DOI: https://doi.org/10.36910/6775-2524-0560-2024-54-10
УДК 004.65+004.51+001
Д'яков Ян Русланович, аспірант
https://orcid.org/0000-0002-6265-7329
Харківський національний університет радіоелектроніки, м. Харків, Україна

ДОСЛІДЖЕННЯ ФУНДАМЕНТАЛЬНИХ ВЛАСТИВОСТЕЙ ОНТОЛОГІЧНИХ БАЗ ЗНАНЬ

Д'яков Я.Р. Дослідження фундаментальних властивостей онтологічних баз знань. Проаналізовано поняття онтології, її характерні властивості, причини виникнення та механізми реалізації. Встановлено зв'язок між онтологіями та базами знань. Уведено та розкрито поняття онтологічної бази знань. Коротко оглянуто бібліотеку інженерних онтології, базу знань OpenCyc та її предка – Cyc. На прикладі OpenCyc розглянуто низку фундаментальних властивостей онтологічних баз знань, зокрема структурну еквівалентність складу онтологій та фундаментальних елементів баз знань, компроміс між надмірністю знань та вузькою спеціалізацією онтологій, необхідність впровадження мови взаємодії та багато ін. Констатовано потребу в функціях різної арності, можливостей до оптимізації запитів та команд задля поглиблення процесу взаємодії користувача із системою. Доведено необхідність у структурах більш високого, абстрактного рівня та модулярності бази знань задля підвищення спроможностей до підтримки, спрощення розробки та роботи зі знаннями. Розглянуто проблеми врегулювання узгодженості у базах знань, зокрема конфліктних випадків між позачасовими істинами та істинами, залежними від часу; приведено переваги використання онтологічних баз даних у таких випадках. Доведено необхідність розробки й підтримки якісних інтерфейсів взаємодії, зокрема командного та візуально-інтерактивного (графічного). Розкрито нюанси ведення документації онтологічних баз знань. Приведено підсумки супутнього дослідження системи OpenCyc. Означено перспективні напрямки подальших досліджень, можливі сфери для впровадження.
Ключові слова: бази знань, комп'ютер, інформатика, відношення, онтологія, елемент.

Diakov Y. Research of the fundamental properties of ontological knowledge bases. The concept of ontology, its characteristic properties, causes of emergence and implementation mechanisms are analyzed. The connection between ontologies and knowledge bases is established. The concept of an ontological knowledge base is introduced and disclosed. The library of engineering ontologies, the OpenCyc knowledge base and its ancestor – Cyc – are briefly reviewed. On the example of OpenCyc, a number of fundamental properties of ontological knowledge bases are considered. In particular, there are structural equivalence of the composition of ontologies and fundamental elements of knowledge bases, a compromise between redundancy of knowledge and narrow specialization of ontologies, the necessity to implement an interaction language etc. The necessity for functions of various degrees, opportunities to optimize requests and commands to deepen the process of user interaction with the system was identified. The necessity for structures of a higher, abstract level and modularity of the knowledge base has been proved in order to increase the capabilities to support the software product, simplify its development and work with knowledge in it. Problems of consistency regulation in knowledge bases are considered: in particular, conflict cases between timeless truths and time-dependent truths; advantages of using ontological databases in such cases are given. The necessity of developing and maintaining high-quality interaction interfaces, in particular command and visual-interactive (graphical) interfaces, has been proven. The nuances of maintaining documentation of ontological knowledge bases are disclosed. The results of the accompanying study of the OpenCyc system are given. Prospective directions for further research, possible areas for implementation are identified.
Keywords: Knowledge base, computer, informatics, relation, ontology, element.

Постановка наукової проблеми. У світлі сучасних тенденцій щодо збільшення обсягів даних, накопичення дублікатів, упередженої інформації – постає питання не лише поглиблення навичок критичного мислення та аналізу даних, але й розробки різноманітних систем централізації знань. Наразі роль таких систем виконують численні загальні (Wikipedia, Britannica, WikiSpooks, Prometheus, Дія.Освіта) та вузькоспеціалізовані (Coursera, Pluralsight, Microsoft Learn, Vue Mastery) освітні платформи, закриті корпоративні експертні системи та бази знань тощо.

Водночас актуалізується проблема визначення істини у базах знань. Ігноруючи філософський аспект поданого питання, доцільно визнати за недопустиме існування протиріч знань усередині системи. Не менш важливим є видалення (або недопущення уведення) дублікатів, врахування упереджених даних на основі конфліктів з наявними знаннями тощо. Подібне є можливим за рахунок впровадження інтегрованої системи визначення істинності та підтримки узгодженості знань.

З численних підходів до імплементації баз знань доцільно звернути увагу на такі, що використовують у своїй основі онтології, тобто опис деякої предметної області з чітким визначенням можливих класів сутностей та зв'язків між ними. Системи подібного принципу роботи мають специфічні характеристики, які доцільно розглянути задля пришвидшеної ідентифікації правил уведення, максимізації ефективності роботи користувача із системою, спрощення розробки та підтримки тощо.

Аналіз останніх досліджень і публікацій. Нюансами конструювання, теоретичного та вузькоспеціалізованого використання експертних систем та баз знань займались Джексон П., Таунсенд

К., Голбрайт Л., Гаврилова Т., Хорошевський В. [1]. Визначенням та прикладними застосуваннями онтологій, систем на базі онтологій цікавились Грубер Т., Лапшин В., Добров Б., Ланде Д. [2]. Питаннями розвитку та життєвого циклу, використання баз знань та баз даних займались Ліпмен А., Азарія А., Демченко Ю., Резніченко В., Верес О. [3]. Розробка та впровадження прикладних інформаційних систем розглядали у своїх наукових працях Фаулер М., Девіс В., Когаловський М., Фролов Е. [4, p. 12-13].

Метою статті є виведення та дослідження на основі системи OpenCyc основних властивостей онтологічних баз знань, що унікально ідентифікують такі системи, та максимізація яких дозволяє підвищити їхню якість на усіх етапах розробки.

Виклад основного матеріалу дослідження. В інженерно-інформаційному аспекті термін "онтологія" вперше був розкритий американським вченим й винахідником Томасом Грубером. Працюючи над проблемами повторного використання елементів баз знань, їхньої акумуляції, узгодженості та синхронізації між різними програмними системи, Грубер сформував принципи декларативної формалізації знань за класами та зв'язками, що об'єднують позитивні сторони об'єктно-орієнтованого програмування та реляційної системи баз даних. Теорія такої формалізації передбачала використання канонічної форми або мови предикатів, та форми онтологій як сукупності термів (класи та відношення між ними, константи) і зв'язків між ними. Групи таких онтологій – бібліотеки – доцільно було використовувати у базах знань як низькорівневий фундамент [2].

У розумінні Грубера поняття "інженерна онтологія" або просто "онтологія" можна розкрити як формальний опис предметної області, що є прийнятним для вирішення деякої задачі. Як наслідок, онтології неможливі без цілі їхнього існування, інакше вони втрачають свій сенс. Крім того, онтології вміщують лише ті поняття й зв'язки, які потрібні для вирішення поставленої задачі; інші елементи предметної області ігноруються на етапі уведення знань. Іншими словами, онтологія не може бути надмірною. Такий підхід є популярним у сучасній теорії програмування під абревіатурою "YAGNI" (аббр. від You Aren't Gonna Need It – Тобі це не знадобиться)[5].

Бази знань у своїй основі є вузькоспеціалізованими інформаційними середовищами, що мають чітко встановлені правила виведення й представляють дані з деякої предметної області, що є потрібними для користувача задля вирішення тих чи інших проблем. Онтологічною у такому випадку доцільно назвати таку базу знань, якщо за результатами розробки вона підпорядковується елементній базі деякої онтології задля уніфікації структуризації власних даних, та / або знання усередині неї можна представити у якості онтології.

Властивості онтологічних баз знань, як власне й баз знань узагалі, доцільно розглянути на прикладі відкритої системи OpenCyc. Система-предок – Cyc – наразі є комбінацією бази даних та засобів штучного інтелекту, що використовує "кодифікований людський здоровий глузд" для когнітивної обробки даних. Система здатна логічно міркувати, використовуючи динамічні дані реального світу, трансформувати, розширювати та спрощувати всі складні людські робочі навантаження [6]. З іншого боку, OpenCyc є спрощеною альтернативою Cyc під ліцензіями відкритого програмного забезпечення задля ознайомлення з функціоналом Cyc й просування комерційної версії як більш повної та функціональної. Означену систему з деякими проблемними аспектами можна назвати онтологічною базою знань. Такі аспекти буде розглянуто нижче.

Остання версія OpenCyc нараховує близько 239 тисяч термінів, серед яких місцям відводиться близько 19 тисяч, організаціям – 26 тисяч, індивідам – 12 тисяч тощо [7].

Так як елементна база OpenCyc не представляє інтересу у контексті розгляданого питання, доцільно зосередити увагу на типових властивостях онтологічних баз даних, що складають список найважливіших акцентів у ході розробки таких систем, та акцентуація на які дозволяє суттєво підвищити якість продукту. Втім, варто зазначити, що означена типові елементи OpenCyc – константи, колекції, предикати, речення тощо – представляють онтологічну основу системи, виконуючи вузькоспеціалізоване завдання організації структурної узгодженості знань. Можливість представлення елементної бази таких програмних продуктів у вигляді онтології є характерною властивістю онтологічних баз знань. Організація такого представлення на етапі проектування системи суттєво спрощує усі подальші етапи її життєвого циклу; публікація його такого представлення – спрощує роботу з системою для користувачів.

У ситуації надмірної кількості даних та необхідності їхньої структуризації та аналізу задля отримання корисних знань, стають актуальними принципи роботи з великими даними (з англ. Big

Data). З іншого боку, специфіка онтології передбачає вузьку спеціалізацію предметної області, тобто деякий зріз даних, корисний для виконання поставленого завдання. Необхідність компромісу між масштабом розгляданої предметної області та надмірністю онтологічного представлення результату призводить до більш поміркованого виділення обсягу знань. Це знижує ефективність методів роботи з великими даними, проте відповідає специфіці онтології.

У випадку OpenCyc, базу знань, що вміщує дані про рослини, тварини, людей, організації, політичні партії тощо – доцільно визнати надмірною; завдання онтологічного представлення таких знань – закріплення здорового глузду стосовно об'єктивної реальності – є занадто загальним.

Незалежно від того, як база знань представляє користувачу інтерфейс взаємодії, усередині кожної такої бази знань працює деяка мова запитів, оператори котрої уніфікують як уведення, так й виведення даних. Типовим прикладом такої мови є SQL (аббр. від Structured Query Language – Мова Структурованих Запитів) та її численні модифікації тощо. Залежно від проектної документації бази знань, користувачу може надаватись командний інтерфейс для прямого уведення команд та запитів; графічний інтерфейс керування системою, що конструює команди та запити на основі взаємодії користувача з графічними компонентами; гібридний інтерфейс, що є графічним, але вміщує поля для уведення команд та запитів користувачем. Подібні мови працюють досить ефективно на площині структурованих даних, що підтримуються онтологічними базами знань. OpenCyc не є виключенням: уведення змінних, речень, конструювання запитів та команд підпорядковується єдиному набору правил, що складає внутрішню мову взаємодії зі знаннями.

Супутньою властивістю подібних мов є можливість використання тих чи інших предикатів (відношень) як у запитах, так й у командах. У командах вони слугують інструментами зв'язування інших елементів баз знань; у запитах – елементами конструювання результату на площині наявних даних. Так, предикат "#$siblings" (з англ. Брати й сестри) середовища OpenCyc у командах дозволяє зв'язувати константи, що є людьми, відношенням сімейної близькості рівня "брати", "сестри" чи "брат та сестра". У запитах той ж предикат із двома шуканими змінними дозволяє вивести усі комбінації персон означеної сімейної близькості, що наявні у базі знань; із однією шуканою змінною – всіх братів та сестер позначеної персони. Означений підхід використання предикатів у базах знань є їхньою типовою властивістю.

Також наявна тенденція стосовно оптимізації запитів та команд. Так, сучасні бази даних (PostgreSQL, MySQL тощо) мають спеціальні інструменти для виконання запитів зі збором статистичних даних стосовно швидкості виконання, кількості низькорівневих команд, сформованих ядром аналізу запитів та багато іншого [3][8][9]. Активне використання поданих інструментів дозволяє порівнювати команди та запити за часом виконань, кількістю підкоманд та, як результат, підбирати оптимальніші алгоритми уведення та виведення знань, пришвидшувати їх тощо. Втім, варто також зазначити, що означена оптимізація має бути врівноважена зі зручністю читання та, як наслідок, подальшої підтримки означених запитів та команд користувачами та командою-розробником.

Знання у системі OpenCyc об'єднані й поділені на модулі або так звані мікротеорії, що мають унікальну назву (зазвичай вона є ідентифікатором тої чи іншої предметної області) та вміщують типові для означеної предметної області константи, функції, твердження тощо. З-поміж інших мікротеорію у системі можна ідентифікувати за постфіксом "Mt".

У якості прикладів доцільно представити та коротко описати наступні мікротеорії OpenCyc:

– biologyMt. Мікротеорія з біології, що вміщує представників рослинного та тваринного світу, їхні колекції; твердження стосовно кількості ніг, рук, типового зросту та ваги; функції, що встановлюють структурні відношення як-от "домен – відділ", "клас – підклас", "сімейство – рід", приналежність тої чи іншої тварини або рослини до структурної одиниці тощо;

– microbiologyMt. Мікротеорія з мікробіології, що наповнена даними стосовно одно- та багатоклітинних організмів, вірусів та паразитів; їхні типові характеристики та взаємовідносини одне між одним; способи харчування, розмноження тощо;

– textMicrotheory. Текстова мікротеорія, що складається з типових текстових функцій-трансформаторів, колекцій мов та діалектів, типових літературних представників тих чи інших мов та ін.

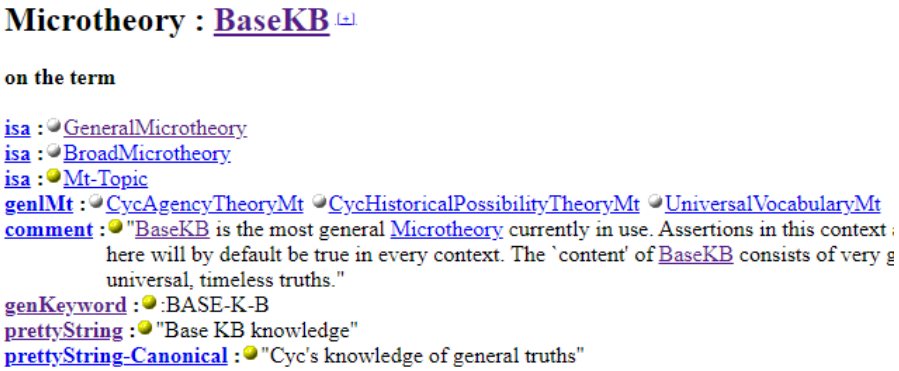
Впровадження механізмів структуризації інформації за модулями (мікротеоріями) є типовим випадком групування елементів, що підвищує продуктивність роботи з великими обсягами даних. Зокрема покращується їхня підтримка різними командами інженерів знань, спрощується потенційна

заміна однієї мікротеорії іншою, визначення області видимості та приналежності для нових знань тощо.

Наслідком виокремлення таких будівельних блоків баз знань є необхідність у ієрархії абстракцій вищого рівня, зокрема генералізованих мікротеорій, що б виступали батьківськими по відношенню до інших, більш вузькоспеціалізованих мікротеорій та вміщували узагальнені елементи. У контексті середовища OpenCyc це твердження, обов'язкові для виконання, та константи й функції, що необхідно використовувати задля наповнення й редагування мікротеорій, генерації команд та запитів тощо.

Прикладом генералізованої мікротеорії OpenCyc є "#$BaseKB". Твердження, яке є істинним у ній, за замовчуванням буде істинним для будь-якої іншої мікротеорії. Вміст "BaseKB" складається з дуже загальних тверджень, які можуть бути використані в більшості або в усіх додатках OpenCyc, а також найфундаментальніших тверджень, які система використовує у виведенні інформації й всіх універсальних (позачасових) істинах.

За допомогою інтерактивної системи взаємодії із OpenCyc доцільно коротко розглянути довідкову сторінку мікротеорії "BaseKB", представлену на рис. 1.


Microtheory : BaseKB

on the term

isa : GeneralMicrotheory
isa : BroadMicrotheory
isa : Mt-Topic
genlMt : CycAgencyTheoryMt CycHistoricalPossibilityTheoryMt UniversalVocabularyMt
comment : "BaseKB is the most general Microtheory currently in use. Assertions in this context here will by default be true in every context. The `content' of BaseKB consists of very g universal, timeless truths."
genKeyword : :BASE-K-B
prettyString : "Base KB knowledge"
prettyString-Canonical : "Cyc's knowledge of general truths"

Рис. 1 – Довідкове вікно мікротеорії "BaseKB" інтерактивної системи взаємодії із системою OpenCyc

З рис. 1 можна побачити, що означена мікротеорія є:

– елементом колекції загальних мікротеорій (англ. General Microtheory), так як у ній присутні генералізовані аксіоми, прийнятні для численних цілей;

– елементом колекції великих мікротеорій (англ. Broad Microtheory). Концепція великої мікротеорії у середовищі OpenCyc передбачає, що її твердження не можуть в повному обсязі бути виведеними тими чи іншими засобами запитів; для виведення потребується додаткова фільтрація;

– елементом колекції тем або топіків "Mt-Topic", що є додатковим переліком абстракції вищого рівня для генералізованої структуризації інформації у системі OpenCyc. Такими топіками виступають зокрема такі як "#$Expectation" (з англ. Очікування), "#$Obligation" (з англ. Зобов'язання), "#$Estimate" (з англ. Оцінка) тощо;

– зв'язаною відношенням відкриття "#$genlMt" з такими мікротеоріями, як "#$CycAgencyTheoryMt" (з англ. Теорія агентів), "#$CycHistoricalPossibilityTheoryMt" (з англ. Теорія історичної можливості) та "#$UniversalVocabularyMt" (з англ. Універсальний словник). Їхній розгляд виходить за межі поточного дослідження.

Отже, середовище OpenCyc вміщує численні високорівневі структурні компоненти (мікротеорії, топіки, колекції) різного сутнісного складу. Наявність таких компонентів є фундаментальною властивістю онтологічних баз знань.

Супутньою властивістю тут також виступає замкнутість ланцюжків компонентів, тобто повнота системи у графовій репрезентації. Так, фундаментальна мікротеорія "BaseKB" є елементом колекцій, які у своїй основі – колекцій як таких – підпорядковуються твердженням мікротеорії "BaseKB" та використовують її функції та константи. Так, елементи у онтологічній базі знань (як власне й у онтологіях) представляють сенс не лише самі по собі, але й у повноті своїх відношень з іншими елементами.

У поданому контексті також актуалізується питання строгості врегулювання узгодженості знань. Окрім випадків, де узгодженість підтримується зовнішніми системами, бази знань, зокрема й онтологічні, можна поділити за характером врегулювання узгодженості на строгі та нестрогі. Строге врегулювання передбачає неможливість уведення знань, які конфліктують із вже наявними; іншими словами, перевірка відбувається до безпосередньої операції вставки. Нестроге або ще ліниве врегулювання дозволяє уведення конфліктних знань, але маркує їх як конфліктні задля подальшого вирішення командними або автоматизованими засобами тої чи іншої бази знань. Інша реалізація нестрогого врегулювання ігнорує аспект маркування, але передбачає можливість використання так званих різонерів (з англ. Risoner, від Reason – Причина), що перевіряють необхідний користувачу сектор бази знань на узгодженість. Як результат, користувачу надаються варіанти вирішення віднайдених конфліктних ситуацій; альтернативно, різонер може самостійно вирішити ті чи інші конфлікти.

Чітке позиціювання за означеним питанням є важливою властивістю баз знань, що координують узгодженість власних структурних елементів на предмет взаємної істинності. Онтологічність у такому випадку виступає перевагою, адже усталеність структури корінних елементів та знань спрощує означений процес на усіх етапах його проведення.

У поданому контексті система OpenCyc має строгий механізм врегулювання узгодженості знань; вона не дає можливості для конфліктних знань бути уведеними у склад власних структур. Так забезпечується низька толерантність системи до конфліктів знань. З іншого боку, неможливо "відтермінувати" момент вирішення таких конфліктів на майбутнє. Втім, OpenCyc має супутню проблему, яку доцільно розглянути у контексті узгодженості знань.

Так, на рис. 2 зображено графічне вікно системи OpenCyc стосовно 43-го президента США, Дж. Буша, що перебуває у базі знань у якості константи. За допомогою викладеного переліку позначено, елементом яких колекцій є та не є ця константа; які предикати стосовно цієї константи визначено тощо.

Рис. 2 – Інформація стосовно 43-го президента США Дж. Буша

За приведеними даними, Дж. Буш є президентом США – на це вказує предикат "#$isa" з другим операндом-колекцією "#$UnitedStatesPresident" (з англ. Президент США). В тому ж списку визначено, що Дж. Буш не є президентом США – на це вказує подібний предикат з префіксом логічної інверсії "#$not". Таким чином, система OpenCyc вміщує як залежні від часу істини (на поточний момент означена персона вже не є президентом США), так й позачасові істини (означена персона позиціонується як така, що коли-небудь була президентом США), що є проблемою узгодженості знань. Механізм вирішення питання узгодженості залежних від часу істин, та позачасових істин є типовою властивістю онтологічних баз знань.

OpenCyc вирішує подану проблему за допомогою контекстних змінних, які вводяться разом із запитом (у поточному випадку це предикати часу або часового інтервалу). Таке рішення притаманне для баз знань, що орієнтовані на більш повне висвітлення предметної області, висвітлення супутніх предметних областей. Недоліком означеного механізму є необхідність не лише постійного уведення контекстних змінних у запити, але й існування таких змінних у тілі бази знань заради підтримки

узгодженості. У такому випадку доцільним є висвітлення поданих змінних у результатах запитів до бази знань – чого не було зроблено у випадку графічного інтерфейсу середовища OpenCyc, відповідно до рис. 2.

Іншим варіантом вирішення проблеми є повернення до основ онтологій. Так як онтологія є вузькоспеціалізованою та створюється для вирішення поставленого завдання, у випадку виникнення істин, залежних від часу, доцільно зафіксувати на глобальному рівні час або часовий проміжок, відносно якого онтологія є істинною. Аналогічна метода може бути використана у контексті бази знань, що ідентифікує її як онтологічну. Якщо виникає необхідність розширення такої бази знань на більш значний часовий проміжок – доцільно створити нову базу знань, зокрема й методом часткового копіювання з попередньої версії. Таким чином онтологічна база знань є так званим знімком (англ. Snapshot) деякого часового інтервалу; ризики неузгодженості позачасових та залежних від часу істин відсутні.

Як вже вказувалось, база знань повинна мати інструменти виведення, тобто механізми отримання знань з системи у вигляді, зручному для користувача.

Система правил виведення є доцільною в онтологічній базі знань зокрема через можливість автоматизації виведення тверджень з вже наявних та отримуваних даних. Така система може бути побудована із використанням правил логічного виведення (дедукція, індукція, абдукція, гібриди), дерев прийняття рішень, нечітких методів, інструментів штучного інтелекту (нейронні мережі, класифікатори) тощо [10][11]. У випадку з OpenCyc, специфіка команд вказує на логічність виведення знань (є підтримка булевих значень, логічні оператори, квантори), можливість проходження як низхідною ієрархією компонентів, так й висхідною – що говорить про індуктивно-дедуктивний характер системи. Підтримку інших принципів побудови у системі OpenCyc не передбачено. Втім, враховуючи наявні інструменти, отримувані знання є досить структурованими для подальшого використання більш високорівневими системи на основі нечітких методів, засобів штучного інтелекту тощо.

Користувач може взаємодіяти із середовищем OpenCyc за допомогою командного рядка операційної системи Windows; уводити команди, запити, перевіряти наявність тих чи інших елементів (рис. 3) тощо. Суттєвою перевагою консольного режиму тут можна навести високий рівень зворотної сумісності: подібним інтерфейсом можна скористатись у майбутньому без ризиків невірної візуалізації, застаріння графічних бібліотек тощо. Базовий командний інструментарій операційних систем залишається порівняно незмінним.

```
CYC(6): #$Dog
[Time: 0.0 secs]
#$Dog
CYC(7): #$DogeCoin

Error: "DogeCoin" is not the name of a constant.
```

Рис. 3 – Перевірка наявності константи у системі за допомогою консолі

Особливості командних інструментів мають відповідати стандартам та тенденціям сучасних концепцій побудови інтерфейсів взаємодії. Так, внормовану систему взаємодії за допомогою командного рядку пропонує програмний менеджер контейнерів Docker, інструмент для тестування навантаженням програмних систем K6, середовища інтерактивних мов програмування Python, SmallTalk тощо [12][13][14].

Альтернативним, більш зручним для користувача є візуально-інтерактивний режим, що втілюється за рахунок системи вікон, кнопок, полів уведення інформації, стилізації елементів тощо. Наразі у інформаційному просторі наявні численні інструменти побудови графічних інтерфейсів на різноманітних операційних системах [15]. Використовуючи командний режим взаємодії із системою у якості фундаменту, користувачі мають можливість створити та публікувати у інтернет-мережі свої графічні інтерфейси (у такому випадку також доцільно враховувати інтелектуальні права компанії-розробника продукту, ознайомитись з принципами роботи цифрових ліцензій тощо [16]). Розробка інтерфейсу командою-розробником бази знань може підвищити популярність створюваного продукту на його первинних стадіях впровадження; розробка ж альтернативних інтерфейсів користувачами призводить до конкуренції й стимулює до ітеративного підвищення якості продукту.

На жаль, система OpenCyc не є такою, що відповідає тенденціям часу стосовно проектування як командного, так й візуально-інтерактивного інтерфейсів взаємодії із користувачем. Наявний командний інтерфейс є складним до опанування й не відповідає за функціональним наповненням (є більш повним) за інтерфейс візуально-інтерактивний ("OpenCyc KB Browser"), що втратив свою актуальність, має проблеми з безпечністю використання тощо. Максимізація зусиль з розробки та підтримки інтерфейсів взаємодії дозволяє спростити роботу користувачів із системою, пришвидшити їхнє навчання та отримання цінного досвіду тощо.

Фундаментальною для кожної програмної системи є її документація, що пришвидшує навчання користувачів, слугує основою для уведення до команди розробки нових розробників, тестерів, проектних менеджерів тощо. У контексті онтологічних баз знань, такій документація доцільно підпорядкуватись деяким специфічним рекомендаціям, а саме:

– доцільно не лише спростити, зробити більш прозорою таку документацію, але й додати інструкцію до використання. У такій інструкції необхідно розкрити центральні елементи візуально-інтерактивного та командного режимів взаємодії, патерни взаємодії із системою (типові можливості користувача, алгоритми їхнього впровадження, відповіді системи) тощо;

– мають бути розкриті центральні характеристики використовуваної онтологічної елементної бази системи; тут доцільно використовувати таблиці, графи, блок-схеми, інтерактивні посилання та інші сучасні інструменти графічної репрезентації задля зв'язування інформації у максимально повну картину.

Інформаційні ресурси OpenCyc є застарілими та не підтримуються; документація доступна у обмеженому вигляді на ресурсах архівації старих інтернет-сторінок. Підтримка актуальної документації до будь-якого програмного продукту, зокрема й баз знань, є життєво важливим для центральної аудиторії такої системи, команд розробників, менеджерів, акціонерів та ін.

Висновки та перспективи подальших досліджень. Концепція онтологічних баз знань є важливою у контексті мінімізації дублікатів знань, суб'єктивних (упереджених) представлень щодо предметних областей, надмірного емоційного забарвлення інформації, засмічення знань елементами супутніх предметних областей тощо.

Віднайдені та проаналізовані фундаментальні особливості – підтримка узгодженості знань, мови уведення та виведення знань, модульність, специфіка документації тощо – є важливими не лише для подальшої актуалізації концепту онтологічних баз знань, але й для узгодження проектної документації, розробки, тестування та підтримки таких систем. Їхнє можливе впровадження у сферу освіти, заміни корпоративних експертних систем, конкуренція із засобами штучного інтелекту доречно дослідити докладніше у наступних статтях.

Як приклад бази знань, OpenCyc є досить потужною системою з великою кількістю вже наявних знань, засобами для їхнього уведення та виведення, абстрактними конструкціями для підтримки узгодженості знань тощо. З іншого боку, невідповідність властивостям онтологічних баз знань у випадку такої системи призвела до ряду недоліків, серед яких нечітка застаріла документація, не орієнтовані на широкий загал користувачів механізми уведення та виведення знань, відсутність підтримки компанією-засновником, розмитість принципів позачасових принципів та супутні логічні хиби тощо. Уведення чіткої онтології та підтримка віднайдених характеристик онтологічних баз знань дозволила б підвищити якість роботи системи, популярність серед користувачів, орієнтованих на бази знань тощо.

Список бібліографічного опису

1. Galbraith, L. Al-Najjar, M. Babu, A. Expert systems in engineering. *IEEE Aerospace and Electronic Systems Magazine*, vol. 3, 2, 1988. pp. 12-14. URL: https://doi.org/10.1109/62.848.

2. Gruber, T. The Role of Common Ontology in Achieving Sharable, Reusable Knowledge Bases. Knowledge Systems Laboratory. 1991. URL: https://www.cin.ufpe.br/~mtcfa/files/10.1.1.35.1743.pdf.

3. Резніченко, В. 60 років базам даних. Проблеми програмування. 2021, № 3, с. 40-71. URL: https://dx.doi.org/10.15407/pp2021.03.040.

4. Fowler, M. & Rice, D & Foemmel, M. Patterns of Enterprise Application Architecture. Addison-Wesley Professional, 2002. p. 560. URL: https://dl.ebooksworld.ir/motoman/Patterns%20of%20Enterprise%20Application%20Architecture.pdf.

5. Gupta, G., Sachdeva, M. How can you use the YAGNI principle to avoid over-engineering software? 2023. URL: https://www.linkedin.com/advice/1/how-can-you-use-yagni-principle-avoid-over-engineering-ny3ge.

6. Cyc | The Next Generation of Enterprise AI. 2023. URL: https://cyc.com.

7. GitHub. OpenCyc. 2023. URL: https://github.com/asanchez75/opencyc.

8. Stonebraker, M., Rowe, L., Hirohama, M. The implementation of POSTGRES. Making Databases Work: the Pragmatic Wisdom of Michael Stonebraker. 2018. p. 519-559. URL: http://dx.doi.org/10.1145/3226595.3226639.

9. Satoto, K., Isnanto, R., Kridalukmana, R., Martono, K. Optimizing MySQL database system on information systems research, publications and community service. *3rd International Conference on Information Technology, Computer, and Electrical Engineering (ICITACEE)*, Semarang, Indonesia, 2016. pp. 1-5. URL: https://doi.org/10.1109/ICITACEE.2016.7892476.

10. Hyun Moon, J., Sun Kang, C. Application of fuzzy decision making method to the evaluation of spent fuel storage options. *Progress in Nuclear Energy*, vol. 39, № 3-4, 2001. pp. 345-351. URL: https://doi.org/10.1016/S0149-1970(01)00019-1.

11. Dastres, R., Soori, M. Artificial Neural Network Systems. *International Journal of Imaging and Robotics (IJIR)*, vol. 21 (2), 2021. p. 13-25. URL: https://hal.science/hal-03349542.

12. Miglani, S. Develop and Test Kubernetes Applications Locally with Docker Desktop: A Guide for Data Scientists. 2023. URL: https://medium.com/cloud-native-daily/develop-and-test-kubernetes-applications-locally-with-docker-desktop-a-guide-for-data-scientists-8626cd8c463c.

13. Running k6 | Grafana k6 documentation. 2023. p. 4. URL: https://grafana.com/docs/k6/latest/get-started/running-k6.

14. Python: An Ecosystem for Scientific Computing. 2011. p. 9. URL: https://osf.io/atr27/download.

15. Kulsreshtha, A. Cross-Platform App Development with .NET MAUI. 2023. p. 5. URL: https://www.linkedin.com/pulse/cross-platform-app-development-net-maui-arpit-kulsreshtha?trk=public_post_main-feed-card_feed-article-content.

16. Thompson, C., Jena, R. Digital licensing [software reuse]. *Internet Computing, IEEE*. 9, 2005. pp. 85-88. URL: http://dx.doi.org/10.1109/MIC.2005.77.

References

1. Galbraith, L. & Al-Najjar, M. & Babu, A. (1988). Expert systems in engineering. IEEE Aerospace and Electronic Systems Magazine, vol. 3, no. 2, pp. 12-14. https://doi.org/10.1109/62.848.

2. Gruber, T. (1991). The Role of Common Ontology in Achieving Sharable, Reusable Knowledge Bases. Knowledge Systems Laboratory. https://www.cin.ufpe.br/~mtcfa/files/10.1.1.35.1743.pdf.

3. Reznichenko, V. (2021). 60 years of databases. Prombles in programming, vol. 3, pp. 40-71. https://dx.doi.org/10.15407/pp2021.03.040.

4. Fowler, M. & Rice, D & Foemmel, M. (2002). Patterns of Enterprise Application Architecture. Addison-Wesley Professional. https://dl.ebooksworld.ir/motoman/Patterns%20of%20Enterprise%20Application%20Architecture.pdf.

5. Gupta, G. & Sachdeva, M. (2023). How can you use the YAGNI principle to avoid over-engineering software? https://www.linkedin.com/advice/1/how-can-you-use-yagni-principle-avoid-over-engineering-ny3ge.

6. Cyc (2023). Cyc | The Next Generation of Enterprise AI. https://cyc.com.

7. GitHub (2023). OpenCyc. https://github.com/asanchez75/opencyc.

8. Stonebraker, M. & Rowe, L. & Hirohama, M. (2018). The implementation of POSTGRES. Making Databases Work: the Pragmatic Wisdom of Michael Stonebraker. pp. 519-559. http://dx.doi.org/10.1145/3226595.3226639.

9. Satoto, K. & Isnanto, R. & Kridalukmana, R. & Martono, K. (2016). Optimizing MySQL database system on information systems research, publications and community service. 3rd International Conference on Information Technology, Computer, and Electrical Engineering (ICITACEE), Semarang, Indonesia, pp. 1-5. https://doi.org/10.1109/ICITACEE.2016.7892476.

10. Hyun Moon, J. & Sun Kang, C. (2001). Application of fuzzy decision making method to the evaluation of spent fuel storage options. Progress in Nuclear Energy, 39, 3–4, pp. 345-351. https://doi.org/10.1016/S0149-1970(01)00019-1.

11. Dastres, R. & Soori, M. (2021). Artificial Neural Network Systems. International Journal of Imaging and Robotics (IJIR), vol. 21 (2), pp. 13-25. https://hal.science/hal-03349542.

12. Miglani, S. (2023). Develop and Test Kubernetes Applications Locally with Docker Desktop: A Guide for Data Scientists. https://medium.com/cloud-native-daily/develop-and-test-kubernetes-applications-locally-with-docker-desktop-a-guide-for-data-scientists-8626cd8c463c.

13. Grafana Labs (2023). Running k6 | Grafana k6 documentation. https://grafana.com/docs/k6/latest/get-started/running-k6.

14. Scientific Python (2011). Python: An Ecosystem for Scientific Computing. https://osf.io/atr27/download.

15. Kulsreshtha, A. (2023). Cross-Platform App Development with .NET MAUI. pp. 5. https://www.linkedin.com/pulse/cross-platform-app-development-net-maui-arpit-kulsreshtha?trk=public_post_main-feed-card_feed-article-content.

16. Thompson, C. & Jena, R. (2005). Digital licensing [software reuse]. Internet Computing, IEEE. 9. pp. 85-88. http://dx.doi.org/10.1109/MIC.2005.77.