техническая информация. – 2005. – № 11. – С. 21–33. 4. Клифтон Б. Google Analytics: профессиональный анализ посещаемости веб-сайтов / Б. Клифтон. – М: Вильямс, 2009. – 400 с. 5. Корнеев В. Базы данных. Интеллектуальная обработка информации / В. Корнеев, А. Гареев, С. Васютин, В. Райх. – М: Нолидж, 2000. – 352 с. 6. Ландэ Д. Основы моделирования и оценки електронных информационных потоков / Д. Ландэ, В. Фурашев, С. Брайчевский, О. Григорьев. – К: Інжиніринг, 2006. – 348 с. 7. Ландэ Д. Основы интеграции информационных потоков: монография / Д. Ландэ. – К: Інжиніринг, 2006. – 240 с. 8. Советов Б. Моделирование систем / Б. Советов, С. Яковлев. – М.: ВШ, 1998. 9. Федорчук А. Контент-мониторинг информационных потоков / А. Федорчук. – К., 2005. – № 3.

УДК 004.89

Є.В. Буров
Національний університет “Львівська політехніка”,
кафедра інформаційних систем та мереж

ФОРМАЛЬНА МОДЕЛЬ ПОДАННЯ ЗНАНЬ У СИСТЕМІ ОНТОЛОГІЧНОГО МОДЕЛЮВАННЯ ЗАДАЧ

Розглянуто математичну формалізацію системи онтологічного моделювання процесу розв’язання задач. Для її побудови використано апарат алгебраїчної теорії систем. Розроблено формальне подання онтологічних моделей та системи їх опрацювання.

Ключові слова: база знань, математична модель, онтологія, онтологічна модель.

In this paper we propose a formalization of ontology-based task execution modelling system. It is built using approach of algebraic systems theory. We show that proposed algebraic system is based on multiple domains, which can be used for ontological models representation and knowledge elucidation, storage and processing

Key words: knowledge base, mathematical model, ontology, ontological model.

Вступ та постановка проблеми

Побудова і впровадження інтелектуальних систем, що ґрунтуються на формалізації та повторному використанні знань, є перспективним напрямом практичного застосування методів штучного інтелекту у програмних системах. В основу таких систем покладено формалізоване подання знань про предметну область, наприклад, у формі онтології. Визначення онтології Грубером [1] як специфікації певної концептуалізації залишає відкритим питання вибору формального апарату та мовних засобів для побудови такої специфікації. Не до кінця вирішеними залишаються завдання аналізу та виявлення суті онтологічного опису предметної області, визначення властивих йому обмежень та переваг. Це вимагає побудови та дослідження формальних моделей для різноманітних аспектів онтологічного моделювання. Отже, актуальним є дослідження методу онтологічного моделювання з використанням математичних моделей онтологій та систем онтологічного моделювання.

Аналіз останніх досліджень і публікацій

Застосування формальних методів для дослідження систем, що використовують дані та знання, має довгу історію. Найчастіше дослідники використовують формальні методи, що ґрунтуються на алгебраїчному підході, теорії множин та логіці предикатів першого порядку [2–4]. Так, Кодд [2] на основі логіки предикатів першого порядку та алгебри множин розробив реляційну алгебру, яку було використано для побудови теоретичних основ реляційних баз даних, зокрема мови запитів до баз даних SQL.

У роботі [3] розвинено та узагальнено алгебраїчний підхід до опису реляційних баз даних, зокрема алгебраїчну модель бази даних, розглянуто питання еквівалентності та симетрії відношень. з використанням багатосортних алгебр. Автор подає модель бази даних у вигляді алгебраїчної структури з використанням багатосортних алгебр. Він проводить алгебраїчне моделювання операцій об'єднання та декомпозиції баз даних.

У роботі [4] використано алгебраїчний підхід для моделювання понять та баз понять. Автор використовує абстрактні типи даних для подання понять та визначає алгебраїчні операції над типами даних.

У роботі [5] запропоновано алгебраїчний підхід до побудови моделей систем та визначено перетворення моделей як алгебраїчні операції. Автори дослідили можливість автоматизації проведення складних логічних виводів. У результаті створено виконувальну метамову як засіб для визначення, побудови, оцінювання та перетворення моделей складних систем.

У [6] запропоновано підхід до побудови програмних систем з використанням інтерпретованих онтологічних моделей, архітектуру та порядок роботи системи моделювання. При цьому недостатньо висвітленими залишаються аспекти математичної формалізації запропонованого підходу, зокрема подання знань з використанням онтологій задач та моделей, залежностей між ними та питання практичної реалізації системи моделювання.

Формулювання цілі статті

Сьогодні дослідження в галузі онтологічного моделювання зосередилися на декларативних онтологіях – онтологіях предметних областей, загальних онтологіях [7]. Онтології задач та онтологічні моделі, які будуються на їх основі, вивчено недостатньо.

Історично онтології задач було запропоновано в результаті розвитку наукового напряму аналізу задач (task analysis). Методи аналізу задач використовуються для визначення та формалізації усіх факторів, що впливають та використовуються в процесі розв'язання задачі експертом. Такі методи широко застосовуються для проектування інтерфейсів комп'ютерних програм, в експертних системах, системах підтримки прийняття рішень [8].

Головним завданням цього напряму є аналіз та специфікація складових задачі, визначення її структури та обмежень. Це дозволяє експерту краще зрозуміти задачу, виявити свої помилки та упущення, змоделювати процес розв'язання задачі та оцінити результати, передати свої знання іншим експертам та спроектувати комп'ютерний інтерфейс для розв'язання задачі.

Значних змін напрям аналізу задач зазнав з появою онтологій. Було запропоновано використовувати онтології задач (task ontologies) для формалізації концептів та відношень задачі [9]. На відміну від інших типів онтологій, таких як загальні або онтології предметних областей (general, domain ontologies), онтології задач:

- будуються окремо для класів подібних задач;
- важливим є концепт мети, пов'язаний із задачею та його формалізація;
- вводиться концепт дії [10] та забезпечується виконання (або моделювання) дії;
- реалізовано виконання моделі, побудованої на основі онтології задачі.

Дослідження онтологій задач тісно пов'язані з концептуальним моделюванням, адже в процесі побудови онтології задачі фактично створюється її формалізована концептуальна модель [11]. Важливим аспектом як концептуального, так і онтологічного моделювання задачі є взаємодія з експертом предметної галузі, який створює та валідує онтологію.

У процесі досліджень онтологій задач були реалізовані середовища моделювання, які дозволяють створювати та виконувати онтологічні моделі для окремих класів задач. Найбільш розвиненим з таких середовищ є CLEPE (Conceptual level programming environment) [7]

Водночас наявні роботи розглядають онтології задач відокремлено, не вирішено також задачу математичного моделювання онтологій задач та формального подання онтологічних моделей.

Метою цієї роботи є побудова формальної моделі подання знань в інтелектуальній системі, що базується на онтологіях задач та онтологічних моделях задач

Виклад основного матеріалу. Математична модель подання знань

Використання онтологічних моделей для побудови програмних систем вимагає розроблення математичного обгрунтування у вигляді відповідної формальної моделі, яку надалі використовуватимуть для формального подання та валідації методів та системи моделювання. Для побудови формальної моделі використано підхід алгебри систем [5], 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

$$E_i = (Name, \Sigma, Ex),$$

де Name – назва типу; Σ – сигнатура багатосортної алгебри; Ex – визначальні співвідношення типу. При цьому сигнатура містить тільки множину елементів і 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. Змінні, що набувають значення екземплярів конкретного типу, позначимо X_{name}, або просто Name. Типи E_i утворюють множину алгебраїчних доменів E, які відповідають концептам (сутностям) предметної області.

$$E = \{E_1, E_2, ..., E_n, \}.$$

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. Згідно з [5] для цього домену визначені операції об'єднання, підстановки, видалення, інтерпретації (merge, substitute, delete, interp).

На практиці корисними є функції, які визначені на змінних різних доменів, наприклад, функція вибору значення атрибуту:

$$F_{selval}(At, key) \to value.$$

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, яка 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 і 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.

Домен відношень Rl. Множиною-носієм цього домену є 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 множину 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 Е. В операції підстановки замість однієї сутності домену Т 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:

$$On = < T, Rl, Cs^* >$$

Кожне відношення $R \in Rl$ задане на множині ролей $\{P_1, P_2, ..., P_n\}$:

$$R(P_1, P_2, ..., P_n)$$

У загальному випадку для кожної ролі P_k визначено функцію ініціалізації $F_{in}^{\ k}$, яка визначає підмножину сутностей, елементам якої дозволено заповнювати роль:

$$F_{in}^{\ k} : P_k \to T_{in}^{\ k} \subseteq T$$

У найпростішому випадку, коли $\forall k : |T_{in}^{\ k}| = 1$, для кожної ролі існує лише один тип сутностей, яким вона може бути ініціалізована. У цьому випадку можна у позначенні відношення замінити ролі відповідними сутностями онтології: $R(E_1, E_2, ..., E_n)$

Сутності онтології утворюють ієрархічну структуру (таксономію) з використанням відношення успадкування (isa-відношення) R_{isa}

Бінарне відношення R_{isa} задане на впорядкованій парі ролей *Нащадок–Предок*:

$$R_{isa}(P_{ch}, P_{pr})$$

Відношення успадкування є транзитивним, так що, якщо $R_{isa}(E_1, E_2)$ та $R_{isa}(E_2, E_3)$, то справедливе $R_{isa}(E_1, E_3)$.

Визначимо функцію F_{pr}, яка для кожної сутності E_j визначає впорядкований список її предків $(E_1, E_2, ..., E_k)$ так, що виконуються $R_{isa}(E_i, E_{i+1})$ та $R_{isa}(E_j, E_1)$. Визначимо також функцію $F_{pr}^{\ 1}(E_j)$, яка повертає безпосереднього предка сутності E_j або порожню множину $\varnothing$.

Важливим різновидом відношення є відношення *Ціле–Частина* (has-parts відношення): $R_{has}(P_{wh}, P_{pt})$, де P_{wh} – роль цілого, а P_{pt} – роль частин цього цілого. Підтип такого відношення R'_{has} визначає конкретні сутності для цілого та набори дозволених сутностей для частин:

$$R'_{has}(E'_{wh}, \{E_{pt}^1, E_{pt}^2, ..., E_{pt}^n\}').$$

Для простоти при поданні такого відношення використовують нотацію:

$$E_{wh} = (E_{pt}^1, E_{pt}^2, ..., E_{pt}^n).$$

З іншого боку, відношення та моделі на рівні онтології є окремими сутностями. Цим сутностям в алгебраїчній системі типів T відповідають типи даних, екземпляри яких зберігають дані про ці відношення та моделі (метадані).

Важливою складовою визначення відношень в онтології є обмеження цілісності, що накладаються на сутності, поєднані відношенням. У визначеній системі типів таким обмеженням відповідає набір булевих виразів $Cs_{Rl,i}$ для кожного типу відношення Rl_i, які повинні бути істинними:

$$\forall cs^{i}_{Rl,i} \in Cs_{Rl,i} : ev(cs^{i}_{Rl}) = true,$$

де ev() – функція оцінки значення виразу.

У деяких випадках обмеження накладають і на значення атрибутів сутностей. Будемо розглядати такі обмеження як обмеження цілісності для унарних відношень.

Визначимо функцію TypeParents(), яка для кожного типу T_i повертатиме впорядковану множину його супертипів і є транзитивним замиканням відношення успадкування.

$$TypeParents(T_i) = (T^1, T^2 ..., T^n),$$

де T^{i+1} є супертипом відносно T^i.

Визначимо функцію TypeName(), яка для кожного типу даних T повертає унікальний ідентифікатор (опис, назву) цього типу. Наприклад, для типу даних T_{MD}, що відповідає моделі:

$$TypeName(T_{MD}) = "Model"$$

Для екземплярів *t* довільного типу *T* визначимо функції, що повертають тип:

$$Type(t) = T$$

Мультимножину усіх екземплярів типу *T* позначимо $Population(T)$.

$$Population(T) = \{t \mid Type(t) = T\}$$

Позначимо мультимножину екземплярів певного типу T_i як $\hat{t}_i$:

$$\hat{t}_i \mid \forall t_i \in \hat{t}_i : Type(t_i) = T_i$$
$$\hat{t}_i \subseteq Population(T_i)$$

Абстрактний тип даних, що відповідає мультимножині еземплярів $\hat{t}_i$, позначимо $\hat{T}_i$.

У загальному випадку довільний об'єкт o можна ідентифікувати як об'єкт багатьох типів.

Визначимо функцію TypeId(), яка повертатиме множину типів, якими можна ідентифікувати об'єкт:

$$TypeId(o) = \{T_o^1, T_o^2, ..., T_o^m\}.$$

У тих випадках, коли необхідно наголосити на семантичній інтерпретації певного типу даних, позначаємо скорочену назву типу у вигляді індекса. Наприклад, позначимо тип даних моделі як T_{MD}, конкретний елемент цього типу – t_{MD} або мультимножину елементів моделей – $\hat{t}_{MD}$.

Визначимо онтологію задачі On_{TS}, яка є частиною загальної онтології On і містить об'єкти, необхідні для розв'язання конкретної задачі

$$On_{TS} \subseteq On.$$

Онтологічна модель Md визначається кортежем з трьох елементів:

$$Md = < On_{TS}, Ac^*_{TS}, Cs^*_{TS} >,$$

де On_{TS} – онтологія задачі, Ac^*_{TS} – множина дій, Cs^*_{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 визначено параметри, які отримують із значень атрибутів елементів, що входять у On_{TS} (або ж вони задані як константи). Функцію ініціалізації параметрів InstPar подамо як набір відображень між атрибутом дії та значеннями атрибутів сутностей та відношень, що входять в онтогічну модель:

$$InstPar : (Ac_{l,i}, pkey_{l,i}) \rightarrow SelValue(At_{l,j}, key_{l,k});$$

$$Ac_{l,i} \in Md_l, At_{l,j} \in On^l_{TS} \in Md_l.$$

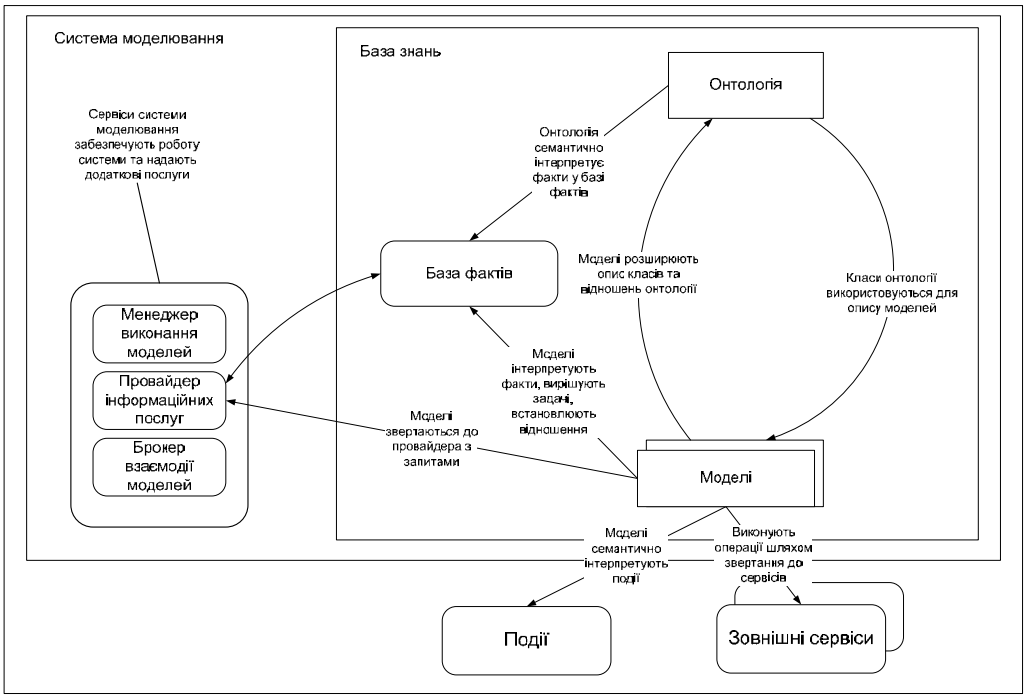
У таблиці подано основні алгебраїчні домени, використані у математичній моделі.

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	$\{(On_{TS}, Ac_{TS}^*, Cs_{TS}^*)\}$	

Структура та порядок роботи системи моделювання

Система моделювання, що реалізує запропонований підхід, складається з таких компонент (рисунок). База фактів містить факти про об'єкти та події зовнішнього світу, необхідні для розв'язання задач системою. Всі факти семантично інтерпретовані, тобто подані як об'єкти певних класів, визначених онтологією. Інформація в базі фактів впорядкована за часом та змінами, що дає змогу відслідкувати стан бази на довільний момент часу або на довільну зміну. База фактів, онтологія та моделі разом утворюють базу знань системи.



Структура системи моделювання

Онтологія містить модель предметної області, подану як таксономія класів. Це створює можливість однозначного трактування усіх об'єктів з бази фактів, визначення єдиної структури слотів та типів властивостей для них. Крім того, онтологія містить відношення, правила та обмеження, які мають загальний, системний характер.

Посилання на моделі використовуються в онтологіях для зберігання складних правил та обмежень, зокрема динамічних обмежень, значення яких залежить від стану бази знань або зовнішнього світу.

Система реагує на визначене коло подій зовнішнього світу та опрацьовує іх, створюючи нові або модифікуючи існучі факти. Для опрацювання цих подій використовують відповідні моделі.

Важливими компонентами системи є сервіси системи моделювання, які забезпечують виконання моделей, їх взаємодію, пошук потрібної інформації у зовнішніх джерелах.

Так, Менеджер виконання моделей запускає або зупиняє моделі, відслідковує використання ресурсів моделей. Брокер взаємодії моделей шукає релевантні моделі для розв'язання задач, ініціалізує та запускає обрані моделі. Провайдер інформаційних послуг за запитом моделі системи звертається до зовнішніх джерел, шукає необхідні дані та їх семантично інтерпретує.

Моделі виконують операції відповідно до відображеної в них логіки, звертаючись із запитами до зовнішніх відносно системи моделювання сервісів. Такими сервісами є сервіси операційної системи, сервіси інформаційної системи підприємства, побудованої за вимогами SOA [12], або довільні веб-сервіси.

Опрацювання знань у системі моделювання

Онтологічні моделі утворюють мережу, що має тип T_{NMD}, який визначається як множина, до якої входять мультимножини типу моделей $\hat{T}_{MD}$ та їх відношень $\hat{T}_{RMD}$.

$$T_{NMD} = \{\hat{T}_{MD}, \hat{T}_{RMD}\}$$

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

Кожна модель може бути в одному з двох станів – активному або пасивному. Відповідно, множину екземплярів моделей поділимо на підмножини активних та пасивних моделей, які не перетинаються:

$$Population(T_{MD}) = \hat{t}^{ac}_{MD} \cup \hat{t}^{ps}_{MD};$$
$$\hat{t}^{ac}_{MD} \cap \hat{t}^{ps}_{MD} = \varnothing.$$

де $\hat{t}^{ac}_{MD}, \hat{t}^{ps}_{MD}$ позначено мультимножини екземплярів активних та пасивних моделей.

Активна модель – це модель, ініціалізована інформацією з певного контексту. Моделі переходять в активний стан на вимогу інших моделей або при настанні певних подій. Активні моделі використовуються для розв'язання поточних задач системи, інтерпретації знань у системі. Якщо потреба в моделі відпала (отримано результат, досягнуто мети), то модель переходить з активного у пасивний стан.

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

Розглянемо процес та структуру взаємодії та активації моделей детальніше. Ініціатором встановлення зв'язку між моделями є модель-активатор. Потреба у встановленні зв'язку виникає не завжди, а лише тоді, коли для розв'язання основної задачі потрібно розв'язати допоміжні задачі, подані в інших моделях. Наприклад, якщо вхідні дані, отримані активатором з контексту, є неповними і необхідно активізувати інші моделі для довизначення даних. Таке довизначення може полягати в пошуку потрібної інформації в базі даних, інтернеті, звертанні до консультанта тощо.

Кожному типу T^i_{RMD} відповідає клас задач, які необхідно розв'язати T^j_{PR} у результаті взаємодії. Своєю чергою, класу задач T^j_{PR} відповідає множина моделей $\hat{T}^j_{MD}$, які можуть бути застосовані для розв'язання задач цього класу.

Під час взаємодії моделей послідовно розв'язують задачі визначення релевантності, оптимального вибору серед релевантних моделей та ініціалізації обраної моделі.

Функція релевантності є відображенням поточного контексту t_{CON} та множини альтернатив $\hat{t}_{MD}$ в множину {true, false}

$$F_{rel} : (t_{CON}, \hat{t}_{MD}) \to (true,\ false).$$

Визначення релевантності моделей дозволяє відібрати для процедури вибору лише релевантні моделі:

$$\hat{t}_{MD}^{re} \subseteq \hat{t}_{MD},$$

тобто такі моделі t_{MD}^{re}, для яких

$$F_{rel}(t_{CON}, t_{MD}^{re}) = true.$$

За відсутності релевантних моделей система моделювання повертає активатору повідомлення про неможливість розв'язання задачі.

Задача оптимального вибору визначає у множині релевантних моделей одну t_{MD}^{op}, застосування якої максимізує функцію вибору F_{ch} з врахуванням критеріїв вибору $\hat{t}_{CR}$ та контексту t_{CON}.

$$F_{ch}(t_{MD}^{op}, \hat{t}_{CR}, t_{CON}) \to \max$$

Функція ініціалізації F_{in} відображає поточний контекст t_{CON} у множину значень атрибутів обраної моделі – t_{VSL}.

$$F_{in} : t_{CON} \to t_{VSL}.$$

Отже, визначимо T_{RMD} як множину

$$T_{RMD} = \{T_{PR}, \hat{T}_{MD}, F_{rel,} F_{ch}, F_{in}\}.$$

Тип моделі T_{MD} складається з типів схеми T_{SCM} та реалізації T_{IMD}:

$$T_{MD} = \{T_{SCM}, T_{IMD}\}.$$

Схема моделі описує її структуру, складові, визначає правила та обмеження на використання моделі, а також перелік можливих операцій та запитів. Схема є компонентом моделі, який видимий для зовнішнього світу. Вона використовується для взаємодії з моделлю.

Схема моделі складається зі слотів $\hat{T}_{SLM}$, відношень між ними $\hat{T}_{RSM}$, правил $\hat{T}_{RUM}$, обмежень $\hat{T}_{CSM}$ та операцій $\hat{T}_{OPM}$:

$$T_{SCM} = \{\hat{T}_{RO}, \hat{T}_{RRO}, \hat{T}_{RUM}, \hat{T}_{CSM}, \hat{T}_{OPM}\}.$$

Слот моделі є атрибутом-роллю. Для кожного слоту визначено функцію F_{RG}, яка специфікує множину класів $\hat{T}_{CL}^{RG}$, об'єктами яких дозволено ініціалізувати слот:

$$F_{RG} : T_{SLM} \to \hat{T}_{CL}^{RG}.$$

Крім того, для слоту моделі визначено правила та обмеження, що діють на рівні слоту – $\hat{T}_{RUSM}$, $\hat{T}_{CSSM}$, операції над значеннями слоту $\hat{T}_{OPSM}$:

$$T_{SLM} = \{\hat{T}_{CL}^{RG}, \hat{T}_{RUSM}, \hat{T}_{CSSM}, \hat{T}_{OPSM}\}.$$

Відношення слотів T_{RESM} специфікується множиною слотів, яку він сполучає $\hat{T}_{SLM}^{resm}$, множиною класів онтології, які використовують для семантичної інтерпретації відношення – $\hat{T}_{CL}^{resm}$, множиною моделей, яку використовують для розуміння та проведення операцій з відношенням – $\hat{T}_{MD}^{resm}$.

$$T_{RESM} = \{\hat{T}_{SLM}^{resm}, \hat{T}_{CL}^{resm}, \hat{T}_{MD}^{resm}\},$$

при цьому для кожного екземпляру відношення слоти, що ним сполучені, належать слотам моделі:

$$\hat{t}_{SLM}^{resm} \subseteq \hat{t}_{SLM}.$$

Відношення моделей відповідає одному з типів відношень, визначених в онтології On:

$$TypeEn(T_{RESM}) = E_{RESM} \in \bar{E}.$$

Модель, що описує відношення, є елементом загальної множини моделей:

$$t_{MD}^{resm} \in Population(T_{MD}).$$

Нехай t'_{BFC} – це певна ситуація, стан, знімок бази фактів. Визначимо тип даних мети T_{GL}, кожен екземпляр якого є специфікацією певної множини станів бази фактів, кожен з яких відповідає досягнутій меті:

$$t_{GL} = \hat{t}_{BFC}^{GL}.$$

Для визначення мети корисно задати функцію мети, яка дозволяє визначити, чи у певному стані t'_{BFC} мети Gl досягнуто:

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

Одним з можливих способів задання функції мети є її задання у вигляді впорядкованого списку тверджень-вимог $\hat{t}_{ASR}$ щодо об'єктів інформаційної бази, які можна перевірити.

$$F_{gl}(t'_{BFC}) = \hat{t}_{ASR}(t'_{BFC}),$$

де $t_{ASR}(t'_{BFC})$ – це вимога-твердження (assertion) щодо значень властивостей об'єктів та їх зв'язків у ситуації t'_{BFC}.

Кожне твердження $t_{ASR}(t'_{BFC})$ є функцією, заданою на множині {true,false}:

$$Range(t_{ASR}(t'_{BFC})) = \{true, false\},$$

де функція Range(f) задає область значення функції f.

Тобто

$$F_{gl}(t'_{BFC}) = true \Leftrightarrow \forall t_{ASR}(t'_{BFC}) \in \hat{t}_{ASR}(t'_{BFC}) : t_{ASR}(t'_{BFC}) = true.$$

Виконувані онтологічні моделі призначені для розв'язання конкретних задач, специфікованих у вигляді мети. Для зручності пошуку потрібних моделей доцільно організувати інформацію про моделі у вигляді онтології OnGl⊂On зі своїми сутностями та відношеннями. У цій онтології визначимо категорії моделей за класами задач, які вони вирішують. Так, наприклад, окремо визначимо моделі класифікації, алгоритмічні моделі, моделі об'єктів та сервісів, моделі керування доступом, ситуаційні моделі. Інформацію про об'єкти, які описують окремі екземпляри моделей, використовує сервіс Менеджер виконання моделей. Брокер взаємодії моделей використовує онтологію мети для пошуку моделі потрібної для розв'язання поставленої задачі.

Висновок

Розроблення та дослідження математичних моделей інтелектуальних систем, що використовують онтології та моделі задач, дадуть змогу зрозуміти теоретичні аспекти таких систем, визначити їхні обмеження, вирішити завдання валідації та верифікації систем онтологічних моделей.

1. Gruber, T.R. A translation approach to portable ontology specifications / Gruber, T.R. // Knowledge Acquisition. – 1993. – vol 5. – P.199–220. 2. Codd E.F. Relational Completeness of Data Base

Sublanguages/Codd E/F//Database systems. – 1972, vol 54, #2.-P.65-98. 3. Плоткин Б.И. Универсальная алгебра, валгебраическая логика и базы данных / Б.И. Плоткин. – М.: Наука, 1991. – 446 с. 4. Бениаминов Е.М. Алгебраические методы в теории без данных и представлений знаний / Бениаминов Е.М. – М.: Научный мир, 2003. – С.184. 5. Koo B Algebra of systems: a metalanguage for model synthesis and evaluation / Koo B, Simmons W. // IEEE Transactions on systems, man and cybernetics, vol 39, N 3, may 2009 6. Буров Є.В. Концептуальне моделювання інтелектуальних програмних систем: монографія / Є.В. Буров. – Львів: Вид-во Львівської політехніки, 2012. – 432 с. 7. Mitsuri Ikeda. Task ontology: Ontology for building conceptual problem solving models/ Mitsuri Ikeda, Kazuhisa Seta, Osamu Kakusho, Riichiro Mizoguchi. // Workshop note of application of ontologies and problem-solving meyhods, ECAI98, 1998 8. Johnson, Peter. Task-Related Knowledge Structures: Analysis, Modelling and Application / Johnson Peter, Johnson Hilary, Waddington Ray, Shouls, Alan// Knowledge Creation Diffusion Utilization, 1988. – P. 35–62 9. Van Welie Martin. An Ontology for Task World Models / Van Welie, Martijn, Van Der Veer, Gerrit C Eliëns // Methods, v. 98, 1998. – P. 57–70. 10. Raubal Martin. Ontology-Based Task Simulation/ Raubal M., Kuhn W. // Spatial Cognition and Computation, v.4, 2004. – P. 15–37 11. Seta Kazuhisa. Building ontologies for conceptual model management / Seta Kazuhisa, Koyama Kazuya, Hayashi Yusuke, Ikeda Mitsuru // WSEAS Transactions on Information Science and Applications, v.3, 2006 . – P. 546–553. 12. Erl, T. SOA Principles of Service Design. / Erl, T. – Prentice Hall, 2007.

УДК 004.652

О.М. Верес[1], Ю.О. Верес[2]
Національний університет "Львівська політехніка",
[1]кафедра інформаційних систем та мереж;
[2]кафедра соціальних комунікацій та інформаційної діяльності

ЗАСТОСУВАННЯ ОБ'ЄКТНО-ОРІЄНТОВАНОГО ПІДХОДУ ДО ПОБУДОВИ МОДЕЛІ СППР

Описано об'єктно-орієнтований підхід моделювання СППР. Запропоновано та описано трирівневу архітектуру концептуальної моделі СППР. Проаналізовано можливість багаторазового використання компонентів різних типів структур. Описано принципи побудови користувацької СППР із використанням багаторазових компонент.

Ключові слова: об'єктно-орієнтований підхід, архітектура, структура, модель, прийняття рішення, система підтримки прийняття рішень.

This article describes an object-oriented approach of DSS modeling. Three-level architecture is suggested and it describes a conceptual model of DSS. The analysis of possible reusable components for various types of structures is analyzed. We describe the principles of building a custom DSS use of multiple components.

Key words: object-oriented paradigm, architecture, frameworks, model, decision making, Decision Support System.

Вступ. Загальна постановка проблеми

Однією з постійних проблем менеджерів є розуміння можливостей та небезпек ринку і своєчасне прийняття рішень, які покращать використання корпоративних ресурсів. Системи підтримки прийняття рішень (СППР) – це комп'ютерні системи, призначені для підвищення ефективності прийняття рішень, допомоги особам, які приймають рішення щодо використання моделі і даних для вирішення частково структурованих та неструктурованих проблем прийняття рішення [1]. СППР – це сукупність інтелектуальних інформаційних застосувань та