Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

Інститут електроенергетики
(інститут)

Факультет інформаційних технологій
(факультет)

Кафедра Програмного забезпечення комп'ютерних систем
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи ступеня

магістра
(назва освітньо-кваліфікаційного рівня)

студентки	*Черечечі Ірини Володимирівни*
	(ПІБ)
академічної групи	*121М-20-1*
	(шифр)
спеціальності	*121 Інженерія програмного забезпечення*
	(код і назва спеціальності)
освітньої програми	*«Інженерія програмного забезпечення»*
	(назва освітньої програми)
на тему:	*Обґрунтування моделі представлення знань за результатами когнітивного моделювання у системі дистанційної освіти*

_____*І.В. Черечеча*

Керівники	**Прізвище, ініціали**	**Оцінка за шкалою**		**Підпис**
		рейтинговою	**інституційною**	
розділів кваліфікаційної роботи				
спеціальний	*Проф. Алексєєв М.О.*			
економічний	*Проф. Вагонова О.Г.*			
Рецензент	*Проф. Коротенко Г.М.*			
Нормоконтролер	*Доц. Приходченко С.Д.*			

Дніпро
2022

Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

ЗАТВЕРДЖЕНО:
Завідувач кафедри
Програмного забезпечення комп'ютерних систем
(повна назва)

_____ І.М. Удовик
(підпис) (прізвище, ініціали)

« » _____ 20 21 року

ЗАВДАННЯ
на виконання кваліфікаційної роботи

спеціальності *121 Інженерія програмного забезпечення*
(код і назва спеціальності)

студентці *121М-20-1* *Черечечі Ірині Володимирівні*
(група) (прізвище та ініціали)

Тема кваліфікаційної роботи *Обґрунтування моделі представлення знань за результатами когнітивного моделювання у системі дистанційної освіти*

1 ПІДСТАВИ ДЛЯ ПРОВЕДЕННЯ РОБОТИ

Наказ ректора НТУ «Дніпровська політехніка» від 10.12.2021 р. № 1036-с

2 МЕТА ТА ВИХІДНІ ДАНІ ДЛЯ ПРОВЕДЕННЯ РОБІТ

Об'єкт досліджень – процес пізнання про знання студента у освітній системі.

Предмет досліджень – методи представлення знань в освітній системі.

Мета НДР – обрати та обґрунтувати найбільш зручну та економну модель для представлення знань в системі дистанційної освіти.

Вихідні дані для проведення роботи – теоретичні та експериментальні дослідження представлення знань, описи інформаційних процесів у системі онлайн освіти.

3 ОЧІКУВАНІ НАУКОВІ РЕЗУЛЬТАТИ

Новизна запропонованих рішень визначається тим, що було обрано та обґрунтовано модель представлення знань за результатами когнітивного моделювання.

Практична цінність результатів полягає у тому, що запропонована у роботі модель дозволяє зберігати та зручно аналізувати знання студента з метою визначення порядку видачі навчального матеріалу для вдосконалення його поточних знань.

4 ВИМОГИ ДО РЕЗУЛЬТАТІВ ВИКОНАННЯ РОБОТИ

Результати досліджень мають містити опис та обґрунтування представлення знань в системах дистанційної освіти.

5 ЕТАПИ ВИКОНАННЯ РОБІТ

Найменування етапів робіт	**Строки виконання робіт (початок – кінець**
Аналіз теми та постановка задачі	12.09.2021-26.09.2021
Знання та їх представлення у комп’ютерних системах	27.09.2021-31.10.2021
Обґрунтування обраного підходу до зберігання знань та розробка тестової онтології	01.11.2021-31.12.2021

6 РЕАЛІЗАЦІЯ РЕЗУЛЬТАТІВ ТА ЕФЕКТИВНІСТЬ

Соціальний ефект від реалізації результатів роботи очікується позитивним завдяки підвищенню якості освітніх сервісів та вивільненні часу викладачів від рутинного тренінгу задля забезпечення їм можливості створення нових більш якісних освітніх матеріалів та вдосконалення створених раніше.

7 ДОДАТКОВІ ВИМОГИ

Завдання видав _____ *Алексєєв М.О.*
(підпис) (прізвище, ініціали)

Завдання прийняла до виконання _____ *Черечеча І.В.*
(підпис) (прізвище, ініціали)

Дата видачі завдання: *12.09.2021 р.*

Термін подання кваліфікаційної роботи до ЕК *20.01.2022*

РЕФЕРАТ

Пояснювальна записка: 111 стор., 17 рис., 5 таблиць, 4 додатки, 63 джерела.

Об'єкт дослідження: процес пізнання про знання студента у освітній системі.

Предмет дослідження: методи представлення знань в освітній системі.

Мета роботи: обрати та обґрунтувати найбільш зручну та економну модель для представлення знань в системі дистанційної освіти.

Методи дослідження. Для вирішення поставлених задач використані наступні методи: аналізу даних, теорії свідомості, теорії когнітивних процесів, методів та засобів штучного інтелекту, алгоритмів та структур даних.

Новизна отриманих результатів визначається тим, що було обрано та обґрунтовано модель представлення знань за результатами когнітивного моделювання.

Практична цінність результатів полягає у тому, що запропонована у роботі модель дозволяє зберігати та зручно аналізувати знання студента з метою визначення порядку видачі навчального матеріалу для вдосконалення його поточних знань.

Область застосування. Запропонований підхід для представлення знань може застосовуватися для зберігання знань студента в будь-якому освітньому сервісі.

Значення роботи та висновки. Запропоновай підхід можна використати у освітніх системах при проектуванні частин для зберігання знань студента. Обрана модель представлення знань дозволяє зберігати та аналізувати знання студента з прив’язкою до часу, що надає можливість аналізувати його прогрес або регрес та за потреби корегувати видачу навчального матеріалу та методику навчання.

Прогнози щодо розвитку досліджень. Необхідно розробити аналітичні механізми автоматичного поповнення онтологій на основі результатів тестування.

У економічному розділі були проведені розрахунки трудомісткості розробки програмного забезпечення, витрат на створення програмного забезпечення і тривалості його розробки.

Список ключових слів: епістемологія, знання, модель представлення знань, пам’ять, система дистанційної освіти, онтологія, когнітивні процеси, запам’ятовування, графи, OWL.

ABSTRACT

Explanatory note: 111 pages, 17 figures, 5 tables, 4 applications, 63 sources.

Object of research is the process of learning about student knowledge in the educational system.

Subject of research is methods of knowledge representation in the educational system.

Purpose of Master's thesis is to choose and explain the most convenient and economical model for the presentation of the knowledge in the system of online education.

Research methods. To solve existing problems used such methods as data analysis, theory of consciousness, theories of cognitive processes, methods and tools of artificial intelligence, algorithms and data structures.

Originality of research is determined by the fact that the model of knowledge representation based on the results of cognitive modeling was chosen and substantiated.

The practical value of the results is that the proposed in the paper model allows to store and conveniently analyze the student's knowledge in order to determine the order of issuance of educational material to improve student's current knowledge.

Scope of application. The proposed approach to the knowledge representation can be used to store student knowledge in any educational service.

The value of the work and conclusions. The proposed approach can be used in educational systems when designing parts to store student knowledge. The chosen model of knowledge presentation allows to store and analyze the student's knowledge with reference to time, which provides an opportunity to analyze his progress or regression and, if necessary, adjust the issuance of educational material and teaching methods.

Research forecast and development. It is necessary to develop analytical mechanisms for automatic addition information to ontologies based on test results.

In the Economics section we calculated complexity of software development, the cost of creating software and the duration of its development, as well as the impossibility of marketing research.

List of key words: epistemology, knowledge, model of knowledge representation, memory, online educational system, ontology, cognitive processes, remembering, graphs, OWL.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

API – Application Programming Interface;

DAML – DARPA Agent Markup Language;

DARPA – Defense Advanced Research Projects Agency;

KIF – Knowledge Interchange Format;

OIL – Ontology Interchange Language;

OWL – Web Ontology Language;

OWL DL – Web Ontology Language, Description Logic;

RDF – Resource Description Framework;

RDFS – Resource Description Framework Schema;

SHOE – Simple HTML Ontology Extensions;

SWRL – Semantic Web Rule Language;

URI – Uniform Resource Identifier;

W3C – World Wide Web Consortium;

XML – Extensible Markup Language;

БЗ – база знань;

ЗВО – заклад вищої освіти.

ЗМІСТ

ВСТУП

Задача вдосконалення освітнього процесу ніколи не втрачала своєї актуальності. Навіть на сьогодні в освітній галузі є багато проблем, які можна було б вирішити, використовуючи комп’ютерні технології та існуючі надбання науки.

Прикладом таких нереалізованих можливостей є формування відбитка знань студента в освітній інформаційній системі. Відображення знань студента в системі надає багато можливостей для відслідковування прогресу студента, аналізу даних, визначення порядку видачі навчальних матеріалів тощо. Маючи в системі представлення знань студента, можна автоматично підлаштовувати навчальну програму індивідуально під кожного окремого студента, забезпечуючи тим самим ефективне навчання: студентам не буде занадто легко, або занадто складно – нові матеріали будуть відповідати їх рівню підготовки. Персоналізація навчальної програми «вручну» викладачем забирає невиправдано багато часу, саме тому навчальні програми зазвичай орієнтуються на деякого «середнього» студента і не враховують індивідуальні особливості окремих студентів.

Крім того, застосування когнітивних моделей до процесу формування відбитка знань може відіграти ключову роль при визначенні рівня підготовленості студента. Когнітивне моделювання – це обчислювальна модель, яка ґрунтується на психологічних уявленнях, демонструючи, як люди вирішують проблеми та виконують завдання. Така модель будується на основі когнітивних навичок людини: їх відсутність або слабкий розвиток можуть бути причиною невдач студента у вивченні певних тем, тому це треба враховувати при підготовці навчального матеріалу та в процесі навчання.

Але одночасно з усіма перевагами підходу до відображення знань студента у системі постає проблема вибору прийнятної моделі представлення знань для ефективної роботи системи.

В цій кваліфікаційній роботі вирішується задача вибору моделі представлення знань за результатами когнітивного моделювання в системі дистанційної освіти. На жаль сучасні дослідження мало присвячені вивченню такого підходу.

В роботі будуть розглянуті існуючі моделі, їх переваги та недоліки, особливості застосування та буде обрана прийнятна модель для представлення знань студента.

Актуальність теми обумовлена потребою в оптимізації освітнього процесу шляхом перекладання рутинних задач викладача на комп'ютерну систему з метою звільнення часу викладача для створення або вдосконалення існуючого навчального матеріалу.

Мета дослідження - вибір та обґрунтування найбільш зручної та економної моделі для представлення знань в системі дистанційної освіти.

Об'єктом дослідження є процес пізнання про знання студента у освітній системі.

Предметом дослідження є підхід до представлення знань в освітній системі.

Методи дослідження. Для вирішення поставлених задач використані методи аналізу даних, теорії свідомості когнітивних процесів, методів та засобів штучного інтелекту, алгоритмів та структур даних.

Новизна отриманих результатів визначається тим, що було обрано та обґрунтовано модель представлення знань за результатами когнітивного моделювання.

Практичне значення роботи полягає у тому, що запропонована у роботі модель дозволяє зберігати та зручно аналізувати знання студента з метою визначення порядку видачі навчального матеріалу для вдосконалення його поточних знань.

Особистий внесок автора. В цій роботі була обрана та обґрунтована модель представлення знань в освітньому онлайн сервісі з урахуванням їх способу збирання, зберігання та аналізу.

Структура та обсяг дипломної роботи. Кваліфікаційна робота складається з вступу, чотирьох розділів, висновку та 4 додатків, а також містить 111 сторінок тексту, 17 рисунків, 5 таблиць, список використаних джерел.

У першому розділі розглянуто поточний стан предметної області, а також описано постановку задачі. У другому розділі описано поняття знань, принципи роботи пам’яті та особливості представлення знань в комп’ютерних системах. Третій розділ містить у собі обґрунтування обраного підходу до зберігання знань та розробку тестової онтології. У четвертому розділі були проведені розрахунки трудомісткості розробки програмного забезпечення, витрат на створення програмного забезпечення і тривалості його розробки.

РОЗДІЛ 1
АНАЛІЗ ТЕМИ ТА ПОСТАНОВКА ЗАДАЧІ

1.1. Загальні відомості з предметної галузі

Історично склалося так, що рік за роком, знання та вміння передаються з одного покоління до іншого. І на сьогодні освіта – це невід’ємна частина життя кожної людини. Людина може навчатися протягом всього життя – починаючи від садка, школи, ЗВО, закінчуючи курсами різноманітної тематики, як для професійного зростання так і у якості хобі.

В наш час існує чимало освітніх закладів, які використовують різні підходи до навчання своїх студентів, але питання якості освіти залишається відкритим. Це відбувається через те, що освіта, як правило, розрахована на деякого «середнього» студента та не враховує індивідуальні особливості кожного студента. Сильним студентам може стати відверто нудно на таких заняттях, а слабкі студенти будуть завжди відчувати стрес, бо завдання не будуть відповідати їх рівню підготовки.

У роботі [1] з психології автор розглядає теорію про стан покоту (Flow Theory). Ключовою концепцією цієї теорії є визначення та опис «зони» (тобто стану потоку) як ситуації, коли людина настільки зайнята та зосереджена на певному завданні, що повністю занурена в нього. Відповідно до теорії потоку для досягнення стану потоку повинні бути дотримані наступні умови:

– Чіткі цілі, в яких чітко простежуються очікування та правила.

– Прямий і негайний зворотний зв'язок, у якому є очевидні успіхи та невдачі завдання, щоб поведінку можна було скорегувати за потреби.

– Хороший баланс між рівнем здібностей і завданням.

Таким чином, автор роботи пропонує трьохканальну модель теорії потоку, показану на рис 1. Коли завдання дуже складне для вирішення, це викликає занепокоєння, оскільки сприймається як занадто складне або тому що рівень кваліфікації людини недостатній для вирішення завдання. Так само, коли

завдання занадто легке, воно викликає нудьгу, оскільки воно недостатньо складне, або тому що рівень кваліфікації людини занадто високий для виконання завдання.

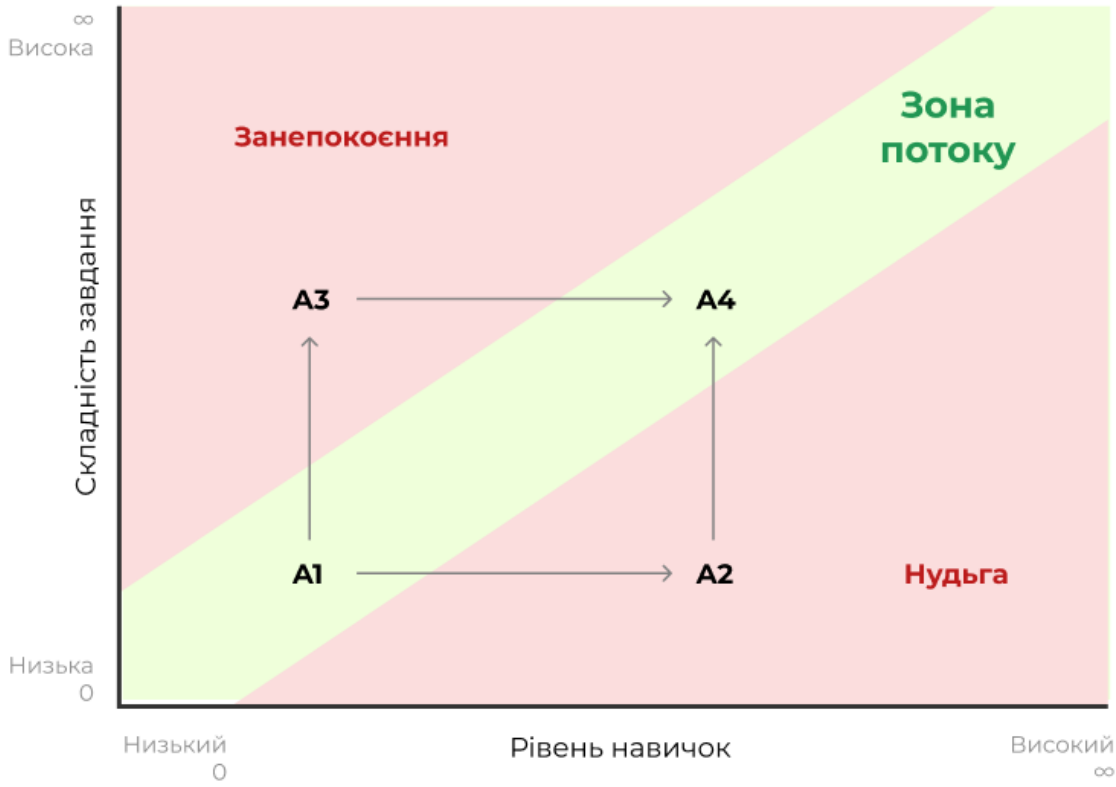

Рис. 1.1. Стан студента з точки зору рівня навичок та рівня складності завдання відповідно до моделі потоку

Тобто і тривога, і нудьга доводять людей до розчарування в тому, чим вони займаються та що вивчають, що може негативно вплинути на ефективність навчання. Тому дуже важливо враховувати індивідуальні особливості студента в освітньому процесі.

У роботі [2] автори запропонували власну модель, що базується на описаній теорії потоку. На базі даної моделі вони створили додаток, що допомагає викладачам правильно підбирати задачі для студентів згідно з теорію потоку.

В умовах пандемії освітня галузь отримала новий виток розвитку – впровадження дистанційної освіти. Хоча цей крок був вимушеним, але він був добре сприйнятий багатьма людьми, бо дистанційна освіта має читало переваг: від економії часу та фінансів на дорогу до освітнього закладу до цілодобового доступу до навчальних матеріалів будь-де та будь-коли.

Більш того, «оцифрування» освітніх процесів надає багато нових можливостей для впровадження нових ефективних підходів до навчання.

Зокрема, автоматизація освітнього процесу надає можливості для реалізації індивідуального підходу до кожного студента, навіть при великій кількості студентів у групі, що було б дуже складно або майже неможливо при звичайній схемі «1 викладач – декілька десятків студентів».

Дану задачу можна було б вирішувати різними способами, але в даній роботі пропонується концепція когнітивного моделювання. Завдяки збору та аналізу даних, отриманих з освітньої платформи, ми можемо виконувати моделювання когнітивної системи студента для того, щоб відобразити структуру представлення знань студента у пам’яті комп’ютера та працювати з цими даними, забезпечуючи індивідуальний підхід до навчання для кожного.

1.2. Історичний огляд процесу розвитку моделей представлення знань

Область дослідження представлення знань почала свій активний розвиток ще наприкінці 1940-х років. Основними концептами в той час були поняття синтаксису виразів або процедур.

Ще у 1948 році Едвард Толман запропонував когнітивні карти [3] як тип інтелектуального представлення, який дає людині можливість отримувати, інтерпретувати, запам'ятовувати, обмірковувати та передавати інформацію про своє оточення та використовувати її. Когнітивні карти створюються і видозмінюються в результаті активної взаємодії суб'єкта з навколишнім світом.

У 1956 році Акерс запропонував графічне рішення для цеху з двома робочими машинами, використовуючи продукційні правила [4]. Це уявлялося як достатня форма представлення процедурних знань.

Семантичний шаблон відповідності, запропонований у 1967 році, представляє знання у шаблонах взаємопов'язаних вузлів і дуг [5]. Цей метод було розроблено для впровадження штучного інтелекту та машинного перекладу в комп’ютерні системи.

Джон Ф. Сова запропонував концептуальні графи у 1976 як графічне представлення інформації [6]. Концептуальний граф – це формалізм представлення знань, що моделює семантику природної мови, по суті є дводольним орієнтованим графом, що має вершини двох типів: концепт (concept) і відношення концепта (conceptual relation) та дуги для з’єднання цих вершин.

Алан Кольмерое і Філіп Руссель винайшли мову Prolog (абревіатура від «PROgrammation en LOGique») у 1972 [7], використавши концепцію логіки першого порядку для представлення знань.

На початку 1980-х років представлення знань почало засновуватися на тому факті, що з'явилось поняття семантичної структури.

Гарднер досліджував помітні відмінності між здібностями окремих індивідів і запропонував теорію множинного інтелекту. Це призвело до винайдення Новаком і Говіном концептуальних карт (1984), щоб допомогти студентам навчитися вчитися [8]. Концептуальна карта (понятійна карта, діаграма концептів) – це діаграма, що відображає зв'язки між концептами (поняттями) деякої предметної області. Це графічний інструмент, який використовується інженерами знань, педагогічними дизайнерами, технічними письменниками та іншими, щоб структурувати знання у вибраній галузі.

У 1985 році Джуда Перл запропонував байєсівські мережі, які поєднали когнітивну науку, штучний інтелект і спрямовану графічну модель [9]. Байєсівська мережа – це графова імовірнісна модель, що представляє собою множину змінних і їх імовірнісних залежностей за Байєсом. Наприклад, байесівська мережа може бути використана для обчислення ймовірності того, на

що хворий пацієнт, за наявності або відсутності ряду симптомів, ґрунтуючись на даних про залежність між симптомами та хворобами.

Згодом, була винайдена семантична мережа (1987), яка фіксує знання як графічну структуру [10]. Вузли позначають поняття, об’єкти, події, час тощо. Дуги представляють такі відношення, як «є», «є екземпляром», «має», «частина», «значення», логічні (і, або, ні) та лінгвістичні відношення, такі як «подобається», «має» тощо.

У 1992 році Сова запропонував розширену роботу з об'єктно-орієнтованою концепцією семантичної мережі [11]. Це призвело до розвитку штучних нейронних мереж на основі онтологій. Онтологія – це сучасна техніка представлення знань, що базується на концептуалізації – абстрактному, спрощеному представленні деякої вибраної частини світу, що містить у собі об’єкти, концепти та інші сутності, які існують у деякій сфері та відношення між ними.

Згодом, у 1994 році Коско придумав нечіткі когнітивні карти (Fuzzy Cognitive Maps) як розширення когнітивних карт [12]. Нечіткі когнітивні карти – це структури з нечіткими графами для представлення причинно-наслідкових міркувань. Нечітка когнітивна карта являє собою когнітивну карту, в якій зв’язки між елементами (наприклад, концепціями, подіями, ресурсами) «ментального ландшафту» можуть використовуватися для обчислення «сили впливу» цих елементів.

Через десяток років у області представлення знань з’явились більш прагматичні моделі.

Наприклад, Тім Бернерс-Лі разом з колегами запропонував семантичну павутину у 2001 році як загальну структуру в дротовій мережі, що дозволяє обмінюватися та повторно використовувати інформацію [13].

У 2004 році робочою групою OWL була створена онтологічна веб-мова для забезпечення потреб комплексних знань про поняття в онтології [14].

Уже понад десять років граф знань (Knowledge Graph) (2012) є базою знань для розширеного пошуку, яку використовує Google. Семантичний пошук інформації збирає дані з великого джерела до баз даних знань і графів [15].

Аналогічно, Facebook прийняв технологію пошуку в графах для асоціації друзів для різних користувачів. Таким чином користувач може знаходити своїх друзів і встановлювати нові зв'язки [16, 17].

GraphQL і Ontotext працювали разом, вирішуючи проблему зменшення складності представлення кожної сутності в структурі графа [18]. Через динамічну зміну інформації та оскільки кожен елемент представлений як сутність разом зі своїми властивостями, процеси розміщення та отримання інформації стають більш складними.

AnzoGraph, розроблений Cambridge Semantics, в основному розроблений для подолання такого розриву та спрощення процесу за допомогою RDF (Resource Description Framework) [18]. RDF є стандартною моделлю для обміну даними в Інтернеті.

W3C запропонував стандартизувати дані графа в 2019 році [19], щоб об'єднати спільноту графів, таких як Neo4j, Oracle, Ontotext і ArangoDB. Був зроблений фокус на розробці потужних графічних баз даних.

Формалізований опис основних подій у області дослідження моделей представлення знань наведено у таблиці 1.1.

Таблиця 1.1

Огляд моделей представлення знань

Автор(и)	**Рік**	**Модель**	**Опис**
Гентцен	1935	Пропозиційна логіка	Логічний висновок випливає з однієї або декількох пропозицій
Акерс, Фрідман	1955	Продукційні правила	Форма представлення знань людини у вигляді речення типу «ЯКЩО (умова), ТО (дія)».

Продовження таблиці 1.1

Автор(и)	**Рік**	**Модель**	**Опис**
Лотфі Заде	1965	Нечітка логіка	Багатозначна логіка, станами якої можуть бути не тільки 1 та 0, але й значення між ними (0.3, 0.26 тощо).
Квіліан	1967	Семантичні мережі	Семантична мережа - інформаційна модель предметної області, що має вигляд орієнтованого графа, вершини якого відповідають об'єктам предметної області, а дуги (ребра) задають відносини між ними.
Джон Генрі Холанд	1970	Генетичний алгоритм	Використовується для знаходження евристичних рішень для проблеми, що розв’язується
Ньювел, Саймон	1972	Процедурні знання (ієрархічне представлення)	Використовують системи, засновані на правилах; знання про те, як виконати конкретну навичку або завдання
Джері Фодорс	1975	Гіпотеза мови думки (LOTH)	Думки виражаються через символи та логіку
Мінський	1975	Фрейми	Являє собою стереотипну ситуацію, що поєднує декларативні знання і структуровані процедурні знання.

Продовження таблиці 1.1

Автор(и)	**Рік**	**Модель**	**Опис**
Шенк	1975	Скрипти	Структурована схема представлення знань, що описує стереотипічну послідовність подій в певному контексті
Шенк	1982	Модель динамічної пам’яті	Пояснює вищі аспекти сприйняття
Джон Сова	1984	Концептуальні графи	Поєднали концепти з їх зв’язками
Ліпп	1984	Нечіткі мережі Петрі	Формалізм, який моделює експертні системи, що містять нечіткі дані
Шастрі	1988	Нейронні мережі	Широко використовується в аналізі даних, розпізнаванні шаблонів, системі виробництва та плануванні
Пачульські та ін.	2000	Мережі та павутина знань	Співвідносить інформацію з різними концептами
Тім Бернерс Лі	2001	Семантична павутина	Фреймворк для повторного використання даних
Пател Шнейдер	2005	Дескрипційна логіка (описова логіка)	Описує визначення та правила для міркування
Гауч	2007	Онтології	Категоризація груп та їх зв’язків
MetaWeb	2007	Freebase	Велика колаборативна база знань, що містить метадані, зібрані, в основному, інтернет-спільнотою

Продовження таблиці 1.1

Автор(и)	**Рік**	**Модель**	**Опис**
Calais	2008	OpenCalais	Семантичні метадані, інтеграція веб-сервісу Calais Thomson Reuters через Refinitiv Intelligent Tagging API у платформу Drupal
MS	2009	Satori	Версія цифрової допомоги
CMU	2010	NELL (Never-Ending Language Learning)	Система семантичного машинного навчання, розроблена дослідницькою групою з Університету Карнегі-Меллона та за підтримки грантів DARPA, Google, NSF та CNPq, частина системи працює на суперкомп'ютерному кластері від Yahoo!
MSRA	2011	Probase	Граф відношення-сутність
Google (Freebase)	2012	Граф знань	Семантична технологія та база знань, використовувана Google для підвищення якості своєї пошукової системи із семантично-пошуковою інформацією, зібраною з різних джерел
Facebook	2013	Facebook Graph Search	Семантична пошукова система
Google	2015	LPG (Labeled Property Graph)	Модель графа з мітками властивостей представлена набором вузлів, зв'язків, властивостей і міток.

Продовження таблиці 1.1

Автор(и)	Рік	Модель	Опис
Cambridge Semantics	2017	AnzoGraph	База даних з горизонтально масштабованими графами, створена для онлайн-аналітики та гармонізації даних.
Stardog	2017	GraphQL	GraphQL – це мова запитів для API-інтерфейсів та середовище, в якому вони виконуються.
W3C	2019	Neo4j	Графова система управління базами даних з відкритим вихідним кодом, реалізована Java.

1.3. Когнітивне моделювання

Когнітивне моделювання - це створення моделей, що наближенні до когнітивних процесів тварин (переважно людей) з метою розуміння цих процесів та їх передбачення [20]. Когнітивні здатні можуть прогнозувати, як взаємодіють декілька аспектів або змінних, як ці аспекти або змінні формують поведінку. Когнітивні моделі допомагають зрозуміти, які взаємопов’язані когнітивні процеси призводять до певного поведінкового результату або до світоглядних змін.

1.4. Представлення знань

Слово «знання» може здатись простим, але його визначення та форма відрізняються у різних предметних областях.

У філософії вивчення знань називається гносеологію, у цій області можна знайти найперші спроби дати визначення терміну «знання». Найбільш

поширений варіант запропонував Томас Гобс у 1651: «знання є доказом істини» [21].

Визначення Гоббса засноване на традиційному арістотелівському погляді на ідеї. Такий підхід відомий як репрезентативна теорія розуму (Representational Theory of the Mind). Навіть на сьогоднішній день репрезентативна теорія розуму продовжує використовуватися в більшості робіт у когнітивній науці [22].

Психологія не така давня наука, як філософія, але все ще має безпосереднє відношення до знань, особливо ті розділи психології, які вивчають процес навчання. У психології за допомогою методів більш емпіричного характеру була розроблена величезна кількість теорій для розуміння та інтерпретації поведінки людини щодо знань. Серед найбільш відомих теорій є конекціоністські теорії, когнітивні теорії та конструктивістські теорії.

Теорії коннекціоністів розглядають теорію розуму, яка передбачає спосіб отримання та представлення інформації людськими знаннями. Коннекціоністські теорії стверджують, що знання можна описати як ряд взаємопов’язаних понять, де кожне поняття пов’язане з іншим через певні асоціації. Коннекціоністські теорії є основою семантики як засобу представлення знань [23].

Конструктивістські теорії стверджують, що знання та значення створені людьми з досвідом та ідеями. Конструктивістські теорії розглядають більш складні чинники міркування, такі як причинність, ймовірність і контекст. Саме тому більшість конструктивістських теорій доповнюють коннекціоністські, стверджуючи, що кожна група асоціацій інтегрує різні шари мислення, де різниця на кожному рівні полягає в силі асоціацій. У результаті найвищим шаром є концепт, тобто організована і стійка структура знання, а нижній шар – це нещільно пов’язані купи ідей [24].

Існує також група теорій, які вивчають поведінку людини, відомі як біхевіористські теорії. Ці теорії справили сильний вплив на психологію загалом і сприйняття навчання людьми. У своїй класичній формі біхевіористи не розглядають внутрішній когнітивний процес, а приділяють увагу лише

зовнішній поведінці, поведінковим реакціям на певні стимули. Відповідно до цього біхевіористські теорії не можуть пояснити процес мислення [25].

Когнітивна наука зосередилася на моделюванні та перевірці попередніх теорій майже з усіх інших наук, таких як нейронаука, психологія, біологія, штучний інтелект тощо. Через це когнітивну науку вважають ідеальною спільною основою, благодатним полем для створення нових теорій або для формалізації існуючих за допомогою комп'ютерних моделей [26]. Знання в цій галузі зазвичай описують за допомогою рівнянь, математичних співвідношень та комп'ютерних моделей.

Знання не є чимось відразу доступним людському мозку. Знання – це річ, яка має бути сконструйована. Чи то у людському мозку, чи то в пам'яті обчислювальної машини. Знання у загальному випадку - це про те, як певний індивід розуміє концепт.

Проблема здобуття, збереження та вилучення знань є першочерговою у завданні побудови інтелектуальної системи. Людина – це створіння, яке краще за інших розуміє, обґрунтовує та інтерпретує знання. Але задача полягає в тому, як знання, набуті людиною, правильно та достовірно відобразити за допомогою комп'ютера. Саме за вирішення такої проблеми відповідає така галузь як представлення знань та міркування ("knowledge representation and reasoning").

Представлення знань – це складова частина галузі штучного інтелекту, яка пов'язана з мисленням агентів штучного інтелекту та з дослідженням того, який вклад мислення дає для розумної поведінки агентів [27].

Представлення знань відповідає за перекладання знань людини про навколишній світ у пам'ять комп'ютера у такому вигляді, щоб комп'ютер міг розуміти та використовувати ці знання для розв'язання задач. Такі знання цілком можуть бути використані для вирішення багатьох проблем реального світу, наприклад: діагностика захворювань, спілкування з людьми природною мовою, розуміння змісту текстів, формування уявлення про явища та процеси, навчання тощо.

Важливою особливістю представлення знань є те, що це не просто зберігання даних у якійсь базі даних. Представлення знань надає машині можливість вчитися на цих знаннях і досвіді, щоб вона могла поводитися розумно, як людина. Після того, як знання отримані, вони повинні бути збережені в базі знань для подальшого використання.

База знань (БЗ) зазвичай складається з фактів, які є істинними за визначенням. Структура фактів може бути різною та залежить від обраного підходу. Механізми висновку напряму залежать від обраної для збереження знань структури.

Дослідження представлення знань спрямоване на пошук оптимальних засобів опису та структурування знань, які мають полегшити завдання створення та обґрунтування бази знань.

Галузь представлення знань в загальному випадку визначає ряд основних понять, що потребують відображення:

– Об’єкт – це набір фактів про речі, які ми можемо спостерігати навколо.

– Події – це різноманітні дії, що відбуваються навколо.

– Виконання – опис поведінки, яка включає в себе знання про те як виконатись якусь дію чи процес.

– Мета-знання – знання про знання, відомості про те, що ми знаємо.

– Факти – правдиві висловлення про реальний світ.

– База знань – центральний компонент агентів, що базуються на знаннях, сховище знань.

1.5. Типи знань

Знання – це усвідомлення ситуацій або фактів, або близьке знайомство з чим-небудь, набуте за допомогою певного досвіду.

Виділяють п’ять основних типів знань: декларативні, процедурні, мета-знання, евристичні знання, структурні знання.

Декларативні знання – це знання, що описують певну сутність, тобто знання про факти, концепти, об’єкти. Даний тип знань також називають описовим.

Процедурні знання – це знання, що описують як виконати певну дію. Такі знання також називають імперативними. Вони включать правила, стратегії, процедури тощо.

Мета-знання – це знання про інші типи знань. По суті набір службової інформації про знання.

Евристичні знання – це знання певних експертів про їх предметну область. Такі знання включать в себе правила, що базуються на попередньому досвіді, усвідомлення підходів.

Структурні знання – це знання, що описують відношення між різними концептами. Такими відношеннями можуть бути: «вид чогось», «частина чогось», «група чогось» тощо.

1.6. Моделі представлення знань

Мета розуміння і визначення того, що являють собою знання і які види знань існують, полягає в тому, щоб дозволити людям використовувати їх у штучних системах. Особливо це стало популярним у наш час, коли людство прагне розробити інтелектуальні технології, які дозволяють комп’ютерам виконувати складні завдання, чи то для допомоги людям, чи то тому, що люди не можуть їх виконувати.

Представлення знань глибоко пов’язане з процесами навчання та міркування, як стверджує Крісп Брайт, визначаючи знання як «психологічний результат сприйняття, навчання та міркування» [28].

Моделі – це представлення деяких теорій, які дозволяють нам проводити моделювання та тести, що відображатимуть результати, передбачені теорією.

Тому, коли говорять про моделі представлення знань, мається на увазі особливий спосіб представлення знань, який дозволить передбачити те, що система знає і на що здатна за допомогою знань і механізмів міркування (висновку).

Інтелект включає в себе такі когнітивні процеси, як навчання, розуміння та міркування. Ці процеси вимагають наявність знань, за допомогою яких можна було б управляти цими когнітивними процесами. Як стверджує когнітивна інформатика, якщо потрібні комп'ютери з когнітивними можливостями [29], то потрібні комп'ютеризовані моделі представлення знань.

В основному виділяють наступні головні способи представлення знань: логічне представлення, семантичні мережі, фрейми, скрипти, продукційні правила та онтології.

1.6.1. Логічна модель представлення знань

Логічне представлення базується на принципах логіки. Тобто знання, необхідні для вирішення поставленої задачі, і сама задача описуються певними твердженнями на логічній мові. Використовується спеціальна мова з чіткими правилами, в яких визначені недвозначні (тобто такі, які трактуються однозначно) вирази (propositions).

Знання являють собою множину аксіом, а задача, що вирішується, по суті є теоремою, яка вимагає доведення. Процес доведення теореми і становить логічну модель подання знань.

Логічне представлення дозволяє робити висновки, які засновані на деяких умовах. Таке представлення визначає деякі обов'язкові правила.

По-перше, воно складається з точно визначеного синтаксису та семантики, яка забезпечує обґрунтований висновок.

По-друге, кожне речення можна перевести в логіку за допомогою синтаксису та семантики [33].

Синтаксис - це правила, завдяки яким вирішується як конструювати вирази, які символи можна використовувати та як їх писати.

Семантика – це правила, завдяки яким можна інтерпретувати даний вираз у логіку, семантика відповідає за привласнення значення до кожного виразу.

Логічну модель можна задати сукупністю:

$$M = < T, P, A, F >, \quad (1.1)$$

де T – множина базових елементів, P – множина правил, A – множина аксіом, F – правило висновку.

Логічне представлення може брати за основу різні типи логіки: логіку висловлень (пропозиційна логіка), логіку предикатів (логіка першого порядку), логіки вищих порядків.

Логіка висловлень - розділ символічної логіки, що вивчає необхідні відношення між висловлюваннями, на підставі чого визначають значення істинності висловлювань; дедуктивна теорія, яка моделює процес виведення одних висловлювань з інших за принципом логічного слідування. [30]

Логіка предикатів (логіка першого порядку) - це сукупність формальних систем, які складаються з формальної мови для опису висловлювань і дедуктивної системи для міркування [31].

Логіка першого порядку є стандартом для формалізації математики в аксіоми, а також відіграє вирішальну роль у представленні знань практично будь-якого роду.

Логіка першого порядку з'явилась як наступник пропозиційної логіки, яка не володіє достатньою виразністю, щоб представити знання реального світу. Основна відмінність між логікою першого порядку та пропозиційною логікою полягає в тому, що логіка першого порядку, як і природна мова, передбачає, що світ містить об'єкти, відношення та функції, тоді як пропозиційна логіка оперує лише фактами. Логіка першого порядку корисна для демонстрації логічних зв’язків та їх аргументації. Крім того, вона дозволяє розбивати логічні висловлювання на слова. Тому логіку предикатів трактують як розширення логіки висловлювань через те, що в ній введені нові терміни та системи аксіом.

Для прикладу розглянемо два логічні твердження «Декарт – математик» і «Піфагор – математик». У пропозиційній логіці ці твердження розглядаються як непов’язані, тоді як у логіці предикатів нам дозволяється ввести предикат «Математик (х)», який створює зв’язок між явними висловлюваннями.

Міркування в логіці першого порядку здійснюється в рамках дедуктивної системи, яка використовується, щоб показати, що одна формула є логічним наслідком іншої.

Як приклад побудови та виведення знань мовою логіки предикатів досить навести відомий силогізм Сократа:

– «Усі люди – смертні» (основний засновок),

– «Сократ – людина» (другорядний засновок),

– «Сократ – смертний» (висновок).

Логіки вищого порядку - форми предикатної логіки, які відрізняються від логіки першого порядку додатковими предикатами над предикатами, кванторами над ними, і, відповідно, багатші на семантику. Логіки вищого порядку з їх стандартними семантиками виразніші, але їх модельно-теоретичні властивості значно складніші для вивчення та застосування в порівнянні з логікою першого порядку.

Перевагами логічного представлення є:

– За допомогою такого представлення можна моделювати логічні міркування.

– Цей вид представлення дуже зручний, оскільки саме його використовують як основу мови програмування.

Основними недоліками мови логічного представлення є:

– Обмежена виразність, оскільки існує безліч фактів та взаємозв'язків, які важко чи навіть неможливо висловити засобами математичної логіки. Наприклад, такий логічний з погляду людини висновок, як «Людина коле дрова сокирою, сокира – гостра, отже людині колоти дрова легко», виразити мовою

логіки предикатів неможливо, оскільки містить так званий сценарний, а чи не логічний висновок.

– Ця техніка не є природньою, тому висновки можуть бути не дуже ефективними.

1.6.2. Семантичні мережі

Семантичні мережі за своєю сутністю є альтернативою логіки предикатів для представлення знань, але вони дотримуються більш природного підходу до представлення знань. Такі мережі засновані на традиційній репрезентативній теорії розуму та асоціативних теоріях.

Семантична мережа – це спосіб представлення знань, що використовує нотацію графів щоб описати структуру знань [33]. Така мережа складається з двох основних елементів: вузлів, що відображають об’єкти, та дуг, що відображають зв’язки між об’єктами.

Формально семантичну мережу можна виразити так:

$$<I, R_1, R_2, \ldots, R_n, M> \quad (1.2)$$

де I – інформаційні об’єкти,

$R_1, R_2, \ldots, R_n$ – типи зв’язків між інформаційними об’єктами,

M – карта, яка встановлює зв'язки між інформаційними сутностями.

Приклад простої семантичної мережі, що відображає невеликий фрагмент знань зображений на рис. 1.2.

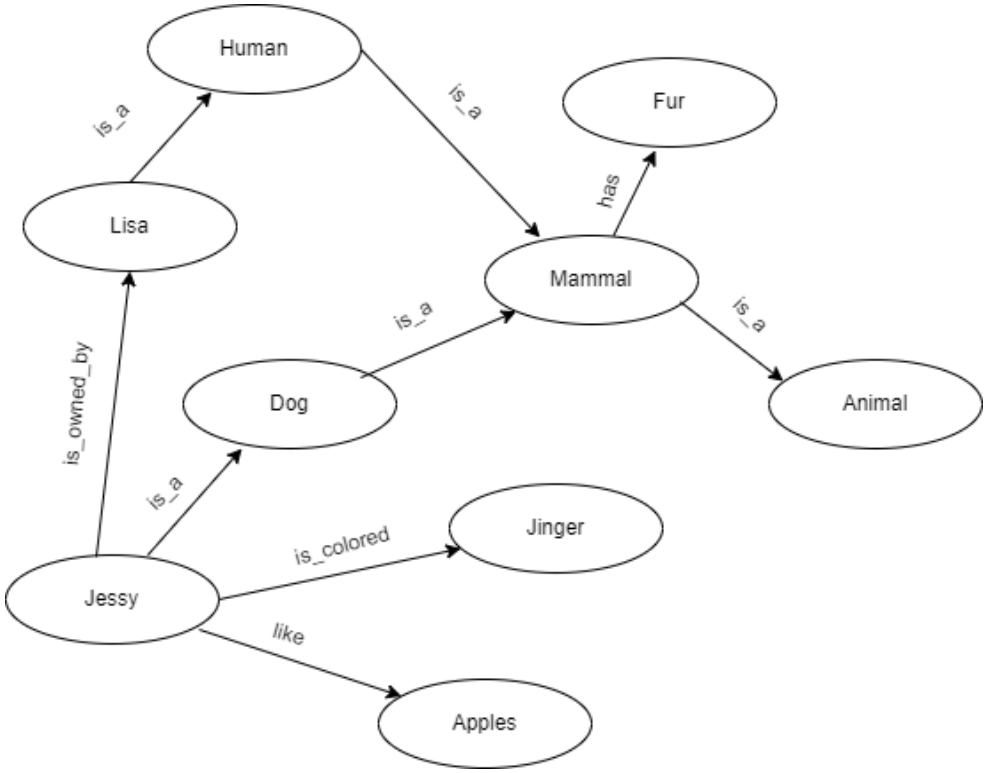

Рис. 1.2. Семантична мережа

Вирази, або знання, на базі яких буда побудована дана мережа приведені нижче:

– Джессі є собакою.
– Джессі рудого кольору.
– Джессі любить яблука.
– Власник Джессі – Ліза.
– Ліза є людиною.
– Собака є ссавцем.
– Людина є ссавцем.
– Ссавці мають волосяний покрив.
– Ссавці є тваринами.

Розрізняють декілька видів семантичних мереж:

– Мережі визначення. Використовуються для опису понять, об’єктів і зв’язків між ними. Вони активно використовують відношення типів «є», «має»,

«екземпляр», щоб сформувати таксономію знань, яка є важливою для механізму міркування.

– Мережа тверджень. Графічно представляє логічні висловлення, тому її елементи виконують додаткові семантичні ролі, які відповідають логічним операторам, таким як екзистенційний квантор, кон'юнкція, диз'юнкція та інші.

– Імплікаційні мережі. Є окремим випадком пропозиційної семантичної мережі, яка використовує імплікацію як первинне відношення для з'єднання вузлів; інші відношення можуть бути вкладені всередину пропозиційних вузлів, але вони ігноруються процедурами висновку.

– Виконувані мережі. Містять механізми, які можуть викликати деякі зміни в самій мережі. Найпоширенішими виконуваними механізмами є передача повідомлень, прикріплені процедури та перетворення графів.

– Навчальні мережі. Коригують свої внутрішні уявлення у відповідь на нову інформацію, щоб ефективно працювати у своєму оточенні. Навчальну мережу можна налаштувати трьома способами: запам’ятовування, зміна ваг і реструктуризація.

– Гібридні мережі. Одночасно включають кілька типів семантичних мереж, щоб повторно використовувати переваги та усунути недоліки вищезгаданих мереж.

– Перевагами семантичних мереж є наступне:

– Семантичні мережі представляють знання у природному вигляді.

– Семантичні мережі передають значення прозоро.

– Дуже прості для розуміння людиною через графічне представлення знань. Також людям легше виявити закономірності та зв’язки, які було б складно або неможливо виявити у лінійній формі.

– Такі мережі можна досить легко розширювати за необхідності.

– Графічна нотація показує прямі зв’язки між спорідненими поняттями, тоді як лінійна нотація повинна покладатися на повторювані входження змінних чи імен, щоб показати ті самі зв’язки.

Недоліками семантичних мереж є:

– Обхід семантичної мережі та пошук необхідного вузла може займати досить багато обчислювального часу. У найгіршому варіанті можна пройти все дерево та не знайти шукане значення.

– Семантичні мережі у своїй основі намагаються моделювати людську пам’ять для збереження інформації, але на практиці неможливо точно відтворити таку величезну мережу.

– Відсутність стандартизації значень вузлів і дуг, що створює складність обслуговування.

– Деякі типи семантичних мереж страждають від специфічних проблем, таких як помилки множинного успадкування, відношення один до багатьох, відсутність формальної семантики, змішування різних рівнів абстракції тощо.

Тим не менш, «природній» підхід до представлення знань робить семантичні мережі широко використовуваною технікою в розробці систем знань.

1.6.3. Фрейми

В психології та філософії широко відоме поняття абстрактного образу. Як приклад, слово «будівля» у людини зазвичай викликає такі асоціації – «споруда, що має стіни, вікна, двері та дах, можливо, декілька поверхів». З даного опису нічого не можна прибрати, але в ньому є пропуски, так звані «слоти». Наприклад, колір стін, кількість вікон, матеріал даху, кількість поверхів тощо. Слоти – це незаповнені значення деяких атрибутів. Слоти можуть бути будь-якого типу та розміру. Також вони мають імена та значення, які називаються фасетами, або аспектами.

Згідно з цією теорією фреймом називається саме цей абстрактний образ.

Тобто, фрейм – це об’єктно-орієнтована структура, подібна до запису, яка складається з набору атрибутів та їх значень для опису певної сутності [33].

Моделі на основі фреймів використовують об’єктно-орієнтований підхід, коли знання структуруються навколо поняття «об’єкт». Наявність «об’єкта» дозволяє групувати пов’язані властивості та процедури, отже, спрощує опис і управління знаннями. У результаті його часто називають представленням «об’єкт-властивість-значення».

У логіці першого порядку висловлювання про світ розкидані, тому важко контролювати послідовність і повноту зафіксованих знань. Об’єктно-орієнтованого представлення можна досягти шляхом об’єднання багатьох властивостей об’єкта одного типу в одну структуру.

За характером об’єкта опису розрізняють два типи фреймів – індивідуальні та загальні. Індивідуальний фрейм представляє один об'єкт, який є екземпляром деякої категорії, тоді як загальний фрейм використовується для опису абстрактної категорії. Отже, індивідуальні фрейми за замовчуванням мають спеціальний слот, який називається “INSTANCE-OF”, де заповнювачем є ім’я (або вказівник на відповідний загальний фрейм). У той же час загальні фрейми можуть мати спеціальний слот під назвою «IS-A» з наповнювачем, який вказує на більш загальний кадр. Наявність цих спеціальних слотів має вирішальне значення, оскільки вони забезпечують механізм успадкування. Приклад зображений на рис. 1.3.

Фрейми у початковій своїй формі походять із семантичних мереж. Але з часом фрейми еволюціонували в сучасні класи та об’єкти.

Ідея фрейму полягає в тому, щоб безпосередньо наслідувати людську пам’ять, яка зберігає ситуації, в яких поєднуються процедурні та декларативні знання. Коли ми опиняємося в ситуації, подібній до тієї, яку ми проживали раніше, ми згадуємо стереотип, збережений у нашій пам’яті, щоб знати, як реагувати на цю нову ситуацію. Ця теорія є спробою об’єднати кілька інших підходів, запропонованих психологією, лінгвістикою та штучним інтелектом.

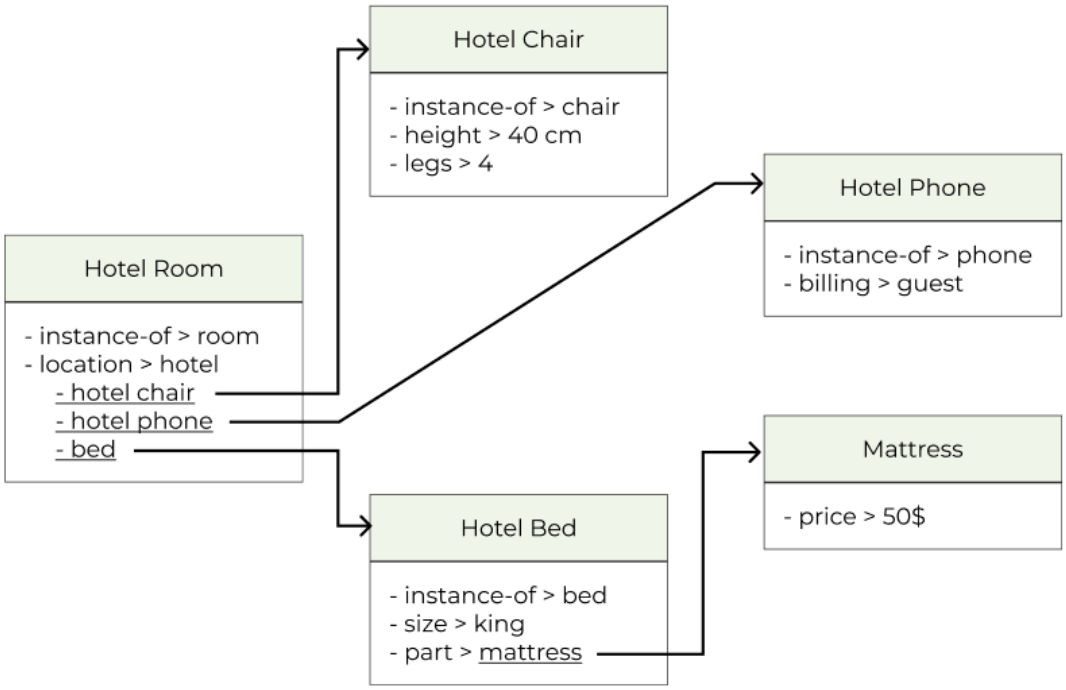

Рис. 1.3. Приклад фрейму

Варто зазначити, що один фрейм сам по собі не дуже корисний та не несе значного смислового навантаження. Частіше використовується система фреймів. Системи фреймів складаються з набору фреймів, які з'єднані між собою.

У форматі фреймів знання про об’єкт або подію можуть зберігатися разом у базі знань.

Технологія фреймів широко використовується в різних програмах, включаючи обробку природної мови та машинний зір.

Приклад фрейму зображений на рис. 1.4. Знання, що відображені у фреймі, словесно можна описати так: існує людина на ім’я Хелен, вона працює графічним дизайнером, її вік – 26 років, вона живе у м. Гріндейл, у США.

Slots	Filter
Name	Helen
Profession	Graphical designer
Age	26
Town	Grindale
Coutry	USA

Рис. 1.4. Приклад фрейму

Перевагами фреймів є:

– Така форма представлення знань робить програмування легшим, через те, що пов’язані дані вже згруповані.

– Фрейми досить гнучкі та їх можна легко розширювати, додаючи нові слоти та зв’язки, легко задавати значення за замовчуванням та шукати відсутні значення.

– Фрейми легко розуміти та візуалізовувати.

– Об’єктно-центрований підхід дозволяє групувати властивості та процедури, пов’язані з об’єктами одного типу, спрощуючи процеси висновку та підтримки узгодженості.

– Значення слотів, які вказують на інші фрейми, роблять фрейми неявно асоційованими один з одним, таким чином зберігаючи пов’язані знання в кластері.

– Крім того, процедури роблять похідні (обчислені) властивості явними і можуть уникнути потенційно непотрібних обчислень.

– Фрейми дозволяють як декларативний, так і процедурний контроль процесу висновку, таким чином зменшуючи загальні зусилля, необхідні для створення системи.

– Елегантна реалізація механізму успадкування підкреслює виразність підходу на основі фрейму.

Серед недоліків фреймів варто зазначити:

– Відсутність формального механізму аргументації (виводу).

– Такий підхід іноді занадто узагальнений.

– Відсутність стандартів значень слотів.

– Залежність від користувача - фрейм, що представляє один і той же об’єкт, може значно відрізнятися для двох різних користувачів.

1.6.4. Скрипти

Також іноді виділяється особливий вид фреймів – скрипти.

Скрипт (або сценарій) – це структурована схема представлення знань, що описує стереотипічну послідовність подій в певному контексті [33].

Подія відповідає дії, що залежить від часу, яка може вказувати на причинно-наслідкові зв’язки. Скрипти дуже схожі на фрейми, за винятком того, що значення, які заповнюють слоти, мають бути впорядковані; проте фрейми вважаються більш загальною технікою представлення знань.

Скрипти дуже корисні, тому що події в реальному світі дотримуються стереотипних моделей, оскільки люди використовують попередній досвід, щоб зрозуміти нові ситуації.

Щоб описати послідовність подій, скрипт використовує серію слотів, що містять інформацію про виділені елементи, які утворюють контекст або беруть участь у подіях. Зазвичай скрипт має назву і складається з таких слотів:

– Трек використовується для представлення варіацій певного скрипту.

– Ролі описують людей, які беруть участь у скрипті.

– Параметри відносяться до об’єктів, які використовуються в послідовності подій.

– Вхідні умови визначають умови, які повинні бути задоволені, перш ніж події в цьому скрипті можуть відбутися або стануть дійсними.

– Сцени описують реальну послідовність подій, що відбуваються.

– Результати стосуються умов, які існують після того, як події в скрипті відбулися.

Приклад скрипту представлений на рис. 1.5.

Скрипт:	Похід у ресторан
Трек:	Звичайний ресторан
Параметри:	Їжа Столи Меню Гроші
Ролі:	Власник Клієнт Офіціант Касир
Вхідні умови:	
Клієнт голодний Клієнт має гроші Власник має їжу	
Результати:	Клієнт не голодний Власник має більше грошей Клієнт має менше грошей Власник має менше їжі

Сцена 1:	"Вхід у ресторан" Клієнт заходить до ресторану Переглядає столи Обирає найкращий Вирішує сісти за обраний стіл Крокує до столу Займає місце
Сцена 2:	"Замовлення їжі" Клієнт просить меню Офіціант приносить меню Клієнт переглядає меню Клієнт обирає що поїсти Клієнт замовляє обрану їжу
Сцена 3:	"Прийом їжі" Офіціант приносить їжу Клієнт оглядає їжу Клієнт їсть
Сцена 4:	"Оплата рахунку" Клієнт просить рахунок Офіціант приносить рахунок Клієнт оплачує рахунок Офіціант здає готівку касиру Офіціант приносить здачу Клієнт залишає чайові Клієнт виходить з ресторану

Рис. 1.5. Приклад скрипту

Знання, які представлені скриптом, зазвичай зберігаються в символічній формі. Цього можна легко досягти за допомогою LISP або будь-якої іншої символічної мови.

Основна функціональність скрипту – відповідати на запитання, пов'язані з ролями, об'єктами та результатами його виконання в контексті заданої ситуації. Процес міркування в скрипті полягає у застосуванні методів пошуку та узгодження шаблонів, які перевіряють сценарій на наявність відповідей. Незважаючи на те, що деякі ситуації не спостерігалися, це дозволяє передбачити,

що з ким і коли станеться. Після запуску скрипту користувач може задавати запитання та отримувати точні похідні відповіді з невеликими знаннями або без них. Ця особливість підкреслює силу методів стереотипного представлення знань.

Підсумовуючи, можна виділити основні переваги скриптового представлення:

– Скрипт дозволяє побудувати єдину узгоджену інтерпретацію з сукупності спостережень.

– Такий вид представлення забезпечує високий ступінь гнучкості і здатність передбачати події.

– Стереотипність скриптів робить їх зручними для застосування до різноманітних ситуацій реального світу.

– Основними недоліками є:

– Складність у відображенні складних понять і відносин.

– Не всі знання можна представити у вигляді скрипту.

– Іноді важко вибрати точну структуру скрипту для досягнення оптимальної продуктивності міркування. Як уже згадувалося, вони менш загальні, ніж фрейми, і їх можна розглядати як частковий випадок фреймів.

– Скрипт надто залежить від уявлень людини про схематизацію тих чи інших ситуацій, тому цілком можливе хибне або недоречне застосування тих чи інших скриптів для розв’язання проблем.

1.6.5. Продукційна модель представлення знань

Продукційні правила складаються з пар (умова-дія): «якщо умова, тоді дія». Системи на базі продукційних правил мають зазвичай три частини: набір правил, робоча пам’ять та цикл розпізнавання-дії [33].

Принцип роботи цієї техніки такий: агент перевіряє умову, та якщо вона існує, тоді спрацьовує правило та виконується відповідна дія. Умовна частина

правила визначає, яке правило можна застосувати до проблеми. І частина дії виконує відповідні кроки вирішення проблеми. Повний процес називається циклом розпізнавання і дії.

Робоча пам’ять складається з опису поточного стану рішення проблеми. Правило може записувати знання у робочу пам’ять. Ці знання можуть запускати інші правила.

Системи, засновані на правилах, часто використовуються, коли присутні процедурні знання. Системи правил також можуть використовуватися для декларативних знань, як правило, з метою класифікації, наприклад, «якщо він гавкає, то це собака, в іншому випадку – не собака».

Найбільш відомою реалізацією цього підходу є експертні системи.

Приклад продукційних правил:

```
IF (working time AND hungry) THEN action (go to cafe)
IF (in the cafe AND hungry AND empty table) THEN action (sit down)
IF (in the cafe AND hungry AND sit at the table) THEN action (make order)
IF (in the cafe AND order is ready) THEN action (eat)
IF (in the cafe AND order is ready AND NOT hungry AND order is unpaid) THEN action (pay)
IF (in the cafe AND NOT hungry AND order is paid) THEN action (get out of the cafe)
```

Перевагами продукційних правил є наступне:

- Вони виражені природною мовою
- Їх можна легко змінювати (редагувати, видаляти, додавати тощо).

Недоліками даного підходу є наступне:

- У даній системі не має жодних можливостей навчання, оскільки система не зберігає результат задачі для подальшого використання.

- Також під час виконання програми багато правил можуть бути активними, тому виробничі системи, засновані на правилах, іноді можуть бути неефективні.

1.6.6. Онтології

Онтологія як техніка представлення знань поєднує в собі вищезгадані моделі представлення знань і має багато активних реалізацій, зокрема Protege, Cyc, Dublin Core тощо.

Онтологія – це сучасна техніка представлення знань, що базується на концептуалізації [33]. Під концептуалізацією мається на увазі абстрактне, спрощене представлення деякої вибраної частини світу, що містить у собі об’єкти, концепти та інші сутності, що існують у деякій сфері та відносини між ними. Концептуалізація може бути реалізована декількома шляхами, у тому числі і за допомогою онтологій. До того ж, широко використовується таке визначення онтології як «явна специфікація концептуалізації».

Формально онтологія складається з концептів, організованих у таксономію, їх параметрів (атрибутів), пов’язаних аксіом та правил висновку. Онтологія разом з набором індивідуальних екземплярів класу складає базу знань.

Елементами, що входять у стандартну нотацію онтології є:

- Клас (концепт). Відповідає за концепт у предметній області.
- Екземпляр (індивідуал). Представляє індивідуальний екземпляр певного класу.
- Слот (властивість, роль). Описує особливість або атрибут концепта чи екземляра.
- Фасет (обмеження ролі). Представляє обмеження слота.
- Відношення. Описують спосіб у який класи та екземпляри можуть бути взаємопов’язані.
- Правила. Вирази типу «якщо-то», які описують можливі логічні висновки з твердження.

- Аксіоми. Це твердження в логічній формі, що містять загальну теорію предметної області.

Загалом, інженерія онтологій включає в себе використання всіх чотирьох основних моделей представлення знань: фрейми використовуються для оголошення знань, семантичні мережі використовуються для визначення відношень, аксіоми визначені з використанням логіки другого порядку, а продукційні системи представляють правила висновку.

Процес побудови онтологій загалом складається з чотирьох етапів:

1. Визначення класів онтології.
2. Організація класів у таксономію.
3. Визначення слотів та опис допустимих значень для них.
4. Заповнення слотів значеннями в екземплярах.

Онтології мають широке практичне застосування в світі.

По перше, вони слугують механізмом для спільного розуміння структури інформації. Особа або програмний агент може отримати доступ до опублікованих онтологій, щоб повторно використовувати наявні знання для побудови інтегрованої, розширеної або спеціальної онтології.

По-друге, онтології можна використовувати для міркувань і виробництва нових знань на основі висновків даних міркувань. Ці дії зазвичай реалізуються відповідно до функціонального інтерфейсу «розкажи і запитай», де клієнт взаємодіє з системою знань, роблячи логічні твердження (розкажи) і ставлячи запити (запитай).

Нарешті, онтології володіють деякими корисними функціями, такими як чітке визначення припущень предметної області, можливість повторного використання знань тощо.

Побудова онтології з метою вирішення реальної проблеми вимагає значної кількості ресурсів – експертів предметної області, технічних інженерів, часу. Оскільки розробка онтології є ітераційним процесом, тягар обслуговування також неминучий. Можна було б отримати користь від повторного використання існуючих онтологій, але це не усуває потреби в участі людини, яка все ще може

бути величезною. Тому має сенс шукати автоматизовані або напівавтоматичні методи створення та підтримки онтології, які б звели необхідні ресурси до мінімуму.

Полегшення побудови онтологій та наповнення їх прикладами як концепцій, так і відношень зазвичай називають навчанням онтологій. Існує кілька типів навчання онтології. Найпопулярнішою вважається онтологія, що вивчає текст; вона створює легкі таксономії за допомогою аналізу тексту та вилучення інформації. Також досліджуються інші типи, такі як пов'язаний аналіз даних, вивчення концепції в логіки опису та OWL, онтології краудсорсингу.

Ще одне важливе завдання, яке постає під час навчання онтології, – це оцінка якості онтології. На практиці якість засвоєної онтології часто залежить від ступеня автоматизації. Зазвичай більша участь людини в генерації напівавтоматичної онтології призводить до вищої якості. Якщо онтологія недостатня для передбачуваного застосування з точки зору якості, вона зрештою вимагатиме значного обсягу постобробки.

Переваги онтологій:

- Онтології надають засоби для представлення будь-яких форматів даних, включаючи неструктуровані, напівструктуровані або структуровані дані, що забезпечує більш гладку інтеграцію даних, легший аналіз концепцій і текстів, а також аналітику на основі даних.
- Забезпечують контекст для певного поняття.
- Дозволяють повторно використовувати дані.
- Забезпечують спільне розуміння структури інформації між людьми або програмними агентами.
- Дозволяють автоматизувати міркування про дані.
- Функціонують як «мозок». Вони «працюють і міркують» з поняттями та відношеннями в спосіб, близький до того, як люди сприймають взаємопов'язані поняття.
- Легко розширити, оскільки відношення та узгодження концепцій легко додати до існуючих онтологій.

Недоліки онтологій:

- Неповна достовірність даних, їх заплутаність і навіть суперечливість (зазвичай для онтологій, створених автоматично).
- Створені вручну онтології можуть мати зайву кількість зв'язків між поняттями, деякі поняття можуть бути практично не висвітлені.
- Остання версія мови веб-онтологій – OWL2 має дещо обмежений набір конструкцій властивостей.

1.7. Постановка задачі

Завданням цієї кваліфікаційної роботи є обґрунтування моделі представлення знань за результатами когнітивного моделювання у системі дистанційної освіти.

Під час виконання поставленого завдання необхідно дослідити особливості процесу когнітивного моделювання, способи зберігання знань в освітніх системах, засоби їхнього опису та моделі їхнього застосування на практиці.

На підставі результатів досліджень необхідно обґрунтувати добір способу представлення знань в системі дистанційної освіти.

1.8. Висновок до розділу

Кожен підхід, що був розглянутий в даному розділі має свої переваги та обмеження, які слід оцінити перед впровадженням інтелектуальних систем, заснованих на знаннях.

У випадку системи дистанційної освіти та представлення у ній знань студента найбільш прийнятною буде онтологічна модель представлення знань через те, що вона частково поєднує у собі всі інші моделі представлення знань, є більш виразною серед них, гнучкою та легко розширюваною.

РОЗДІЛ 2
ЗНАННЯ ТА ЇХ ПРЕДСТАВЛЕННЯ У КОМП'ЮТЕРНИХ СИСТЕМАХ

2.1. Когнітивні процеси

Когнітивні навички є однією з найбільш детально досліджуваних тем у різних галузях, а особливо у галузі представлення знань. Розуміння роботи мозку вкрай важливо для побудови комп'ютерних моделей у цій області, які допоможуть точно представляти знання та ефективно працювати з ними.

Когнітивні навички (функції мозку) – це навички, які ґрунтуються на мозковій діяльності, вони необхідні для отримання знань, маніпулювання інформацією та міркування для вирішення будь-яких завдань [34]. Когнітивні навички охоплюють області сприйняття, уваги, пам'яті, навчання, прийняття рішень і мовлення.

Когнітивні навички, які іноді називають загальним інтелектом, мають важливе значення для адаптації та виживання людини. Вони включають здатність міркувати, планувати, вирішувати проблеми, мислити абстрактно, розуміти складні ідеї, швидко навчатися та вчитися на досвіді [34]. Крім запам'ятовування або наслідування, інтелект підтримує здатність розуміти ситуації, з'ясовувати, що потрібно, і планувати подальші дії.

Наприклад, відповіді на телефонні дзвінки включають сприйняття (чути мелодію дзвінка), прийняття рішення (відповідати чи ні), моторні навички (підняття слухавки), мовні навички (розмовляти та розуміти мову), соціальні навички (інтерпретувати тон голосу та правильно взаємодіяти з іншою людиною).

У статтях [35-37] автори виділяють наступні основні когнітивні навички:

– Сприйняття – це розпізнавання та інтерпретація сенсорних подразників (нюх, дотик, слух тощо).

– Увага – здатність підтримувати концентрацію на певному об'єкті, дії

чи думці.

– Пам’ять – поділяється на короткочасну/робочу пам'ять (обмежена пам'ять) і довготривалу пам'ять (необмежене зберігання).

– Моторні навички – це здатність мобілізувати м’язи тіла, а також здатність маніпулювати оточуючими об’єктами.

– Мовлення – це навички, що дозволяють людині перекладати звуки в слова і створювати словесні результати.

– Візуальна та просторова обробка – це здатність обробляти вхідні візуальні стимули, розуміти просторові відносини між об’єктами, а також візуалізувати образи та сценарії.

– Виконавчі функції – це здібності, які забезпечують цілеспрямовану поведінку, наприклад здатність планувати та досягати цілі. До них належать:

– Гнучкість – здатність швидко переключатися на відповідний розумовий режим.

– Теорія розуму – проникнення у внутрішній світ інших людей, їхні плани, їхні симпатії та антипатії.

– Передбачення – передбачення на основі розпізнавання образів.

– Вирішення проблеми – визначення проблеми правильно, щоб потім генерувати рішення та вибрати правильне.

– Прийняття рішень – здатність приймати рішення на основі вирішення проблем, неповної інформації та емоцій (наших та чужих).

– Робоча пам'ять – здатність зберігати та маніпулювати інформацією «онлайн» в режимі реального часу.

– Емоційна саморегуляція – здатність визначати власні емоції та керувати ними для гарної роботи.

– Послідовність – здатність розбивати складні дії на керовані одиниці та розставляти їх у правильному порядку.

– Гальмування – здатність протистояти відволіканню і внутрішнім потягам.

Таким чином, когнітивні навички – це розумові здібності, які є основою для розумної діяльності людини, а також вони є життєво важливими для ефективного навчання студентів. Когнітивні навички доповнюють одна одну, щоб ефективно функціонувати та визначають успішність результатів навчання.

Наявність або відсутність, сильний чи слабкий розвиток тих чи інших когнітивних навичок напряму впливає на перспективи студента щодо якісного засвоєння певного типу навчальних матеріалів.

Важливим фактом є те, що когнітивні навички можна розвивати та підтримувати, оскільки деякі з віком можуть погіршуватись, і це можна враховувати у процесі навчання.

Тому оцінка когнітивних навичок студента є дуже важливою для створення ефективного індивідуального плану навчання, який забезпечить продуктивне засвоєння студентом навчального матеріалу.

В даній роботі вибір моделі представлення знань пропонується провести на основі результатів когнітивного моделювання. Когнітивне моделювання – це збір інформації про когнітивні процеси та представлення їх в комп'ютерному вигляді.

Для того, щоб мати змогу оцифрувати когнітивні процеси, потрібно отримати їх математичну модель. Для отримання такої моделі необхідно коректно класифікувати когнітивні навички. На сьогодні єдиної правильної класифікації когнітивних навичок не існує, різні автори пропонують свої варіанти, які мають багато подібних елементів, але все ж відрізняються.

Першою широко відомою класифікацією є таксономія Блума. Таксономія Блума – це сукупність трьох ієрархічних моделей, які використовуються для класифікації цілей навчання за рівнями складності та специфічності. Ці три моделі охоплюють навчальні цілі в трьох сферах: когнітивній, афективній та сенсорній. Модель класифікації когнітивних здібностей частіше за інші була у центрі уваги більшої частини традиційної освіти і часто використовувалась для структурування цілей навчальної програми та оцінювання. Автором таксономії є Бенджамін Блум. Робота була опублікована у 1956 році як своєрідна

класифікація результатів і цілей навчання [38]. Протягом більш ніж півстоліття відтоді таксономія Блума використовувалася в галузі освіти.

З часом деякі вчені проводили ревізію когнітивної таксономії Блума або складали свої системи класифікації [39-40].

Найбільш досконалою та широко використовуваною стала ревізія Андерсона та Кретвола 2001 року [41]. Так як Андерсон був учнем Блума, а Кретвол був його партнером, то вони, знаючи Блума особисто, змогли створити нову адаптацію таксономії, яка врахувала багато моментів, які непокоїли самого Блума в його моделі, а також врахували критику інших щодо його оригінальної таксономії.

Для моделювання в цій роботі була обрана саме ця класифікація. Вона найбільш точно та коректно відображає когнітивні навички студента в процесі навчання, бо по суті є вдосконаленою версією таксономії Блума, створеною за участі когнітивних психологів, теоретиків навчальної програми та дослідників освітнього процесу, а також спеціалістів з тестування та оцінювання. До того ж ця таксономія набагато краще алгоритмізується, що є суттєвою перевагою для використання її в системі дистанційної освіти.

2.2. Принципи роботи людської пам’яті

Не залежно від системи класифікації когнітивних навичок, які були коротко розглянуті у попередньому підрозділі, така навичка як пам’ять завжди займає в них ключову роль.

Людський мозок наділений здатністю фіксувати у собі нову інформацію. Цей процес життєво важливий для людини і дозволяє навчатися та базуватися на попередньому досвіді. Саме тому для створення моделі представлення знань студента у системі дистанційної освіти потрібно дослідити механізми роботи людської пам’яті.

Навчання – це здатність зберігати інформацію та використовувати її для того, щоб реагувати по-різному на різні ситуації.

Люди завжди прагнули пізнати механізми роботи мозку та зокрема пам’яті, але основні дослідження у цій області припадають на другу половину XX століття.

Перша найбільш повна та ширико використовувана модель пам’яті була запропонована стендфордськими вченими Аткінсоном та Шифріном у 1968 році [42], пізніше вона була вдосконалена Лонго [43] та ін. Сучасний вигляд цієї схеми представлений на рис. 2.1.

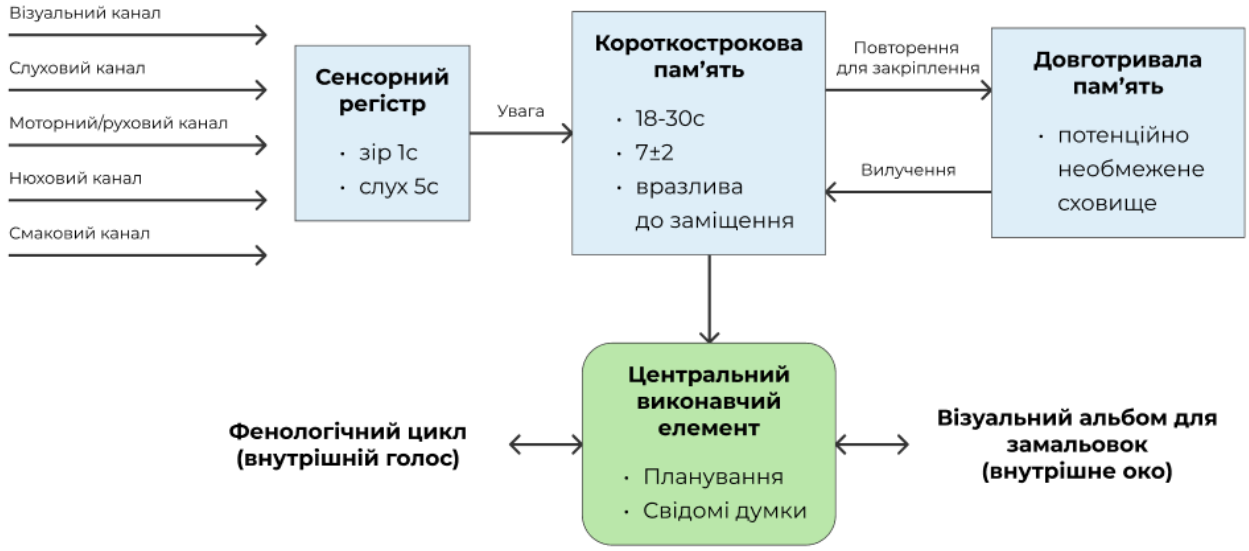

Рис. 2.1. Модель пам’яті

Усі люди мають певну кількість сенсорних входів, так званих каналів або способів сприйняття інформації, таких як: візуальний канал, слуховий канал, кінестетичний (моторний/руховий) канал, нюховий канал, смаковий канал тощо.

Сенсорний реєстр (буфер) – це перший етап, який проходить інформація на шляху до збереження у людській пам’яті. Якщо людина сприймає якусь інформацію, але не приділяє цьому уваги, то дана інформація зберігається у сенсорному реєстрі на протязі 1-5 секунд в залежності від її типу.

Поширений приклад, якщо якась людина, наприклад дитина, дивиться телевізор і її увага сфокусована на цьому процесі, та паралельно інша людина

говорить цій дитині якусь інформацію, то дитина не запам’ятає щойно сказане і через декілька хвилин вже не зможе нічого згадати, але якщо її перепитають відразу, то вона зможе достатньо точно переказати, про що їй говорили.

Але після перепитування увага дитини вже буде сфокусована не на телевізорі, а привернута до щойно сказаної інформації та ця інформація зможе пройти до наступного етапу – збереження в короткочасній пам’яті.

Збереження в короткочасній пам’яті відбувається тільки якщо людина звертає увагу на інформацію, що надходить через якийсь із сенсорних каналів. Інформація, збережена у короткочасній пам’яті знаходиться там приблизно 18-30 секунд. Вважається, що в середньому короткочасна пам’ять людини здатна утримувати в один момент часу 7±2 значень. У роботі [44] були проведені дослідження з цього приводу. Як приклад, для більшості людей не складно буде запам’ятати 7 цифр або 7 слів. Невеликим трюком для запам’ятовування більшої кількості інформації є її розбиття на групи. Так інформація запам’ятовується у короткочасну пам’ять значно легше.

Також важливим є те, що короткочасна пам’ять вразлива до заміщення даних. Нові дані можуть перезаписувати ті дані, що вже там зберігаються.

Якщо людині потрібно, щоб інформація не зникла безслідно, а перемістилась до довготривалої пам’яті, то повинні бути виконані певні дії – повторення, так звані «репетиції» для закріплення. Коли інформація потрапляє до довготривалої пам’яті, у мозку людини відбувається зміна зв’язків між нейронами, створюються нові патерни (шаблони) нейронів. Потенційно довготривала пам’ять не має обмежень, але вона має властивість погіршуватися з віком.

Інформацією, що збережена у довготривалій пам’яті, людина зазвичай користується. Для цього виконується операція вилучення інформації та вона потрапляє до короткострокової пам’яті, звідки потрапляє до центрального виконавчого елементу – відділів головного мозку вищого порядку, які управляють та запускають процеси використання інформації. Саме завдяки центральному виконавчому елементу людина здатна до усвідомленого

мислення, планування, виконання, організації тощо.

Центральний виконавчий елемент має дві частини:

– Фенологічний цикл (внутрішній голос, яким людина користується в процесі міркування).

– Візуальний альбом для замальовок (внутрішнє око, яким людина користується щоб уявляти певні образи).

У науці також існує поняття робочої пам’яті.

Але науковці сперечаються з цього приводу, бо деякі вважають, що робоча пам’ять – це те ж саме що й короткочасна пам’ять. Але деякі висувають думку, що принципи роботи робочої пам’яті ґрунтуються на використанні як короткострокової, так і довготривалої пам’яті для виконання розумової операції на обмеженій ємності сховища. Наприклад, виконати додавання двох щойно почутих цифр.

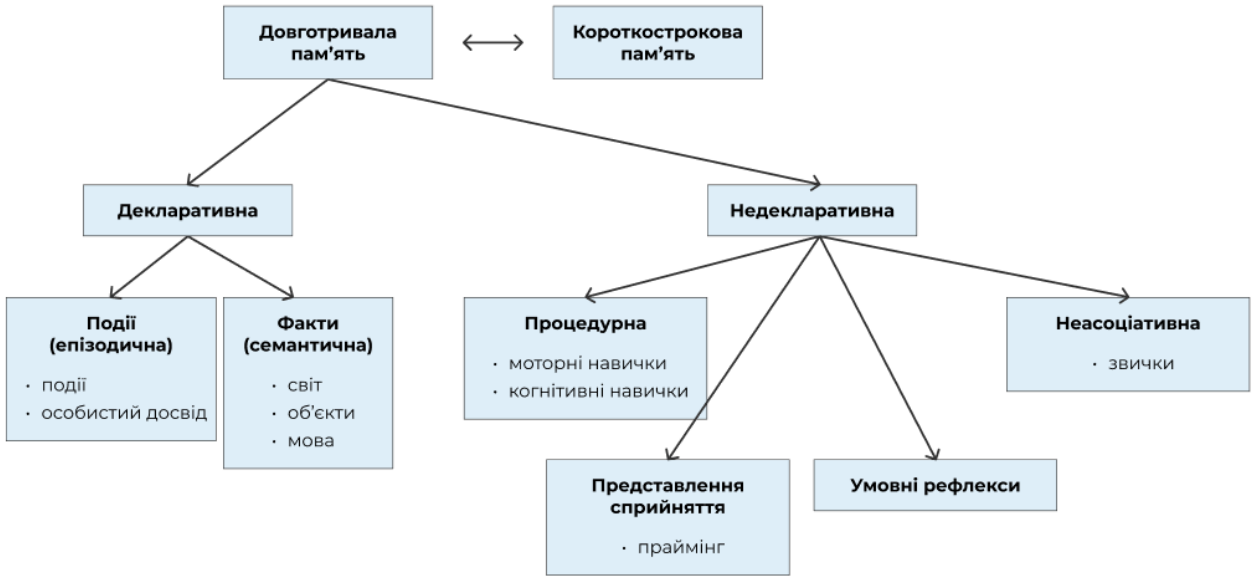

Рис. 2.2. Класифікація пам’яті

Довготривала пам’ять поділяється на дві основні категорії: декларативна та недекларативна.

Декларативна пам’ять – це свідомі спогади людини, про які вона може

свідомо думати, які може свідомо використовувати. Декларативна пам’ять включає в себе події (епізодична пам’ять) та факти (семантична пам’ять). Події – це спогади про речі, що відбувались в житті у людини за її участі, по суті – автобіографія життя. Факти – це певні знання про світ навколо. І хоча факти та події тісно переплетені між собою, але вони зберігаються у різних частинах мозку. Наприклад, людина може пам’ятати, коли сталася якась важлива подія, але не може згадати де була, або що робила у той час.

Недекларативна пам’ять – це пам’ять, яку людина не усвідомлює до кінця. Людина може вміти робити щось, але не до кінця усвідомлювати як це відбувається. До недекларативної пам’яті відносяться процедурна пам’ять, неасоціативна пам’ять, представлення сприйняття та умовні рефлекси.

Процедурна пам’ять включає в себе когнітивні та моторні навички.

Неасоціативна пам’ять проявляється в утворені логічно не пов’язаних зв’язків шляхом звикання до них.

Представлення сприйняття – це інша форма недекларативної пам’яті, являє собою систему пам'яті, функція якої полягає в ідентифікації об'єктів і слів, що дозволяє швидко розпізнавати стимули, які зустрічалися раніше. Сприйняття специфічно розпізнаються у формі, яку раніше відчували (наприклад, слово, яке бачать, або слово, яке почуто). Включає в себе поняття праймінгу. Праймінг (ефект попередження, фіксована установка) – у психології представляє механізм імпліцитної пам'яті, що забезпечує неусвідомлюваний і мимовільний вплив деякого стимулу на обробку наступних стимулів.

Яскравим прикладом умовних рефлексів є відома собака Павлова. Павлов проводив свій експеримент і спочатку подавав світловий сигнал та корм одночасно, а потім обмежувався подачею лише сигналу. Було помічено, що у собаки з часом починає виділятися слина навіть у тому випадку, коли корм після світлового сигналу вона не отримує. Отже, виділення слини та шлункового соку – це прояви умовного рефлексу чи реакції на той чи інший подразник, що надходить ззовні.

Як вже опосередковано згадувалося вище, коли говорять про основні

механізми, які пов’язані з пам’яттю, то виділяють наступні процеси:

– Фіксація.
– Зберігання.
– Вилучення.

При моделюванні структури пам’яті за допомогою комп’ютерних засобів саме ці поняття є ключовими: потрібно визначити як зафіксувати знання, у якій формі їх зберігати та у який спосіб відшукувати та вилучати необхідні частини. Механізм роботи цих процесів здебільшого визначається моделлю представлення, що була обрана.

2.3. Онтології як засіб представлення знань

Як було сказано у розділі 1, для представлення знань у розроблюваній системі дистанційної освіти було обрано представлення у вигляді онтологій.

У наш час існує багато визначень онтології. Здебільшого це залежить від завдання, для якого використовується онтологія. Класичне визначення, запозичене з філософії, говорить, що онтологія означає «систематичний опис існування».

Згідно Н. Гуаріно [45], «онтологія – це логічна теорія, що враховує передбачуваний зміст формального словника, тобто його онтологічну прихильність до конкретної концептуалізації світу». Виходячи з такого визначення, онтологія складається з понять, їх визначень, зв'язків між поняттями та обмежень, що виражаються у вигляді аксіом.

Деякі автори [46] визначають онтологію як «формальна, явна специфікація спільної концептуалізації». У цьому контексті концептуалізація відноситься до абстрактної моделі якогось явища у світі, наприклад бізнесу, яка визначає відповідні концепції цього явища, тобто бізнес-концепції. «Явна» означає, що тип використовуваних концепцій і обмеження на їх використання чітко визначені, а «формальна» означає, що онтологія повинна бути зрозумілою машині. «Спільна» відображає уявлення про те, що онтологія охоплює

консенсуальні знання – вона не обмежується певною особою, а приймається групою.

У контексті області представлення знань існує більш примітивне визначення: онтологія – це формальний опис знання як набору понять у межах предметної області та відношень, які існують між ними.

Однією з переваг онтологій є те, що вони не тільки вводять представлення знань для спільного та повторного використання, але також надають можливість додавати нові знання про предметну область. Онтологія дозволяє повторно використовувати знання в базі знань, забезпечуючи концептуалізацію, відображаючи припущення та вимоги, зроблені під час вирішення проблеми з використанням бази знань.

Онтології останнім часом стали дуже поширеною темою для досліджень у різних галузях. Онтології використовуються в штучному інтелекті, семантичній мережі, програмній інженерії, біомедичній інформатиці, бібліотекознавстві та інформаційній архітектурі як форма представлення знань про світ або його частину.

Вивченням процесу розробки, методології побудови і життєвого циклу онтологій, а також наборів інструментів і мов, які їх підтримують займається спеціальний розділ інженерії знань, що має назву інженерія онтологій. Онтологічна інженерія спрямована на забезпечення стандартних компонентів для побудови моделей знань. Найцікавіше в інженерії онтологій полягає в тому, що, незважаючи на існування багатьох методологій, інструментів і мов, вибір відповідної техніки розробки онтології – досить складна задача.

2.4. Мови представлення онтологій

Для програмування онтологій були створені спеціальні засоби, так звані, мови онтологій.

Онтологічна мова, або мова представлення онтологій – це формальна мова, що використовується для програмної побудови онтологій.

Онтологічні мови дозволяють кодувати знання про конкретні предметні області і часто включають правила міркування (висновку), які підтримують обробку цих знань. Мови онтологій зазвичай є декларативними мовами, майже завжди є узагальненнями фреймових мов і зазвичай базуються або на логіці першого порядку, або на логіці опису (дескрипційній логіці) [47].

Згідно з роботою [48], мови онтологій повинні відповідати наступним критеріям:

1. Достатньо компактний синтаксис.

2. Добре визначена семантика, щоб можна було точно сказати, що представляється.

3. Достатня виразна сила для репрезентації людського знання.

4. Ефективний, потужний і зрозумілий механізм міркування.

5. Має бути придатною для створення великих баз знань.

Більше того, мова онтологій повинна описувати поняття у такий спосіб, який може бути прочитаний машиною. Таким чином, мова онтологій повинна мати не тільки можливість визначати певний словник понять, а й засоби для формального визначення словника так, щоб з його допомогою можна було б виконати процес автоматизованого міркування [49].

Відповідно до [50], мови онтологій можна розділити на дві категорії. Перша категорія – це традиційні мови онтологій. До цієї категорії відносять:

– Мови, засновані на логіці предикатів першого порядку (KIF, CycL).

– Мови на основі фрейму (Ontolingua, F-logic і OCML).

– Мови на основі логіки опису (Loom) тощо.

Друга категорія – це веб-стандарти, які використовуються для полегшення обміну в Інтернеті, а мови онтологій, які сумісні з веб-стандартами, називаються мовами онтологій на основі веб.

Однак окремі мови можна віднести до обох категорій, тобто вони засновані на певному веб-стандарті та певній парадигмі, як фрейми або логіка предикатів першого порядку. Наприклад, OWL DL заснований на RDF і логіці опису. Описані категорії візуально представлені на рис. 2.3.

Рис. 2.3. Категорії онтологічних мов

В наступних підрозділах будуть описані найбільш поширені на сьогодні мови. В даній роботі були виділені наступні мови: KIF, RDF, OIL, DAML і OWL, SWRL.

2.4.1. KIF

Knowledge Interchange Format (KIF) – це комп’ютерно-орієнтована мова для обміну знаннями між різними програмами. Основою семантики KIF є концептуалізація світу в термінах об'єктів і відношень між цими об'єктами. Основні особливості KIF полягають у наступному [51]:

– Декларативна семантика. Можливо зрозуміти значення виразів, не звертаючись до інтерпретатора для маніпулювання цими виразами.

– Логічна вичерпність. Можна виразити довільні речення у формі предикатів першого порядку.

– Метазнання. Представлення знання про знання. Це дозволяє нам зробити представлення знань явним і вводити нові конструкції без зміни мови.

– Здатність до перекладання. Ця особливість забезпечує практичні засоби перекладу декларативних баз знань на типові мови представлення знань

та навпаки.

– Представлення правил немонотонного міркування.

– Читабельність. Хоча мова KIF не призначена для взаємодії з людьми, вона корисна для опису семантики мови репрезентації та допомоги людям у проблемах перекладу бази знань.

Спочатку KIF був створений у DARPA Knowledge Sharing Effort [52]. Він був розроблений як найсучасніший інструмент інтерлінгви (штучна міжнародна мова). Було кілька версій KIF і створено кілька мов, заснованих на KIF, наприклад SUOKIF.

Основними недоліками KIF є:

– Висока виразність ускладнює побудову повністю відповідних систем.

– Отримані системи мають тенденцію бути «важкими», тобто вони більші та в деяких випадках менш ефективні, ніж системи, які використовують більш обмежені мови.

2.4.2. RDF

Resource Description Framework (RDF) – це стандарт для опису ресурсів у мережі (веб-метаданих), розроблений World Wide Web Consortium (W3C), дозволяє специфікувати семантику даних на основі XML стандартизованим, взаємосумісним способом [53].

RDF призначений для читання та розуміння комп’ютерами. Він підходить для опису будь-якого веб-ресурсу і забезпечує взаємодію між програмами, які обмінюються зрозумілою машині інформацією в Інтернеті. Він написаний мовою розмітки (XML). RDF став загальновизнаною мовою та формалізмом представлення, який може служити міжмовним інтерлінгвом у всьому світі для обміну інформацією.

RDF заснований на ідеї ідентифікації речей в Інтернеті за допомогою

посилань URI і вираження тверджень про ресурси в термінах трійки об’єкт-атрибут-значення. Його мета полягає в тому, щоб додати формальну семантику в Інтернет і забезпечити модель даних і синтаксичну конвенцію для представлення семантики даних стандартизованим способом. Зв'язки між ресурсами описуються в термінах названих властивостей і значень.

RDF має значні переваги перед XML:

– Структура об’єкт-атрибут забезпечує природні семантичні одиниці, оскільки всі об’єкти є незалежними сутностями.

– Модель RDF все ще може використовуватися, навіть якщо синтаксис XML змінюється або зникає, оскільки RDF описує незалежний рівень.

– Розширювана об’єктно-орієнтована система типів, схема RDF (RDFS), була введена як шар на верхній частині базової моделі RDF. RDFS можна розглядати як набір примітивів онтологічного моделювання. У XML немає цього шару, і деякі розробники, використовуючи XML, в кінцевому підсумку створюють шар на верхній частині XML, щоб інтегрувати ці онтологічні примітиви.

– RDFS дозволяє розробникам визначити певний словник для даних RDF і вказати тип об’єкта, до якого ці атрибути можуть бути застосовані. Цей механізм забезпечує базову систему типів для моделей RDF та інтерпретації виразів RDF.

RDF відігравав важливу роль як основа для мови розмітки агентів DARPA (DAML), чиї рівні логіки мають бути побудовані на основі базової структури RDF. Однак недоліком RDF є відсутність формальної базової семантики моделі [54].

2.4.3. OIL

Ontology Interchange Language (OIL) – це повноцінна веб-онтологічна мова на основі RDFS [55]. Мова заснована на трьох елементах, а саме, системах

на основі фреймів, дискрипційній логіці та веб-стандартах. Вона була створена відповідно до вимог описаних у [56].

OIL відповідає наступним вимогам:

– Є інтуїтивно зрозумілою для користувача.

– Має чітко визначену формальну семантику з усталеними властивостями аргументації для забезпечення повноти, правильності та ефективності.

– Має належний зв'язок з існуючими мовами Інтернету, такими як XML та RDF, щоб забезпечити сумісність.

Не всі мови онтологій, відповідають вимогам [56]. Основні переваги OIL полягають у наступному:

– Заснована на парадигмах фреймового та об'єктно-орієнтованого моделювання. OIL базується на понятті класу та його суперкласів та атрибутів. Відношення можна визначити як незалежні сутності.

– Має добре визначену формальну семантику на основі дискрипційної логіки. Крім того, значення будь-якого виразу можна описати математично точним способом, що дає змогу отримувати висновок з описом концепції та автоматично виводити класифікаційні таксономії.

– Створена на основі WEB-стандартів. OIL має чітко визначений синтаксис у XML. OIL також визначається як розширення RDF та RDFS.

– Пропонує різні рівні складності. Мова має чотири шари: Core OIL, Standard OIL, Instance OIL (Standard OIL + RDFS) і Heavy OIL [56].

Проте у OIL виділяють наступні недоліки:

– Неможливо визначити значення за замовчуванням.

– Неможливо надати мета-клас і підтримувати конкретний домен.

– Переклад з OIL на RDF і назад більше не гарантує отримання ідентичної онтології з точки зору моделювання (хоча семантична еквівалентність все ще зберігається).

2.4.4. The DARPA Agent Markup Language (DAML)

Мова розмітки агентів DARPA (DARPA Agent Markup Language, DAML) – це ініціатива, яку спонсорував уряд США, спрямована на розробку мови та інструментів для полегшення концепції семантичної мережі. DAML складається з двох частин, мови онтології та мови для вираження обмежень і додавання правил висновку. DAML також включає відображення інших семантичних веб-мов, таких як SHOE, OIL, KIF, XML і RDF [57].

Мова DAML є розширенням XML і RDF. Останній випуск мови (DAML+OIL) надає багатий набір конструкцій для створення зрозумілих машині онтологій і для розмітки інформації. Мова онтології (DAML +OIL) має чітко визначену теоретико-модельну семантику, а також аксіоматичну специфікацію, яка визначає передбачувані інтерпретації мови. Це робить її однозначно інтерпретованою комп’ютерною мовою.

DAML+OIL є результатом злиття між DAML та OIL. Вона спеціально розроблена для використання в Інтернеті. І вона використовує XML і RDF, додаючи формальну суворість дискрипційної логіки. DAML+OIL використовує об'єктно-орієнтований підхід, описуючи структуру в термінах класів і властивостей. Згідно з [58], з формальної точки зору, DAML+OIL можна розглядати як еквівалент дуже виразної логіки опису. DAML+OIL забезпечує механізми для явного представлення послуг, процесів і бізнес-моделей.

2.4.5. OWL

OWL – це стандартна мова онтологій для семантичної мережі. OWL являє собою мовне розширення схеми RDF. OWL сумісна з ранніми мовами онтологій, такими як SHOE, DAML+OIL, і надає інженеру більше можливостей для вираження семантики [59]. Вона включає кон’юнкцію, диз’юнкцію, екзистенційні та універсальні кількісні змінні, які можна використовувати для здійснення логічних висновків та отримання знань.

Внаслідок того, що деякі конструкції OWL дуже складні, було розроблено три різновиди OWL, або три «підмови»:

– OWL Lite – підтримує ієрархію класифікації та прості функції обмежень, такі як обмеження потужності. Перевага OWL Lite полягає в тому, що її легше зрозуміти (для користувачів) і легше реалізувати (для розробників інструментів) [60]. Недоліком є обмежена виразність.

– OWL DL – включає всі конструкції мови OWL з обмеженнями, такими як поділ типів. Відповідає логіці опису. Перевага полягає в тому, що така версія мови дозволяє ефективно підтримувати міркування [60]. Однак повна сумісність з RDF втрачена.

– OWL Full – підтримує максимальну виразність і синтаксичну свободу RDF без жодних обчислювальних гарантій. OWL повністю сумісний з RDF як синтаксично, так і семантично [60]. Однак така версія мови стає нерозбірливою і не придатною для повного міркування в окремих випадках.

OWL має наступні недоліки [61]:

– Деякі конструкції дуже складні, тому було розроблено три підмови.

– Міркування не надто неефективне, оскільки існує компроміс із складними витратами часу.

– Мова не є інтуїтивно зрозумілою, користувачу потрібно бути досить кмітливим, щоб побудувати ефективні конструкції знань.

2.5. Порівняння розглянутих мов

Очевидно, кожна мова онтологій має переваги та недоліки, і, таким чином, вибір мови онтології має керуватися метою, для якої дана мова буде використовуватися.

Як видно з наведеного вище опису, мова OWL претендує на те, що є найпопулярнішою мовою та має найбільшу спільноту. OWL є найбільш поширеною семантичною веб-мовою. Однак основним недоліком OWL є

відсутність допоміжних інструментів. На даний момент було знайдено лише два інструменти – Protégé та OWL Validator – які підтримують OWL та дозволяють розробляти онтології OWL.

Хоча KIF і RDF + RDF(S) мають недоліки, такі як важка вага в першому випадку і відсутність формальної базової семантики моделі в другому випадку, вони мають ширше застосування.

Серед розглянутих мов можна виділити одну спільну рису – це багатошаровий підхід і використання стандартів. На основі певного стандарту було розроблено ряд мов. Наприклад, RDF використовується для розробки OWL, XML використовується для розробки RDF тощо. Основною перевагою використання багатошарового підходу до створення певної мови є вдосконалення та уточнення існуючої мови. Наприклад, RDF покращено та розширено для створення OWL.

Останнім часом також популярний «підмовний» або модульний підхід. Це означає, що мова складається з кількох модулів або підмов, де кожна підмова доповнює попередню підмову. Це твердження можна визначити за допомогою рівняння (2.1).

$$L = \{L_i, i = 1, \dots, n, L_i < L_{i+1}\} \tag{2.1}$$

де L – мова, що складається з підмови L_i, і кожна наступна підмова доповнює попередню підмову.

Цей підхід використовується в OWL, який складається з OWL Lite, OWL DL і OWL Full, а також DAML+OIL.

Іншим прикладом є SWRL, що дозволяє користувачам писати правила в термінах концепцій OWL. Він створений, щоб забезпечити більш потужні можливості дедуктивного міркування, ніж OWL. Семантично, SWRL побудовано на тій же логіці опису, що й OWL, і SWRL забезпечує аналогічні сильні формальні гарантії під час виконання висновку.

Основна перевага підмовного підходу полягає в тому, що існуючу мову

можна розширити шляхом створення підмови без необхідності визначати нову мову за допомогою набору нових конструкцій.

2.6. Інструменти для розробки онтології

Для розробки OWL онтологій використовують такий інструмент як Protégé – це безкоштовний редактор онтології з відкритим вихідним кодом і фреймворк для побудови інтелектуальних систем [62].

Protégé підтримується сильною спільнотою академічних, державних і корпоративних користувачів, які використовують Protégé для створення рішень, заснованих на знаннях, у таких різноманітних областях, як біомедицина, електронна комерція та організаційне моделювання.

Архітектуру плагінів Protégé можна адаптувати для створення як простих, так і складних програм на основі онтології. Розробники можуть інтегрувати Protégé із системами правил або іншими засобами для вирішення проблем, щоб створити широкий спектр інтелектуальних систем. Перевагами даного інструменту є:

– Активна спільнота. Protégé активно підтримується сильною спільнотою користувачів і розробників, які ставлять питання, пишуть документацію та створюють плагіни.

– Підтримка стандартів W3C. Protégé повністю підтримує найновішу мову Web Ontology OWL 2 і специфікації RDF від Консорціуму World Wide Web.

– Відкрите середовище, здатне до розширювання. Protégé заснований на Java, є розширюваним і забезпечує середовище plug-and-play, що робить його гнучкою базою для швидкого створення прототипів і розробки додатків.

2.7. Висновок до розділу

В цьому розділі були розглянуті принципи роботи людської па’мяті та когнітивні процеси. Було зазначено, що в системі використовується ревізія

таксономії Блума, яку виконали Картвел та Андерсон, як основа для визначення переліку когнітивних навичок.

Також у цьому розділі детально було розглянуте поняття онтології та проаналізовані інструменти для реалізації онтологій, так звані, онтологічні мови. В результаті аналізу існуючих на сьогодні інструментів було обрано мову OWL, так як на сьогоднішній день вона є найбільш поширеним і повним інструментом, яким дозволяє описувати інформацію такого характеру, має найбільшу спільноту.

РОЗДІЛ 3
ОБҐРУНТУВАННЯ ОБРАНОГО ПІДХОДУ ДО ЗБЕРІГАННЯ ЗНАНЬ ТА РОЗРОБКА ТЕСТОВОЇ ОНТОЛОГІЇ

3.1. Архітектура системи

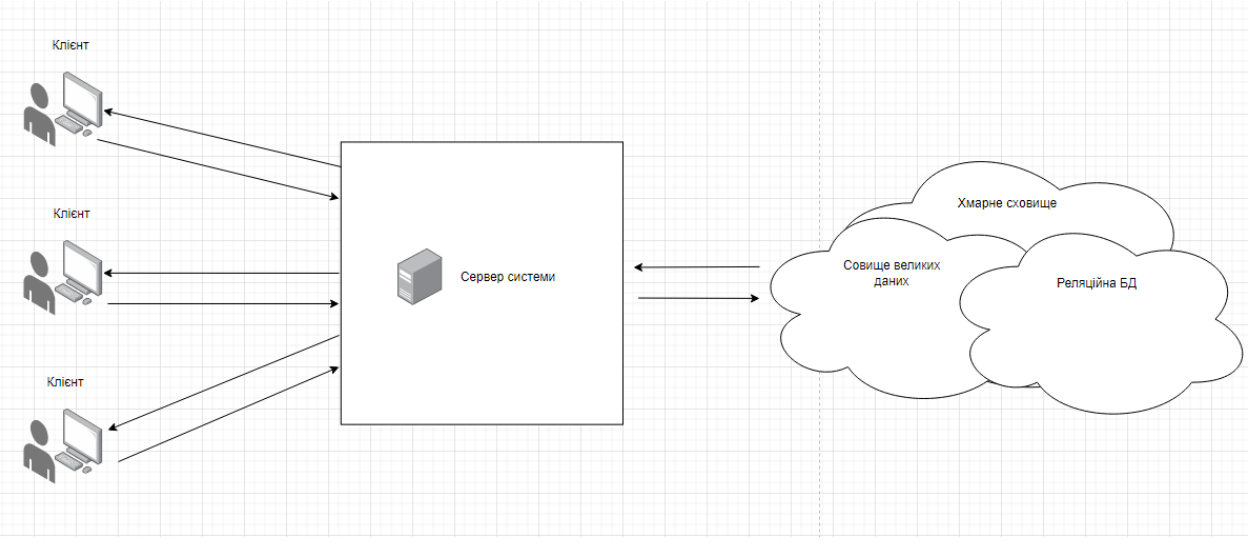

Рис. 3.1. Архітектура системи дистанційної освіти

На рис. 3.1 зображені основні елементи архітектури системи дистанційної освіти. В системі є клієнти – це ті, хто користуються сервісом, це можуть бути студенти, викладачі, адміністратори тощо. Основна цільова аудиторія – це студенти. Тому на їх прикладі розглянемо роботу системи.

Клієнтська сторона відправляє запити на сервер, а сервер в свою чергу їм відповідає. Як правило, сервер відправляє такі дані як освітній контент (курси, окремі теми курсів, навчальні матеріали), тести та задачі.

Серед основних даних, які клієнтська сторона відправляє на сервер, можна виділити наступні:

- Реакції на перегляд навчальних матеріалів (перший перегляд, кількість повторних переглядів, час перегляду).
- Відомості про задачі (кількість спроб рішення певної задачі, час, витрачений на рішення кожної задачі, текст рішення/відповідь).

Сервер системи отримані дані відправляє в сховище великих даних. Воно поділяється на дві частини: власне сховище великих даних та додаткова реляційна база даних, за допомогою якої буде відбуватись аналітика даних. Аналітика впливає на характер віддачі сервером клієнту контенту.

До сховища великих даних висуваються наступні вимоги:

1. Повинне вміти зберігати звичайні дані у вигляді реляційних таблиць або колекцій об’єктів.
2. База даних повинна вміти зберігати графи (чи у формі матриці суміжності, чи у вигляді переліку ребер тощо).
3. Повинна бути можливість вбудовувати код в систему великих даних для того, щоб виконувати аналітичну роботу саме на стороні великих даних, щоб не навантажувати сервер системи.

Процеси когнітивного моделювання, формування онтології студента та порівняння її з ідеальною онтологією відбуваються на стороні великих даних.

Мета створення такої системи полягає в вихованні у людини гнучкості свідомості. Вважається, що за наявності певних когнітивних навичок можна надати студенту іншу дисципліну (суміжну або ні), де необхідні такі ж когнітивні навички, та очікувати від нього певних успіхів у швидшому засвоєнні навчального матеріалу нової дисципліни.

За допомогою такого підходу студенту можна буде рекомендувати певні дисципліни, для вивчення яких в нього є суттєві перспективи.

3.2. Види знань у системі

Для реалізації принципу, що буде описаний далі в роботі, знання пропонується поділити на два типи: об’єктивні та суб’єктивні. Під об’єктивними знаннями маються на увазі дійсні знання у певній предметній області. Суб’єктивні знання відображають знання певної людини у цій же предметній області.

Відповідно до цього, пропонується побудувати дві онтології: еталонну онтологію та онтологію знань студента.

Еталонна (ідеальна) онтологія – це об’єктивне відображення знань з предметної області сформоване в результаті експертних оцінок, на основі голосування більшості експертів. Еталонна онтологія відображає стан знань «ідеального» студента.

Онтологія знань студента – це відображення точки зору системи про те, що знає, або чого не знає студент, згідно з еталонною онтологією.

3.3. Фіксація знань студента

Для кожного студента в системі будується його онтологічна картина, в якій деякі з понять або зв’язків можуть бути спотворені або відсутні.

Для того, щоб отримати представлення знань студента, а саме його онтологію знань, проводиться тестування при вступі на курс.

Не проходження показує системі, що навички у студента відсутні і такого студента потрібно вчити з нуля. Якщо студент проходить навчання з нуля, то система запускає його у повний курс. Він вивчає матеріали та поступово виконує завдання по темам. Після кожного тестування картина знань студента уточнюється.

Проходження вступного тесту показує певний рівень навичок студента, що дозволяє робити висновки про те, які теми студент може засвоїти, а які ні.

В ході навчання на курсі студент поступово буде виконувати завдання, та по результатам виконання цих завдань онтологія студента буде уточнюватись і доповнюватись, відображаючи його поточні знання.

Когнітивне моделювання дозволяє налаштувати фільтр, через який ми пропускаємо знання студента та намагаємось побудувати дану онтологію знань студента. Саме когнітивне моделювання є сполучною ланкою між онтологією об’єктивної реальності та онтологією студента. Пропускаючи через когнітивну модель знання студента ми отримуємо їх відбиток у системі.

В даній роботі задача відображення цієї когнітивної моделі не ставилась, тому вона розглядається як чорний ящик, на вхід у який подаються результати тестування, а на виході система отримує онтологію знань студента і кількісну оцінку когнітивних навичок.

Тести, що надаються студенту, повинні перевіряти кожен з описаних в ідеальній онтології моментів: наявність концептів, правильність розстановки і напрямків зв'язків між концептами тощо. Тести мають включати перевірку якнайбільшої кількості навичок і повинні бути побудовані таким чином, щоб вони перехрещувались – тобто декілька разів перевіряли одну і ту ж навичку в різних комбінаціях для уточнення онтологічної картини знань студента.

Після отримання онтології знань студента система порівнює її з ідеальною онтологією. Існують математичні алгоритми, що дозволяють отримати відповідь на це питання (наприклад, алгоритм ізоморфності графа), але оскільки в даній роботі розглядається саме підхід до зберігання знань, то задача визначення ізоморфності графа не ставилася.

В процесі навчання відомо, що саме повинен знати студент, можна відслідковувати його реакцію на нові знання шляхом аналізу правильності виконання завдань і, таким чином, можливо сформувати відбиток знань студента в системі дистанційної освіти.

Варто зазначити, що на початкових етапах онтологія скоріш за все буде не повністю відповідати дійсності, тому що, по-перше, для комплексного тестування потрібен дуже досконалий інструмент – чітко продумані та детальні тести, по-друге, такі досконалі тести стали б багатогодинною фізіологічно тяжкою процедурою, яку до того ж не має сенсу проходити тільки один раз. Але така точність на початкових етапах не потрібна, тому що в будь-якому випадку уточнення та модифікація онтології буде відбуватись постійно в процесі навчання.

3.4. Процес навчання

Предметом навчання у контексті запропонованої концепції є процес доповнення та виправлення помилкових або викривлених моментів в онтології знань студента.

Онтологія, що представляє знання, є по суті орієнтованим графом і відповідно вже є математичною моделлю.

Для виправлення зв'язків, враховуючи те, що граф орієнтований, можна вирішити задачу розфарбування графа. Алгоритм розфарбовування графа призначає вершинам графа певні кольори задовольняючи визначеним умовам. У цьому алгоритмі потрібно розфарбувати вершини графа за допомогою n кольорів, і будь-які дві суміжні вершини не повинні мати однаковий колір.

Хроматичне число графа – це найменше число n, таке, при якому вершини графа можуть бути розфарбовані за допомогою n кольорів таким чином, що не знайдеться двох суміжних вершин одного кольору. Алгоритм розфарбування графа дозволяє знаходити значення хроматичного числа n довільного графа та відповідну до цього значення схему розфарбування вершин.

Розфарбовування вершин, що відповідає цьому числу, розбиває множину вершин графа на n підмножин, кожна з яких містить вершини одного кольору. Ці множини є незалежними, оскільки в межах однієї множини немає двох суміжних вершин.

Існує декілька основних типів алгоритмів розфарбовування графів: алгоритми, що використовують пошук ушир або пошук у глибину, генетичні алгоритми та так зване «жадібне» зафарбовування (greedy colouring).

Псевдокод для реалізації алгоритму «жадібного» зафарбовування наведений нижче.

```
Вхід:
G = (V, E)          Ненапрямлений простий граф

Вихід:
Схема розфарбування вершин для графа G

begin
    {v1, v2, … , vn} – sequence of vertices in V
    for v:=v1 to vn
    begin
          Assign vertex v the smallest possible color such
          that no conflict exists between v and its colored
          neighbors.
    end
end
```

Ступінь вершини дає можливість віднести її до того, чи іншого кольору. Тобто можна почати фарбувати граф від вершин з найменшим ступенем (з найменшою кількістю ребер) та поступово рухатись по вершинах в глибину графа.

Таким чином можна побудувати порядок викладення навчального матеріалу та методологію навчання. Спочатку виправити ті концепти, які залежать від меншої кількості інших концептів, а потім рухатись до більш складних понять.

3.5. Опис характеру даних, що зберігаються в системі

В системі дистанційної освіти визначені наступні типи інформації, яку система отримує та зберігає:

- Матеріали курсів.
- Інформація про «ідеального» студента: ідеальна когнітивна картина та еталонна онтологія (в рамках курсу).

- Інформація про реального студента: його когнітивна картина та онтологія суб’єктивних знань.
- Варіанти рішення завдань студента (дані тестів).

Наведені вище дані існують по різним законам в часі та просторі, мають різні характеристики частоти використання, часу зберігання, об’єму тощо, тому їх варто зберігати в різних місцях та з різними правилами для оптимізації роботи системи.

Для того, щоб правильно декомпозувати інформацію, необхідно поділити її на категорії за деякими характеристиками, такими як: об’єм, швидкість накопичення, час існування, актуальність тощо. Визначивши ці характеристики для певної групи інформації, що надходить до системи можна говорити про спосіб її зберігання.

При побудові будь якого сховища даних приймається до уваги фактор статичності/динамічності інформації. Таким чином виділяють умовно «довідкову» інформацію та оперативну інформацію. Оперативна інформація поділяється на інформацію з малою швидкістю зміни та з великою швидкістю зміни. Цей поділ існує тому, що інформацію з різною швидкістю зміни краще зберігати у різних частинах сховища та можливо у різному вигляді, оскільки інформацію, що змінюється повільніше можна ефективно проіндексувати та швидко виконувати пошук. А інформація, яка накопичується швидко, можливо не потребує індексації, оскільки буде використана тільки під час аналітичної обробки і взагалі потім може бути видалена з системі, тому що має виключно службовий характер.

Для зберігання навчальних матеріалів використовуються способи, які вимагають мінімальної трансформації перед видачою клієнту в браузер (у вигляді html-сторінок). Оскільки клієнти в системі веб-орієнтовані, то до системи висувається вимога, щоб матеріали зберігались в такому вигляді, щоб їх без трансформації можна було б віддати клієнту. Ці матеріали практично не змінюються з часу створення курсу.

Виходячи з матеріалів курсу експерти створюють еталонну онтологію предметної області, яку система намагається сформувати у студента та перелік когнітивних навичок та їх «ідеальних» оцінок.

Інформація про «ідеального» студента являє собою одноразово сформовану майже не змінювану статичну довідкову інформацію: еталонна онтологія, побудована експертами, та ідеальна когнітивна картина для певного курсу з оцінками когнітивних навичок, сформована тими ж експертами. Така інформація змінюється вкрай нечасто, тому її можна вважати умовно константною.

Враховуючи те, що для написання та зберігання онтології є спеціалізовані інструменти, то еталонна онтологія може зберігатись у вигляді окремих спеціальних файлів (наприклад, .owl) в розподіленому сховищі. Ці засоби є неодноразово перевіреними та достатньо ефективними. Крім того, з онтологією повинно бути зручно працювати за допомогою незалежних інструментів. Вона повинна зберігатись в такому вигляді, щоб операції з нею можна було б виконувати спеціалізованими засобами без трансформації даних.

Когнітивні навички та їх оцінки зберігаються у форматі словника «ключ-значення».

Для кожного окремого користувача в сховищі системи повинна бути виділена певна область. Інформація про реального студента змінюється в часі. Деяка змінюється часто, наприклад результати тестів, а деяка не дуже часто. Онтології знань, наприклад, наростають повільно, але все ж таки вони змінюються у часі.

Область, виділена для зберігання інформації про студента повинна бути поділена на дві зони: в першій зоні повинна зберігатись інформація, що змінюється повільно (історія зміни та остання версія онтології суб’єктивних знань та когнітивних навичок), в другій зоні повинні зберігатись дані, що швидко накопичуються (поточна статистика студента по проходженню тестів, завдань тощо).

Завдяки тому, що система зберігатиме історію зміни онтології та когнітивних навичок, можна буде відслідковувати прогрес студента у часі і визначати, чи відбувається «позитивна трансформація», чи ні.

Онтологія суб'єктивних знань, так само як і ідеальна онтологія, повинна зберігатись у вигляді окремих спеціальних файлів (наприклад, .owl), щоб з нею легко можна було виконувати маніпуляції без додаткових трансформацій даних.

Когнітивні навички конкретного студента мають прив'язку до часу. Система повинна розуміти, що на певний момент часу у студента були виявлені певні навички. Це потрібно для того, щоб можна було зіставити маніпуляції, що проводилися зі студентом (використання певних методологій навчання, видача певних навчальних матеріалів) з його поточним станом когнітивних навичок і знань, та визначити їх ефективність відносно студента і скорегувати порядок видачі навчальних матеріалів якщо в цьому є потреба. Інформацію такого типу можна зберігати у вигляді об'єктів.

Дані тестів, відповіді на завдання, коди програм система накопичує для проведення аналітичних операцій з метою корекції видачі навчального матеріалу за потреби. Дана інформація може зберігатись у вигляді набору об'єктів, або звичайних реляційних таблиць. В даному випадку формат зберігання не є принципово важливим, бо аналітичний модуль, який буде обробляти дану інформацію буде зовнішнім, та він буде вилучати дану інформацію з бази та обробляти її окремо. Тому формат зберігання варто обирати виходячи з того, які засоби зберігання великих даних використовуються.

3.6. Реалізація тестової еталонної онтології

Кожна область знань має велику кількість концепцій та власну термінологію. Ці абстракції повторити для всіх систем в рамках кваліфікаційної роботи неможливо, тому пропонується розглянути запропоновану концепцію на прикладі вивчення мов програмування.

На рисунку 3.2 представлена частина онтології, що відображає фрагмент об’єктивної реальності у обраній області – вивчення мов програмування.

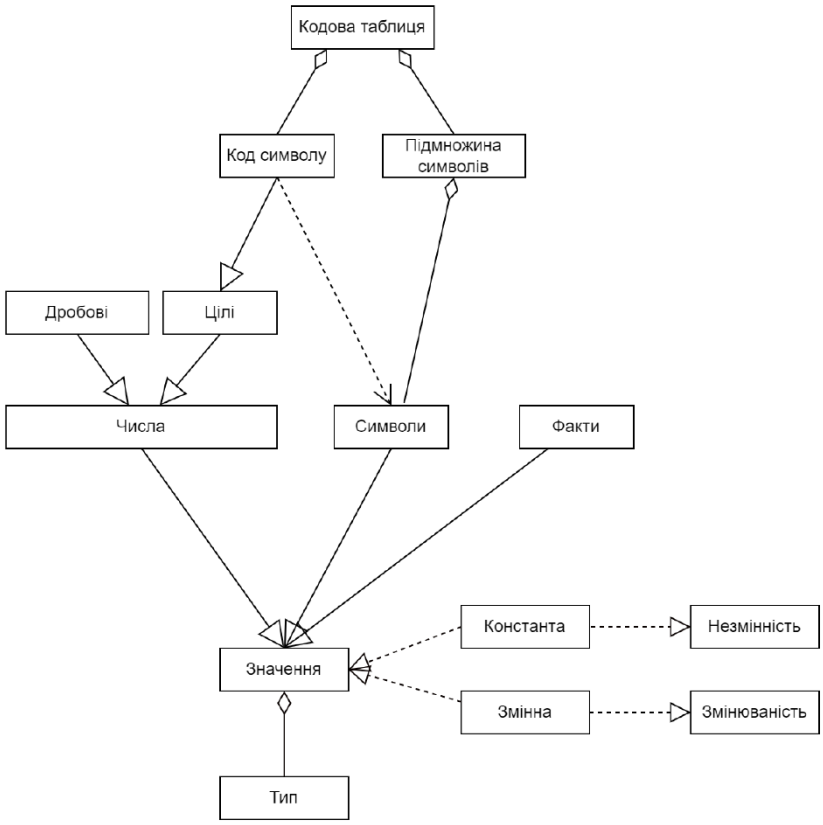

Рис 3.2. Фрагмент об’єктивної реальності у області вивчення мов програмування

Фрагмент цієї ж онтології, реалізований мовою онтологій OWL за допомогою спеціалізованого інструмента роботи з онтологіями Protégé (рис. 3.3).

Приклад реалізації зв’язки класів «Число» та «Значення» на OWL:

```
	<owl:Class
rdf:about="http://www.semanticweb.org/dell/ontologies/2022/0/untitled-
ontology-2#Number">
	<rdfs:subClassOf
rdf:resource="http://www.semanticweb.org/dell/ontologies/2022/0/untitled-
ontology-2#Value"/>
	</owl:Class>
```

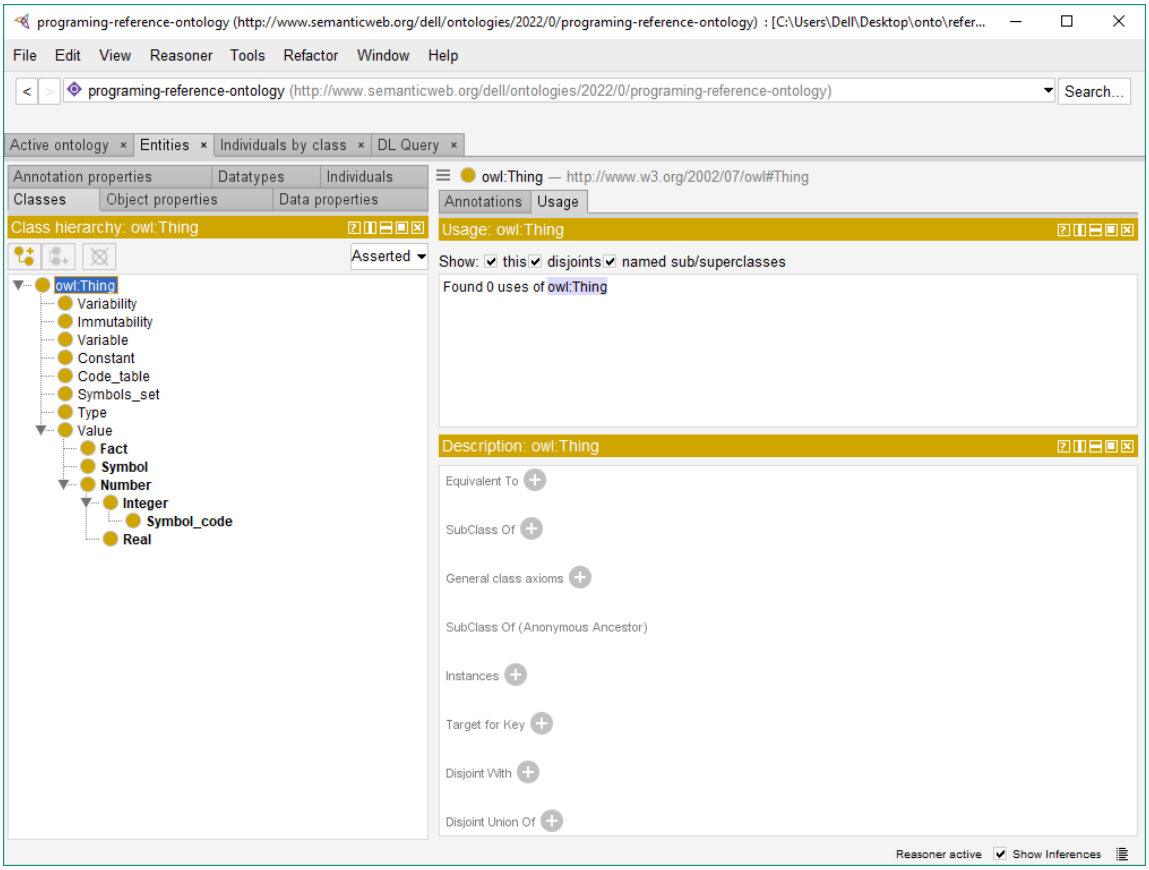

Рис. 3.3. Вигляд робочого простору Protégé

Приклад реалізації зв’язку типу «has a» на OWL:

```
	<!--	http://www.semanticweb.org/dell/ontologies/2022/0/untitled-ontology-2#has_a -->
	<owl:ObjectProperty rdf:about="http://www.semanticweb.org/dell/ontologies/2022/0/untitled-ontology-2#has_a">
		<rdfs:domain rdf:resource="http://www.semanticweb.org/dell/ontologies/2022/0/untitled-ontology-2#Value"/>
		<rdfs:range rdf:resource="http://www.semanticweb.org/dell/ontologies/2022/0/untitled-ontology-2#Type"/>
	</owl:ObjectProperty>
```

Повний код онтології наведений у додатку А.

Було проведене оцінювання студента, що бажав вступити на курс з вивчення мови Python, з метою перевірки наявності базових знань у студента.

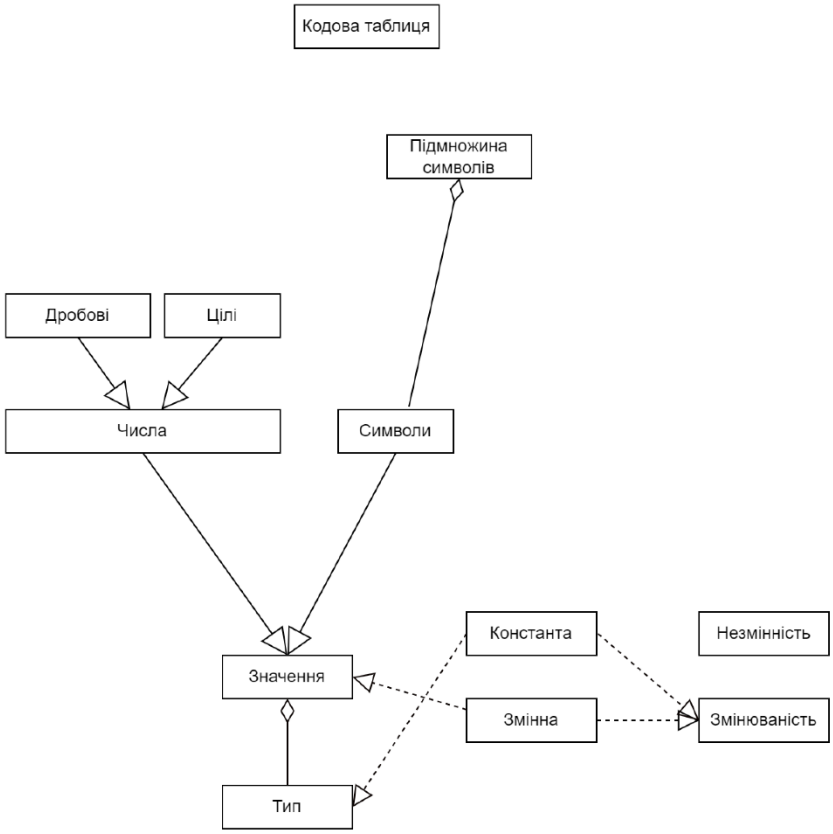

Рис 3.4. Фрагмент суб’єктивної реальності студента у області вивчення мов програмування

Після проведення тестування та аналізу результатів за допомогою когнітивної моделі була отримана онтологія, представлена на рис. 3.4. Ця онтологія містить суб’єктивну реальність студента в даній предметній області (онтологію знань студента). Як видно з рисунку, деякі концепти відсутні, певні зв’язки спотворені або також взагалі відсутні. З такою онтологією система вже може працювати та підбирати навчальні матеріали відповідно до онтології суб’єктивної реальності студента.

3.7. Опис системи дистанційної освіти

Для тестування роботи онтологій був розроблений фрагмент системи дистанційної освіти з використанням мови Python і фреймворків Django та Django Rest Framework.

У системі дистанційної освіти студенту надається можливість увійти в особистий кабінет, де він у будь-який час, та з будь-якого пристрою може мати доступ до всіх навчальних матеріалів та може вирішувати задачі.

Далі детальніше буде розглянуто сторінки, пов'язані зі здобуванням знань та їх фіксацією у системі.

На рис. 3.5 наведено скріншот особистого кабінету студента у системі дистанційної освіти. На вкладці «Профіль» студент може переглядати та редагувати свої особисті дані, відслідковувати повідомлення та отримувати інформацію щодо дійсних підписок.

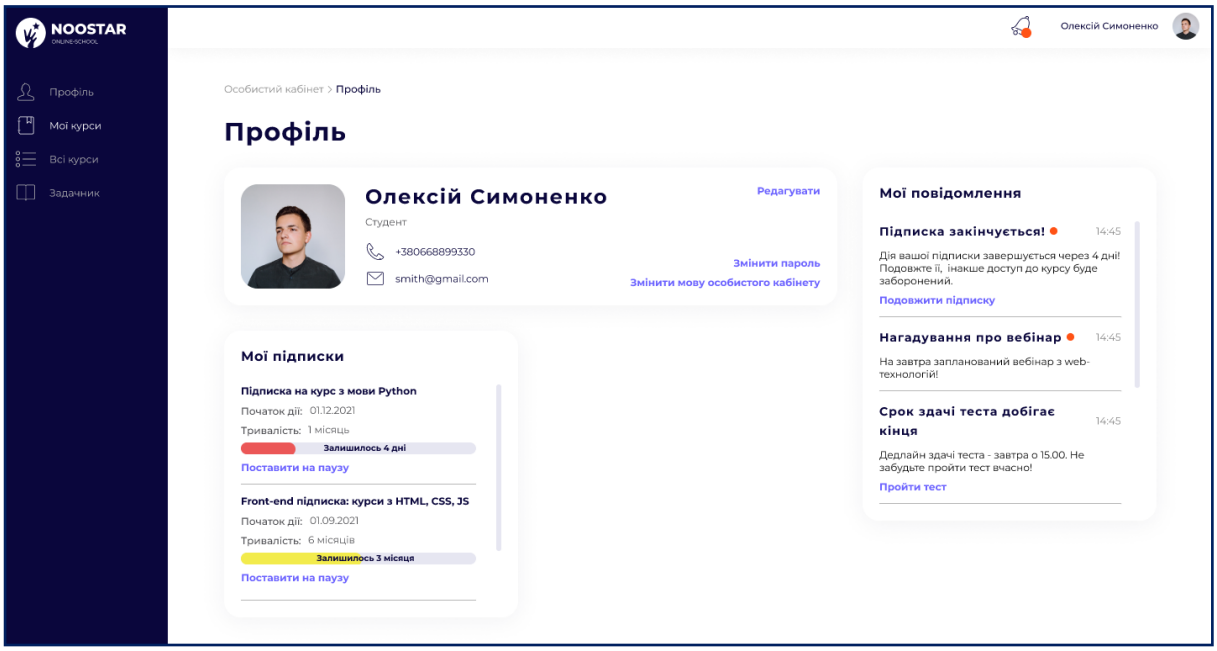

Рис. 3.5. Особистий кабінет студента

На цій сторінці представлено низку елементів, які роблять запити на сервер системи за допомогою різних API, але одним з ключових API на даній

сторінці є отримання інформації про студента. Опис даного API вказаний у таблиці 3.1.

Таблиця 3.1

Опис API для отримання інформації про студента

URL функції API	/api/1.0/profile
Метод	GET
Ресурс	користувач
Вхідні параметри	access_token у заголовку запиту

Вихідні параметри	Об’єкт користувача, JSON наступного формату
	{
	"uuid": "76ffecbc-c75d-451b-8e92-93b0d8b0707e",
	"first_name": "Irina",
	"last_name": "Cherechecha",
	"photo": "https://dev.noostar.dp.ua/media/IMG_20210904_223350_841-01.jpg",
	…
	}

Приклад коду, що реалізує виклик цієї функції API:

```
class ProfileAPI(generics.GenericAPIView):
    serializer_class = ProfileSerializer
    permission_classes = [permissions.AllowAny]
    def get(self, request):
        user = request.user
        if not user:
            return Response({'status': '401', 'message': 'Unauthorized'}, status=401)
        else:
```

```
                return
Response(self.get_serializer(self.get_queryset().get(user=user)).d
ata, status=200)
```

При виборі курсу студент може переглядати інформацію про сам курс, про його тематику, про базові навички, які необхідні для навчання на цьому курсі, і про те, які навички очікуються по завершенню навчання. Саме це показано на рис. 3.6-3.7.

Рис. 3.6. Сторінка курсу (частина 1)

Ключовим API на цій сторінці є отримання довідкової інформації про курс. Оскільки клієнтський застосунок підтримує серверний пре-рендеринг, та однією з вимог до системи є не вимушувати користувача довго чекати, то довідкові дані про всі курси надходять до клієнтського застосунку ще на етапі завантаження списку курсів на сторінці «Всі курси». Опис даного API, що викликається для перегляду всіх доступних курсів, приведений у таблиці 3.2.

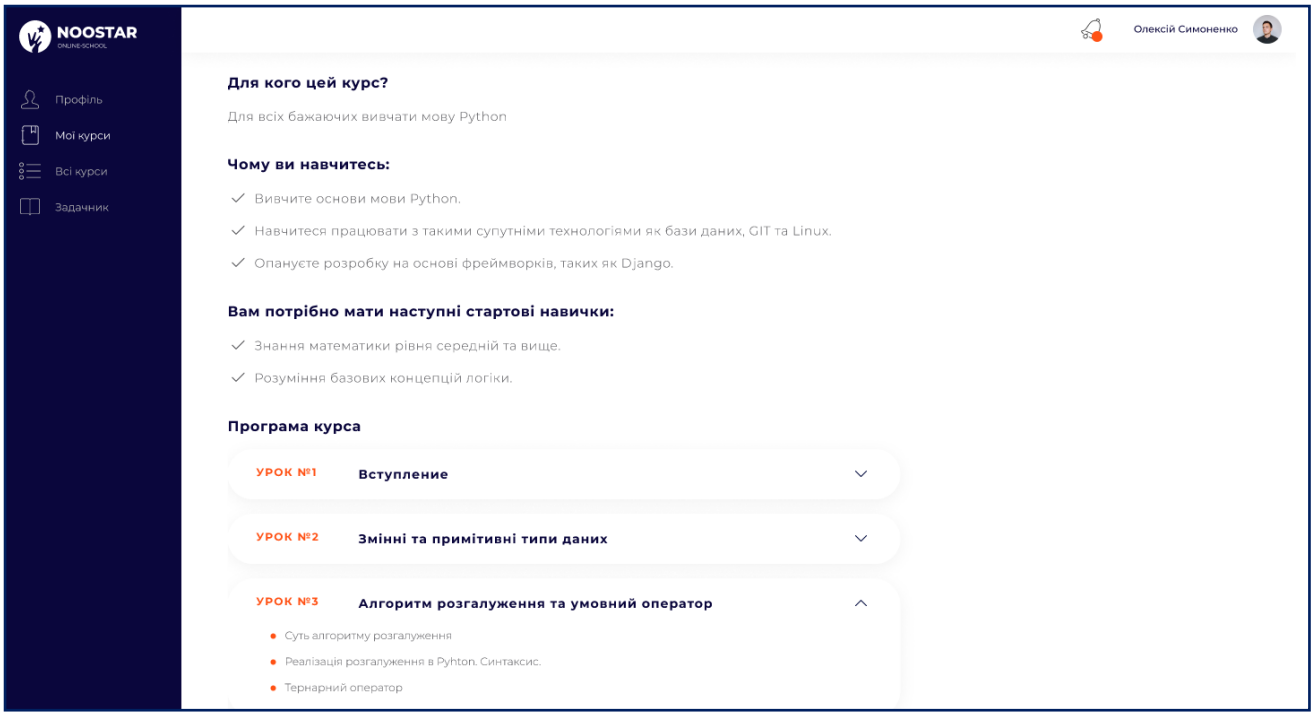

Рис. 3.7. Сторінка курсу (частина 2)

Таблиця 3.2

Опис API для отримання довідкової інформації про курс

URL функції API	/api/1.0/course
Метод	GET
Ресурс	курс
Вхідні параметри	-

Вихідні параметри	Масив об’єктів курсів, JSON формату
	`[`
	`  {`
	`  "uuid": "ce12c8e2-9476-4b5b-9dbb-e7bffc79887f",`
	`  "tags": […],`
	`  "titles": […],`
	`  …`
	`]`

Приклад коду, що реалізує виклик цієї функції API:

```
class CourseAPI(generics.GenericAPIView):
    …
    permission_classes = [permissions.AllowAny]
    serializer_class = CourseSerializer
    queryset = Course.objects.all()
    def get(self, request):
        return
Response(self.get_serializer(self.filter_queryset(self.get_queryse
t()), many=True).data, status=200)
```

Перед вступом на курс студенту пропонується пройти тестування, яке допоможе зафіксувати його когнітивні навички та рівень базових знань. Сторінку тестів показано на рис. 3.8. За допомогою інформації, отриманої з подібних тестів, система буде будувати уявлення про студента: його онтологію суб'єктивних знань та когнітивну картину.

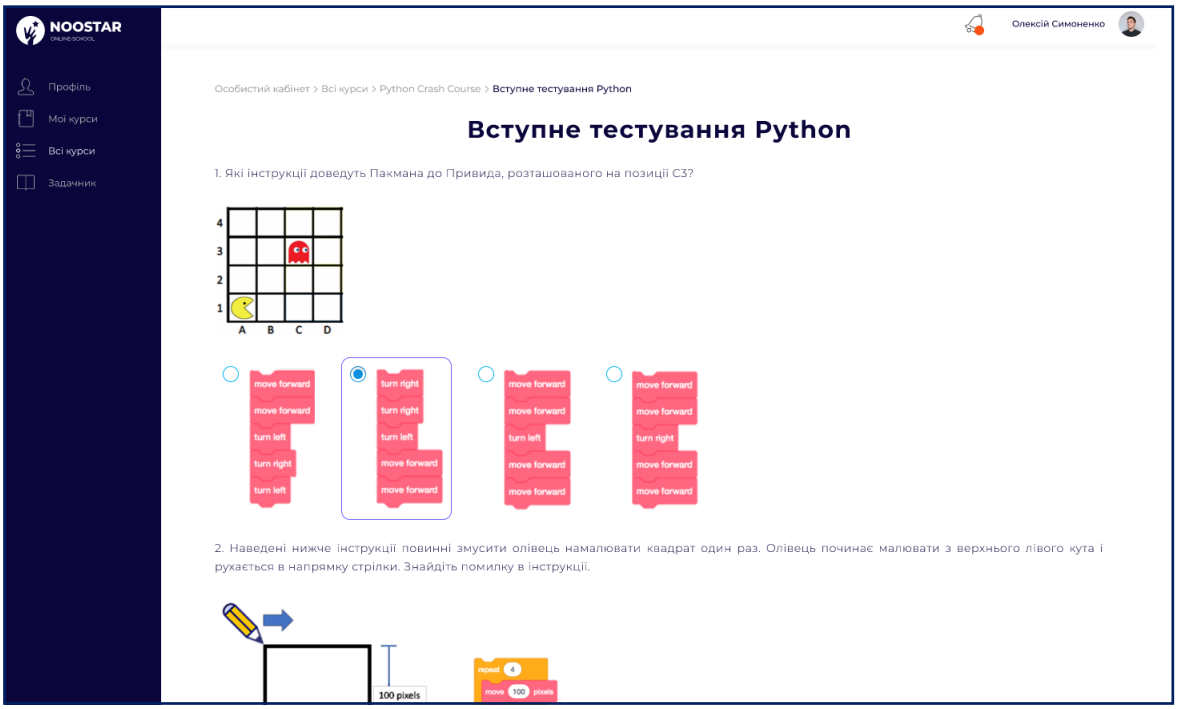

Рис. 3.8. Сторінка тестів

На сторінці проходження тесту основним функціоналом є відправка даних пройденого тесту на сервер. За це відповідає API, описане у таблиці 3.3.

Таблиця 3.3

Опис API для відправки результатів тестування

URL функції API	/api/1.0/course/<course_uuid>/entry_test
Метод	PUT
Ресурс	курс
Вхідні параметри	Дані тесту у форматі JSON

Вихідні параметри	JSON наступного формату
	{
	"status": "success",
	"message": "Test was sent to server."
	}

Приклад коду, що реалізує дане API:

```
class TestResultAPI(generics.GenericAPIView):
    serializer_class = TestResultSerializer
    queryset = TestResult.objects.all()

    def put(self, request):
        if 'student_uuid' not in request.data:
            return createResponseWithStatus(
            'student_uuid is required', 400)
        if 'test_results' not in request.data:
            return createResponseWithStatus(
            'test answers is required', 400)
        try:
            process_data(request.test_results)
            return createResponseWithStatus(
```

```
            'Test was sent to server ', 200)
    except Exception as e:
            return createResponseWithStatus(
            f'An error occured {e}', 400)
```

В процесі навчання студент буде переглядати навчальні матеріали курсу, які йому пропонує система. Типова сторінка теми курсу зображена на рис. 3.9. Студент може обирати необхідну тему в меню зліва, та матеріали з цієї теми будуть відображені на екрані.

Далі, після вивчення певної теми студент буде знову проходити тестування та виконувати задачі, тим самим поповнюючи свою онтологію та вдосконалюючи когнітивну картину.

Коли студент переходить на сторінку курсу з навчальними матеріалами, то клієнтським застосунком робиться запит, описаний в таблиці 3.4.

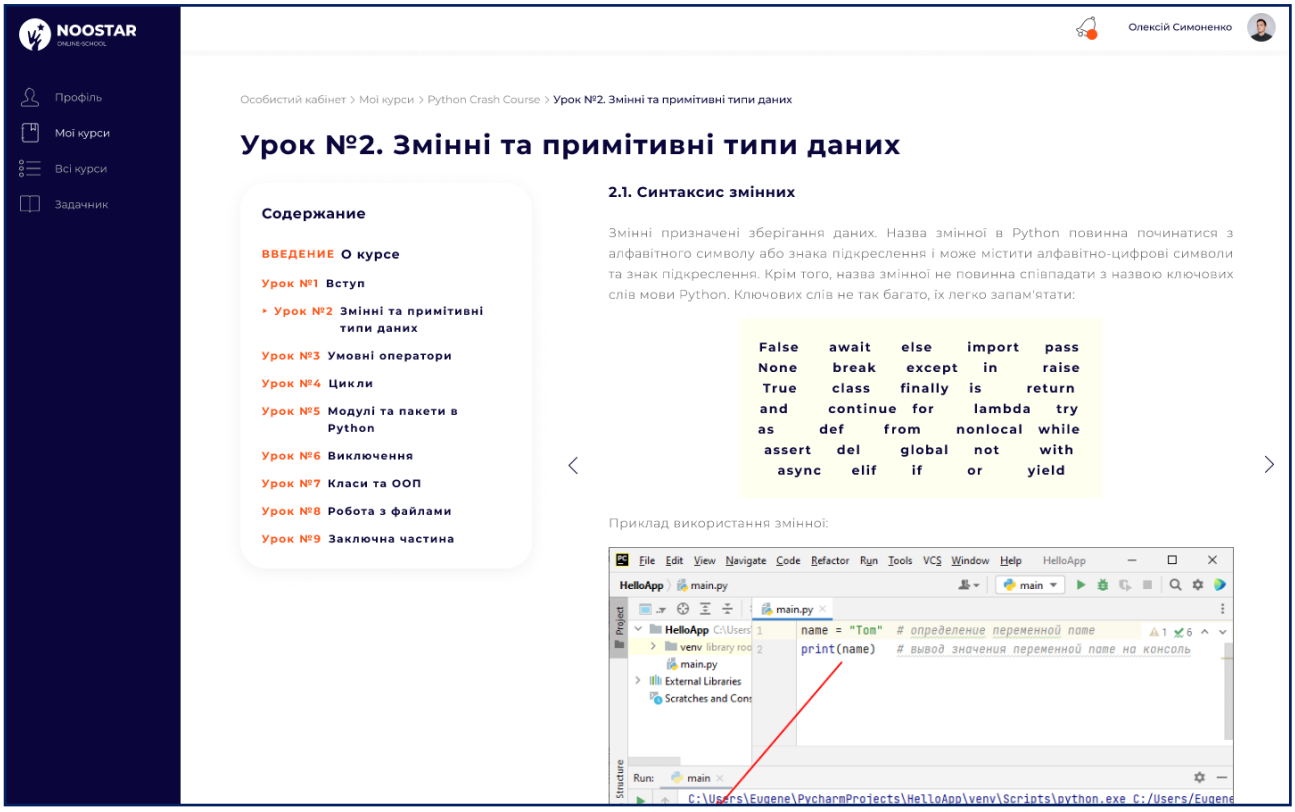

Рис. 3.9. Сторінка курсу

Таблиця 3.4

Опис API для отримання контенту курсу

URL функції API	/api/1.0/ courses/<str:course_uuid>/content/
Метод	GET
Ресурс	курс
Вхідні параметри	Ідентифікатор курсу у посиланні, access_token у заголовку

Вихідні параметри	Об’єкт курсу, JSON наступного формату
	{
	"owner":"5f11ffe7",
	"difficulty":"A",
	"price":155,
	"titles": [
	…
	],
	}

Приклад коду, що реалізує дане API:

```
class CourseContentAPI(generics.GenericAPIView):
    …
    def get(self, request, course_uuid):
        user = request.user
        user_courses = UserCourseSubscription.objects.filter(user=user, course=course_uuid).order_by("-date_start")
        if not user_courses or len(user_courses) == 0:
            return Response({'status': 'error', 'message': 'course not found'}, status=400)
        latest_course = user_courses[0]
        if latest_course.date_finished:
            return Response({'status': 'error', 'message': 'subscription was expired'}, status=403)
```

```
            if latest_course.is_paused:
                return Response({'status': 'error', 'message':
'subscription was paused'}, status=408)
            return
Response(self.get_serializer(self.get_queryset().filter(course=cou
rse_uuid), many=True).data, status=200)
```

3.8. Висновок до розділу

В даному розділі була описана архітектура системи дистанційної освіти та розглянуті типи інформації, які необхідно зберігати у системі. Для кожного типу інформації був описаний її характер та рекомендований формат зберігання.

Також був описаний процес навчання та його особливості і була розроблена тестова еталонна онтологія для області вивчення мов програмування.

Для перевірки концепції використання онтологій був розроблений фрагмент системи дистанційної освіти з використанням мови Python і фреймворків Django та Django Rest Framework.

РОЗДІЛ 4
ЕКОНОМІКА

4.1. Визначення трудомісткості розробки

Початкові параметри:

1. Прогнозоване число операторів – 1027.
2. Коефіцієнт складності програми – 1,4.
3. Коефіцієнт корекції програми в ході її розробки – 0,05.
4. Погодинна заробітна плата програміста – 200 грн/год.
5. Коефіцієнт збільшення витрат праці внаслідок неповного та/або недостатнього опису задачі – 1,3.
6. Коефіцієнт кваліфікації програміста, обумовлений стажем роботи в даній спеціальності – 1,2.
7. Вартість машино-години ЕОМ –12 грн/год.

Процес нормування праці в ході створення програмного забезпечення є достатньо складним через творчий характер роботи програміста. Відповідно трудомісткість розробки програмного забезпечення може бути розрахована на основі системи декількох моделей з різною точністю оцінки.

Трудомісткість розробки програмного забезпечення можна розрахувати за формулою:

$$t = t_o + t_u + t_a + t_n + t_{\text{отл}} + t_{\text{д}}\text{, людино-годин,} \qquad (4.1)$$

де t_o – витрати праці на підготовку й опис поставленої задачі (приймається 50 людино-годин);

t_u – витрати праці на дослідження алгоритму рішення задачі;

t_a – витрати праці на розробку блок-схеми алгоритму;

t_n – витрати праці на програмування по готовій блок-схемі;

$t_{\text{отл}}$ – витрати праці на налагодження програми на ЕОМ;

$t_д$ – витрати праці на підготовку документації.

Складові витрати праці визначаються через умовне число операторів у програмному забезпеченні, яке розробляється.

Умовне число операторів визначається за формулою:

$$Q = q \cdot C \cdot (1 + p), \tag{4.2}$$

де q – передбачуване число операторів (1027);

C – коефіцієнт складності програми (1,4);

p – коефіцієнт корекції програми в ході її розробки (0,05).

Підставивши у формулу (4.2) початкові дані, отримуємо умовне число операторів в програмі:

$$Q = 1027 \cdot 1{,}3 \cdot (1 + 0{,}05) = 1401{,}86$$

Витрати праці на вивчення опису задачі t_u визначаються з урахуванням уточнення опису і кваліфікації програміста:

$$t_u = \frac{Q \cdot B}{(75..85) \cdot k}, \text{ людино-годин}, \tag{4.3}$$

де B – коефіцієнт збільшення витрат праці внаслідок неповного та/або недостатнього опису задачі;

k – коефіцієнт кваліфікації програміста, обумовлений стажем роботи на даній спеціальності. При стажі роботи від 3 до 5 років він складає 1,2.

З урахуванням початкових даних, використавши формулу (4.3) отримуємо витрати праці на вивчення опису завдання:

$$t_u = \frac{1401{,}86 \cdot 1{,}3}{75 \cdot 1{,}2} = 20{,}25, \text{ людино-години},$$

Витрати праці на розробку алгоритму рішення задачі визначаються за наступною формулою:

$$t_a = \frac{Q}{(20...25)\cdot k}, \text{ людино-годин,} \quad (4.4)$$

де Q – умовне число операторів програми;

k – коефіцієнт кваліфікації програміста.

Підставивши відповідні значення в формулу (4.4), отримуємо:

$$t_a = \frac{1401,86}{20\cdot 1,2} = 58,41 \text{ людино-годин.}$$

Витрати на складання програми по готовій схемі становлять:

$$t_n = \frac{Q}{(20...25)\cdot k}, \text{ людино-годин,} \quad (4.5)$$

Підставивши відповідні значення в формулу (4.5), отримуємо:

$$t_n = \frac{1401,86}{25\cdot 1,2} = 46,73, \text{ людино-годин,}$$

Витрати праці на налагодження програми на ЕОМ:

– за умови автономного налагодження одного завдання:

$$t_{\text{отл}} = \frac{Q}{(4...5)\cdot k}, \text{ людино-годин,} \quad (4.6)$$

– за умови комплексного налагодження завдання:

$$t_{\text{отл}}^{k} = 1,5 \cdot t_{\text{отл}}, \text{ людино-години,} \quad (4.7)$$

З формули (4.6) виходить:

$$t_{\text{отл}} = \frac{1401,86}{5 \cdot 1,2} = 233,64, \text{ людино-години,}$$

Підставивши значення в формулу (4.7), отримуємо:

$$t_{\text{отл}}^{k} = 1,5 \cdot 250,25 = 350,47, \text{ людино-години,}$$

Витрати праці на підготовку документації визначаються за формулою:

$$t_{\text{д}} = t_{\text{др}} \cdot t_{\text{до}}, \text{ людино-годин,} \quad (4.8)$$

де $t_{др}$-трудомісткість підготовки матеріалів і рукопису:

$$t_{\text{др}} = \frac{Q}{(15..20) \cdot k}, \text{ людино-годин,} \quad (4.9)$$

$t_{до}$ - трудомісткість редагування, печатки й оформлення документації:

$$t_{\text{до}} = 0,75 \cdot t_{\text{др}}, \text{ людино-годин,} \quad (4.10)$$

Підставляючи відповідні значення в формули (4.8-4.10) відповідно отримаємо:

$$t_{\text{др}} = \frac{1401,86}{18 \cdot 1,2} = 64,9, \text{ людино-годин,}$$

$$t_{\text{до}} = 0,75 \cdot 69,5 = 48,68, \text{ людино-годин,}$$

$$t_{\text{д}} = 69,5 + 52,13 = 113,58, \text{ людино-годин,}$$

Повертаючись до формули (4.1), отримаємо повну оцінку трудомісткості розробки програмного забезпечення:

$$t_д = 50 + 20{,}25 + 58{,}41 + 46{,}73 + 350{,}47 + 113{,}58 = 639{,}44 \approx 639,$$

людино-годин,

4.2. Розрахунок витрат на створення програмного забезпечення

Витрати на створення програмного забезпечення $K_{ПО}$ включають витрати на заробітну плату виконавця програми $З_{ЗП}$ і витрати машинного часу, необхідного на налагодження програми на ЕОМ:

$$K_{ПО} = З_{ЗП} + З_{МВ}, \text{ грн.} \quad (4.11)$$

Заробітна плата виконавців розраховується за наступною формулою:

$$З_{ЗП} = t \cdot C_{ПР}, \text{ грн} \quad (4.12)$$

де t – загальна трудомісткість, людино-годин;

$C_{ПР}$ – середня годинна заробітна плата програміста, грн/година.

Підставивши дані у формулу (4.12) отримуємо:

$$З_{ЗП} = 639{,}44 \cdot 200 = 127\,888 \text{ грн.}$$

Вартість машинного часу, необхідного для налагодження програми на ЕОМ, визначається за формулою:

$$З_{МВ} = t_{отл} \cdot C_{мч}, \text{ грн.} \quad (4.13)$$

де $t_{отл}$ – трудомісткість налагодження програми на ЕОМ, год;

$C_{мч}$ – вартість машино-години ЕОМ, грн/год (12 грн/год).

Вартість необхідного для налагодження машинного часу за (3.13):

$$З_{МВ} = 350,47 \cdot 12 = 4205,64, \text{ грн}$$

Звідси витрати на створення програмного продукту:

$$К_{ПО} = 127\,888 + 4205,64 \approx 132093 \text{ грн.}$$

Очікуваний період створення програмного забезпечення:

$$T = \frac{t}{B_k \cdot F_p}, \text{ міс,}$$

де B_k – число виконавців (дорівнює 1);

F_p – місячний фонд робочого часу (при 40 годинному робочому тижні F_p=176 годин).

Звідси очікуваний період створення програмного забезпечення дорівнює:

$$T = \frac{639}{1 \cdot 176} \approx 3,6, \text{ міс,}$$

4.3. Маркетингові дослідження ринку збуту розробленого програмного продукту

Так як тема кваліфікаційної роботи присвячена дослідженням, пов’язаним з когнітивним моделюванням та представленням знань, то можна зробити висновок, що робота комерційних перспектив не має, тому маркетингові дослідження проведені не були.

4.4. Оцінка економічної ефективності впровадження програмного забезпечення

Оскільки виконана робота має виключно дослідницький характер і не передбачає створення кінцевого продукту, то чисельний розрахунок економічної ефективності виконаний бути не може.

Тим не менш результати роботи можуть бути використані під час створення ефективних автоматизованих онлайн систем освіти, що дозволить досягти соціального ефекту у вигляді підвищення якості освітніх сервісів та вивільненні часу викладачів від рутинного тренінгу задля забезпечення їм можливості створення нових більш якісних освітніх матеріалів та вдосконалення створених раніше.

4.5. Висновок до розділу

У даній кваліфікаційній роботі було проведене дослідження для вибору моделі представлення знань за результатами когнітивного моделювання в системі дистанційної освіти. У економічному розділі були визначені витрати на реалізацію даного програмного забезпечення. Трудомісткість склала 639 людино-годин. Вартість даного продукту склала 132 093 грн. Очікуваний час створення – 3,6 місяця.

ВИСНОВКИ

В ході виконання кваліфікаційної роботи було розглянуто існуючі моделі представлення знань, їх інструментарій, особливості застосування, переваги та недоліки. Ці дані було проаналізовано та обрано найбільш прийнятну модель – онтологію. Було обґрунтовано вибір онтологічної моделі представлення знань студента у системі дистанційної освіти. Основними причинами вибору даної моделі стали її виразність, здатність відображати складні зв'язки, можливість визначення своїх типів зв'язків та потенціал до реалізації ефективного автоматизованого механізму висновку. Також онтологічна модель поєднує у собі декілька більш простих моделей, таких як фрейми, семантичні мережі, продукційні правила, логічне представлення, у такій комбінації, що дозволяє будувати дуже гнучкі онтологічні мережі, які легко розширювати та підтримувати. А оскільки представлення знань студента – це змінювана структура, яка з часом буде нарощувати об'єм, то онтологічне представлення знань є прийнятним.

Був розглянутий інструментарій, який використовується для побудови онтологій. В результаті аналізу була обрана мова OWL, яка на сьогодні є найбільш повним засобом для створення онтологій і має найбільшу спільноту.

Була описана архітектура системи дистанційної освіти, типи і характер інформації, які будуть зберігатись у даній системі, та процес навчання на основі фіксації когнітивних навичок та онтології знань студента і порівняння їх з ідеальними для певної предметної області.

Для демонстрації процесу створення онтології була розроблена тестова еталонна онтологія з предметної області вивчення мов програмування, фрагмент якої наведений у роботі. Для розробки онтології мовою OWL був використаний такий спеціалізований інструмент як Protégé.

В ході роботи над проектом також була виконана розробка фрагменту системи дистанційної освіти за допомогою мови Python і фреймворків Django та Django Rest Framework.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Csikszentmihalyi M. Flow: The psychology of optimal experience / Csikszentmihalyi M. – New York: Harper Perennial, 1990. – URL: https://core.ac.uk/download/42629223.pdf (дата звернення: 20.06.2021).
2. Toward A Unified Modeling of Learner's Growth Process and Flow Theory / [Challco, G. & Andrade, F. & Borges et al.] // Educational Technology & Society. – 2016. – 19(2). – P. 215-227. – URL: https://www.researchgate.net/publication/301233159_Toward_A_Unified_Modeling_of_Learner's_Growth_Process_and_Flow_Theory (дата звернення: 27.06.2021).
3. Tolman, E. C. Congitive Maps in Rats and Men / Tolman, E. C. // Image and Environment: Cognitive Mapping and Spatial Behavior. – Chapter 2. – P. 27.
4. Akers, S. B. A graphical approach to production scheduling problems / Akers, S. B. // Operations Research. – 1956. – Vol. 4, issue 2. – P. 244–245.
5. Quillian, M. R. Word concepts: A theory and simulation of some basic semantic capabilitie. / Quillian, M. R. // Behavioral Science. – 1967. – 12(5). – P. 410–430.
6. Sowa, J. F. Conceptual graphs for a database interface / Sowa, J. F. // IBM Journal of Research and Development. – 1976. – 20:4. – P. 336-357.
7. Colmerauer, A. The birth of Prolog / Colmerauer, A., Roussel, P. // November, – 1992. – URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.85.7438&rep=rep1&type=pdf (дата звернення 03.07.2021).
8. Novak, J. D. Learning, Creating, and Using Knowledge: Concept maps as facilitative tools in schools and corporations / Novak, J. D. // Journal of e-Learning and Knowledge Society. – 2010. – Vol. 6, n. 3. – P. 21 - 30.
9. Pearl, J. Bayesian Networks / Pearl, J., Russell, S. // Lecture notes, University of California. – URL:

https://www.cs.ubc.ca/~murphyk/Teaching/CS532c_Fall04/Papers/hbtnn-bn.pdf (дата звернення: 14.07.2021).

10. Sowa, J. F. Semantic Networks. In Stuart C Shapiro. Encyclopedia of Artificial Intelligence / Sowa, J. F. – Wiley, New York. – 1987. – URL: http://www.jfsowa.com/pubs/semnet.htm (дата звернення: 20.07.2021).
11. Sowa, J. F. Principles of Semantic Networks. / Sowa, J. F. SanMateo. – 1992.
12. Kosko, B. Fuzzy cognitive maps / Kosko, B. // International Journal of Man-Machine Studies. – 1986. – Vol. 24, Issue 1. – P. 65-75. – URL: https://www.sciencedirect.com/science/article/abs/pii/S0020737386800402 (дата звернення: 17.07.2021).
13. Berners-Lee, T. The Semantic Web / Berners-Lee, T., Hendler, J., Lassila, O. // Scientific American Magazine. – 2001.
14. Web Ontology Language (OWL). *W3C Semantic Web* : website. – URL: https://www.w3.org/OWL/ (дата звернення: 23.07.2021).
15. Singhal, A. Introducing the Knowledge Graph: Things, Not Strings. / Singhal, A. // Official Blog (of Google). 2012. – URL: https://blog.google/products/search/introducing-knowledge-graph-things-not/ (дата звернення 24.07.2021).
16. Facebook Graph Search. *Wikipedia* : website. – URL: https://en.wikipedia.org/wiki/Facebook_Graph_Search (дата звернення 04.08.2021).
17. Facebook Announces Its Third Pillar “Graph Search” That Gives You Answers, Not Links Like Google. *TechCrunch* : website. – URL: https://techcrunch.com/2013/01/15/facebook-announces-its-third-pillar-graph-search/ (дата звернення 05.08.2021).
18. Malhotra, M. Evolution of Knowledge Representation and Retrieval Techniques. International Journal of Intelligent Systems and Applications. / Malhotra, M., Gopalakrishnan Nair, T.R // 2015. Vol 07. P.8-28. – URL: https://www.researchgate.net/publication/280578562_Evolution_of_Knowledge_Representation_and_Retrieval_Techniques (дата звернення 05.08.2021).

19. W3C Workshop on Web Standardization for Graph Data. *W3C* : website. – URL: https://www.w3.org/Data/events/data-ws-2019/ (дата звернення 05.08.2021).
20. Sun, R. The Cambridge Handbook of Computational Psychology / Sun, R. – New York: Cambridge University Press. – 2008.
21. Hobbes, T. Elements of Law, Natural and Political / Hobbes, T. // Routledge. – 1969.
22. Pitt, D. Mental Representation / Pitt, D. The Stanford Encyclopedia of Philosophy. – Spring 2020 Edition. – Edward N. Zalta (ed.). – URL: https://plato.stanford.edu/archives/spr2020/entries/mental-representation/ (дата звернення 05.08.2021).
23. Buckner, C. Connectionism / Buckner, C. and Garson J. // The Stanford Encyclopedia of Philosophy. – Fall 2019 Edition. – Edward N. Zalta (ed.). – URL: https://plato.stanford.edu/archives/fall2019/entries/connectionism/(дата звернення 14.08.2021).
24. Constructivist Learning Theory. *Educational Technology* : website. – URL: https://educationaltechnology.net/constructivist-learning-theory/ (дата звернення 14.08.2021).
25. Graham, G. Behaviorism / Graham, G. The Stanford Encyclopedia of Philosophy. – Spring 2019 Edition. – Edward N. Zalta (ed.). URL: https://plato.stanford.edu/archives/spr2019/entries/behaviorism/ (дата звернення 15.08.2021).
26. Marr, D. Vision: A computational investigation into the human representation and processing of visual information / Marr, D. – 1982. – San Francisco. – URL: http://www.contrib.andrew.cmu.edu/~kk3n/80-300/marr2.pdf (дата звернення 22.08.2021).
27. Davis, R. What is a Knowledge Representation? / Davis, R., Shrobe, H., and Szolovits P. // AI Magazine. – 14(1):17-33, 1993. – URL:

https://groups.csail.mit.edu/medg/ftp/psz/k-rep.html#kr (дата звернення 01.09.2021).

28. Crisp-Bright, A.K. The Effects of Domain and Type of Knowledge on Category-Based Inductive Reasoning / Crisp-Bright, A.K., Feeney, A. – 2010. – Memory. – P. 67-72.
29. Wang, Y. On Cognitive Informatics / Wang, Y. – 2003. – Brain and Mind, 4. – P. 151-167.
30. Логіка висловлювань. *Вікіпедія*: веб-сайт. – URL: https://uk.wikipedia.org/wiki/%D0%9B%D0%BE%D0%B3%D1%96%D0%BA%D0%B0_%D0%B2%D0%B8%D1%81%D0%BB%D0%BE%D0%B2%D0%BB%D1%8E%D0%B2%D0%B0%D0%BD%D1%8C (дата звернення 21.09.2021).
31. First-order logic. *Wikipedia* : website. – URL: https://en.wikipedia.org/wiki/First-order_logic (дата звернення 21.09.2021).
32. Федорук П.І. Модель представлення знань в адаптивній системі дистанційного навчання та контролю знань «EduPRO» / Федорук П.І., Масловський С.М. // Штучний інтелект, 3. – 2011. – Прикарпатський національний університет ім. Василя Стефаника, м. Івано-Франківськ, Україна. – С. 463-472.
33. Lenko V. Knowledge representation model / Lenko V., Pasichnyk V., Shcherbyna Y. // – Вісник Національного університету "Львівська політехніка". – Комп’ютерні науки та інформаційні технології. – 2017. – № 864. – С. 157-168. – URL: http://nbuv.gov.ua/UJRN/VNULPKNIT_2017_864_23 (дата звернення 29.09.2021).
34. Kiely, K. Cognitive function/ Kiely, K. – In Michalos. – Encyclopedia of Quality of Life and Well-Being Research. – 2014. – Springer. – P. 974–978.
35. What are cognitive abilities and skills, and can we boost them? Sharpbrains : *website*. – URL: https://sharpbrains.com/what-are-cognitive-abilities/.

36. What are cognitive skills? LearningRx : *website*. – URL: https://www.learningrx.com/what-is-brain-training-/what-are-cognitive-skills-/.
37. Cognitive Skills: What They Are, Why They Matter, How to Improve Them? Edublox Tutor : *website*. – URL: https://www.edubloxtutor.com/what-are-cognitive-skills/.
38. Bloom, B.S. Taxonomy of Educational Objectives, Handbook: The Cognitive Domain / Bloom, B.S. – 1956. – David McKay, New York.
39. Dalton, J. Extending Children's Special Abilities – Strategies for primary classrooms / Dalton, J. & Smith, D. – 1986. – Curriculum Branch, Schools Division, Melbourne.
40. Aisbett, J. Cognitive Classification / Aisbett, J., Gibbon G. – 1999. – School of Information Technology The University of Newcastle, University Drive, Callaghan.
41. Anderson, L. W. A Taxonomy for Learning, Teaching, and Assessing: A Revision of Bloom's Taxonomy of Educational Objectives / Anderson, L. W., Krathwohl, D. R. – 2001. – Allyn & Bacon. – Boston, MA (Pearson Education Group).
42. Atkinson, R. C. Human memory: A proposed system and its control processes / Atkinson, R. C., Shiffrin, R. M. – In K. W. Spence & J. T. Spence, The psychology of learning and motivation: II. – Academic Press. – 1968.
43. Longo, F. Learning and Memory: How it Works and When it Fails. Stanford. *Youtube* : website. URL: https://www.youtube.com/watch?v=a_HfSnQqeyY (дата звернення 08.10.2021).
44. Miller, G. The magical number seven, plus or minus two: Some limits on our capacity for processing information / Miller, G. // The psychological review, 63. – 1956. – P. 81-97.
45. Guarino, N. Formal Ontology and Information Systems / Guarino, N. // In: Proc. of FOIS 1998, Trento, Italy. – IOS Press, Amsterdam. – 1998.
46. Gruber, T.R. A Translation Approach to Portable Ontology Specifications / Gruber, T.R. // Knowledge Acquisition 5. – 1993.

47. Ontology language. *Wikipedia* : website. – URL: http://en.wikipedia.org/wiki/Ontology_language (дата звернення 18.09.2021).
48. Berners-Lee, T. The Semantic Web as a language of logic. – 2009. – URL: http://www.w3.org/DesignIssues/Logic.html (дата звернення 28.10.2021).
49. Gutierrez-Pulido, J.R. Ontology languages for the semantic web: A never completely updated review / [Gutierrez-Pulido, J.R., Ruiz, M.A.G., Herrera, R., et al.] // Knowl.-Based Syst. – 2006.
50. Su, X. A Comparative Study of Ontology Languages and Tools / Su, X., Ilebrekke, L. // In: Pidduck, A.B., Mylopoulos, J., Woo, C.C., Ozsu, M.T. (eds.) – CAiSE 2002. – LNCS, vol. 2348. – P. 761–765. – Springer, Heidelberg. – 2002.
51. Genesereth, M.R. Knowledge Interchange Format (KIF) / Genesereth, M.R. // Draft proposed American National Standard (dpANS), NCITS.T2/98-004. – Stanford University. – 2004. – URL: http://logic.stanford.edu/kif/dpans.html (дата звернення 29.10.2021).
52. Patil, R.S. An Overview of the DARPA Knowledge Sharing Effort / [Patil, R.S., Fikes, R.E., Patel-Schneider, P.F. et al.] // In: Huhns, M.N., Singh, M.P. (eds.) – KR 1992. – P. 243–254. – Morgan Kaufmann Publishers Inc., San Francisco. – 1992.
53. Resource Description Framework (RDF). *W3C*: website. – URL: https://www.w3.org/RDF/ (дата звернення 01.11.2021).
54. Kothari, C.R. Relation Elements for the Semantic Web / Kothari, C.R., Russomanno, D.J. // LNCS (LNAI), vol. 3127. – P. 275–286. – Springer, Heidelberg. – 2004.
55. Broekstra, J. Enabling knowledge representation on the Web by extending RDF Schema / [Broekstra, J., Klein, M.C.A., Decker, S. et al.]. – Computer Networks. – 2002.
56. Fensel, D. OIL: An Ontology Infrastructure for the Semantic Web / [Fensel, D., Harmelen, F.V., Horrocks et al.]. – IEEE Intelligent Systems. – 2001.

57. About the DAML Language. *Web Archive* : website. – URL: https://web.archive.org/web/20030207042906/http://www.daml.org/about.html (дата звернення 11.11.2021).
58. Horrocks, I. DAML+OIL: a Description Logic for the Semantic Web // Horrocks, I. / IEEE Data Eng. Bull. – 2002.
59. Web Ontology Language (OWL). *W3C*: website. – URL: https://www.w3.org/2001/sw/wiki/OWL (дата звернення 16.10.2021).
60. Antoniou, G. Web Ontology Language: OWL / Antoniou, G., Harmelen, F.V. // Handbook on Ontologies. – 2004.
61. Berendt, B. A. Roadmap for Web Mining: From Web to Semantic Web / [Berendt, B., Hotho, A., Mladenič, D., et al.] // LNCS (LNAI), vol. 3209. – P. 1–22. – Springer, Heidelberg. – 2004.
62. A free, open-source ontology editor and framework for building intelligent systems. *Protégé* : website. – URL: https://protege.stanford.edu/ (дата звернення 21.11.2021).
63. Методичні рекомендації до виконання кваліфікаційних робіт здобувачами другого (магістерського) рівня вищої освіти спеціальностей 121 «Інженерія програмного забезпечення» та 122 «Комп'ютерні науки» / Б.І. Мороз, О.В. Іванченко, О.В. Реута, О.С. Шевцова; М-во освіти і науки України, Нац. техн. ун-т "Дніпровська політехніка". – Дніпро : НТУ «ДП», 2021.

ДОДАТОК А

КОД ОНТОЛОГІЇ

reference_ontology_programming.owl

```
<?xml version="1.0"?>
<rdf:RDF    xmlns="http://www.semanticweb.org/dell/ontologies/2022/0/programing-
reference-ontology#"
     xml:base="http://www.semanticweb.org/dell/ontologies/2022/0/programing-
reference-ontology"
     xmlns:owl="http://www.w3.org/2002/07/owl#"
     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
     xmlns:xml="http://www.w3.org/XML/1998/namespace"
     xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
     xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#">
    <owl:Ontology
rdf:about="http://www.semanticweb.org/dell/ontologies/2022/0/programing-
reference-ontology"/>


    <!--

///////////////////////////////////////////////////////////////////////////////
///////
    //
    // Object Properties
    //

///////////////////////////////////////////////////////////////////////////////
///////
     -->



    <!--    http://www.semanticweb.org/dell/ontologies/2022/0/untitled-ontology-
2#depends -->

    <owl:ObjectProperty
rdf:about="http://www.semanticweb.org/dell/ontologies/2022/0/untitled-ontology-
2#depends">
        <rdfs:domain
rdf:resource="http://www.semanticweb.org/dell/ontologies/2022/0/untitled-
ontology-2#Symbol_code"/>
        <rdfs:range
rdf:resource="http://www.semanticweb.org/dell/ontologies/2022/0/untitled-
ontology-2#Symbol"/>
    </owl:ObjectProperty>



    <!--    http://www.semanticweb.org/dell/ontologies/2022/0/untitled-ontology-
2#has_a -->

    <owl:ObjectProperty
rdf:about="http://www.semanticweb.org/dell/ontologies/2022/0/untitled-ontology-
2#has_a">
```

```
        <rdfs:domain
rdf:resource="http://www.semanticweb.org/dell/ontologies/2022/0/untitled-
ontology-2#Value"/>
        <rdfs:range
rdf:resource="http://www.semanticweb.org/dell/ontologies/2022/0/untitled-
ontology-2#Type"/>
    </owl:ObjectProperty>


    <!--

///////////////////////////////////////////////////////////////////////////
///////
    //
    // Classes
    //

///////////////////////////////////////////////////////////////////////////
///////
     -->



    <!--    http://www.semanticweb.org/dell/ontologies/2022/0/untitled-ontology-
2#Code_table -->

    <owl:Class
rdf:about="http://www.semanticweb.org/dell/ontologies/2022/0/untitled-ontology-
2#Code_table"/>



    <!--    http://www.semanticweb.org/dell/ontologies/2022/0/untitled-ontology-
2#Constant -->

    <owl:Class
rdf:about="http://www.semanticweb.org/dell/ontologies/2022/0/untitled-ontology-
2#Constant"/>



    <!--    http://www.semanticweb.org/dell/ontologies/2022/0/untitled-ontology-
2#Fact -->

    <owl:Class
rdf:about="http://www.semanticweb.org/dell/ontologies/2022/0/untitled-ontology-
2#Fact">
        <rdfs:subClassOf
rdf:resource="http://www.semanticweb.org/dell/ontologies/2022/0/untitled-
ontology-2#Value"/>
    </owl:Class>



    <!--    http://www.semanticweb.org/dell/ontologies/2022/0/untitled-ontology-
2#Immutability -->

    <owl:Class
rdf:about="http://www.semanticweb.org/dell/ontologies/2022/0/untitled-ontology-
2#Immutability"/>
```

```
    <!--    http://www.semanticweb.org/dell/ontologies/2022/0/untitled-ontology-
2#Integer -->

    <owl:Class
rdf:about="http://www.semanticweb.org/dell/ontologies/2022/0/untitled-ontology-
2#Integer">
        <rdfs:subClassOf
rdf:resource="http://www.semanticweb.org/dell/ontologies/2022/0/untitled-
ontology-2#Number"/>
    </owl:Class>



    <!--    http://www.semanticweb.org/dell/ontologies/2022/0/untitled-ontology-
2#Number -->

    <owl:Class
rdf:about="http://www.semanticweb.org/dell/ontologies/2022/0/untitled-ontology-
2#Number">
        <rdfs:subClassOf
rdf:resource="http://www.semanticweb.org/dell/ontologies/2022/0/untitled-
ontology-2#Value"/>
    </owl:Class>



    <!--    http://www.semanticweb.org/dell/ontologies/2022/0/untitled-ontology-
2#Real -->

    <owl:Class
rdf:about="http://www.semanticweb.org/dell/ontologies/2022/0/untitled-ontology-
2#Real">
        <rdfs:subClassOf
rdf:resource="http://www.semanticweb.org/dell/ontologies/2022/0/untitled-
ontology-2#Number"/>
    </owl:Class>



    <!--    http://www.semanticweb.org/dell/ontologies/2022/0/untitled-ontology-
2#Symbol -->

    <owl:Class
rdf:about="http://www.semanticweb.org/dell/ontologies/2022/0/untitled-ontology-
2#Symbol">
        <rdfs:subClassOf
rdf:resource="http://www.semanticweb.org/dell/ontologies/2022/0/untitled-
ontology-2#Value"/>
    </owl:Class>



    <!--    http://www.semanticweb.org/dell/ontologies/2022/0/untitled-ontology-
2#Symbol_code -->

    <owl:Class
rdf:about="http://www.semanticweb.org/dell/ontologies/2022/0/untitled-ontology-
2#Symbol_code">
        <rdfs:subClassOf
rdf:resource="http://www.semanticweb.org/dell/ontologies/2022/0/untitled-
ontology-2#Integer"/>
    </owl:Class>
```

```
    <!--    http://www.semanticweb.org/dell/ontologies/2022/0/untitled-ontology-
2#Symbols_set -->

    <owl:Class
rdf:about="http://www.semanticweb.org/dell/ontologies/2022/0/untitled-ontology-
2#Symbols_set"/>



    <!--    http://www.semanticweb.org/dell/ontologies/2022/0/untitled-ontology-
2#Type -->

    <owl:Class
rdf:about="http://www.semanticweb.org/dell/ontologies/2022/0/untitled-ontology-
2#Type"/>



    <!--    http://www.semanticweb.org/dell/ontologies/2022/0/untitled-ontology-
2#Value -->

    <owl:Class
rdf:about="http://www.semanticweb.org/dell/ontologies/2022/0/untitled-ontology-
2#Value"/>



    <!--    http://www.semanticweb.org/dell/ontologies/2022/0/untitled-ontology-
2#Variability -->

    <owl:Class
rdf:about="http://www.semanticweb.org/dell/ontologies/2022/0/untitled-ontology-
2#Variability"/>



    <!--    http://www.semanticweb.org/dell/ontologies/2022/0/untitled-ontology-
2#Variable -->

    <owl:Class
rdf:about="http://www.semanticweb.org/dell/ontologies/2022/0/untitled-ontology-
2#Variable"/>
</rdf:RDF>

…
```

Продовження коду приведене на диску.

ДОДАТОК Б

КОД ПРОГРАМИ

/custom_auth/view.py

```
class ProfileAPI(generics.GenericAPIView):
    serializer_class = ProfileSerializer
    permission_classes = [permissions.AllowAny]
    queryset = Profile.objects.all()

    def get(self, request):
        user = request.user
        if not user:
            return Response({'status': '401', 'message': 'Unauthorized'},
status=401)
        else:
            return
Response(self.get_serializer(self.get_queryset().get(user=user)).data,
status=200)

    def put(self, request):
        try:
            with transaction.atomic():
                if 'user' not in request.data:
                    raise 'Missing required parameters'
                user_data = request.data['user']

                # user data block
                email = user_data.get('email')
                password = user_data.get('password')
                confirm_pass = user_data.get('confirm_password')

                if email is None or password is None or confirm_pass is None:
                    raise 'Missing required parameters'

                if password != confirm_pass:
                    raise 'Passwords mismatch'

                # user data block
                first_name = request.data.get('first_name')
                last_name = request.data.get('last_name')

                if first_name is None or last_name is None:
                    raise 'Missing required parameters'

                serializer = ProfileSerializer(data=request.data)
                serializer.is_valid(raise_exception=True)
                serializer.save()

                user = copy.copy(request.data['user'])

                request.data.clear()
                request.data.update({"email": user['email']})
                request.data.update({"password": user['password']})

                token_obtain_pair_serializer =
TokenObtainPairSerializer(data=request.data)
                token_obtain_pair_serializer.is_valid()
```

```
        except Exception as e:
            return Response({'status': 'error', 'message': f'Profile creation
error {e}'}, status=500)

        return Response(token_obtain_pair_serializer.validated_data, status=201)

    def post(self, request):
        try:
            with transaction.atomic():
                if not request.user.is_authenticated:
                    raise 'User not authenticated'

                user = UserSerializer(request.user)

                request.data['user'] = user.data

                profile = ProfileSerializer(request.user.profile,
data=request.data)
                profile.is_valid(raise_exception=True)
                profile.save()

        except Exception as e:
            return Response({'status': 'error', 'message': f'Profile update
error {e}'}, status=500)
        return Response(profile.validated_data, status=200)


class ChangePasswordAPI(generics.GenericAPIView):

    def post(self, request):
        data = request.data

        if not data.get('password') or not data.get('confirm_password') or not
data.get('previous_password'):
            return Response({'status': 'error', 'message': 'Missing required
params'}, status=400)

        if data.get('password') != data.get('confirm_password'):
            return Response({'status': 'error', 'message': 'Passwords
mismatch'}, status=400)

        if not request.user.check_password(data.get('previous_password')):
            return Response({'status': 'error', 'message': 'Password is
invalid'}, status=400)

        if data.get('password') == data.get('previous_password'):
            return Response({'status': 'error', 'message': 'New password and old
password should not be same'},
                            status=400)

        try:
            request.user.set_password(data.get('password'))
            request.user.save()
        except Exception as e:
            return Response({'status': 'error', 'message': f'Error with changing
password:{e}'}, status=400)

        return Response({'status': 'success', 'message': 'password successfully
changed'}, status=200)
…
```

Продовження коду приведене на диску.

/custom_auth/serializers.py

```
class ProfileUserDataSerializer(serializers.ModelSerializer):
    class Meta:
        model = CustomUser
        fields = ['roles', 'email', 'password']
        extra_kwargs = {
            'email': {
                'validators': [],
            }
        }

    def to_representation(self, instance):
        return instance.as_json()


class ProfileSerializer(serializers.ModelSerializer):
    user = ProfileUserDataSerializer()

    class Meta:
        model = Profile
        fields = '__all__'

    def create(self, validated_data):
        user = CustomUser.objects.create_user(
            validated_data['user']['email'],
            validated_data['user']['password'],
            is_active=True
        )
        if 'second_name' not in validated_data:
            validated_data['second_name'] = None

        if 'phone' not in validated_data:
            validated_data['phone'] = None

        profile = Profile.objects.create(
            first_name=validated_data['first_name'],
            second_name=validated_data['second_name'],
            last_name=validated_data['last_name'],
            phone=validated_data['phone'],
            user=user
        )
        return profile

    def update(self, instance, validated_data):
        instance.first_name = validated_data['first_name']
        instance.second_name = validated_data['second_name']
        instance.last_name = validated_data['last_name']
        instance.phone = validated_data['phone']
        instance.save()
        return instance


class UserSerializer(serializers.ModelSerializer):
    class Meta:
        model = CustomUser
        fields = ['email', 'password']
        read_only_fields = tuple()
…
```

Продовження коду приведене на диску.

/courses/views.py

```
class CourseAPI(generics.GenericAPIView):
    permission_classes = [permissions.AllowAny]
    serializer_class = CourseSerializer
    queryset = Course.objects.all()
    filter_backends = [filters.DjangoFilterBackend]
    filterset_class = CourseFilter
    parser_classes = [JSONParser, MultiPartParser]

    def update_table_of_content(self, course, should_table_updated):
        if should_table_updated:
            create_table_of_content(course)
    def get(self, request):
        return
Response(self.get_serializer(self.filter_queryset(self.get_queryset()),
many=True).data, status=200)

    def post(self, request):
        if 'difficulty' in request.data and (
                request.data.get('difficulty') is None or
request.data.get('difficulty') == ""):
            return Response({'status': 'error', 'message': 'Difficulty is should
not be empty'}, status=400)

        if 'price' in request.data and (request.data.get('price') is None or
request.data.get('price') == ""):
            return Response({'status': 'error', 'message': 'Price is should not
be empty'}, status=400)

        if 'rating' in request.data and (request.data.get('rating') is None or
request.data.get('rating') == ""):
            return Response({'status': 'error', 'message': 'Rating is should not
be empty'}, status=400)

        if 'is_deleted' in request.data and (
                request.data.get('is_deleted') is None or
request.data.get('is_deleted') == ""):
            return Response({'status': 'error', 'message': 'is_deleted is should
not be empty'}, status=400)
        try:
            with transaction.atomic():
                is_topic_updated = False
                if len(request.data.get("entities_to_delete", [])) != 0:
                    for entity in request.data.get("entities_to_delete"):
                        entity_uuid = entity.get("uuid")
                        type = entity.get("type")

                        if not entity_uuid or not type:
                            raise Exception('Uuid and type are required
parameters to delete entity')
                        model = entities_to_delete_mapping.get(type)
                        if not model:
                            raise Exception(f"Unsupported entity {type}")

                        model.objects.get(uuid=entity_uuid).delete()

                course = Course.objects.get(uuid=request.data.get('uuid'))
…
```

Продовження коду приведене на диску.

/courses/serializers.py

```
class MediaFileContentSerializer(serializers.ModelSerializer):
    media_file = MediaFileSerializer()

    class Meta:
        model = MediaFileContent
        exclude = ['content']


class MediaFileContentDeserializer(serializers.ModelSerializer):
    class Meta:
        model = MediaFileContent
        fields = '__all__'

    def create(self, validated_data):
        return MediaFileContent.objects.create(
            language=validated_data['language'],
            content=validated_data['content'],
            media_file=validated_data['media_file']
        )


class TopicTitleSerializer(serializers.ModelSerializer):
    class Meta:
        model = TopicTitle
        exclude = ['topic']


class TopicTitleDeserializer(serializers.ModelSerializer):
    class Meta:
        model = TopicTitle
        fields = "__all__"

    def create(self, validated_data):
        return TopicTitle.objects.create(
            title=validated_data['title'],
            identifier=validated_data['identifier'],
            topic=validated_data['topic'],
            language=validated_data['language'],
        )


class ContentTitleSerializer(serializers.ModelSerializer):
    class Meta:
        model = ContentTitle
        exclude = ['content']


class ContentSerializer(serializers.ModelSerializer):
    content_titles = ContentTitleSerializer(many=True)
    media_contents = MediaFileContentSerializer(many=True)

    class Meta:
        model = Content
        exclude = ['topic']


class ContentDeserializer(serializers.ModelSerializer):
    class Meta:
```

```
        model = Content
        fields = '__all__'

    def create(self, validated_data):
        return Content.objects.create(
            topic=validated_data['topic'],
            last_changed=validated_data['last_changed']
        )


class TopicDeserializer(serializers.ModelSerializer):
    class Meta:
        model = Topic
        fields = "__all__"

    def create(self, validated_data):
        return Topic.objects.create(
            course=validated_data['course'],
            topic_level=validated_data['topic_level'],
            parent=validated_data['parent'],
            order=validated_data['order'],
        )
class CourseTitleSerializer(serializers.ModelSerializer):
    class Meta:
        model = CourseTitle
        exclude = ['course']


class CoursedDescriptionSerializer(serializers.ModelSerializer):
    class Meta:
        model = CourseDescription
        exclude = ['course']


class CourseAuthorsSerializer(serializers.ModelSerializer):
    class Meta:
        model = CourseAuthors
        exclude = ['course']


class CourseCoversSerializer(serializers.ModelSerializer):
    cover = MediaFileSerializer()

    class Meta:
        model = CourseCover
        exclude = ['course']


class CourseTableOfContentSerializer(serializers.ModelSerializer):
    class Meta:
        model = CourseTableOfContent
        exclude = ['course']
…
```

Продовження коду приведене на диску.

ДОДАТОК В

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
«ДНІПРОВСЬКА ПОЛІТЕХНІКА»

Факультет інформаційних технологій
Кафедра програмного забезпечення комп'ютерних систем

ВІДГУК

Керівника економічної частини Професора Вагонової О.Г.
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання, посада, місце роботи)

На магістерську роботу
Студента ІІ курсу групи 121м-20-1 Черечечі Ірини Володимирівни
(прізвище, ім'я, по батькові)

На тему: Обґрунтування моделі представлення знань за результатами когнітивного моделювання у системі дистанційної освіти.

«___»_____2022 р. _____

ДОДАТОК Г

ПЕРЕЛІК ДОКУМЕНТІВ НА ДИСКУ

Ім’я файла	**Опис**
Пояснювальні документи	
Диплом_Черечеча.doc	Пояснювальна записка роботи. Документ Word.
Диплом_Черечеча.pdf	Пояснювальна записка роботи в форматі PDF
Програма	
Program.rar	Архів. Містить коди створення онтології та коди програми системи дистанційної освіти.
Презентація	
Презентація_Черечеча.ppt	Презентація роботи