Міністерство освіти і науки України
Кам’янець-Подільський національний університет імені Івана Огієнка
Фізико-математичний факультет
Кафедра комп’ютерних наук

Дипломна робота
бакалавра
з теми: «**РОЗРОБКА ОНТОЛОГІЧНОЇ СИСТЕМИ ПРОЄКТУВАННЯ**»

Виконав: студент 4 курсу, групи KN1-B18
спеціальності 122 Комп’ютерні науки
Яцук Денис Андрійович
Керівник: Смалько О. А., кандидат педагогічних наук, доцент, доцент кафедри комп’ютерних наук
Рецензент: Оптасюк С. В., кандидат фізико-математичних наук, доцент, завідувач кафедри фізики

Кам’янець-Подільський – 2022

ЗМІСТ

ВСТУП

Цілеспрямована діяльність ґрунтується на знаннях, і ці знання включають як загальні, так і ситуаційно-специфічні знання цільової предметної області. Використання знань у відповідних сферах діяльності забезпечує ефективність процесу створення системи ситуаційного управління. Проектування сучасної комплексної системи ситуаційного управління, орієнтовано на її призначення та керовано подіями, і такі системи повинні бути спеціально налаштовані для вирішення актуальних проблем у різних ситуаціях. Проектування є етапом життєвого циклу розробки системи і реалізує процес визначення архітектури системи для відповідно до її призначення та задоволення визначених вимог.

Ситуаційна управлінська діяльність, пов’язана з консолідацією та обробкою інформації гетерогенного походження для отримання семантичної інформації. Впровадження та гармонізація різних компонентів діяльності організації в єдину архітектуру здійснюється шляхом конвергенції. Конвергенція може бути досягнута на основі ієрархічного модельованого підходу шляхом створення системи моделей, які описують різні аспекти цільової системи ситуаційного управління з використанням відповідної моделі знань. Процеси конвергенції засновані на корисних цінностях і принципах, накопчених різними технологіями під час їхнього розвитку. Процес ситуаційного управління може розглядатися як ланцюжок збільшення цінності інформації, і ситуаційне рішення по суті є інформаційним продуктом. У роботі обговорювався підхід, що базується на знаннях, які стосуються ситуаційного управлінні та проектування систем ситуаційного управління. Результат процесу ситуаційного управління може розглядатися як інформаційний продукт проектної діяльності. У роботі пропонуються архітектурні моделі побудови конвергентної системи ситуаційного управління з використанням знань предметної області.

Актуальною задачею у галузі комп’ютерних наук є максимальне наближення до практичного впровадження існуючих теоретичних основ кібернетики та штучного інтелекту, їхнє доповнення новими теоретичними розробками з

урахуванням останніх досягнень інженерної практики з подальшим урахуванням перспективних тенденцій та особливостей нових компонентів обчислювальної техніки з метою забезпечення сталого розвитку суспільства у всіх сферах діяльності.

Отже, **актуальність роботи** обґрунтовується необхідністю розробки ефективних формалізованих методик архітектурно-структурної організації та проектування кібернетичних організаційних систем, які дозволять не лише проектувати кібернетичні системи ситуаційного управління (ССУ) з обробкою загальномовних та предметних знань, а й автоматизувати процес побудови баз знань предметних областей на основі обробки великих обсягів текстової інформації, створюючи електронні колекції баз знань предметних дисциплін та їх основі проводячи складні міждисциплінарні наукові дослідження.

Метою дипломної роботи є розробка онтологічної бази знань для проєктування ССУ.

Завдання, які вирішуються в дипломній роботі:

- проаналізувати особливості побудови систем ситуаційного управління та визначити структуру знань щодо проєктування таких систем;
- провести аналіз архітектурних моделей кібернетичних систем та обрати архітектурну модель для проєктування;
- проведення аналізу та обґрунтувати вибір редактора онтологій;
- розробити онтологію для архітектурної моделі систем ситуаційного управління.

Об’єктом дослідження є знання-орієнтовані засоби проєктування складних систем.

Предметом дослідження є онтологічна система проєктування систем ситуаційного управління.

РОЗДІЛ 1.
МОДЕЛЮВАННЯ ЗНАНЬ ТА ОНТОЛОГІЧНИЙ ПІДХІД ДО ПРОЄКТУВАННЯ

1.1. Моделі представлення знань

Однією з основних проблем, характерних для систем, заснованих на знаннях, є проблема представлення знань. Це обумовлено тим, що форма подання знань впливає на характеристики та властивості системи.

Для обробки знань про реальний світ з використанням комп'ютера, здійснюється розробка моделей знань. При цьому необхідно враховувати такі фактори як однорідність подання, що забезпечує ефективне використання механізму логічного виведення, керування знаннями, простоту розуміння як розробниками, так і користувачами системи.

Найбільш поширеними моделями представлення знань є [1]:

- продукційні;
- логічні;
- фреймові;
- семантичні мережі;
- онтології.

В продукційних системах знання представляються у вигляді сукупності спеціальних інформаційних одиниць. В загальному випадку продукційна система включає наступні компоненти: базу даних, що містить множину фактів; базу правил, що містить набор продукцій; інтерпретатор (механізм логічного виводу) або правила роботи з продукціями.

Логічні моделі представлення знань реалізуються засобами логіки предикатів. Предикатом називається функція, що приймає тільки два значення – істинне та хибне – і призначена для виразу властивостей об'єктів або зв'язків між ними.

Фрейм частіше за все визначають як структуру даних для представлення стереотипних ситуацій. Модель представлення знань на основі фреймів використовує концепцію організації пам'яті, розуміння та навчання людини.

Фрейм – це одиниця представлення знань, деталі якої можуть змінюватись з поточною ситуацією. Фрейм у будь-який момент може бути доповнений різноманітною інформацією, що стосується способів застосування даного фрейма, наслідків цього застосування та інше.

Семантична мережа описує знання у вигляді мережевих структур. В якості вершин мережі виступають поняття, факти, об’єкти, події та інше, а в якості дуг мережі – відношення, якими пов’язані вершини між собою. Семантичні мережі часто розглядають як загальний формалізм для представлення знань.

Онтологія описує основні концепції (положення) предметної області і визначає відносини між ними. Процес побудови онтологій складається зі створення таких блоків:

- класів та їх властивостей (classes, properties).
- властивостей кожної концепції, що описують різні функціональні можливості і атрибути концепції (слоти (slots), іноді звані ролі).
- обмеження по слотам (також відомих як аспекти / грані (slot facets), іноді звані обмеження ролей). Онтологія разом з множиною індивідуальних екземплярів класів складають базу знань.

З інших методів представлення знань популярністю користується представлення знань по прикладам. Працюючи з системою такого типу, користувач задає їй декілька прикладів розв’язків задач з актуальної предметної області. На основі цих прикладів система самостійно будує базу знань, яка потім використовується для рішення інших задач.

1.2. Характеристика онтологічного підходу до проєктування

Онтологічний підхід до проектування будь-яких класів складних систем (у тому числі й комп’ютерних систем) дає можливість провести чітку та ієрархічну декомпозицію процесів проєктування будь-якої системи заданого призначення на такі проектні дії, багато з яких можуть виконуватись паралельно і кожна з яких може бути виконана локально, тобто не виходячи межі конкретних онтологій [2], [3], [4]. Таким чином, ієрархічній системі проектних дій ставиться у відповідність

ієрархічна система онтологій, у рамках якої відповідні проектні дії можуть бути виконані. Очевидно, що це суттєво прискорює проектну діяльність шляхом її розпаралелювання та локалізації області пошуку рішення при виконанні кожної проектної дії. Для досягнення поставленої мети необхідно вирішувати такі задачі.

1. Підвищити ефективність проектування комп'ютерних систем на основі загальної (комплексної, інтегрованої, цілісної) технології проектування комп'ютерних систем, у межах якої було б узгоджено всі необхідні спеціальні предметні технології, тобто було б гарантовано сумісність проектних рішень, які можна одержати у межах цих технологій. Сумісність таких проектних рішень – це сумісність різних видів компонентів комп'ютерних систем, які можуть бути продуктами розробки у випадку різних і незалежних один від одного колективів розробників. Зокрема, має бути гарантована сумісність різних видів знань, що входять до складу бази знань, різних моделей розв'язання задач, що використовуються інтелектуальним розв'язувачем задач, сумісність різних моделей розуміння зовнішньої інформації, яка надходить до інтелектуальної системи по різних каналах, у різній формі та на різних мовами.

2. Створити загальну інтегровану технологію проектування інтелектуальних систем на основі загальної формальної теорії комп'ютерних систем.

3. Знизити трудомісткість не тільки при початковій розробці комп'ютерних систем, але й у процесі постійного вдосконалення (модернізації, реінжинірингу) під час експлуатації на основі технології проектування таких систем.

4. Забезпечити простоту та зрозумілість формальних моделей комп'ютеризованих систем, які є продуктами (результатами) їх проектування, не лише інтерпретаторами, які використовуються для їх реалізації на різних платформах, а й усіма розробниками подібних моделей.

5. Створити єдиний універсальний принцип, «кістяк», який дозволяє будувати ієрархічні багаторівневі моделі представлення та обробки знань будь-якої конфігурації як основи розробки моделей представлення знань і моделей обробки знань для різних інтелектуальних систем. Для моделей представлення знань необхідно мати можливість переходу від знань до метазнань, від метазнань до

метаметазнань тощо, і, зокрема, від опису дій (як внутрішніх, так і зовнішніх) до опису дій будь-якого вищого рівня. Для моделей обробки знань необхідно мати можливість переходу від агентів, здатних виконувати дії одного рівня, до колективних агентів, здатних виконувати дії будь-якого вищого рівня.

В основі онтологічного проектування будь-яких систем лежить розробка цілого комплексу взаємопов'язаних предметних областей та відповідних їм онтологій. Основна перевага онтологічного підходу до проектування - це суттєве підвищення гнучкості як самих розроблюваних систем, так і самої проектної діяльності завдяки чіткому поділу тих проектних дій, які можуть виконуватися локально в рамках відповідних предметних областей і не вимагати жодного узгодження з проектними діями в інших предметних областях та тих проектних дій, які мають бути узгоджені між різними предметними областями, але процедура узгодження яких чітко визначена. Гнучкість та чіткість декомпозиції онтологічних моделей проектованих систем є основою для ефективної організації колективної проектної діяльності.

База знань являє собою модель понять і взаємозв'язків між ними, яка грає роль інтелектуально розвинутого тезаурусу предметної області, що забезпечує:

- універсальність постановок задач;
- гармонізація різних компонент засобів автоматизації при запозиченні та спадкуванні інформації по ходу аналітичних процедур;
- ефективну комунікацію між представниками різних професійних і відомчих груп;
- можливість інтелектуального моніторингу стану справ в організації, базуючій на комплексних моделях, що включають об'єкти, що належать різним функціональним областям і розглядаються одночасно в різних функціональних ракурсах.

З урахуванням розглянутих вище функціональних вимог і викликів по відношенню до засобів ситуаційного управління може бути використаний наступний підхід до формування структур знань, заснований на досвіді попередніх досліджень.

Основу менеджменту знань становить комплексна онтологічна модель:

$$M = <BO , V> ,$$

де *BO* – множина базових онтологій; *V* – множина корпоративних точок зору, зумовлених участю їх носіїв в ділових процесах.

Підсистема обробки знань (ПОЗ) у загальному вигляді представляється структурою, представленою на рис. 1.1.

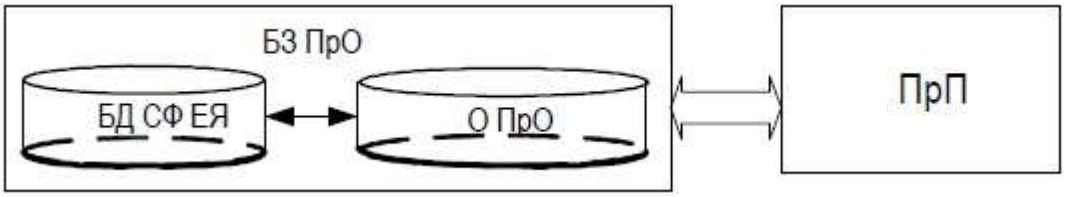

Рис. 1.1. Узагальнене структурне представлення ПОЗ

На рисунку прийняті наступні позначення:

БЗ ПрО – база знань предметної області;

БД СФ ЕЯ – база даних суджень і фактів із заданої ПрО природною мовою;

Про ПрО – онтологія ПрО;

ПрП – прикладний процесинг.

Сучасна теорія і практика проектування баз знань (БЗ) орієнтовані на широке використання існуючих знань про проблемну область (ПО). В результаті створені БЗ відображають лише статичну картину "Вчора ПО". Знання-орієнтовані вимоги до ССУ, що розвиваються, визначають необхідність мати в складі системи механізми інформаційної та операційної підтримки аналізу результатів власної діяльності, які пропонується розглядати як механізми проблемної орієнтації БЗ ССУ, що самостійно розвиваються, з урахуванням проблем "сьогодні – завтра ПО".

Виходячи з цього, в основу моделей проблемно-орієнтованої бази знань пропонується покласти:

- перелік факторів, що характеризують проблемну область, проблему в процесі нормального функціонування об’єкта;
- перелік станів, у яких може перебувати проблемна область в процесі нормального функціонування об’єкта;
- перелік факторів, під впливом яких може змінитися поточний стан проблемної області, перехід її в проблемну ситуацію (ПС), порушивши при цьому нормальне функціонування об’єкта;

- перелік проблемних станів (прогнозованих), в які може перейти об’єкт під впливом руйнівних факторів;
- перелік (передбачуваних) сценаріїв, де кожен сценарій – це перелік деяких дій впливу на об’єкт для недопущення, запобігання проблемної ситуації або впливу на проблемну ситуацію для виведення об’єкта з неї і повернення його в режим нормального функціонування;
- набір елементів знань (декларативних, прецедентних, алгоритмічних тощо), які можуть допомогти усвідомити ситуацію і прийняти відповідне правильне рішення, вибрати той чи інший сценарій виходу з проблемної ситуації і представити А (<ОМ, Зн>) як поточну апроксимацію С (ПО) у вигляді 6-рівневого ієрархічного дерева (рис. 1.2):

$$A(< OM, Зн >) = \bigcup_{i=1}^{6} A_i(< OM^i, Зн^i >).$$

Змінені сценарії виходу з відповідної ПС, після тестування і підтвердження практикою ефективності прийнятого рішення, розглядаються як нові елементи знань і викликають хвилю консолідації структурних змін в ССУ.

Перелік факторів у даному випадку являють собою теж елементи знань, оскільки для них задається шкала допустимих значень регламентації функціонування об’єкта.

Якщо прийняти такий підхід до моделювання ПрОрБЗ, то можна сформувати технологічні кроки з її створення, ведення та використання, тобто запропонувати методологічні підходи до створення ПрОрБЗ, що забезпечують інтелектуалізацію розвиваються ССУ.

Таким чином, предметно-орієнтована база знань ПС надає можливість відслідковувати стан ПО за допомогою моніторингу значень факторів, що ідентифікують стан ПС. При виявленні відхилень значень цих факторів від їх нормальних значень можна, автоматично рухаючись по дереву, знайти той сценарій, використання якого може забезпечити запобігання ПС або ж її ліквідацію.

У процесі реалізації того чи іншого сценарію іноді доводиться вносити зміни в сценарії запобігання або ліквідації ПС. Аналіз реалізації сценарію в свою чергу може привести до виявлення нових факторів двох типів: як описують регламентне

функціонування ПО, так і тих, які привели до виникнення ПС, що призводить до зміни онтологічної схеми бази знань.

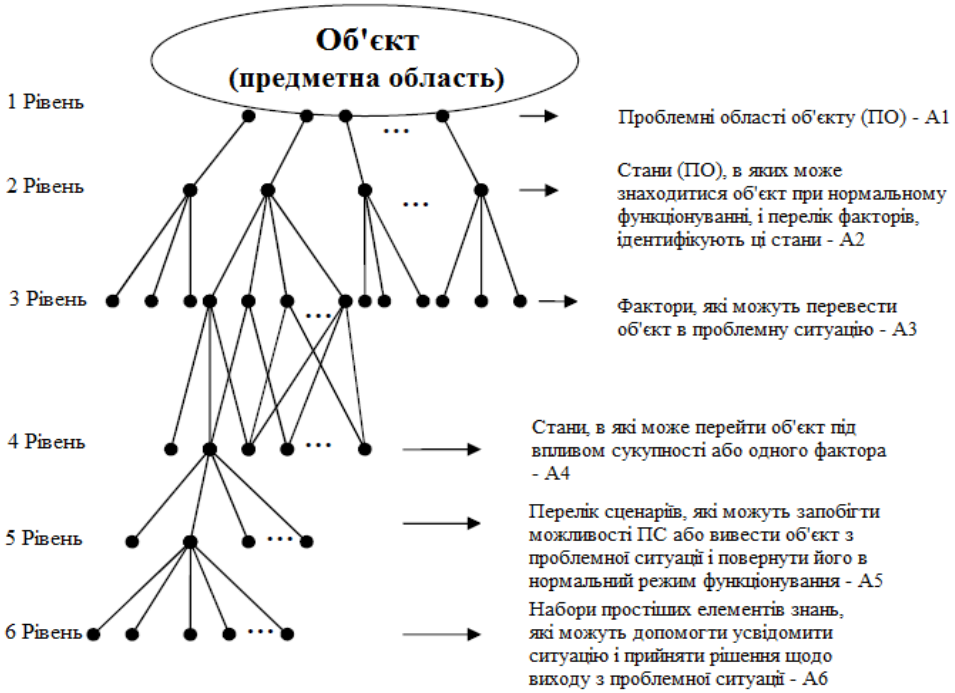

Рис. 1.2. Ієрархічна структура ПрОрБЗ

Методологічні аспекти створення проблемно-орієнтованих баз знань, що дозволяють оперативно розвивати інтелектуальні можливості ССУ:

- структуризація предметної області (об’єкта) з точки зору функціонування її проблемно-орієнтованих областей;
- визначення наборів показників-факторів, за значеннями яких можна визначити стан кожної з проблемних областей і шкал (інтервалів) їх значень, що ідентифікують нормальне регламентне функціонування об’єкта, з точки зору кожної з певних областей;
- визначення сукупності ПС, в яких може перебувати проблемна область в процесі нормального регламентного її функціонування;
- визначення переліку факторів, які можуть зруйнувати нормальний стан проблемної області, створити в ній проблемну ситуацію і порушити при цьому нормальне функціонування об’єкта;
- визначення сукупності ПС, в яких може опинитися ПО, які можуть виникнути під впливом одного або декількох руйнівних чинників;

- розробка набору сценаріїв, стратегій (плану дій), реалізація кожного з яких може запобігти виникненню ПС або ж ліквідувати її наслідки;
- визначення набору допоміжних елементарних знань, елементів традиційних баз знань, які можуть допомогти усвідомити ситуацію і прийняти відповідне рішення - вибрати той чи інший сценарій запобігання ПС або її ліквідації;
- побудова дерева проблемно-орієнтованої бази знань як ситуативної моделі ПО.

Висновки до розділу

Досліджено поширені моделі представлення знань, проаналізувано особливості побудови систем ситуаційного управління, охарактеризовано онтологічний підхід до проєктування систем ситуаційного управління, а також визначено структуру знань в контексті проєктування таких систем.

РОЗДІЛ 2.
ОНТОЛОГІЧНА МОДЕЛЬ ПРОЄКТУВАННЯ БАГАТОРІВНЕВИХ СИСТЕМ

Будь-який процес проектування загалом складається з ключових підпроцесів: розпізнавання потреб, визначення проблеми, розробка вимог, синтез, аналіз та оптимізація, оцінка та реалізація. Підпроцеси процесу проектування для системи ситуаційного управління (ССУ) можуть базуватися на популярній моделі ситуаційної обізнаності [5]. Незважаючи на те, що усвідомлення ситуації є основним етапом прийняття рішень в процесах проектування ССУ, виконання проектних рішень та оцінка результатів проектування також є важливими етапами. Процес проектування ССУ включає аналіз потреб, розробку вимог, синтез архітектури ССУ, аналіз, оптимізацію, оцінку, подання архітектури ССУ.

Тому головною проблемою конвергентної ССУ є гармонізація різних ІТ всередині цільової ССУ. Конвергенція може бути досягнута на основі ієрархічного модельного підходу шляхом створення системи моделей, які описують різні аспекти цільової КС з використанням відповідної моделі знань. Основною метою створення системи моделей є запобігання розривів у ланцюгах створення цінностей проектованої ССУ, а отже, і в ланцюгах створення цінності ССУ у цілому.

Відповідно до моделі архітектури ССУ необхідно описати всі її складові та ССУ в цілому адекватними моделями з максимально можливим рівнем формалізму – від словесного до програмного. Процес створення моделей для компонентів ССУ та ССУ в цілому є процесом збільшення її цінності через етапи проектування. Перелік етапів проектування ССУ та типи моделей, які можна використовувати, наведено в таблиці 2.1.

Різноманітність і вихідна невизначеність набору послуг, необхідних для реалізації конкретної моделі предметної діяльності, вимагають широкого вибору функціональних можливостей сервісів, їх доступності, гнучкості та автономності. Відповідність цим вимогам може бути забезпечена на основі агентно-

орієнтованого підходу. Таким чином, предметно-орієнтовану інформаційну систему можна представити у вигляді багатоагентної системи (МАС) [6].

Таблиця 2.1 – Стадії проєктування ССУ та види застосовних моделей

EA design stage	Вид моделі					
	Вербальна	Функціональна	Структурна	Процесна	Даних	Аналітична
Ситуаційний аналіз	+ [a]	-	-	+	+	+
Інженерія вимог	+	-	-	-	+	-
Синтез	-	+	+	+	+	+
Аналіз	-	+	+	+	+	+
Оптимізація	-	+	+	+	+	+
Оцінка	+	+	+	+	+	+
Реалізація	+	+	+	+	+	-

’+’ застосовна,’ –’ не застосовна

Різноманітність і вихідна невизначеність набору послуг, необхідних для реалізації конкретної моделі предметної діяльності, вимагають широкого вибору функціональних можливостей сервісів, їх доступності, гнучкості та автономності. Відповідність цим вимогам може бути забезпечена на основі агентно-орієнтованого підходу. Таким чином, предметно-орієнтовану інформаційну систему можна представити у вигляді багатоагентної системи (МАС) [6].

Враховуючи вищезазначені вимоги, формально багаторівневу систему можна визначити як кортеж [7].

$$S =< L, I^0, C^S, T > \qquad (1)$$

де $L=\{l_1, l_2, \dots l_n\}$ множина рівнів системи, $I^0=\{i^0_1, i^0_2, \dots i^0_n\}$ – множина зовнішніх інтерфейсів системи, $C^s=\{c^s_1, c^s_2, \dots c^s_n\}$ – множина функцій управління системою в цілому, T – цільова функція (функція призначення) системи (може бути задано неявно як метафункція у вигляді набору вимог до інших компонентів системи).

Кожен рівень, у свою чергу, може бути представлений кортежом

$$l_j = <U^l, I^l, C^a, C^l> \quad (2)$$

де U^l – множнина сервісів рівня l_j, I^l – множина інтерфейсів рівня l_j, C^a– множина функцій управління застосунком рівня l_j, C^l – множина функцій управління рівнем l_j.

Кожен сервіс u^l_k реалізується як композиція функцій певної підмножини F^i_k множини функцій F^l, які реалізуються на рівні l_j .

Реалізація описаної моделі включає два аспекти:

- семантичний (змістовний);
- архітектурні (конструктивно-функціональні).

Семантика кожного рівня природно описується за допомогою семантичних мереж, які можна використовувати для представлення онтологій.

Розширення концепції інтелектуального МАС для реалізації багаторівневих систем полягає у визначенні набору агентів кожного рівня, виходячи з функціональних вимог для цього рівня. Віднесення агента до певного рівня визначає тип агента. Таким чином, багаторівневий МАС можна формально описати як множину агентів A різних типів:

$$A = \{A^l\} \quad (3)$$

де A^l – множина агентів рівня l.

На основі семантичних і функціональних вимог до кожного рівня, множина агентів A^l рівня l визначається відображенням семантики (онтології) O^l за рівнем його функціональності F^l.

$$A^l : O^l \times F^l \quad (4)$$

В результаті набір послуг U^l кожного рівня буде визначатися функцією управління агентом C^l відповідного рівня:

$$U^l = C^l(A^l) \quad (5)$$

Деякі рівні багаторівневої системи можуть бути представлені не тільки програмними агентами, а й іншими об’єктами та суб’єктами системи, такими як адміністратори, розробники, керівники команд, персонал, регламент тощо, які

мають різну фізичну реалізацію (природну або технологічну), але включені в схеми управління та взаємодії системних агентів.

Модель предметної області знань визначається архітектурною моделлю ССУ та її контекстом взаємодії з навколишнім середовищем. Архітектура ССУ забезпечує реалізацію процесу ситуаційного управління як композиції мотивацій, знань, можливостей, ресурсів та обмежень.

Побудова та використання ССУ у різних сферах застосування вимагають розробки методів та засобів конвергенції інформаційних технологій для здійснення адекватного ситуаційного управління в цільовій проблемній області. Конвергенція визначається як глибока інтеграція знань, ресурсів і всієї раціональної людської діяльності для досягнення спільної мети, здатність відповідати на нові питання про зміну відповідної фізичної або соціальної екосистеми.

Формальний опис предметної області СУ, для якої створюється проблемно-орієнтований ССУ, являє собою ієрархію понять (тверджень) і функціональних перетворень, якими будуть керувати користувачі. Формальний опис предметної області також повинен містити узагальнений опис моделі процесу СУ. Композиція технологій СУ повинна здійснюватися з урахуванням архітектури ССУ.

Висновки до розділу

Проведено дослідження особливостей онтологічної моделі проєктування багаторівневих систем, вивчено стадії проєктування таких систем, формується архітектурна модель для проєктування систем ситуаційного управління.

РОЗДІЛ 3.
АРХІТЕКТУРА КІБЕРНЕТИЧНИХ СИСТЕМ

Слово кібернетика грецького походження. Воно зустрічається ще в працях грецького філософа Платона, що жило близько 2400 років тому, і означає мистецтво керування кораблем. Ампер у 1832 р. назвав кібернетикою науку про керування державами, що, на його думку, варто було б створити. Норберт Вінер у 1948 р. у книзі «Кібернетика або керування у тварині і машині» виклав основні ідеї нової науки про загальні закони керування складними системами.

Кібернетика – наука про загальні закони процесів організації, керування і переробки інформації в складних системах різної фізичної природи: машинах, технічних пристроях, живих організмах.

Виникнення науки кібернетики обумовлений науково-технічною революцією, створенням складних самоврядних верстатів, автоматичних ліній, обчислювальних машин. Значну роль зіграло також розвиток нейрофізіології - науки про системи керування і регулювання в живому організмі. Створення науки про загальні закони керування викликано також розвитком наук про керування організаційними системами, зокрема, державними, економічними, суспільними та іншими системами. Розвиток кібернетики був би неможливим без прогресу електроніки та створення засобів комп'ютерної техніки.

Предметом кібернетики є кібернетичні системи, які представляються як упорядковані сукупності взаємодіючих складових (артефактів, компонентів систем), об'єднаних виконанням визначеної функції на основі обміну інформацією.

Компонентами кібернетичної системи можуть бути об'єкти різної фізичної природи: неживі предмети, живі та штучні (технічні) об'єкти, процеси, явища тощо. Наприклад, компонентами комп'ютера є його блоки; мозку – нейрони; колективу – співробітники, члени колективу.

Архітектура організації (enterprise architecture, EA) широко використовується для опису різноманітних кібернетичних організаційних систем по всьому світу і зазвичай асоціюється з популярними еталонними моделями (фреймворками) EA (TOGAF, Zachman, FEAF тощо). Аналіз артефактів EA, які використовуються в

успішній практиці ЕА, показує, що поняття ЕА можна краще пояснити за допомогою вдосконаленої таксономії, що визначає шість загальних типів артефактів ЕА: міркування, стандарти, бачення, ландшафти, контури та проекти[8].

Аналіз артефактів ЕА, виявлених в організаціях, які успішно практикують ЕА, показує, що всі артефакти ЕА можна згрупувати в шість загальних типів на основі того, що описують артефакти ЕА та як артефакти описують ЕА.

По-перше, всі артефакти ЕА можна класифікувати на основі об'єктів їх опису (ЩО вони описують), від більш загальних до більш конкретних, на правила, структури та зміни. Правила описують загальні глобальні правила, що визначають організацію або її підрозділи, Структури описують структури організації високого рівня або її частин, тоді як Зміни описують конкретні запропоновані зміни в організації.

По-друге, усі артефакти ЕА можна класифікувати з використанням термінології їх опису (ЯК вони описуються) на визначені діяльністю та IT-орієнтовані. Артефакти ЕА, орієнтовані на діяльність, зазвичай нейтральні щодо технологій і використовують ділову термінологію (гроші, клієнти, можливості, цілі діяльності, конкурентні переваги тощо), тоді як артефакти ЕА, орієнтовані на IT, зазвичай є суто технічними та використовують термінологію, специфічну для IT (системи, програми, бази даних, платформи, мережі тощо).

Перетин двох ортогональних класифікацій, описаних вище, створює таксономію з шістьма загальними типами артефактів ЕА: домовленості (припущення), стандарти, бачення, ландшафти, плани та проєкти.

Архітектура складових систем ССУ системи визначається кортежем архітектурних артефактів:

$$A = \langle P, C, R, O, T, G, L, S \rangle,$$

де P – принципи побудови системи, C – модель спроможностей діяльності, R – плани розвитку (дорожня карта), O – оцінки управлінських рішень, T – еталонні технологічні моделі, G – настанови, інструкції, керівництва, L – архітектурні діаграми (ландшафт системи), S – проєктні рішення.

Проєктна діяльність по створенню ССУ повинна поєднувати (інтегрувати) знання, що стосуються опису архітектури організації, доступних технологій та спроможностей організації, та вимог до конкретної реалізації ССУ. Спираючись на підходи інженерії систем та управління життєвим циклом систем [9] проект по створенню ССУ повинен передбачати розробку процесів управління діяльністю в рамках процесі управління життєвим циклом архітектури організації в якій впроваджується ССУ.

Функціонування ССУ відбувається в умовах змінюваності ситуацій в рамках моделі ситуаційного управління на основі перцептивного циклу когнітивної (пізнавальної) діяльності. Ситуативна змінюваність умов функціонування, потребує відповідної ситуаційної адаптації архітектури ССУ до цих умов. Така адаптація може бути проведена на основі використання моделей знань предметної області в програмно-керованій архітектурі цільової системи в рамках реалізації концепції «архітектура як код» (architecture as code, AaC). Концепція AaC є розвитком та узагальненням концепцій програмного управління конфігураціями інформаційно-комунікаційних та програмно-насичених систем, яка отримала поширення в рамках технологій хмарних обчислень у вигляді IaC, IaaS, PaaS, SaaS тощо.

Прийняття проєктних рішень повинно узгоджуватись з призначенням та архітектурою цільової системи з точок зору на можливі ризики стосовно критичних активів на відповідних етапах її життєвого циклу.

Адаптований відповідно до положень стандарту ISO/IEC/IEE 15288 [10] життєвий цикл ССУ включає наступні стадії:

- задум;
- розробка;
- побудова (виробництво);
- використання та підтримка використання;
- модернізація;
- виведення з експлуатації.

Стадія формування задуму ССУ включає в себе процеси дослідження предметної сфери її застосування та визначення основних підходів до її розробки та побудови. Знання, отримані на цьому етапі повинні відображати специфіку

виділеного (конкретного) тематичного домену предметної сфери у вигляді когнітивної мережі основних концептів. Семантика такої мережі може бути описана семантичною моделлю Кріпке (не плутати зі структурою (моделлю) Кріпке для недетермінованих скінчених автоматів) [11]:

$$\langle W, R, \Vdash \rangle \qquad (1)$$

де $\langle W, R\rangle$ - шкала (фрейм) Кріпке на множині вузлів (світів) W з відношеннями R (множиною стрілок або упорядкованих пар) на W: $R \subset W \times W$; $\Vdash$ – символ (відношення) істинності (оцінки, виконання). Наприклад, модальна формула $w \Vdash A$ означає що "w задовольняється A", "A виконується у w", або "w визначає (спонукає, викликає) A". Семантика Кріпке (реляційна семантика, фрейм-семантика) дозволяє створювати логічні моделі ситуацій в базисі інтуіціоністської (конструктивної) та модальних логік [12].

Після визначення семантики системи у вигляді моделі знань (1) проводиться розробка системи. Розробка ССУ як складної системи включає в себе такі групи процесів (рис. 3.1):

- аналіз потреб та моделювання;
- інженерія вимог (визначення та управління вимогами);
- побудова архітектури та затвердження.

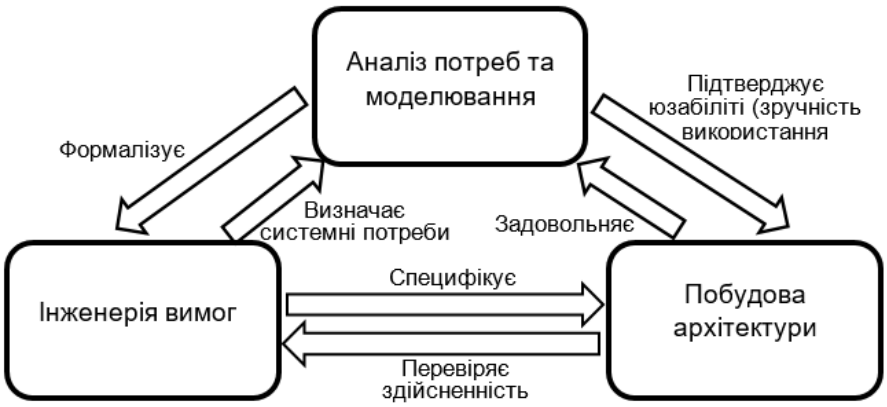

Рис. 3.1. Процеси розробки системи

Задум цільової системи та її ССУ представляється у вигляді переліку потреб та моделей на основі аналізу яких проводиться формалізація вимог у відповідності з потребами. Виконання процесів інженерії вимог забезпечує визначення

системних потреб та специфікацію архітектури ССУ у відповідності з архітектурою цільової системи та формалізованих моделей. Зокрема, одним з результатів інженерії вимог є затверджена політика безпеки цільової системи, що зберігається у БЗ ССУ. Побудова архітектури ССУ специфікується формалізованими вимогами до системи з БЗ ССУ та підтверджуються результатами аналізу на основі моделей системи та тестових експлуатаційних задач.

Експлуатація ССУ являє по суті ситуаційне управління, пов'язане з забезпеченням захищеності цільової системи в умовах загроз інформаційній безпеці. Ситуацію можна визначити як *усвідомлені знання суб'єкта про динаміку навколишнього середовища, що представлені певними видами інформаційних повідомлень та є основою для побудови обґрунтованої інтерпретації послідовності зміни станів (динаміки) світу (предметної області) з певної точки зору* [13]. Ситуаційне управління визначається як метод управління на основі використання множини концепцій, моделей, доступних технологій для розпізнавання, пояснення, впливу і передбачення ситуацій, які виникли або можуть виникнути у динамічних система на протязі наперед визначеного часу роботи [14].

Прийняття рішень в ході здійснення ситуаційного управління можна описати моделями на основі проектного підходу і процесів його реалізації. Важливим елементом застосування такого підходу є управління життєвим циклом проекту. Ситуаційне управління (СУ), як цілеспрямована діяльність, реалізується через послідовність процесів: усвідомлення/оцінка ситуації, визначення цілей СУ, розробка планів досягнення цілей СУ, прийняття рішень щодо планів їх та затвердження, виконання затвердженого плану СУ, моніторинг виконання плану СУ, коригування плану виконання СУ, оцінка/збереження результатів виконання СУ. Послідовність процесів СУ показана на рис.3.2.

Рис. 3.2. Процеси ситуаційного управління в ССУ

При ефективному управління проектом потрібно враховувати чотири основні види обмежень, що впливають на виконання проекту: ресурси, бюджет, час і

масштаб. Власне бюджет і час також є ресурсами, але через свою специфіку і визначальний вплив на проектну діяльність вони розглядаються окремо. Загалом ресурси проекту слід розглядати як систему активів, направлених на забезпечення виконання проекту з визначеними показниками якості. Отже, узагальнена система ресурсів проекту включає наступні види ресурсів: зацікавлені сторони і персонал, бюджет, матеріали, обладнання, інфраструктуру, час.

Деталізована послідовність процесів ситуаційного управління в рамках ССУ має такий вид:

[визначення цілей] → [оцінка (диференціація) середовища] → [мотивація досягнення цілі] → [визначання доступних ресурсів] → [визначення підцілей] → [визначення плану (шаблону) досягнення цілі] → [виконання плану] → [виявлення відхилень] → [коригування плану] → [оцінка результатів діяльності].

Виконання ситуаційних проектів визначається випадковими подіями, на які повинна реагувати ССУ. Як правило ці події пов’язані з інцидентами безпеки щодо цільової системи. Реакція на такі події в рамках ситуаційного проекту реагування на події може носити типовий і нетиповий характер.

Проектна діяльність, пов’язана з експлуатацією ССУ складається з реалізацією проектів таких типів:

- регулярні (регламенті);
- ситуаційні (подійні);
- модернізаційні (удосконалювальні).

Регулярні (регламентні) проекти ситуаційного управління визначаються на основі аналізу характерних ознак (сигнатури) інциденту безпеки та пошуку відповідного сценарію ситуаційного управління в БЗ ССУ. Ситуаційні проекти реалізуються в рамках сценарію ситуаційного управління (рис.3.2). Модернізаційні проекти реалізуються в рамках запланованих змін та удосконалень, передбачених у моделі життєвого циклу цільової системи.

Висновки до розділу

Застосування системний підходу до розробки та використання ССУ поєднує архітектурну, процесну та проектну точки зору на основі семантичного опису предметного домену цільової системи у вигляді формалізованої моделі знань. Відповідно до моделі життєвого циклу цільової системи формується модель життєвого циклу ССУ з прив’язкою до архітектури цільової системи. Цілісність такого поєднання забезпечується розробкою та підтримкою в актуальному стані моделі знань системи.

РОЗДІЛ 4.
МОДЕЛЬ ЗНАНЬ АРХІТЕКТУРИ КІБЕРНЕТИЧНИХ СИСТЕМ

4.1. Редактори онтологій

Онтологія – формальний явний опис понять аналізованої предметної області (класів, іноді їх називають поняттями), властивостей кожного поняття, що описують різні властивості і атрибути поняття (слотів (іноді їх називають ролями чи властивостями)), і обмежень, накладених на слоти (фацетов, іноді їх називають обмеженнями ролей).

При створенні онтологій (як і при проектуванні програмного забезпечення або написанні електронного документа) доцільно користуватися відповідними інструментами. Будемо називати інструментальні програмні засоби, створені спеціально для проектування, редагування та аналізу онтологій, редакторами онтологій.

Основна функція будь-якого редактора онтологій полягає в підтримці процесу формалізації знань та поданні онтології як специфікації (Точного і повного опису).

У більшості, сучасні редактори онтологій надають засоби «кодування» (в сенсі опису) формальної моделі у тому чи іншому вигляді. Деякі дають додаткові можливості по аналізу онтології, використовують механізм логічного висновку. Останнім часом кількість загальнодоступних редакторів онтологій перевищило 100. Але зрідка можна зустріти універсальний й водночас корисний засіб.

4.1.1. Редактор онтології Ontolingua

Система Ontolingua була розроблена в KSL (Knowledge Systems Laboratory) Стенфордського університету і стала першим інструментом інженерії онтологій. Вона складається з сервера і мови подання знань.

Сервер Ontolingua організований у вигляді набору онтологій, що відносяться до Web-додатків, які надбудовуються над системою подання знань Ontolingua. Редактор онтологій – найбільш важливий додаток сервера Ontolingua є Web-додатком на основі форм HTML. Крім редактора онтологій Сервер Ontolingua включає мережевий додаток Webster (одержання визначень концептів), сервер

OKBC (доступ до онтології Ontolingua по протоколу OKBC) і Chimaera (аналіз, об’єднання, інтегрування онтологій). Всі додатки, крім сервера OKBC, реалізовані на основі форм HTML. Система подання знань реалізована на Lisp.

Сервер Ontolingua також надає архів онтологій, що включає велику кількість онтологій різних предметних областей, що дозволяє створювати онтології з вже існуючих. Сервер підтримує спільну розробку онтології декількома користувачами, для чого використовуються поняття користувачів і груп. Система включає графічний браузер, що дозволяє переглянути ієрархію концептів, включаючи екземпляри. Ontolingua забезпечує використання принципу множинного спадкоємства і багатий набір примітивів. Збережені на сервері онтології можуть бути перетворені в різні формати для використання іншими додатками, а також імпортовані з ряду мов в мову Ontolingua.

4.1.2. Редактор онтології OntoEdit

OntoEdit спочатку був розроблений в інституті AIFB (Institute of Applied Informatics and Formal Description Methods) Університету Karlsruhe (зараз комерціалізувати Ontoprise GmbH) виконує перевірку, перегляд, кодування і модифікацію онтологій. В даний час OntoEdit підтримує мови вистави: FLogic, включаючи машину виведення, OIL, розширення RDFS і внутрішню, засновану на XML, сериализацию моделі онтології використовуючи OXML – мова представлення знань OntoEdit (OntoEdit’s XML-based Ontology representation Language). До достоїнств інструмента можна віднести зручність використання; розробку онтології під керівництвом методології і за допомогою процесу логічного висновку; розробку аксіом; розширювану структуру за допомогою плагінів, а також дуже хорошу документацію.

Так само як і Protégé, OntoEdit – автономний Java-додаток, який можна локально встановити на комп’ютері, але його коди закриті. Архітектура OntoEdit подібна Protégé.

Існує дві версії OntoEdit: вільно розповсюджується OntoEdit Free (обмежена 50 концептами, 50 відносинами і 50 екземплярами) і ліцензована OntoEdit Professional (немає обмежень на розмір). Природно, що OntoEdit Professional має

більш широкий набір функцій і можливостей (наприклад, машину виведення, графічний інструмент запитів, більше модулів експорту та імпорту, графічний редактор правил, підтримка баз даних JDBC тощо).

4.1.3. Редактор онтології OilEd

OilEd – автономний графічний редактор онтологій, розроблений в Манчестерському університеті в рамках європейського IST проекту On-To-Knowledge. Інструмент заснований на мові OIL (зараз адаптований для DAML + OIL, в перспективі – OWL), який поєднує в собі фреймову структуру і виразність дескриптивной логіки (Description Logics) з сервісами міркування, що дозволило забезпечити зрозумілий і інтуїтивний стиль інтерфейсу користувача та переваги підтримки міркування (виявлення логічно суперечливих класів і прихованих відносин підкласу).

З недоліків можна виділити відсутність підтримки примірників. Існуюча версія не забезпечує повну середу розробки - не дозволені розробка онтологій великого масштабу, міграція та інтеграція онтологій, контроль версій і т.д. OilEd можна розглядати як "NotePad" редакторів онтологій, що пропонує достатню функціональність, щоб дозволити користувачам будувати онтології і продемонструвати, як можна використовувати механізм міркування FaCT для перевірки онтології на несуперечливість.

Останнім часом спостерігається зростання популярності редактора OilEd. Він використовується як для навчання, так і для дослідження. Інструмент вільно поширюється по загальнодоступній ліцензії GPL.

4.1.4. Редактор онтології Protégé

Редактор підтримує формалізми і формати представлення.

Під формалізмом розуміється теоретичний базис, який лежить в основі способу представлення онтологічних знань. Прикладами формалізмі можуть служити логіка предикатів (First order logics – FOL), дескриптивна логіка, фреймові моделі (Frames), концептуальні графи тощо. Формалізм, використовуваний редактором, може істотно впливати не тільки на внутрішні структури даних, але і визначати формат представлення або навіть користувальницький інтерфейс.

Формат представлення онтології задає вид зберігання та спосіб передачі онтологічних описів. Під форматами маються на увазі мови представлення онтологій: RDF, OWL, KIF, SCL. Таким чином, деяка формальна модель представляється у формалізмі FOL і може бути виражена засобами мови KIF. Редактори онтологій звичайно підтримують роботу з кількома формалізмами і форматами представлення, але часто тільки один формалізм є «рідним» (native) для даного редактора.

Важливою характеристикою є функціональність редактора, тобто безліч сценаріїв його використання. Базовий набір функцій забезпечує:

- роботу з одним або більше проектами;
- збереження проекту у потрібному формалізмі та форматі (Експорт);
- відкриття проекту;
- імпорт з зовнішнього формату;
- редагування метаданих проекту (в широкому сенсі: від налаштування форм редагування і представлення даних, до підтримки версій проекту)

Редагування онтології. Набір можливих дій зазвичай включає створення, редагування, видалення понять, відносин, аксіом і інших структурних елементів онтології, редагування таксономії. До додаткових можливостей редакторів відносять підтримку мови запитів (для пошуку нетривіальних тверджень), аналіз цілісності, використання механізму логічного висновку, підтримка користувацького режиму.

Редактор Protégé. З моменту його створення Protégé багато років використовувався експертами в основному для концептуального моделювання в області медицини [15].

Останнім часом його стали використовувати в інших предметних областях. Зокрема при створенні онтологій для Semantic web. Спочатку єдиною моделлю знань підтримуваної Protégé була фреймова модель. Цей формалізм зараз є «рідним» для редактора, але не єдиним. Protégé має відкриту, легко розширювану архітектуру і крім фреймів підтримує всі найбільш поширені мови представлення знань (SHOE, XOL, DAML + OIL, RDF / RDFS, OWL). Protégé підтримує модулі

розширення функціональності (plug-in). Розширювати Protégé для використання нової мови простіше, ніж створювати редактор цієї мови «з нуля». Модель знань Protégé Protégé заснований на моделі подання знань OKBC (Open Knowledge Base Connectivity). Основними елементами є класи, екземпляри, слоти (представляють властивості класів та примірників) і фасети, що задають додаткову інформацію про слоти.

4.1.5. Порівняння редакторів онтологій

Нижче наводиться таблиця, що описує основні характеристики найбільш популярних редакторів онтологій (таблиця 4.1).

Таблиця 4.1 – Порівняння найбільш популярних редакторів онтології

Назва	**Короткий опис**	**Формалізми, мови, формати**	**URL**
Ontolingua	Сумісна розробка онтологій	OKBC,KIF	www.ksl.stanford.edu/software/ontolingua/
Protege	Створення, перегляд онтологій	JDBC,UML,XML,XOL,SHOE,RDF/RDFS,DAML+OIL,OWL	protege.stanford.edu
OntoSaurus	Web-браузр баз знань на мові LOOM	LOOM	www.isi.edu/isd/ontosaurus.html
OntoEdit	Розробка и підтримка онтологій	F-Logic,RDFS,OIL,OXML	www.ontoknowledge.org/tools/ontoedit.shtml
OilEd	Розробка онтологій, підтримка логічного виведення	DAML+OIL	oiled.man.ac.uk
WebOnto	Багатокористувальниць-ка розробка онтологій	OCML	kmi.open.ac.uk/projects/webonto/
WebODE	Створення онтології за допомогою методології Methontology	F-Logic,LOOM, Ontolingua	webode.dia.fi.upm.es/WebODEWeb/index.html

Як видно з таблиці, найкращу функціональність має редактор онтологій Protégé.

4.2. Розробка онтології

Відношення між артефактами утворюють архітектурну модель системи, яка поєднує в собі регламенти діяльності, структуру діяльності, напрями діяльності, технологічні стандарти, структури та рішення.

Моделі спроможностей діяльності (МСД) (карти спроможностей) надають компактні (на одній сторінці) структуровані представлення (“карти”) усіх

спроможностей організації, іноді разом з іншою допоміжною інформацією, як-от стратегія та цілі діяльності, коло зацікавлених суб'єктів (стейкхолдери) тощо. МСД, як правило, розробляються спільно архітекторами та керівниками організації, а потім «накладаються» на модель діяльності для визначення перспективних напрямів розвитку, визначення пріоритетів майбутніх витрат на технологічний розвиток та забезпечення узгодженості між новими технологіями та бажаними результатами цільової діяльності. МСД, як правило, є «точками входу» в технологічний процес ситуаційного управління для суб'єктів, що приймають рішення (керівних суб'єктів-супроводжувачів).

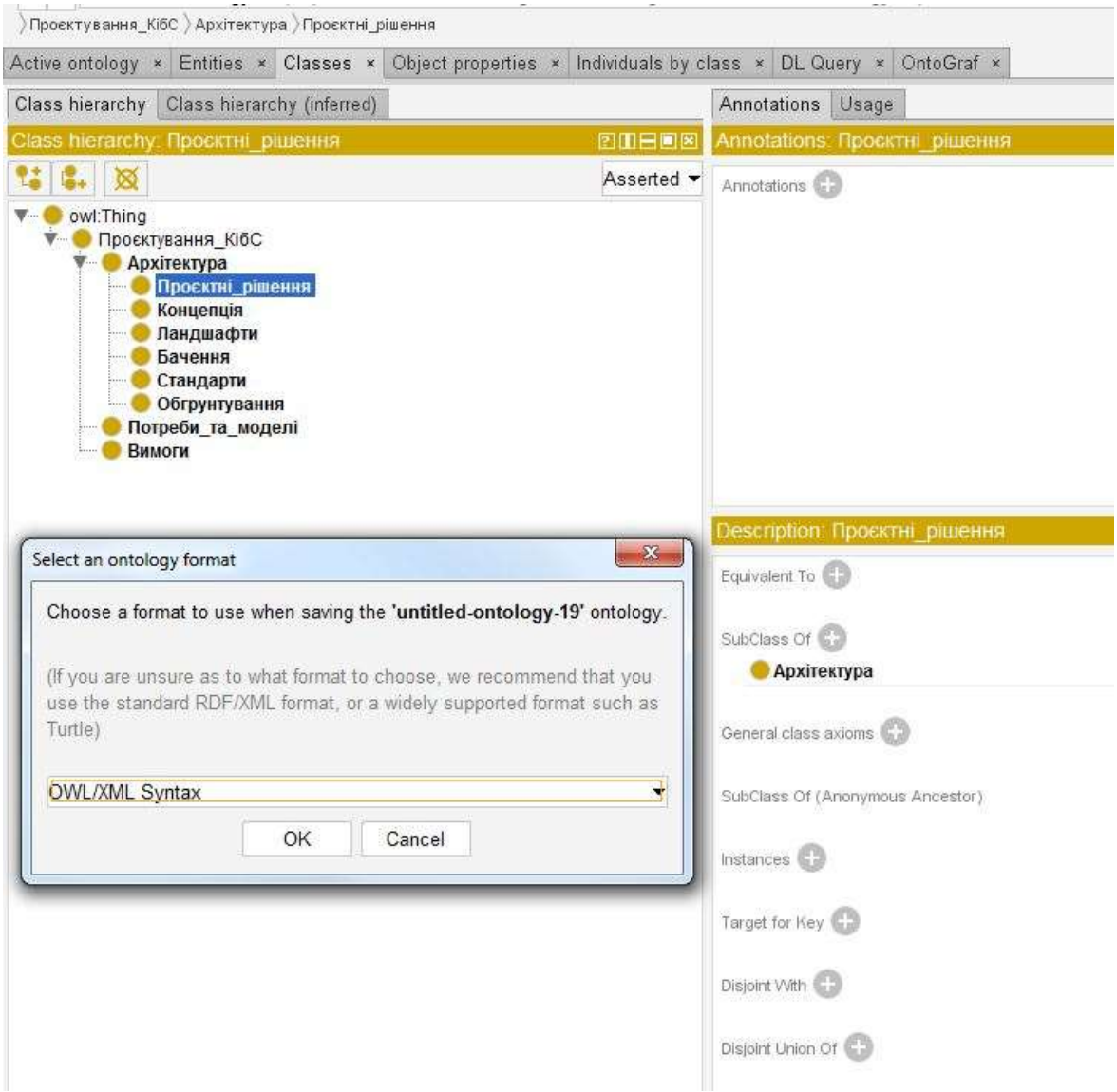

Рис.4.1. Вибір формату збереження онтології

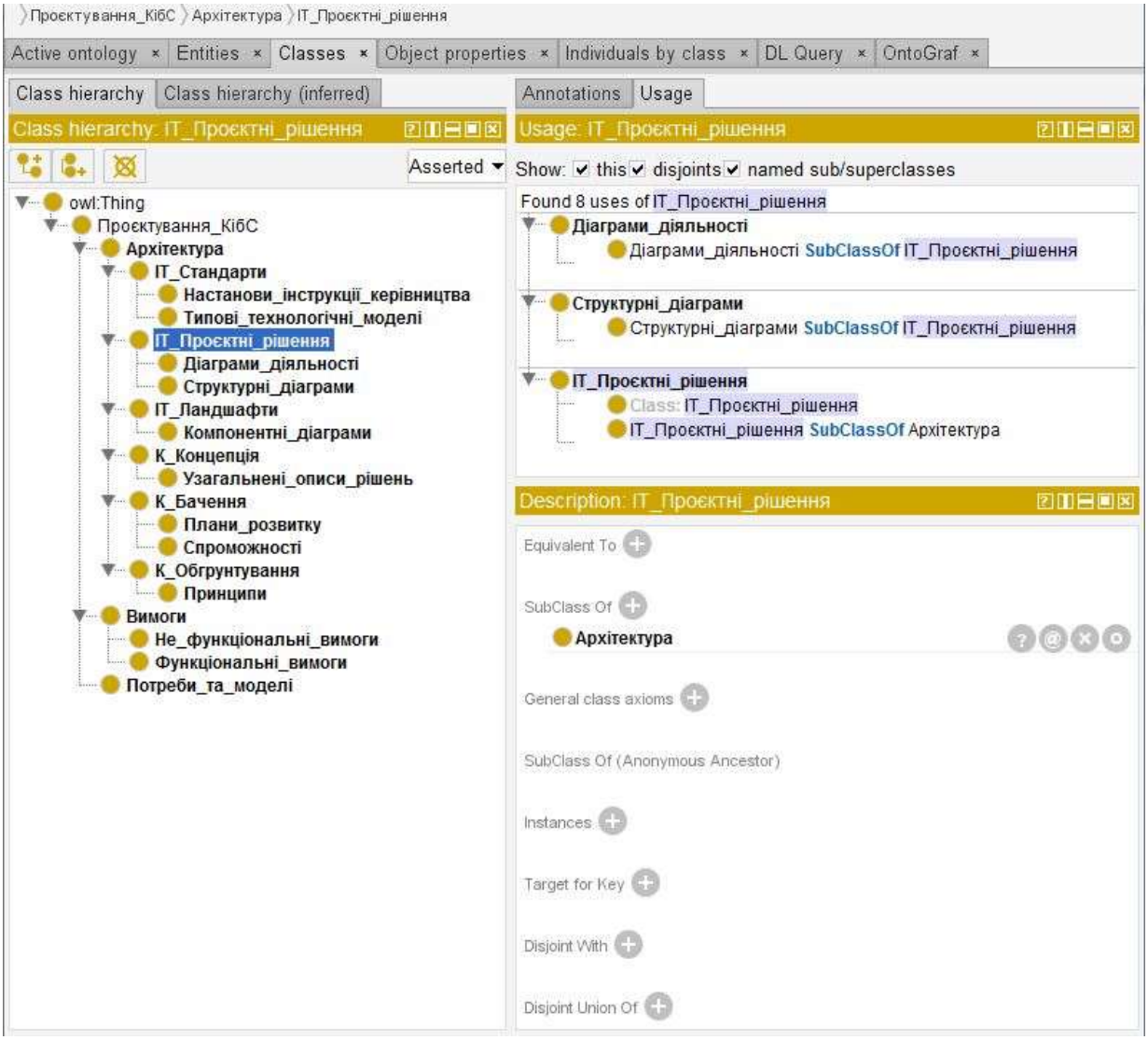

Рис.4.2. Вигляд ієрархії концептів онтології

Висновки до розділу

На основі проведеного аналізу було показано ефективність застосування онтологій для побудови корпоративних баз знань.

Побудова онтології рішень має враховувати орієнтацію на знання про всі стадії процесів їх вироблення, суб'єктний і об'єктний аспекти розгляду, співіснування різних точок зору, наявність взаємовпливів між рішеннями.

Було проведено порівняння редакторів онтологій та обґрунтовано вибір редактора Protégé для створення спеціалізованої бази знань.

Розроблено концептуальний каркас моделі знань з проєктування кібернетичних систем.

ВИСНОВКИ

В результаті виконання дипломної роботи поставлені задачі виконані в повному обсязі.

В результаті виконання роботи були отримані наступні результати:

1. Проаналізовано архітектури та етапи функціонування систем ситуаційного управління та ситуаційних центрів, на основі чого було визначено місце і роль баз знань.
2. Проаналізовано моделі та засоби представлення знань, на основі чого була вибрана онтологічна модель.
3. Проведені аналіз та порівняльна характеристика редакторів онтології та на основі обґрунтованого вибору для створення баз знань ситуаційного центру обрано редактор онтології Protégé.
4. Створено онтологічну модель знань для підтримки проєктування кібернетичних систем.

Результати роботи можуть бути використані при створенні баз знань систем ситуаційного управління та при вивченні навчальних дисциплін, пов’язаних з побудовою моделей та обробкою знань в різних предметних областях.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Субботін С. О. Подання й обробка знань у системах штучного інтелекту та підтримки прийняття рішень, Запоріжжя: Запорізький національний технічний університет, 2008, p. 341.

2. Добров Б., Иванов В., Лукашевич Н., Соловьев В. Онтологии и тезаурусы: модели, инструменты, приложения, Москва: БИНОМ, 2008.

3. Коваленко О. Онтологічна модель системи прийняття рішень методом голосування, *Системи підтримки прийняття рішень. Теорія і практика: збірник доповідей науково-практичної конференції з міжнародною участю*, Київ, Інститут проблем математичних машин і систем НАН України, 2009, pp. 51-54.

4. Коваленко О. Онтологія та модель трансформації інформації в ситуаційних агентних ситсемах, *Електронне моделювання,* т. 42, № 5, pp. 5-23, 2020.

5. Endsley M. Toward a Theory of Situation Awareness in Dynamic Systems, *The Journal of the Human Factors and Ergonomics Society,* т. 37, № 1, pp. 32-64, 1995.

6. Wooldridge M. An Introduction to Multiagent Systems. 2nd edition, Chichester: John Wiley and Sons Ltd., 2009, p. 488.

7. Kovalenko O. Systems Convergence for Situational Control and Decision Making in Distributed Environments, *16th International Conference on Advanced Trends in Radioelectronics, Telecommunications and Computer Engineering – Proceedings (TCSET-2022),* Lviv-Slavske, Ukraine, IEEE, 2022, pp. 344-347.

8. Kotusev S. Enterprise Architecture and Enterprise Architecture Artifacts: Questioning the Old Concept in Light of New Findings, *Journal of Information Technology*, т. 34, № 2, pp. 102-128, 2019.

9. Коваленко О. Системна інженерія та життєвий цикл систем, *Електронне моделювання,* т. 40, № 6, p. 61–82, 2018.

10. ISO/IEC/IEEE 15288:2015 Systems and software engineering – System life cycle processes, 2015. Available: http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=63711. [Дата звернення: 20.03.2022].

11. Kripke S. Semantical Considerations on Modal Logic, *Acta Philosophica Fennica,* № 16, pp. 83-94, 1963.

12. Коваленко О. Є. Застосування модальної логіки при прийнятті рішень на моделях знань, *Математичне та комп'ютерне моделювання. Серія: Технічні науки: зб. наук. праць,* № 6, pp. 106-112, 2012.

13. Коваленко О. Ситуаційне управління як проектна діяльність, *Системи підтримки прийняття рішень. Теорія і практика: Збірник доповідей одинадцятої дистанційної науково-практичної конференції з міжнародною участю*, Київ, 5 червня 2017.

14. Jakobson G., Buford J., Lewis L. Situation Management: Basic Concepts and Approaches, *Information Fusion and Geographic Information Systems, LNG&C, vol. XIV / V. V. Popovich, M. Schrenk, and K. V. Korolenko, Eds.*, Heidelberg, Springer, 2007, pp. 18-33.

15. Джарратано Д., Райли Г. Экспертные системы: Принципы разработки и программирование, Москва: Изд.дом «Вильямс», 2007, p. 1152.

16. Borst W. *Construction of Engineering Ontologies*. PhD thesis, Institute for Telematica and Information Technology, University of Twente, Enschede, TheNetherlands, 1997.

17. Smith B. Beyond Concepts: Ontology as Reality Representation. In A. C. Varzi and L. Vieu, editors, *Form al On tology in Information Systems – Proceedingsof the Third International Conference (FOIS 2004)*, pages 73–85. IOS Press, Amsterdam, 2004.

18. Uschold M. Ontologies and Semantics for Seamless Connectivity. *SIGMOD Record,* 33(4):58–64, 2004. Available www.researchgate.net/publication/220416182_Ontologies_and_Semantics_for_Seamless_Connectivity. [Дата звернення: 19.05.2022].

ДОДАТКИ

Додаток

Фрагмент БЗ «Проєктування КС» (KibSys) в форматі Java-коду

```
package KibSys;

import KibSys.impl.*;


import java.util.Collection;

import org.protege.owl.codegeneration.CodeGenerationFactory;
import org.protege.owl.codegeneration.WrappedIndividual;
import org.protege.owl.codegeneration.impl.FactoryHelper;
import org.protege.owl.codegeneration.impl.ProtegeJavaMapping;
import org.protege.owl.codegeneration.inference.CodeGenerationInference;
import org.protege.owl.codegeneration.inference.SimpleInference;

import org.semanticweb.owlapi.model.OWLClass;
import org.semanticweb.owlapi.model.OWLOntology;
import org.semanticweb.owlapi.model.OWLOntologyStorageException;

/**
 * A class that serves as the entry point to the generated code providing
access
 * to existing individuals in the ontology and the ability to create new
individuals in the ontology.<p>
 *
 * Generated by Protege (http://protege.stanford.edu).<br>
 * Source Class: MyFactory<br>
 * @version generated on Fri Jun 03 10:54:39 EEST 2022 by Oleksa
 */
public class MyFactory implements CodeGenerationFactory {
    private OWLOntology ontology;
    private ProtegeJavaMapping javaMapping = new ProtegeJavaMapping();
    private FactoryHelper delegate;
    private CodeGenerationInference inference;

    public MyFactory(OWLOntology ontology) {
          this(ontology, new SimpleInference(ontology));
    }

    public MyFactory(OWLOntology ontology, CodeGenerationInference inference)
{
        this.ontology = ontology;
        this.inference = inference;
        javaMapping.initialize(ontology, inference);
        delegate = new FactoryHelper(ontology, inference);
    }

    public OWLOntology getOwlOntology() {
        return ontology;
    }

    public void saveOwlOntology() throws OWLOntologyStorageException {
        ontology.getOWLOntologyManager().saveOntology(ontology);
    }

    public void flushOwlReasoner() {
```

```
        delegate.flushOwlReasoner();
    }
    public boolean canAs(WrappedIndividual resource, Class<? extends
WrappedIndividual> javaInterface) {
     return javaMapping.canAs(resource, javaInterface);
    }

    public  <X extends WrappedIndividual> X as(WrappedIndividual resource,
Class<? extends X> javaInterface) {
     return javaMapping.as(resource, javaInterface);
    }
    public Class<?> getJavaInterfaceFromOwlClass(OWLClass cls) {
        return javaMapping.getJavaInterfaceFromOwlClass(cls);
    }

    public OWLClass getOwlClassFromJavaInterface(Class<?> javaInterface) {
          return javaMapping.getOwlClassFromJavaInterface(javaInterface);
    }

    public CodeGenerationInference getInference() {
        return inference;
    }

    /* ************************************************
     * Class http://www.semanticweb.org/oleksa/ontologies/2022/4/untitled-
ontology-19#IT_Ландшафти
     */
    {

javaMapping.add("http://www.semanticweb.org/oleksa/ontologies/2022/4/untitled-
ontology-19#IT_Ландшафти", IT_Ландшафти.class, DefaultIT_Ландшафти.class);
    }
    /**
     * Creates an instance of type IT_Ландшафти.  Modifies the underlying
ontology.
     */
    public IT_Ландшафти createIT_Ландшафти(String name) {
           return delegate.createWrappedIndividual(name,
Vocabulary.CLASS_IT_ЛАНДШАФТИ, DefaultIT_Ландшафти.class);
    }
    /**
     * Gets an instance of type IT_Ландшафти with the given name.  Does not
modify the underlying ontology.
     * @param name the name of the OWL named individual to be retrieved.
     */
    public IT_Ландшафти getIT_Ландшафти(String name) {
           return delegate.getWrappedIndividual(name,
Vocabulary.CLASS_IT_ЛАНДШАФТИ, DefaultIT_Ландшафти.class);
    }
    /**
     * Gets all instances of Функціональні_вимоги from the ontology.
     */
    public Collection<? extends Функціональні_вимоги>
getAllФункціональні_вимогиInstances() {
           return
delegate.getWrappedIndividuals(Vocabulary.CLASS_ФУНКЦІОНАЛЬНІ_ВИМОГИ,
DefaultФункціональні_вимоги.class);
    }
}
```