Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)

Кафедра Штучного інтелекту
(повна назва)

# КВАЛІФІКАЦІЙНА РОБОТА
# Пояснювальна записка

рівень вищої освіти другий (магістерський)

Створення експертних систем за допомогою сервіс-орієнтованого підходу
(тема)

Виконав:
студент 2 курсу, групи СШІм-20-1
Кварацхелія Т.В.
(прізвище, ініціали)

Спеціальність 122 Комп'ютерні науки
(код і повна назва спеціальності)

Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Освітня програма Системи штучного інтелекту
(повна назва спеціалізації)

Керівник доц. Золотухін О.В.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ В.О. Філатов
(підпис) (прізвище, ініціали)

2021 р.

Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)

Кафедра Штучного інтелекту
(повна назва)

Рівень вищої освіти другий (магістерський)

Спеціальність 122 Комп'ютерні науки
(код і повна назва)

Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Освітня програма Системи штучного інтелекту (СШІ)
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«_____» _____ 20 ___ р.

# ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Кварацхелії Тимуру Вахтанговичу
(прізвище, ім'я, по батькові)

1. Тема роботи Створення експертних систем за допомогою сервіс-орієнтованого підходу

затверджена наказом університету від 8 листопада 20 21 р. № 1695Ст
2. Термін подання студентом роботи до екзаменаційної комісії _____ _____ 20 21 р.
3. Вихідні дані до роботи Для перевірки працездатності сервісу використовувалася експертна система діагностики захворювань; використовувалися документації щодо роботи з платформою Google App Engine та роботи з JESS для продукційних систем. Бібліотеки JESS використовують розширену версію алгоритму Rete для роботи з правилами виведення.

4. Перелік питань, що потрібно опрацювати в роботі
1) Аналіз предметної області і постановка задачі
2) Структура та етапи створення ЕС
3) Запропонований сервіс створення ЕС
4) Практичне використання результатів дослідження

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри)_____
_____
_____
_____
_____
_____
_____
_____
_____
_____

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1 )

| Найменування розділу | Консультант (посада, прізвище, ім'я, по батькові) | Позначка консультанта про виконання розділу | |
|---|---|---|---|
| | | підпис | дата |
| Основна частина | к.т.н., доцент Золотухін О.В. | | |
| | | | |

**КАЛЕНДАРНИЙ ПЛАН**

| № | Назва етапів роботи | Терміни виконання етапів роботи | Примітка |
|---|---|---|---|
| 1. | Отримання завдання на кваліфікаційну роботу | 01.09.2021 | виконано |
| 2. | Аналіз завдання та пошук літератури за темою | 02.09.2021–08.09.2021 | виконано |
| 3. | Опрацювання літератури та аналіз об'єкту | 09.09.2021–15.09.2021 | виконано |
| 4. | Вибір програмних засобів для розробки системи | 16.09.2021–18.09.2021 | виконано |
| 5. | Розробка програмного засобу | 19.09.2021–05.11.2021 | виконано |
| 6. | Аналіз отриманих результатів | 06.11.2021–16.11.2021 | виконано |
| 7. | Оформлювання пояснювальної записки | 17.11.2021–24.11.2021 | виконано |
| 8. | Проходження нормоконтролю | 25.11.2021–28.11.2021 | виконано |
| 9. | Оформлення презентаційних матеріалів | 29.11.2021–01.12.2021 | виконано |
| 10. | Попередній захист | 02.12.2021 | виконано |
| 11. | Представлення кваліфікаційної роботи | | |
| | | | |

Дата видачі завдання 1 вересня 2021 р.

Студент _____
(підпис)

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

# РЕФЕРАТ

Пояснювальна записка: 63 с., 15 рис., 2 дод., 22 джерела.

ЕКСПЕРТНА СИСТЕМА, СЕРВІС, CLOUD COMPUTING, IAAS, PAAS, SAAS.

Об'єкт дослідження: провайдери хмарних технологій, що мають платформу для розробки сервісів. Експертні системи та механізм для виведення правил JESS.

Мета кваліфікаційної роботи: реалізувати сервіс для створення та використання експертних систем за допомогою хмарних технологій, та розмістити його в інфраструктуру одного з хмарних провайдерів.

Методи дослідження: вивчення документації та літературних джерел з галузі хмарних технологій та експертних систем. Узагальнення отриманих відомостей. Теоретичний аналіз та синтез.

Результати: сервіс, реалізований за допомогою передових технологій, що дозволяє створювати та використовувати експертні системи у хмарній інфраструктурі.

# РЕФЕРАТ

Пояснительная записка: 63 с., 15 рис., 2 прил., 22 источника.

СЕРВИС, ЭКСПЕРТНАЯ СИСТЕМА, CLOUD COMPUTING, IAAS, PAAS, SAAS.

Объект исследования: провайдеры облачных технологий, предоставляющих платформу разработки сервисов. Экспертные системы и механизм вывода правил JESS.

Цель квалификационной работы: реализовать сервис для создания и использования экспертных систем с помощью облачных технологий и разместить его в инфраструктуру одного из облачных провайдеров.

Методы исследования: изучение документации и литературных источников в области облачных технологий и экспертных систем. Обобщение полученых сведений. Теоретический анализ и синтез.

Результаты: сервис, реализуемый с помощью передовых технологий, позволяющий создавать и использовать экспертные системы в облачной инфраструктуре.

## ABSTRACT

Explanatory note: 63 p., 15 fig., 2 ann., 22 sources.

CLOUD COMPUTING, EXPERT SYSTEM, IAAS, PAAS, SAAS, SERVICE.

Object of research: providers of cloud technologies that provide a platform for the development of services. Expert systems and JESS rule engine.

Methods: studying of documentation and literature in field of cloud computing and expert systems. A generalization of the findings. Theoretical analysis and synthesis.

Results and novelty: the implemented service based on advanced technologies, allowing to create and use expert systems in the cloud infrastructure.

ЗМІСТ

# ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ ТА ТЕРМІНІВ

БД – База даних;

БЗ – База знань;

ВІД – Хмарні технології;

ЕС – Експертна система;

ІТ – Інформаційні технології;

ПЗ – Програмне забезпечення;

GAE – Google App Engine;

CC – Cloud Computing – хмарні обчислення;

IaaS – Infrastructure as a service – інфраструктура як сервіс;

PaaS – Platform as a service – платформа як сервіс;

SaaS – Software as a service – програмне забезпечення як сервіс.

# ВСТУП

Проблема розробки сервісу, який допомагає вирішувати завдання, важкі для експерта-людини, і який отримує результати, що не поступаються якістю та ефективності рішень, які отримують експерт, є актуальною. Такий сервіс повинен мати режим для введення знань експерта в базу знань та режим консультації користувачів. Обсяг бази знань і кількість користувачів сервісу заздалегідь невідомий, як і кількість необхідних обчислювальних ресурсів до роботи даного сервісу. Таку проблему можна вирішити, реалізувавши сервіс із використанням хмарних обчислень (cloud computing).

Сьогодні технологія cloud computing (CC) – одна з обговорюваних тем в IT-середовищі. Нове рішення, що пропонує віртуалізувати ряд бізнес-процесів, є привабливим для клієнтів, але, як і все нове, викликає велику кількість дискусій про переваги, вигоди та практичну реалізацію хмарних послуг. Концепція CC полягає у наданні кінцевим користувачам віддаленого доступу до динамічно масштабованих обчислювальних ресурсів, послуг та додатків (включаючи операційні системи та інфраструктуру), підтримку та оновлення яких здійснює провайдер послуг.

Фактично, технологія CC дозволяє компаніям отримати доступ до бізнес-додатків за допомогою інтернету, що особливо актуально для невеликих організацій, обмежених в IT-ресурсах, стартапів і великих корпоративних замовників, зацікавлених у гнучких рішеннях, що легко масштабуються.

Технології CC дозволяють створювати послуги за допомогою спеціальних середовищ розробки. На сьогоднішній день існує безліч хмарних провайдерів, які надають дані платформи.

З використанням хмарних обчислень стало можливо відокремити апаратне забезпечення, операційні системи та програмне забезпечення, які користувач звик представляти як єдине ціле у себе на стаціонарному комп'ютері або на іншому пристрої. Іншими словами, маючи пристрій здатний

відкривати браузер або клієнт, що взаємодіє з хмарним сервісом, пристрій вводу/виводу та вихід в інтернет, можна працювати з різними додатками та виділити необхідний обсяг обчислювальних ресурсів та пам'яті, що надаються спеціальними хмарними провайдерами.

На сьогоднішній день існує безліч таких провайдерів. Це Amazon, Microsoft, Google та багато інших. Деякі з провайдерів також пропонують послуги з надання середовища для розробників та розміщення розроблених сервісів у їхній інфраструктурі. Таким чином, всі складнощі, пов'язані з апаратним забезпеченням, зберіганням баз даних та ін, інкапсульовані хмарним провайдером. Абстрагуючись від таких складнощів, можна скоротити витрати в бізнесі, прискорити розробку додатків, підвищити рівень безпеки, якість обслуговування та «еластичність» сервісів до необхідного рівня.

Метою даної роботи є створення сервісу, побудованого за допомогою технологій CC, що дозволяє створювати і використовувати експертні системи. Експертна система – програма, що моделює поведінку експерта, компетентного у певній галузі. Як користувачів сервісу передбачається розглядати як звичайних користувачів, так і експертів. Як машина логічного висновку правил розглядається оболонка експертних систем Java Expert System Shell (JESS). JESS дозволяє вводити Java-об'єкти в робочу пам'ять і робити висновок за допомогою правил, які визначені в системі.

Інтерфейс користувача являє собою режим консультації, в якому система отримує відомості про поточне завдання та дає по ній рекомендації. Пошук експертних систем здійснюється за ключовими словами, введеними користувачем, які описують проблему користувача. Інтерфейс експерта є режим введення знань. У цьому режимі експерт вибирає область, де спеціалізується і дає рекомендації. Складені правила заносяться в основу знань. За допомогою оцінок користувача, можна буде визначити достовірність складеної експертом системи і, при необхідності, внести до неї зміни або доповнення.

# 1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАВДАННЯ

## 1.1 Ключові фактори розвитку хмарних технологій

Вперше ідея того, що сьогодні називають СС, була озвучена JCR Licklider, в 1970 році. В ті роки він був відповідальним за створення ARPANET (Advanced Research Projects Agency Network). Його ідея полягала в тому, що кожна людина на землі буде підключена до мережі, з якої вона отримуватиме не тільки дані на програми та програми. Інший вчений John McCarthy висловив ідею про те, що обчислювальні потужності надаватимуться користувачам як послуга (сервіс). На цьому розвиток хмарних технологій (ВІД) було припинено до 90-х років, після чого її розвитку сприяла низка факторів.

Розширення пропускної спроможності Інтернету, в 90-ті роки не дозволило отримати значного стрибка у розвитку обчислювальної техніки, так як практично жодна компанія не технології на той час не були готові до цього. Проте сам факт прискорення Інтернету дав поштовх якнайшвидшому розвитку СС.

Однією з найважливіших подій у цій галузі було поява Salesforce.com в 1999 року. Дана компанія стала першою компанією, що надала доступ до свого додатку через сайт, по суті, дана компанія стала першою компанією, що надала своє програмне забезпечення за принципом – програмне забезпечення як сервіс (SaaS).

Наступним кроком стала розробка хмарного веб-сервісу Amazon у 2002 році. Даний сервіс дозволяв зберігати, інформацію та проводити обчислення.У 2006, Amazon запустила сервіс під назвою Elastic Compute cloud (EC2), як веб-сервіс, який дозволяв його користувачам запускати свої власні програми. Сервіси Amazon EC2 та Amazon S3 стали першими доступними сервісами СС. Інша віха в розвиток СС відбулася після створення Google, платформи Google Apps для веб-додатків у бізнес секторі. Значну роль у розвитку ВІД відіграли

технології віртуалізації, зокрема програмне забезпечення, що дозволяє створювати віртуальну інфраструктуру. Розвиток апаратного забезпечення сприяло не так швидкому зростанню ВІД, скільки доступності даної технології для малого бізнесу та індивідуальних осіб. Щодо технічного прогресу, то значну роль у цьому відіграло створення багатоядерних процесорів та збільшення ємності накопичувачів інформації.

## 1.2 Cloud Computing в теперішній час

Вікіпедія дає таке визначення Cloud Computing – технологія розподіленої обробки даних, в якій комп'ютерні ресурси та потужності надаються користувачеві як Інтернет-сервіс. Надання користувачеві послуг як Інтернет-сервіс є ключовим. Однак під Інтернет-сервісом не варто розуміти доступ до сервісу лише через Інтернет, він може здійснюватися також через звичайну локальну мережу з використанням веб-технологій.

З визначення та історії видно, що основою для створення та швидкого розвитку хмарних обчислювальних систем послужили великі інтернет сервіси, такі як Google, Amazon та ін, а так само технічний прогрес, що по суті говорить про те, що поява СС була лише справою часу. Розвиток перерахованих вище напрямків дозволив хмарним системам стати доступнішими. Розвиток багатоядерних процесорів призвело до:

– збільшення продуктивності, при тих же розмірах обладнання;

– зниження вартості устаткування, як наслідок експлуатаційних витрат;

– зниження енергоспоживання хмарної системи, для більшості ЦОД це справді проблема при нарощуванні потужностей ЦОД.

Збільшення ємностей носіїв інформації, зниження вартості зберігання 1 Мб дозволило:

– безмежно (принаймні так позиціонує себе більшість «хмар») збільшити обсяги інформації, що зберігається;

– знизити вартість обслуговування сховищ інформації, значно збільшивши обсяги даних, що зберігаються.

Розвиток технології багатопотокового програмування призвело до:

– ефективне використання обчислювальних ресурсів багатопроцесорних систем;

– гнучкий розподіл обчислювальних потужностей хмар.

Розвиток технологій віртуалізації призвів до:

– створення програмного забезпечення, що дозволяє створювати віртуальну інфраструктуру незалежно від кількості наданих апаратних ресурсів;

– легкість масштабування, нарощування систем;

– зменшення витрат на адміністрування хмарних систем;

– доступність віртуальної інфраструктури через мережу Інтернет.

Збільшення пропускної спроможності призвело до:

– збільшення швидкості роботи з хмарними системами, зокрема віртуальний графічний інтерфейс та робота з віртуальними носіями інформації;

– зниження вартості Інтернет-трафіку для роботи з великими обсягами інформації;

– проникнення СС маси.

## 1.3 Призначення експертних систем

На початку вісімдесятих років у дослідженнях з штучного інтелекту сформувався самостійний напрямок, який отримав назву «експертні системи» (ЕС). Мета досліджень з ЕС полягає в розробці програм, які при вирішенні завдань, важких для експерта-людини, отримують результати, що не поступаються за якістю та ефективності рішенням, які отримують експерт.

Програмні засоби (ПЗ), що базуються на технології експертних систем, або інженерії знань, набули значного поширення у світі. Важливість експертних

систем полягає в наступному:

– технологія експертних систем істотно розширює коло практично значущих завдань, які вирішуються на комп'ютерах, вирішення яких дає значний економічний ефект;

– технологія ЕС є найважливішим засобом у вирішенні глобальних проблем традиційного програмування: тривалість і, отже, висока вартість розробки складних програм;

– висока вартість супроводу складних систем, яка часто в кілька разів перевищує вартість їхньої розробки; низький рівень повторної використання програм тощо;

– об'єднання технології ЕС із технологією традиційного програмування додає нові якості до програмних продуктів за рахунок: забезпечення динамічної модифікації додатків користувачем, а не програмістом; більшої «прозорості» додатка (наприклад, знання зберігаються на обмеженому ЕЯ, що не вимагає коментарів до знань, спрощує навчання та супровід); кращої графіки; інтерфейсу та взаємодії.

ЕС призначені для про неформалізованих завдань, тобто. ЕС не відкидають і замінюють традиційного підходи до розробки програм, орієнтованого рішення формалізованих завдань.

Неформалізовані завдання зазвичай мають такі особливості:

– помилковістю, неоднозначністю, неповнотою та суперечливістю вихідних даних;

– хибністю, неоднозначністю, неповнотою і суперечливістю знань про проблемну галузь і завдання, що розв'язується;

– великий розмірністю простору рішення, тобто. перебір під час пошуку рішення дуже великий;

– даними та знаннями, що динамічно змінюються.

Слід наголосити, що неформалізовані завдання становлять великий і дуже важливий клас завдань. Багато фахівців вважають, що це завдання є найбільш

масовим класом завдань, розв'язуваних ЕОМ.

Експертні системи та системи штучного інтелекту відрізняються від систем обробки даних тим, що в них в основному використовуються символьний (а не числовий) спосіб подання, символьний висновок та евристичний пошук рішення (а не виконання відомого алгоритму).

Експертні системи застосовуються на вирішення лише важких практичних (не іграшкових) завдань. За якістю та ефективності рішення експертні системи не поступаються рішенням експерта-людини. Рішення експертних систем мають «прозорість», тобто. можуть бути пояснені користувачеві на якісному рівні. Ця якість експертних систем забезпечується їхньою здатністю розмірковувати про свої знання та умовиводи. Експертні системи здатні поповнювати свої знання під час взаємодії з експертом. Необхідно відзначити, що в даний час технологія експертних систем використовується для вирішення різних типів завдань (інтерпретація, передбачення, діагностика, планування, конструювання, контроль, налагодження, інструктаж, управління) у найрізноманітніших проблемних галузях, таких як фінанси, нафтова та газова промисловість , енергетика, транспорт.

## 1.4 Структура експертних систем

Типова статична ЕС складається з таких основних компонентів:

– вирішувача (інтерпретатора);

– робочої пам'яті (РП), званої також базою даних (БД);

– основи знань (БЗ);

– компонентів набуття знань;

– пояснювального компонента;

– діалогового компонента.

База даних (робоча пам'ять) призначена для зберігання вихідних і

проміжних даних задачі, що вирішується в даний момент. Цей термін збігається за назвою, але не за змістом з терміном, що використовується в інформаційно-пошукових системах (ІПС) та системах управління базами даних (СУБД) для позначення всіх даних (насамперед довгострокових), що зберігаються в системі.

База знань (БЗ) в ЕС призначена для зберігання довгострокових даних, що описують область, що розглядається (а не поточних даних), і правил, що описують доцільні перетворення даних цієї області.

Вирішувач, використовуючи вихідні дані з робочої пам'яті та знання з БЗ, формує таку послідовність правил, які, будучи застосованими до вихідних даних, призводять до вирішення задачі.

Компонент придбання знань автоматизує процес наповнення ЕС знаннями, який здійснюється користувачем-експертом.

Пояснювальний компонентпояснює, як система отримала розв'язання задачі (або чому вона не отримала рішення) і які знання вона при цьому використовувала, що полегшує експерту тестування системи та підвищує довіру користувача до отриманого результату. Структура ЕС представлена на рисунку 2.1.

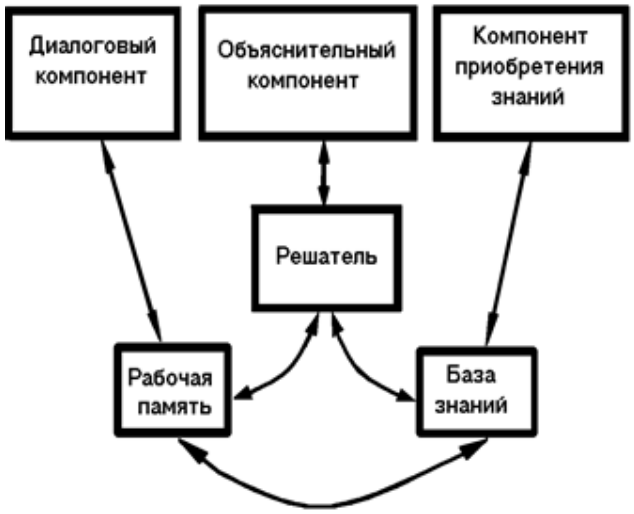


Рисунок 2.1 – Структура ЕС

Діалоговий компонент орієнтований на організацію дружнього спілкування з користувачем, як у ході вирішення завдань, так і в процесі набуття знань та пояснення результатів роботи.

У розробці ЕС беруть участь представники таких спеціальностей:

– експерт у проблемній галузі, завдання якої вирішуватиме ЕС;

– інженер із знань – спеціаліст із розробки ЕС (використовувані ним технологію, методи називають технологією (методами) інженерії знань);

– програміст з розробки інструментальних засобів, призначених для прискорення розробки ЕС.

Слід зазначити, що відсутність серед учасників розробки інженерів за знаннями (тобто їх заміна програмістами) або призводить до невдачі процес створення ЕС, або значно подовжує його.

Експерт визначає знання (дані та правила), що характеризують проблемну область, забезпечує повноту та правильність введених в ЕС знань.

Інженер зі знань допомагає експерту виявити та структурувати знання, необхідні для роботи ЕС, здійснює вибір того ІВ, яке найбільш підходить для даної проблемної галузі, та визначає спосіб представлення знань у цьому ІВ, виділяє та програмує (традиційними засобами) стандартні функції (типові для даної проблемної галузі), які будуть використовуватись у правилах, що вводяться експертом.

Програміст розробляє ІВ (якщо ІВ розробляється наново), що містить у межі всі основні компоненти ЕС, і здійснює його поєднання з тим середовищем, в якому воно буде використане.

Експертна система працює у двох режимах: режимі придбання знань та у режимі вирішення задачі (називається також режимом консультації або режимом використання ЕС).

У режимі набуття знаньспілкування з ЕС здійснює (через посередництво інженера зі знань) експерт. У цьому режимі експерт, використовуючи компонент набуття знань, наповнює систему знаннями, які дозволяють ЕС у

режимі вирішення самостійно (без експерта) вирішувати завдання із проблемної галузі. Експерт описує проблемну область у вигляді сукупності даних та правил. Дані визначають об'єкти, їх характеристики та значення, що існують в галузі експертизи. Правила визначають способи маніпулювання з даними, характерні для цієї області.

Необхідно відзначити, що режиму придбання знань у традиційному підході до розробки програм відповідають етапи алгоритмізації, програмування та налагодження, що їх програміст виконує. Таким чином, на відміну від традиційного підходу у випадку ЕС, розробку програм здійснює не програміст, а експерт (за допомогою ЕС), який не володіє програмуванням.

У режимі консультаціїспілкування з ЕС здійснює кінцевий користувач, якого цікавить результат та (або) спосіб його отримання. Необхідно відзначити, що в залежності від призначення ЕС користувач може не бути фахівцем у даній проблемній галузі (у цьому випадку він звертається до ЕС за результатом, не вміючи отримати його сам), або бути фахівцем (у цьому випадку користувач може сам отримати результат, але він звертається до ЕС з метою прискорити процес отримання результату, або покласти на ЕС рутинну роботу). У режимі консультації дані завдання користувача після обробки їх діалоговим компонентом надходять у робочу пам'ять. Вирішувач на основі вхідних даних з робочої пам'яті, загальних даних про проблемну область та правила з БЗ формує розв'язання задачі. ЕС при розв'язанні задачі не тільки виконує запропоновану послідовність операції, а й попередньо формує її. Якщо реакція системи не зрозуміла користувачеві, він може вимагати пояснення: «Чому система ставить те чи інше питання?», «як відповідь, збираний системою, отримано?».

Структуру, наведену на рисунку 2.1 називають структурою статичної ЕС. ЕС даного типу використовуються в тих додатках, де можна не враховувати зміни навколишнього світу, що відбуваються за час розв'язання задачі.

## 1.5 Найбільш поширені експертні системи

– OpenCyc – потужна динамічна ЕС із глобальною онтологічною моделлю та підтримкою незалежнихконтекстів;

– WolframAlpha - Пошукова система, інтелектуальний «обчислювальний двигун знань»;

– MYCIN – найбільш відома діагностична система, яка призначена для діагностики та спостереження за станом хворого при менінгіті та бактеріальних інфекціях;

– HASP/SIAP – інтерпретуюча система, яка визначає місце розташування та типи суден у Тихому океані за даними акустичних систем стеження;

– CLIPS – популярна оболонка для побудови ЕС;

– Акінатор – інтернет-гра. Гравець має загадати будь-якого персонажа, а Акінатор повинен його відгадати, ставлячи запитання. База знань автоматично поповнюється, тому програма може відгадати практично будь-якого відомого персонажа.

# 2 ТЕОРЕТИЧНІ ДОСЛІДЖЕННЯ

## 2.1 Категорії та сервіси хмарних технологій

В даний час виділяють три категорії «хмар»:

– громадські;

– приватні;

– гібридні.

Публічна хмара – це IT-інфраструктура, що використовується одночасно безліччю компаній та сервісів. Користувачі цих хмар не мають можливості керувати та обслуговувати цю хмару, вся відповідальність з цих питань покладена на власника цієї хмари. Абонентом пропонованих сервісів може стати будь-яка компанія та індивідуальний користувач. Вони пропонують легкий і доступний за ціною спосіб розгортання веб-сайтів або бізнес-систем з великими можливостями масштабування, які в інших рішеннях були б недоступні. Приклади: онлайн-сервіси Amazon EC2 і Simple Storage Service (S3), Google Apps/Docs, Salesforce.com, Microsoft Office Web.

Приватна хмара– це безпечна IT-інфраструктура, контрольована та експлуатована на користь однієї-єдиної організації. Організація може керувати приватною хмарою самостійно або доручити це завдання зовнішньому підряднику. Інфраструктура може розміщуватися або у приміщеннях замовника, або у зовнішнього оператора, або частково у замовника та частково у оператора. Ідеальний варіант приватної хмари це хмара, розгорнута на території організації, яка обслуговується та контролюється її співробітниками.

Гібридна хмара – це IT-інфраструктура, що використовує найкращі якості публічної та приватної хмари при вирішенні поставленого завдання. Часто такий тип хмар використовується, коли організація має сезонні періоди активності, іншими словами, як тільки внутрішня IT-інфраструктура не справляється з поточними завданнями, частина потужностей перекидається на

публічну хмару (великі обсяги статистичної інформації, які у необробленому вигляді не становлять цінності для підприємства) , а також для надання доступу користувачам до ресурсів підприємства (до приватної хмари).

Сьогодні більшість програм розробляються в одному середовищі, тестуються в іншому середовищі, а розвертаються в третьому. Тепер весь перелік операцій з розробки, тестування та розгортання веб-додатків можна виконати в одному інтегрованому середовищі, виключивши витрати на підтримку окремих середовищ для окремих етапів. Гнучка схема ціноутворення дозволяє у кілька разів знизити витрати. Для розгортання веб-застосунків не потрібно більше купувати обладнання та програмне забезпечення, немає необхідності організовувати підтримку – все це можна взяти в оренду.

Основні послуги, що надаються СС:

а) IaaS (інфраструктура як сервіс) – це надання комп'ютерної інфраструктури як послуги з урахуванням концепції СС. Замість закупівлі та самостійного забезпечення роботи фізичних серверів замовник отримує можливість керувати віртуальними серверами, які працюють на обладнанні постачальника послуг. Як приклад можна навести найбільший IaaS-сервіс Amazom EC2, яким користуються багато великих компаній. Користувачеві надається комп'ютерна інфраструктура, зазвичай віртуальні платформи (комп'ютери) пов'язані у мережу, які він самостійно налаштовує під власні цілі;

б) PaaS (платформа як сервіс) – передбачає наявність готової до роботи віртуальної платформи – засоби розробки додатків у хмарній середі. Для роботи з віддаленою платформою компанії не потрібно встановлювати додаткове програмне забезпечення. Все необхідне доступне через звичайний браузер. Найбільш відомими платформами для розробників є Google Apps Engine, Apple iPhone Apps, Microsoft Windows Azure. Ці платформи надають інтегровані платформи для розробки, тестування, розгортання та підтримки веб-застосунків як послуги. PaaS надає автоматичне виділення та звільнення

необхідних ресурсів залежно від кількості користувачів, що обслуговуються додатком, надійність і безпека вже вбудовані в PaaS і не вимагають додаткових витрат, наприклад, у вигляді розробки або конфігурування. Програми, розгорнуті на основі PaaS, повинні автоматично та надійно підтримувати використання у веб-масштабі, забезпечувати безпеку обміну конфіденційною інформацією та виконання грошових транзакцій. Розробники повинні мати можливість вільно створювати додатки з підтримкою безпеки даних про клієнтів, мережного трафіку, вихідного коду (інтелектуальної власності), навіть у разі відмови обладнання, на якому розгорнуто платформу. Здатність створювати вихідний код і надавати його в загальний доступ усередині розробки значно підвищує продуктивність зі створення додатків на основі PaaS. Можливість визначення, зміни та відстеження графіків виконання, завдань, областей відповідальності, ролей (проектувальники, розробники, тестери, QC) на основі прав доступу. PaaS доступна скрізь;

в) SaaS (додатки у вигляді сервісів) – надання програмного забезпечення як послуги, скористатись якою можна віддалено через інтернет. Як найбільш очевидні приклади SaaS можна навести поштовий сервіс Gmail і Google Calendar. Цей вид послуги зазвичай позиціонується як «програмне забезпечення на вимогу», це програмне забезпечення розгорнуте на віддалених серверах і користувач може отримувати доступ до нього через Інтернет, причому всі питання оновлення та ліцензій на дане програмне забезпечення регулюється постачальником цієї послуги. Оплата в даному випадку провадиться за фактичне використання програмного забезпечення;

г) Everything as a Service. При такому вигляді сервісу користувачеві буде надано все від програмно-апаратної частини та до управління бізнес-процесами, включаючи взаємодію між користувачами, від користувача потрібна лише наявність доступу до мережі Інтернет. Даний вид сервісу це більш загальне поняття по відношенню до наведених нижче послуг, що є більш окремими випадками:

1) апаратне забезпечення як послуга (Hardware as a Service). У цьому випадку користувачеві послуги надається обладнання, на правах оренди, яке він може використовувати для власних цілей. Даний варіант дозволяє економити на обслуговуванні даного обладнання, хоча за своєю суттю мало чим відрізняється від виду послуги «Інфраструктура як сервіс» за винятком того, що ви маєте голе обладнання на основі якого розгортаєте свою власну інфраструктуру з використанням найбільш відповідного програмного забезпечення;

2) робоче місце як послуга (Workplace as a Service). У разі компанія використовує СС для організації робочих місць своїх співробітників, налаштувавши і встановивши все необхідне програмне забезпечення, необхідне роботи персоналу;

3) дані як послуга (Data as a Service). Основна ідея цього виду послуги полягає в тому, що користувачеві надається дисковий простір, який може використовувати для зберігання великих обсягів інформації;

4) безпека як сервіс (Security as a Service). Даний вид послуги надає можливість користувачам швидко розгортати продукти, що дозволяють забезпечити безпечне використання веб-технологій, безпеку електронного листування, а також безпеку локальної системи, що дозволяє користувачам даного сервісу економити на розгортанні та підтримці своєї власної системи безпеки.

## 2.2 Хмарні платформи

Інтерес до хмарних платформ (PaaS) зростає з кожним роком. З одного боку, зараз домінують такі великі гравці як Google, Microsoft і Salesforce, з іншого боку, це не зупиняє незалежні команди розробляти нові PaaS-проекти, щоправда виконані в дещо іншій «ваговій категорії» – це перш за все heroku.com, picloud .com, kodingen.com та інші. Далі будуть розглянуті лише

найбільші гравці на ринку PaaS, які мають свою власну хмарну серверну інфраструктуру та PaaS-рішення на її базі:

– Windows Azure. Ця хмарна платформа складається з трьох головних складових: Windows Azure (операційна система, що забезпечує розподілені обчислення), SQL Azure (це оптимізований для розподілених обчислень SQL Server), Windows Azure AppFabric (колекція дуже різнорідних сервісів/додатків, у тому числі для нехмарного використання) ) [5]. Для створення мережі датацентрів для Windows Azure Platform Microsoft витратила 2,3 мільярда доларів. Фізично в цю платформу входить 6 датацентрів, у кожному з яких зазвичай встановлено від 200 тис. до 400 тис. серверів. В інфраструктуру входять також 22 вузли доставки контенту (Content Delivery Nodes, CDN), які розкидані по всьому світу. Основна програмна платформа для запуску в цій PaaS – це .NET Framework, додатки на якій мають бути скомпільовані у CLR. Але незважаючи на це, Azure також дозволяє здійснювати запуск та PHP-додатків. Крім цього, доступні два додаткові SDK: Java SDK for AppFabric і Ruby SDK for AppFabric, що відповідно дозволяє створювати свої додатки для AppFabric мовами Java і Ruby;

– Amazon Web Services – ветеран CC, що просуває свої популярні сервіси Elastic Compute Cloud та Simple Storage Service, які дозволяють розмістити на хмарній платформі будь-яке програмне середовище [6]. Незважаючи на те, що AWS спроектована так, що має бути максимально універсальною, межа між визначенням цієї платформи як IaaS або PaaS часом дуже тонка. Нещодавнє відкриття «чистої»PaaS-служби Amazon Elastic Beanstalkзабезпечило швидкий вихід компанії на PaaS-ринок. Тепер Amazon пропонує запуск програм Java, і, як сказано в прес-релізі, найближчим часом додатково буде забезпечена підтримка Ruby та Ruby on Rails. Вихід цього сервісу розглядають як симетричну відповідь Amazon на запуск VMForce для Java-додатків (спільними зусиллями Salesforce та VMware). Крім цього необхідно враховувати ту виняткову роль AWS у розміщенні на своїх потужностях (в рамках AWS та

EC2) так званих бездомних PaaS-провайдерів, з яких найвідоміші, цеKodingen, DotCloud, CloudBees, AppHarbor, PiCloud, Herokui багато-багато інших. Успіхи та доходи багатьох цих молодих PaaS-платформ – це насамперед успіх IaaS-платформи Amazon;

– Google App Engine (GAE) – один із найперших хмарних PaaS-сервісів, який був запущений у квітні 2008 року [7]. GAE орієнтований на підтримку роботи бізнес-додатків написаних мовами Python та Java. Для мови Python підтримується безліч його популярних розширень та фреймворків. В рамках GAE існує безліч специфічних рішень, часто несумісних з реалізацією цих технологій за межами GAE. Наприклад, найпопулярніший Django web framework може використовуватися на GAE, але з різними модифікаціями, той же Grails web application framework також вимагає модифікацій при перенесенні (в даному випадку існує додатковий додатковий плагін App Engine Plugin) і т.д. Тому дуже багато розробників на GAE побоюються, що вони можуть «ув'язнути» зі своїми рішеннями на цій дуже специфічній платформі. що також не додає плюсів до її реалізації та популярності. Ще один приклад для ілюстрації підходу Google, це використання власної мови вибірки GQL. GQL, хоч і схожий на звичний SQL, все-таки має багато специфіки, вимагаючи переробки та адаптації своїх додатків саме до цієї платформи. У GAE ви не можете писати безпосередньо до файлової системи – файлова система надається в режимі «тільки читання». Тому зберігати всі дані в GAE можна лише через Google Datastore API; У GAE ви не можете писати безпосередньо до файлової системи – файлова система надається в режимі «тільки читання». Тому зберігати всі дані в GAE можна лише через Google Datastore API; У GAE ви не можете писати безпосередньо до файлової системи – файлова система надається в режимі «тільки читання». Тому зберігати всі дані в GAE можна лише через Google Datastore API;

– Force.com – одна з перших і на даний момент найвідоміших PaaS-платформ, що просувається лідером хмарного ринку компанією Salesforce,

призначена насамперед для швидкого створення власних SaaS-рішень [8]. І хоча компанія сама називає свою платформу як «development as a service», більш стандартним і усталеним позначенням такого типу хмарної платформи є «platform as a service» (PaaS). Фізично хмара force.com розташована в 8-ми дата-центрах, географічно розкиданих по всьому світу. Розробка ведеться мовами Apex (дуже схожий по синтаксису на Java) і Visualforce (використовується XML-мова для створення інтерфейсів користувача, Ajax, Flex і всього різноманіття HTML-сутностей). Написаний додаток інтегрується в salesforce.com і є доступним для продажу (прокату) в рамках можливостей SaaS-платформи SalesForce;

– Heroku – це великий постачальник платформи для розробок Ruby-додатків, зокрема він надає розробникам онлайновий Rack-інтерфейс і Ruby on Rails [9]. На початок 2011 року сервіс успішно подолав позначку 120000 додатків, що надаються на його платформі. Heroku називає себе принциповим послідовником NoSQL-рішень, тому як addon-рішень, які до речі можна встановити буквально одним кліком, компанія пропонує Cloudant, Membase, MongoDB, Redis та багато інших популярних NoSQL-продуктів. У листопаді 2010 року Heroku уклала партнерський договір з Facebook, який просуває новий продукт – Heroku Facebook App Package, – який дозволяє компаніям будь-якого масштабу створювати свої програми для Facebook, при цьому за твердженням Hiroku,

– Hivext. Молодий вітчизняний гравець на ринку PaaS-рішень, що дозволяє вести розробку бізнес-логіки веб-додатків на різних мовах програмування (Java, JavaScript, PHP тощо) [10]. Цікаво, що з цього різноманіття принципово відсутня підтримка .NET. Наразі Hivext розгорнуть на власних серверах у трьох територіально рознесених датацентрах: у Києві, Житомирі (Україна) та Санкт-Петербурзі (Росія). Розцінки на послуги поки не відомі, але передбачається, що саме середовище та інструментарій для розробки надаватимуться безкоштовно. Програми зберігатимуться в «хмарі», а

плата стягуватиметься за хостинг та за звернення до цього додатка клієнтів розробника. Для збільшення швидкості розробки програм платформа включає широкий набір вже готових базових сервісів,

Головна сутність усіх нововведень – відкритіший процес розробки. Якщо раніше на ринку, і у MS Azure, і у GAE, пропонувався жорстко заданий набір інструментів розробки (як правило з дуже специфічних мов та закритих стандартів), то з покупкою та інтеграцією у свій сервіс хмарних розробок Ruby-платформи Heroku, Salesforce показала перший приклад принципово нової стратегії над ринком – опора на визнані відкриті технології та відомі стандарти. Втім, незважаючи на поточні роботи з інтеграції можливостей Heroku в послуги Force.com, розглянемо цей сервіс самостійним пунктом, яким він і бачиться для споживача.

## 2.3 Характеристики хмарних технологій

Сервіси, побудовані за допомогою технологій СС, мають наступні переваги:

а) доступність. Хмари доступні всім, з будь-якої точки, де є Інтернет, з будь-якого комп'ютера, де є браузер. Це дозволяє користувачам (підприємствам) економити на закупівлі високопродуктивних, дорогих комп'ютерів. Також співробітники компаній стають мобільнішими так, як можуть отримати доступ до свого робочого місця з будь-якої точки земної кулі, використовуючи ноутбук, нетбук, планшетник або смартфон. Немає необхідності в покупці ліцензійного ПЗ, його налаштування та оновлення, необхідно просто заходьте на сервіс та користуєтеся його послугами, заплативши за фактичне використання;

б) низька вартість. Проосновні фактори, що знизили вартість використання хмар наступні:

1) зниження витрат на обслуговування віртуальної інфраструктури,

викликане розвитком технологій віртуалізації, за рахунок чого потрібний менший штат для обслуговування всієї IT інфраструктури підприємства;

2) оплата фактичного використання ресурсів, користувач хмари платить за фактичне використання обчислювальних потужностей хмари, що дозволяє ефективно розподіляти свої кошти. Це дозволяє користувачам (підприємствам) заощаджувати на купівлі ліцензій до ПЗ;

3) використання хмари на правах оренди дозволяє користувачам знизити витрати на закупівлю дорогого обладнання, і зробити акцент на вкладення коштів на налагодження бізнес-процесів підприємства, що у свою чергу дозволяє легко розпочати бізнес;

4) розвиток апаратної частини обчислювальних систем у зв'язку з чим зниження вартості обладнання.

в) гнучкість – необмеженість обчислювальних ресурсів (пам'ять, процесор, диски), за рахунок використання систем віртуалізації, процес масштабування та адміністрування «хмар» стає досить легким завданням, оскільки «хмара» самостійно може надати вам ресурси, які вам необхідні, а ви платите лише за фактичне їх використання;

г) надійність – надійність «хмар», що особливо знаходяться у спеціально обладнаних ЦОД, дуже висока так, як такі ЦОД мають резервні джерела живлення, охорону, професійних працівників, регулярне резервування даних, високу пропускну спроможність Інтернет каналу, високу стійкість до DDOS атак;

д) безпека – «хмарні» сервіси мають досить високу безпеку при належному її забезпеченні, проте при недбалому відношенні ефект може бути повністю протилежним;

е) великі обчислювальні потужності – користувач «хмарної» системи може використовувати усі її обчислювальні здібності, заплативши лише за фактичний час використання. Підприємства можуть використовувати цю можливість для аналізу великих обсягів даних.

Незважаючи на всі переваги СС, вони також мають і ряд недоліків:

– постійне з'єднання з мережею– для отримання доступу до послуг «хмари» потрібне постійне з'єднання з Інтернетом. Однак у наш час це не такий і великий недолік, особливо з приходом технологій стільникового зв'язку 3G та 4G;

– програмне забезпечення та його кастомізація– є обмеження щодо ПЗ, яке можна розгортати на «хмарах» та надавати його користувачеві. Користувач ПЗ має обмеження у ПЗ і іноді не має можливості налаштувати його під свої власні цілі.

– конфіденційність даних зберігаються на публічних «хмарах» в даний час викликає багато суперечок, але в більшості випадків експерти сходяться в тому, що не рекомендується зберігати найбільш цінні для компанії документи на публічному «хмарі», оскільки в даний час немає технології, яка б гарантувала 100%. конфіденційність даних, що зберігаються;

– надійність – що стосується надійності інформації, що зберігається, то з упевненістю можна сказати, що якщо інформація, що зберігається в «хмарі», була втрачена, то це безповоротно;

– безпека – «хмара» сама по собі є досить надійною системою, проте при проникненні на неї зловмисник отримує доступ до величезного сховища даних. Ще один мінус це використання систем віртуалізації, в яких як гіпервізор використовуються ядра стандартні ОС такі, як Linux, Windows та ін, що дозволяє використовувати віруси;

– дорожнеча обладнання – для побудови власної хмари компанії необхідно виділити значні матеріальні ресурси, що не вигідно щойно створеним і малим компаніям.

Моделі розгортання, які використовуються.

Приватна хмара (Private Cloud). Хмарна інфраструктура, підготовлена для ексклюзивного використання єдиною організацією, що включає кілька споживачів (наприклад, бізнес-одиниць). Така хмара може перебувати у

власності, управлінні та обслуговуванні у самої організації, у третьої сторони та розташовуватися як на території підприємства, так і за його межами.

Хмара спільноти та комунальна хмара (Community cloud). Хмарна інфраструктура, підготовлена для ексклюзивного використання конкретної спільноти споживачів від організацій, які мають спільні проблеми (наприклад, місії, вимоги безпеки, політики). Хмара може перебувати у власності, управлінні та обслуговуванні в однієї або більше організацій у співтоваристві, у третій стороні розташовуватися як на території організацій, так і за їх межами.

Публічна (або загальна) хмара (Public cloud). Хмарна інфраструктура, підготовлена широкому загалу для відкритого використання. Воно може перебувати у власності, управлінні та обслуговуванні у ділових, наукових та урядових організацій у будь-яких їх комбінаціях. Хмара є на території хмарного провайдера.

Гібридна хмара (Hybrid cloud). Хмарна інфраструктура є композицією з двох або більше різних інфраструктур хмар (приватні, громадські або державні), що мають унікальні об'єкти, але пов'язані між собою стандартизованими або власними технологіями, які дозволяють переносити дані або програми між компонентами (наприклад, для балансування навантаження між хмарами).

## 2.4 Етапи розробки експертних систем

Розробка ЕС має суттєві відмінності від розробки стандартного програмного продукту. Досвід створення ЕС показав, що використання при розробці методології, прийнятої в традиційному програмуванні, або надмірно затягує процес створення ЕС, або взагалі призводить до негативного результату.

Використовувати ЕС слід лише тоді, коли розробка ЕС можлива, виправдана і методи інженерії знань відповідають завданню, що вирішується.

Щоб розробка ЕС була можливою для даної програми, необхідне одночасне виконання принаймні таких вимог:

– існують експерти в даній галузі, які вирішують завдання значно краще, ніж початківці;

– експерти сходяться в оцінці запропонованого рішення, інакше не можна буде оцінити якість розробленої ЕС;

– експерти здатні вербалізувати (виразити природною мовою) і пояснити використовувані ними методи, інакше важко розраховувати на те, що знання експертів будуть «витягнуті» і вкладені в ЕС;

– вирішення завдання потребує лише міркувань, а чи не дій;

– завдання має бути надто важким (тобто. її рішення має займати в експерта кілька годин чи днів, а чи не тижнів);

– завдання хоч і має бути виражена у формальному вигляді, проте, все-таки має ставитися до досить «зрозумілої» і структурованої області, тобто. мають бути виділені основні поняття, відносини та відомі (хоча б експерту) способи отримання розв'язання задачі;

– вирішення завдання не повинно значною мірою використовувати «здоровий глузд» (тобто широкий спектр загальних відомостей про світ і про спосіб його функціонування, які знає та вміє використовувати будь-яка нормальна людина), оскільки подібні знання поки не вдається (у достатній кількості ) вкласти у системи штучного інтелекту.

Додаток відповідає методам ЕС, якщо вирішуване завдання має сукупність наступних характеристик:

– завдання може бути природним чином вирішена за допомогою маніпуляції із символами (тобто за допомогою символічних міркувань), а не маніпуляцій із числами, як заведено в математичних методах та в традиційному програмуванні;

– завдання має мати евристичну, а чи не алгоритмічну природу, тобто. її рішення має вимагати застосування евристичних правил. Завдання, які

можуть бути гарантовано вирішені (з дотриманням заданих обмежень) за допомогою деяких формальних процедур, не підходять до застосування ЕС;

– завдання має бути досить складним, щоб виправдати витрати на розробку ЕС. Однак вона не повинна бути надмірно складною (рішення займає в експерта години, а не тижні), щоб ЕС могла її вирішувати;

– завдання має бути досить вузьким, щоб вирішуватися методами ЕС, і практично значущою.

Під час розробки ЕС, зазвичай, використовується концепція «швидкого прототипу». Суть цієї концепції у тому, що розробники намагаються відразу побудувати кінцевий продукт. На початковому етапі вони виробляють прототип (прототипи) ЕС. Прототипи повинні задовольняти двом суперечливим вимогам: з одного боку, вони повинні вирішувати типові завдання конкретного додатка, а з іншого - час і трудомісткість їх розробки повинні бути незначними, щоб можна було максимально запаралелити процес накопичення і налагодження знань (здійснюваний експертом) з процесом вибору (розробки) програмних засобів (здійснюваним інженером зі знань та програмістом). Для задоволення зазначених вимог, як правило, під час створення прототипу використовуються різноманітні засоби, що прискорюють процес проектування.

Прототип повинен продемонструвати придатність методів інженерії знань для цієї програми. У разі успіху експерт за допомогою інженера знань розширює знання прототипу про проблемну область. При невдачі може знадобитися розробка нового прототипу або розробники можуть дійти висновку про непридатність методів ЕС даного докладання. У міру збільшення знань прототип може досягти такого стану, коли він успішно вирішує всі завдання цієї програми. Перетворення прототипу ЕС на кінцевий продукт зазвичай призводить до перепрограмування ЕС мовами низького рівня, що забезпечують як збільшення швидкодії ЕС, так і зменшення пам'яті. Трудомісткість і час створення ЕС значною мірою залежить від типу

використовуваного інструментарію.

У ході робіт зі створення ЕС склалася певна технологія їхньої розробки, що включає шість наступних етапів (рисунок 2.1).

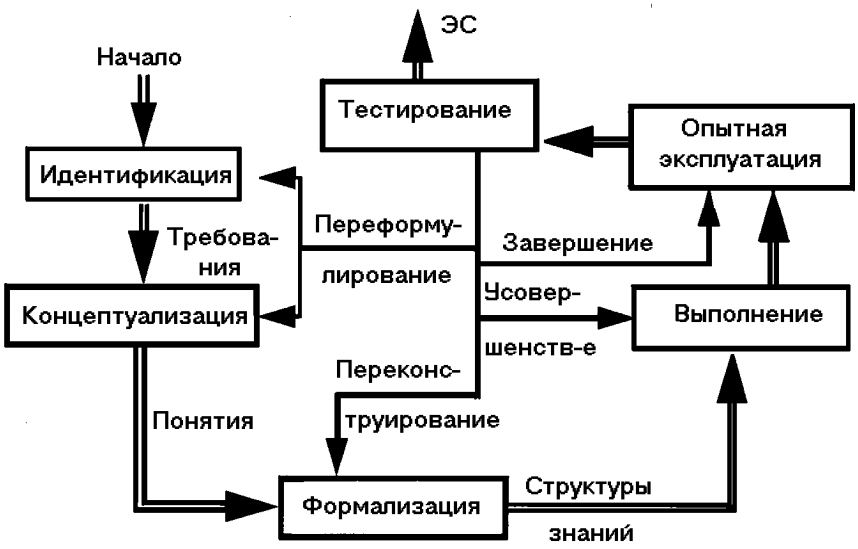


Рисунок 2.1 – Етапи створення ЕС

На етапі ідентифікації визначаються завдання, що підлягають вирішенню, виявляються цілі розробки, визначаються експерти та типи користувачів.

На етапі концептуалізації проводиться змістовний аналіз проблемної галузі, виявляються поняття та їх взаємозв'язки, що використовуються, визначаються методи вирішення завдань.

На етапі формалізації вибираються ІВ і визначаються способи подання всіх видів знань, формалізуються основні поняття, визначаються способи інтерпретації знань, моделюється робота системи, оцінюється адекватність цілям системи зафіксованих понять, методів рішень, засобів подання та маніпулювання знаннями.

На етапі виконання здійснюється наповнення експертом бази знань. У зв'язку з тим, що основою ЕС є знання, цей етап є найважливішим і трудомістким етапом розробки ЕС. Процес придбання знань поділяють на

отримання знань з експерта, організацію знань, що забезпечує ефективну роботу системи, та подання знань у вигляді, зрозумілому ЕС. Процес набуття знань здійснюється інженером зі знань на основі аналізу діяльності експерта з вирішення реальних завдань.

## 2.5 Подання знань у експертних системах

Перше і основне питання, яке треба вирішити при поданні знань – це визначення складу знань, тобто. визначення того, що представляти в експертній системі. Друге питання стосується того, як представляти знання. Слід зазначити, що ці дві проблеми є незалежними. Справді, обраний спосіб уявлення може бути непридатним у принципі чи неефективним висловлювання деяких знань.

Питання «Як представляти» можна розділити на дві, значною мірою, незалежні завдання: як організувати (структурувати) знання та як уявити знання у вибраному формалізмі.

Прагнення виділити організацію знань у самостійну задачу викликано, зокрема, тим, що це виникає для будь-якої мови уявлення та способи вирішення цього завдання є однаковими (або подібними) незалежно від використовуваного формалізму.

До кола питань, які вирішуються при поданні знань, включають:

– визначення складу представлених знань;
– організацію знань;
– уявлення знань, тобто. визначення моделі уявлення.

Склад знань ЕС визначається такими факторами:

– проблемним середовищем;
– архітектурою експертної системи;
– потребами та цілями користувачів;
– мовою спілкування.

Відповідно до загальної схеми статичної експертної системи для її функціонування потрібні такі знання:

– знання про процес вирішення задачі (тобто керуючі знання), які використовуються інтерпретатором (вирішуватель);

– знання про мову спілкування та способи організації діалогу, які використовуються лінгвістичним процесором (діалоговим компонентом);

– знання про способи подання та модифікації знань, що використовуються компонентом набуття знань;

– підтримують структурні та керуючі знання, що використовуються пояснювальним компонентом.

Склад знань про мову спілкування залежить як мови спілкування, і від необхідного рівня розуміння.

Предметні знання містять дані про предметної області та способи перетворення цих даних при вирішенні поставлених завдань. Необхідно відзначити, що стосовно предметних знань знання про подання та знання про управління є метазнаннями. У предметних знаннях можна назвати описувачі і власне предметні знання. Описувачі містять певну інформацію про предметні знання, таку як коефіцієнт визначеності правил і даних, заходи важливості і складності. Власне предметні знання розбиваються на факти та твердження, що виконуються. Факти визначають можливі значення сутностей та характеристик предметної області. Затвердження, що виконуються, містять інформацію про те, як можна змінювати опис предметної області в ході вирішення завдань. Іншими словами, виконувані твердження – це знання, що задають процедури обробки.

## 2.6 Організація знань у робочій системі

Робоча пам'ять (РП) експертних систем варта зберігання даних. Дані робочої пам'яті можуть бути однорідні або поділяються на рівні за типами даних. У разі кожному рівні робочої пам'яті зберігаються дані відповідного

типу. Виділення рівнів ускладнює структуру експертної системи, але робить систему ефективнішою. Наприклад, можна виділити рівень планів, рівень агенди (упорядкованого списку правил, готових до виконання) та рівень даних предметної галузі (рівень рішень).

У сучасних експертних системах дані в робочій пам'яті розглядаються як ізольовані чи пов'язані. У першому випадку робоча пам'ять складається з безлічі простих елементів, а в другому – з одного або кількох (при кількох рівнях у РП) складних елементів (наприклад, об'єктів). При цьому складний елемент відповідає множині простих, об'єднаних в єдину сутність. Теоретично обидва підходи забезпечують повноту, але використання ізольованих елементів у складних предметних галузях призводить до втрати ефективності.

Дані в РП у найпростішому випадку є константами та (або) змінними. При цьому змінні можуть трактуватися як характеристики деякого об'єкта, а константи як значення відповідних характеристик. Якщо РП потрібно аналізувати одночасно кілька різних об'єктів, що описують поточну проблемну ситуацію, необхідно вказувати, яких об'єктів ставляться аналізовані характеристики. Одним із способів вирішення цього завдання є явна вказівка того, до якого об'єкта відноситься характеристика.

Якщо РП складається із складних елементів, то зв'язок між окремими об'єктами вказується явно, наприклад, завданням семантичних відносин. У цьому кожен об'єкт може мати внутрішню структуру. Слід зазначити, що з прискорення пошуку та зіставлення дані у РП може бути пов'язані як логічно, а й асоціативно.

## 2.7 Визначення та принцип роботи JESS

JESS – двигун правил і середовище скриптів, повністю написана мовою Java Ернестом Фрідманом-Хіллом в Sandia National Laboratiries в місті Лівермор, штат Каліфорнія [11]. Мова JESS є вузькоспеціалізованою формою

мови Lisp. JESS дозволяє розробляти Java-додатки, які мають здатність виводу, використовуючи знання, які представлені у вигляді декларативних правил. JESS це невеликий, легкий, і один з найшвидших движків правил на сьогоднішній день. Потужна скриптова мова дозволяє отримати доступ до всіх Java API. JESS включає повнофункціональне середовище розробки, засноване на платформі Eclipse. JESS має безліч унікальних функцій, включаючи зворотний логічний висновок, запити до робочої пам'яті і може безпосередньо управляти та виводити Java об'єкти. JESS також є потужним середовищем скриптів,

JESS використовує розширену версію алгоритму Rete для обробки правил. Rete – ефективнийалгоритм зіставлення зі зразком для продукційних систем, експертних систем і баз знань. Rete став основою багатьох популярних експертних систем, включаючиCLIPS, JESS, Drools, BizTalk Rules Engine і Soar. При буквальній реалізації експертна система перевіряє застосовність кожногоправила виведення до кожного фактом бази знань, При необхідності виконує його і переходить до наступного правила, повертаючись на початок при вичерпанні всіх правил. Навіть для невеликого набору правил та фактів такий метод працює неприйнятно повільно. Алгоритм Rete забезпечує більш високу ефективність. При використанні Rete експертна система будує спеціальнийграф або префіксне дерево, вузлам якого відповідають частини умов правил Шлях від кореня до листа утворює повну умову деякої продукції. У процесі роботи кожен вузол зберігає список фактів, що відповідають умові. При додаванні чи модифікації факту він проганяється мережею, у своїй відзначаються вузли, умовам яких цей факт відповідає. За виконання повної умови правила, коли система досягає листа графа, правило виконується.

Алгоритм Rete жертвує обсягом пам'яті заради швидкості. Найчастіше швидкість зростає порядки (оскільки ефективність теоретично залежить від кількості правил у системі). В експертних системах з великою кількістю правил класичний Rete вимагає надто багато пам'яті, але з'явилися нові алгоритми, у

тому числі засновані на Rete, що обмежуються розумним обсягом.

## 2.8 Синтаксис мови JESS

Основні одиниці синтаксису мови JESS:

– символи – основна концепція мови. Вони схожі на ідентифікатори в інших мовах. Символи можуть містити літери, цифри та певні знаки. Символ не може починатися з цифри, але може починатися з цих символів (деякі мають спеціальне призначення). Існують спеціальні символи, які JESS інтерпретує спеціальним чином: nil, TRUE, FALSE. Символи реєстрозалежні;

– числа. JESS використовує функції java parseInt(java.lang.String), parseLong(java.lang.String) та parseDouble(java.lang.String) для аналізу цілих, довгих і дробових значень;

– рядки. Символьні рядки в JESS поміщені в лапки. Коса характеристика може бути використана виділення спеціальних символів у рядку;

– списки – фундаментальна одиниця у JESS. Списки завжди складаються з набору дужок, що огороджують, і одного або більше символів, чисел, рядків або інших списків усередині цих дужок;

– виклик функцій. Як і в Lisp, весь код в JESS (керівні структури, присвоєння, виклик процедур) набувають вигляду виклику функції. Деякі функції мають імена, які виглядають так само, як і Java оператори і мають таку ж логіку роботи. Виклик функцій у JESS – це простий список, голова якого – ім'я функції;

– змінні. Ідентифікатори, які починаються зі знака питання, є змінними JESS. Ім'я змінної може містити будь-яку комбінацію літер, цифр та символів, окрім символу точки. Змінні можуть посилатися на символи, числа, рядки чи списки. Присвоїти значення змінної можна за допомогою функції bind;

– визначення функцій. Для визначення функції використовується конструкція deffunction. Ім'я функції має бути символьним. Кожен параметр має бути змінною. Коментар повинен бути укладений у лапки. Функція може мати довільну кількість виразів. Специфікатор повернення дає значення функції, що повертається;

– шаблонів. Кожен факт має шаблон. Факт отримує ім'я та список слотів із шаблону. Тому шаблон в JESS схожий на клас Java. Шаблони створюються автоматично, якщо їх не створювати вручну. Шаблон створюється за допомогою конструкції deftemplate. Оголошення шаблону включає ім'я, рядок документації, умову розширення, список об'яв та список визначення слотів;

– оголошення правил. Правила JESS схожі на вирази «if then» у процедурній мові, але вони використовуються по-іншому. У той час як «if then» виконується у певний час та у певному порядку, залежно від того, як програміст написав їх, правила JESS виконуються тоді, коли їхня ліва (умовна) частина задоволена.

## 2.9 Основні функції та призначення Google App Engine

Google App Engine дозволяє виконувати веб-програми в інфраструктурі Google. Програми App Engine легко створювати, підтримувати та масштабувати у міру зміни трафіку та потреб у просторі для зберігання. Робота в App Engine не потребує підтримки серверів. Програму можна зробити доступною для всіх або обмежити доступ тільки для членів певної організації. Google App Engine підтримує програми, написані кількома мовами програмування. У середовищі виконання Java на App Engine можна створювати програми за допомогою стандартних технологій Java, включаючи JVM, сервлетів Java та програмної мови Java, або за допомогою інших мов, використовуючи інтерпретатор або компілятор на базі JVM, наприклад, JavaScript або Ruby. App Engine також

надає спеціальне середовище виконання Python, включаючи інтерпретатор Python та стандартну бібліотеку Python. Середовище виконання Java та Python забезпечують швидке та безпечне виконання програми, без втручання з боку інших програм системи.

Почати використовувати App Engine можна безкоштовно. Всі програми можуть безкоштовно використовувати до 500 МБ простору для зберігання і достатню кількість процесорного часу і трафіку, щоб ефективно обслуговувати програми з рівнем відвідувань близько 5 мільйонів переглядів сторінок на місяць. Коли буде включено функцію оплати для програми, межі обмежень на безкоштовні ресурси будуть збільшені, і оплату потрібно буде проводити лише за ресурси, використані понад безкоштовну квоту.

Google App Engine спрощує створення програми, яка надійно виконуватиметься навіть в умовах високого завантаження та роботи з великими обсягами даних. App Engine включає такі функції:

– динамічна робота в Інтернеті з повною підтримкою основних веб-технологій;

– постійне сховище із запитами, сортуванням та транзакціями;

– автоматичне масштабування та регулювання навантаження;

– API для автентифікації користувачів та надсилання електронної пошти за допомогою облікових записів Google;

– повнофункціональне локальне середовище розробки, яке імітує роботу Google App Engine;

– черги завдань для виконання завдань, що не належать до дії веб-запиту;

– заплановані завдання для запуску подій у певний час та через регулярні часові інтервали.

Програма може виконуватися в одному з двох доступних середовищ виконання: середа Java та середа Python. Кожне середовище надає стандартні протоколи та основні технології розробки веб-додатків.

Програми виконуються в безпечному середовищі з обмеженим доступом до операційної системи. За допомогою цих обмежень App Engine може розподіляти веб-запити для програми між кількома серверами, а також запускати та зупиняти сервери відповідно до потреб трафіку.

У середовищі виконання Java можна розробляти програми за допомогою звичайних інструментів веб-розробки Java та основних API. Додаток взаємодіє із середовищем за допомогоюстандарту Java Servlet і може використовувати звичайні технології веб-додатків, наприклад, JavaServer Pages(JSPs). У середовищі виконання Java використовується Java 6. SDK Java App Engine підтримує розробку програм за допомогою Java 5 або 6. Середовище включаєплатформу Java SE Runtime Environment (JRE) 6та бібліотеки. Обмеження середовища тестування реалізовані в JVM. Програма може використовувати будь-який байтовий код JVM або будь-яку функцію бібліотеки в межах обмежень тестового середовища. Програма може мати доступ до більшості служб App Engine за допомогою стандартних Java API. Для сховища даних App Engine, SDK Java включає реалізацію інтерфейсівоб'єктів даних Java (JDO) та API Java Persistence (JPA). Програма може використовуватиAPI JavaMailдля надсилання електронних листів за допомогою служби Mail App Engine. API HTTP java.net мають доступ до служби URL Fetch App Engine. App Engine також включає API нижнього рівня для своїх служб, за допомогою яких можна реалізовувати додаткові адаптери або використовувати служби безпосередньо з програми.

App Engine надає потужну службу розподіленого зберігання даних із підтримкою механізму запитів та транзакцій. Розширення розподіленого сховища даних разом із зростанням кількості даних можна порівняти зі зростанням розподіленого веб-сервера зі зростанням трафіку. Сховище даних App Engine відрізняється від звичайних реляційних баз даних. Об'єкти даних мають вигляд і набір властивостей. Запити можуть витягувати об'єкти певного виду, відфільтровані та відсортовані за значеннями властивостей. Значення

властивостей можуть належати до будь-якої з підтримуванихтипів значень властивостей. Об'єкти сховища даних вимагають використання схеми. Структура об'єктів даних надається та дотримується в коді програми. Інтерфейси Java JDO/JPA та інтерфейс сховища даних Python включають функції для застосування та дотримання структури в рамках програми. Програма може отримати прямий доступ до сховища даних, щоб використовувати необхідну частину структури. Сховище даних є суворо узгодженим і використовує оптимістичне управління паралелізмом. Оновлення об'єкта відбувається в рамках транзакції, виконання якої може бути повторено певну кількість разів, якщо інші процеси одночасно намагаються оновити той самий об'єкт. Програма може виконувати кілька операцій сховища даних у межах однієї транзакції. При цьому всі операції успішно виконуються, або не виконуються зовсім, забезпечуючи таким чином цілісність даних. У розподілених мережах сховище даних реалізує транзакції з допомогою «груп об'єктів». Транзакція управляє об'єктами не більше однієї групи. Об'єкти однієї групи зберігаються разом, щоб забезпечити ефективніше виконання транзакцій. Програма може призначати об'єкти певним групам у момент створення об'єктів.

App Engine підтримує інтеграцію програми з обліковими записами Google для виконання аутентифікації користувачів. Програма дозволяє користувачеві виконати вхід, використовуючи свій обліковий запис Google, і отримати доступ до адреси електронної пошти та відображуваного імені, пов'язаного з обліковим записом. За допомогою облікових записів Google користувач може швидше почати користуватися програмою, оскільки йому не доведеться створювати новий обліковий запис. Крім того, економиться час розробника на реалізацію системи облікових записів користувача для програми. API Users може вказувати програмі на те, чи є поточний користувач адміністратором програми. Ця можливість спрощує створення сайтів, доступних тільки для адміністратора.

App Engine надає кілька служб, що дозволяють виконувати основні операції керування програмою. Наступні API забезпечують доступ до служб:

– URL Fetch. За допомогою служби App Engine URL Fetch програма може отримати доступ до ресурсів в Інтернеті, наприклад, до веб-служб або інших даних. Служба URL Fetch отримує веб-ресурси за допомогою високошвидкісної інфраструктури Google, отримуючи веб-сторінки для різних продуктів Google;

– mail. Програми можуть надсилати повідомлення електронної пошти за допомогою служби Mail Сервера програми. Для надсилання електронних листів служба Mail використовує інфраструктуру Google;

– Memcache. Служба Memcache надає додатку високопродуктивний кеш пам'яті зі структурою «ключ-значення», до якого можуть отримувати кілька екземплярів програми. Служба Memcache також може бути використана для даних, при обробці яких не потрібно використовувати функції постійного зберігання та транзакційні функції сховища даних, наприклад тимчасові дані або дані, скопійовані зі сховища даних в кеш для прискорення доступу;

– обробка зображень. Служба Image дозволяє програмі працювати із зображеннями. За допомогою API цієї служби можна змінювати розмір, кадрувати, повертати та отримувати дзеркальне відображення зображень у форматах JPEG та PNG;

– заплановані завдання та служба Task Queues. Програми можуть виконувати завдання, які не стосуються відповідей на веб-запити. Програма може виконувати ці завдання за налаштованим вами розкладом, наприклад, щодня або щогодини. Програма також може виконувати завдання, додані в чергу самим додатком, наприклад, фонові завдання, створені під час обробки запиту.

Всі набори засобів для розробки App Engine(SDK) для Java та Python включають програму веб-сервера, яка імітує роботу всіх служб App Engine на локальному комп'ютері. Кожен SDK включає всі API і бібліотеки, доступні в App Engine. Веб-сервер також імітує безпечне середовище тестування, включаючи перевірку спроб доступу до ресурсів системи, заборонених у

середовищі виконання App Engine. Кожен SDK також включає інструмент для завантаження програми App Engine. При створенні нового основного випуску програми, яка вже виконується на App Engine, можна завантажити новий випуск як нову версію. Стара версія продовжуватиме працювати для користувачів, доки ви не перейдете на нову версію. Нову версію можна тестувати на App Engine, у той час як стара продовжує працювати.

Консоль адмініструваннянадає веб-інтерфейс для керування програмою, що виконується на App Engine. Консоль адміністрування може бути використана для створення нових програм, налаштування назв доменів, зміни активної версії програми, вивчення журналу доступу та помилок та перегляду сховища даних програми.

Є можливість створити обліковий запис і опублікувати програму, яку користувачі можуть відразу ж почати безкоштовно використовувати без додаткових зобов'язань. Додаток на безкоштовному обліковому записі може використовувати до 500 МБ простору для зберігання та до 5 мільйонів переглядів сторінок на день. Коли використання ресурсів потрібно збільшити, можна увімкнути функцію оплати, встановити максимальний щоденний бюджет та розподілити його між усіма ресурсами відповідно до своїх потреб. На один обліковий запис розробника можна зареєструвати до 10 додатків. Для кожної програми ресурси розподіляються в рамках обмежень або квот. Квота визначає, яка кількість певного ресурсу може бути використана програмою протягом одного календарного дня.

## 3.10 Використання хмарних побчислень під час пандемії

Як показав 2021 рік, держави прагуть скоротити кількість локдаунів, але незважаючи на те, що в 2021 році вони вкрай рідкісні, все одно пандемічні ризики для бізнесу залишаються ключовими і найболючішими, тому що, по суті, вони впливають на виживання бізнесу за рахунок швидкості адаптації. у

періоди оголошення локдаунів. Євген Горохов, співзасновник M1Cloud та виконавчий директор Stack Group, розповів, що у 2021 році успішність бізнесу як і раніше визначатиметься, у тому числі тим, які інструменти бізнес навчився швидко розгортати та використовувати у періоди віддаленої роботи та локдаунів, які, як показує практика, що вводяться без попереднього попередження. Як ми й прогнозували, у 2021 році зберігаються і, мабуть, ще кілька років зберігатимуться обмеження, пов'язані з переміщенням людей між країнами, усередині країн діють заходи, пов'язані з відвідуванням місць масового скупчення людей.

Звичайно, значна частина великого та середнього бізнесу внесла падемічні ризики у стратегії розвитку на 2021 рік, тому багато компаній виділили суттєві бюджети на IT-рішення, у тому числі для віддаленої роботи, роботи онлайн-сервісів, розробки програм, які забезпечують безперервність роботи бізнесу. Багато компаній, протягом 2021 року, так і не повернули весь персонал до офісів, багато співробітників залишилися працювати у віддаленому форматі, тому тепер бізнесу доведеться постійно мати справу з віддаленими робочими столами та забезпечувати інформаційну безпеку. Попитом користувалися технології VDI, які дозволяють оперативно розгортати віддалені робочі місця у захищених хмарах. Тепер VDI як інструмент для безперервної роботи бізнесу назавжди увійшов у портфелі великих та середніх компаній.

Щодо клієнтських сервісів, то бізнес активно вкладався весь рік у розвиток віддалених форматів обслуговування клієнтів, тепер споживач може все зробити сам, без особистого звернення до компанії, використовуючи свій смартфон. Відповідно бізнес активно займається розробкою програмних продуктів. Значну частину розробки додатків бізнес веде у хмарному середовищі, що дозволяє настроювати та швидко масштабувати обчислювальні ресурси під різні завдання, зокрема самостійно, використовуючи автоматизовані сервіси управління хмарою. У 2021 році також можна спостерігати посилення тренду з перенесення даних до захищених хмар

відповідно до законодавства щодо захисту персональних даних, тому що все більше людей користуються онлайн-сервісами і зберігати персональні дані на власних серверах стає для бізнесу дорого, крім регуляторних ризиків, тому стала популярна послуга Object Storage as a service – об'єктне зберігання для різноформатних даних підписки, це один з найдешевших варіантів, який найчастіше використовується для зберігання важких медіафайлів.

Відповідно, попит на хмарні обчислювальні ресурси в 2021 році продовжує зростати, хоча, як і в 2020 році, зростання попиту може компенсуватися оптимізацією ресурсів з боку галузей, які найбільше страждають у локдаун, хоча тренд на такий, що від обчислювальних потужностей, особливо в захищеному сегмент компанії відмовляються в останню чергу, тому що це одна з нечисленних можливостей зберегти бізнес на плаву в поточних кризових умовах. Тому можна спрогнозувати, що незважаючи на те, що пандемічні ризики зберігаються, локдауни хоч і рідко, але трапляються, хмарний ринок продовжить своє зростання в межах 20-30%, а сегмент захищених хмар зростатиме вищими темпами – на рівні 30-40% . Отже, бізнес продовжує шукати найефективніші IT-рішення та впроваджувати цифрові інструменти, які допомагають залишатися продуктивними в умовах пандемічних обмежень. Всі ці рішення та інструменти вимагають все більше хмарних ресурсів, які дозволяють скоротити витрати бізнесу на організацію операційних процесів та взаємодію з клієнтами, насамперед, за рахунок переведення витрат з капітальних до операційних, а також за рахунок відмови від дорогого обладнання, яке треба амортизувати та вчасно оновлювати.

# 3 ПРАКТИЧНА РЕАЛІЗАЦІЯ

## 3.1 Режим введення знань експерта

Реалізований сервіс поділено на дві частини. Частина, яка призначена для введення знань (Expert view) дозволяє переглядати існуючі ЕС зареєстрованого користувача і створювати нові. Створення ЕС складається з кількох етапів. На першому етапі створення експерт повинен вибрати назву ЕС і визначити набір ключових слів, за якими користувачі зможуть її знайти. Назва ЕС має бути унікальною і повинна відображати предметну область, в якій компетентний експерт, що її створює. На додаток до ключових слів, які були задані на цьому етапі, будуть додані нові на наступному. Початковий етап створення ЕС представлений на рисунку 3.1.

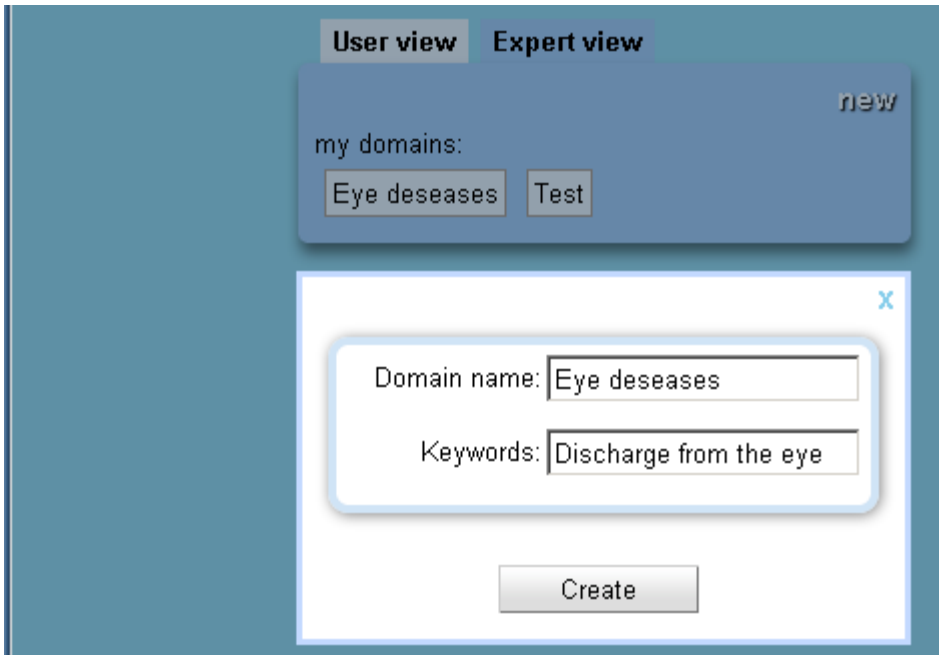


Рисунок 3.1 – Створення ЕС

Наступний етап є основним. Тут відбувається створення всіх правил

виведення ЕС за допомогою інтуїтивно зрозумілого інтерфейсу:

– вікно з підказкою (tooltip) містить перелік тих фактів, які необхідні спрацьовування поточного правила, тобто. ліва частина продукційного правила. Вікно з підказкою необхідно експерту для того, щоб він міг зрозуміти, який йому необхідно зробити висновок, маючи наведений список фактів;

– вікно попереднього перегляду. Показує те, як виглядатиме поточна ітерація консультації під час режиму консультації користувачів з урахуванням html-форматування тексту;

– панель управління експерта – комплексний віджет, що складається із вікна введення та вікна вибору варіантів відповіді. Вікно введення дозволяє вводити експерту рядок консультації, застосовуючи до неї HTML-форматування тексту. Вікно вибору варіантів відповіді визначає, як користувач може відповісти на поточний рядок консультації експерта. Це може бути стандартний варіант відповіді типу так-ні, або експерт може придумати довільну кількість своїх варіантів (рисунок 3.2).

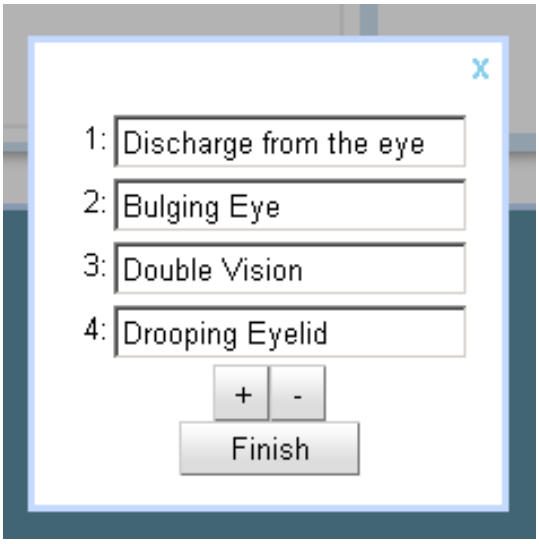


Рисунок 3.2 – Довільні варіанти відповідей

У разі вибору стандартного варіанту, користувач побачить пару кнопок з відповідною назвою, у другому випадку буде представлений список, що

випадає. На панелі також розміщені кнопки Preview для попереднього перегляду і Next для переходу до складання наступного правила. За допомогою кнопки Back можна в будь-який момент повернутися до попереднього правила,

Процес створення правил з допомогою цього інтерфейсу представлений на рисунку 3.3.

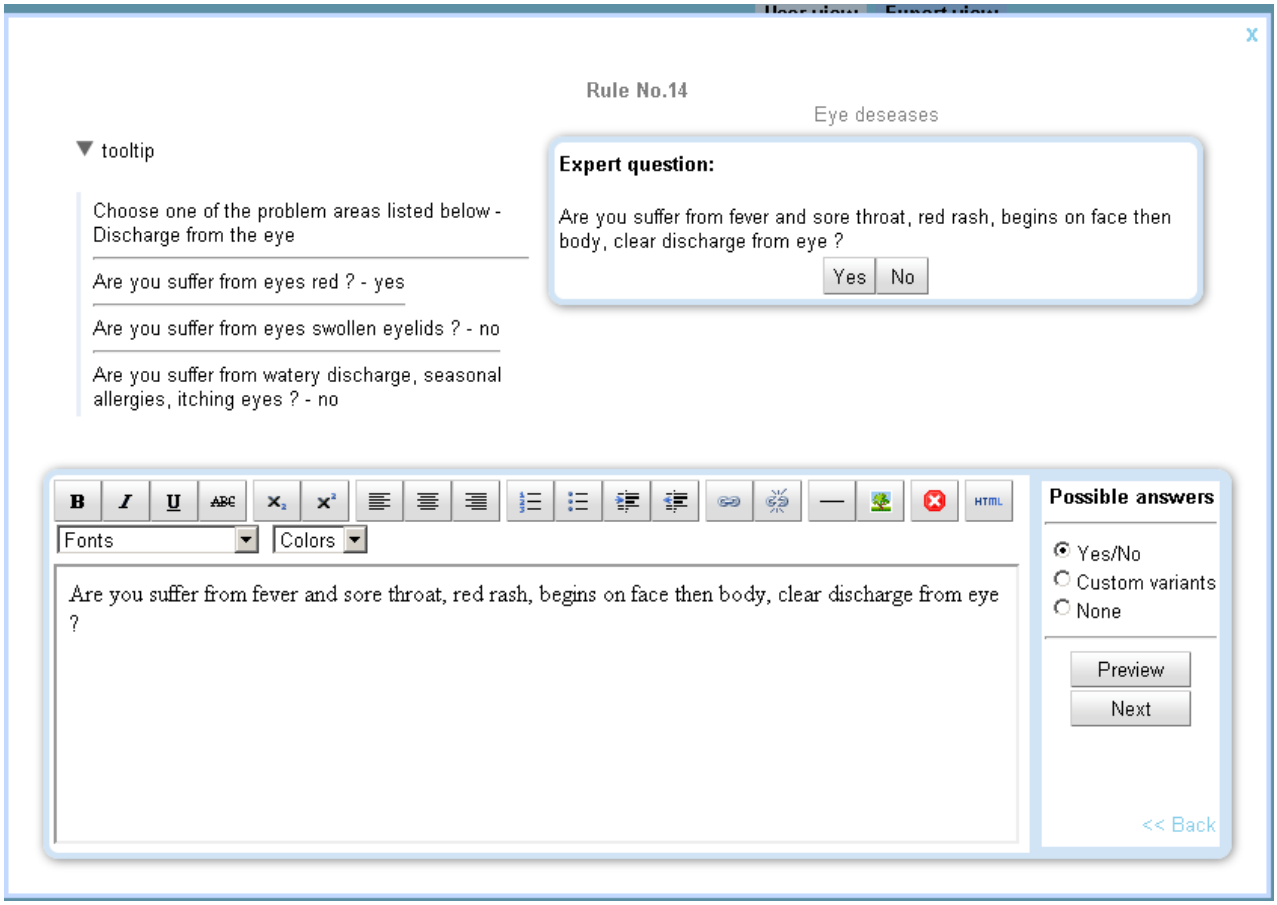


Рисунок 3.3 – Створення правил ЕС

Цю концепцію створення правил експертної системи можна як динамічного створення вузлів дерева з довільним числом дочірніх елементів. Користувачі розпочинають консультацію з кореня дерева та рухаються до листя. На рисунку 3.4 представлено дерево, де вершинами q є рядки консультації, а дугами v – варіанти відповіді користувача. Фактами є шаблони з двома слотами: рядок консультації та відповідь. Правила представляють

структуру, що містить у лівій частині список таких фактів, а у правій виведення нового рядка консультації з можливими варіантами відповідей. Якщо правило не виводить можливих варіантів відповідей, то його можна подати у вигляді листа дерева. Таке правило додатково виводить шаблон зі слотами like та dislike для визначення релевантності рішення експерта. Для цього обчислюється коефіцієнт,

Правило, що є корінь дерева, вимагає наявності у лівій частині будь-яких фактів, тобто. воно спрацьовує автоматично зі стартом експертної системи. Далі правила можуть бути складені за двома стратегіями: в ширину та в глибину. У першому випадку спочатку заповнюються всі правила поточного рівня, потім йде перехід на наступний рівень. У другому випадку спочатку завжди заповнюється найглибший вузол на поточній позиції. Упорядкування, таким чином, переходить на найглибший рівень дерева, на якому вузли не мають дочірніх елементів. На рисунку 3.4 вузли пронумеровані так, якби вони заповнювалися завширшки.

Рядки консультації в різних правилах на одному рівні можуть бути однаковими. Якщо експерт не знає, який рядок консультації скласти з певним набором фактів, то він може залишити його порожнім, а поточне правило буде позначене для подальшого редагування.

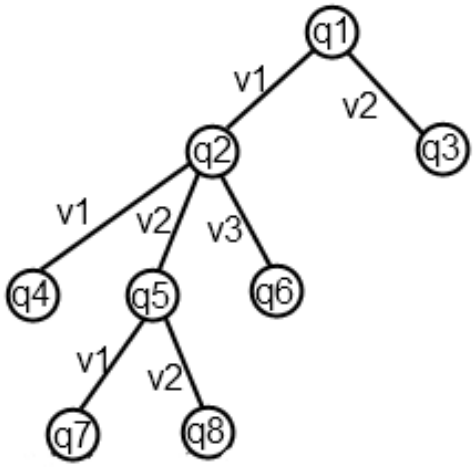


Рисунок 3.4 – Подання правил ЕС

Для налагодження сервісу використовувалася частина експертної системи з визначення очних захворювань. У процесі створення експертної системи формується код, що має синтаксис мови JESS. Текст отриманої програми представлено у додатку А.

## 3.2 Режим консультації користувача

Частина сервісу, яка призначена для консультації користувачів (User view), дозволяє знаходити та використовувати існуючі в базі експертні системи. Для цього потрібно ввести в поле для пошуку ключові слова, які описують його проблему. Якщо базі існують експертні системи, задовольняють запиту користувача, вони будуть виведені. Після вибору експертної системи починається процес консультації, який представлений на рисунку 3.5. Кожному користувачеві індивідуально виділяється робоча пам'ять для внесення фактів.

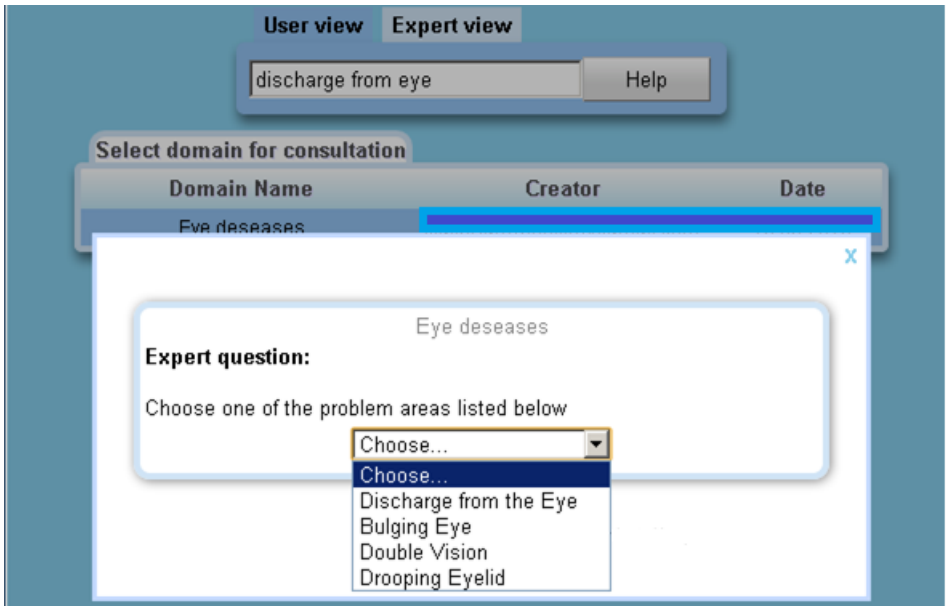


Рисунок 3.5 – Консультація користувачів

Після досягнення остаточного виведення (аркуша дерева), у процесі консультації, користувач може оцінити рішення експерта. Якщо оцінка цього

рішення вже була незадовільною, або ця частина експертної системи була пропущена експертом, додатково виведеться дві опції редагування (рисунок 3.6). Перша опція дозволяє відредагувати виведення рядка консультації поточного правила, а після редагування коефіцієнт оцінювання буде обнулено. Друга опція, якщо поточний користувач зареєстрований і має свої експертні системи, дозволяє доповнити цю частину своєї експертної системи. У цьому випадку поточне нерелевантне правило буде замінено першим (кореневим) правилом ЕС, що додається. Так само, для правильного спрацьовування правил, до їх лівої частини будуть додані необхідні факти. Коефіцієнти, що показують релевантність правил ЕС, що додається, збережуться. Зрозуміло, дані не є обов'язковими і можуть бути використані користувачем тільки в тому випадку, якщо він вважає себе досить компетентним з цього питання.


Рисунок 3.6 – Оцінювання рішень експерта

## 3.3 Адміністрування сервісу

Консоль адміністрування GAE є окремим сервісом, що дозволяє стежити за сервісами, що розробляються.

У головному меню представлений графік залежностей різних параметрів (запити, помилки, ресурси процесора тощо) від часу (рисунок 3.7).

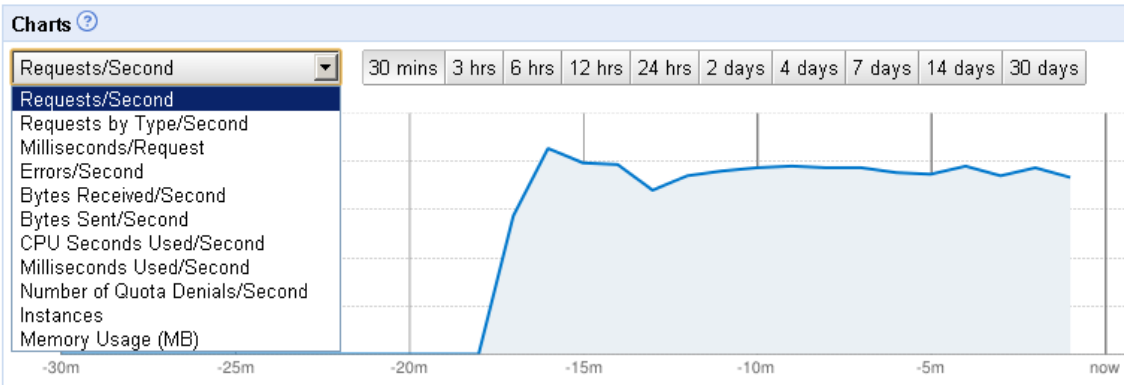


Рисунок 3.7 – Моніторинг ресурсів

Часовий інтервал можна змінювати в діапазоні від останніх 30 хвилин роботи сервісу до 30 днів. Нижче знаходиться статус квот, що виділяються для сервісу (рисунок 3.8). Квоти обнуляються кожні 24 години. У підрозділі Versions можна керувати версіями сервісу.

Instances

| App Engine Release | Total number of instances | Average QPS* | Average Latency* |
|---|---|---|---|

**Billing Status:** Free - Settings

| Resource | | Usage |
|---|---|---|
| Frontend Instance Hours | 0% | 0.00 of 28.00 Instance Hours |
| Backend Instance Hours | 0% | 0.00 of 9.00 Instance Hours |
| Datastore Stored Data | 0% | 0.00 of 1.00 GBytes |
| Logs Stored Data | 0% | 0.00 of 1.00 GBytes |
| Task Queue Stored Task Bytes | 0% | 0.00 of 0.49 GBytes |
| Blobstore Stored Data | 0% | 0.00 of 5.00 GBytes |
| Code and Static File Storage | 3% | 0.03 of 1.00 GBytes |
| Datastore Write Operations | 0% | 0.00 of 0.05 Million Ops |
| Datastore Read Operations | 0% | 0.00 of 0.05 Million Ops |
| Datastore Small Operations | 0% | 0.00 of 0.05 Million Ops |
| Outgoing Bandwidth | 0% | 0.00 of 1.00 GBytes |
| Recipients Emailed | 0% | 0 of 100 |
| Stanzas Sent | 0% | 0 of 10,000 |
| Channels Created | 0% | 0 of 100 |
| Logs Read Bandwidth | 0% | 0.00 of 0.10 GBytes |

Рисунок 3.8 – Таблиця квот

У розділі Data можна керувати даними, внесеними до стандартної бази даних GAE Datastore або Blob, за допомогою підрозділів Datastore Viewer та

Blob Viewer відповідно (рисунок 3.9). В браузері можна робити запити до БД, за допомогою синтаксису GQL, редагувати і видаляти отримані результати. Також можна створювати нові сутності різних типів.

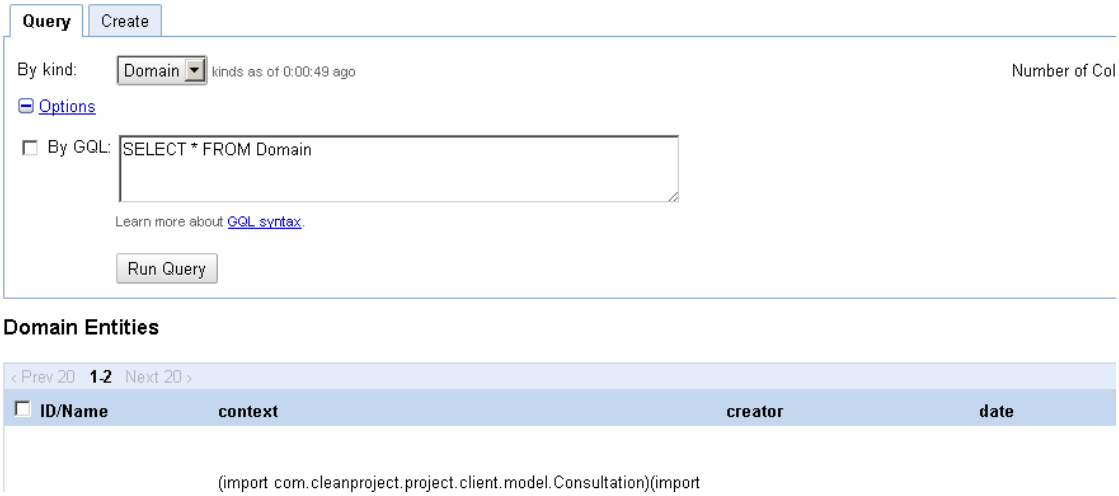


Рисунок 3.9 – Оглядач Datastore

У розділі адміністрування можна переглянути ідентифікатори сервісу, виставити термін закінчення cookie, тип аутентифікації та виділити необхідні обчислювальні ресурси для сервісу (рисунок 3.10).

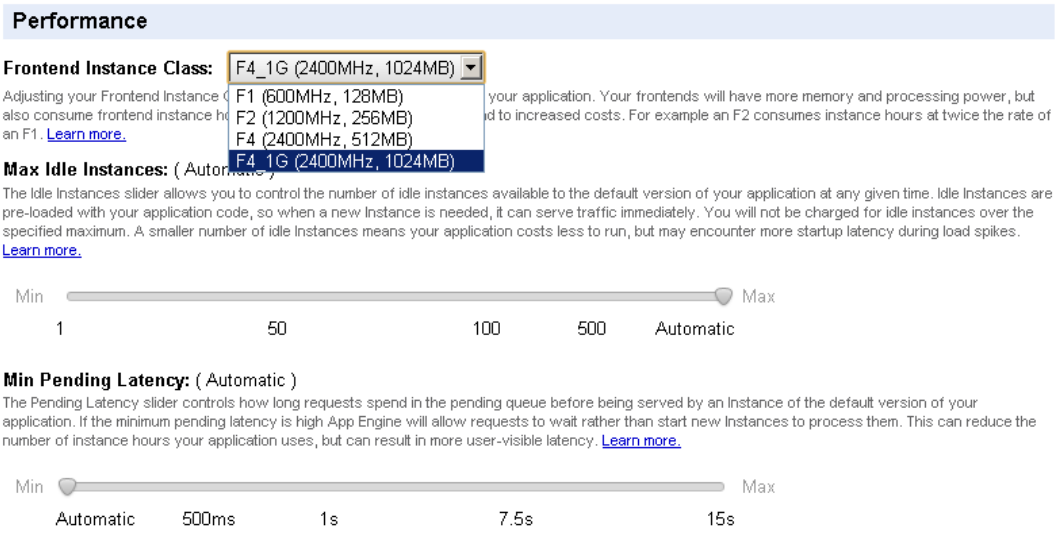


Рисунок 3.10 – Виділення ресурсів

# ВИСНОВКИ

Завдяки технології Cloud Computing, IT стає послугою. Це вигідно для всіх. Компаніям, які можуть значно скоротити витрати на IT, сконцентрувавши ресурси, що вивільнилися, на розвитку власного бізнесу, і частково перевести капітальні витрати в операційні (крім того, компанії малого та середнього бізнесу отримують додаткові переваги від використання недоступних раніше корпоративних IT-технологій). Ринку IT та телеком, сприяючи виникненню нових бізнес-моделей та стимулюючи появу стартапів. І, як наслідок, державі та суспільству: підвищується рівень інформатизації країни, та розвиток економіки отримує додатковий стимул. Незважаючи на те, що на тему ВІД є велика кількість інформації як правдивої, так і хибної, на ринку спостерігається значна обізнаність щодо PaaS. Більшість компаній впроваджують ОП з метою запуску більш інноваційних додатків, орієнтованих на клієнта, таких як «програмне забезпечення як послуга» (SaaS), мобільні додатки, інтернет-торгівля та соціальні служби, до того ж роблячи це на низці різних мов. У той час як зниження витрат є одним з приводом для впровадження PaaS, бажання покращити операційну готовність, функціональність і час безвідмовної роботи набувають все більшого значення.

Розроблений, за допомогою ВІД, сервіс дозволить створювати, зберігати, розширювати та використовувати ЕС. Сервіс орієнтований на методику Web 2.0, тому зі зростанням користувачів сервісу, якість та кількість розроблених експертних систем зростатиме. Це призведе до розширення баз знань, і користувачі зможуть вирішувати все складніші проблеми з великою ймовірністю успіху.

# ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Sarna D. Implementing and Developing Cloud Computing Applications. Auerbach Publications. 2010. 316 p.

2. Borko F. Handbook of Cloud Computing. Armando Escalante Editors. 2010. 665 p.

3. Hurwitz J., Kaufman M., Halper Dr. F. Cloud Computing for dummies. Wiley Publishing, Inc. 2010. 339 p.

4. Velte A.T., Velte T.J., Elsenpeter R. Cloud Computing на практичний інструмент. The McGraw-Hill company. 2010. 353 p.

5. Windows Azure: Documentation. URL: http://www.windowsazure.com (дата звернення 10.10.2021).

6. Amazon Web Services: Documentation. URL: http://aws.amazon.com (дата звернення 10.10.2021).

7. Google App Engine: Documentation. URL: developers.google.com (дата звернення 10.10.2021).

8. Force.com: Documentation. URL: http://www.force.com (дата звернення 10.10.2021).

9. Heroku: Documentation. URL: http://www.heroku.com (дата звернення 13.10.2021).

10. Hivext: Documentation. URL: http://ide.hivext.com (дата звернення 13.10.2021).

11. JESS: Documentation. URL: http://www.JESSrules.com (дата звернення 13.10.2021).

12. Експертні системи. URL: http://khpi-iip.mipk.kharkiv.edu (дата звернення 15.10.2021).

13. Джарратано Д., Райлі Г. Експертні системи: принципи розробки та програмування, 4-те видання. М.: ТОВ «І.Д. Вільямс», 2007. 1152 с.

14. Люггер Д. Ф. Штучний інтелект: стратегії та методи вирішення

складних проблем, 4-те видання. М.: ТОВ «І.Д. Вільямс», 2003. 864 с.

15. Friedman-Hill E. JESS in Action: Java Rule-Based Systems (In Action Series). Manning Publ. Co., 2003. 443 p.

16. Jackson K., Philpott D. GovCloud: Cloud Compuiting for Buisiness of Government. Government Training Inc., 2011. 542 p.

17. Машнін Т. Web-сервіси Java. БХВ-Петербург., 2012. 560 с.

18. Ньюкомер Е. Веб-сервіси. XML, WSDL, SOAP та UDDI. Для професіоналів. Видавничий дім «Пітер», 2011. 256 с.

19. Леонов В. Google Docs, Windows Live та інші хмарні технології. Ексмо-Прес., 2012. 304с.

20. Фінгар П. DOT.CLOUD. Хмарні обчислення – бізнес-платформа XXI століття. Аквамаринова Книга., 2011. 256 с.

21. Клементьєв І. П., Устинов В. А. Введення в Хмарні обчислення. УГУ., 2009. 233с.

22. Розподілені хмарні (Cloud) обчислення. URL: https://sites.google.com (дата звернення 17.10.2021).