УДК 681.3.06

В.М. Зінькович

ОНТОЛОГІЧНЕ МОДЕЛЮВАННЯ ПРЕДМЕТНОЇ ОБЛАСТІ З ПРОБЛЕМАТИКОЮ e-SCIENCE

Розглядається проблема застосування результатів онтологічного моделювання предметних областей з проблематикою e-Science для побудови е-інфраструктур інформаційно-технологічної підтримки ведення науково-дослідних робіт. Розглянуті найбільш визначальні аспекти використання інструментарію Protégé для рішення цієї проблеми. Наведено приклад застосування Protégé щодо рішення зазначеної проблематики.

Вступ

Проблематика **e-Science** визначається перетином сфер інформаційних технологій та науки. Сьогодні активно розроблюється, головним чином, один її аспект – виконання наукових та проектних робіт у рамках національного та інтернаціонального співробітництва шляхом обчислювальних GRID-технологій.

У даній роботі серед інших проблем **e-Science**, релевантних наведеній, обрано проблему інформаційно-технологічного (ІТ) забезпечення науково-дослідних робіт (НДР). Розробка і рішення проблеми базується на використанні узагальненої онтологічної моделі предметної області (ПрО), у ролі к*онцептів* якої подані проблема або задача користувача, метод її рішення, інструментальний засіб та технологія інструментальної підтримки методу, подані в моделі через типізовані специфікації. Усім перелікованим концептам поставлені у відповідність такі *артефакти* e-Science, як інформаційні, технологічні та інструментальні ресурси Інтернет.

Самі концепти ПрО відображені в цієї моделі через *класи*, а відношення між ними визначаються через їх *атрибути* або *слоти*. До основи методу створення моделі покладено побудову засобів взаємодії та інтеграції зазначених класів. Модель створюється як *дерево графів* та подається в одному з форматів мов – HTML, RDF, OWL та ін.

Онтологічна модель, створена таким чином, подає собою Базу Знань (БЗ), яка править за узагальнений *понятійний каркас* взаємодії деякої множини артефакттів ПрО, і в подальшому використовується при побудові моделей *проблемних областей*, які обумовлені тематичними категоріями проблематики користувачів моделі ПрО. Передумовою побудові та використанню проблемних областей є створення *екземплярів* певних класів моделі ПрО, які утворюють склад бази даних (БД) моделі. Екземпляри класів ідентифікують конкретні артефакти e-Science, інтерпретуючи їх як *компоненти повторного використання* (КПВ). Самі КПВ мають бути розташовані у будь-якому репозиторії або у мережі Інтернет.

Головним призначенням БД моделі є отримання *е-інфраструктури* середовища ІТ-підтримки НДР користувача через набір запитів до цієї БД, які відбивають його вимоги щодо компонентів інфраструктури – тематичної категорії, проблематики, методичної та технологічно-інструментальної підтримки у вигляді впорядкованого набору посилань на артефакти, тобто конкретні КПВ.

Слід відмітити, що в статті розглядаються *інформаційні* онтологічні моделі ПрО, які визначають лише інформаційні та логічні зв’язки між різними типами КПВ,

необхідні для складання КПВ у конфігурації без їх подальшого застосування.

Підхід апробовано на моделі ПрО ІТ-підтримки наукових досліджень. Рішення на такій моделі подають собою інформаційні технології, адаптовані до конкретної проблематики користувачів, мають *віртуальний* характер і формуються виключно за привхідними критеріями користувачів щодо тематичних категорій досліджень, постановок задач дослідження, вибору методів їх рішення та інструментально-технологічною підтримки. Побудову та апробацію онтологічної моделі ПрО здійснено за допомогою онтологічного редактора PROTÉGÉ версії 3.46 [1, 2] .

Основні концепції моделювання ПрО з проблематикою e-Science

Взагалі концепції підходу до моделювання ПрО визначаються сутністю ПрО та метою використання моделі. У нашому випадку об'єктом *відпрацьовування методології* моделювання обрана предметна область *методологічної, технологічної і програмно-інструментальної підтримки* НДР певних тематичних категорій. Базові концепти ПрО – це узагальнені типи артефактів, що визначені наступним чином:

- тематична категорія НДР;
- задача або проблема певної тематичної категорії;
- методологія вирішення задачі (тобто сукупність методів);
- програмно-інструментальний засіб підтримки методу;
- технологія інструментальної підтримки методів.

Усі ці концепти ПрО подані в моделі у вигляді класів, об'єднаних у дерево графа, коренем якого є клас, що відповідає тематичній категорії НДР, а ребра графа відбивають семантичні зв'язки між класами. Самі класи подані в онтологічній моделі Protégé через специфікації або *дескриптори* класів.

Екземплярами класів мають бути артефакти e-Science, в ролі яких виступають ресурси повторного використання, а саме інформаційні та програмно-інструментальні КПВ, конкретизовані згідно з визначенням концептів, що асоційовані з класами. Так, для обраної ПрО в ролі інформаційних КПВ виступають описи проблем або задач НДР, методології їх рішення та описи технологій застосування інструментальної підтримки цих методологій, а в ролі програмно-інструментальних КПВ – ідентифікатори інструментів або URL (Uniform Resource Locator) – уніфіковані локатори ресурсів. Саме КПВ мають визначати кінцеву конфігурацію ІТ-підтримки конкретної НДР.

Зв'язність моделі. Найбільш визначальними для моделювання ПрО є метод формування зв'язності моделі. Базові концепти ПрО моделюються в Protégé у вигляді класів з атрибутами (слотами), які характеризують їх властивості та відношення між ними, що подаються через ієрархію спадкування класів та їх логічні зв'язки.

Найбільш суттєвими для побудови моделі ПрО постають два типи зв'язку між класами:

- прямі зв'язки (*direct relationships*), що відбивають відношення *ієрархії* успадкування класів у вигляді *Subclass-Superclass*. Такі зв'язки реалізують, окрім усього іншого, принцип *множинного успадкування* класів, що дозволяє при побудові ієрархії класів виділяти відповідні *Use Cases*, тобто *варіанти використання цього класу в контекстах різної проблематики*. Таким чином принцип *множинного успадкування* класів можна використати при виділенні проблемних областей у моделі ПрО.

Прикладом застосування принципу множинного успадкування може служити використання одного і того ж програмно-

інструментального засобу для вирішення задач різних тематичних категорій НДР. Це здійснюється шляхом з'єднання класів через слоти певного типу, прикладом яких можуть бути слоти з іменами вигляду *belong-to* (належить до) або *produce_ something* (виробляючий щось).

Прикладом *другого* типу зв'язків можуть служити зв'язки, які фіксують відношення між класом, що подає програмно-інструментальний засіб, та класом, що подає технологію його використання.

Для формування між класами зв'язків другого типу Protégé подає слоти с типами значень *Instance* або *Class*. Такі зв'язки можна проінтерпретувати наступним чином: якщо клас А містить слот типу *Class* або *Instance*, який містить клас В з множини класів, допущених для слота, то ми кажемо, що клас В залежить від класу А.

Наведені типи слотів можна прокоментувати наступним чином:

- слоти типу *Class*. Значенням їх можуть бути суперкласи, та ні в якому разі *власні* підкласи. Так, якщо на стадії моделювання класу програмно-інструментального засобу, такому слоту надається як значення ім'я суперкласу *Методологія*, то на стадії створення екземплярів такого класу цей слот отримує як значення ім'я одного з підкласів суперкласу *Методологія*, тобто (умовно) *Метод 1*, *Метод 2*,…, …, *Метод N*.
- слоти з типом значення *екземпляр* (*Instance*). Для таких слотів у стадії моделювання ПрО як джерело можливих значень вказується який-небудь клас. На стадії створення екземпляра класу з таким слотом останній отримує як значення ім'я одного з екземплярів прив'язаного до слота класу.

Екземплярізація побудованої моделі ПрО. Як зазначено вище, використанню моделі ПрО передує створення екземплярів класів певних підмножин класів моделі ПрО, що відбивають конкретну тематичну проблематику.

Створення екземплярів класів та формування відношень (*relations*) між екземплярами класів здійснюється через ввід відповідних значень у слоти обраних класів. Результатом створення екземплярів класів моделі ПрО стає формування моделі даних конкретної проблемної області користувача у формі БД.

Рішення завдань користувача моделі ПрО. *Простір рішень* завдань користувача модельованого ПрО визначається сформованою проблемною областю, а окреме рішення подає конфігурацію артефактів e-Science, що становить інфраструктуру ІТ-підтримки НДР користувача. Сама інфраструктура складає собою упорядкований через логічні зв'язки набір КПВ певних типів.

Для рішення такого завдання користувач формує набір запитів (*queries*) до БД, які містять його вимоги та потреби щодо функціональності та ефективності такої підтримки. Самі запити зберігаються у бібліотеці запитів (*Query Library*), припускаючи тим самим їх *повторне використання* і за потребою – *комбінування*. Результати запитів подаються як в екранних форматах редактора Protégé, так і у вигляді текстових файлів.

Запити не становлять частину створеної БД, а є засобом ідентифікації та формування визначеної конфігурації екземплярів, виходячи з властивостей класів і слотів та з вимог користувача.

Структура окремого запиту виглядає наступним чином:

- *Назва запиту – Ім'я класу – Ім'я слота – Меню критеріїв – Значення критерію,*

де

- *Назва запиту* (*Query name*) становить собою рядок довільного тексту, який уводиться користувачем;

- *Ім'я класу* вибирається з меню – переліку класів моделі;

- *Ім'я слоту* вибирається через меню з переліком слотів обраного класу;

- *Меню критеріїв* (*Criteria menu*) містить перелік критеріїв;

- *Значення критерію* уводиться відповідно з типом значення слота обраного класу.

Перелік критеріїв *меню критеріїв* для слотів, найбільш специфічних для визначення зв'язності класів та екземплярів класів, виглядає наступним чином:

- для слота типу *String*: contain, does not contain, is, is not, begin with, end with;
- для слотів типів *Integer* и *Float*: is, is greater than, is less than;
- для слота типу *Symbol*: is, is not;
- для слота типу *Class*: contains, does not contain;
- та, подібним чином, для слота типу *Instance*: contains, does not contain.

Ввід *значення критерію* для слота типу *Instance* дещо відрізняється від інших типів. Так як значення можна надавати:

- один з екземплярів класу, що вказаний в опису слота, або
- назву потрібного користувачу запита з Query Library.

Як приклад, можна розглянути наступну ситуацію. Запит «*Методи моделювання ПрО, що мають максимальну кількість мов форматування результатів*» шукає екземпляри класу *Method*, значення слота *ontoredactor* якого є також результатом запиту «*Онторедактор, кількість мов форматування результатів яких більш ніж 2*». Формально ці запити виглядають наступним чином:

«Онторедактор, кількість мов форматування результатів яких більш ніж 2» – *language_number – is greater than* – 4 та

«Методи моделювання ПрО з максимальною кількістю мов форматування результатів» – *ontoredactor – contains* – «Онторедактор, кількість мов форматування результатів яких більш ніж 2».

Комбінування запитів. Простий запит базується на класі, слоті чи разом на обох. Комбінований запит подається через множину запитів, зв'язаних в одиничний, або через ланцюжок запитів. Це дозволяє обмежувати чи навпаки розширювати результати запитів шляхом комбінування декількох критеріїв. У разі випадку множини запитів додатково постають такі можливості:

- використання опції *Match All* вказує, що будь-який знайдений у БД екземпляр має задовольняти усім критеріям (*перетинання* або AND), специфікованим у множині запитів;
- використання опції *Match Any* вказує, що знайдений у БД екземпляр має задовольняти у крайньому разі одному з критеріїв, специфікованих у множині запитів.

Результатом *одиничного запиту* постає деяка підмножина екземплярів вказаного в запиті класу, яка задовольняє заданому критерію.

Результатом *множини запитів* постає деяка більш обмежена підмножина екземплярів зазначеного в запиті класу, що задовольняє сукупності критеріїв.

Ланцюжок запитів задається для пошуку в БД сукупності екземплярів множини класів, зазначених у зчеплених компонентах ланцюжка. Формування ланцюжка запитів виконується шляхом зчеплення окремих запитів через слоти типу *Instance*, значенням яких задається ім'я потрібного запиту з бібліотеки Query Library Protégé.

Збережені у бібліотеці запити можна використовувати як пов'язуючи значення для слотів типу *Instance*. Результуючий запит шукає екземпляри, у яких значення слота становить результат пошуку.

Необхідність створення запитів такого типу виникає досить часто. Наприклад, з цією метою шукається клас, що має слот типу *Instance*, і виникає потреба ді-

знатись, які з екземплярів цього класу мають певну властивість. Для створення комбінованого запиту треба спочатку створити запит для властивості, а потім створити запит для класу.

Результати запиту експортуються у текстовий файл *protégé_query_results.txt*. За умовчанням, текстовий файл містить набір ідентифікаторів усіх екземплярів та їх класів. Крім того, при формуванні вихідних результатів запитів користувач зазначає необхідність виводу значень потрібних йому *додаткових* слотів екземпляра.

Приклад онтологічного моделювання в Protégé

У цьому розділі розглянуто найбільш визначальні моменти *реалізації* підходу до онтологічного моделювання ПрО ІТ-підтримки НДР, що наведено у цій статті.

1. Базовим концептом, що обумовлює вибір проблемної області, тобто підмоделі моделі ПрО, визначено *тематичну категорію* НДР. У ролі інших базових концептів визначені інформаційні, інформаційно-технологічні ресурси обраної ПрО, а саме – *описи проблеми/задачі НДР*, документи, релевантна література та інструктивні матеріали щодо *методів її рішення і засоби технологгічної та інструментальної підтримки.*

2. *Тематичні категорії наукових досліджень* організовані в ієрархічне дерево *тематичного класифікатора*, вершинам якого у моделі відповідають *абстрактні класи* з іменами категорій, крім кореневого класу, що має ім'я **Topics** (рис. 1).

InfoTechModel: Class Hierarchy

- :THING
  - TOPICS
    - ІНФОРМАТИКА
      - Штучний Інтелект
      - Математичне Моделювання
      - Теорія Ігор
      - Теорія Кодування
    - ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ та ОС
      - Інформаційні Ресурси та Системи
      - Обробка та Аналіз Даних
        - Системи Текстового Пошуку
        - Засоби Побудови Електронних Бібліотек
      - Фундаментальні Основи IT
    - МАТЕМАТИКА
      - Алгебра
      - Векторний Аналіз
      - Диференціальні Обчислення
      - Топологія
    - МЕХАНІКА
      - Теоретична Механіка
    - ФІЗИКА
  - PROBLEM/ TASK
  - METHOD
  - TOOL
  - URLReference

Рис. 1. Ієрархія класів моделі предметної області

3. Іншим базовим концептам поставлено у відповідність *типові структури даних*, визначені у вигляді *конкретних класів* з іменами **Problem/Task**, **Method**, **Tool** та **URLReference**. Дескриптори одного з них показано на рис. 2.

4. Класи (та екземпляри) моделі пов'язані між собою двома засобами – через принцип спадкування і побудови *ієрархічного* дерева та через з'ясування семантичних відношень між класами, які здійснюються через *атрибути* або *слоти* класів. Коренем ієрархічного дерева править системний клас Protégé - **:THING** (*річ*).

5. Семантичні зв'язки потенційно більш виразні та дозволяють аналізувати БД проблемної області більш змістовно (рис. 3). Граф, що відбиває зв'язки такого типу, дозволяє через досить складні запити отримувати замість окремих КПВ цілу низку КПВ, що становить шуканий результат.

6. Зв'язки, що формуються слотами першого типу, визначають *семантичну належність* екземплярів класу – власника слота до іншого класу, наприклад, екземпляр класу **Problem/Task** через значення слота **sub_location** має належати до певної тематичної категорії, яка відбивається одним з абстрактних класів тематичного класифікатора (рис. 3).

7. Відношення, що встановлюються слотами другого типу, визначають *семантичний контроль* екземпляра класу-власника слота над певною множиною (у загальному випадку) екземплярів зв'язаного класу. Наприклад, екземпляр класу **Method** через значення свого слота типа *Instance* визначає можливість використання для своєї підтримки множини релевантних екземплярів класу **Tool**.

8. Зв'язки другого виду реалізовані через слоти типу *Class*, значеннями яких мають бути класи; і типу *Instance,* значеннями яких мають бути екземпляри класів; та слотів, що формують *інверсне* відношення між екземплярами двох класів шляхом зв'язування їх через слоти *Instance*.

9. *Інверсний* зв'язок між екземплярами пари класів, пов'язаних через слоти типу *Instance*, формується через визначення для цих слотів *обмежувачів ролі* (*фацетів*). Значеннями цих фацетів мають бути вирази вигляду *inverse-slot = <ім'я слоту зв'язаного класу>*. Такий зв'язок взаємно синхронізує зміну значень слотів під час створення нових екземплярів цих класів.

Template Slots

	Slot Name	Documentation	Type	Cardinality
	method_name	Стисле ім.'я Методу	String	0:1
	method_annotatation	Анотований опис Метода	String	0:1
	problem_task	Проблема/Задача, що вирішується за підтримки даного методу. Метод може підтримувати рішення декількох проблем	Problem/Task	0:*
	sub_category_method	Тематична категорія методу	String	0:1
	meth_keyword_list	Список ключових слів	{}	0:*
	method_documentation	Список адрес документів по методу	URLReference	0:*
	meth_relevant_info	Список адрес джерел релевантної інформації	URLReference	0:*
	tool_list	Список інструментальних засобів підтримки методу	Tool	0:*
	method_consultant		URLReference	0:*

Рис. 2. Class Method. Дескриптор класу Метод

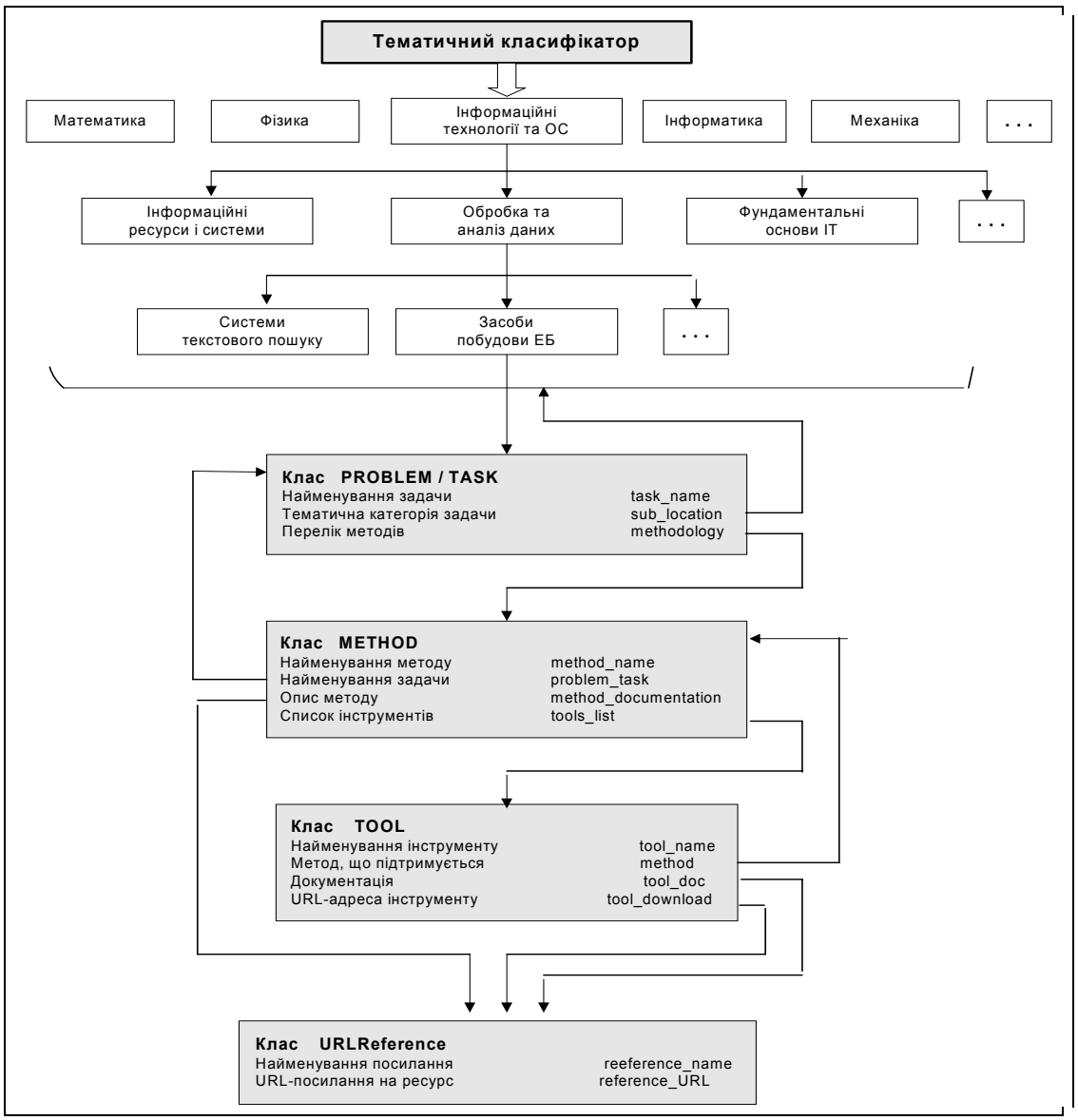

Рис. 3. Структура графа узагальненої моделі ПрО IT-підтримки НДР

Висновки

Якщо перші етапи розглянутої методології автоматизувати важко, то всі інші, починаючи з формування та вводу запитів до БД проблемної області, становлять реальну перспективу автоматизації.

Для переходу від інтерактивного формування та вводу запитів у БД проблемної області до автоматизованого найбільш ефективною і перспективною є орієнтація на розробку відповідного програмного забезпечення мовами Java та SPARQL. Так SPARQL (*SPARQL Query Language for Protégé*) [3] подає собою мову запитів до даних, наведених за моделлю RDF, і може бути також використана до моделей у форматі *RDF Schema* і *OWL*. Ця мова забезпечує протокол для передачі цих запитів та відповідей на них і рекомендована консорціумом W3C Working Draft як стандарт та одна з технологій семантичної павутини.

Проте застосування мови SPARQL має свої вади, наприклад, SPARQL вбудований до Protégé версії не вище 3.2, і Protege 4 не підтримує запити на цій мові. Крім того, використання вбудованої до Protégé мови SPARQL, всупереч її позитивним якостям, має той же недолік –

її використання підтримується лише в інтерактивному режимі. Подолання цих вад SPARQL та автоматизація роботи з запитами на цій мові можливо через створення відповідних *Java*-застосувань.

Застосований підхід використовується в інтегрованому середовищі генеруючого програмування на платформі Eclipse проекта «Розробка теоретичних основ генеруючого програмування, прикладних та інструментальних засобів його підтримки» [4]. Цей підхід забезпечує подання компонентів повторного використання при описі моделей предметних областей різного призначення. В даному середовищі Eclipse підтримує формати RDF та OWL і запити SPARQL.

1. Protégé-Frames User's Guide. http://protégé.stanford.edu/doc/frames/
2. Protégé-OWL API Programmer's Guide. http://protege.stanford.edu/doc/owl/guide.html
3. SPARQL Query Language for RDF. http://www.w3.org/TR/rdf-sparql-query/
4. *Лавріщева Е.М.* Генерувальне програмування програмних систем і сімейств // Проблеми програмування. – 2009. – № 1. – С. 3–16.

Отримано 10.05.2011

Про автора:

Зінькович Валерій Митрофанович,
провідний інженер-програміст.

Місце роботи автора:

Інститут програмних систем
НАН України,
03187, Київ-187,
проспект Академіка Глушкова, 40.
Тел. 044 526 3098.
e-mail: vmz45@ukr.net