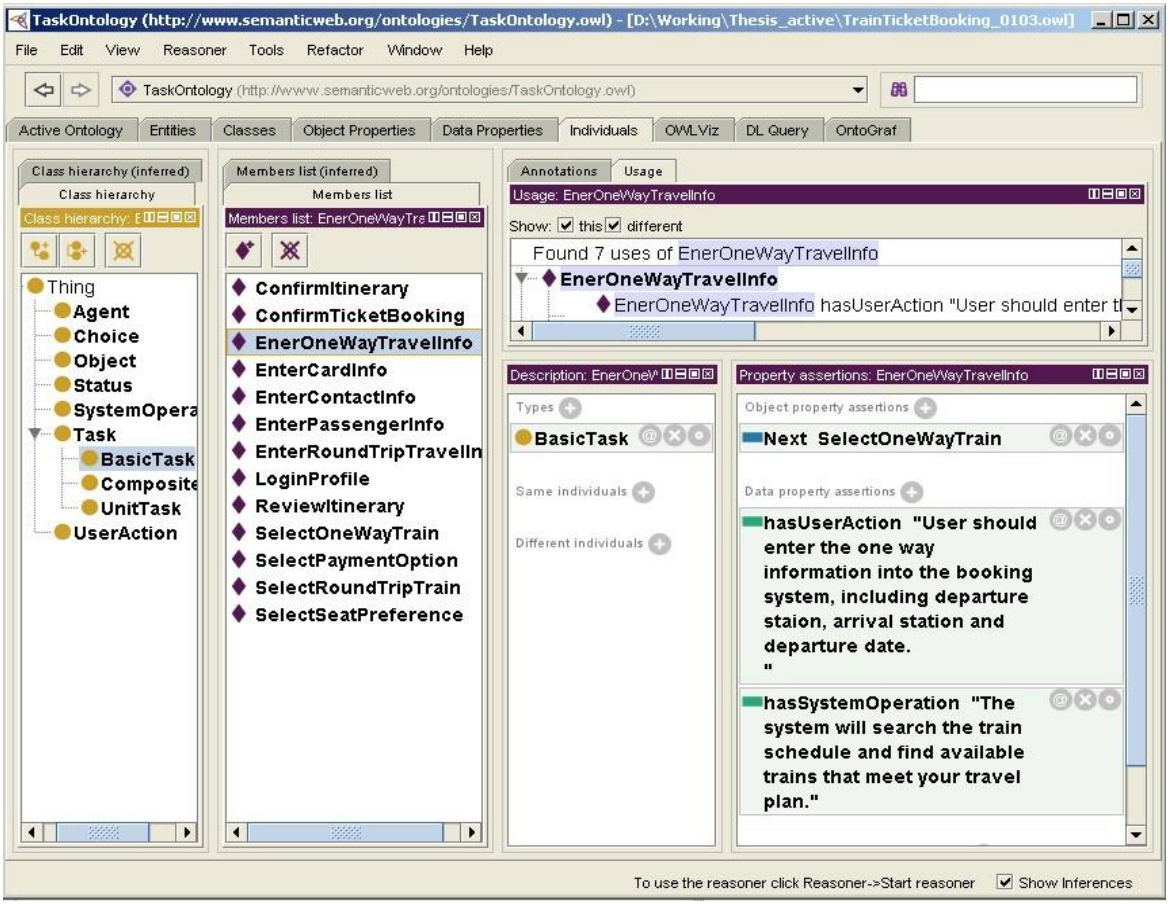


Рисунок 3.7 - Екземпляри BasicTask в онтології TrainTicketBooking

Екземпляри Основне завдання та їх властивості показані на рис, 3.7. Усі вони мають властивості даних має дію користувача та має дію системи, які вказують на дії, які користувачі мають виконати, та операції на стороні системи в конкретному екземплярі Основне завдання. Крім того Підтвердьте бронювання квитків, усі інші екземпляри мають власність об'єкта Далі. Це тому, що Підтвердьте бронювання квитків є останню підзавданням для всього завдання.

Якщо користувачі вибрали завдання онлайн-бронювання квитків на потяг, система діалогу отримає знання про це завдання з онтології завдання бронювання залізничних квитків. На рис. 3.8 представлений інтерфейс діалогу, який представлений користувачам для першого підзавдання цього завдання.

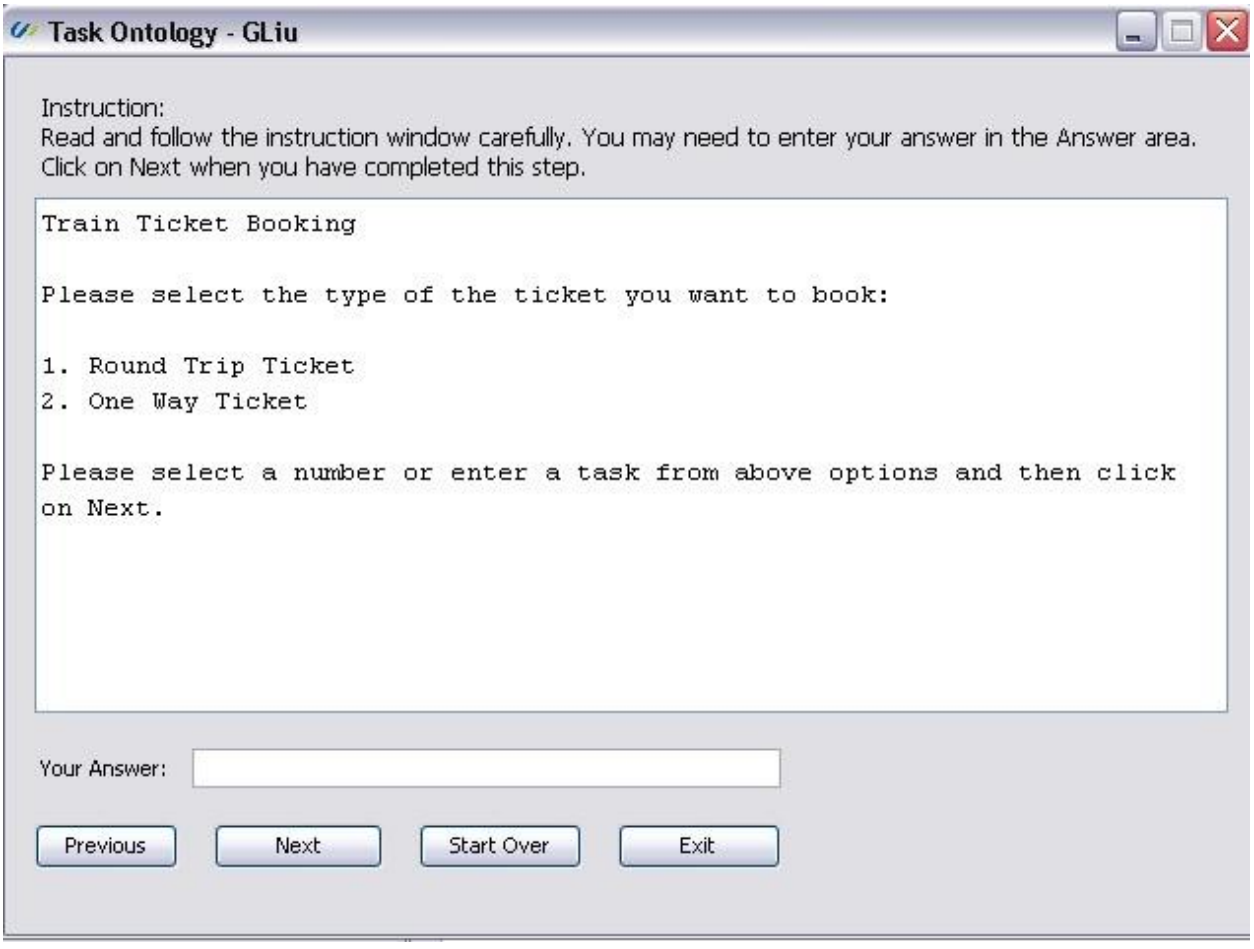


Рисунок 3.8 - Інтерфейс діалогу для завдання бронювання квитків на поїзд

У більшому вікні користувачам надаються два варіанти: квиток туди й назад або в одну сторону Квиток. Користувачі можуть ввести номер або назву

потрібного завдання. Наступне підзавдання діалогової системи буде визначатися вибором користувачів. У цьому прикладі: квиток в обидва боки або квиток в один кінець.

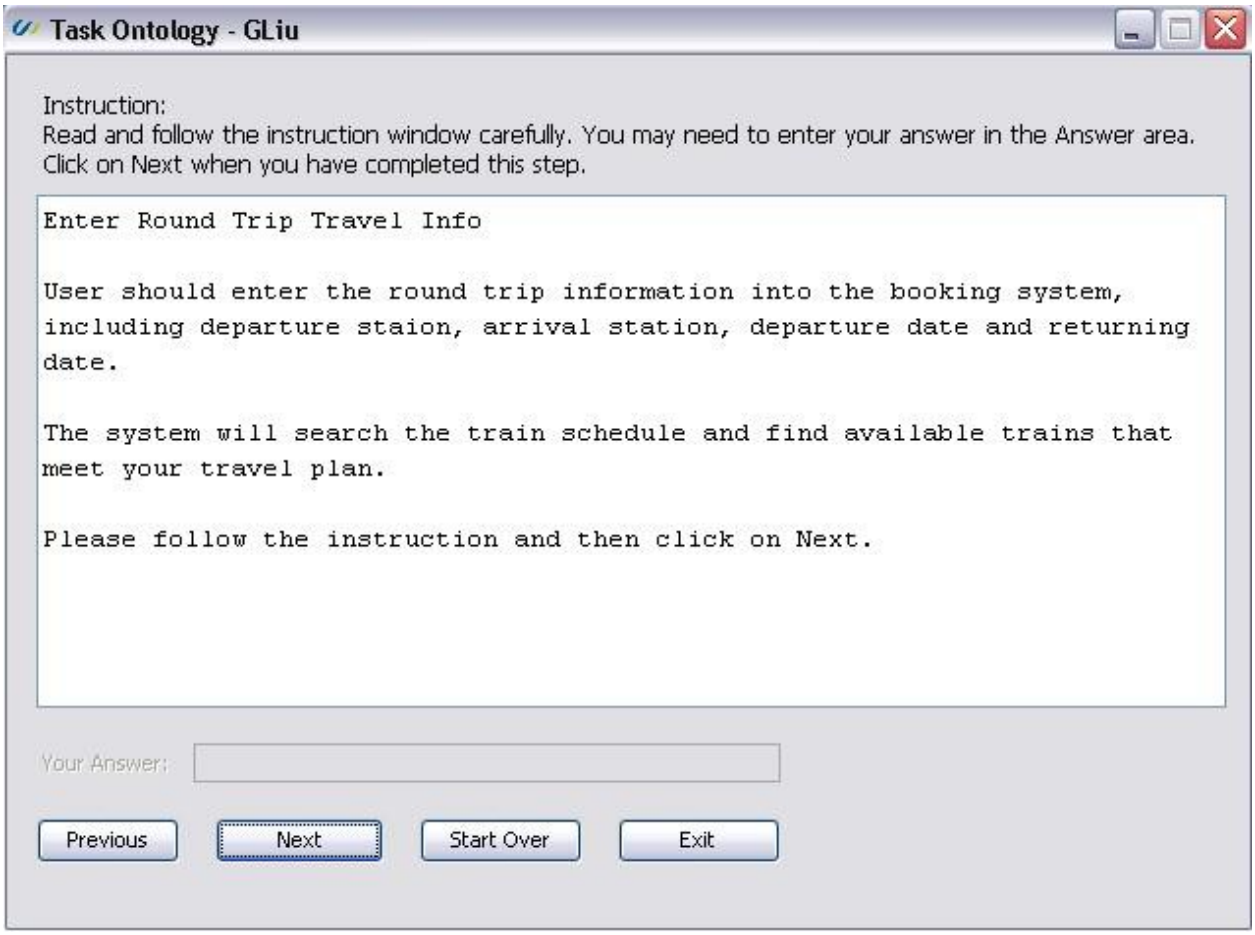


Рисунок 3.9 - Інтерфейс діалогу для завдання введення інформації про подорожі в обидва боки

Якщо квиток туди й назад вибрано користувачами, система розпочне підзавдання з бронювання поїздки туди й назад. Перше основне завдання – це ввести інформацію про подорожі в обидва боки, тому воно з’явиться в діалоговому інтерфейсі. Після того, як користувачі виконали це підзавдання, вони повинні натиснути «Далі», щоб продовжити. Потім система діалогу вказуватиме користувачам вибрати поїзд туди й назад, що є підзавданням після введення інформації про подорожі в обидва боки. Рисунок є діалоговим інтерфейсом для вибору поїзда туди й назад.

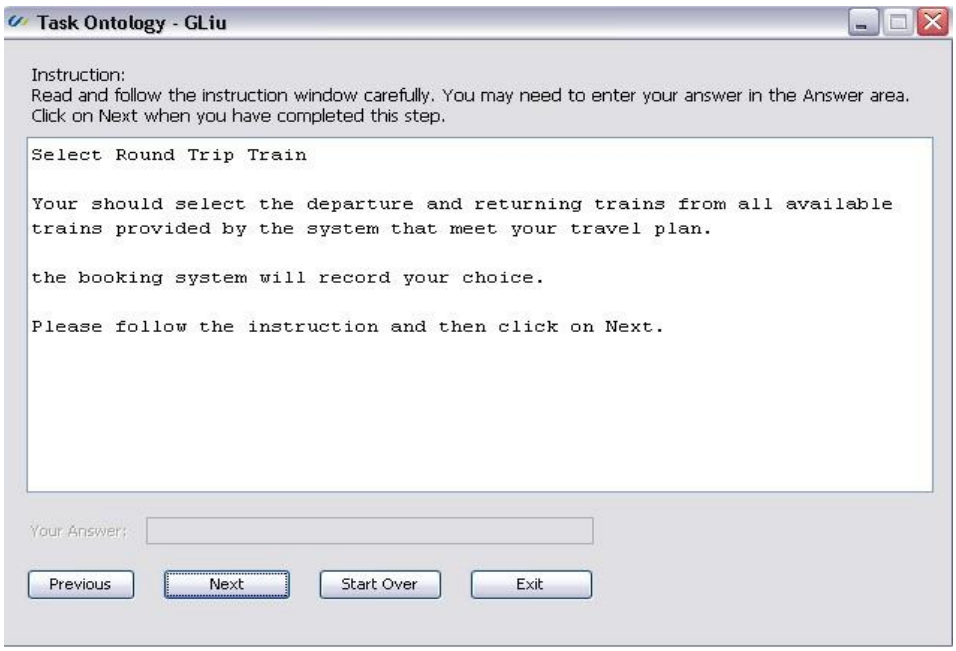


Рисунок 3.10 - Інтерфейс діалогу для завдання вибору поїзда туди й назад

На першому кроці, якщо користувачі вибирають квиток в один кінець, тоді в діалоговому інтерфейсі буде відображатися «Введіть інформацію про подорож в одну сторону», а не «Введіть інформацію про подорож в обидва боки». На рисунку показаний інтерфейс діалогу для завдання «Ввести інформацію про подорож в одну сторону».

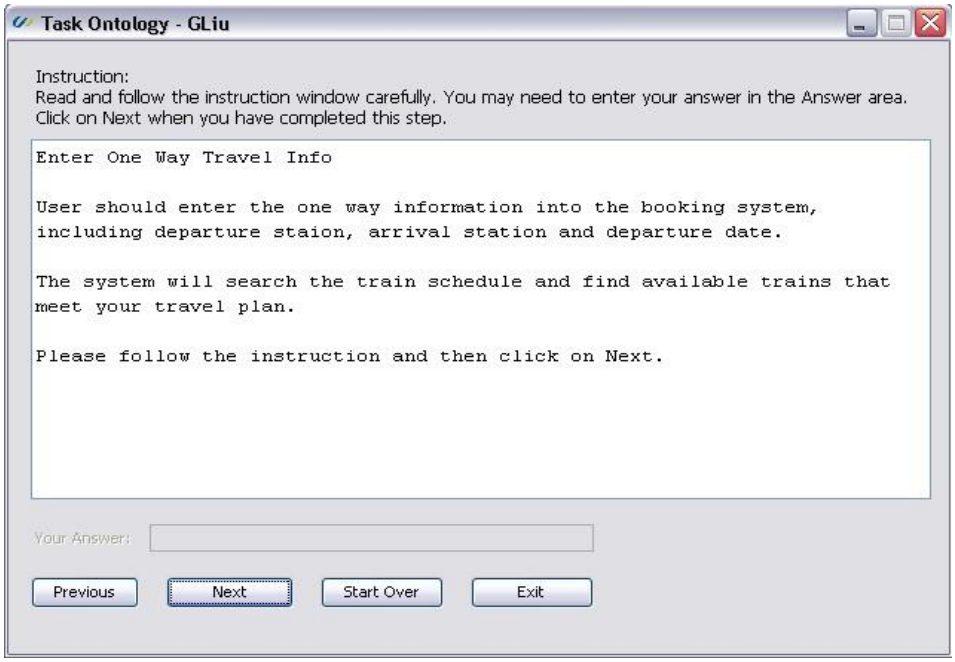


Рисунок 3.11 - Інтерфейс діалогу для завдання введення інформації про подорож в одну сторону

Після введення інформації про подорож в одну сторону, наступним підзавданням буде Виберіть Потяг в одну сторону. Після завершення або Виберіть поїзд в обидва боки або виберіть потяг в одну сторону, перегляньте маршрут стане наступним підзавданням для обох. Користувачі побачать інтерфейс діалогу, показаний на рис. 3.12 .

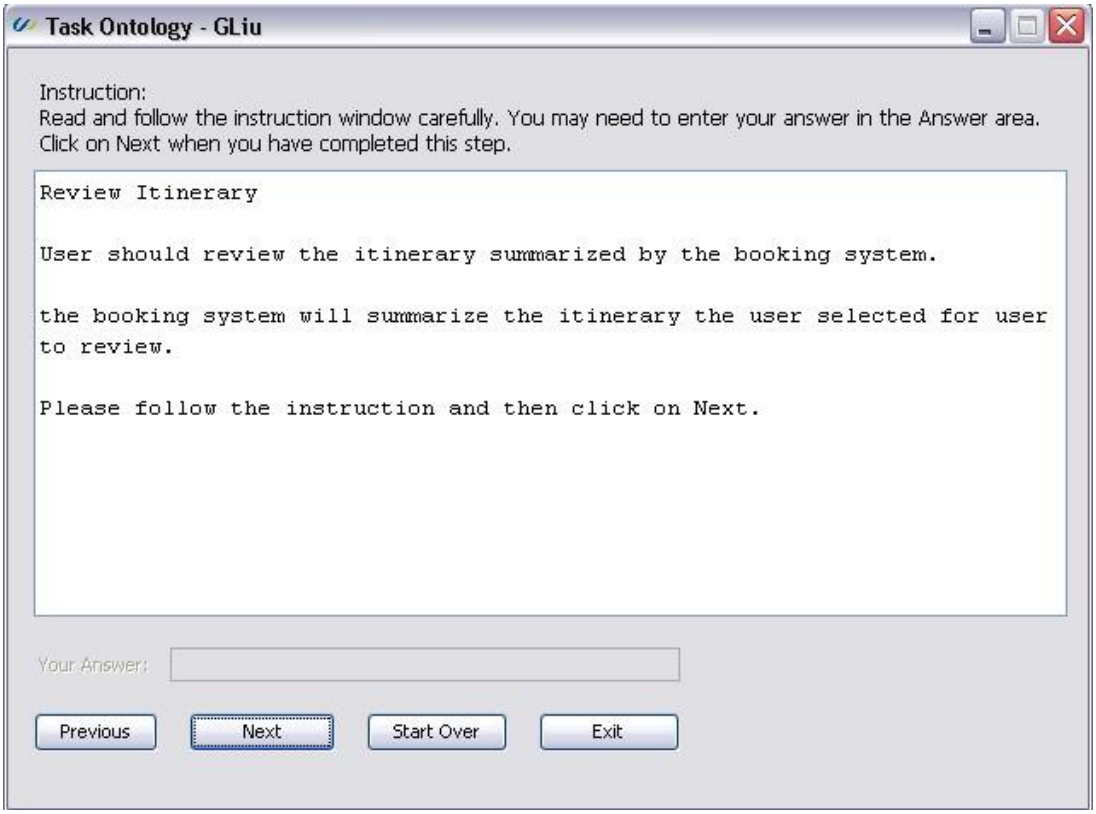


Рисунок 3.13 Інтерфейс діалогу для маршруту огляду

Користувачі можуть слідувати інструкціям діалогової системи, щоб виконати все завдання. Інтерфейс діалогу надасть варіанти «Почати знову» та «Вийти», щоб користувачі могли продовжити роботу з іншими завдань, повторити поточне завдання або вийти з діалогової системи.

ВИСНОВКИ

В представленій дипломній роботі запропоновано модель онтології завдання, яку можна створити в онтології задач для конкретних завдань. На основі моделі онтології завдання структура менеджера діалогів також модифікована, щоб включати загальний компонент управління діалогом і компонент міркування онтології. Компонент обґрунтування онтології витягує знання, специфічні для задачі, з його онтології завдання.

Таким чином, менеджер діалогів є незалежним від завдань, і система діалогів, заснована на ньому, може застосовуватися до нових завдань або змінених існуючих завдань.

Крім того, модель онтології завдань містить різноманітні типи зв'язків між завданнями та їх підзадачами, включаючи декомпозицію, послідовні та альтернативні зв'язки та їх властивості. Вона також враховує контекст виконання завдання, що дозволяє обробляти більш складні завдання більш гнучко порівняно з попередніми підходами.

Запропонований метод сприяє поширенню та повторному використанню системи знань і діалогів між різними доменами. Прийняття онтології та запропонованої моделі завдання має вплив на майбутні дослідження та розробку систем діалогу, незалежних від предметної області. Це також впливає на застосування систем діалогу в більш широкому діапазоні сфер.

Модель онтології завдань була використана для створення онтології завдань для двох завдань з різних доменів. На основі моделі та модифікованого менеджера діалогів була розроблена й експериментована система діалогів з двома завданнями. Експеримент показує, що діалогова система здатна справлятися з різними завданнями. Іншими словами, система діалогу не залежить від завдання чи домену.

Системи діалогу використовуються в багатьох областях, але всі вони обмежені певними специфікаціями. З іншого боку, розробка та підтримка

діалогової системи коштує дорого. Метод, розроблений у цій роботі, сприятиме повторному використанню знань та обміну діалоговими системами та допоможе знизити вартість розробки. Майбутня робота може дослідити методи, що автоматизують побудову онтології завдання, щоб полегшити процес створення онтології для різних завдань та доменів.

СПИСОК ПОСИЛАНЬ НА ДЖЕРЕЛА

1. AMOL RDF Record, 2001 http://metadata.net/harmony/amol.rdf
2. “AMOL Vase Description,”, 2001. http://metadata.net/harmony/amol.html
3. CIDOC Conceptual Reference Model Special Interest Group, http://cidoc.ics.forth.gr/
4. “CIMI ABC Modeling Examples,”, 2001. http://metadata.net/harmony/cimi_modelling.htm
5. CIMI Consortium, 2001 http://www.cimi.org
6. Dublin Core Metadata Element Set, Version 1.1, 1999 http://www.dublincore.org/documents/dces/
7. Dublin Core Metadata Initiative, http://dublincore.org
8. Dublin Core Qualifiers, 2000 http://purl.org/DC/documents/rec/dcmes-qualifiers20000711.htm
9. “Functional Requirements for Bibliographic Records,” International Federation of Library Associations and Institutions March 1998. http://www.ifla.org/VII/s13/frbr/frbr.pdf
10. The Harmony Project, http://metadata.net/harmony
11. Images and Metadata Records Provided by CIMI Members for Harmony/CIDOC CRM Museum
12. Interoperability Research, 2001 http://archive.dstc.edu.au/CIMI/
13. INDECS Home Page: Interoperability of Data in E-Commerce Systems, http://www.indecs.org/
14. RDF: Image Metadata, 2001 http://swordfish.rdfweb.org/discovery/2001/03/demos/image.html
15. Semantic Web Activity: Resource Description Framework (RDF), 2001 http://www.w3.org/RDF Proc. Int’l. Conf. on Dublin Core and Metadata Applications 2001 169
16. W. Y. Arms, Digital libraries. Cambridge, Ma.: MIT Press, 2000.
17. T. Baker, “A Grammar of Dublin Core,” D-Lib Magazine, 6 (10), 2000.

18. D. Bearman, G. Rust, S. Weibel, E. Miller, and J. Trant, “A Common Model to Support Interoperable Metadata. Progress report on reconciling metadata requirements from the Dublin Core and INDECS/DOI Communities,” D-Lib Magazine, 5 (January 1999), 1999.
19. D. Bearman and J. Trant, “ Electronic Records Research Working Meeting May 28-30, 1997, A Report from the Archives Community,” D-Lib Magazine, 1997.
20. G. Booch, J. Rumbaugh, and I. Jacobson, The unified modeling language user guide. Reading Mass.: Addison-Wesley, 1999.
21. D. Brickley and R. V. Guha, “Resource Description Framework (RDF) Schema Specification,” World Wide Web Consortium, W3C Candidate Recommendation CR-rdf-schema-20000327, March 27 2000. http://www.w3.org/TR/rdf-schema
22. D. Brickley, J. Hunter, and C. Lagoze, “ABC: A Logical Model for Metadata Interoperability,” Harmony Project, Working Paper 1999. http://www.ilrt.bris.ac.uk/discovery/harmony/docs/abc/abc_draft.html
23. M. Doerr and J. Hunter, DELOS Working Group on Ontology Harmonization: Full report from the first meeting at CNR Rome, 2001 http://metadata.net/harmony/Rome_Full_Rep_V2.doc
24. M. Gorman, The concise AACR2, 1988 revision. Chicago: American Library Association, 1989.
25. J. Hunter, “MetaNet - A Metadata Term Thesaurus to Enable Semantic Interoperability Between Metadata Domains,” Journal of Digital Information, 1 (8), 2001.
26. J. Hunter and D. James, “Application of an Event Aware Metadata Model to an Online Oral History Archive,” presented at ECDL 2000, Lisbon, 2000.
27. J. Hunter and C. Lagoze, “Combining RDF and XML Schemas to Enhance Interoperability Between Metadata Application Profiles,” presented at WWW10, Hong Kong, 2001.

28. ICOM/CIDOC Documentation Standards Group, CIDOC Conceptual Reference Model, 1998 http://www.ville-ge.ch/musinfo/cidoc/oomodel/
29. C. Lagoze, “Business Unusual; How "event awareness" may breathe life into the catalog,” presented at Bicentennial Conference on Bibliographic Control in the New Millennium, Library of Congress, Washington DC, 2000.
30. C. Lagoze, “Keeping Dublin Core Simple: Cross Domain Discovery or Resource Description?,” D-Lib Magazine, 7 (1), 2001.
31. C. Lagoze, J. Hunter, and D. Brickley, “An EventAware Model for Metadata Interoperability,” presented at ECDL 2000, Lisbon, 2000.
32. O. Lassila and R. R. Swick, “Resource Description Framework: (RDF) Model and Syntax Specification,” World Wide Web Consortium, W3C Proposed Recommendation PR-rdf-syntax19990105, January 1999. http://www.w3.org/TR/PRrdf-syntax/
33. J. McCarthy, “Programs with common sense,” in Semantic Information Procession, M. Minsky, Ed.: MIT Press, 1968, pp. 403-418.
34. L. Miller, RDF querying using Squish, 2000 http://swordfish.rdfweb.org/rdfquery/
35. J. Perkins, ABC/Harmony CIMI Collaboration Project Description, 2000 http://www.cimi.org/public_docs/Harmony_long_desc.html.
36. J. L. Peterson, Petri net theory and the modeling of systems. Englewood Cliffs N.J.: Prentice-Hall, 1981.
37. J. F. Sowa, Conceptual Graphs, 2001 http://www.bestweb.net/~sowa/cg/.

БІБЛІОГРАФІЧНА ДОВІДКА

Тема дипломної роботи бакалавра: "Імплементація онтологічних моделей в модальних та діалогових формах "

Обсяг пояснювальної записки: 68 аркушів

Дата закінчення дипломної роботи 13 червня 2022р.

Підпис студента _____