Київський Національний Університет імені Тараса Шевченко
Факультет комп'ютерних наук та кібернетики
Кафедра інтелектуальних програмних систем

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття ступеня магістра

за спеціальністю 121 Інженерія програмного забезпечення

на тему:

Мультиагентна інтелектуальна система управління навчальним процесом

Виконала студентка 2-го курсу магістратури:
Уложенко Марина Андріївна

Науковий керівник:
доцент, кандидат фіз.-мат. наук
Шкільняк Оксана Степанівна

Засвідчую, що в цій роботі немає запозичень з праць інших авторів без відповідних посилань

Роботу розглянуто й допущено до захисту на засіданні кафедри інтелектуальних програмних систем,
протокол №12 від 12.05.2021
Завідувач кафедри
професор О. І. Провотар

Київ – 2021

РЕФЕРАТ

Обсяг роботи 98 сторінок, 5 ілюстрацій, 90 джерел посилань.

Ключові слова: СИСТЕМА УПРАВЛІННЯ НАВЧАЛЬНИМ ПРОЦЕСОМ, ІНТЕЛЕКТУАЛЬНА СИСТЕМА УПРАВЛІННЯ НАВЧАЛЬНИМ ПРОЦЕСОМ, МУЛЬТИАГЕНТНА ІНТЕЛЕКТУАЛЬНА СИСТЕМА УПРАВЛІННЯ НАВЧАЛЬНИМ ПРОЦЕСОМ, НАВЧАЛЬНИЙ ПРОЦЕС, УПРАВЛІННЯ НАВЧАЛЬНИМ ПРОЦЕСОМ, МУЛЬТИАГЕНТНИЙ ПІДХІД, АГЕНТ.

Об'єктом роботи є управління навчальним процесом за допомогою використання мультиагентної інтелектуальної системи управління навчальним процесом. Предметом роботи є мультиагентна інтелектуальна система управління навчальним процесом.

Метою роботи є створення мультиагентної інтелектуальної системи управління навчальним процесом.

Методи розроблення: мультиагентний підхід, регресійний та кореляційний аналізи даних. Інструменти розроблення: динамічно типізована мова програмування Python, фреймворк для розробки агентів та мультиагентних систем Pade, бібліотека sklearn мови програмування Python.

Результати роботи: виконано аналіз основних особливостей, прикладів, переваг та недоліків класичних та інтелектуальних систем управління навчальним процесом; проаналізовано основні класифікації, типи та архітектури, особливості, переваги та недоліки агентів та мультиагентних систем; досліджено способи взаємодії агентів (комунікації та кооперації) та протоколи комунікації; проаналізовано основні переваги та недоліки використання інформаційних технологій в управлінні навчальним процесом; спроектовано та розроблено мультиагентну інтелектуальну систему управління навчальним процесом.

ЗМІСТ

ВСТУП

Оцінка сучасного стану об'єкта розробки. Правильно ораганізований навчальний процес забезпечує можливість комфортної та ефективної співпраці викладачів та студентів. У зв'язку з впровадженням інформаційних технологій у навчальний процес, почали широко використовуватись веб системи управління навчальним процесом. З розвитком технологій розвиваються і потреби, і побажання користувачів, які вже існуючі системи не можуть задовольнити. Тому в сфері систем для вищих навчальних закладів починають розвиватись інтелектуальні системи управління навчальним процесом, які надають користувачам про-активний, автономний та реактивний функціонал даної системи. Такі властивості систем, а саме про-активність, автономність та реактивність, є базовими принципами функціонування агентів та мультиагентного підходу. Тому для побудови інтелектуальної системи управління навчальним процесом почали використовувати зокрема і мультиагентний підхід, а для задоволення потреб користувачів у складній обробці великих масивів даних використовують відповідні до задачі розділи аналізу даних.

Актуальність роботи та підстави її виконання. Під час навчального процесу рівень знань, швидкість та якість сприйняття інформації, а також зацікавленість студентів у навчанні може відрізнятись не тільки серед студентів, а й у одного і того ж студента в залежності від часу та складності навчального матеріалу. Тому для підвищення рівня знань, які отримує студент, необхідно адаптувати навчальний процес під його потреби шляхом надання відповідних рекомендацій у вивченні навчальних матеріалів, освоєнні базових дисциплін для даного курс тощо. Окрім потреби в адаптивності навчального процесу, інтелектуальні системи управління навчальним процесом задовольняють такі актуальні потреби користувачів, як: відслідковування успішності та відвідуваності, зручні інструменти для роботи з навчальними матеріалами,

налагодження комунікації між учасниками навчального процесу, зберігання внесених у систему змін, автоматизація процесу створення звітів, моніторинг та інформування про екстраординарні ситуації тощо. Також позитивний ефект на якість навчального процесу надає доступність навчальних матеріалів для студентів в будь-який час та в будь-якому місці, оскільки таким чином вони зможуть повторно ознайомитись з інформацією.

Мета й завдання роботи. Метою кваліфікаційної роботи є створення мультиагентної системи управління навчальним процесом. Для досягнення даної цілі заплановано наступні задачі:

— аналіз існуючих
  - систем управління навчальним процесом;
  - інтелектуальних систем управління навчальним процесом;
— формулювання функціональних та нефункціональних вимог до системи;
— вибір типу архітектури агентів та мультиагентної системи;
— проектування архітектури агентів та системи;
— розробка інтерфейсу користувачів;
— реалізація системи;
— тестування системи.

Об'єкт, методи та засоби розроблення. Мультиагентна інтелектуальна система управління навчальним процесом є складною та багатокомпонентною системою, якій притаманна велика кількість обов'язкових та необов'язкових вимог. Тому розробці системи передував аналіз вже існуючих як класичних, так і інтелектуальних систем управління навчальним процесом з метою виділення набору найпопулярнішого та найнеобхіднішого функціоналу. Як згадувалося вище, даний набір властивостей системи має задовольняти таким вимогам, як проактивність, автономність та реактивність. На основі подібних властивостей для реалізації наведеної системи був обраний мультиагентний підхід. Оскільки при роботі системи відбувається різноманітний аналіз великої кількості інформації, то в рамках даної роботи були проаналізовані основні види та алгоритми аналізу даних, деякі з яких (кореляційний і регресійний) використовуються в межах

систем. З огляду на ці причини, а також через ефективність та зручність реалізації функцій для аналізу даних, для розробки було вирішено обрати мову програмування Python.

Можливі сфери застосування. Мультиагентна інтелектуальна система управління навчальним процесом може застосовуватись у вищих навчальних закладах на рівні факультетів, а також може бути масштобованою для управління навчальним процесом в усьому вищому навчальному закладі.

РОЗДІЛ 1
СИСТЕМИ УПРАВЛІННЯ НАВЧАЛЬНИМ ПРОЦЕСОМ

У зв'язку з впровадженням інформаційних технологій та комп'ютерних систем у більшість сфер повсякденного життя, зокрема і з впровадженням даних систем у навчальний процес та в управління навчальним процесом, виникли такі поняття як система управління навчальним процесом, система управління навчальними матеріалами, система для зберігання навчальних об'єктів тощо. Широку популярність здобули системи управління навчальним процесом, які можуть бути визначені за допомогою наступних означень та створюються першочергово з метою управління навчальним процесом, а не навчальними матеріалами.

Означення 1.1. Системи управління навчальним процесом — це сімейство систем, які допомагають реалізувати управління всіма навчальними активностями, так само як, і надають можливість демонстрації матеріалів, пов'язаних з навчальними курсами. Зокрема системи управління навчальним процесом класифікують користувачів, надаючи їм певні права на доступ до навчальних модулів та пов'язаних з користувачем навчальних груп. [1]

Означення 1.2. Система управління навчальним процесом — це веб система, яка створюється та використовується для планування, реалізації та оцінки навчального процесу. [2]

У відповідності до [3, 4] на теперешній час 97 % вищих навчальних закладів використовують системи управління навчальним процесом більшість з даних вищих навчальних закладів використовують системи управління навчальним процесом впродовж близько 8 років і 15 % з них планують в наближчі 3 роки або ж перейти до нової версії системи, що використовується, або ж обрати нову систему управління навчальним процесом.

1.1. Основні типи користувачів

Система управління навчальним процесом на відміну від інших подібних систем у вищих навчальних закладах має за основну мету управління навчальним процесом, а тому при її розробці та реалізації зазвичай виокремлюють такі типи користувачів як студент, викладач, працівник деканату. Кожен з типів користувачів має свій набір функціоналу та прав доступу до модулів системи та навчальних матеріалів.

Студент є користувачем системи, який використовує систему з метою отримання знань, тобто доступу до навчальних матеріалів, завантаження виконаних завдань та отримання відгуку про них. Тому при проектуванні та реалізації такої системи мають бути спроектовані та реалізовані наступні функції для користувача типу студент: перегляд навчальних матеріалів, синхронні та/або асинхронні способи комунікації, завантаження виконаних завдань, проходження тестів, отримання зворотнього відгуку про виконані завдання тощо.

Викладач є користувачем системи, який створює курси, додає навчальні матеріали, керує навчальним процесом в межах курсів, а також надає зворотній відгук про виконані завдання студентам та комунікує з ними. Тому при проектуванні та реалізації такої системи мають бути спроектовані та реалізовані наступні функції для користувача типу викладач: створення/модифікація/видалення курсів, створення/модифікація/видалення звітів, створення/модифікація/видалення навчальних матеріалів, створення/модифікація/видалення тестів, синхронні та/або асинхронні способи комунікації тощо.

Працівник деканату є користувачем системи, який управляє навчальним процесом. Тому при проектуванні та реалізації такої системи мають бути спроектовані та реалізовані наступні функції для користувача типу працівник деканату: розповсюдження та перегляд тестів, розповсюдження та перегляд звітів, розповсюдження та перегляд навчальних матеріалів, розповсюдження та перегляд курсів, синхронні та/або асинхронні способи комунікації тощо.

1.2. Основні потреби користувачів та функції систем управління навчальним процесом

Відповідно до [2], системи управління навчальним процесом створюються з метою:

— надання доступу користувачам системи до навчальних матеріалів незалежно від місця перебування користувача та часу;

— створення централізованого сховища навчальних матеріалів та даних, пов'язаних з навчальним процесом;

— моніторингу та надання звітів для підвищення рівня навчання та успішності студентів;

— підвищення активності студентів в таких діяльностях як виконання та надання на перевірку завдань, участь у наукових конференціях тощо;

— покращення комунікації між студентами та викладачами, зокрема в межах змішаного або дистанційного навчального процесу;

— отримання аналітики про навчальний процес з метою його покращення;

— тощо.

Як зазначалося вище системи управління навчальним процесом мають перелік обов'язкових характеристик та додаткові, які покликані на покращення системи та підвищення її конкурентноздатності на ринку подібних систем, який, згідно з [3], хоч і перевантажений, але відкритий до нових систем управління навчальним процесом нового покоління, які будуть враховувати сучасні потреби користувачів у навчанні та навчальному процесі. На малюнку 1.1 наведено приклад переліку функцій системи управління навчальним процесом та способу їх об'єднання у категорії за типом призначення функціоналу.

1.2.1. Функціональні вимоги. Основною метою систем управління навчальним процесом є безпосередньо управління навчальним процесом, а не навчальними матеріалами. Тому до основних функцій систем управління навчальним процесом, зокрема і в [5–7], відносять:

Рисунок 1.1. Функціонал систем управління навчальним процесом.

— створення та управління курсами;
— створення та управління звітами;
— організація комунікації між користувачами;
— доступ користувачів до матеріалів у відповідності з їх правами доступу;
— управління комунікацією між користувачами;
— тощо.

Системи управління навчальним процесом можуть, але не зобов'язані включати елементи управління навчальними матеріалами. Дана частина навчального процесу може бути винесена в межі іншої системи.

1.2.2. Додаткові можливості. Згідно з [8–11], оскільки основні функціональні вимоги до систем управління навчальним процесом не повністю покривають усі активності і процеси у вищих навчальних закладах, то для підвищення конкурентноздатності системи управління навчальним процесом на ринку систем для вищих навчальних закладів деякі розробники додають до систем управління навчальним процесом додатковий функціонал, наприклад:

— створення та управління навчальними матеріалами;
— створення та управління тестами;
— способи заохочення студентів до здачі завдань;
— додаткові дії, що базуються на отриманій аналітиці, з метою покращення

навчального процесу;

— тощо.

Додатковий функіонал може бути наданий користувачам або ж в межах однієї системи, або ж за допомогою створення корпорації систем, які для обміну даними між окремими системами використовують навчальні об'єкти (найменші елементи навчальних матеріалів) та інші об'єкти, які відповідають зазделегідь визначеним стандартам.

1.2.3. Нефункціональні вимоги. Оскільки, як зазначено зокрема в [2, 4, 6, 7, 12, 13], системи управління навчальним процесом здебільшого реалізуються як веб системи, для яких характерне високе навантаження через велику кількість користувачів та великі об'єми даних, то до систем управління навчальним процесом висувають наступні вимоги:

— зручність інтерфейсу користувача;

— реалізація інтерфейсу користувача в пастельних тонах;

— час відгуку до 1 секунди;

— підтримка не менше ніж 10000 користувачів одночасно;

— тощо.

Разом з якістю основного функціоналу систем управління навчальним процесом, конкурентноздатність даних систем формує якість, різноманіття та спеціалізація додаткових функцій та якість забезпечення нефункціональних вимог.

1.2.4. Подальший розвиток функціоналу. З плином часу потреби користувачів і сам навчальний процес змінюються і з'являється необхідність у розвитку систем управління навчальним процесом. Згідно з [2, 3, 6, 14, 15] в подальшому можливі наступні напрямки розвитку систем управління навчальним процесом: додавання нового функціоналу або ж повне переосмислення навчального процесу загалом і також підходу до створення систем управління навчальним процесом.

Додавання нового функціоналу найімовірніше буде базуватись на використанні аналітичних підходів та моделей, що не додасть суттєвих змін до способів побудови та управління курсами, навчальним матеріалами, способами комунікації, звітами тощо.

На відміну від попереднього способу повне переосмислення начального процесу загалом і також підходу до створення систем управління навчальним процесом може враховувати наступні особливості та потреби сучасного процесу управління навчальним процесом такі як:

— індивідуальний підхід до навчання кожного студента(побудова індивідуального плану навчання для досягнення певного рівня);
— доступ до систем управління навчальним процесом з мобільних пристроїв;
— тренінги для користувачів (оскільки згідно з [3] 51% користувачів систем управління навчальним процесом стверджують, що вони могли б ефективніше використовувати дані системи якщо б повністю розуміли їх функціонал та способи його використання);
— інтуїтивно зрозумілий інтерфейс та проста реалізація складного функціоналу (відповідно до [2, 3], більшість як викладачів так і студентів не використовують складний функціонал систем управління навчальним процесом у зв'язку з складністю його використання та відсутністю тренінгів по використанню даного функціоналу);
— тощо.

Виходячи з даних підходів, можна стверджувати, що в подальшому можливий як розвиток даного покоління систем управління навчальним процесом так і виникнення нового покоління систем управління навчальним процесом.

1.3. Структура систем управління навчальним процесом

Системи управління навчальним процесом у відповідності до свого різноманітного та об'ємного функціоналу, а також у відповідності до нефункціоналних вимог мають складну структуру. Згідно з [16, 17] система

управління навчальним процесом складається щонайменше з п'яти наступних модулів:

— **Модуль курсів** — для створення, управління та розповсюдження курсів, подекуди навчальних матеріалів.

— **Модуль користувачів** — для створення, управління та розповсюдження персональних даних користувача, інформації про дії користувача та доступні курси, а також про програми даних курсів. Також даний модуль відповідає за авторизацію, аутентифікацію користувача та контроль за правами доступу різних типів користувачів.

— **Модуль екзаменів** — для створення та управління електронними іспитами та відповідями на них з математичною точністю, автоматичного контролю відповідей на тести (зазвичай текстові запитання оцінюється викладачем), автоматичної генерації документів про досягнення певного рівня (такі як сертифікати тощо).

— **Модуль звітів** — для створення різних типів аналітик на основі даних, які зберігаються у системі управління навчальним процесом, автоматичної генерації звітів на основі даних аналітики. Така аналітика може включати успішність учнів на іспитах, участь студентів у курсах, звички користувачів при використанні системи тощо.

— **Модуль зв'язків** — для зберігання в системі управління навчальним процесом даних, які описують з'єднання між частинами системи, її користувачами, їх ролями та матеріалами користувачів. Наприклад, створення зв'язку між студентами та викладачами та курсом, створення зв'язку між файлом з виконаними завданнями та студентом і курсом, створення зв'язку між студентом та його оцінками, створення зв'язку між завданнями та курсом тощо.

Основні частини систем управління навчальним процесом зображені на малюнку 1.2.

Рисунок 1.2. Основні структурні модулі систем управління навчальним процесом.

1.4. Класифікація систем управління навчальним процесом

Оскільки ринок систем управління навчальним процесом перевантажений, то для зручності їх прийнято класифікувати у відповідності до способу використання, способу реалізації, способу оплати за використання системи тощо.

1.4.1. Класифікація за способом доступу та оплатою за використання системи. У відповідності до даної класифікації, яка детально розглянута в [2, 10], широко відомі такі типи систем управління навчальним процесом як системи з відкритим кодом, платні системи та системи засновані на хмарних технологіях.

Системи управління навчальним процесом з відкритим кодом зазвичай безкоштовні, дозволяють модифікації в коді системи зі сторони користувачів, підтримуються великою спільнотою користувачів та/або розробників, також мають високий рівень безпеки як всієї системи так і її даних тощо. Але натомість системи управління навчальним процесом з відкритим кодом не мають технічної служби підтримки, а отже слід покладатися на спільноту користувачів та/або розробників, які можуть не надати необхідну відповідь на питання та підтримку системи. Також в таких системах відсутні відповідальні за неточності та помилки,

а отже виправлення недоліків покладається на користувачів.

Натомість платні системи управління навчальним процесом є платними, централізованими (всі дані зберігаються в одному місці), широко використовуваними, мають високий рівень безпеки як всієї системи так, і її даних, мають швидку і надійну службу технічної підтримки тощо. Але в той же час такі системи можуть бути дорогими, не гнучкими, не пристосуваними до потреб користувача, мають закритий код, а також завжди існує ризик втрати сервісів і відповідно даних через неоплату чи недоримання політик компанії розробника системи.

Системи управління навчальним процесом, засновані на хмарних технологіях, є недорогим та популярним рішенням для реалізації систем управління навчальним процесом, а також системи даного типу надають широкий спектр функціоналу, який кожен користувач системи управління навчальним процесом може обрати для себе, що є важливим оскільки кожен користувач має свої потреби в додатковому функціоналі. Ще однією з переваг даного підходу є те, що більшість студентів вже ознайомлені з великою кількістю хмарних рішень, таких як YouTube, Skype, Facebook та Google Drive. Але одночасно системи управління навчальним процесом даного типу можуть не мати технічної служби підтримки, а отже слід покладатися на спільноту користувачів та/або розробників, які можуть не надати необхідну відповідь на питання та підтримку. В таких системах можуть бути відсутніми відповідальні за неточності та помилки, а отже виправлення недоліків покладається на користувачів.

При виборі системи управління навчальним процесом згідно даної класифікації необхідно враховувати рівень знань користувачі (чи необхідна технічна підтримка), потребу в модифікаціях система, зокрема і коду системи, а також фінансові можливості споживача.

1.4.2. Класифікація за способом використання. Системи управління навчальним процесом використовують не тільки у вищих навчальних закладах, а й в багатьох організаціях та інших закладах навчання. Тому, як наведено в [16],

системи управління навчальним процесом поділяють на корпоративні, академічні та інтегровані системи управління навчальним процесом з системами управління навчальними матеріалами.

Корпоративні системи управління навчальним процесом дозволяють організаціям поширювати відносно короткі (від кількох годин до декількох днів) курси та надають можливість підключення користувачів та курсів для проходження даних курсів.

Академічні системи управління навчальним процесом — це зазвичай веб системи, за допомогою яких студенти та викладачі взаємодіють. В межах систем даного типу студенти можуть спілкуватись з іншими студентами та/або викладачами асинхронними чи/та синхронними засобами, завантажувати виконані завдання, проходити тести тощо. Також за допомогою систем даного типу викладачі можуть створювати, поширювати та управляти курсами.

Інтегровані системи управління навчальним процесом з системами управління навчальними матеріалами подібні до корпоративних систем управління навчальним процесом, але також надають додатковий функціонал відмінний від функціоналу корпоративних систем управління навчальним процесом для розробки та створення, а також управління навчальними матеріалами.

Вибір системи на основі даної класифікації має базуватись не на типі установи, що буде використовувати систему, а на її потребах у функціональності системи і типах користувачів, які підтримує система.

1.5. Огляд подібних систем

Оскільки системи управління навчальним процесом популярні у вищих навчальних закладах і, як зазначено у [3], ринок систем управління навчальним процесом перевантажений, то користувачі обирають систему управління навчальним процесом в залежності від її характеристик та рейтингу серед тих, хто вже використовує дані системи. Так згідно з [18–20], найпопулярнішою системою управління навчальним процесом є Moodle, а Blackboard Learn згідно з [21, 22]

є найстарішою та найпопулярнішою платною системою управління навчальним процесом. Досить часто такі старі, великі та подекуди із застарілими на даний момент технологіями розробки системи випереджають за популярністю новіші системи. Moodle та Blackboard Learn є яскравими прикладами даного феномену, який полягає в тому, що такі великі хоча й не сучасні в технічному плані системи вже надають той функціонал, який нові системи можуть лише запропонувати, і ще й додатково додають щось нове до свого поточного функціоналу. У відповідності до [23, 24] Docebo та Adobe Captivate Prime є найпопулярнішими платними системами управління навчальним процесом на основі хмарних технологій. Docebo та Adobe Captivate Prime є яскравими прикладами того, що одна й та сама система управління навчальним процесом може належати до різних типів у різних класифікацій.

1.5.1. Moodle. Moodle - це багатоплатформна веб система управління навчальним процесом, яка була вперше розроблена в серпні 2002 року і доступна на офіційному сайті [25]. Також на офіційному сайті розміщена детальна документація по даній системі і контактні дані служби підтримки. Moodle розповсюджується під публічною ліцензією GNU General Public License. Користувачі системи можуть обирати зберігати дані на власному сервері чи за допомогою використання хмарних технологій. Також доступний мобільний додаток для роботи з системою, який можна завантажити на App Store [26] для IOS, на Google Play Market [27] для Android та на Microsoft Store [28] для Windows Phone.

Дана система підтримує такі типи користувачів, як студент та викладач, та надає їм наступний функціонал:

— для студентів:

- виконання та завантаження завдань;
- перегляд власних оцінок та відвідуваності;
- доступ до навчальних матеріалів;
- синхронні та асинхронні способи комунікації;

- проходження тестів;
- засоби для групової роботи;
- нагадування про важливі події;
- тощо.

— для викладачів:

- створення та управління курсами;
- створення та управління навчальними матеріалами;
- створення та управління тестами;
- автоматизація процесу перевірки тестів, звітів тощо;
- використання сторонніх засобів для дистанційного навчання шляхом підключення плагінів;
- тощо.

Перевагами даної системи, які досить тривалий час підтримують конкурентноздатність та популярність системи, є:

— широкий та масштобований спектр функціоналу;

— високий рівень безпеки даних;

— можливість підключення інших додатків для покращення навчального процесу шляхом використання плагінів;

— швидка та надійна служба підтримки;

— підтримка десятків мов;

— тощо.

Moodle, як і всі системи, має свої недоліки, зокрема:

— відсутність способів комунікації між студентом та викладачем напряму;

— використання деяких плагінів та функціоналу потребує оплати послуг;

— малий перелік типів звітів в базовій безкоштовній версії;

— орієнтованість на малі та середні навчальні заклади;

— тощо.

Moodle підходить для малих та середніх навчальних закладів, які або ж мають можливість оплачувати додатковий функціонал або ж їх потреби покриває базовий безкоштовний пакет системи з можливим використанням інших систем.

1.5.2. Adobe Captivate Prime. Adobe Captivate Prime — це система управління навчальним процесом, яка базується на використанні хмарних технологій. Перша версія системи мала назву Flashcam і розроблялася як система запису дій на екрані компанією Nexus Concepts. Перший випуск Flashcam був випущений в травні 2001 року. Натомість останній стабільний випуск Adobe Captivate Prime (CS6) відбувся в 2012 компанією Adobe Systems Inc. Дана система є програмним забезпеченням, розробленим за допомогою мови програмування Object Pascal. Adobe Captivate Prime може використовуватись на таких операційних системах як Windows та Mac OS X з п'ятої версії даної системи і доступна разом з документацією по ній на офіційному сайті системи [29].

Від початку Adobe Captivate Prime була розроблялася як програма запису з екрану. Тому дана система надає окрім класичних функцій систем управління навчального процесу можливість запису дій користувача під час заняття та їх відтворення. Також дана система надає наступний функціонал:

— створення та редагування презентацій;
— створення та модифікація тестів;
— демонстрація дій викладача;
— демонстрація функціонування роботи інших програмних застосунків таких як ігри, симуляції, відео тощо;
— повноцінний запис дій з екрану;
— редагування аудіо;
— використання шаблонів проектів;
— використанні інтерактивних елементів під час презентацій та запису відео;
— створення та редагування розкладу;
— перегляд файлів PDF, PPT, DOCX, SCORM форматів;
— тощо.

Система здобула свою популярність через зручність використання, яка в свою чергу була сформована з функціоналу та наступних переваг:

— підтримка відеоформату SWF;
— можливість конвертування .swf в .avi;

— сумісність з іншими програмами Adobe;

— сумісність з Microsoft Power Point;

— сумісність з іншими програмами під час запису відео;

— коментування SWF;

— анімовані ефекти тексту;

— інтерфейс у стилі Microsoft Power Point;

— підтримка PDF, PPT, DOCX, SCORM и AICC-сумісних пакетів;

— інтуїтивно зрозумілий інтерфейс системи;

— тощо.

Оскільки Adobe Captivate Prime LMS не має чіткої структури для системи управління навчальним процесом у вищому навчальному закладі, дана система має наступні недоліки при використанні у вищих начальних закладах:

— відсутність ієрархічної структури для таких типів користувачів як студент, викладач, працівник деканату тощо;

— можливість додавання користувача до однієї групи в ролі студента і викладача;

— відсутність моніторингу відвідуваності занять користувачами;

— не можливість додавання студентів без надсилання повідомлень на електронну пошту;

— тощо.

Adobe Captivate Prime надає великий перелік функцій для навчання та формування звітів, демонстрації програм та продуктів, підтримує різні типи користувачів та формати даних. Тому дана система управління навчальним процесом досить зручна у використанні в якості корпоративної системи управління навчальним процесом в межах підприємства.

1.5.3. Docebo. Docebo — це платна система управління навчальним процесом на основі хмарних технологій, яка розповсюджується під безкоштовною GNU General Public License v2.0 і розроблена за допомогою використання мови програмування PHP. Система доступна на офіційному сайті [30] і згідно з [31]

використовується більше ніж 2000 установ та організацій. Мобільна версія системи доступна на офіційному сайті [32]. Дана система орієнтована на середні та може бути масштабована для використання у великих установах та організаціях здебільшого в якості корпоративної системи управління навчальним процесом.

Для підтримки своєї популярності розробники Docebo надають та покращують наступний функціонал для користувачів системи:

— централізоване зберігання курсів;
— управління курсами;
— розповсюдження курсів;
— вимірювання результатів за допомогою інформаційних панелей та спеціальної аналітики;
— підримка більше ніж 35 API (зокрема HRI, CRM (Salesforce), Content authoring eCommerce (Stripe, Shopify), Collaboration and productivity tools (Slack, G Suite), Single sign-on (SSO), Content partners, Web conferencing (GoToMeeting, GoToWebinar, Zoom) тощо);
— моніторинг дій користувачів системи;
— створення, управління та коментування блогів користувачів системи;
— створення та управління мітками на курсах з метою пришвидшення їх пошуку та управління курсами;
— можливість користувачів створювати електронні винагороди за проходження їх курсів;
— створення та управління навчальним планом;
— створення правил для автоматичних нагадувань користувачам систем або ж слухачам певних курсів;
— управління нагадуваннями;
— покупка та продаж курсів;
— тощо.

Docebo поєднує можливості, переваги та недоліки як платних систем управління навчальним процесом так і систем управління навчальним процесом на основі хмарних технологій. Так дана система поєднує наступні переваги:

— ефективна, швидка аналітика, яка покриває більшість потреб користувачів;
— підримка більше ніж 35 API (зокрема HRI, CRM (Salesforce), Content authoring eCommerce (Stripe, Shopify), Collaboration and productivity tools (Slack, G Suite), Single sign-on (SSO), Content partners, Web conferencing (GoToMeeting, GoToWebinar, Zoom));
— підтримка більше ніж 40 мов;
— сумісність з стандартами електронного навчання SCORM 1.2;
— версія системи для мобільних телефонів, яка доступна на офіційному сайті;
— можливість продажу та покупки курсів;
— оплата лише присутньої на занятті кількості користувачів;
— тощо.

Також користувачі даної системи управління навчальним процесом помітили наступні недоліки системи:

— навчальні відео лімітовані за розміром;
— не доступність деякого функціоналу основної версії системи з мобільної версії;
— неякісна технічна підтримка;
— тощо.

Розробники Docebo за плату надають користувачам вище зазначений функціонал та намагаються виправити раніше наведені недоліки за допомогою використання хмарних технологій.

1.5.4. Blackboard Learn. Blackboard Learn - це платна кросплатформна система управління навчальним процесом, яка доступна разом з детальною документацією на офіційному сайті [33]. Вперше дана система була випущена компанією Blackboard Inc. 21 січня 1997 року. Останнє стабільне оновлення системи датується жовтнем 2014 року. Також на офіційному сайті системи доступна мобільна версія Blackboard Learn [34].

Оскільки Blackboard Learn часто використовується вищими навчальними

закладами для дистанційного навчання, то дана система надає користувачам наступний функціонал:

— створення та управління курсами;
— синхронні та асинхронні способи комунікації (оголошення, чати, електронні листи, обмін повідомленнями тощо);
— виставлення та перегляд відміток про відвідуваність користувачем курсів;
— виставлення та перегляд оцінок користувачам в межах курсів;
— створення та управління тестами;
— перегляд згенерованих звітів;
— обмін файлами;
— можливість створення електронних винагород за проходження курсів;
— можливість подання навчального процесу у вигляді гри;
— тощо.

Впродовж свого існування дана система управління навчальним процесом надає користувачам наступні переваги:

— настроювана відкрита архітектура та зручний дизайн, що дозволяє інтеграцію з інформаційними системами студентів та протоколами аутентифікації;
— великий вибір асинхронних та синхронних способів комунікації користувачів системи;
— можливість інтеграції з великою кількістю інших систем управління навчальним процесом;
— сумісність з стандартами електронного навчання SCORM;
— швидка та якісна технічна підтримка;
— тощо.

Користувачі даної системи виділяють наступні недоліки при її використанні:

— якість мобільної версії системи;
— відсутність безкоштовної тестової версії системи;
— використання при розробці застарілих технологій;
— тощо;

Blackboard Learn може досить ефективно використовуватись в малих та середніх організаціях та установах для впровадження та управління навчальним процесом, а також розповсюдження та управління навчальними матеріалами.

1.5.5. Порівняння систем управління навчальним процесом. З вищих наведених оглядів можна зробити висновок, що всі розглянуті системи управління навчальним процесом надають своїм користувачам мінімально-необхідний для систем управління навчальним процесом функціонал такий як: створення та управління курсами, створення та управління обліковими записами, надання прав користувачам на доступ до курсів та до матеріалів тощо. Всі наведені системи є веб додатками, що робить їх використання зручним для користувачів, оскільки вони мають доступ до курсів та навчальних матеріалів незважаючи на час та місце їх перебування. Деякі користувачі хочуть мати доступ до даних систем за допомогою мобільних пристроїв. Так, Moodle та Blackboard Learn надають таку можливість. В той час як мобільний додаток системи Moodle є якісним і широковживаним, мобільна версія Blackboard Learn не завжди задовільняє потреби користувачів і не використовується так часто як аналогічна функція системи Moodle. Також досить зручним для користувачів, особливо викладачів, є сумісність типів навчальних матеріалів, які викладачі розміщують в межах курсів, з стандартами електронного навчання Scorm. Так, дані стандарти підтримують Moodle, Decobo та Blackboard Learn.

Розробники деяких систем вирішили підтримувати свою популярність шляхом покращення вже існуючого функціоналу, як наприклад Blackboard Learn використовує для цих цілей покращення основного функціоналу та надання великої кількості якісних асинхронних та синхронних способів комунікації. Інші ж системи надають користувачам принципово новий функціонал, наприклад Adobe Captivate Prime надає користувачам можливість запису дій на екрані користувача. Також для залучення нових користувачів розробники даних систем додають підтримку великої кількості мов.

Окрім функціоналу, що надають системи, кожна система має свої переваги та

недоліки, які базуються на типі та якості архітектури системи, технологіях для створення системи, типі та віці системи управління навчальним процесом тощо. Тому при виборі системи слід враховувати її функціонал, переваги, недоліки, цільову аудиторію, та не слід розглядати системи управління навчальним процесом певного типу непридатними для використання в іншій ситуації. Тому, що система може бути, як Moodle, масштобавана у відповідності до задачі, або ж і система і навчальний процес можуть бути змінені, так наприклад Adobe Captivate Prime може використовуватись у вищих навчальних закладах з малою кількістю користувачів при відсутності необхідності побудови складних аналітичних залежностей про навчальний процес і можливістю додаткової документації даного навчального процесу або ж вручну або ж шляхом використання іншої сумісної з Adobe Captivate Prime системи.

РОЗДІЛ 2
ІНТЕЛЕКТУАЛЬНІ АГЕНТИ ТА МУЛЬТИАГЕНТНІ СИСТЕМИ

Складність задач зростає разом з розвитком технологій. На даний момент існує чимало задач, які не можуть бути розв'язані без використання складних підходів та технологій. Прикладом такого розподіленого підходу до вирішення задач є мультиагентний підхід, який ще й надає можливість створювати про-активні, реактивні та автономні системи.

2.1. Агенти та мультиагентні системи

Агенти та мультиагентні системи є складовими мультиагентного підходу до розподілених обчислень та побудови розподілених систем. Але згідно з [35–38] не існує чіткого єдиного визначення поняття агент, яке б покривало всі характеристики агентів та було прийняте науковою спільнотою. Для зручності та уникнення розбіжностей у поглядах співрозмовників були уведені певні характеристики поняття агент. Згідно з [39] виділяють інтелектуальних та звичайних агентів. Звичайні агенти виконують прості дії за допомогою використання простих способів кооперації та комунікації. На відміну від них інтелектуальні агенти використовують більш витончені алгоритми в своїй роботі, кооперації та комунікації, а також агенти даного типу описуються за допомогою різних визначень наведених в наступних розділах. Вулдрідж в [35] запропонував *слабке* та *сильне* визначення поняття інтелектуального агента.

Означення 2.1. *Слабке* визначення агента — це визначення, яке розглядає агента як окрему незалежну сутність для якої властиві наступні ознаки:

— *автономність* — інтелектуальний агент здатен до автономних дій направлених на досягнення поставлених цілей;

— *про-активність* — інтелектуальний агент здатен виконувати направлену

на цілі поведінку за рахунок взяття ініціативи для задоволення розроблених в рамках проекту цілей;

— *реактивність* — інтелектуальний агент здатен сприймати його навколишнє середовище та відповідальний за вчасне врахування змін навколишнього середовища задля досягнення зпроектованих цілей;

— *соціальні здібності* — інтелектуальний агент здатен взаємодіяти з іншими агентами задля досягнення зпроектованих цілей.

Означення 2.2. *Сильне* визначення агента — це визначення, яке розширює *слабке* визначення додаючи опис поведінки агента за допомогою використання таких ментальних характеристик як переконання, бажання та цілі (наміри).

Згідно з [35] згадані в *сильному* визначенні поняття агент ментальні якості можуть бути визначені за допомогою означень 2.3, 2.4 та 2.5.

Означення 2.3. Переконання — це атрибут агента, який описує його уявлення про навколишній світ.

Означення 2.4. Бажання — це атрибут агента, який описує множину завдань, які агент хоче виконати.

Означення 2.5. Намір — це атрибут агента, який описує множину цілей, які агент намагається досягти в даний момент.

Згідно з [35–38] агент може бути описаний за допомогою його атрибутів, початкового стану, його переконань про початковий стан навколишнього середовища тощо. Основною задачею агента є спостереження за навколишнім середовищем та реагування на зміни в навколишньому середовищі з метою досягнення поставлених цілей. Для досягнення поставлених цілей агенти використовують функцію прийняття рішень. Простий агент може бути визначений як функція f, яка проектує множину перцепцій P на множину можливих для виконання агентом дій A

$$f : P \to A.$$

Крім того, що існують інтелектуальні та звичайні агенти, у відповідності до [36, 39, 40] агенти можуть бути розділені на різні типи згідно таких класифікацій,

як класифікація агентів в залежності від архітектури, класифікація агентів в залежності від внутрішньої структури та класифікація агентів в залежності від задач, які вони виконують.

Однією з найважливіших властивостей агентів є властивість під назвою соціальні здібності, яка забезпечує кооперацію та комунікацію між агентами та навколишнім середовищем, яке також може бути агентом, та широко використовується при розробці мультиагентних систем, які можуть бути визначені за допомогою означення 2.6.

Означення 2.6. У відповідності до [38], мультиагентна система — це система, яка складається з не нульової кількості агентів.

Існує велика кількість типів агентів та способів їх реалізації, а також архітектур мультиагентних систем з використанням даних типів агентів. Як зазначено в [41], розрізняють такі архітектури мультиагентних систем, як архітектура мультиагентних систем на основі логік, реактивна архітектура, шарова архітектура, архітектура на основі переконань, намірів та бажань тощо.

2.2. Типи агентів

Оскільки поняття агент є досить абстрактним і згідно з [35] не існує чіткого визначення даного поняття, то існує чимало класифікацій агентів, деякі з яких залежать від тлумачення суті поняття агент. Так у [40] Нікос Дракос класифікує агентів за задачами, які вони виконують, а у [36, 39, 41] Вулдрідж та інші класифікують агентів за їх внутрішньою структурою. Також в [39] визначають типи агентів в залежності від їх архітектури. У відповідності до [41] агент одного типу може бути визначений в термінах агенту іншого типу згідно певної класифікації. Також агент може відразу належати до кількох типів у відповідності до різних класифікацій.

2.2.1. Класифікація агентів в залежності від архітектури. В даному випадку поняття архітектури згідно з [39] інтерпретується як машина, на якій

працює агент. Як було зазначено раніше агент характеризується сприйняттям інформації з власних джерел та навколишнього середовища, аналізом даної інформації, а також за потреби відповідною зміною внутрішнього стану та взаємодією з навколишнім середовищем на основі даного аналізу. Так у відповідності до [39], в залежності від архітектури виділяють наступні типи агентів:

— **Агент-Людина** — в якості агента виступає людина; інформація з довкілля сприймається за допомогою вух, очей та інших органів чуття; людина опрацьовує інформацію та змінює свій внутрішній стан, наприклад свої переконання, або ж взаємодіє з середовищем, наприклад каже або робить щось.

— **Агент-Робот** — в якості агента виступає робот; інформація з навколишнього середовища сприймається за допомогою камер, радарів та інших сенсорів робота; для опрацювання інформації, зміни власного стану та взаємодії з навколишнім середовищем за потреби в даному випадку використовуються рефлексивні мотори, генератори звуків та інше механічне та програмне оснащення робота.

— **Програмний агент** — в якості агента виступає програма; інформація з навколишнього світу сприймається за допомогою відслідковування натиснень клавіш клавіатури та миші, голосових повідомлень користувача програми, взаємодії з іншими програмами та глобальною мережею Інтернет; для зміни власного стану в даному випадку використовуються програмні команди та зміни програмних регістрів; для взаємодії з навколишнім середовищем агенти даного типу використовують виведення повідомлень та зображень на екран користувача, генерацію та відтворення голосових повідомлень, запити до глобальної мережі Інтернет тощо.

Надалі в даній роботі розглядаються лише програмні агенти.

2.2.2. Класифікація агентів в залежності від задач, які вони виконують. В залежності від сприйняття поняття агент, можна покладати на

нього різні типи задач для виконання. Тому у [40] виділяють наступні типи агентів:

— **Виконавчі агенти** — агенти, які відповідальні за планування рівня навичок агентів за участі всієї групи або ж її підгруп.

— **Агенти, що співпрацюють** — агенти, які відповідальні за незалежні дії кожного агента та прямий внесок кожного агента у досягненні загальної мети, коли для виконання дії залучено більше одного агента.

— **Агенти-учасники** — агенти, які відповідальні за прямий внесок агента при виконанні дії коли дану дію виконує один агент.

— **Агенти зв'язку** — агенти, які відповідальні за управління комунікацією між агентами.

— **Сервісні агенти** — агенти, які відповідальні за надання послуг низького рівня іншим агентам або компонентам системи.

При розробці мультиагентної системи один агент може виконувати декілька задач з наведених вище, наприклад співпрацювати з іншими агентами, а також підтримувати комунікацію з ними. Отже, в залежності від задачі, яку даний агент виконує в певний момент часу, він може відноситись до різних типів згідно даної класифікації в різні моменти часу.

2.2.3. Класифікація агентів в залежності від внутрішньої структури.

В залежності від того, що вкладається в поняття агент, які функції покладаються на агента, а також в залежності від способів, технологій та підходів для реалізації агента та його зовнішнього середовища будуть відрізнятись його внутрішня будова та способи комунікації та кооперації з іншими агентами та навколишнім середовищем. Згідно з [36, 39, 41] розрізняють наведені у даному підрозділі типи агентів у відповідності до внутрішньої структури спроектованого та/або реалізованого агента.

2.2.3.1. Реактивні агенти. У відповідності до [36, 39, 41], реактивні агенти— це агенти, які не зберігають та не обробляють інформацію про минулі події, а лише сприймають інформацію з навколишнього середовища в

певні моменти часу, обробляють дану інформацію за допомогою примітивних алгоритмів та за потреби взаємодіють з середовищем. Агенти даного типу можуть отримувати інформацію з навколишнього середовища шляхом отримання повідомлень від інших агентів та частин системи або ж роблячи певні виміри або ж надсилаючи запити в навколишнє середовище. Якщо припустити, що агент обробляє отриману інформацію за допомогою функції, яка переводить отримані виміри у множину натуральних чисел, то роботу агента даного типу, а саме його функцію дії, можна математично сформулювати наступним чином. Для зручності позначимо раніше описану функцію так $f(env) \rightarrow \mathbb{N}$, де env - це отримана з навколишнього середовища інформація.

$$agent\_action = \begin{cases} action_1, f(env) == 0, \\ ..... \\ action_n, f(env) == n \end{cases}$$

Також згідно з [39], основним недоліком агентів даного типу є лімітованість варіантів дій агента та простота функції прийняття рішення, водночас головною перевагою агентів даного типу є простота в розробці та реалізації при вирішенні простих задач з кінцевою кількістю варіантів дій агента.

2.2.3.2. Агенти на основі моделі. Згідно з [36], агенти на основі моделі мають однакову структуру з реактивними агентами, але можуть містити так звані одномісні стани для зберігання поточного стану агента, який може змінюватись в залежності від обробленої агентом інформації, але даний внутрішній стан не завжди може бути використаний при аналізі отриманої агентом інформації. Дана можливість робить більш гнучкою поведінку та розробку даних агентів, але в цілому агенти даного типу мають ті ж переваги та недоліки що й реактивні агенти, але в іншій мірі.

2.2.3.3. Агенти, які використовують перцепції. Згідно з [36, 41], агенти, які використовують перцепції — це агенти, які сприймають зміни в навколишньому середовищі за допомогою функції, яка проектує стани середовища

у множину перцепцій, на основі яких агент приймає рішення про свої подальші дії. Якщо визначити множину станів агента як S, множину перцепцій як P, а множину допустимих дій агента як A, то математично роботу агента можна описати наступним чином.

Агент сприймає зміну в навколишньому середовищі та опрацьовує її за допомогою функції

$$see : S \to P,$$

яка перетворює елементи множини станів навколишнього середовища S у елементи множини перцепцій агента P.

Потім на основі отриманої перцепції за допомогою функції

$$action : P^* \to A,$$

яка проектує елементи множини перцепцій P на елементи множини дій агента A, агент обирає дію для виконання і виконує її. В подальшій роботі дії агента циклічно повторюються в наступному порядку:

— сприйняття інформації з навколишнього середовища;

— перетворення сприйнятої інформації в перцепцію;

— вибір агентом дії для виконання на основі перцепції.

Основною перевагою агентів даного типу є спрощення проектування та реалізації функції прийняття агентом рішень про подальші дії шляхом винесення певного функціоналу в функції перетворення інформації про навколишнє середовище в перцепцію та зменшенням за рахунок даної функції ймовірних сценаріїв поведінки агента. Так, якщо існує два стани навколишнього середовища $s_1, s_2 \in S$ такі, що $s_1 \neq s_2$ та $see(s_1) = see(s_2)$, то без використання перцепцій агенти би обробляв два сценарії, а з використанням перцепій агент обробляє один сценарій. Інколи обробка цих двох станів в функції прийняття рішення агентом про подальші дії буває надлишковою та складною, а отже використання перцепцій в даному випадку доречне та ефективне.

2.2.3.4. Агенти з внутрішніми станами. У відповідності до [36, 41], агенти з внутрішніми станами — це агенти, які приймають рішення про

подальші дії на основі знань про минулі внутрішні стани агента та/або системи. Структура даного типу агентів схожа на структуру агентів, які використовують перцепції, але на відміну від них агенти даного типу при обчислені функції, яка визначає наступну дію агента як реакцію на зміну навколишнього середовища, враховуються не тільки перцепції, а й внутрішні стани агента.

Якщо визначити отримання агентом інформації про зміну стану навколишнього середовища за допомогою функції

$$see : S \to P,$$

де S — множина станів навколишнього середовища, а P — множина перцепцій агента, то функція вибору наступної дії агентом буде мати наступний вигляд

$$action : I \to A,$$

де I — множина внутрішніх станів агента, а A — множина допустимих дій агента на зміни у навколишньому середовищі.

Для переходу між внутрішніми станами агента вводиться наступна функція, яка проектує поточний внутрішній стан агента та поточну перцепцію у відповідний новий внутрішній стан агента.

$$next : I, P \to I,$$

де I — множина внутрішніх станів агента, P — множина перцепцій.

Функція $next$ виконується агентом перед функцією вибору наступної дії агента в залежності від зміни навколишнього середовища, тобто функцію $action$ можна переписати наступним чином

$$action : next(I, P) \to A,$$

де I — множина внутрішніх станів агента, P — множина перцепцій агента, A — множина допустимих дій агента на зміни у зовнішньому середовищі. При такому перевизначенні функції вибору наступної дії агентом функцію $next$ не потрібно виконувати перед виконанням функції $action$.

Функціонування агента даного типу зводиться до циклу, який складається з таких кроків, як сприйняття змін у навколишньому середовищі та побудова відповідної перцепції за допомогою функції *see*, оновлення внутрішнього стану агента за допомогою виконання функції *next*, вибір та виконання наступної дії за допомогою виконання функції *action*.

2.2.3.5. Агенти, що базуються на цілях. Згідно з [39] агенти, що базуються на цілях — це агенти, які мають множину внутрішніх станів та при виборі дії на зміни у навколишньому середовищі враховують відстань між поточним станом агента та бажаним кінцевим станом. Для досягнення цієї мети агенти даного типу використовують алгоритми пошуку та планування. Основним недоліком даного типу агентів є те, що вони не завжди обирають найбільш оптимальний шлях до бажаного стану.

2.2.3.6. Корисні агенти. У відповідності до [39], корисні агенти — це агенти, які при виборі наступної реакції на зміну у навколишньому середовищі враховують поточний внутрішній стан агента та ваги можливих альтернативних шляхів до бажаного кінцевого результату. Зазвичай агенти даного типу на відміну від агентів, що базуються на цілях, обирають шлях з мінімальною вартістю за допомогою аналізу витрат та винагород, який проводиться для кожного альтернативного шляху від кожного поточного стану до бажаного кінцевого результату.

2.2.3.7. Агенти, що навчаються. Згідно з [39], агенти, що навчаються — це агенти, які мають здатність навчатись на основі минулих дій та змін навколишнього середовища. На початку свого функціонування агент даного типу може не мати ніяких знань, але впродовж взаємодії з навколишнім середовищем агенти накопичують знання та вчаться на основі свого попереднього досвіду, також використовують дані знання у власній роботі. Для агентів, що навчаються, виділяють наступні чотири важливі компоненти, які дозволяють їм навчатись та функціонувати на основі власного попереднього досвіду:

— **Критик** — даний модуль оцінює наскільки ефективно агент працює відносно до встановленого показника ефективності.

— **Навчальні елементи** — даний компонент бере вхідні дані від Критика та допомогає агенту покращити показник ефективності, навчаючись на змінах навколишнього середовища.

— **Елемент продуктивності** — даний модуль приймає рішення про дії, які необхідно виконати для поліпшення роботи агента.

— **Генератор проблем** — даний компонент бере вхідні дані від інших компонент та пропонує дії, які призводять до кращого досвіду.

Для коректної роботи агентів даного типу необхідно детально розробити модель агента з переліком його внутрішніх станів, функцію для вибору реакції на зміну у навколишньому середовищі з врахуванням попереднього досвіду, а також модель навчання агента з усіма взаємозв'язками між компонентами моделі навчання.

2.2.3.8. Агенти, які використовують практичні міркування. У відповідності до [36], агенти, які використовують практичні міркування, — це агенти, які використовують бібліотеку планів для вибору дії, яка буде задовільняти цілі агента.

Бібліотека планів — це множина планів, які можуть бути використані агентом для досягнення цілей агента. Плани не створюються самим агентом, а створюються частиною системи, яка має назву планувальник.

Планувальник — це система або її окрема частина, яка відповідальна за розробку планів на основі таких вхідних даних як ціль агента, намір або завдання агента, поточний стан навколишнього середовища агента (інколи переконання агента), множину можливих реакцій агента на зміни в навколишньому середовищі тощо. Результатом роботи планувальника є побудований за допомогою алгоритмів планування план.

Елементарною одиницею плану є дія і вона може бути описана за допомгою наступних понять:

— назва дії;

— перелік передумов — перелік фактів, без виконання яких не можливе виконання дії;

— перелік видалень — перелік фактів, які стануть не дійсними після виконання дії;

— перелік додавань — перелік фактів, які стануть істиними після виконання дії;

— тощо.

Для визначення можливих дій агента зазвичай використовують наступну математичну модель.

$$Ac = \{\alpha_1, ..., \alpha_n\}$$

Кожен елемент даної моделі може мати наступну структуру.

$$(P_\alpha, D_\alpha, A_\alpha),$$

де

— P_α — множина формул першого порядку, які описують передумови дії α;

— D_α — множина формул першого порядку, які описують ті факти, які стануть хибними після виконання дії α (перелік видалень);

— A_α — множина формул першого порядку, які описують ті факти, які стануть істиними після виконання дії α (перелік додавань);

— тощо.

Тоді задача планування може бути описана наступним чином.

$$(\Delta, \sigma, \gamma),$$

де

— Δ — множина формул першого порядку, які описують переконання агента про початковий стан зовнішнього середовища;

— $\sigma = \{(P_\alpha, D_\alpha, A_\alpha) \| \alpha \in Ac\}$ — множина унікальних дескрипторів кожної дії $\alpha \in A$;

— γ — множина формул першого порядку, які описують цілі/наміри/задачі агента.

План π — це послідовність дій та він може бути визначений наступним чином.

$$\pi = (\alpha_1, ..., \alpha_n)$$

План π для вирішення задачі планування визначає $n + 1$ базу даних намірів агента

$$\Delta_0, ..., \Delta_n,$$

де

$$\Delta_0 = \Delta$$

та

$$\Delta_i = (\Delta_{i-1}\ \Delta\alpha_i \text{ для } 1 \leqslant i \leqslant n.$$

Лінійний план π є застосовним до задачі (Δ, σ, γ) тоді і тільки тоді, коли передумови кожної дії виконуються в попередній базі даних переконань агента.

Оскільки агенти даного типу не створюють нові плани, а використовують вже згенеровані плани з бібліотеки планів, то основною важливою особливістю таких агентів є ознака вибору плану з бібліотеки планів для певної задачі, яка полягає в тому, що поточний стан множини переконань агента має або покривати або збігатись з множиною передумов плану.

2.2.3.9. Гібридні агенти. Згідно з [36], кількість типів агентів надзвичайно велика, а також кожен тип агентів має свої переваги та недоліки. Отже, інколи виникає необхідність у поєднанні переваг різних типів агентів, а також в мінімізації недоліків різних типів агентів за рахунок один одного. Для даної задачі використовуються гібридні агенти. Даний тип агентів може бути використаний лише в межах шарової архітектури, де кожен шар — окремо функціонуюча підсистема. При використанні гібридних агентів розрізнять два види шарової архітектури:

— **горизонтальна** — тип шарової архітектури, в якій кожен програмний шар під'єднаний напряму до входів, тобто сенсорів, та виходів, тобто можливих дій агентів, та може взаємодіяти з будь-яким іншим шаром;

— **вертикальна** — тип шарової архітектури, в якій до входів, тобто сенсорів, та виходів, тобто можливих дій агентів, під'єднується щонайбільше один шар і кожен з інших шарів в якості своїх вхідних даних отримує вихідні дані іншого шару.

В рамках шарової архітектури, особливо вертикальної, розрізняють однонаправлений та двонаправлений потоки управління. Однонаправлений потік передає управління знизу в гору, шар за шаром. При використанні даного потоку агенти нижчого шару отримують в якості вхідних даних вихідні дані суміжного з ним з знизу шару, а свої вихідні дані вони передають в якості вхідних даних агентам суміжного з даним шаром шару. В обох шарових архітектурах кожен шар може складатися з агентів окремого типу, який відмінний від інших типів агентів у інших шарах.

Незважаючи на те, що використання гібридних агентів та шарової архітектури покликані на примноження переваг та зменшення недоліків різних типів агентів в рамках однієї системи, кожна з шарових архітектур має свої недоліки:

— **горизонтальна** — гнучка, оскільки кожен шар функціонує окремо, але при використанні даного типу архітектури можливе виникнення проблем з взаємодією різних типів агентів;

— **вертикальна** — менше взаємодії між шарами, а отже менше проблем з сумісністю різних типів агентів, але це також робить даний тип архітектури менш гнучким.

Тому при використанні гібридних агентів необхідно враховувати сумісність різних типів агентів, їх переваги та недоліки, а також переваги та недоліки їх взаємодії. В загальному на роботу системи впливає функціонування кожної з її підсистем тому ще необхідно враховувати переваги та недоліки обраного типу архітектури, а також детально продумувати архітектурні рішення всередині кожного шару.

2.3. Взаємодія агентів

Більшість задач, які можна вирішити за допомогою використання одного єдиного агента, можна вирішити за допомогою інших простіших технологій та абстракцій. Тому особливий інтерес викликає розв'язування задач з використанням декількох агентів та їх взаємодії. Сама суть використання мультиагентного підходу полягає в розподіленому розв'язанні задачі за допомогою взаємодії окремих агентів, які виконують певні аспекти або ж підзадачі початкової задачі. Згідно з [42–45] взаємодія агентів є важливим аспектом функціонування агентів та може розглядатися в рамках комунікації та кооперації. Дані два поняття тісно пов'язані один з одним. Оскільки без виконання умов обох явищ неможливе жодне з них.

2.3.1. Комунікація агентів. Комунікація в мультиагентних системах зазвичай відбувається шляхом обміну повідомленнями. У відповідності до [42, 46–48] комунікація в межах мультиагентних систем зазвичай реалізується за допомогою використання теорії переговорів. Кожен агент має свої цілі та наміри та інколи їх виконання залежить від інших агентів, тут і постає потреба у використанні комунікації для досягнення мети.

Згідно з [46, 49] визначають наступні типи перемовин:

— **репрезентативні** — результатом даних перемовин є інформування іншого агента або групи агентів;

— **директивні** — результатом даних перемовин є спонукання до виконання певної дії іншим агентом або групою агентів;

— **зобов'язання** — результатом даних перемовин є взяття зобов'язань за виконання певної дії агентом або групою агентів;

— **виразні** — результатом даних перемовин є повідомлення власного внутрішнього стану агентом;

— **декларативні** — результатом даних перемовин є інформування про ультиматум чи фактичну дію.

Оскільки при комунікації агенти повинні розуміти один одного, то вони обмінюються повідомленням у зазделегідь узгодженій формі. У відповідності до [42, 44], зазвичай для цього використовуються онтології та/або мови комунікації між агентами. Згідно з [35, 50, 51], FIPA ACL та KQML є широко застосовними мовами комунікації між агентами.

2.3.1.1. Мова запиту і маніпуляцій даними KQML та формат обміну даними KIF. У відповідності до [50], мова запиту і маніпуляцій даними KQML та формат обміну даними KIF були розроблені за ініціативи ARPA. KQML надає стандартний синтаксис для повідомленнь та визначає типи даних повідомлень, наприклад інформування, відповідь на повідомлення тощо.

KIF слугує для опису синтаксису повідомлень. Для даних цілей у KIF використовуються предикат логік першого порядку.

Як наведено в [50, 51], KQML складається з наступних частин:

— **Шар контенту** — для надання однакового синтаксису повідомлень для агентів, які комунікують.

— **Шар комунікації** — для кодування низькорівневих комунікаційних параметрів таких як ідентифікатори відправника та отримувача повідомлення.

— **Шар повідомлення** — для кодування повідомлення, визначення мережевих протоколів (агенти можуть знаходитись на різних машинах і використовувати глобальну мережу Інтернет для комунікації), знаходження можливих взаємодій з мовним агентом KQML та надання повідомлень.

Зазвичай, згідно з [50], синтаксична структура KQML базується на S-виразах, які зокрема використовувались в мові LISP. Так загальна синтаксична структура повідомлення мовою KQML може мати наступний вигляд:

(KQML perfomative (дія, наприклад інформувати всіх, відправити повідомлення найближчому агенту)

: *sender* < ідентифікатор відправника >

: *receiver* < ідентифікатор отримувача >

: *language* < тип мови >

: *ontology* < онтологія >

: *content* < вираз >

....)

Згідно з [50, 51] основним недоліком KQML є відсутність чіткої семантики.

2.3.1.2. Мова комунікації FIPA ACL. Згідно з [51, 52], FIPA ACL — протокол комунікації агентів, запропонований організацією під назвою Foundation for Intelligent Physical Agents (FIPA). У відповідності до [51, 52], мова комунікації агентів, протоколи взаємодії, дії комунікації та мова контенту є складовими мови FIPA ACL. Назви перфомативі FIPA ACL відмінні від назв таких же за функціоналом пермофативів KQML. Також, згідно з [51, 52], FIPA ACL має чітку семантику.

У відповідності до [51, 52], структура повідомлення в FIPA ACL складається з ряду параметрів, обов'язковим з яких є вказання перфомативу, але більшість повідомлень обов'язково потребуть вказання ідентифікаторів відправника та отримувача повідомлення, вміст повідомлення.

Хоча FIPA ACL підтримує велику кількість дій по надсиланню повідомлень, але згідно з [52] всі вони побудовані на основі дій інформування та запитів.

У відповідності до [51, 52], в FIPA ACL для слідкування за комунікаційними діями агентів використовуються протокол взаємодії, які є частиною бібліотеки протоколів FIPA ACL. Зазвичай для опису протоколів взаємодії використовують розширення UML нотації (AUML) та розфарбовані мережі Петрі.

2.3.1.3. Протокол CNP. CNP — це протокол розподілу завдань, який був запропонований 1980 Р. Дж. Смітом та використовується для розподілу завдань між агентам [53] агент менеджер закріплює завдання за підходящим агентом-підрядником на основі заявок конструктора агентів для цього завдання.

2.3.2. Кооперація агентів. Кооперація агентів визначає тип взаємодії агентів та правила, згідно з якими вони взаємодіють. Згідно з [37, 45], кооперація може використовувати P2P архітектуру або ж керуватись спеціальним агентом для контролю процесу кооперації в межах системи. Система може вирішувати або одну задачу, тоді кооперація агентів націлена на виконання спільної мети, або ж система може мати комплексну архітектуру та вирішувати декілька задач, тоді зазвичай кооперація агентів забезпечує виконання цілей та задач окремих агентів.

У відповідності до [37, 45], необхідність використання кооперації в мультиагентних системах визначає ряд наступних факторів:

— об'єкти функціонують у спільному зовнішньому середовищі;

— обмежені спільні ресурси;

— необхідність прийняття рішень в умовах невизначеності;

— обмеженість компетенцій та знань агентів;

— необхідність синхронізації дій окремих агентів;

— тощо.

Кооперація агентів може відбуватись як в рамках комунікації між двома агентами, так і шляхом комбінування елементів в певні функціональні групи для досягнення спільної мети шляхом досягнення кожним агентом спільних намірів, що розглянуто в [45].

2.3.2.1. Типи об'єднань агентів. Проектування мультиагентних систем є досить складним процесом, тому для його спрощення ввводять поняття ролі. Згідно з [37], роль — це характеристика агента, яка визначає його дозволені, заборонені та обов'язкові дії в межах кооперації з іншими агентами. Так в межах системи мінімізується кількість можливих сценаріїв від кількості агентів в системі до поточної кількості ролей.

В залежності від причини кооперації та задачі у [45] виокремлюють наступні типи об'єднань агентів та закріплені за ними ролей:

— об'єднання агентів, в якому агенти виконують свої зобов'язання по відношенню до інших агентів тоді і лише тоді, коли вони не суперечать

власним цілям та переконанням;

— об'єднання агентів, в якому кожен агент має перелік зобов'язанн та передумов для їх виконання, і який зазвичай використовується для обміну ресурсами та забезпечення підтримки з боку інших агентів;

— об'єднання агентів, в якому агенти розглядаються як один механізм, кожен агент зацікавлений у виконанні загальних цілей, а також його власні цілі не суперечать загальним;

— тощо.

У відповідності до [54], командна робота агентів може визначатись за допомогою теорії спільних намірів. Також існують об'єднання агентів, які конкурують. В даному випадку агентам необхідно координувати власну поведінку та визначити взаємовигідну угоду.

2.3.2.2. Принципи кооперації. Згідно з [42], сформульовано три базових принципи поведінки агентів:

— прогнозування взаємодії;

— розв'язування взаємодії;

— оцінка взаємодії.

Дані принципи набувають в залежності від типу системи та обраної математичної моделі.

Найскладніше проектувати механізми взаємодії при наступних ситуаціях:

— кожна під-задача задачі вирішується хоча б одним агентом;

— всі цілі задачі досяжні за поточних обчислювальних ресурсів;

— дії агентів цілеспрямовані та несуперечливі;

— агенти формують рішення задачі у формі, в якій дане рішення могло б інтегруватися в більш глобальне рішення;

— тощо.

У відповідності до [45], кооперація можлива при наступних сценаріях:

— агенти мають спільну, але не завжди явну, мету, яку жоден агент не може досягти самостійно;

— дії агентів дозволяють досягти не лише власних цілей, а й цілей інших агентів;

— тощо.

Також, у відповідності до [45], можливі наступні причини невдалої взаємодії агентів:

— відсутність у агентів дій та цілей, які задовільняють визначення та потреби поняття кооперація;

— агенти не можуть отримати інформацію про те коли необхідно взаємодіяти з іншим агентом або групою агентів;

— недостатні апаратні потужності для забезпечення взаємодії між агентами;

— наявність непередбачуваних зовнішніх ефектів, які заважають взаємодії між агентами;

— тощо.

Проектування взаємодії агентів є складним процесом та має враховувати типи агентів, які взаємодіють, тип взаємодії, особливості задачі та наявні ресурси.

2.3.3. Архітектура мультиагентних систем. Навіть найпростіші агенти для свого функціонування мають необхідну архітектуру, наприклад, згідно з [40], шар для під'єднання до системи, мережевий інтерфейс, шар для виконання завдання є складовими примітивної архітектури агента. Але для вирішення складних розподілених задач використовують мультиагентні системи, архітектура яких в рази складніша за архітектуру одного агента. Так класичними прикладами архітектур мультиагентних систем є архітектура мультиагентних систем на основі логік, реактивна архітектура, шарова архітектура та архітектура на основі переконань, цілей та бажань агентів тощо.

2.3.3.1. Архітектура мультигентних систем на основі логік. Згідно з [41], архітектура мультиагентних систем на основі логік — це підхід до побудови інтелектуальних агентів, їх взаємодії та комунікації в якості автоматичних довідників теорем, які розглядають інтелектуальну поведінку агентів як генерацію символьного представлення зовнішнього середовища агента

та бажаної поведінки агента. Окрім того, що в рамках даної архітектури агент розглядається як автоматичний довідник теорем, існує множина формул першого порядку, які описують переконання агента про його зовнішнє середовище та відповідні дії агента в залежності від його намірів. Агент виконує певні дії коли формула першого порядку з даної множини набуває істиних значень. Взаємодія агентів проектується також за допомогою використання формул першого порядку.

2.3.3.2. Реактивна архітектура. У відповідності до [41], реактивна архітектура — це спосіб побудови мультиагентних систем з використанням реактивних агентів. Зазвичай в системах даної архітектури для опису поведінки виконання завдання використовують не символьне представлення, а скінченний автомат. Головна особливість цієї архітектури полягає в тому, що агенти будують перцепцію на основі вхідної інформації та проектують її на відповідну дію з множини допустимих дій агента. Дана особливість також є описом поведінки агента в межах системи. Кожен агент будує ієрархію його можливих типів поведінки (пари персепції та відповідної дії агента). Якщо більше ніж один доречний тип поведінки може бути використаний в даній ситуації, то буде обраний той тип поведінки, який знаходиться вище згідно даної ієрархії. Більша складність системи досягається шляхом використання підходів кооперації агентів, описаних в минулому підрозділі.

2.3.3.3. Архітектура на основі переконань, цілей та бажань агентів. Згідно з [35, 36, 41], архітектура на основі переконань, цілей та бажань агентів — це тип архітектури мультиагентних систем, який використовує для характеристики агента такі ментальні поняття як переконання, цілі, наміри та бажання агента. Переконання описують уявлення агента про навколишнє середовище. Наміри описують цілі агента, які не були обрані агентом для досягнення в даний момент часу. Бажання — це цілі, які агент обрав для реалізації в даний момент часу. Взаємодія агентів даного типу архітектури залежить від архітектури системи та агентів, задачі. Деякі підходи до організації взаємодії між агентами даної архітектури описані в минулому підрозділі. Так, наприклад, агенти можуть

бути згруповані в певні об'єднання, а також за потреби можливе використання механізмів спільних зобов'язань та спільних намірів.

2.3.3.4. Шарова архітектура. Як зазначено в [41], шарова архітектура— це спосіб створення мультиагентних систем, який заснований на формуванні шарів, які складаються з агентів, та конструювання взаємодії даних шарів. Зазвичай кожен шар складається з довільної кількості агентів одного типу.

Розрізняють вертикальну та горизонтальну шарові архітектури. Основна відмінність між ними полягає у взаємодії між шарам. В горизонтальному типі даної архітектури кожен шар на пряму підключений до вхідних даних, тобто перцепцій, та вихідних даних, тобто можливих дій агента. Також в межах горизонтальної шарової архітектури можлива взаємодія між собою різних шарів. Натомість у вертикальній шаровій архітектурі кожен шар взаємодія лише з двома своїми сусідніми шарами. Тобто у вертикальній шаровій архітектурі тільки один шар напряму під'єднаний до вхідних даних, тобто перцепцій, та тільки один шар напряму під'єднаний до вихідних даних, тобто можливих дій агента. У вертикальній архітектурі шари взаємодіють один з одними у ієрархічній послідовності за допомогою використання одно- або дво- напрвленого потоку управління.

2.4. Огляд фреймворків та додатків для розробки мультиагентних систем

Оскільки існує велика кількість типів агентів, архітектур мультиагентних систем, способів та принципів взаємодії агентів (комунікації та кооперації), а також існує великий попит на використання мультиагентних систем, то й існує велика кілька кількість додатків та фреймворків для створення мультиагентних систем для різних мов програмування, типів агентів, різних типів архітектур мультиагентних систем та різних способів взаємодії агентів. На даний момент широко використовуються такі об'єктно-орієнтовані мови програмування як Python та Java, тому в даному підрозділі розглянуті наступні фреймоврки та

додатки для створення агентів на даних мовах програмування: Jade, Jack, Jason, Pade, Mesa, Spade.

2.4.1. Jade. У відповідності до [54–57], Jade — це фреймворк для створення мультиагентних систем на мові програмування Java, який доступний на офіційному сайті [58]. Даний фреймворк надає можливість створення різних типів агентів, зокрема реактивних та агентів на основі намірів, бажань та переконань. Згідно з [56, 57], даний фреймворк також надає можливість проектування та реалізації взаємодії між агентами за допомогою використання протоколу FIPA ACL. У відповідності до [57], Jade є фреймворком, який підтримує повний цикл проектування та розробки агентів, зокрема даний фреймворк надає наступний функціонал:

— створення та підтримка середовища для функціонування агентів;
— простий спосіб створення агентів за допомогою використання перевизначення поведінки, наслідування та бібліотеки класів;
— створення та підтримка графічного додатку для відображення та управління функціонування інтелектуальних агентів;
— підтримка мови FIPA ACL;
— можливість відладки мультиагентної системи та окремих агентів за допомогою великої кількості додатків для налогодження роботи програми;
— можливість паралельного виконання та перегляду поведінки агентів;
— можливість створення та управління різними типами агентів;
— можливість організації агентів у відповідності до різних архітектур мультиагентних систем;
— тощо.

Всі агенти, які створені за допомогою даного фреймворка запускаються незалежно один від одного на окремих хостах. Згідно з [56, 57], при створені агентів за допомогою використання фреймворка Jade всі вони можуть наслідуватись від різних класів в залежності від типу агента, але в кінцевому випадку всі вони будуть наслідуватись від єдиного батьківського класу. Також у

відповідності до [55], фреймворк має батьківський контейнер, який від початку свого створення містить два агенти:

— **AMS** — агент, який надає каталог з інформацію про доступних в межах платформи та проекту агентів. Доступ до даного агента можна отримати за допомогою класу агента DF та його статичних методів (register, deregister, modifyandsearch);

— **DF** — агент, який контролює події в межах платформи та відповідальний за додавання та видалення агентів з платформи;

У відповідності до [55, 57], для створення агента за допомогою використання фреймворка Jade розробнику необхідно наслідувати клас *Agent*, який надає наступний функціонал:

— підтримка ACL повідомлень;

— планування та виконання декількох дій одночасно;

— тощо.

Згідно з [56], для реалізації поведінки агента, створеного за допомогою даного фреймворку, необхідно наслідувати клас *jade.Behaviour* для перевизначення функцій подій для даного агента та даної події. В процесі визначення поведінки агента розробник може перевизначити наступні функції класу *jade.Behaviour*:

— *action*() — послідовність кроків, яка має бути виконана при даній події;

— *done*() — умова, яка визначає чи виконане завдання;

— тощо.

Згідно з [56], фреймворк Jade окрім підтримки стандартів FIPA ACL підтримує цикл існування агента, який був запропонований FIPA:

— **Етап ініціалізації** — агент вже створений, але ще не зареєстрований агентом AMS.

— **Етап активації** — агент зареєстрований агентом AMS, має ім'я та може взаємодіяти з іншими агентами.

— **Етап призупинення** — робота агент зупинена через призупинення потоку, в якому виконується агент.

— **Етап очікування** — агент заблокований та очікує на виконання певної

дії.

— **Етап видалення** — агент закінчив свою роботу, потік агент закінчив своє виконання і вони були видалені з реєстру агентом AMS.

— **Етап транспортування** — агент був переміщений в нове місце знаходження.

Jade забезпечує створення, налаштування та підтримку мультиагентних систем в простій та зрозумілій формі.

2.4.2. Jack. Згідно з [59], Jack — це комерційний крос-платформний фреймворк для створення мультиагентних систем мовою програмування Java, який був розроблений для використання в якості частини складніших систем та доступний на офіційному сайті [60]. Перша версія даного фреймворку була випущена в 1997 році випускником Австралійського Інституту Штучних Технологій. Натомість випуск останньої стабільної версії Jack датується липнем 2015 року.

Агенти, які створені за допомогою використання даного фреймворку, в межах інших програм виглядають як звичайні об'єкти, які мають доступ до відповідних компонент системи. Дані агенти не використовують жодної спеціальної мови для взаємодії між собою. Наведений фреймворк розвивався як множина незалежних одна від одної та розширюваних частин.

У відповідності до [59], фреймворк складається з наступних розширень мови програмування Java:

— набір синтаксичних доповнення (ключові слова, набір тверджень для визначення атрибутів та інших властивостей компонентів, набір тверджень для визначення статичних відносин, набір маніпуляцій станами агентів тощо);

— компілятор, який трансформує синтаксичні доповнення в класи та твердження мови програмування Java;

— ядро (множина класів), яке надає підтримку згенерованого коду в реальному часі.

У відповідності до [59], розробник може розширити або змінити архітектуру агента за допомогою використання плагінів. Так за допомогою використання плагінів можна забезпечити реалізацію та підтримку агентів на основі бажань, намірів та переконань. Також за допомогою використання даних розширень можна додати та визначити мову для взаємодії агентів один з одним.

Згідно з [59], в межах даного фреймворку доступне середовище для розробки та виконання агентів, яке надає наступний функціонал:

— запуск проектів;

— графічний додаток для написання планів та їх під'єднання до агентів;

— управління взаємодією агентів один з одним;

— компіляція проектів;

— тощо.

Також агент, які створені шляхом використання даного фреймворку, можуть бути розроблені за допомогою мови програмування Java та мови для створення та розроблення планів фреймворку Jack в будь-якому середовищі виконання мови програмування Java. Так наприклад для використання агентів, які створені за допомогою даного фреймворку, в Eclipse достатньо додати новий плагін. Загалом розробка агентів за допомогою даного фреймворку в середовищі виконання мови програмування Java є більш зручною для великих мультиагентних систем з великим зовнішнім середовищем.

У відповідності до [59], Jack має наступні переваги:

— крос-платформеність;

— модульна архітектура;

— легка інтеграція з іншими системами;

— можливість використання декількох типів агентів в межах одного проекту;

— підтримка групових міркувань агентів;

— підтримка різних протоколів взаємодії агентів (FIPA, KQML тощо) шляхом використання плагінів;

— інтеграція з кодом мови програмування Java шляхом використання інструментів для забезпечення управління проектами (Maven, Gradle

тощо);

— тощо.

Також згідно з [59], Jack має наступні недоліки:

— вартість;

— відсутність вбудованої підтримки J2EE та XML;

— велике споживання ресурсів;

— тощо.

Основна зручність використання агентів, які побудовані за допомогою використання фреймворку Jack, є можливість інтеграції та використання їх з будь-якими технологіями та кодом мови програмування Java у будь-якому середовищі розробки мови програмування Java.

2.4.3. Jason. У відповідності до [37, 61], Jason — це безкоштовний інтерпретатор для розширення мови взаємодії агентів AgentSpeak, який доступний на офіційному сайті [62]. Згідно з [37, 61], даний фреймворк є мовою на основі логік для створення агентів на основі бажань, намірів та переконань. Даний фреймворк надає широкий спектр функціоналу, який згаданий в [61] та включає наступні функції:

— створення, налагодження та спостереження за агентами на основі переконань, бажань та намірів;

— підтримка протоколу взаємодії агентів FIPA;

— підтримка функціонування розподілених агентів за допомогою використання глобальної мережі Інтернет;

— власна мова для створенння агентів на основі переконань, намірів та бажань;

— підтримка інтеграції з іншими системами (Cartago, ROS тощо);

— підтримка архітектури мультиагентних систем на основі логік;

— тощо.

Jason є зручним фреймворком на мові програмування Java для створення мультиагентних систем з використанням агентів на основі переконань, бажань

та намірів. Також даний фреймворк є простим у вивченні та використанні.

2.4.4. Pade. У відповідності до [63, 64], Pade — крос-платформне безкоштовне середовище для розробки, виконання та управління мультиагентними системами на мові програмування Python, яке доступне на офіціному сайті [63], а також може бути інстальоване за допомогою використання команди pip install pade. Дана технологія має детальну документацію [63, 64].Відповідно до неї Pade розповсюджується під ліцензією MIT та був розроблений Smart Grids Group (GREI) з Федерального Університету Сеара з використанням бібліотеки Twisted. А також оскільки дане середовище є середовищем з відкритим кодом, то його код може бути завантажений з репозиторію GitHub і модифікований в подальшому.

Згідно з [63], для створення агента за допомогою даної технології необхідно створити клас агента та наслідувати його від классу *Agent* фреймворку Pade. Також, у відповідності до [63, 64], Pade надає наступний функціонал:

— створення та виконання агентів в реальному часі;
— використання абстракцій об'єктно-орієнтованого програмування при розробці агентів;
— підтримка мови взаємодії агентів FIPA ACL;
— підтримка циклічної поведінки агентів;
— фільтрування повідомлень агентів;
— інтеграція з базами даних;
— тощо.

Pade забезпечує створення, налаштування та підтримку мультиагентних систем в простій та зрозумілій формі, а також надає весь необхідний функціонал для створення, виконання та управління мультиагентними системами.

2.4.5. Mesa. Згідно з [65], Mesa — це фреймворк для розробки та управління мультиагентними системами на мові програмування Python, який доступний разом з детальною документацією на офіціному сайті [65], а також може бути інстальований за допомогою використання команди pip install mesa. У

відповідності до [65], Messa розповсюджується під ліцензією Apache2 та основними особливостями даного фреймворку є:

— модульність складових компонент;
— візуалізація за допомогою використання браузерів;
— вбудовані інструменти для аналітики функціонування мультиагентної системи, а також окремих агентів.

Як наведено в [65], функціонал фреймворку Mesa схожий з функціоналом аналогічних систем, таких як NetLogo, Repast, та MASON і надає наступні можливості:

— створення моделей агентів шляхом використання вбудованих компонент або власних реалізацій;
— візуалізація моделей агентів за допомогою використання браузерів;
— аналіз роботи моделі та окремих агентів за допомогою інструментів для аналізу даних мови програмування Python;
— тощо.

Кожен розробник може внести свій вклад у розвиток даного фреймворку шляхом надсилання ідей та готових реалізацій розширень для Mesa, який є зручним інструментом на мові програмування Python для симуляції виконання роботи мультиагентних систем, що дозволяє використовувати даний фреймворк в наукових та дослідницьких цілях.

2.4.6. Spade. У відповідності до [66, 67], Spade — це платформа для мультиагентних систем, яка написана на мові Python і доступна на офіційному сайті [66]. Дана платформа розповсюджується під ліцензією MIT та має детальну документацію, яка доступна за наступним посиланням [67].

Згідно з офіційною документацією, Spade має наступні особливості:

— використання Asyncio при розробці даної платформи;
— мультиагентна платформа базується на XMPP;
— сповіщення про присутність дозволяють платформі дізнаватись поточний стан агентів у режимі реального часу;

— веб інтерфейс;

— модель агентів базується на їх поведінці;

— інтеграція з будь-яким XMPP сервером;

— підтримка метаданих FIPA з використанням XMPP;

— тощо.

Згідно з [66, 67], функціонал даної платформи може бути розширений за допомогою використання плагінів. Так, наприклад, до даного функіоналу можна додати підтримку агентів на основі бажань, намірів та переконань, а також можливість побудови графіків шляхом використання відповідних плагінів, які доступні за наступними посиланнями [68, 69].

РОЗДІЛ 3
ІНТЕЛЕКТУАЛЬНІ СИСТЕМИ УПРАВЛІННЯ НАВЧАЛЬНИМ ПРОЦЕСОМ

Мультиагентна інтелектуальна система управління навчальним процесом — це система, яка поєднує в собі властивості інтелектуальної системи управління навчальним процесом та мультиагентного підходу, а також переваги та недоліки обох своїх складових. Тобто мультиагентні інтелектуальні системи управління навчальним процесом є підтипом інтелектуальних систем управління навчальним процесом, які, у відповідності до [1], можуть бути описані за допомогою означення 3.1.

Означення 3.1. Інтелектуальні системи управління навчальним процесом — це системи управління навчальним процесом, які враховують побажання користувачів про реактивність системи та потреби користувачів в потужніших інструментах для управління навчальним процесом.

Для задоволення потреб та бажань користувачів при створенні інтелектуальних систем управління навчальним процесом використовують різні типи аналізу даних, мультиагентний підхід, онтології тощо.

3.1. Функціонал інтелектуальних систем управління навчальним процесом

Оскільки інтелектуальні системи управління навчальним процесом є розширенням систем управління навчальним процесом, то інтелектуальні системи управління навчальним процесом мають надавати базовий обов'язковий функціонал систем управління навчальним процесом, який наведений зокрема в [5–7] та включає ряд наступних функцій:

— створення та управління курсами;

— підтримка синхронної та/або асинхронної комунікації;
— управління особистими даними користувачів;
— підтримка механізму ролей та доступ до навчальних матеріалів в залежності від прав доступу користувача певної ролі;
— забезпечення створення звітів;
— тощо.

Також інтелектуальні системи управління навчальним процесом мають забезпечувати реалізацію нефункціональних вимог систем управління навчальним процесом та можуть включати довільні додаткові функції систем управління навчальним процесом.

Також згідно з [1, 70], інтелектуальні системи управління також мають надавати функціонал, який є специфічними для інтелектуальних систем управління навчальним процесом, не є складовою класичних систем управління навчальним процесом, а також вирішує задачі, які є типовими для інтелектуальних систем управління навчальним процесом. У відповідності до [1, 70], даний функціонал інтелектуальних систем управління навчальним процесом може включати наступні функції:

— створення тестів з множини питань з курсу;
— різні типи аналізу поведінки користувачів;
— відслідковування рівня знань учнів/студентів в межах навчального процесу;
— динамічне створення розкладу навчального процесу;
— можливість генерації різних видів звітів для викладачів про їх курси та учнів/студентів, що проходять дані курси;
— тощо.

Інтелектуальні системи управління навчальним процесом хоча й схожі на класичні системи управління навчальним процесом, задовольняють ті ж самі обов'язкові вимоги, але при цьому досить відрізняються від них. Оскільки інтелектуальні системи управління навчальним процесом задовольняють такі потреби користувачів як реактивність, аналітичність,

автономність функціонування системи.

3.2. Структура інтелектуальних систем управління навчальним процесом

Досить часто при розробці інтелектуальних систем управління навчальним процесом використовують аналіз даних та мультиагентний підхід. Оскільки побудова інтелектуальної системи управління навчальним процесом разом з використанням аналізу даних схожа на побудову системи управління навчальним процесом та відрізняється лише додаванням додаткових модулів для забезпеченння функцій, що базуються на аналізі даних, то в даному підрозділі розглянуто способи побудови інтелектуальних систем управління навчальним процесом на основі агентів та основні особливості різних типів аналізу даних при проектуванні та розробці інтелектуальних систем управління навчальним процесом.

3.2.1. Побудова інтелектуальних систем управління навчальним процесом з використанням мультиагентного підходу. Мультиагентний підхід до побудови систем — це спосіб побудови систем з використанням агентів та взаємодії між ними. У відповідності до [71–74], розрізняють два типи побудови інтелектуальних систем управління навчальним процесом з використанням мультиагентних систем: інтеграція агентів у систему управління навчальним процесом та побудова системи повністю на основі агентів.

3.2.1.1. Інтеграція агентів у систему управління навчальним процесом. Як було зазначено у [71], даний спосіб побудови інтелектуальних систем управління навчальним процесом базується на побудові класичної системи управління навчальним процесом та додавання агентів до даної системи з використанням інших компонент системи в якості середовища для функціонування агентів. Так агенти зможуть шляхом взаємодії з середовищем отримувати наступну інформацію для виконання своїх функцій: відомості про

курси та користувачів, інформацію про відвідуваність та оцінки, результати тестування тощо. Згідно з [71], при використанні даного підходу часто використовують наступні агенти:

— **Агент викладач** — для побудови моделі викладача.
— **Агент студент** — для побудови навчальної моделі студента.
— **Агент управління** — для надання контролю за взаємодією агентів один з одним та з навколишнім середовищем.
— **Агент бази даних** — для доступ та маніпуляцій з даними з бази даних системи управління навчальним процесом.
— тощо.

Складність даного підходу полягає в доступі до даних, які можуть бути організовані не належним чином для їх використання у розподіленому підході для вирішення задач. Приклад даної системи, який демонструє використання ролей користувачів в даних системах, наведено в [17].

3.2.1.2. Інтелектуальна система управління навчальним процесом на основі агентів. Згідно з [71], даний підхід до побудови інтелектуальних систем управління навчальним процесом полягає у проектуванні та реалізації системи базуючись повністю на використанні агентів та взаємодії між ними. Так у відповідності до [71, 74], зазвичай в даних системах використовують наступні типи агентів:

— **Агент викладач** — для побудови моделі викладача;
— **Агент студент** — для побудови навчальної моделі студента;
— **Керівник навчання** — для координації дій усіх залучених у навчальний процес агентів;
— тощо.

Основною перевагою даного підходу є відмовостійкість системи. В найгіршому випадку може перестати працювати агент під назвою Керівник навчання, тоді не будуть встановлюватись нові зв'язки між студентами, викладачами та курсами, але в той же час все, що було створено раніше, буде функціонувати у звичайному

режимі. При відмові одного з агентів студента або викладача система буде недоступною лише для одного користувача. Приклад використання даного підходу для побудови інтелектуальної системи управління навчальним процесом наведено в [73]. Даний приклад демонструє те, що система на основі даного підходу може бути більш реактивною та гнучкою ніж аналогічні системи, які створені за допомогою використання інших підходів. При використанні даного підходу можливе створення неоднорідного навчального середовища, в якому будуть враховуватись індивідуальні навчальні вподобання студентів, їх навчальний рівень та їх швидкість навчання в загальному навчальному процесі групи студентів.

3.2.1.3. Подальший розвиток побудови інтелектуальних систем управління навчальним процесом з використанням мультиагентного підходу. Кожен з вище наведених підходів має свої переваги та недоліки, але існує великий спектр задач та систем, де дані підходи можуть бути використані. Згідно з [72], дані підходи активно розвиваються та надалі можуть бути вдосконалені шляхом використання наступних агентів:

- **Агент для створення персоналізованих рекомендацій** — для створення рекомендацій для студента на основі його поведінки та навчальної моделі.
- **Агент планування** — для створення планів по покращенню успішності студентів на основі їх минулої успішності, поведінки та навчальної моделі.
- **Агент комунікації** — для збору та аналізу частоти, типів та напрямків комунікації користувача.
- **Агент тестування** — для тестування студентів на початку та вкінці курсу з метою аналізу відповідності вмісту курсу навчальним можливостям та потребам студента.
- тощо.

В залежності від типу задачі та системи залежить і підхід до побудови системи, а також його адаптація та модифікація в межах системи. Тому дані підходи

будуть розвиватись до тих пір, поки існує попит на використання мультиагентного підходу в проектуванні інтелектуальних систем управління навчальним процесом.

3.2.2. Використання аналізу даних при побудові інтелектуальних систем управління навчальним процесом. Існує велика кількість видів аналізу даних та їх способів використання при проектуванні та розробці класичних та інтелектуальних систем управління навчальним процесом, деякі з них та їх застосування в інтелектуальних системах управління навчальним процесом розглянуті в даному підрозділі.

3.2.2.1. Розвідувальний аналіз даних. У відповідності до [75, 76], основним завданням розвідувального аналізу даних є перетворення та/або графічна інтерпретація даних з метою попереднього аналізу цих даних та інформації, яку вони описують. Згідно з [75, 76], даний вид аналізу даних використовує наступні методи:

— для аналізу однієї змінної:

- пробіт-графік;
- ймовірнісний графік;
- звисні гістобари;
- зображення "Скринька з вусами";
- зображення "Стебло-листок";
- підвішена коренеграма;

— для аналізу декількох змінних:

- таблиця спряженості;
- діаграма розсіювання.

У відповідності до [77], розвідувальний аналіз даних при побудові інтелектуальних систем управління навчальним процесом використовується для визначення поведінкових патернів користувачів з метою надання персоналізованого навчального плану в подальшому.

3.2.2.2. Кореляційний аналіз даних. Згідно з [75], даний вид аналізу даних використовується для встановлення факту наявності зв'язку між двома величинами або ж між двома групами величин шляхом визначення коефіцієнтів кореляції, аналізу отриманих коефіцієнтів кореляції та виявленого зв'яку, а також побудови довірчого інтервалу для отриманих коефіцієнтів кореляції.

У відповідності до [76, 77], кореляційний аналіз даних при розробці та реалізації інтелектуальних систем управління навчальним процесом використовується для встановлення факту зв'язку між двома величинами, наприклад між поведінкою студента та вподобаннями у стилі навчання, або ж залученістю студента в активну комунікацію під час заняття та успішністю студента тощо.

3.2.2.3. Дисперсійний аналіз даних. У відповідності до [78], основним завданням даного виду аналізу даних є аналіз результатів, які на пряму чи дотично залежать від якісних оцінок. Також, згідно з [78], виділяють одно- та багато- факторний дисперсійний аналіз даних.

Дисперсійний аналіз даних при проектуванні та побудові інтелектуальних систем управління навчальним процесом використовується для дослідження зміни певних показників в залежності від зміни певних факторів, наприклад, залежність зміни успішності студента в залежності від зміни темпу викладання навчальної дисципліни тощо.

3.2.2.4. Регресійний аналіз даних. У відповідності до [78, 79], даний вид аналізу даних використовується для побудови математичного представлення моделі зв'язку між величинами. Згідно з [78, 79], в залежності від вигляду даного математичного представлення моделі розрізняють лінійну регресію, поліноміальну регресію тощо.

Згідно з [80], регресійний аналіз даних використовується при проектуванні та розробці інтелектуальних систем управління навчальним процесом з метою побудови моделі залежності успішності студентів від різних складових навчального процесу.

3.2.2.5. Коваріаційний аналіз даних. У відповідності до [81], даний вид аналізу даних використовується для побудови математичної моделі залежності величини від якісних та/або кількісних чинників. Тобто в певному сенсі даний вид аналізу даних є поєднанням регресійного та дисперсійного видів аналізу даних.

Згідно з [82], коваріаційний аналіз даних використовується для оцінки та побудови математичної моделі впливу поведінки студентів на їх навчальні досягнення.

3.2.2.6. Дискримінантний аналіз даних. Згідно з [83], даний вид аналізу даних використовується для пошуку елементів, що відрізняються в різних вибірках даних. Зазвичай в рамках даного виду аналізу даних в якості залежної змінної використовується змінна з певного заздалегідь заданого інтервалу значень.

У відповідності до [84], даний вид аналізу даних використовується при проектуванні та реалізації інтелектуальних систем управління навчальним процесом для відслідковування зміни різних показників системи та користувачів в залежності від зміни часу, наприклад відслідковування зміни рівня зацікавленості студентів в залежності від періоду навчання тощо.

3.2.2.7. Кластерний аналіз даних. У відповідності до [85], даний вид аналізу даних використовується для групування множини об'єктів у кластери, складові яких є однотипними. Також, згідно з [85], даний вид аналізу даних здебільшого грунтується на підрахунку відстані між заданим об'єктом та кластерами, що розглядаються.

Згідно з [86], кластерний аналіз даних використовується при проектуванні та реалізації інтелектуальних систем управління навчальним процесом для групування множин певних об'єктів у менші підгрупи в залежності від заздалегідь визначених факторів, наприклад кластеризація курсів в залежності від їх наповнення тощо.

3.3. Класифікація інтелектуальних систем управління навчанням

Інтелектуальні системи управління навчальним процесом виникли як розширення систем управління навчальним процесом та розширюють їх функціонал шляхом додавання великої кількості нового функціоналу, який не надається системами управління навчальним процесом та не може бути наданий за допомогою простих алгоритмів, методів та технологій. Тому інтелектуальні системи управління навчальним процесом прийнято класифікувати в залежності від даного додаткового функціоналу, а також в залежності від використовуваних технологій та алгоритмів.

Класифікація даних систем в залежності від технологій та алгоритмів є досить великою та базується на перевагах та недоліках як інтелектуальних систем управління навчальним процесом, так і на перевагах та недоліках використовуваних технологій та алгоритмів. Здебільшого задачі визначають використовувані підходи, алгоритми та технології, тому в даному підрозділі розглянута класифікація інтелектуальних систем управління навчальним процесом в залежності від додаткового функціоналу, який вони надають. Так у [72] виділяють наступні типи інтелектуальних систем та їх основні характеристики:

— адаптивні системи управління навчальним процесом;

— інтелектуальні репетиторські системи;

— адаптивні освітні гіпермедіа системи;

— тощо.

3.3.1. Адаптивні системи управління навчальним процесом. Адаптивні системи управління навчальним процесом — це інтелектуальні системи управління навчальним процесом, які складаються з моделі контенту, моделі інструктора та моделі учня. Основною метою даного типу інтелектуальних систем управління навчальним процесом є розподіл та доставка персоналізованого вмісту курсів на основі вподобань та стилю учня.

3.3.2. Інтелектуальні репетиторські системи. Інтелектуальні репетиторські системи — це інтелектуальні системи управління навчальним процесом, які складаються з моделі бази знань, моделі учня, моделі викладача, системи управління та штучного інтелекту компоненти. Даний тип інтелектуальних систем управління навчальним процесом додатково надають наступний функціонал:

— моніторинг активностей користувача;

— визначення рівня знань учня;

— розробка підходящих адаптивних навчальних інструкцій для кожного учня на основі його навчального рівня та вподобань;

— надання можливості учням залишати відгуки про систему, навчальні матеріали та рекомендації;

— тощо.

Основною метою інтелектуальних репетиторських систем є побудова індивідуального навчального плану для кожного учня на основі матеріалів курсу, рівня знань учня, навчальних вподобань учня тощо. В рамках даних систем учню можуть бути запропоновані інші курси замість поточного в залежності від його навчальної моделі. Зазвичай дані системи використовуються для викладання курсів та організації навчального процесу під час репетиторства.

3.3.3. Адаптивні освітні гіпермедіа системи. Адаптивні освітні гіпермедіа системи — це інтелектуальні системи управління навчальним процесом, які збирають та опрацьовують особисті дані учня та створюють сервіси з наступним функціоналом:

— побудова індивідуального навчального плану для кожного учня;

— отримання навчальної моделі кожного учня (уподобань, інтересів, знань в даній області тощо);

— покращення індивідуального навчального прогресу кожного учня;

— тощо.

Адаптивні освітні гіпермедіа системи підбирають матеріали учню в залежності від навчальної моделі учня.

3.4. Огляд подібних систем

Оскільки існує велика кількість типів інтелектуальних систем, підходів та технологій до їх реалізації, то існує і велика кількість вже реалізованих інтелектуальних систем управління навчальним процесом, які задовільняють різноманітні потреби користувачів. Оскільки більшість сучасних систем управління навчальним процесом, зокрема Moodle, Blackboard Learn та Decobo, використовують в тій чи іншій мірі аналіз даних, то в даному підрозділі розглянуті деякі з інтелектуальних систем управління навчальним процесом, які були створені за допомогою використання мультиагентного підходу.

3.4.1. ALLEGRO. ALLEGRO — це мультиагентне середовище для навчання, яке було описане М. Вікаррі, Деметріо А. Овалле та Джовані А. Джіменез в [87]. Дана система було розроблена за допомогою використання автономних, гнучких та адаптивних агентів, аргументованого виведення, взаємодії різних навчальних об'єктів та може використовуватись як в школах так і у вищих навчальних закладах, де є потреба та можливості впровадження індивідуального навчання.

Оскільки ALLEGRO надає мінімальний функціонал систем управління навчальним процесом та додаткові функції, які були реалізовані за допомогою використання наступних штучних та символізуючих користувачів системи агентів:

— штучні агенти:

- **Агент Вихователь** — для управління навчальним процесом.
- **Агент Інтерфейс** — для встановлення зв'язку між штучними та символізуючими користувачів системи агентами.
- **Агент Експерт** — для управління властивими для його навчальної області знаннями.

- **Агент Діагност** — для визначення рівня знань студента.
- **Агент Спільнота** — для управління співпрацею студентів, які вивчають однаковий предмет та/або мають однакову проблему.

— символізуючі користувачів системи агенти:

- **Агент Студент** — для моделювання користувача системи, основною ціллю якого є отримання нових знань та навичок.
- **Агент Модератор** — для моделювання користувача системи, який управляє навчальним процесом.
- **Агент Викладач** — для моделювання користувача системи, який взаємодіє з студентами для забезпечення знань та допомоги в їх засвоєнні студентам, управління курсами, відслідковування прогресу студентів та розробки персональних та/або командних рекомендацій.

Основною перевагою ALLEGRO є індивідуальне та командне навчання, але в той же час дана система не підтримує способів комунікації між студентами напряму та інших способів входження в систему окрім використання логіну та паролю.

3.4.2. MAS-PLANG. Згідно з [88], MAS-PLANG — це мультиагентна веб система для дистанційного навчання, яка була розроблена за допомогою використання мови програмування Java, JavaScript, Flash, XML. Побудована на основі використання даних технологій система окрім мінімального функціоналу для систем управління навчальним процесом надає користувачам наступний функціонал:

— вдосконалене управління особистими даними користувачів системи;

— розсилка нагадувань;

— вибір відповідної педагогічної стратегії для організації навчальних ресурсів;

— аналіз поведінки користувачів;

— управління та розповсюдження курсів;

— генерування зворотнього зв'язку користувачам системи про їх поведінку;

— тощо.

При проектуванні та розробці MAS-PLANG була використана вертикальна двошарова архітектура мультиагентних систем, в якій на вищому рівні розташовані агенти інтерфейсу, а на нижчому інформаційні агенти. Для взаємодії між агентами в даній системі використовується протокол FIPA. В межах MAS-PLANG функціонують наступні інтерактивні, автономні, про-активні, навчальні агенти:

— **Програмований агент** — для автоматичного управління завданнями та заняттями.

— **Синтетичний агент** — для надання повідомлень іншим агентам про особливий характер поведінки студента.

— **Агент Монітор** — для відслідковування поведінки користувачів системи для генерації зворотнього зв'язку користувачам системи про їх дії в подальшому.

— **Навігаційний агент** — для управління зверненнями до бази даних, а також взаємодією між дидактичними агентами та агентами користувача.

— **Агент користувача** — для створення та управління навчальною моделлю користувача (студента).

— **Дидактичний агент** — для вибору відповідної педагогічної стратегії для організації навчальних ресурсів.

— тощо.

MAS-PLANG є зручною у використанні мультиагентною інтелектуальною веб системою управління навчальним процесом, функціонал якої може бути розширений за допомогою додавання та переходу до використання сучасних технологій та алгоритмів, нових шарів агентів, а також нових типів агентів та взаємодії між ними.

3.4.3. I-MINDS. Згідно з [89], I-MINDS — комп'ютерна кооперативна система управління дистанційним навчальним процесом, при розробці якої використовувався мультиагентний підхід. Дана система надає користувачам

наступний функціонал:

— управління особистими даними користувачів;

— створення та управління курсами;

— збір статистичних даних про залученість студентів до курсів;

— синхронні та асинхронні способи комунікації;

— підтримка кооперативного навчання;

— оцінка вкладу студентів у командну роботу;

— відстеження всієї діяльності студентів, їх прогресу та співпраці з іншими студентами;

— збереження питань та розбиття їх на групи в залежності від ключових слів;

— інструменти для розробки та управління тестами;

— відстеження активності студента під час занять;

— тощо.

Дана система була розроблена за допомогою використання слабкозв'язної шарової архітектури мультиагентних систем, де нижній шар використовує сокети для комунікації в різноманітних ситуаціях, а другий шар системних протоколів комунікує з верхнім шаром. Для зберігання даних в I-MINDS використані реляційна база даних MySQL, а для зберігання аудіо та відео — Macromedia Flash Communication Server.

Інтелектуальні агенти, які забезпечують особливий функціонал I-MINDS, розташовані на другому найвищому шарі та складаються з об'єднання залежних та незалежних від вмісту курсів модулів. Незалежні від вмісту курсів агенти призначені для аналізу та управління навчальним процесом в загальному та абстрактному плані. Натомість залежні від вмісту курсів агенти займаються управління та аналізом інформації, пов'язаної з курсами інформації.

I-MINDS окрім вище наведених компонент складається ще з наступних інтелектуальних компонент:

— **Агент Викладач** — для представлення інтерфейсу користувача типу викладач, комунікації та кооперації з агентами студентів.

— **Агент Студент** — для представлення інтерфейсу користувача типу студент, комунікації та кооперації з агентами інших студентів та викладачів, відслідковування всіх активностей та прогресу навчання студента та взаємодії з іншими користувачами системи.
— **Агент Група** — для підтримки коопераивного навчання, визначення вкладу кожного студента у командну роботу.
— **Модуль розподілу питань** — для зберігання та групування питань згідно ключових слів.
— **Модуль профілювання студентів** — для відстеження активності студента під час занять.
— **Кооперативне навчання студентів** — для формування кожної групи студентів.
— тощо.

I-MINDS є прикладом системи повністю побудованої за допомогою використання агентів, так навіть для користувачів системи розроблені відповідні агенти, які збирають інформацію про користувачів, їх дії та успіхи з метою побудови навчальної моделі даного користувача системи, що в подальшому використовується для покращення якості навчального процесу.

РОЗДІЛ 4
МУЛЬТИАГЕНТНА ІНТЕЛЕКТУАЛЬНА СИСТЕМА УПРАВЛІННЯ НАВЧАЛЬНИМ ПРОЦЕСОМ

Дана мультиагентна інтелектуальна система розроблена за допомогою використання мультиагентного підходу, а саме інтеграції агентів у класичну систему управління навчальним процесом. Також при проектуванні та реалізації даної системи використовувались кореляційний та регресійний типи аналізу даних. Отже, проектування та побудова даної системи складалася з таких частин, як розробка вимог та реалізація класичної системи управління навчальним процесом, розробка вимог та визначення типу агентів та типу мультиагентної архітектури, побудова алгоритмів функціонування деяких агентів за допомогою використання відповідних видів аналізу даних та реалізація агентів разом з відповідними алгоритмами.

4.1. Розробка вимог

Наведена система є інтелектуальною системою управління навчальним процесом, а тому має задовольняти мінімальним функціональним та нефункціональним вимогам класичних систем управління навчальним процесом. Такі функціональні вимоги наведені в 1 розділі даної роботи, а також в [5–7], і включають наступні функції:

— управління особистими даними користувачів;
— управління курсами;
— управління навчальним процесом;
— побудова примітивних звітів;
— надання доступу до навчальних матеріалів;
— синхронні та/або асинхронні способи комунікації;

— тощо.

Дана система підтримує три типи користувачів: студент, викладач та працівник деканату. Кожен з наведених типів користувачів має свої права доступу до частин та функцій системи. Так, права доступу студента обмежені доступом до навчальних матеріалів, переглядом їх особистих даних та статистики на основі цих даних, використаннями синхронних та/або асинхронних способів комунікації тощо. Права доступу викладача ширші ніж у студента. Так, крім раніше перелічених функцій, викладач має доступ до наступного функціоналу: управління курсами, перегляд особистої інформації студентів, виставлення оцінок та відміток про відвідуваність занять студентом в межах курсу, побудова звітів про курс тощо. Найбільш повні права доступу має користувач системи типу працівник деканату, а саме він/вона має доступ до такого функціоналу як: перегляд особистої інформації студентів та викладачів, прив'язування викладачів та студентів до курсів, генерація різноманітних звітів, доступ до даних груп, потоків та підгруп.

Оскільки існує велика кількість як систем управління навчальним процесом, так і інтелектуальних систем управління навчальним процесом, то існує і чимала кількість можливого для реалізації функціоналу. Також провідні позиції на ринку систем управління навчальним процесом займають системи, які були розроблені великими корпораціями впродовж багатьох років та накопичили вже достатній об'єм функціональних потужностей. Тому в рамках даної роботи були проаналізовані потреби користувачів у новому функціоналі за допомогою використання опитувань потенційних користувачів системи та аналізу статистичних звітів про задоволеність студентів та викладачів від використання систем даного типу та про їх побажання в покращенні та оновленні функціоналу подібних систем, зокрема [2, 3, 9, 90]. Так, наприклад даний функціонал включає:

— створення та управління тестами;
— перегляд студентами відомостей про їх успішність та відвідуваність в режимі реального часу;
— підтримка синхронної та/або асинхронної комунікації між студентами;

— підтримка синхронної та/або асинхронної комунікації між викладачами;
— надсилання виконаних завдань студентами;
— створення та управління тестами;
— можливість проходження тестів студентами;
— можливість викладачів приймати участь в роботі студента;
— повторне надсилання переробленого завдання за дозволу викладача;
— зручний інтерфейс для побудови та управління розкладом;
— можливість відкривання документів в браузері без скачування;
— інструменти для швидкого отримання інформації про викладачів та студентів за їх прізвищем, ім'ям та ім'ям по батькові;
— тощо.

Серед нефункціональних вимог до даної мультиагентної інтелектуальної системи управління навчальним процесом можна виділити наступні:

— крос-платформенінсть системи;
— підтримку мобільної версії сайту;
— ергономічність інтерфейсу користувача;
— підтримку .pdf файлів та файлів створених за використання Microsoft Office;
— відповідність можливостей користувачів їх правам доступу;
— реалізація інтерфейсу користувача в пастельних тонах;
— безпека особистих даних користувачів;
— тощо.

Для надання інтелектуального, проактивного, реактивного та автономного функціоналу у наведеній системі використовуються агенти, а також відповідні алгоритми аналізу даних. В даній роботі інтелектуальні реактивні агенти в межах реактивної архітектури мультиагентних систем надають наступний функціонал:

— відстеження відвідуваності студентів та генерація повідомлень та звітів про екстраординарні ситуації;
— відстеження успішності студентів та генерація повідомлень та звітів про екстраординарні ситуації;

— автоматична побудова списків на перескладання та автоматичне заповнення відомостей для студентів, які потрапили на перескладання;
— автоматичне сповіщення викладачів та студентів про зміни у розкладі;
— відстеження залежності між успішністю та відвідуваністю студента;
— відстеження залежності між тривалістю модулів та успішністю студентів;
— прогнозування ймовірності здачі студентом курсу в залежності від раніше зібраних аналітичних даних та повідомлення викладача про студентів, які можуть не скласти курс;
— моніторинг середнього рівня знань групи, зокрема в залежності від відвідуваності або тривалості модуля, та повідомлення про його зміну викладача;
— повідомлення студенту про відмінність рівня успішності, зокрема в залежності від відвідуваності та тривалості модулів, від середнього рівні по групі;
— тощо.

Основою роботи агентів даної системи є побудова навчальної моделі студента з метою надання рекомендацій студентам для навчання та покращення рівня навального процесу в подальшому.

4.2. Розробка архітектури мультиагентної інтелектуальної системи управління навчальним процесом

Оскільки наведена система як і решта класичних та інтелектуальних систем управління навчальним процесом надає широкий спектр функціоналу, то її архітектура є комплексною та складною. Тому, умовно її розділимо на такі складові як модуль користувачів, модуль занять та курсів, модуль розкладу, модуль звітів, модуль комунікації та відповідні інтелектуальні реактивні агенти, які взаємодіють один з одним шляхом використання протоколу комунікації агентів FIPA.

4.2.1. Модуль користувачів. Даний модуль поєднує в собі типи користувачів системи, їх ролі та права доступу, функціонал для роботи з особистими даними користувачів. Так, у кожного користувача наявний власний кабінет, де зберігається та може бути модифікована його особиста інформація. Така інформація може включати ім'я, ім'я по батькові, прізвище, номер телефону, кафедру, пошту, номер групи для студентів, перелік курсів викладача, перелік курсів, які проходить студент тощо. В залежності від ролі, особиста інформація може варіюватись. Структура даного модуля та типів користувачів наведена на рисунку 4.1.

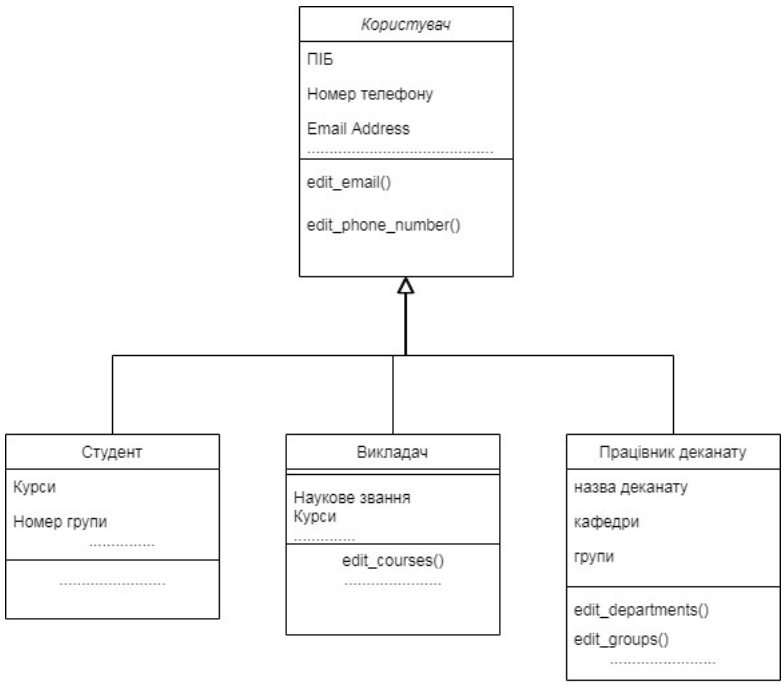

Рисунок 4.1. Основні структурні компоненти модуля користувачів.

Права доступу користувачів обмежують функціонал, який вони можуть використовувати та наведений на рисунку 4.1 в якості доступних для користувача функцій. Деяка особиста інформація не може бути змінена користувачами відповідного типу. Так, викладачі не можуть змінювати кафедру, а студенти не можуть змінювати номер групи. Дані зміни можуть вносити в систему лише працівник деканату та викладач відповідно. Також жоден з користувачів не може редагувати власні ім'я, ім'я по батькові та прізвище без відома користувача з правами доступу працівника деканату. Студенти формують навчальні групи,

викладачі працюють на певній кафедрі та працівники деканату, як і решта користувачів системи працюють в межах певного деканату, що відображено в ієрархічній структурі яка зображена на рисунку 4.2.

Рисунок 4.2. Основні структурні компоненти модуля користувачів.

Для уникнення потрапляння до системи випадкових користувачів було розроблено алгоритм реєстрації користувачів в системі в залежності від ролі:

1. реєструється адміністратор;
2. представник деканату надсилає документ про працівників деканату (ПІБ, номер телефону та пошти) адміністратору;
3. адміністратор створює сутність деканату в системі та реєструє працівників деканату;
4. представник кафедри надсилає працівнику деканату документ про викладачів кафедри (ПІБ, номер телефону та пошти);
5. працівник деканату створює сутність кафедри в системі та реєструє викладачів в системі;
6. староста групи надсилає документ про групу (ПІБ, номер телефону та пошти) викладачу-куратору групи;
7. викладач-куратор групи створює в системі сутність та реєструє студентів в системі.

Реєстрація групи осіб з використанням документу шаблону відбувається автоматично після натискання відповідних команд та завантаження файлу в систему. Наведений алгоритм хоч і зменшує ймовірність потрапляння до системи

стороннього користувача, проте не гарантує виконання даного факту. Так, само як і реєстрація за номером студентського квитка або ж фото користувача.

4.2.2. Модуль занять та курсів. Даний модуль системи є основною компонентою даної системи, оскільки в його межах відбувається управління курсами, навчальними матеріалами та приналежністю викладачів та студентів до курсів. В межах модуля занять та курсів реалізовані такі складові, як курси, модулі, заняття, тести, навчальні матеріали, завдання тощо, взаємозв'язки між якими зображені на рисунку 4.3.

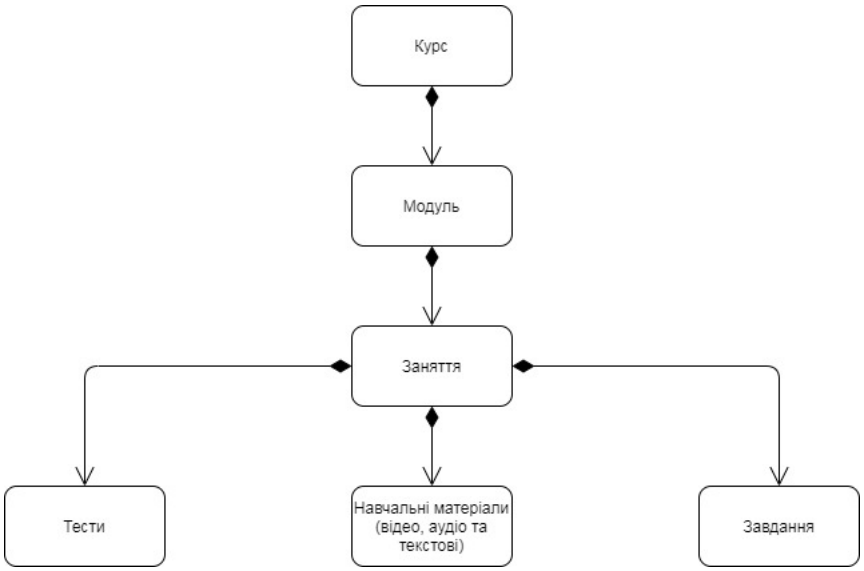

Рисунок 4.3. Основні структурні компоненти модуля занять та курсів.

Так, курс складається з серії занять, які характеризуються їх номером, темою, за потреби посиланням на заняття та датою його проведення. В межах заняття викладач може викласти відео, аудіо та текстові версії навчальних матеріалів, додати тести або завдання.

Тести характеризуються їх назвою та переліком питань. Вони можуть бути завантажені за допомогою використання документа або ж введення питань та можливих відповідей до тестів. Викладач може обрати автоматичну перевірку результатів тестів, після чого він має зазначити правильні відповіді. Також викладач може генерувати білети для тестів автоматично з вже наявних в системі питань шляхом вибору кількості питань в білеті. Студенти проходять тестування

та надсилають свої варіанти відповідей впродовж відведеного викладачем часу. Якщо студент не встиг надіслати свої відповіді до закінчення даного інтервалу часу, то система автоматично робить це за нього.

Для перевірки рівня засвоєння навчальних матеріалів студентами викладач може розміщувати аудіо, відео та текстові завдання. Студенти після виконання даних завдань надсилають їх викладачу. За потреби викладач може прокоментувати рішення та надати студенту можливість повторно надіслати завдання. Перед повторним надсиланням рішення завдання викладач має виставити попередній бал за дане завдання відповідному студенту. Тести, завдання та навчальні матеріали усіх занять курсу доступні до використання в межах усього курсу. В рамках даного модуля також можливе розбиття групи студентів на підгрупи як явним вибором студентів для підгруп, так і автоматичним поділом згідно кількості студентів в підгрупах та списку студентів або ж за допомогою кластеризації в залежності від рівня знань студентів, який визначається навчальною моделлю студента.

Викладач може вносити в систему відомості про відвідуваність занять студентами за допомогою використання журналу відвідувань. Також за виконанні завдання та тести викладач може виставляти оцінки студентам у відповідному журналі. При розбиті курсу на модулі викладачу необхідно зазначити розбаловку згідно даних модулів, яку в подальшому він може редагувати. Ця функція необхідна для функціонування агента, який перевіряє, чи може студент набрати мінімальну кількість балів необхідних для отримання допуску до екзамену чи заліку.

4.2.3. Модуль розкладу. Даний модуль системи активно взаємодіє з модулем занять та курсів з метою отримання інформації про назви курсів, періодичність їх занять, дату початку занять, дні тижня, в які відбувається заняття, тип занять тощо. На основі отриманих даних від модуля курсів та занять даний модуль генерує графічне представлення інформації про розклад для викладачів та студентів. Так, користувачі системи можуть переглянути розклад

в залежності від курсу, номеру групи та ПІБ викладача.

При спробах зміни розкладу система перевіряє, чи відсутні такі ситуації, як викладання одним викладачем одночасно декількох занять, зайнятість приміщення під декілька занять одночасно, декілька занять одночасно в одній групі. При виникненні однієї з даних ситуацій система не вносить такі зміни в базу даних та повідомляє про дану ситуацію відповідного користувача системи. Вносити зміни в розклад можуть викладачі та працівники деканату. Так, викладач може переносити заняття, призначати додаткове заняття тощо. В залежності від обраної дії у описі курсу змінюються кількість проведених занять та за потреби кількість занять, які залишилося провести. Працівники деканату також можуть переносити та призначати додаткові пари, а ще вони можуть відміняти пари та унеможливлювати перенесення та відміну пар викладачем в межах курсу.

Також даний модуль надає інформацію для агента, який повідомляє студентів та викладачів про зміни в розкладі.

4.2.4. Модуль звітів. Даний модуль системи тісно взаємодіє з рештою модулів системи для отримання інформації необхідної для генерації звітів. Різні типи користувачів мають доступ до автоматичної генерації різних типів звітів. Так, викладачі можуть генерувати звіти про успішність або відвідуваність їх курсів, звіти про результати тестів, кінцевий звіт з оцінками студентів за курс.

Також даний модуль збирає та зберігає в базі даних інформацію необхідні для роботи агентів, які будують складні звіти та проводять аналіз залежностей успішності від поведінки користувачів системи.

4.2.5. Модуль комунікації. Даний модуль забезпечує асинхронні та синхронні способи комунікації між студентами в межах групи, викладачами в межах кафедри та між студентами та викладачами в межах курсу.

4.2.6. Агент моніторингу відвідуваності. Даний агент відслідковує відвідуваність студента в межах певного курсу та автоматично створюється

в межах системи для кожної пари студент-курс. В подальшому як і в наведеному випадку вхідними даними агента будуть ідентифікатори відповідних параметрів, для яких він створюється. Викладач або працівник деканату задає максимальну кількість занять, які можна пропустити підряд та/або в межах всього курсу. Якщо студент перевищив кількість пропусків підряд, то відповідний звіт надсилається викладачу курсу та працівнику деканату. Якщо ж студент перевищив загальну кількість можливих для пропуску в межах курсу пар, то окрім надсилання звіту відповідним особам, даний агент інформує агента, який автоматично будує списки на перескладання, і той вносить студента до списків на перескладання.

Робота агента для моніторингу відвідуваності може бути проілюстрована за допомогою наступного псевдокоду.

$days\_counter = 0$
$days\_counter\_all = 0$
while student and course exist **do**
  $a = get\_from\_db(student, course)$
  if a is was not present **then**
    $days\_counter+ = 1$
    $days\_counter\_all+ = 1$
    if $days\_counter == n$ **then**
      generate report
      send report
      if $days\_counter\_all == max\_days\_amount$ **then**
        inform earrangement agent
      end if
    end if
  else
    $days\_counter = 0$
  end if

end while

Інформацію про відвідування занять студентом агент отримує шляхом запитів до бази даних з використанням ідентифікатора студента та курсу. Життєдіяльність агента обмежується наявністю в системі відомостей про студента та курс.

4.2.7. Агент моніторингу змін у розкладі. Даний агент повідомляє студентів та викладачів про зміни у розкладі. Даний агент створюється при першому запуску системи. Його функціонування полягає у перевірці файлі логування з метою знаходження записів про зміни у розкладі. При знаходженні відповідного типу запису агент робить запит до бази даних про пов'язаних з курсом студентів та викладачів та повідомляє їх про зміни у розкладі. Основну ідею роботи даного агента для моніторингу змін у розкладі може бути проілюстрована за допомогою наступного псевдокоду.

while system exists **do**
  if find changes in logs **then**
    $a = get\_from\_db(logs\_info)$
    send message to a.getStudent()
    send message to a.getLecturer()
  end if
end while

Життєдіяльність даного агента визначається часом існування мультиагентної інтелектуальної системи управління навчальним процесом. Для того, щоб даний агент не навантажував систему, він функціонує через певні інтервали часу.

4.2.8. Агент моніторингу настання дедлайнів. Даний агент інформує студентів та викладачів, а також агента, який відслідковує успішність студентів, про наближення закінчення терміну виконання завдання або ж наближення дати раніше запланованого тестування. Агент моніторингу

дедлайнів створюється при створенні курсу та його вхідними параметрами є ідентифікатор курсу та ідентифікатори групи, яка проходить курс. Робота даного агента може бути проілюстрована за допомогою наступного псевдокоду.

while group and course exists **do**
  if current_date is near course_id.get_from_db() **then**
    send notification to students
    send notification to lecturers
    inform performance agents
  end if
end while

Життєдіяльність даного агента обмежується існування в системі сутностей групи та курсу з відповідними ідентифікаторами.

4.2.9. Агент моніторингу успішності студента. Даний агент створюється для кожної пари студента-курсу і для меншого навантаження системи очікує інформаційного запиту від агенту, який відслідковує настання дедлайнів, і потім починає виконувати свою основну роботу: моніторинг успішності студента, який полягає у інформуванні агента для побудови списків на перескладання коли студент не зможе отримати фактично мінімальний бал для допуску до екзамену або заліку та інформування агентів, які проводять аналіз залежності успішності від інших факторів, про зміну рівня успішності студента, тобто відсоткового співвідношення отриманих до максимально можливих для отримання балів. Роботу агента для моніторингу успішності студента можна проілюстравати за допомогою наступного псевдокоду.

$average\_permance = 0$
while course and student exists **do**
  wait for info request from deadline agent

if $average\_performance \neq count\_average\_performance(get\_from\_db())$ **then**

$average\_permance = count\_average\_performance(get\_from\_db())$

info other agents

end if

end while

Життєдіяльність даного агента обмежується існуванням в межах системи сутностей студента та курсу з відповідними ідентифікаторами.

4.2.10. Агент автоматичного формування списків студентів на перескладання. Даний агент формує список студентів, які з різних причин не можуть отримати позитивну оцінку з курсу та повинні проходити перескладання. Ці списки в подальшому використовуються при друкуванні відомостей, де автоматично прописано, що студент не склав даний залік або екзамен. Даний агент очікує на інформативні запити від агентів, які відслідковують успішність та відвідуваність студента, та потім робить відповідний запис до бази даних. Формат даного запису до бази даних має вигляд (ідентифікатор студента, ідентифікатор курсу, причини). Роботу агента для автоматичного формування списків студентів на перездачі можна проілюструвати за допомогою наступного псевдокоду.

while system exists **do**

if info message from performance agent **then**

save to db information about current student

end if

if info message from attendance agent **then**

save to db information about current student

end if

end while

Оскільки даний створюється при запуску системи, то його життєдіяльність визначається часом функціонування мультиагентної інтелектуальної системи

управління навчальним процесом.

4.2.11. Агент моніторингу та передбачення залежності між успішністю та відвідуваністю студента. Такий агент аналізує залежності між успішністю та відвідуваністю шляхом аналізу поточних характеристик студента з використанням моделі "Випадковий ліс" та створюється для кожної пари студент-курс. Модель будується з певною періодичністю на вибірці даних. Її точність залежить від розподіленості та наявності в даних для навчання викидів. Тому в процесі функціонування системи дані про відвідуваність та успішність зберігаються без згадування особистих даних та використовуються для навчання моделі. Щоб не допустити перенавчання моделі та того, що вона буде розпізнавати всі можливі сценарії за рахунок наявності їх в моделі, "Випадковий ліс" кожного разу використовує нові підмножини з наявних даних, а інформація в системі зберігається за два останні роки з метою підтримки актуальності наявних даних. Дані, на яких навчається модель, розділяються на підмножини для побудови та навчання "Дерев рішень", які в подальшому об'єднуються у "Випадковий ліс" і результати роботи даних дерев використовуються при прогнозуванні успішності в залежності від відвідуваності. Для покращення точності моделі виконується передобробка даних, а саме видалення викидів з вибірки для навчання моделі. Оскільки для коректного функціонування даного агента необхідна достатня кількість даних, то при першому запуску системи слід або ж завантажити в неї дані з минулої системи управління навчальним процесом, або ж завантажити для тренування моделі відповідні датасети. Для визначення точності отриманого та наданого студенту результату використовується значення коефіцієнту забруднення Джині, який характеризує ймовірність некоректності отриманого результату. Тобто, якщо позначити коефіцієнт забруднення Джині як $koef$, то ймовірність отримання коректного результату буде складати $(1 - koef) * 100\%$.

Також агент зберігає поточні результати роботи та інформує про зміну даного значення користувача та інших агентів, які займаються побудовою навчальної

моделі студента. Оскільки поведінка кожного студента є індивідуальною та на початковому етапі можуть бути відсутні дані відповідного користувача, то для демонстрації існування зв'язку та його міри використовується кореляційний коефіцієнт Пірсона. Про зміну коефіцієнта агент також інформує студента та відповідних агентів. Якщо з'являється ймовірність нескладання студентом курсу в подальшому, то агент також інформує і викладача даного курсу. Оскільки агенти даного типу створюються для кожної пари студент-курс, то життєдіяльність таких агентів визначається часом наявності відомостей про відповідного студента та курс у базі даних.

4.2.12. Агент моніторингу та передбачення залежності між успішністю та тривалістю модулів. Даний агент аналізує залежність між успішністю та тривалістю модулів. Для свого функціонування він, так само як і агент моніторингу та передбачення залежності між успішністю та відвідуваністю студента, використовує модель "Випадковий ліс" і коефіцієнт забруднення Джині для отримання ймовірності коректності результату. Аналогічним чином в роботі агента використовується коефіцієнт кореляції Пірсона для встановлення наявності та міри зв'язку між досліджуваними характеристиками.

Про будь-які зміни поточних показників агент інформує студента та відповідних агентів, які займаються побудовою навчальної моделі, а при прогнозування незадовільних значень повідомляє викладача. Оскільки агенти даного типу створюються для кожної пари студент-курс, то життєдіяльність таких агентів визначається часом наявності відомостей про відповідного студента та курс у базі даних.

4.2.13. Агент навчальної моделі студента в межах курсу. Даний агент акумулює дані, отримані від агентів моніторингу залежності успішності від відуваності та моніторингу залежності успішності від відвідуваності для побудови рекомендацій студента на основі його поведінки та вподобань. При будь-яких змінах агенти такого типу починають цикл комунікації з метою виявлення середнього рівня по групі та повідомлення викладачу про зміну середнього

групи рівня та студентів, які спричинили дану зміну. Студенти також будуть проінформавані, якщо їх рівень вищий або нижчий від середнього по групі більше ніж на 10%. Комунікація відбувається з ініціативи поточного агента шляхом використання запиту інформування всіх інших агентів даного типу для цієї групи та курсу: цей агент повідомляє решті про зміну своїх показників і про потребу повторного обчислення значення середнього по групі та порівняння поточних показників з оновленим значенням середнього. Тому після акту комунікації можливе інформування студентів іншими агентами, я а не саме поточним. Оскільки агенти даного типу створюються для кожної пари студент-курс, то життєдіяльність таких агентів визначається часом наявності відомостей про відповідного студента та курс у базі даних.

4.2.14. Агент навчальної моделі студента. Такий агент акумулює дані, отримані від агентів навчальної моделі студента в межах курсу та використовується при кластеризації для розбиття групи студентів на підгрупи. В подальшому планується розширення функціоналу даного агента шляхом додавання підтримки інших методів прогнозування та інших методів побудови рекомендацій на основі отриманої інформації та її аналізу. Оскільки агенти даного типу створюються для кожного студента, то життєдіяльність таких агентів визначається часом наявності відомостей про відповідного студента у базі даних.

4.2.15. Використані технології. Розроблена мультиагентна інтелектуальна система управління навчальним процесом є прикладом використання мультиагентного підходу, аналізу даних та машинного навчання в межах веб системи для вищих навчальних закладів. Розробка системи базувалася на використанні наступних технологій:

— Python 3.7 — динамічно типізована мова програмування;
— Pade — фреймворк для розробки агентів та мультиагентних систем на мові програмування Python;
— PostgreSQL — реляційна база даних;
— MongoDb — NoSQL база даних для збереження розріджених даних;

— пакет Logging; — пакет мови програмування Python для логування дій та подій у системі;
— Flask — фреймворк для розробки веб систем на мові програмування Python;
— Bootstrap — фреймворк для верстки;
— sklearn — бібліотека мови програмування.

В ході даного проекту були спроектовані та реалізовані такі структурні модулі як: модуль користувачів, модуль занять та курсів, модуль звітів, агент моніторингу відвідуваності, агент моніторингу змін у розкладі, агент моніторингу настання дедлайнів, агент моніторингу успішності студента, агент автоматичного формування списків на перескладання, агент моніторингу та передбачення залежності між відвідуваністю та успішністю, агент моніторингу та передбачення залежності між успішністю та тривалістю модулів, агент навчальної моделі студента в межах курсу, агент навчальної моделі студента.

Подальший розвиток системи полягає в наступному:

— додавання до функціоналу агентів навчальних моделей студентів нових типів аналітики та алгоритмів по побудові рекомендацій для покращення навчального процесу;
— додання надсилання персоналізованих рекомендацій студентам для покращення їх рівня знань;
— покращення роботи з навчальними матеріалами;
— надання можливості користувачам генерувати звіти відповідно до власного шаблону;
— побудова нових типів агентів, які будуть збирати та надавати аналітику агентам, які займаються побудовою навчальної моделі;
— забезпечення автоматичного заповнення журналу відвідуваності студентів на занятті, коли для його проведення використовуються онлайн застосунки;
— тощо.

ВИСНОВКИ

Метою цієї роботи є побудова мультиагентної інтелектуальної системи управління навчальним процесом. В межах роботи були проаналізовані наступні особливості даної предметної області:

— системи управління навчальним процесом (основні характеристики, класифікації та будова, приклади: Moodle, Adobe Captivate Prime, Docebo, Blackboard Learn);

— інтелектуальні системи управління навчальним процесом (основні характеристики, відмінності, класифікації та будова, приклади: ALLEGRO, MAS-PLANG, I-MINDS);

— класифікації агентів (в залежності від архітектури, внутрішньої структури, задач, які агенти виконують);

— типи агентів (реактивні, на основі моделі, які використовують перцепції, з внутрішніми станами, що базуються на цілях, корисні, які навчаються, які використовують практичні міркування, гібридні агенти);

— способи взаємодії агентів та їх характеристики (комунікація та кооперація);

— типи архітектури мультиагентних систем (на основі логік, реактивна, шарова та на основі переконань, бажань та намірів);

— протоколи комунікації агентів (KIF, KQML, FIPA, CNP);

— фреймворки для реалізації агентів та мультиагентних систем (Jade, Jack, Jason, Pade, Mesa, Spade).

В процесі проектування та розробки мультиагентної інтелектуальної системи управління навчальним процесом були виконані наступні кроки:

— розробка функціональних та нефункціональних вимог;

— вибір типу архітектури системи та агентів;

— проектування архітектури системи та агентів;

— вибір стеку технологій;

— розробка прототипу системи;

— тестування прототипу системи.

Під час виконання було підтверджено актуальність даного підходу до побудови інтелектуальних систем управління навчальним процесом.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. T. Orzechowski, "The use of multi-agent‘s systems in e-learning platforms," in *Siberian Conference on Control and Communications SIBCON-2007*, 2007, pp. 64–71.
2. *Trends and Future of Learning Management Systems (LMSs) in Higher Education*, 2017.
3. E. Dahlstrom, D. C. Brooks, and J. Bichsel, *The Current Ecosystem of Learning Management Systems in Higher Education: Student, Faculty, and IT Perspectives.* Educase Center for Analysis and Research, 2014.
4. D. Ifenthaler, *Implementation of Web-Enhanced Features for Successful Teaching and Learning. The Utility of Technological Progressions of Learning Management.*
5. W. R. Watson and S. L. Watson, "An argument for clarity: What are learning management systems, what are they not, and what should they become," *TechTrends*, no. 2, pp. 199–205, 2017.
6. M. S. A. Alshammari, "Academics‘ adoption and usage of learning management systems in saudi arabia‘s universities," Ph.D. dissertation, Computer Science and Informatics Centre for Computing and Social Responsibility Faculty of Technology De Montfort University, 2015.
7. "University of Minnesota Learning Management System Review," University of Minnesota, Technical Report, 2017.
8. "Lms market dynamics," Tech. Rep., 2017.
9. K. A. Al-Busaidi and H. Al-Shihi, "Key factors to instrictors‘ satisfaction of learning management systems in blended learning," *J Comput High Educ*, pp. 18–39, 2012.
10. T. Naz and M. Khan, "Functionality gaps in the design of learning management systems," *International Journal of Advanced Computer Science and Applications*, 2019.

11. C. Hodges and M. Grant, "Theories to support you: Purposeful use of learning management system features," 2015, pp. 481–486.
12. V. Stantchev, R. Colomo-Palacios, PedroSoto-Acosta, and S. Misra, "Learning management systems and cloud file hosting services: A study on students' acceptance," *Computers in Human Behavior*, 2013.
13. N. Sclater, "Web 2.0, personal learning environments, and the future of learning management systems," *EDUCAUSE*, p. 4, 2008.
14. S. D. Steven Lonn, "Saving time or innovating practice: Investigating perceptions and uses of learning management systems," *Computers & Education*, 2009.
15. S. Graf and K. Kinchuk, "Providing adaptive courses in learning management systems with respect to learning styles," 2007.
16. S. Foreman, *The LMS Guidebook - Learning Management Systems Demystified.* ATD Press, 2018.
17. D. Ulker and Y. Yilmaz, "Learning management systems and comparison of open-source learning management systems and proprietary learning management systems," 2016.
18. elarningindustry open-source LMSs top. [Online]. Available: https://elearningindustry.com/top-open-source-learning-management-systems
19. capterra open-source LMSs top. [Online]. Available: https://blog.capterra.com/top-8-freeopen-source-lmss/
20. paradissolutions open-source LMSs top. [Online]. Available: https://www.paradisosolutions.com/blog/top-open-source-elearning-solution-2020/
21. trustradius proprietary LMSs top. [Online]. Available: https://www.trustradius.com/learning-management-lms
22. getstream proprietary LMSs top. [Online]. Available: https://getstream.io/blog/best-learning-management-systems/
23. elearningindustry cloud-based LMSs top. [Online]. Available: https://elearningindustry.com/top-cloud-based-learning-management-systems-for-corporate-training

24. softwareadvice cloud-based LMSs top. [Online]. Available: https://www.softwareadvice.com/lms/cloud-comparison/
25. Moodle. [Online]. Available: https://moodle.com/
26. Moodle at app store. [Online]. Available: https://itunes.apple.com/ru/app/moodle-classic/id1403448117?mt=8
27. Moodle. [Online]. Available: https://play.google.com/store/apps/details?id=com.moodle.classic
28. Moodle at microsoft store. [Online]. Available: https://www.microsoft.com/en-us/p/moodle-desktop/9p9bwvhdc8c8?activetab=pivot:overviewtab
29. Adobe captivate. [Online]. Available: https://www.adobe.com/ua/products/captivateprime.html
30. Docebo. [Online]. Available: https://www.docebo.com/
31. elearningindustry top cloud-based learning management systems. [Online]. Available: https://elearningindustry.com/the-best-learning-management-systems-top-list
32. Docebo mobile version. [Online]. Available: https://www.docebo.com/mobile-learning-lms-app-elearning-platform/
33. Blackboard. [Online]. Available: https://www.blackboard.com/blackboard-learn/index.html
34. mobile version of the Blackboard Learn LMS. [Online]. Available: https://www.blackboard.com/teaching-learning/learning-management/mobile-learning-solutions
35. M. J. Wooldridge and N. R. Jennings, "Intelligent agents: Theory and pracice," *The Knowledge Engineering Review*, vol. 10:2, pp. 115–152, 1995.
36. M. Wooldridge, Ed., *An Introduction to MultiAgent Systems.* John Wiley & Sons Ltd., 2009.
37. L. Sterling and K. Taveter, Eds., *The Art of Agent-Oriented Modeling.* The MIT Press, 2009.
38. D. Weyns, Ed., *Architecture-Based Design of Multi-Agent Systems.* Springer, 2010.

39. educba agents types. [Online]. Available: https://www.educba.com/intelligent-agents/

40. *Computer Based Learning Unit.* University of Leeds, 1995.

41. M. J. Wooldridge, "Intelligent agents," in *Multiagent systems.A modern approach to Distributed Artificial Intelligence*, G. Weiss, Ed. Massachusetts institute of Technology, 1999.

42. J. S. Rosenschein and M. R. Genesereth, "Communication and coopeation," Tech. Rep., 1984.

43. J. H. Miller, "Communication and cooperation," *Journal of Economic Behavior & Organization*, vol. 47, 2002.

44. A. Haddadi, Ed., *Communication and Cooperation in Agent Systems.* Springer Link, 1995.

45. J. E. Doran, S. Franklin, N. R. Jennings, and T. J. Norman, "On cooperation in multi-agent systems," 2020.

46. N. Sclater, "Speech act types in conversations in the "new interchange" series," 2017.

47. M. Gonzalez-Lloret, "Conversation analysis and speech act performance," 2010.

48. J. Rocha, I. Boavida-Portugal, and E. Gomes, "Introductory chapter: Multi-agent systems," in *Multi-agent systems*, 2016.

49. M. Colombetti and M. Verdicchio, "An analysis of agent speech acts as institutional actions," 2002.

50. T. Finin, D. McKay, and R. Fritzson, *An Overview of KQML: A Knowledge Query and Manipulation Language.* KQML Advisory Group, 1992.

51. K. Subramaniam, "Agent communication languages and protocols," 2002.

52. *FIPA Communicative Act Library Specification*, 2002.

53. S. Kaur, H. Kaur, and S. K. Sehra, "Modification of contract net protocol(cnp) : A rule-updation approach," vol. 4, no. 11, 2013.

54. L. Padgham and M. Winikoff, *Developing Intelligent Agent Systems: A practical guide.* John Wiley & Sons Ltd., 2004.

55. G. Gaire, D. Greenwood, and F. Bellifemine, *Developing Multi-Agent Systems with JADE*. John Wiley & Sons Ltd., 2007.
56. S. Lang, *Web Development with Jade*, 1st ed. Packt Publishing Ltd., 2014.
57. F. Bellifmine, F. Bergenti, and G. Gaire, "Jade - a java agent development framework," in *Multi-Agent Programming Languages, PLatforms and Applications*, R. H. Bordini, M. Dastani, J. Dix, and A. El Fallah Seghrouchni, Eds. Springer Science+Business Media, Inc., 2005.
58. Jade. [Online]. Available: http://jade.tilab.com/
59. Jack github. [Online]. Available: https://github.com/uclnlp/jack
60. Jack. [Online]. Available: http://rockhopper.aosgrp.com/products/jack/
61. H. Bordini and F. Hubner, "Jason. A Java-based interpreter for an extended version of AgentSpeak," 2007.
62. Jason. [Online]. Available: http://jason.sourceforge.net/wp/
63. Pade. [Online]. Available: https://pade.readthedocs.io/en/latest/
64. Pade github. [Online]. Available: https://github.com/grei-ufc/pade
65. Mesa. [Online]. Available: https://mesa.readthedocs.io/en/stable/
66. Spade. [Online]. Available: https://pypi.org/project/spade/
67. Spade documentation. [Online]. Available: http://spade-mas.readthedocs.io/
68. Bdi agents with agentspeak. [Online]. Available: https://github.com/javipalanca/spade_bdi
69. bokeh plots for agents. [Online]. Available: https://github.com/javipalanca/spade_bokeh
70. A. Fardinpour, M. Pedram, and M.Burkle, "Intelligent learning management systems: Definition, features and measurement of intelligence," 2014.
71. R. Maia and M. Netto, ""work in progress - a distributed approach to a learning management system using multi-agent technology," 2005.
72. R. J. Oskouei and N. M. Kor, "Proposing a novel adaptive learning management system: an application of behavior mining & intelligent agents," *Intelligent Atumation & Soft Computing*, pp. 199–205, 2017.
73. A. Aftab, M. Aslam, A. M. Martinez-Enriquez, Z. ul Quayyum, and A. Z. Syed,

"Agent based intelligent learning management system for heterogeneous learning eviroment," in *IEEE 14th International Multitopic Conference*, 2011, pp. 76–81.

74. E. Sklar and D. Richards, "Agent-based systems for human learners," *The knowlrdge Engineering Review*, vol. 25:2, pp. 111–135, 2010.
75. Velleman, P. F. Hoaglin, and D. C., *Applications, Basics, and Computing of Exploratory Data Analysis.* Duxbury Press, 1981.
76. J. T. Behrens, "Principles and procedures of exploratory data analysis." *Journal TOC*, 1997.
77. T. Purwoningsih, H. B. Santoso, and Z. A. Hasibuan, "Online learners' behaviors detection using exploratory data analysis and machine learning approach," 2019.
78. N. G. Das, *Statistical methods.* Teta McGraw-Hill, 2009.
79. S. Chatterjee and A. S. Hadi, *Regression analysis by Example.* Wiley, 2012.
80. S. Hussain, S. Gaftandzhieva, M. Maniruzzaman, R. Doneva, and Z. F. Muhsin, "Regression analysis of student academic performance using deep learning," *Spring Link*, 2020.
81. P. Giudici, S. Ingrassia, and M. Vichi, Eds., *Statisticals Models for Data Analysis.* Springer, 2013.
82. E. O. Ugwoke, N. I. Edeh, and J. C. Ezemma, "Effect of flipped classroom on learningmanagement systems and face-to-face learningenvironments on students' gender, interest andachievement in accounting," 2018.
83. C. J. Huberty, "Discriminant analysis," *Sage journals*, vol. 45, 1975.
84. P. Xanthopoulos, P. M. Pardalos, and T. B. Trafalis, *Linear Discriminant Analysis*, 2012.
85. L. Kaufman and P. J. Rousseeuw, *Finding Groups in Data. Introduction to Cluster Analysis.* Wiley, 2005.
86. S. Valsamidis, S. Kontogiannis, I. Kazanidis, T. Theodosiou, and A. Karakos, "A clustering methodology of web log data for learning management systems," *Jstor*, 2010.
87. Rosa Maria Vicari and Demetrio Arturo Ovalle and Jovani Alberto Jim̃nez-Builes, "Allegro: Teaching/learning multi-agent environment using instructional planning

and cases- based reasoning (cbr)," *CLEI Electronic Journal*, vol. 10, no. 1, June 2007.

88. C. Pena, J. Marzo, and J. Rosa, *Intelligent Agents in a Teaching and Learning Environment on the Web.*
89. L. Soh, N. Khandaker, X. Liu, and H. Jiang, *A Computer-Supported Cooperative Learning System with Multiagent Intelligence.*
90. T. Orzechowski, "The use of multi-agent's systems in e-learning platforms," in *E-learning Experiences and Future*, S. Soomro, Ed. InTech, 2010.