НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ “ЛЬВІВСЬКА ПОЛІТЕХНІКА”

Буров Євген Вікторович

УДК 004.652.4

МЕТОДИ ТА ЗАСОБИ ПОБУДОВИ ПРОГРАМНИХ СИСТЕМ НА ОСНОВІ ОНТОЛОГІЧНИХ МОДЕЛЕЙ ЗАДАЧ

Спеціальність 01.05.03 – математичне та програмне забезпечення обчислювальних машин і систем

Автореферат
дисертації на здобуття наукового ступеня
доктора технічних наук

Львів - 2015

Дисертацією є рукопис.

Робота виконана у Національному університеті «Львівська політехніка»

Міністерства освіти і науки України, м.Львів

Науковий консультант: доктор технічних наук, професор, професор кафедри інформаційних систем та мереж Національного університету «Львівська політехніка», лауреат державної премії України в галузі науки та техніки Пасічник Володимир Володимирович

Офіційні опоненти:

доктор технічних наук, професор,
Сидоров Микола Олександрович,
Національний авіаційний університет, м.Київ,
завідувач кафедри інженерії програмного забезпечення

доктор технічних наук, професор,
Виклюк Ярослав Ігорович,
Буковинський університет, м.Чернівці,
проректор з наукової роботи
та міжнародних відносин

доктор фізико-математичних наук, професор,
Цегелик Григорій Григорович,
Львівський національний університет імені Івана Франка, м.Львів, завідувач кафедри математичного моделювання соціально-економічних процесів

Захист відбудеться 29 квітня 2015 р. о 13 годині на засіданні спеціалізованої вченої ради Д 35.052.05 у Національному університеті “Львівська політехніка” (79013, м. Львів, вул. С. Бандери, 12).

З дисертацією можна ознайомитись у науково-технічній бібліотеці Національного університету “Львівська політехніка” (79013, м.Львів, вул. Професорська, 1).

Автореферат розіслано __ березня 2015 р.

Учений секретар
спеціалізованої вченої ради,
доктор технічних наук, професор Р. А. Бунь

ЗАГАЛЬНА ХАРАКТЕРИСТИКА РОБОТИ

Актуальність теми. Аналіз існуючих тенденцій розвитку інформаційних технологій, зокрема використання Інтернет та систем електронної комерції, систем мобільного доступу, мережецентричних систем, показує, що темпи змін та складність програмних систем продовжують зростати.

Це відповідає загальним тенденціям глобалізації у світовій економіці, необхідності швидко реагувати на зміни у бізнесовому середовищі. Підвищення мобільності бізнес-процесів на сьогодні спирається на широке використання програмних систем та технологій і в свою чергу ставить високі вимоги до якості програмних систем та їх здатності швидко та безпомилково адаптуватися до змін бізнесового середовища.

В цих умовах актуальним є розроблення нових архітектурних принципів, методів та засобів спрямованих на спрощення побудови, модифікації та адміністрування програмних систем та підвищення їх якості.

Високий рівень складності призводить до виникнення ряду недоліків та вузьких місць, якими характеризуються існуючі архітектури програмних систем та підходи до їх проектування. Зокрема:

– проектування програмної системи базується на фіксованому наборі вимог до системи, які часто визначені нечітко, в результаті чого отримують системи, які погано реагують на зміну вимог та зовнішніх чинників, які дорогі у підтримці та експлуатації і в яких нерідко трапляються збої та аварії;

– проектування найчастіше подається, як циклічне повторення таких робіт як визначення вимог, проектування архітектури, придбання обладнання, кодування програмного забезпечення, тестування, впровадження та експлуатація; реалізація такого підходу зазвичай займає багато часу і не дозволяє гнучко реагувати на зміни середовища та вимог до системи;

– в існуючих програмних системах правила та алгоритми функціонування жорстко зашиті у коді; будь-яка зміна алгоритму та правил вимагає перекодування та тестування, що займає багато часу та дорого;

– існує відрив системи бізнес-процесів від програмної системи, яка забезпечує її функціонування; у результаті зміна у бізнес-процесах не може бути оперативно відображена у програмній системі.

На вирішення науково-технічної проблеми побудови програмних систем, здатних до адаптації в умовах змін середовища функціонування, спрямовані наукові дослідження за декількома теоретичними та технологічними напрямами.

Зокрема, в галузі технологій розробки програмного забезпечення проводяться дослідження методів гнучкої розробки (Agile programming). Ці методи є ефективними при створенні відносно простих систем, в них складно спланувати терміни реалізації визначених функціональних вимог.

З метою врахування специфіки бізнес-процесів в архітектуру програмної системи вводиться рівень бізнес-логіки (Microsoft .NET WPF application

framework, Domain driven design). Водночас, таке запровадження зазвичай носить статичний характер та враховує лише стан бізнес-правил на певний момент часу, реалізація ж бізнес-правил як частини загального коду системи суттєво ускладнює її модифікацію.

На вирішення зазначених вище недоліків спрямовані модельно-орієнтовані підходи, в яких виділено формалізацію алгоритму роботи програмної системи від системи опрацювання цього алгоритму. Такий підхід запропоновано Меллором та Шлаером, які спочатку будують модель програмної системи, що потім компілюється у код. Водночас, практичне застосування цього підходу показало, що його складність є співмірною зі складністю створення програмної системи традиційним способом. У роботах Росса, Хея (Ross, Hay) та ряду інших авторів запропоновано декларативний опис та збереження бізнес-правил в окремій спеціалізованій системі. При цьому частиною програмної системи є так звана машина правил для контролю виконання бізнес-правил. Такий підхід дозволяє гнучко налаштовувати систему при зміні бізнес-правил. Його обмеженість обумовлена неможливістю подати усі особливості бізнес-середовища у вигляді правил, відсутності декларативного опису середовища.

Перспективним напрямом у вирішенні проблеми адаптації програмної системи до змін середовища є використання підходів інтелектуальних систем, зокрема онтологічного моделювання (Грубер, Калініченко). На відміну від класичних модельних підходів, орієнтованих на компільовані моделі чи опрацювання правил, у підході онтологічного моделювання будують формальну модель предметної області (онтологію), яка може бути повторно використана для побудови інших програмних систем для цієї ж предметної області.

Для побудови програмних систем з використанням онтологічного підходу доцільно обрати об’єктом розгляду задачу, як найменшу ідентифіковану та необхідну частину будь-якої роботи. *Поняття задачі* визначається в літературі як усвідомлена проблемна ситуація з визначеними умовами (даними) та метою.

Аналіз існуючих підходів до побудови та модифікації програмних систем показує, що незважаючи на значний обсяг зусиль, спрямованих на підвищення здатності програмної системи адаптуватися до зміни середовища функціонування, ця проблема загалом ще не вирішена.

Отже, існує наукова-прикладна проблема підвищення якості розроблення та функціонування програмних систем, в аспектах пов’язаних зі складністю їх побудови, та адаптації до актуального стану предметної області. Зазначена проблема відображає фундаментальне протиріччя між змінним характером середовища, в якому функціонує програмна система, та вимогами швидкої та коректної адаптації складної програмної системи до цих змін, на подолання якого і спрямоване це дисертаційне дослідження. У дисертаційній роботі подано вирішення вказаної науково-прикладної проблеми на базі запропонованої оригінальної теоретично-методологічної концепції та методів

побудови та модифікації програмних систем шляхом використання онтологічних моделей задач.

Зв'язок роботи з науковими програмами, планами, темами. Дисертаційна робота виконана в межах наукового напряму «Нові комп'ютерні засоби та технології інформатизації суспільства» визначеного пріоритетним у переліку актуальних проблем Міністерством освіти і науки України, концепції програми інформатизації НАН України, визначеної пріоритетним напрямом, згідно розпорядження № 146 від 27.02.2004 р. та за тематикою наукових досліджень кафедри інформаційних систем та мереж Національного університету «Львівська політехніка» , зокрема в рамках держбюджетної НДР «Моделі та методи побудови інтелектуальних систем бізнес-аналітики на основі інтегрованих інформаційних ресурсів», номер державної реєстрації 0110U001102. Автором розроблено архітектуру, принципи функціонування, методи подання та опрацювання знань, інструментальні засоби моделювання інтелектуальної системи бізнес-аналітики на основі онтологічних моделей задач.

Мета і задачі дослідження. Метою роботи є розв'язання науково-прикладної проблеми підвищення якості програмних систем завдяки спрощенню процесів їх побудови, модифікації, адміністрування та підтримки шляхом використання онтологічних моделей задач.

Для досягнення поставленої мети в дисертаційній роботі вирішено такі задачі.

1. Аналіз існуючих підходів до побудови та архітектур програмних систем з метою визначення вузьких місць та резервів для підвищення їх якості та спрощення завдань створення, модифікації та супроводу таких систем.
2. Розвиток теоретичних засад подання та опрацювання знань про предметну область у програмних системах на основі онтологічних моделей задач з метою розробки та теоретичного обґрунтування принципів подання та опрацювання знань у таких системах.
3. Розроблення архітектурних принципів та методів побудови програмної системи на основі онтологічних моделей задач, розробка ефективних методів взаємодії моделей у системі та визначення способів реалізації адаптивних можливостей у такій системі.
4. Розроблення методів вирішення задач з використанням онтологічних моделей на основі розроблених теоретичних засад та архітектурних принципів та їх практичне застосування з метою визначення практичної цінності отриманих результатів.
5. Розробка та дослідження інструментальних засобів для побудови та моделювання програмних систем на основі онтологічних моделей задач.
6. Дослідження переваг застосування онтологічних моделей для побудови програмних систем, розроблення методів визначення їх впливу на характеристики якості програмного забезпечення та формування відповідних оціночних співвідношень

Об'єктом дослідження є процеси побудови та функціонування програмних систем.

Предметом дослідження є архітектури, методи побудови та функціонування програмних систем на основі онтологічних моделей задач.

Методи дослідження. Для вирішення задачі аналізу підходів до побудови програмних систем використано: загальну теорію систем, системний аналіз, методи аналізу та моделювання бізнес-процесів, теорію еволюції програмних систем. У процесі розроблення теоретичних засад подання та опрацювання знань у програмних системах на основі онтологічних моделей задач використано теоретико-множинні підходи, теорію типів, алгебру систем, теорію побудови інтелектуальних систем на базі онтологій, теорію схем. При розробленні архітектурних принципів побудови програмних систем на основі онтологічних моделей використано апарат модельно-орієнтованих підходів до побудови програмних систем, загальні архітектурні принципи побудови програмних систем, методи сервісно-орієнтованих систем. Під час розроблення методів вирішення задач з використанням онтологічних моделей застосовано методи концептуального моделювання та систем підтримки прийняття рішень. Для розроблення та дослідження інструментальних засобів побудови та моделювання програмних систем на основі онтологічних моделей задач використано xml- технології для специфікації подання моделей. У процесі дослідження ефективності застосування онтологічних моделей задач застосовано методології визначення параметрів якості програмного забезпечення, подані у міжнародних стандартах ISO 9126, ISO/IEC 2510, а також методи оцінювання тривалості виконання завдань проекту.

Наукова новизна одержаних результатів. Узагальнено наукова новизна результатів полягає у розробці нових принципів та методів побудови програмних систем, які базуються на використанні онтологічних моделей задач, що зменшує складність їх побудови та адаптації до змін предметної області. Одержано такі нові наукові результати:

- вперше розроблено механізм взаємодії моделей задач, який забезпечує вирішення задач пошуку релевантних моделей, вибору моделі та її ініціалізації; цей механізм уможливлює вирішення складних задач у системі, багатоваріантність методів рішення задачі та повторне використання знань у системі;
- вперше розроблено означення контексту та метод його визначення, в якому на відміну від існуючих, використовується база знань, який не залежить від задач, що використовують контекст, а тільки від стану бази знань;
- вперше розроблено методологію побудови онтології предметної області на основі аналізу комплексу задач, що вирішують у цій області; на відміну від існуючих, ця методологія визначає ітеративний процес розбудови онтології та забезпечує кращу її обґрунтованість, спрощує процес створення та модифікації онтології, а також створює можливість її валідації через комплекс використаних моделей;
- вперше запропоновано та обгрунтовано принцип побудови модельно-орієнтованих програмних систем з використанням інтерпретації моделей

задач, який на відміну від xUML, дає змогу спростити процес побудови та модифікації системи;

- вперше розроблено новий, базований на онтологічних моделях задач, метод керування доступом до ресурсів програмної системи, який порівняно з методом RBAC та ABAC не має ефекту накопичення прав доступу з часом, надає права у контексті виконуваних виробничих завдань, спрощує процес надання та вилучення прав;
- вперше проаналізовано ефективність застосування онтологічних моделей задач для побудови програмних систем та розроблено відповідні оціночні співвідношення, які дають змогу оцінити вплив на характеристики якості програмного забезпечення;
- отримала подальший розвиток концепція побудови баз знань, яка полягає у включенні у базу знань окрім онтології та інформаційної бази ще моделей задач; така концепція, на відміну від існуючих, поєднує переваги онтологічного та процедурного підходів, та орієнтована на підтримку вирішення задач з використанням бази знань і дозволяє формалізувати, накопичувати та повторно використовувати досвід щодо способів виконання задач.

Практичне значення одержаних результатів. Практичне значення дисертаційної роботи полягає у:

- дослідженні та розробці способів застосування моделей задач для моделювання бізнес-процесів та систем бізнес-аналітики підприємств, визначенні типів моделей, їх структури та способів використання;
- розробці онтології галузі тестування програмного забезпечення на основі аналізу процесів та завдань тестування;
- розробці способів подання онтологічних моделей бізнес-процесів та методів їх використання для автоматизації тестування програмних продуктів;
- розробці способів створення на основі онтологій задач туристичних сервісів, які використовують для прийняття рішень інформацію з контексту поїздки;
- розробці структури, принципів функціонування інструментальних засобів для створення та моделювання програмних систем на базі онтологічних моделей.

Результати роботи впроваджено у навчальному процесі та використовувалися зокрема, для компаній «Флоор Бест», «Анат», а також в інформаційно-програмних системах ряду підприємств.

Особистий внесок здобувача. Усі наукові результати дисертаційної роботи отримані автором особисто. У друкованих працях, опублікованих у співавторстві, автору належать: застосування методології побудови онтології на основі аналізу задач до предметної галузі тестування [3]; постановка задачі, метод керування доступом на базі моделей, структура моделі, приклади використання моделей [14]; методи відображення подій у моделях проектування [16]; способи подання та параметричні моделі сервісів розподілених інформаційних систем [17]; постановка задачі, класифікація моделей, принципи

функціювання системи підтримки договірного процесу [24]; постановка задачі, форма подання ситуаційної моделі, принципи використання та опрацювання моделей у туристичному сервісі [25]; постановка задачі, форма подання та способи опрацювання моделей у туристичному порталі [26]; алгоритм консенсусного оцінювання у вигляді моделей знань [27]; архітектура та методи використання онтологічних моделей задач для побудови програмних систем [28]; принципи та методи застосування онтологічних моделей в системах підтримки прийняття рішень [29].

Апробація результатів дисертації. Основні результати дисертаційних досліджень неодноразово доповідалися на міжнародних наукових конференціях, зокрема на конференціях «Комп'ютерні науки та інформаційні технології» ITCE (м.Вінниця, 2010), «Інтелектуальні системи прийняття рішень і проблеми обчислювального інтелекту» ISDMCI (м. Євпаторія, 2011), «Системний аналіз та інформаційні технології» SAIT (м. Київ, 2011, 2014), «Комп'ютерні науки та інформаційні технології» CSIT (м. Львів, 2009-2012), «Якість технологій – якість життя» (Польща, м.Перемишль, 2014).

Публікації. За результатами дисертаційних досліджень опубліковано 36 наукових праць, зокрема одноосібну монографію «Концептуальне моделювання інтелектуальних програмних систем» [1], 26 статей у фахових наукових виданнях [2-27], 9 публікацій у матеріалах міжнародних конференцій [28-36]. Серед цих наукових праць 5 статей опубліковано у виданнях, індексованих у міжнародних базах даних [2-6], 26 одноосібних публікацій.

Структура та обсяг роботи. Дисертаційна робота складається зі вступу, шести розділів, висновків, списку використаних джерел з 184 найменувань на 18 сторінках та 3 додатків на 23 сторінках. Загальний обсяг дисертацій 328 сторінок, з них 283 сторінок основного тексту, 75 рисунків, 17 таблиць.

ОСНОВНИЙ ЗМІСТ РОБОТИ

У вступі обґрунтовано актуальність теми, сформульовано мету та основні завдання досліджень, показано зв'язок із науковими програмами, планами, темами, сформульовано наукову новизну.

У першому розділі проведено аналіз існуючих тенденцій розвитку архітектур, технологій та процесів побудови програмних систем у контексті виявлення можливостей спрощення їх побудови та збільшення адаптаційних можливостей до змін вимог. Це особливо стосується таких програмних систем, вимоги до яких змінюються з часом і таких, що за своєю структурою та функціями повинні постійно відповідати стану свого середовища функціонування (клас E за Леманом). Відомо, що еволюція програмних систем класу E призводить до збільшення складності та кількості помилок, зменшення якості, погіршення функціональної цілісності системи. Це призводить до того, що програмні системи необхідно періодично кардинально переробляти, а це в свою чергу, вимагає значних фінансових та часових витрат. Сучасні тенденції розвитку програмних систем, що відображають процеси глобалізації світової

економіки та полягають у зростанні темпів змін вимог і підвищенні ступеня інтеграції найрізноманітніших програмних систем, тільки поглиблюють зазначену вище проблему.

Аналіз можливостей програмної системи адаптуватися до змін вимог проведено базуючись на багаторівневій концептуальній моделі програмної системи (рис. 1), яка в цілому відповідає таким моделям програмних систем як (WPF application framework – Майкрософт, Domain Driven Design – Еванс) та методології загальної теорії систем (Месарович, Такахара) в застосуванні до багаторівневих моделей систем. Ця модель має три рівні: рівень бізнес-процесів, на якому відбувається моделювання бізнес-процесів та визначається бізнес-логіка системи, рівень аплікацій та сервісів, що безпосередньо реалізують головні функції системи, та рівень пристроїв, який надає базові інфраструктурні послуги зі збереження, передавання та опрацювання даних.

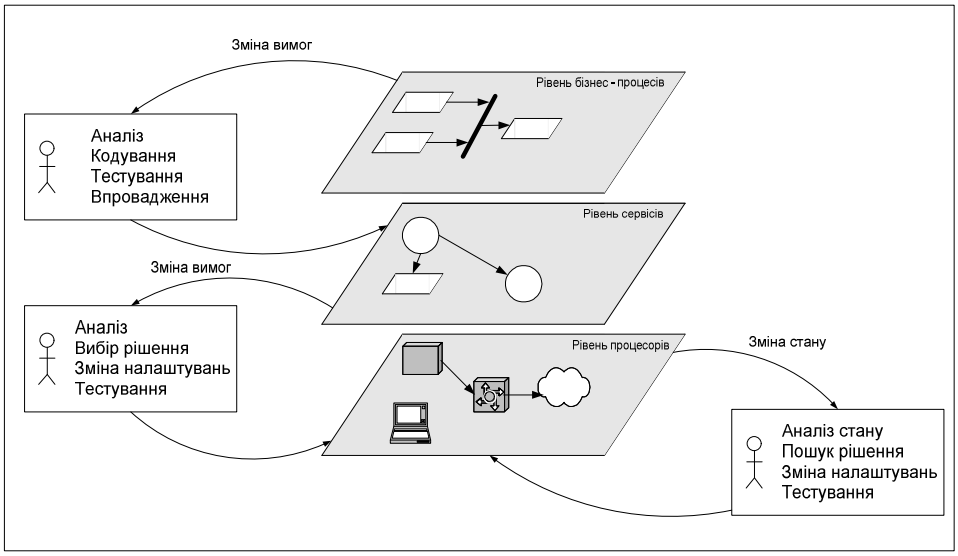


Рис. 1. Багаторівнева концептуальна модель програмної системи

Моделювання бізнес-процесів є складовою частиною багатьох методологій побудови програмних систем, зокрема RUP, об’єктно-орієнтованих методологій, що використовують бізнес-орієнтовані розширення мови UML (Еріксон та ін). Водночас, таке моделювання сьогодні використовується для формулювання статичних бізнес-вимог до програмної системи, які в подальшому реалізуються у програмному коді. Зміна бізнес-середовища призводить і до зміни бізнес-вимог і, як наслідок, до необхідності перекодування програмної системи.

Аналогічно, на рівні аплікацій та сервісів з метою зменшення складності та спрощення внесення змін у програмну систему бізнес-функції системи інкапсулюють у вигляді слабопов’язаних програмних сервісів (архітектура SOA) з визначеними та опублікованими інтерфейсами доступу. Водночас, гнучка у використанні система сервісів SOA будується на основі статичної

моделі бізнес-процесів і у випадку зміни цієї моделі вимагає перебудови як сервісів, так і їх інтерфейсів. Аналіз модельно-орієнтованих підходів до створення програмних систем MDA, xUML (Шлаер та Меллор), що базуються на побудові системи модельних специфікацій програмної системи з наступною компіляцією їх у код, показав значну складність створення та підтримки сукупності модельних специфікацій, співмірних зі складністю підтримки коду, створеного традиційним способом.

Завдання щодо зменшення складності підтримки та масштабування програмних систем на рівні пристроїв на сьогодні вирішується шляхом розробки технологій віртуалізації систем збереження та опрацювання даних. Аналіз концепцій побудови адаптивних комп'ютерних систем, зокрема «Електронного бізнесу на вимогу» (E-business on demand, IBM), «Автономних обчислень» (Autonomous computing, Hewlett-Packard), «Комунальних обчислень», "Динамічних систем» (Utility computing, Dynamic systems, Microsoft) показав, що в їх основу покладено принципи та технології інтелектуальних систем, повторне використання та накопичення знань про систему та середовище функціонування.

Аналіз існуючих архітектур програмних систем у контексті їх адаптаційних можливостей до зміни вимог показав, що класичні архітектури, зокрема клієнт-серверні та компонентні (CORBA, DCOM) спрощують адаптацію системи шляхом інкапсуляції функцій та спеціалізації компонент системи, повторного використання коду. Водночас, у таких системах немає чіткої відповідності між вимогами та компонентами системи, що ускладнює їх адаптацію. У сучасних архітектурах програмних систем, таких як Common application architecture, .NET -WPF application framework (Microsoft), Domain-driven architecture (Evans) підкреслюється важливість розуміння процесів бізнес-середовища, в якому функціонує програмна система і яке є джерелом вимог до неї. Невід'ємною складовою частиною побудови програмної системи при цьому є побудова бізнес-моделі та врахування результатів моделювання у вигляді бізнес-компонент та сервісів. Основою при побудові програмної системи архітектури SOA є формування та реалізація стратегії менеджменту бізнес-процесів з використанням програмної системи. При цьому не тільки формують модель бізнес-середовища, але й визначають у цій моделі можливі напрямки еволюції вимог та враховують їх в архітектурі програмної системи. Водночас у системах з такими архітектурами відображається стан середовища тільки на певний момент часу. Зміна середовища з часом вимагає повторного моделювання та перекодування системи за його результатами. Модель середовища відокремлена від програмної системи і не може бути використана для проведення оперативних змін. Деякі спеціалізовані програмні системи, наприклад, експертні системи, мають архітектуру, в якій модель предметної області подана як набір правил, є частиною системи та опрацьовується інтерпретатором, що виконує логічне виведення. Зміна вимог до такої системи призводить до зміни правил, що не вимагає перекодування. Недоліками систем, що базуються на правилах є відсутність цілісної моделі предметної області, і як

наслідок – складність узгодження великих наборів правил, і те, що не всі властивості предметної області можуть бути сформульовані у вигляді правил. З врахуванням цих недоліків перспективним для вирішення задачі побудови програмних систем є застосування апарату онтологічного моделювання, та онтологій як цілісних формальних моделей предметної області.

Узагальнені результати аналізу програмних архітектур у розрізі врахування вимог середовища у вигляді моделі предметної області подано у табл. 1.

Таблиця. 1. Моделювання предметної області в архітектурах програмних систем

Архітектура	Яким чином відбувається моделювання предметної області	Недоліки
Клієнт-серверні, DCOM, CORBA, Java	У явному вигляді моделювання не проводиться	Бізнес-логіка та дані неявно включені код, що ускладнює його модифікацію
NET, WPF, Common application architecture, Domain-driven architercture	Побудова окремої бізнес-моделі та на її основі – програмних бізнес-компонент та сервісів	Бізнес-компоненти відображають стан середовища в один момент часу
SOA	Побудова бізнес-моделі та аналіз стратегії і тенденцій розвитку бізнес-середовища	Складність реалізації непередбачених змін
MDA	На основі модельної специфікація предметної області будують специфікацію програмної системи, яку компілюють у код.	Значна складність розробки модельної специфікації
Архітектури, що базуються на правилах	Модель середовища, подана у вигляді правил є частиною програмної системи.	Відсутність цілісної моделі середовища, складність узгодження правил
Архітектури, що базуються на онтології	Модель предметної області подана онтологією, є цілісною.	Складність побудови та супроводу онтології.

Перспективним для вирішення проблеми збільшення гнучкості є розвиток концепцій, архітектур, методів та засобів побудови програмних систем, що базуються на принципах використання активних моделей середовища функціонування та технологій інтелектуальних систем опрацювання знань, зокрема онтологій.

У другому розділі обгрунтовано теоретичні засади подання та опрацювання знань в інтелектуальних програмних системах на базі онтологічних моделей задач. Розроблено формальну модель подання знань у системі. Формально означено поняття контексту знань та фактів для забезпечення взаємодії моделей. Обгрунтовано методологію побудови онтологій на основі онтологічних моделей задач.

В розділі показано, що значна частина розробок у галузі подання знань на сьогодні використовує декларативний підхід. Зокрема до цієї категорії відносять фреймові системи, онтологічні системи на базі OWL та ін. Перевагою декларативного підходу до подання знань є можливість створення цілісної формальної моделі предметної області, можливість використання потужних механізмів логічного виведення. До його недоліків відносять значну складність онтології та процесів менеджменту, непристосованість до подання та використання процедурних абстракцій.

Альтернативою до декларативного підходу до подання знань є процедурний підхід, зокрема SBD (Schema based design) (Pezzulo). Перевагами процедурного підходу є простота побудови та використання схем, орієнтація на вирішення задач. Недоліками процедурного підходу є складність узгодження схем, їх злабка пов'язаність, відсутність цілісності моделі предметної області, поданої набором схем.

Враховуючи взаємно доповнювальний характер декларативного (онтологічного) та процедурного підходів, у роботі запропоновано їх поєднання в єдиний підхід до подання знань у системі. При цьому загальна онтологія визначає набір сутностей, відношень та інтерпретацію (аксіоми, обмеження) для всієї предметної області. Онтологічні моделі задач використовують компоненти з загальної онтології та визначають сутності, відношення, обмеження та дії у контексті окремої задачі.

Використання онтологічних моделей задач для побудови програмних систем вимагає розробки математичного обґрунтування у вигляді відповідної формальної моделі, яка в подальшому буде використовуватися для формального подання методів опрацювання знань та системи моделювання. Для побудови формальної моделі використано підхід алгебри систем Коо, що визначає алгебраїчну систему шляхом комбінації декількох алгебраїчних доменів.

Нехай у предметній області існує n множин об'єктів: $A_1, A_2, ... A_n$. Об'єкти, що належать кожній множині класифікують як екземпляри конкретного поняття. Ці множини є множинами-носіями для n багатосортних алгебр. Окремі екземпляри, що належать цим множинам позначатимемо $a_1, ... a_n$.

Домени концептів (сутностей) E. На основі кожної множини A_i визначимо абстрактний тип даних $E_i = (Name, \Sigma, Ex)$, де *Name* – назва типу, Σ – сигнатура багатосортної алгебри, *Ex* – визначальні співвідношення типу. При цьому сигнатура містить тільки множину елементів, а не містить операцій та відношень.

Окремі типи позначатимемо E_i (довільний тип, його назва не визначена) або E_{name}, де *name* – назва типу, якщо в контексті розгляду важливо вказати на конкретний тип. Типи E_i утворюють множину алгебраїчних доменів E, які відповідають концептам (сутностям) предметної області.

Домен атрибутів At. Визначимо алгебраїчний домен атрибутів *At* на списку значень атрибутів у вигляді пар *(key, value)*. Елемент пари *key* визначає ідентифікатор атрибуту, а *value* – його значення. Для цього домену визначено операції об'єднання, підстановки, видалення, інтерпретації *{merge, substitute, delete, interp}*.

Булевий домен Cs. До булевого домену належать вирази, результат підрахунку яких належить булевій множині *{true, false}*. Операндами виразів є змінні, що належать іншим алгебраїчним доменам. Будемо інтерпретувати елементи булевого домену як обмеження *Cs*. Операціями для булевого домену є булеві операції *{and, or, negation}*, а також операція інтерпретації *interp*, яа відображає спрощення та підрахунок булевих виразів. Екземпляром булевого домену є конкретне обмеження.

Домен сутностей з атрибутами T. Визначено на множині кортежів (E_i, At_j, Cs_j^*). Для кожного *i* існує тільки один *j*, що входить в елемент цього домену: $\forall i \exists^1 j : (E_i, At_j)$. Кожне обмеження $Cs_j \in Cs_j^*$ є виразом з операндами, що належать домену At_j. Операціями над елементами цього домену є об'єднання та поділ сутностей *{merge, split}*. Операція об'єднання сутностей інтерпретується як формування нової сутності, як спільного набору властивостей та обмежень сутностей-операндів. Операція поділу сутності – зворотна до об'єднання. Екземпляром домену сутностей з атрибутами є кортеж відповідних фактів. Окремі типи з атрибутами позначатимемо як T_i (довільний тип, його назва не визначена) або T_{name}, де name – назва типу, якщо в контексті розгляду важливо вказати на конкретний тип. Змінні, що приймають значення екземплярів конкретного типу, позначимо як x_{name}, або просто *name*. Множину елементів, що мають тип *name* позначимо як S_{name} або $S(T_{name})$: $S_{name} = \{x \mid x \in T_{name}\}$.

Домен відношень Rl. Множиною – носієм цього домену виступають структури виду: $\{(T_{1,i} \times T_{2,i} \times ... \times T_{k,i}, At_i^r, Cs_i^*)\}$. Кожна структура є кортежем, що містить декартовий добуток алгебраїчних типів даних з домену *T*, тип з домену атрибутів *At* (визначає атрибути відношення) та множина обмежень *Cs*.

Кожне обмеження $Cs_i \in Cs_i^*$ є булевим виразом з операндами, що належать доменам $At_{1,i}, At_{2,i}, ..., At_{k,i}, At_i^r$.

Над відношеннями визначено операції об'єднання відношень, відокремлення, підстановка *{merge, split, substitute}*. Операції об'єднання та відокремлення відношень трактуються подібно до аналогічних операцій з домену *E*. В операції підстановки замість однієї сутності домену *T* підставляємо відношення, яке при цьому трактується як сутність з атрибутами (реіфікація).

Онтологія є кортежем, поданим доменами понять (концептів) з атрибутами, відношень та обмеженнями з булевого домену, що стосуються доменів понять та відношень:

$$On = (T, Rl, Cs^*). \quad (1)$$

Визначимо онтологію задачі On_{TS} яка є частиною загальної онтології On і містить об'єкти, необхідні для вирішення конкретної задачі $On_{TS} \subseteq On$.

Онтологічна модель задачі Md_{TS} визначається кортежем з трьох елементів: $Md_{TS} = (On_{TS}, S_{ac-TS}, S_{cs-TS})$, де On_{TS} – онтологія задачі, $S_{ac\text{-}TS}$ – множина дій, $S_{cs\text{-}TS}$ – множина додаткових обмежень.

Дія Ac є сутністю з атрибутами (алгебраїчний домен T), яка інтерпретується як команда до певного зовнішнього сервісу або команда на виконання іншої онтологічної моделі. Для команди визначені параметри, які отримують із значень атрибутів елементів, що входять у On_{TS} (або ж вони задані як константи). Виконання онтологічної моделі задачі реалізують як формування та надсилання команди на виконання визначеної у моделі дії до сервісу виконання.

У табл. 2 подані основні алгебраїчні домени, використані у формальній моделі.

Таблиця 2. Алгебраїчні домени моделі

Домен	Множина-носій	Оператори
Сутностей (n доменів) – E	$\{a_i \mid Type(a_i) = E_i\}$ факти визначених типів	
Атрибутів – At	$\{(key, value)^*\}$	{merge, substitute, delete, interp}
Булевий домен Cs	{true, false}	{and, or, negate, interp}
Сутностей з атрибутами – T	$\{(E_i, At_j, Cs_j^*)\}$	{merge, split}
Відношень	$\{(T_{1,i} \times T_{2,i} \times ... \times T_{k,i}, At_i^r, Cs_i^*)\}$	{merge, split}
Онтологія	(T, Rl, Cs^*)	{merge, split}
Онтологічних моделей задач	$\{(On_{TS}, Ac_{TS}^*, Cs_{TS}^*)\}$	

З кожною дією в онтологічній моделі асоційовано поняття мети T_{GL}, визначеної як набір станів бази фактів, кожен з який відповідає досягнутій меті. Задана функція мети, яка дозволяє визначити, чи у певному стані t'_{BFC} мета Gl є досягнута:

$$F_{gl}(t'_{BFC}) = \begin{cases} true \mid t'_{BFC} \in t_{GL}, \\ false \mid t'_{BFC} \notin t_{GL}. \end{cases} \quad (2)$$

Формальне визначення поняття контексту необхідне для забезпечення взаємодії виконувальних моделей. Моделі підчас взаємодії обмінюються відносно невеликими обсягами інформації, передаючи тільки посилання,

“опорні точки” у семантичній мережі взаємопов’язаних фактів. Усю потрібну для виконання операції інформацію модель повинна отримати самостійно, з контексту. В результаті аналізу робіт у галузі контекстно-залежного комп'ютингу показано, що існуючі визначення контексту або занадто загальні та інтуїтивні (і таким чином не можуть бути використані для вирішення практичних задач), або занадто специфічні (придатні для вирішення тільки конкретних задач).

Зазначені проблеми запропоновано вирішити шляхом дотримання таких архітектурних рішень, які з одного боку є достатньо загальними, а з іншого – придатні для вирішення конкретних задач.

1. Використати базу знань як проміжний рівень між сервісами, що добувають інформацію з зовнішнього світу та застосуванням, замість використання контексту певного реального об’єкту використовувати контекст відповідного об’єкту бази знань.

2. Запровадити семантичну інтерпретацію контексту як частини цієї бази знань, яка відображає стан предметної області.

3. Відокремити контекст від задачі, яка вирішується з застосуванням контексту. Таким чином контекст певного об'єкту визначається тільки цим об'єктом, його властивостями і зв'язками з зовнішнім середовищем.

4. Визначити, що застосування, яке використовує контекст певного об'єкту, формулює критерії релевантності інформації з контексту, стратегії пошуку релевантної інформації у контексті та спосіб використання отриманих результатів пошуку.

У роботі обгрунтовано формальне визначення контексту на рівні онтології (контекст знань) та похідний від нього контекст рівня фактів. Центральним елементом контексту на рівні знань є певний клас T^{co}. Контекстом знань нульового рівня є сам клас та порожня множина його зв’язків:

$$Con^0_{kn}(T^{co}) = (S^0_{co}(T^{co}), S^0_{co}(T^{co}_{RCL})) = (T^{co}, \varnothing). \quad (3)$$

Контекст знань першого рівня містить усі зв’язки та класи, з якими клас контексту поєднаний безпосередньо:

$$Con^1_{kn}(T^{co}) = Con^0_{kn} \bigcup (S^1_{co}(T^{co}), S^1_{co}(T^{co}_{RCL})), \quad (4)$$

де $\forall T^i \in S^1_{co}(T^{co}) \exists T^j_{RCL} : T^{co} \in T^j_{CL-RCL}, T^i \in T^j_{CL-RCL}$ та

$\forall (T^i, T^j) \in T^i_{CL-RCL} \in T^i_{RCL} \in S^1_{co}(T^{co}_{RCL}) : T^i \in S^0_{co}(T^{co}), T^j \in S^1_{co}(T^{co})$.

Аналогічно контекстом рівня k є

$$Con^k_{kn}(T^{co}) = Con^{k-1}_{kn} \bigcup (S^k_{co}(T^{co}), S^k_{co}(T^{co}_{RCL})), \quad (5)$$

де $\forall T^i \in S^k_{co}(T^{co}) \exists (T^j_{RCL}, T^l \in S^{k-1}_{co}(T^{co})) : T^i \in T^j_{CL-RCL}, T^l \in T^j_{CL-RCL}$ та

$$\forall (T^i, T^j) \in T^i_{CL-RCL} \in T^i_{RCL} \in S^k_{co}(T^{co}_{RCL}) : T^i \in S^{k-1}_{co}(T^{co}), T^j \in S^k_{co}(T^{co}).$$

Контекстом певного класу T^{co} вважаємо контекст максимального рівня m, який можна побудувати у даній локальній базі знань:

$$Con_{kn}(T^{co}) = Con_{kn}^{m}(T^{co}) \mid m = \max(k). \quad (6)$$

Визначимо контекст моделі T_{MD}^{co} на рівні знань як контекст усіх класів, які використовуються у моделі:

$$Con_{kn}(T_{MD}^{co}) = \bigcup Con_{kn}(T^{co}) \mid T^{co} \in T_{RG}^{co}. \quad (7)$$

В роботі показано як використання онтологічних моделей задач дозволяє спростити вирішення задач *менеджменту складних онтологій*, зокрема розроблено методологію побудови онтології, що базується на аналізі та онтологічному моделюванні задач системи. Спрощення вирішення задач менеджменту та підвищення якості онтології відбувається шляхом конкретизації критеріїв включення сутностей та відношень предметної області в онтологію, ітеративності процесу розбудови онтології та визначення об'єктивних критеріїв валідації онтології.

Процес створення онтології виконується для кожної задачі системи та містить наступні кроки: а) кожну задачу аналізують та визначають усі необхідні для її вирішення сутності та відношення, обмеження та операції; б) будують концептуальну модель задачі з використанням виявлених компонент, причому ця модель повинна у першу чергу використовувати компоненти з існуючої онтології; в) в онтологію додають компоненти задачі, яких у ній не було; г) оновлену онтологію валідують та вирішують можливі протиріччя.

Використання онтологічних моделей задач для побудови онтології має суттєві переваги перед традиційними підходами. Зокрема, на кожному кроці визначено чіткі критерії включення компонента в онтологію; побудова онтології є ітеративним процесом, в якому додавання нової задачі до компетенції онтології призводить до її розширення; на кожному етапі зміст онтології відповідає комплексу задач, що вирішуються з її допомогою; відносно невелика кількість компонент задачі дозволяє онтологічному інженеру зосередитися на виявленні та формалізації відношень та обмежень, забезпечити потрібну глибину онтології; повторно використовують знання отриманні під час онтологічного моделювання інших задач.

Побудова онтології на основі моделей задач дозволила провести *валідацію онтології*, що в традиційних підходах здійснюється експертами на основі суб'єктивних оцінок. Відомими вимогами до валідної онтології є вимоги повноти, коректності та відсутності надлишковості. Зокрема, онтологія вважається *повною*, якщо усі релевантні для визначеної компетенції аспекти предметної області відображені у ній. Онтологія вважається *коректною*, якщо знання, визначені у ній коректні для визначеної предметної області та релевантні для функцій, які виконує інтелектуальна система, що використовує онтологію.

По аналогії з цими визначеннями вважаємо *повною* модель, якщо вона містить усі необхідні для вирішення поставленої задачі сутності, відношення,

обмеження та операції. *Коректною* вважаємо модель, яка вирішує поставлену задачу згідно із заданими критеріями ефективності.

Нехай для заданої предметної області визначено множину повних та коректних моделей $S(T_{MD})$. Тоді онтологія, побудована на основі цієї множини є повною, якщо:

$$\forall Cm_i \in T_{MD}^i \in S(T_{MD}) : Cm_i \in On, \tag{8}$$

де Cm_i – довільний компонент онтології. Онтологія є коректною, якщо вона побудована на основі коректних моделей. Верифікація онтології здійснюється вбудованими засобами середовища розробки онтологічних моделей.

У третьому розділі обгрунтовано архітектурні принципи побудови програмної системи на основі онтологічних моделей задач та методи організації взаємодії моделей. Розроблено методи підвищення адаптивних можливостей програмної системи шляхом накопичення та повторного використання знань про предметну область у формі онтологічних моделей задач.

Програмна система онтологічного моделювання задач складається з компонент, представлених на рис. 2. База фактів містить факти про об'єкти та події зовнішнього світу, необхідні для вирішення задач системою. Всі факти семантично інтерпретовані, тобто подані як об'єкти певних класів, визначених онтологією. База фактів, онтологія та моделі у сукупності утворюють базу знань системи. Онтологія містить модель предметної області, подану як таксономія класів. Це створює можливість однозначного трактування усіх об'єктів з бази фактів. Онтологічні моделі інкапсулюють знання про способи вирішення задач. На основі моделей формують факт-моделі ініціалізовані конкретними фактами з бази фактів.

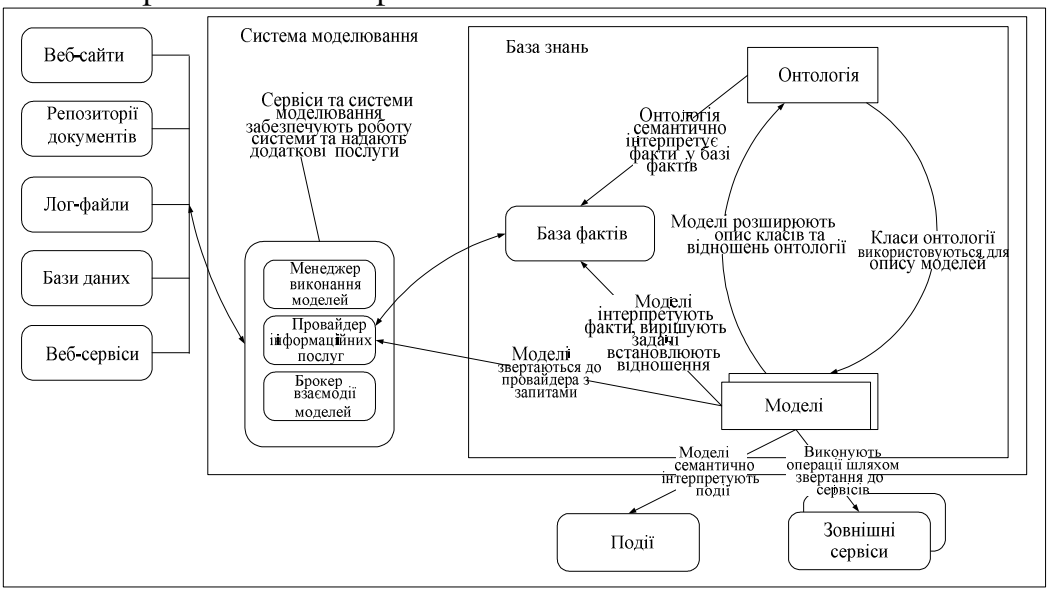


Рис. 2. Структура програмної системи онтологічного моделювання задач

Система реагує на визначене коло подій зовнішнього світу та опрацьовує їх, створюючи нові або модифікуючи існуючі факти. Важливими компонентами

системи є сервіси системи моделювання, які забезпечують виконання моделей, їх взаємодію, пошук потрібної інформації у зовнішніх джерелах. Так, *Менеджер виконання моделей* реалізує запуск або зупинку моделей, відслідковує використання ресурсів моделей. *Брокер взаємодії моделей* реалізує пошук релевантних моделей для вирішення задач, ініціалізує та запускає обрані моделі. *Провайдер інформаційних послуг* за запитом моделі системи звертається до зовнішніх джерел, реалізує пошук необхідних даних та їх семантичну інтерпретацію.

Моделі виконують операції, відповідно до відображеної в них логіки, звертаючись з запитами до зовнішніх відносно системи моделювання сервісів. Такими сервісами є сервіси операційної системи, сервіси інформаційної системи підприємства, побудованої з дотриманням вимог SOA, або довільні веб-сервіси.

Зазначені у першому розділі недоліки існуючих модельно-орієнтованих підходів до побудови програмних систем, значною мірою пояснюються складністю як предметної галузі, так і відповідних моделей. Ці недоліки в роботі запропоновано усунути шляхом реалізації програмної системи як набору простих взаємодіючих інтерпретованих моделей задач. Виконувальний та інтерпретований характер опрацювання моделей у запропонованому підході створює ряд додаткових переваг порівняно з традиційними методами розробки програмного забезпечення і з xUML, зокрема:

- зникає необхідність при будь якій зміні логіки виконання проводити перекомпіляцію та розгортання системи;
- алгоритм виконання, який у традиційних методах розробки програмного забезпечення є жорстко прописаний у коді, у запропонованому підході перенесено у модель, що виконується інтерпретатором моделей;
- виконувальність моделі дозволяє повторно використовувати знання, інкапсульовані у моделях та онтологіях для збільшення інтелекту системи та її адаптаційних можливостей;
- наявність виконувальної та релевантної у будь-який час моделі дозволяє опублікувати інформацію про модель та використати її в операціях аналізу та оптимізації самої моделі.

На відміну від класів онтології, моделі не утворюють чіткої ієрархії і формують динамічну мережу, в якій зв’язки та самі моделі можуть змінюватися, відображаючи процеси навчання, зміни у зовнішньому світі, або процес вирішення певної задачі.

Активна факт-модель – це екземпляр визначеного типу моделі ініціалізований інформацією з певного контексту. Факт-моделі створюються на вимогу інших моделей або при настанні певних подій. Їх використовують для вирішення поточних задач системи.

Взаємодія моделей T_{RMD} – це тип даних, що відображає відношення активації, яке використовується для прийняття рішення щодо активації моделі

(створення та запуску на виконання нової факт-моделі) (рис.3). Ініціатором виступає модель-активатор. Потреба у активації та створенні нової факт-моделі виникає тоді, коли для вирішення основної задачі потрібно вирішити допоміжні задачі, подані в інших моделях.

Кожному підтипу T_{RMD} - T_{RMD}^{i} відповідає клас задач T_{PR}^{j}, які необхідно вирішити в результаті взаємодії. У свою чергу, класу задач T_{PR}^{j} відповідає множина моделей $S(T_{MD}^{j})$, які можуть бути застосовані для вирішення задач цього класу. Під час взаємодії моделей послідовно вирішуються задачі визначення релевантності, оптимального вибору серед релевантних моделей та ініціалізації обраної моделі.

Функція релевантності є відображенням поточного контексту моделі-активатора $Con(t_{MD}^{act})$ та множини альтернатив $S(T_{MD})$ у множину {true, false}:

$$F_{rel} : Con(t_{MD}^{act}), S(T_{MD}) \to (true, false). \quad (9)$$

Визначення релевантності моделей дозволяє відібрати для процедури вибору тільки релевантні моделі $S(T_{MD}^{re}) \subseteq S(T_{MD})$, тобто такі типи моделей T_{MD}^{re}, для яких

$$F_{rel}(Con(t_{MD}^{act}), T_{MD}^{re}) = true. \quad (10)$$

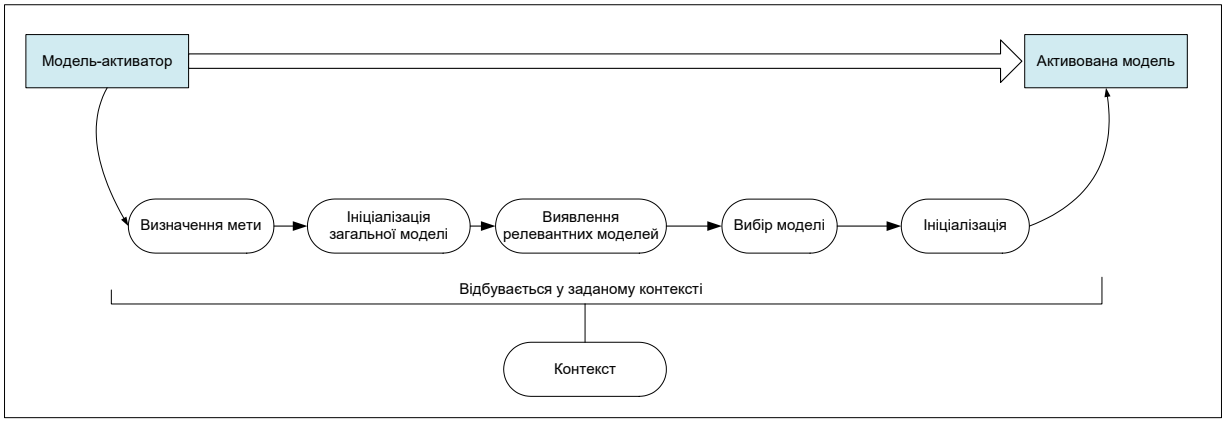


Рис. 3. Процес активації моделі задачі

При відсутності релевантних моделей система моделювання повертає активатору повідомлення про неможливість вирішення задачі.

Задача оптимального вибору визначає у множині релевантних моделей одну T_{MD}^{op}, застосування якої максимізує функцію вибору F_{ch} з врахуванням критеріїв вибору $S(t_{CR})$ та контексту $Con(t_{MD}^{act})$:

$$F_{ch}(T_{MD}^{op}, S(t_{CR}), Con(t_{MD}^{act})) \to \max. \quad (11)$$

Функція ініціалізації F_{in} проводить відображення біжучого контексту $Con(t_{MD}^{act})$ у множину значень атрибутів обраної моделі $-\{At^{j} \in T_{MD}^{sel}\}$:

$$F_{in} : Con(t_{MD}^{act}) \to \{At^{j} \in T_{MD}^{sel}\}. \quad (12)$$

Накопичення та повторне використання знань у програмній системі здійснюється шляхом створення та використання онтологічних моделей задач на трьох рівнях концептуальної моделі програмної системи. Зокрема, на рівні бізнес-процесів визначають моделі бізнес-процесів та оперують поняттями, що відображають характеристики бізнес процесів.

На рівні сервісів розглядають застосування та загальносистемні сервіси та ресурси, які використовують бізнес-процеси для виконання операцій. Тут враховують сумісне використання ресурсів, формулюють та впроваджують загальносистемні корпоративні політики, керують пріоритетністю виконання завдань.

Рівень пристроїв містить моделі виконавців завдань (процесорів). Виконавцем може бути як машина певного типу, так і людина. На цьому рівні визначають загальні характеристики, обмеження та моделі функціонування процесорів.

Таким чином, програмна система розглядається як кортеж множин моделей:

$$PS = (S_{MD-BP}, S_{MD-SE}, S_{MD-PC}). \quad (13)$$

На кожному рівні моделювання розглядається комплекс взаємопов'язаних моделей, які використовують для вирішення задач адаптації у межах цього рівня. Так, на рівні бізнес-процесів адаптувати поведінку системи можна шляхом зміни пріоритетності виконання завдань, переходом на інший варіант структури процесу. Одним з варіантів адаптаційних рішень є зміна вимог до інфраструктури. Рішення, про вибір того чи іншого варіанта адаптаційної поведінки вибирається у межах рівня на основі наявної бази знань залежно від характеристики ситуації, яка вимагає проведення адаптації системи.

У роботі головну увагу приділено тій частині загальної бізнес-моделі, яка виконується з використанням комп'ютерної техніки та інформаційних технологій і є перспективною для впровадження інтелектуальних технологій – рівні керування інфраструктурою та реалізації процесів (process implementation layer) (Osterwalder). У роботі розглянуто задачу класифікації моделей, що використовуються при моделюванні бізнес-процесів на основі різних класифікаційних ознак (рис. 4).

Так, за ознакою приналежності до конкретної предметної області, поділимо моделі на загальні та специфічні. Загальні моделі використовуються для подання знань у багатьох предметних областях. Прикладом таких загальних моделей виступають часові моделі, моделі особи, деякі моделі документів та моделі структури організації. Специфічні моделі використовують для подання знань у конкретній предметній області. Прикладом такої моделі є модель процесу розробки програмного продукту. Ще одну класифікацію моделей визначимо за класами задач, які вони вирішують та особливостями архітектури моделі: алгоритмічні, ситуаційні моделі, моделі класифікації, моделі операцій та ін. Як правило, усі моделі одного класу посилаються на одну спільну,

загальну модель, що подає задачу моделі у найбільш загальному вигляді та часто мають спільні програмні засоби для інтерпретації.

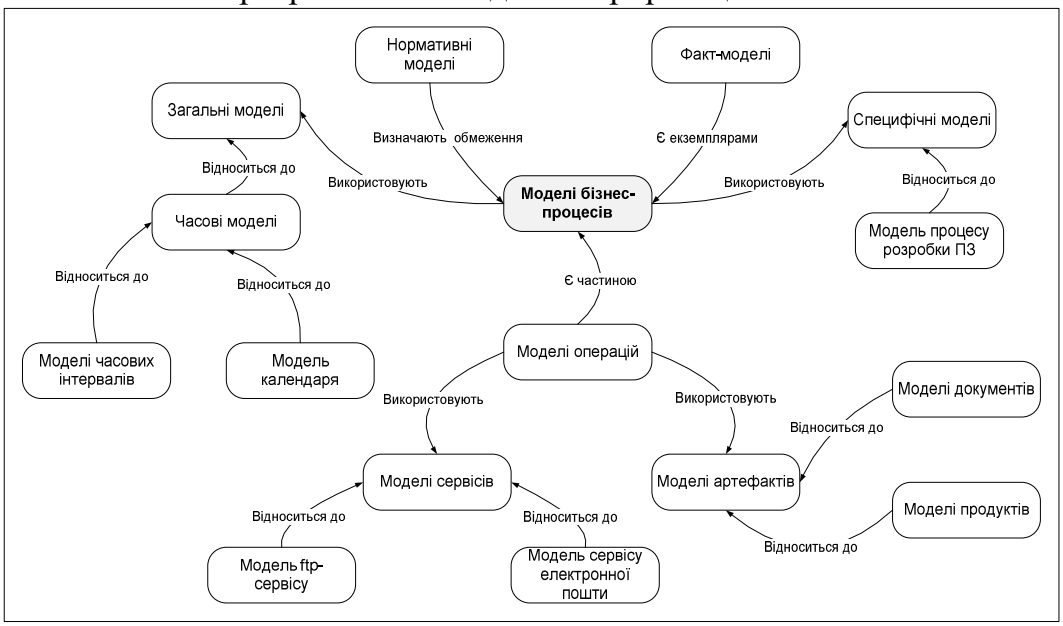


Рис. 4. Схема залежностей між різними типами моделей у програмній системі на базі онтологічних моделей задач

Головними компонентами другого рівня моделювання є *моделі сервісів*. При побудові моделей сервісів доцільно дотримуватися таких правил:

- усі сервіси потрібно впорядкувати в межах онтології сервісів; підтримка онтології сервісів дозволить здійснювати пошук сервісів за класами задач, зберігати числові параметри та інші метадані про сервіси;
- для клієнтів таких інтелектуальних сервісів важливо розуміти не тільки мету та параметри сервісу, але й порядок його роботи, функціональні обмеження та залежності; цю інформацію клієнти використовують для прийняття рішення у процесі взаємодії з сервісом; знання про порядок роботи інтелектуального сервісу доцільно подати окремою моделлю сервісу;
- онтологічні моделі доцільно використати і для подання механізмів роботи самого сервісу; сервіс, що використовує моделі може враховувати свій стан та стан зовнішнього середовища, співпрацювати з іншими інтелектуальними сервісами, визначати ефективніші методи обслуговування запиту клієнта.

В роботі розроблено архітектуру інтелектуального сервісу, побудованого на основі онтологічних моделей, яка специфікує як семантику доступу до сервісу, так і семантику виконання запитів сервісом (рис. 5).

Клієнтом сервісу виступає певна зовнішня відносно сервісу активована модель, яка формує запит до сервісу. Інтелектуальний сервіс, з метою організації взаємодії з клієнтами, має одну або декілька схем моделей, що відіграють роль інтерфейсу сервісу. Запит до сервісу полягає у заповненні ролей інтерфейсної моделі значеннями запиту та генерування команди виконання запиту. Запити класифікуються та типізуються відповідно до

існуючої онтології типів запитів. Різні типи запитів мають різні передумови та алгоритми виконання. При цьому інтерфейсна модель перевіряє необхідні передумови у сенсі наявності необхідних для виконання запиту даних та відомостей про клієнта залежно від типи запиту. Якщо якась передумова запиту не виконана, то клієнту повертається відповідь з детальним поясненням причини відхилення запиту. Якщо усі передумови для виконання запиту дотримані, запит виконується сервісом з використанням набору його внутрішніх моделей. До таких моделей відносяться: моделі виконання запиту та моделі оркестровки. Моделі виконання запитів є алгоритмічними моделями, які залежно від типу запиту та його параметрів визначають процес його виконання як послідовність операцій. Моделі оркестровки містять схему взаємодії з зовнішніми сервісами у процесі виконання запиту та координації їх роботи. Моделі стану сервісу відображають поточний стан сервісу і використовуються для його налаштування, адміністрування, пошуку можливих несправностей.

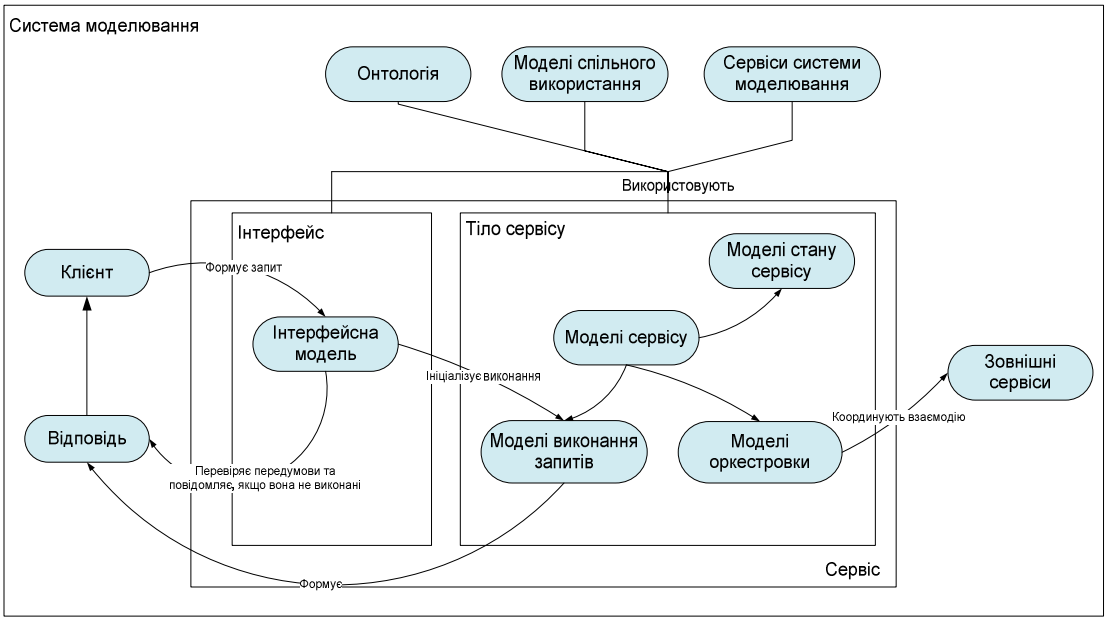


Рис. 5. Архітектура інтелектуального сервісу програмної системи на базі онтологічних моделей задач

Інтелектуальні сервіси у програмній системі утворюють мережу сервісів (Реєстр сервісів, service inventory), яка є множиною сервісів:

$$S(t_{SE}) = \{t_i \mid Type(t_i) = T_{SE}\}. \quad (14)$$

На основі базових сервісів будують композиції сервісів, які взаємодіють задля досягнення певної мети. Сервіси не залежать від бізнес-процесів, так що кожен окремий сервіс приймає участь у багатьох композиціях.

Композиція сервісів T_{SE-CM} – це підмножина сервісів та зв'язки між ними:

$$T_{SE-CM} = (S(T_{SE}), S(T_{SE-REL})), \quad (15)$$

де $S(T_{SE}) \subset Population(T_{SE}), S(T_{SE-REL}) \subset Population(T_{SE-REL})$.

Зв'язок двох сервісів $T_{SE}^{i}, T_{SE}^{j}\text{-}t_{SE-REL}$ – це пара $(T_{SE}^{i}, T_{SE-RQ}^{j})$, де T_{SE-RQ}^{j} – запит, що належить інтерфейсу сервісу T_{SE}^{j}.

Тип сервісу T_{SE} визначається як структура з типів його інтерфейсної моделі та реалізації:

$$T_{SE} = (T_{SE-IN}, T_{SE-BD}, T_{SE-RE}), \tag{16}$$

де T_{SE-IN} – інтерфейсна модель сервісу, T_{SE-BD} – тіло сервісу, T_{SE-RE} – реалізація сервісу.

Тіло сервісу містить моделі стану та керування T_{MD-ST}, моделі виконання запитів T_{MD-RQ} та моделі оркестровки T_{MD-OR}:

$$T_{SE-BD} = (S(T_{MD-ST}), S(T_{MD-RQ}), S(T_{MD-OR})). \tag{17}$$

Реалізація інтелектуального керованого моделями сервісу містить такі частини:

$$T_{SE-RE} = (T_{INT}, T_{SE-PM}, T_{SE-LIB}), \tag{18}$$

де T_{INT} – інтерпретатор моделей, T_{SE-PM} – сховище параметрів, T_{SE-LIB} – бібліотека процедур. Сховище параметрів та бібліотека процедур відіграють допоміжну роль у роботі інтерпретатора сервісу.

Моделювання на рівні пристроїв відноситься до третього рівня моделювання. Моделювання на цьому рівні вирішує дві групи завдань.

Завдання першої групи пов'язані з забезпеченням ефективної роботи конкретного пристрою, його компонентів або комплексу пристроїв. Прикладами таких об'єктів керування виступають комп'ютери, жорсткі диски або набір комп'ютерів, об'єднаних у мережу. Моделі цього рівня відслідковують стан пристроїв, виявляють ранні ознаки майбутніх збоїв та відмов, автоматизують виконання деяких функцій людини-адміністратора. Моделі також забезпечують інтелектуальне реагування комп'ютерної системи на непередбачувані події.

Завдання другої групи зводяться до адаптації налаштувань пристрою до вимог продуктивності, які ставлять моделі рівня сервісів та бізнес-процесів. При цьому використовують моделі оцінки продуктивності системи та досвід експлуатації. Наприклад, на період проведення відеоконференції пріоритезують трафік конференції у мережі та блокують усі не пов'язані з виконанням виробничих завдань передавання.

На вхід системи моделювання надходять запити, завдання на виконання та команди керування. На виході системи є відповіді на запити та статусні повідомлення. Типовими клієнтами системи виступають моделі сервісів, що функціонують на рівні моделювання сервісів інтелектуальної системи. Для доступу до підсистеми моделювання використовують інтерфейсні моделі. Ці моделі згруповано відповідно до задач керування та подають основні параметри та вихідні дані для проведення змін у системі. Для вирішення поставлених задач використовують інші моделі, які інкапсулюють знання про способи вирішення задач, залежно від їх ефективності за різних умов та станів

середовища. Для адаптації системи до стану середовища та реагування на події використовують ситуаційні моделі, які ідентифікують передбачені заздалегідь ситуації та ініціюють визначені дії з проведення налаштування пристрою.

Розроблену структуру системи моделювання на рівні пристроїв наведено на рис. 6.

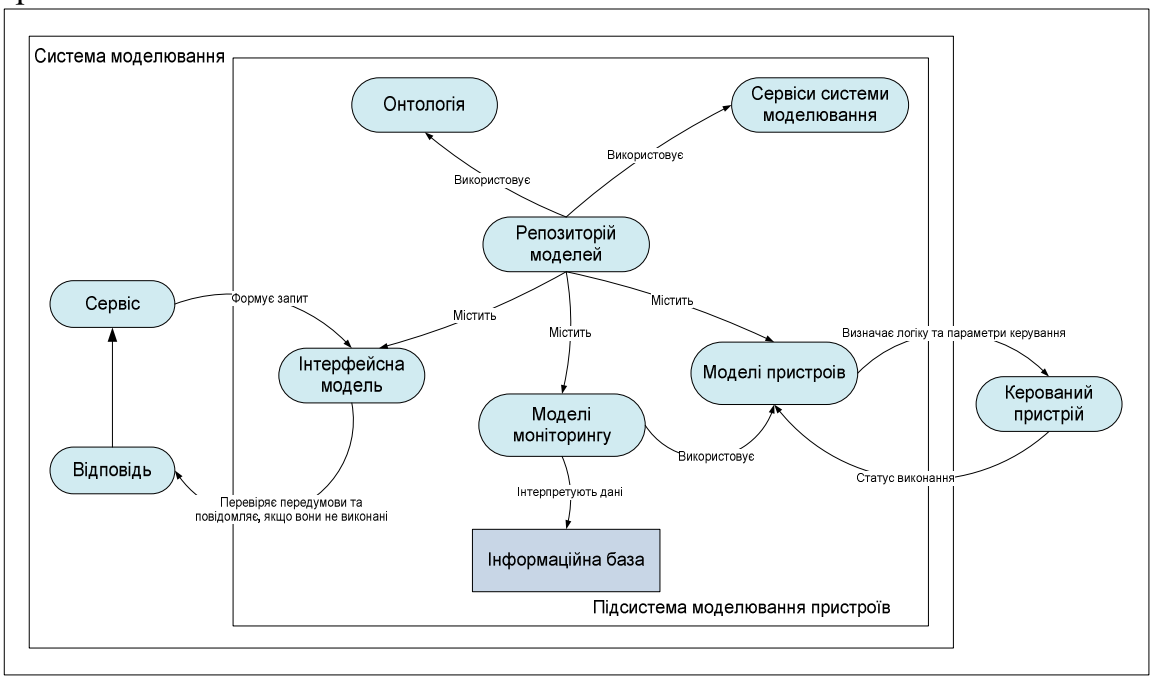


Рис. 6. Структура системи моделювання рівня пристроїв з використанням моделей задач

Усі рішення приймають на основі даних, що накопичуються в інформаційній базі. Тому важливою складовою частиною системи виступає підсистема моніторингу. Ця підсистема визначає, які параметри модельованого пристрою і як часто треба вимірювати та записувати в інформаційну базу. Моделі системи здатні впливати на підсистему моніторингу, модифікуючи політики моніторингу.

Система моделювання взаємодіє з модельованим пристроєм, передаючи йому команди керування у форматі відповідної мови керування. Для підтримки такої взаємодії використовують моделі пристроїв, які інкапсулюють знання про способи керування пристроєм та здійснюють перетворення завдання на керування у формат мови керування конкретним пристроєм.

У четвертому розділі наведено приклади вирішення практичних задач з використанням онтологічних моделей. У роботі розглянуто задачі предметної області розробки програмного забезпечення, підтримки прийняття рішень та туризму.

В роботі розглянуто застосування онтологічних моделей задач для *відображення структури бізнес-процесів* підприємства у програмній системі з метою покращення відповідності вимогам та збільшення адаптаційних

можливостей такої системи в плані підтримки бізнес-процесів. Модель бізнес-процесу MD_{bp} визначена кортежем:

$$Md_{bp} = (S(Md_{bo}), S(Md_{dc}), S_{rl}, S_{ac}, S_{cs}), \quad (19)$$

де $S(Md_{bo})$ – множина моделей бізнес-операцій, $S(Md_{dc})$ – множина моделей прийняття рішення, S_{rl} – множина відношень, S_{ac} – множина дій, S_{cs} – множина обмежень.

Продемонстровано доцільність застосування розробленої *методології побудови онтології* на основі попереднього аналізу бізнес-процесів і задач на прикладі галузі тестування програмного забезпечення (рис. 7).

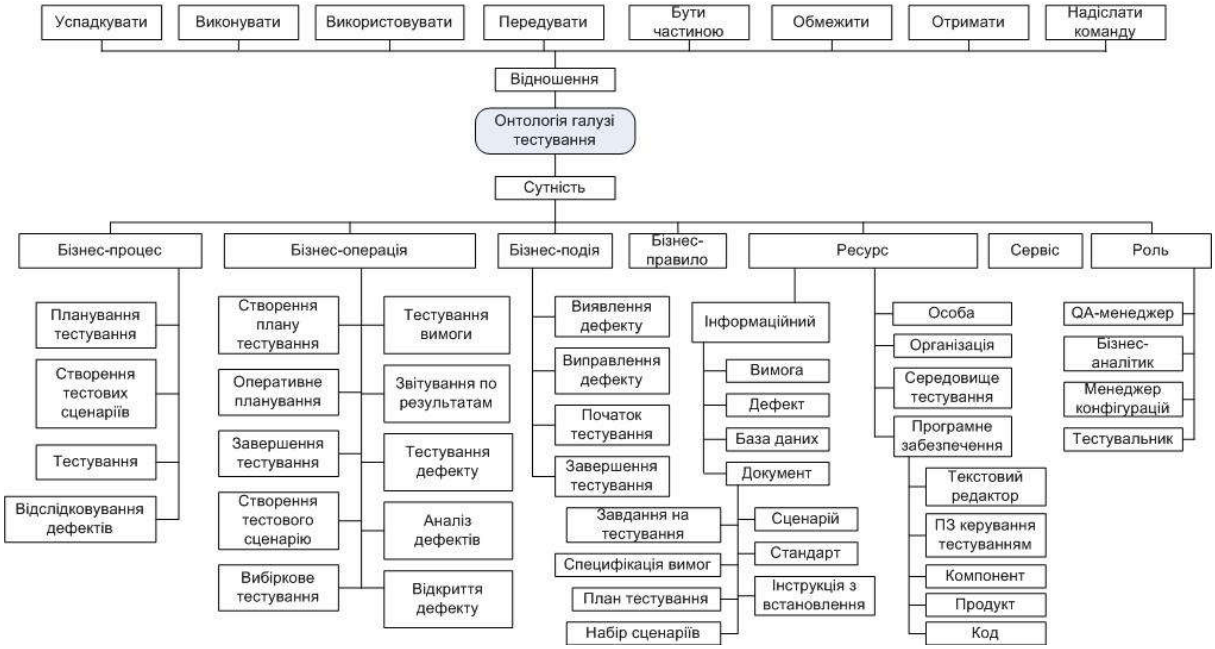


Рис. 7. Сутності та відношення онтології тестування програмного забезпечення

За основу аналізу процесів галузі тестування були прийняті процеси, сформульовані у стандарті ISO/IEC/ieee 29119. Стандарт визначає три групи процесів:

• організаційні – в результаті виконання цих процесів формулюють загальну політику та стратегію тестування на рівні проектної організації; ці процеси виконуються керівниками організації;

• керування у межах проекту – планування тестування, віслідковування стану тестування продукту, критеріїв його якості, завершення;

• тестування – створення тестових сценаріїв, формування тестових середовищ, виконання тестів, звітування про дефекти.

У роботі розглянуто процеси тестування, що виконуються у контексті конкретного проекту, тобто процеси другої та третьої груп. Визначено типові процеси та задачі і побудовано їх онтологічні моделі.

В роботі запропоновано використання онтологічних моделей для вирішення *задачі керування доступом до ресурсів інформаційної системи.* Показано, що у порівнянні з відомими методами DAC, MAC, RBAC, ABAC метод керування доступом, який використовує онтологічні моделі, забезпечує динамічне надання та вилучення прав доступу в контексті бізнес-процесів, що виконуються у системі, відсутність розширення прав з бігом часу, спрощення процесу призначення прав та документальне обгрунтування усіх операцій, підтримку використання нормативних процедур. Загальну схему моделі керування доступом до ресурсів наведено на рис. 8.

Модель керування доступом до ресурсів Md_{ar} подано кортежем:

$$Md_{ar} = (S_{ro}, S_{rs}, S(Rl_{ar}), S_{ac}, S_{cs}), \tag{20}$$

де S_{ro} – множина ролей, S_{rs} – множина ресурсів, $S(Rl_{ar})$ – множина відношень, S_{ac}, S_{cs} – множини дій та обмежень.

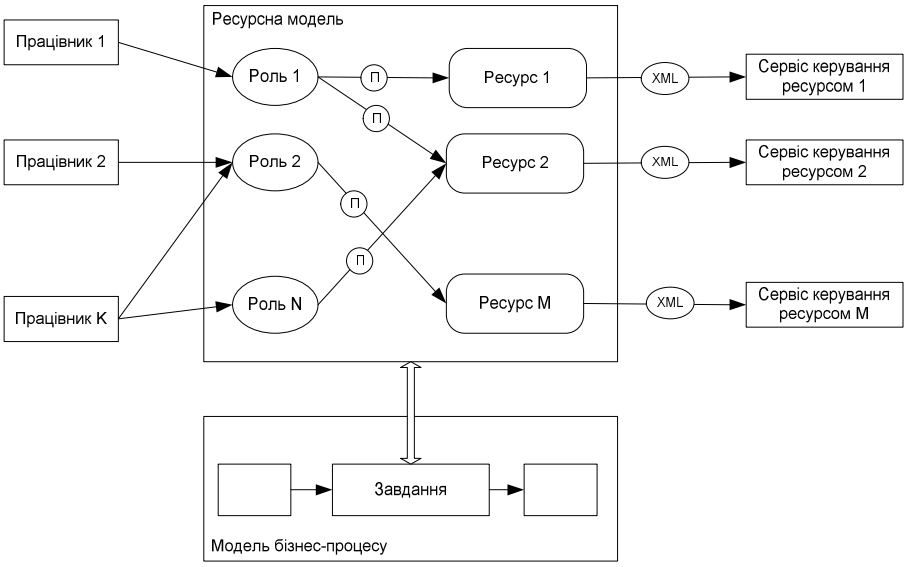


Рис. 8. Структура моделі керування доступом до ресурсів у програмній системі

Розроблено метод використання онтологічних моделей для вирішення *задач автоматизованого тестування зборок програмних продуктів.* Продемонстровано організацію взаємодії моделей у процесі виконання багатоетапного бізнес-процесу автоматизованого тестування. З використанням описаного підходу створено та апробовано автоматизовану систему тестування програмних продуктів, яка успішно функціонує. Для реалізації системи було обрано скриптову мову VBScript. В ролі програмного засобу автоматизованого тестування використовується HP QuickTest Professional, який теж використовує VBScript як мову програмування.

Система автоматизованого тестування періодично перевіряє наявність нового файлу-інсталятора на ftp сайті розробника продукту. Якщо він наявний,

файл інсталятора скачується та розархівовується у визначений тестовий каталог. Після цього запускається процес інсталяції в автоматичному режимі, який не вимагає вводу інформації користувачем. Після його завершення система перевіряє успішність інсталяції, конфігурує та перезапускає деякі системні сервіси.

Якщо продукт успішно інстальовано, запускають засіб автоматизованого тестування, що проводить обрані тести. Результати тестування записують у протокол, який після завершення тестування пересилається по електронній пошті представникам розробника. Після завершення тестування тестований програмний продукт деінсталюється, тестовий каталог очищується і середовище готується до нового тестування.

Розроблена система дозволила систематично перетестовувати декілька значних за розміром (розмір файлу інсталятора 500-700 Мбайт) програмних продуктів за одну ніч. Експлуатація системи тестування довела її високу надійність, гнучкість, простоту модифікації та розвитку.

Використання онтологічних моделей для автоматизації тестування програмних продуктів дозволяє створювати інтелектуальні системи тестування, які здатні швидко адаптуватися до змін у функціональності тестованого продукту, а також врахувати та адекватно відреагувати на зміни в апаратно-програмному тестовому середовищі.

Перспективним напрямом застосування програмних систем на базі онтологічних моделей задач є *системи підтримки прийняття рішень.* Загалом, процес прийняття рішення починається з аналізу проблеми та ситуації і закінчується вибором рішення з множини альтернатив. Такий процес включає фази: аналізу, проектування, вибору рішення та реалізації. На етапі аналізу розглядають бізнесове середовище з врахуванням досвіду експерта. Експерт ідентифікує та класифікує проблему, виробляє вимоги щодо її вирішення. На цьому етапі доцільно застосувати онтологічні моделі, які відображають досвід вирішення попередніх задач та порівнюють параметри існуючого стану предметної області зі станами предметної області у випадку попередньо вирішених задач.

На етапі проектування експерт визначає цілі, розробляє альтернативні способи досягнення цілей, критерії якості способу вирішення задачі та обирає метод її вирішення. При цьому експерт постійно співставляє та оцінює вплив запропонованого ним альтернативного рішення на предметну область, виконуючи таким чином валідацію свого підходу до вирішення проблеми. Моделі цього етапу дозволяють визначити існуючі методи реалізації поставлених цілей, обмеження та побудувати альтернативні рішення.

На етапі вибору експерт оцінює побудовані альтернативи з використанням критеріїв та обирає одну з альтернатив. При цьому для обраного рішення додатково прогнозується ефект від його впровадження – проводиться

валідація. Моделі, які використовують на цьому етапі, допомагають експерту застосувати математичні та інші методи порівняння альтернатив.

Модель підтримки прийняття рішення Md_{dc} подана кортежем:

$$Md_{dc} = (S(T_{dc}), S(Rl_{dc}), S_{ac}, S_{cs}), \quad (21)$$

де $S(T_{dc})$ – множина сутностей релевантних для прийняття рішення, $S(Rl_{dc})$ – множина релевантних відношень, S_{ac}, S_{cs} – множини дій та обмежень.

Як приклад, у роботі наведено *онтологічні моделі для підтримки консенсусної валідації моделей* колективом експертів, а також моделі підтримки прийняття рішень у договірному процесі *формування угоди на розробку програмного забезпечення.*

Доцільність використання онтологічних моделей для побудови інтелектуальних програмних сервісів у роботі проілюстровано на прикладах *інтелектуального сервісу завантаження та сервісів туристичних послуг.* Зокрема, розроблено структуру моделей сервісу завантаження та визначено порядок його роботи. Запит, що надійшов, опрацьовується препроцесором завантажень. Цей препроцесор безпосередньо не здійснює завантаження, а тільки готує завдання до завантаження. Він, наприклад, планує час проведення завантаження, проводить пошук додаткової інформації, перевіряє правильність та виконувальність завдання. Опрацьовані завдання зберігають у сховищі завдань на завантаження.

За принципом роботи препроцесор завантажень є процесором правил, тобто весь алгоритм його роботи сформульований як набір правил. Правила роботи препроцесора зберігаються в окремій моделі – моделі попереднього опрацювання завантажень. Як кожна інша модель системи, модель попереднього опрацювання завантажень може змінюватися, додаючи системі необхідної гнучкості (система не потребує зміни коду). Підчас роботи окремі правила використовують дані з інших моделей, а також з історії попередніх завантажень. Правила можуть стосуватися як окремих завантажень, так і їх груп, створених за певною ознакою (наприклад, усіх завантажень з даного сайта, або всіх завантажень певної категорії контенту).

Процесор завантажень безпосередньо виконує завантаження згідно з графіком завантажень та додатковими умовами, сформульованими у правилах моделі завантажень. Аналогічно до препроцесора завантажень, процесор завантажень є процесором правил і весь алгоритм його роботи сформульовано як набір правил моделі завантаження.

У розділі продемонстровано переваги використання онтологічних моделей для *побудови гнучких туристичних сервісів,* зокрема використання інформації з контексту туристичної поїздки для підтримки прийняття рішень з формування пропозицій для клієнта.

У п’ятому розділі наведено архітектуру та функціональні можливості розробленого прототипу середовища моделювання програмних систем на основі онтологічних моделей.

Основними акторами у системі моделювання є:

- експерт предметної галузі – він створює або модифікує онтологічну модель, валідує її, здійснює нагляд над її використанням;
- агент моделювання – виконує моделі у середовищі моделювання.

Зведену інформацію про акторів, сценарії та відповідні інструментальні програмні засоби або сервіси системи моделювання відображено у табл. 3.

Таблиця 3. Сценарії використання комплексу моделювання

Актор	Сценарій використання	Програмний засіб (сервіс)
Експерт предметної галузі	Створення моделі. Підчас створення моделі відбувається її валідація та тестування, задаються правила для автоматизованої верифікації.	Редактор онтологій та моделей
	Оцінка результатів експлуатації моделей. Модифікація моделей.	
	Тестування моделі	
	Опрацювання онтології	
Агент моделювання	Виконання моделей	Інтерпретатор моделей
Середовище моделювання	Організація взаємодії моделей.	Брокер взаємодії моделей
	Пошук інформації	Провайдер інформаційних послуг

Моделі створюються та валідуються людиною-експертом у даній предметній галузі і відображають знання цієї людини щодо способу вирішення певної задачі. У розділі розроблено xml-орієнтовані мовні засоби для подання та опрацювання онтологічних моделей, визначено протокольні засоби взаємодії з сервісами середовища виконання моделей.

Для створення або модифікації моделі використовують інструментальний засіб – програму *Редактор онтологій та моделей*.

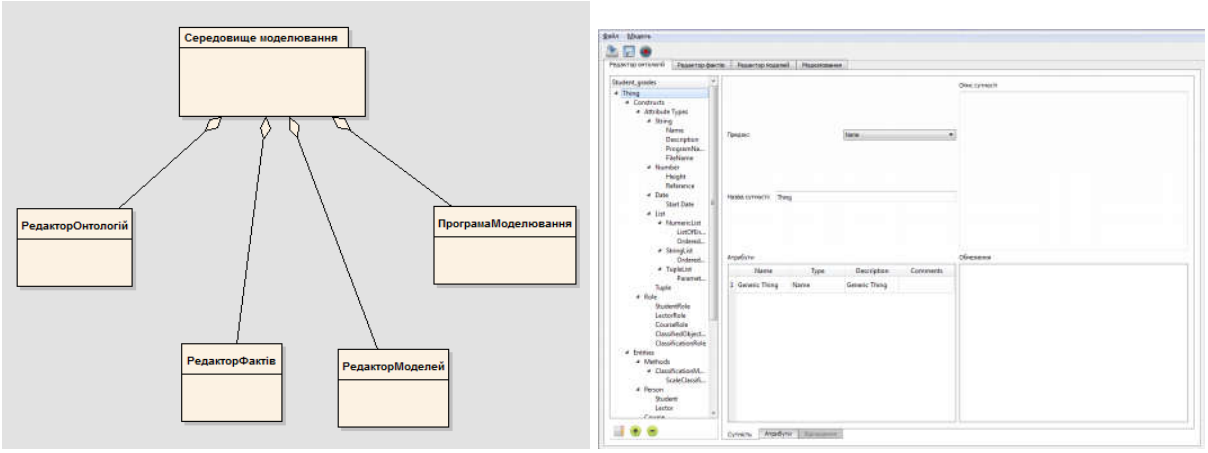
Основні функції (варіанти використання) цієї програми такі:

- створення схеми моделі з використанням класів та фактів онтології; визначення додаткових обмежень;
- робота з онтологією: пошук класів онтології та фактів; модифікація класів та відношень; визначення залежностей між певними класами онтології та існуючими моделями; визначення впливу змін в онтології на існуючі моделі;
- робота з базою фактів: пошук потрібних фактів, додавання або вилучення фактів з бази;
- визначення або налаштування існуючого інтерпретатора моделей – компонента, що виконує модель;

• тестування моделей та мереж (агрегатів) моделей: запуск та виконання моделей на тестовій базі фактів, перевірка відповідності поведінки моделей очікуваним результатам.

Основними вимогами до прототипу системи моделювання є гнучкість, можливість розширення, швидкість розробки та модифікації, підтримка можливості реалізації графічного інтерфейсу. Відповідно до цих вимог у ролі програмної платформи для створення прототипу було обрано мову Python (версії 2.7) та графічну бібліотеку PyQt, яка є переносом на платформу Python відомої відкритої та безплатної бібліотеки Qt. Мова Python є простою у використанні та модифікації і широко застосовується як мова для побудови програмних прототипів.

Прототип системи моделювання складається з чотирьох компонент, об'єднаних спільним інтерфейсом: *Редактора Онтологій, Редактора фактів, Редактора Моделей та Програми Моделювання* (рис. 9,а). Цей факт відображено і у робочому вікні середовища (рис. 9,б), в якому наявні вкладки редактора онтологій, редактора фактів, редактора моделей та програми моделювання.



а) б)

Рис. 9. Структурні компоненти середовища моделювання (а) та головне вікно середовища моделювання (б)

Функціональні можливості прототипу середовища моделювання проілюстровано на прикладах вирішення ряду практичних задач, зокрема задач класифікації та автоматизованого тестування.

У шостому розділі проаналізовано фактори, що впливають на ефективность застосування онтологічних моделей задач для побудови програмних продуктів, та розроблено оціночні формули для визначення ступеня збільшення окремих складових характеристик якості програмного продукту на різних етапах його життєвого циклу. Розглянуто характеристики якості визначені у стандарті ISO/IEC 9126-1. На вибірці проектів з розробки програмного забезпечення проаналізовано ступінь покращення характеристик якості програмного продукту при застосуванні онтологічних моделей задач.

На етапі концептуалізації розробник програмного продукту вирішує задачу побудови концептуалізації предметної області в контексті задач, які повинен вирішувати програмний продукт. Помилки, допущені на етапі концептуалізації, є найбільш вагомими і виправлення їх нерідко вимагає перероблення всього програмного продукту. При цьому великий обсяг робіт з первинної концептуалізації та аналізу предметної області фактично дублюється. При використанні онтологічного підходу розробник використовує концептуалізацію предметної області, відображену в онтології. Уникнення повторної концептуалізації безпосередньо впливає на такі характеристики якості як відповідність вимогам, точність, інтероперабельність, зрілість, використання ресурсів на етапі проектування, переносимість та можливість співіснування з іншими продуктами.

Для різних програмних продуктів, що використовують спільну концептуалізацію простіше організувати взаємодію, адже вони грунтуються на спільному наборі концептів з однаковими характеристиками. Завдяки багатократному використанню єдиної онтології у різних реально функціонуючих програмних продуктах досягається високий ступінь зрілості концептуалізації, а також стабільності та надійності, що досягається шляхом багатократного та різностороннього тестування, яке проходить онтологія підчас створення та експлуатації продуктів на її базі.

Запропоновано формулу для оцінки ступеня повторного використання концептуалізації:

$$C_{en-\det} = 1 - \frac{N_{el-new}}{N_{el-new} + N_{el-old}}, \tag{22}$$

де N_{el-new} – кількість доданих (або змінених) елементів онтології, N_{el-old} – кількість використаних (та не змінених) елементів онтології. Аналіз вибірки проектів показав, що показник повторного використання концептів для різних пар продуктів змінювався в інтервалі від 0,1 до 0,88.

Продукти, що базуються на спільній концептуалізації мають кращу інтероперабельність завдяки уникненню помилок, пов'язаних з різним поданням предметної області в них. Подамо оцінку $C_{\text{int}}(A,B)$ погіршення інтероперабельності, яке виникає через відмінності концептуалізації для двох продуктів *A* та *B*, у вигляді:

$$C_{\text{int}}(A,B) = \frac{N_{diff}^{A} + N_{diff}^{B}}{N_{comm}^{AB}}, \tag{23}$$

де N_{diff}^{A} – кількість відмінних елементів в онтології продукту *A* порівняно з продуктом *B*, N_{diff}^{B} – кількість відмінних елементів в онтології продукту *B* порівняно з продуктом *A*, N_{comm}^{AB} – кількість спільних елементів онтології, які використовують продукти *A* та *B*.

Використання єдиної онтології на *етапі дизайну та кодування* дозволяє розширити ступінь повторного використання коду, поданого онтологічними

моделями. При порівнянні підходу, що базується на онтологічних моделях з об'єктно-орієнтованим програмуванням, кожен клас ООП має декларативну частину подану оголошенням змінних та процедурну частину, подану специфікацією методів. При використанні онтологічних моделей всі моделі (які є аналогами методів) мають спільну декларативну частину подану онтологією.

При використанні об'єктно-орієнтованого програмування переносимість обмежена повторним використанням груп взаємозалежних класів. При цьому методи класу не є переносимими і можуть використовуватися тільки з екземплярами цього класу. Використання онтологій для побудови програмної системи дозволяє уникнути багатократного декларування об'єктів у різних класах, підвищити переносимість коду, зменшити використання ресурсів на стадії кодування шляхом повторного використання коду. Характеристики якості, які покращуються при використанні онтологічного підходу: *переносимість, зменшення використання ресурсів у процесі проектування.*

Для визначення ступеня переносимості коду розроблено такі оцінки:

1. Відсоток програмних конструкцій (класів, моделей, сервісів) у наявних бібліотеках коду, які можна використовувати без змін:

$$C_{port-code1} = \frac{N_{portable}}{N_{all}}. \tag{24}$$

Зауважимо, що у випадку використання онтологічних моделей $C_{port-code1} = 1$.

2. Кількість додаткових програмних конструкцій (класів, моделей), які необхідно долучити до коду продукту при використанні конкретного класу (моделі) – $C_{port-add}$.

Аналіз вибірки проектів зі створення програмного забезпечення продемонстрував, що різні програмні продукти мають однакові функції, що можуть бути подані моделями. Кількість спільних моделей, які можуть бути перенесені в інший програмний продукт, для різних пар проектів лежить в інтервалі від 0,1 до 0,75.

На етапі тестування програмного продукту використання онтологічних моделей призводить до зменшення обсягів тестування. При цьому тестуванню підлягають розроблені або модифіковані моделі, а не продукт загалом.

Оцінити ступінь зменшення трудомісткості проведення тестування запропоновано для програмного продукту перенесеного на нову (онтологічну) платформу. При цьому доступні тестові сценарії як для традиційної архітектури, так для нової. Оцінкою, що вимірює ступінь зменшення трудомісткості проведення тестування, є відношення кількості сценаріїв у новій платформі до кількості сценаріїв у традиційній платформі з врахуванням середньої складності (наприклад, кількості кроків) кожного сценарію:

$$C_{test} = \frac{N_{\text{mod}} \times C_{av-test-\text{mod}}}{N_{old} \times C_{av-test-old}}. \tag{25}$$

Важливим додатковим фактором покращення якості програмного продукту, побудованого з використанням онтологій, є те, що моделі, які повторно використовуються у програмному продукті, попередньо вже пройшли тестування та практично використовуються в інших програмних продуктах, що дозволяє говорити про повторне використання тестування, проведеного для інших програмних продуктів, а також припустити зменшення кількості помилок та збільшення ступеня зрілості продукту.

Основні характеристики якості програмного продукту, покращені за умови використання онтологічних моделей на етапі тестування: *придатність до тестування, стабільність, зрілість.*

На етапі супроводу та експлуатації програмного продукту вирішуються задачі адаптації при змінах у бізнесовій ситуації та проблемній області.

Запропоновано оцінювати складність модифікації програмного продукту традиційним способом:

$$C_{\text{m}-old} = T_{ov} \times \sum_{i=1}^{n} K_{ov}^{i} + T_{ch} \times \sum_{j=1}^{m} K_{ch}^{j}, \tag{26}$$

де n – загальна кількість програмних елементів (класів, функцій), m – кількість програмних елементів, які підлягають модифікації, K_{ov} – метрика складності i-го компонента, T_{ov} – середній час, що витрачається на аналіз, тестування та компоновку з розрахунку на один програмний компонент та одну одиницю складності, K_{ch} – метрика складності компонента, що змінюється, T_{ch} – середній час, що витрачається додатково на аналіз, перекодування та перетестування з розрахунку на один програмний компонент, що змінюється.

Зміни у вимогах та предметній області, які спонукають до розробки нової версії продукту, мають різний характер, що робить доцільним їх окремий аналіз. У роботі проаналізовано такі варіанти змін:

а) заміна методу вирішення задачі або формування нового методу комбінацією існуючих (подання предметної області залишається без змін);

б) розробка нового методу вирішення задачі, або уточнення існуючого;

в) уточнення подання предметної області шляхом введення нових сутностей, відношень та обмежень (при цьому визначені попередньо елементи предметної області не змінюються);

г) зміна розуміння предметної області, яка відображена у модифікації та заміні попередньо визначеного формального опису цієї області.

У випадку заміни методу вирішення задачі або формування нового методу як комбінації з існуючих, модифікація моделі є тривіальною і проводиться експертом предметної області в середовищі проектування. Додатковий час при цьому витрачається на тестування та підготовку даних для нього. Об’єктом зміни виступає модель. Оцінити складність проведення зміни можна з використанням формули:

$$C_{md} = K_m \times (T_a + T_m + T_t), \tag{27}$$

де K_m – оцінка складності (кількість компонент) моделі, T_a – час, що витрачається у середньому на аналіз та документування, T_m – модифікацію та T_t – тестування у розрахунку на один компонент.

Аналіз прикладів зміни методу рішення задачі для вибірки реальних проектів показав суттєве (в 5-8 разів) зменшення часових витрат на проведення зміни при використанні моделей задач.

Якщо потрібно розробити нову модель або модифікувати існуючу, то порівняно з першим випадком збільшуються витрати часу на створення і тестування моделі. Аналіз змін з вибірки реальних проектів показав, що зменшення часу у цьому випадку сягає 2-5 разів у порівнянні з традиційним підходом.

У випадку введення нових сутностей, відношень або обмежень в онтологію при збереженні існуючих її частин без змін проектувальник онтології проводить валідацію запропонованих змін. Він аналізує ці зміни на предмет відсутності конфліктів з наявними елементами онтології та обгрунтовує доцільність модифікації онтології. Модифікація онтології проводиться у редакторі онтологій. Складність проведення модифікації залежить від кількості нових елементів доданих до онтології:

$$C_{md-ont-add} = K_{\mathrm{mod}} \times (T_{an} + T_{\mathrm{mo}n}) + T_{md}, \qquad (28)$$

де $C_{mod\text{-}ont\text{-}add}$ – оцінка складності, K_{mod} – кількість доданих до онтології компонент, T_{an} – час на аналіз та валідацію, T_{mon} – час на проведення модифікації у редакторі онтологій з розрахунку на один компонент, T_{md} – час на модифікацію залежних від введених елементів онтології моделей. Аналіз змін з вибірки реальних проектів показав, що зменшення часу у цьому випадку сягає 1,5-3 разів у порівнянні з традиційним підходом.

Причиною зміни існуючих елементів онтології є переосмислення картини світу, а також виявлення змістовних та логічних помилок в онтології. Зміна існуючих елементів онтології вимагає змінити та перетестувати усі моделі, які працюють зі зміненими елементами. Така задача є доволі трудомісткою, адже кількість моделей у репозиторії моделей є значною. Архітектура та функціональні можливості системи моделювання роблять тривіальним вирішення задачі знаходження усіх моделей, на які впливають зміни в онтології. Для кожної такої моделі необхідно провести аналіз впливу змін, визначити обсяг необхідних модифікацій, провести зміну моделі та перетестувати змінену модель:

$$C_{md-on-rev} = (K_{on} \times T_{on}) + K_{md-on} \times (T_{an} + T_{md-\mathrm{mod}} + T_{test}), \qquad (29)$$

де $C_{mod\text{-}on\text{-}rev}$ – оцінка складності, K_{on} – кількість змінених елементів онтології, T_{on} – час, що витрачається на модифікацію одного елемента онтології у редакторі онтологій, $K_{md\text{-}on}$ – кількість моделей, які необхідно змінити, T_{an} – час на аналіз та проектування зміни в моделі, $T_{md\text{-}mod}$ – час на проведення зміни, T_{test} – час на тестування.

Аналіз змін з вибірки реальних проектів показав, що зменшення часу в цьому випадку сягає 1,2-2 рази у порівнянні з традиційним підходом. При цьому суттєво збільшуються загальні часові витрати на проведення модифікації.

Результати аналізу ступеня скорочення часу на проведення модифікації програмної системи для вказаних вище випадків а)-г), показують значний розкид оцінок залежно від особливостей конкретних проектів та колективів розробників. Водночас, доцільно ранжувати їх на основі експертних оцінок. Порівнюючи складності отримуємо шкалу метрик, в якій складність зростає:

$$(C_{md}, C_{md-ont-add}, C_{md-ont-rev}, C_{m-old}). \quad (30)$$

Таким чином, використання онтологічних моделей для побудови програмних систем призводить до покращення таких характеристик якості як *коректність, кількість дефектів, відповідність функціональним вимогам та зручність впровадження змін*.

ВИСНОВКИ

У дисертаційній роботі вирішено науково-прикладну проблему зменшення складності процесів створення та модифікації програмних систем шляхом використання онтологічних моделей задач. Основні наукові і практичні результати роботи полягають у такому:

1. Обгрунтовано підхід до побудови баз знань, який полягає у включенні у базу знань окрім онтології та інформаційної бази ще й онтологічних моделей. Такий підхід, на відміну від існуючих, поєднує переваги декларативного та процедурного підходів, орієнтований на підтримку вирішення задач з використанням бази знань.
2. Розроблено механізм взаємодії моделей, який забезпечує вирішення задач пошуку релевантних моделей, вибору моделі та її ініціалізації. Цей механізм робить можливим вирішення складних задач у системі, багатоваріантність методів вирішення задачі та повторне використання знань у системі.
3. Розроблено підхід до визначення контексту, який на відміну від існуючих, використовує базу знань та не залежить від задач, що використовують контекст, а тільки від стану бази знань. У межах цього підходу розроблено формальну специфікацію та спосіб визначення контексту для факту та класу бази знань.
4. Обгрунтовано нові способи вирішення задач менеджменту онтологій, що базуються на моделях задач. На відміну від існуючих, вони спрощують процес створення та модифікації онтології, а також створюють можливість її валідації через комплекс використаних моделей.
5. Розроблено методологію побудови онтології предметної області на основі аналізу комплексу задач, що вирішують у цій області. На відміну від існуючих, ця методологія визначає ітеративний процес розбудови онтології та забезпечує кращу її обґрунтованість.

6. Запропоновано та обгрунтовано принцип побудови модельно-орієнтованих програмних систем з використанням інтерпретації моделей знань, який на відміну від xUML дозволяє спростити процес побудови та модифікації системи.
7. Розроблено новий, оснований на моделях, метод керування доступом до ресурсів програмної системи, який порівняно з методом RBAC та ABAC не має ефекту накопичення прав доступу з часом, дозволяє надавати права у контексті виконуваних виробничих завдань, спрощує процес надання та вилучення прав.
8. Розроблено способи застосування виконувальних моделей для моделювання бізнес-процесів та систем бізнес-аналітики підприємств, визначення типів моделей, їх структури та способів використання.
9. Розроблено онтологію галузі тестування програмного забезпечення на основі аналізу процесів та завдань тестування.
10. Розроблено способи подання онтологічних моделей бізнес-процесів та методів їх використання для автоматизації тестування програмних продуктів.

СПИСОК ОПУБЛІКОВАНИХ ПРАЦЬ ЗА ТЕМОЮ ДИСЕРТАЦІЇ

1. Буров Є. В. Концептуальне моделювання інтелектуальних програмних систем: монографія / Є. В. Буров. – Львів : Вид-во Львівської політехніки, 2012. – 432 с. – ISBN 978-617-607-189-1. (27,0 умов. друк. арк.).
2. Burov E. Complex ontology management using task models / Burov E. // International Journal of Knowledge-Based and Intelligent Engineering Systems. – Amsterdam : IOS Press, 2014. – Vol 18, no 2. – P. 111-120. [Scopus, Academic Source Complete, ACM Guide to Computing Literature, CompuScience].
3. Burov E. Modeling software testing processes with task ontologies / Burov E, Pasitchnyk V., Gritsyk V. // British Journal of Education and Science. – London : London University Press, 2014. – № 2(6). – P. 256-263. [Scopus, РИНЦ].
4. Burov Y. Business process modelling using ontological task models / Burov Y. // Econtechmod. – Lublin : Polish Academy of Sciences, 2014. – № 1. – P. 11-23. [Google Scholar].
5. Буров Є. В. Опрацювання знань у когнітивній інформаційній системі керованій моделями / Буров Є. В. // Східно-європейський журнал передових технологій. – Харків : Технологічний центр, 2009. – №. 6/7(42). – С. 40-49. [IndexCopernicus,Ulrich's Periodicals Directory,DRIVER].
6. Буров Є. В. Опрацювання контексту у когнітивній інформаційній системі керованій моделями / Буров Є. В. // Східно-європейський журнал передових технологій. – Харків :Технологічний центр, 2010. – №1/7(43). – С. 40-47. [IndexCopernicus,Ulrich's Periodicals Directory,DRIVER].
7. Буров Є. В. Ефективність застосування онтологічних моделей для побудови програмних систем / Буров Є. В. // Математичні машини та

системи. – К. : Інститут проблем математичних машин та систем НАН України, 2013. – №1. – С. 44-55.

8. Буров Є. В. Архітектура опрацювання знань у когнітивній інформаційній системі / Буров Є. В. // Вісник Національного університету „Львівська політехніка” : Комп'ютерні науки та інформаційні технології. – 2009. – № 650. – С. 28-37.
9. Буров Є. В. Застосування виконувальних моделей для проектування сервісно-орієнтованих інтелектуальних інформаційних систем / Буров Є. В. // Вісник Національного університету ”Львівська політехніка” : Комп'ютерні науки та інформаційні технології. – 2009. – № 638. – С. 200-205.
10. Буров Є. В Застосування моделей процесорів та пристроїв для проектування інтелектуальних інформаційних систем / Буров Є. В. // Вісник Національного університету „Львівська політехніка” : Інформаційні системи та мережі. – 2009. – № 631. – С. 29-35.
11. Буров Є. В. Система моделювання інтелектуальної мережі бізнес-процесів / Буров Є. В. // Вісник Національного університету „Львівська політехніка” : Інформаційні системи та мережі. – 2008. – № 610. – С. 34-39.
12. Буров Є. В. Проектування інтелектуальної інформаційної мережі з використанням сервісно-орієнтованого підходу та моделей виконання запитів / Буров Є. В. // Вісник Національного університету ”Львівська політехніка” : Комп'ютерні системи проектування. Теорія і практика. – 2008. – № 626. – С. 10-15.
13. Буров Є. В Архітектура інструментального комплексу для моделювання інтелектуальних систем / Буров Є. В. // Вісник Національного університету „Львівська політехніка” : Комп'ютерні науки та інформаційні технології. – 2010. – № 686. – С. 34-43.
14. Буров Є. В. Використання моделей для керування доступом до ресурсів інтелектуальної інформаційної системи / Буров Є. В., Гульова А. В. // Вісник Національного університету „Львівська політехніка” : Інформаційні системи та мережі. – 2010. – № 673. – С. 59-68.
15. Буров Є. В. Актуальні проблеми автоматизації проектування розподілених інформаційних систем / Буров Є. В. // Вісник Національного університету „Львівська політехніка” : Інформаційні системи та мережі. – 1997. – № 315. – С. 15-22.
16. Буров Є. В. Моделювання подій у процесі проектування розподілених систем / Буров Є. В., Бурова О. Я. // Вісник Національного університету „Львівська політехніка” : Інформаційні системи та мережі. – 1998. – № 330. – С. 13-17.
17. Буров Є. В. Параметричне проектування та моделювання сервісів розподілених інформаційних систем / Буров Є. В., Пелещищин А. М. // Вісник Національного університету „Львівська політехніка” : Інформаційні системи та мережі. – 1999. – № 383. – С. 4-11.
18. Буров Є. В. Автоматизація аналізу функціональної структури розподіленої інформаційної системи / Буров Є. В. // Вісник Національного університету

„Львівська політехніка” : Інформаційні системи та мережі. – 2000. – № 406. – С. 39-50.

19. Буров Є. В. Система формальних специфікацій мережі сервісів та процесорів для проектування розподілених інформаційних систем / Буров Є. В. // Вісник Національного університету „Львівська політехніка” : Інформаційні системи та мережі. – 2001. – № 438. – С. 11-20.
20. Буров Є. В. Автоматизація проектування систем керування доступом у розподіленій інформаційній системі / Буров Є. В. // Вісник Національного університету „Львівська політехніка” : Інформаційні системи та мережі. – 2003. – № 489. – С. 12-25.
21. Буров Є. В. Система формальних специфікацій для конфігурування розподіленої інформаційної системи / Буров Є. В. // Вісник Національного університету ”Львівська політехніка” : Інформаційні системи та мережі. – 2004. – № 519. – С. 29-36.
22. Буров Є. В. Автоматизація визначення та моніторингу параметрів угоди про якість обслуговування (SLA) для дослідження впливу сервісів інформаційної системи на параметри бізнес-процесів / Буров Є. В. // Вісник Національного університету „Львівська політехніка” : Інформаційні системи та мережі. –2007. – № 589. – С. 33-35.
23. Буров Є. В. Інтелектуальний сервіс завантаження керований моделями / Буров Є. В. // Вісник Національного університету „Львівська політехніка” : Інформаційні системи та мережі. –2008. – № 621. – С. 55-60.
24. Буров Є. В. Інтелектуальна система підтримки прийняття рішень у договірному процесі / Буров Є. В., Калінчук Ю. О., Ломтєв А. В. // Вісник Національного університету ”Львівська політехніка” : Інформаційні системи та мережі. –2009. – № 653. – С. 24-31.
25. Буров Є. В. Інтелектуальний туристичний сервіс з опрацюванням контексту ситуації / Буров Є. В., Городецька А. І. //Вісник Національного університету ”Львівська політехніка” : Інформаційні системи та мережі. – 2010. – № 689. – С. 27-36.
26. Буров Є. В. Опрацювання контекстних даних в інтелектуальному туристичному порталі / Буров Є. В., Королюк Ю. В. // Вісник Національного університету ”Львівська політехніка” : Комп'ютерні науки та інформаційні технології”. –2010. – № 672. – С. 228-236.
27. Буров Є. В. Консенсус експертів як засіб для підвищення достовірності моделей знань в інтелектуальний системі / Буров Є. В., Крамаренко М. Б. // Вісник Національного університету ”Львівська політехніка” : Комп'ютерні науки та інформаційні технології. –2011. – № 694. – С. 212-220.
28. Буров Є. В. Побудова програмних систем з використанням онтологічних моделей задач / Буров Є. В., Пасічник В. В. / Системний аналіз та інформаційні технології: матеріали XVI Міжнародної науково-технічної конференції „SAIT’2014”. – К. : ННК “ІПСА” НТУУ “КПІ”, 2014. – С. 196-198.
29. Буров Є. В. Інтелектуальні програмні системи прийняття рішень на базі виконувальних моделей / Буров Є. В., Пасічник В. В. / Системний аналіз та

інформаційні технології: матеріали Міжнародної науково-технічної конференції „SAIT’2011”. – К. : ННК “ІПСА” НТУУ “КПІ”, 2011. – С. 34-36.

30. Burov E. Software systems based on ontological models / Burov Eugene / Computer Science and Information Technologies (CSIT’2011): Proc. of the VI-th Int. Conf., – Lviv : Publishing House Vezha&Co., 2011. – P. 148-151.
31. Burov Y. Using ontological models for complex ontology management / Burov Y. / Computer Science and Information Technologies (CSIT’2012): Proc. of the III Int. Conf. – Lviv : Vezha&Co., 2012. – C. 16-18.
32. Буров Є. В. Інтелектуальна система прийняття рішень у галузі туризму з використанням моделей ситуацій / Буров Є. В. / Системний аналіз та інформаційні технології: матеріали Міжнародної науково-технічної конференції „SAIT’2011”. – К. : ННК “ІПСА” НТУУ “КПІ”, 2011. – С. 209-210.
33. Буров Є. В. Опрацювання знань в інтелектуальній інформаційній системі керованій моделями / Буров Є. В. / Матеріали науково-технічної конференції ”Комп’ютерні науки та інформаційні технології” (ITCE’2010). – Вінниця : Вид-во ВНТУ, 2010. – С. 414-416.
34. Burov Y. A tool for modeling model-driven intellectual software systems / Burov Y. / Computer Science and Information Technologies (CSIT’2010): Proc. of the IV-th Int. Conf. – Lviv : Vezha&Co., 2010. – C. 56-59.
35. Буров Є. В. Тестування програмного продукту з використанням моделей знань / Є. В. Буров / Інтелектуальні системи прийняття рішень і проблеми обчислювального інтелекту (ISDMCI’2011): Тези доп. Міжнар. наук.-техн. конф. – Херсон : Вид-во ХНТУ, 2011. – С.47-50.
36. Буров Є. В. Автоматизація проектування та моделювання когнітивних бізнес-систем / Буров Є. В. / Computer Science and Information Technologies (CSIT’2008): Proc. of the III-rd Int. Conf. – Lviv : Vezha&Co., 2008. – C. 104-108.

АНОТАЦІЇ

Буров Є. В. Методи та засоби побудови програмних систем на основі онтологічних моделей задач. – На правах рукопису.

Дисертація на здобуття наукового ступеня доктора технічних наук за спеціальністю 01.05.03 – математичне та програмне забезпечення обчислювальних машин і систем. – Національний університет «Львівська політехніка» міністерства освіти та науки України, Львів, 2015.

У дисертації вирішено науково-прикладну проблему побудови програмних систем, здатних до адаптації в умовах змін середовища функціювання. Отримала подальший розвиток концепція побудови баз знань, яка полягає у включенні у базу знань окрім онтології та інформаційної бази ще моделей задач. Така концепція, на відміну від існуючих, поєднує переваги онтологічного та процедурного підходів, і дозволяє формалізувати, накопичувати та повторно використовувати досвід про способи виконання задач.

Розроблено механізм взаємодії моделей задач, який забезпечує вирішення задач пошуку релевантних моделей, вибору моделі та її ініціалізації. Цей механізм уможливлює вирішення складних задач у системі, багатоваріантність методів рішення задачі та повторне використання знань у системі. Розроблено означення контексту, яке на відміну від існуючих, використовує базу знань та не залежить від задач, що використовують контекст, а тільки від стану бази знань. Вперше розроблено методологію побудови онтології предметної області на основі аналізу комплексу задач, що вирішують у цій області. На відміну від існуючих, ця методологія визначає ітеративний процес розбудови онтології та забезпечує кращу її обґрунтованість, спрощує процес створення та модифікації онтології, а також створює можливість її валідації через комплекс використаних моделей. Обгрунтовано принцип побудови модельно-орієнтованих програмних систем з використанням інтерпретації моделей задач, який на відміну від xUML, дає можливість спростити процес побудови та модифікації системи. Розроблено інструментальні засоби для побудови та моделювання роботи програмних систем на основі онтологічних моделей задач.

Ключові слова: онтологія задачі, програмна система, онтологічна модель, адаптація, проблемна область, сервісно- орієнтовано архітектура.

Буров Е. В. Методы и средства построения программных систем на основе онтологических моделей задач. – На правах рукописи.

Диссертация на соискание ученой степени доктора технических наук по специальности 01.05.03 – математическое и программное обеспечение вычислительных машин и систем. – Национальный университет «Львовская политехника» министерства образования и науки Украины, Львов, 2015.

В диссертации решена научно-прикладная проблема повышения качества программных систем в аспектах сложности їх разработки и модификации, адаптации к актуальному состоянию предметной области.

В *первом разделе* проведен анализ существующих тенденций развития архитектур, технологий и процессов построения программных систем в контексте выявления возможностей для увеличения адаптационных возможностей при изменении требований. В разделе проанализированы существующие архитектурные решения, направленные на решение проблемы адаптации, и сделан вывод о перспективности использования для решения этой проблемы методов построения программных систем, базирующихся на принципах использования активных моделей среды функционирования и технологий интеллектуальных систем обработки знаний, в частности онтологий.

Во *втором разделе* определены теоретические основы представления и обработки знаний в программных системах на базе онтологических моделей задач. Разработана формальная модель представления знаний в системе. Формально определено понятие контекста фактов для обеспечения взаимодействия моделей. Предложена методология построения онтологий на основе онтологических моделей задач, которая определяет итеративный

процесс развития онтологии и обеспечивает лучшую ее обоснованность, упрощает процесс создания и модификации онтологии.

В *третьем разделе* разработаны архитектурные принципы построения программной системы на основе онтологических моделей задач. Предложены методы организации взаимодействия моделей. Разработаны методы использования онтологических моделей задач на трех уровнях моделирования программной системы – уровне бизнес-процессов, сервисов и приложений, устройств.

В *четвертом разделе* приведены примеры решения практических задач с использованием онтологических моделей. В разделе рассмотрено применение онтологических моделей задач для отображения структуры бизнес-процессов предприятия в программной системе с целью увеличения адаптационных возможностей. Продемонстрировано целесообразность применения разработанной методологии построения онтологии на основе предварительного анализа бизнес-процессов и задач на примере области тестирования программного обеспечения. В работе предложено использование онтологических моделей для решения задачи управления доступом к ресурсам информационной системы. Показано, что по сравнению с известными методами DAC, MAC, RBAC, ABAC метод управления доступом на основе онтологических моделей обеспечивает динамическое предоставление и изъятие прав доступа в контексте бизнес-процессов, выполняемых в системе, отсутствие расширения прав с течением времени, упрощение процесса назначения прав и документальное обоснование всех операций. Разработан метод использования онтологических моделей для решения задач автоматизированного тестирования сборок программных продуктов..

В *пятом разделе* приведена архитектура и функциональные возможности разработанного прототипа системы моделирования программных систем на основе онтологических моделей. Функциональные возможности прототипа системы моделирования проиллюстрированы на примерах решения ряда практических задач, в частности задач классификации и автоматизированного тестирования.

В *шестом разделе* определены факторы, влияющие на эффективность применения онтологических моделей задач для построения программных продуктов и разработаны оценочные формулы для определения степени увеличения отдельных составляющих характеристик качества программного продукта на разных этапах его жизненного цикла. На выборке проектов по разработке программного обеспечения проанализирована степень улучшения характеристик качества программного продукта при применении онтологических моделей задач.

Ключевые слова: онтология задачи, программная система, онтологическая модель, адаптация, проблемная область, сервисно-ориентированная архитектура.

Burov Y. Methods and tools for software systems construction using ontological task models. – Manuscript.

Dissertation for scientific degree of doctor of technical science in specialty 01.05.03 – mathematical and software supply of computing machines and systems. – Lviv Polytechnic National University, Lviv, 2015

The problem of building software systems capable to adapt in a changing environment has been solved in presented dissertation. The analysis of existing trends in software architectures, technologies, and processes of building software systems in the context of identifying opportunities to simplify their construction and increase adaptive capacity to requirements change was performed. Existing concept of knowledge-based system was enhanced by inclusion in knowledge base apart from ontology also ontological task models. This change realizes the advantages of both declarative and procedural approaches to knowledge base construction and allows for formalization, storage and reuse of knowledge about task execution.

The mechanism of task models interaction was developed. It uses the resolving of tasks of relevant models lookup, model selection and initialization and allows to implement a complex task resolution in a software system. For the purpose of model interaction the definition of context was provided and implemented. The methodology of ontology management based on task analysis and modeling including methods of ontology creation, modification, validation was developed. Differently from existing methodologies, proposed methodology defines an iterative process of ontology constructuion, simplifies it, creates a possibility of ontology validation using ontological models.

Based on ontological models new access control method was proposed. This method, when compared to RBAC, ABAC doesn't have an effect of permissions accumulation over time, and allows to assign permissions dynamically in context of executed business processes. Software tools prototype for building and modeling software system based on ontological models was developed.

Keywords: task ontology, software system, ontological model, adaptation, domain, service-oriented architecture.

Підписано до друку 23.03.15 р.
Формат 60х90 1/16. Папір офсетний.
Друк на різографі. Умовн. друк. арк.2,3 Обл.-видав. арк. 1,8.
Тираж 150 прим. Зам. _____

Поліграфічний центр
Видавництва Національного університету “Львівська політехніка”
вул. Колесси, 2, 79000, Львів
Реєстраційне свідоцтво серії ДК № 751 від 27.12.2001 р.