УДК 004.94 https://doi.org/10.15407/pp2020.02-03.031

МОДЕЛЬ ІНФОРМАЦІЙНОГО ОБ'ЄКТА ДЛЯ ЕЛЕКТРОННОЇ БІБЛІОТЕКИ ТА ЇЇ ВЕРИФІКАЦІЯ

О.В. Новицький, В.А. Резніченко

Запропоновано підхід формальної верифікації UML 2.0 через відображення OWL-DL в UML 2.0. У результаті розроблено оригінальний метод до відображення OWL-DL в UML 2.0 через дескриптивну логіку. Забезпечено повноту відображення UML-OWL через стереотипи та мічені значення UML 2.0 на рівні M0, M1 метамоделі MOF. Запропоновано модель інформаційного об'єкта для семантичної електронної бібліотеки, яка побудована з використанням мови UML. Також здійснено перевірку запропонованої моделі шляхом відображення моделі UML в OWL з наступною валідацією побудованої онтології за допомогою різонерів.

Ключові слова: електронна бібліотека, семантична електронна бібліотека, зв'язані дані, інформаційний об'єкт, відображення UML-OWL, дескриптивна логіка, різонер.

Предложен подход формальной верификации UML 2.0 с использованием отражения OWL-DL в UML 2.0. В результате разработан оригинальный метод к отображению OWL-DL в UML 2.0 через дескриптивную логику. Обеспечена полнота отображения UML-OWL через стереотипы и меченые значение UML 2.0 на уровне M0, M1 метамодели MOF. Предложена модель информационного объекта для семантической электронной библиотеки, которая построена с использованием языка UML. Также осуществлена проверка предложенной модели путем отражения модели UML в OWL и последующей валидации построенной онтологии с помощью ризонеров.

Ключевые слова: электронная библиотека, семантическая электронная библиотека, связанные данные, информационный объект, отображение UML-OWL, дескриптивная логика, ризонер.

An approach for formal verification of UML 2.0 using mapping OWL-DL in UML 2.0 is proposed. As a result, an original approach for mapping OWL-DL to UML 2.0 through description logic has been proposed. The completeness of the mapping of UML-OWL through stereotypes and labeled UML 2.0 values at the level of M0, M1 of the MOF metamodel is provided. A model of the information object (IO) for the semantic electronic library, which is described by using the UML language, is proposed. The proposed IO model was also verified by mapping it into OWL and then validating the constructed ontology by using risoners.

Key words: digital library, semantic digital library, linked data, information object, mapping UML-OWL, description logic, reasoning engine.

Вступ

Електронні бібліотеки наразі функціонують в мережі Інтернет. І на даний момент розвитку науки існує необхідність публікувати інформаційні ресурси в мережі Інтернет. Для публікації семантичних даних, широкого розповсюдження набула методологія зв'язаних даних (LD) [1].

Щоб опублікувати дані в Інтернеті, ми спочатку повинні ідентифікувати елементи, що представляють інтерес в нашому домені. Вони є сутності чиї властивості і відносини ми хочемо описати в даних. В термінології Веб Архітектури, всі елементи, що представляють інтерес, називаються ресурсами.

У великому різноманітті технологій, що пов'язані з Semantic Web, важливо встановити співвідношення в якому перебувають LD до цих технологій. Зв'язані дані дають бачення використання Інтернету для підключення відповідних даних, які раніше не були пов'язані між собою, або використовуючи Web, знизити бар'єри для зв'язування даних, які в даний час пов'язані з використанням інших методів. Або більш конкретно, LD – це підхід, який використовується для опису методів для виявлення, спільного використання та підключення частин даних, пошуку інформації та знань в Semantic Web, використовуючи URIs, SPARQL і RDF [1, 2].

На даний момент, нема чіткого розуміння того, що являє собою ЕБ в середовищі Semantic Web. Багато робіт були присвячені даній проблемі. Зокрема в [3] робиться більше акцент на використання онтологій та побудови онтологій для ЕБ. В роботі [4] розглядається розвиток ЕБ в напрямку семантичних соціальних ЕБ, в яких значна увага приділяється обміну знань та більш потужним механізмам взаємодії користувачів і проникнення соціальних комунікацій в семантичні ЕБ (СЕБ).

Розробка моделі даних ЕБ в рамках підходу LD дасть змогу наблизити електронні бібліотеки до повноцінної реалізації Semantic Web.

Однак, коли виникає питання розробки програмного забезпечення, одним із центральних аспектів є формалізація вимог до програмного продукту. В такій ситуації, широкого застосування набув UML (англ. Unified Modeling Language) це уніфікована мова моделювання, використовується у парадигмі об'єктно-орієнтованого програмування. Вона є невід'ємною частиною уніфікованого процесу розробки програмного забезпечення. UML є мовою широкого профілю, це відкритий стандарт, що використовує графічні позначення для створення абстрактної моделі системи, названої UML-моделлю. UML був створений для визначення, візуалізації, проектування й документування в основному програмних систем. UML не є мовою програмування, але в засобах виконання UML-моделей як інтерпретованого коду можлива кодогенерація.

UML є стандартизованим та формалізованим засобом для розробки та аналізу програмного забезпечення. Для підтримки розробки великих додатків існують складні CASE інструменти, які забезпечують зручні умови для редагування, зберігання і доступу до діаграм UML. Однак в таких інструментах відсутні засоби інтелектуального аналізу. Для UML характерно надлишковість та протиріччя, які важко виявити. Вважається, що UML діаграма класів, є одним з найбільш важливих компонентів UML. Для перевірки моделей, створених за допомогою діаграми класів UML, необхідно провести над цією моделлю логічні судження. Тобто перевірка на несуперечність, здійснимість, класифікацію та реалізацію концептів.

В роботі [5] представлено формальний апарат для вираження діаграми класів засобами дескриптивної логіки ALCQI. В даній роботі ми намагаємось побудувати UML модель інформаційного об'єкту (ІО) для ЕБ. Для перевірки цієї моделі, ми будуємо відображення UML до Дескриптивної логіки з подальшим відображенням до OWL-DL.

Огляд досліджень з проблематики верифікації UML

Існують різні методології верифікації UML моделей діаграм класів. Так, найбільш простим є підхід, пов'язаний з побудовою драйвера [6]. Даний підхід дозволяє формально оцінити правильність створення діаграми класів і виявити найбільш характерні помилки. Ще одним підходом до верифікації діаграми класів є метод шаблонів, який запропонований в [7]. Поняття шаблону в даному підході відрізняється від добре відомого і зрозумілого поняття шаблону проектування, в даній роботі запропоновано декілька шаблонів помилок для виявлення протиріч у моделі. В роботі [8] розглянуто підхід на основі побудови ідентифікаційного графа. Суть даного методу полягає у побудові ідентифікаційного графа, що представляє собою орієнтований граф. Вузлами даного графа є класи, а також асоціативні зв'язки між ними, а дуги пов'язують асоціації з тими класами, між якими на діаграмі класів і вказані відповідні асоціативні зв'язки.

При виникненні питання щодо розроблення програмного забезпечення та інтеграції класичних ЕБ до СЕБ, одним із центральних аспектів є формалізація вимог до моделі ІО. У такій ситуації широкого застосування набула UML. В базис UML покладено стандарт MOF (мета-об'єктний засіб). Мета-об'єктний засіб (Meta-Object Facility) – це стандарт для розробки, керованої моделями, розроблений OMG (Ontology Definition Metamodel) [9]. MOF виник у процесі розроблення UML. OMG потребував засобу архітектурного метамоделювання для визначення UML. MOF реалізовано як чотиришарову архітектуру моделей, кожна з яких характеризується як екземпляр рівня, який вище розташований (рис. 1).

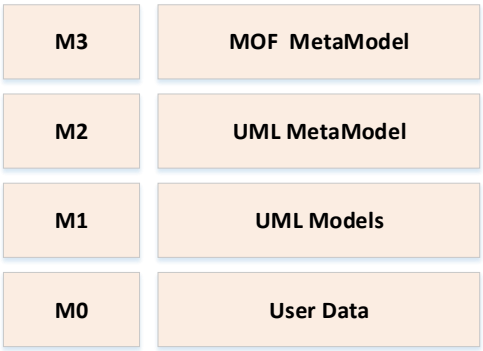

Рис. 1. Чотирьохрівнева архітектура метамоделі

Найнижчий рівень, M0, містить дані користувача – фактичні об'єкти реального світу, якими і має маніпулювати програмне забезпечення. Наступний рівень, M1 визначають як рівень, що містить моделі, призначені для даних користувача рівня M0. Рівень M2 містить моделі інформації, що зберігається на рівні M1. Нарешті, рівень M3 містить модель інформації, що зберігається на рівні M2, і має назву метамодель. Класичним прикладом моделі рівня M2 є UML.

Для підтримання розробки великих додатків існують складні інструменти, що забезпечують умови для редагування, зберігання й доступу до діаграм UML. Однак у таких інструментах відсутні засоби інтелектуального аналізу. Для UML характерні надлишковість та протиріччя, які важко виявити. Діаграма класів UML є одним з найбільш важливих компонентів UML. Для перевірки моделей, створених за допомогою діаграми класів UML, необхідно провести відповідно до цієї моделі логічні судження. В UML немає вбудованих засобів для її верифікації та валідації. Проте існують певні погляди вирішення цієї проблеми [10, 11]. Завдяки використанню семантичних методів і здійсненню відображення моделі UML в OWL, верифікація та валідація зводяться до перевірки на несуперечність, здійснимість, класифікацію та реалізацію концептів.

Представлення інформації про домен засобами OWL та UML

Мови OWL та UML були розроблені для різних цілей. OWL призначена для подання знань про інформаційний складник предметної сфери, UML розроблено насамперед для підтримання розробки

програмного забезпечення. Хоча мови різні, але основна їх мета – формальне представлення знань. У ранніх роботах [12–14] було висвітлено, як можливо визначати онтологію засобами UML. В UML є кілька вбудованих засобів для формалізації семантики. Наприклад, Object Constraint Language (OCL) [15] є декларативною мовою описання правил, що використовуються в UML. Вона, на відміну від UML, має лінійну нотацію, а не графічну. Однак, так само, як і в UML, в OCL відсутня формальна модель семантики, теоретична модель та формальне доведення теорем. Тому в UML не може бути механізмів для автоматизації міркувань.

Водночас консорціум Object Management Group розробив метамодель визначення онтології (Ontology Definition Metamodel – ODM), яка визначає набір мета-моделей UML [16] і профілів для представлення UML в RDF і OWL. UML профілі в специфікації ODM адаптують нотації UML, щоб надати форму візуального представлення для RDF і OWL.

У роботах [5, 17] здійснено спроби щодо трансформування діаграми класів UML до DL (Дескриптивная логика). Проте в цих роботах повнота відображення не знайшла місце.

Представлення знань та інформації про розроблення програмного забезпечення, як правило, мають різні цілі щодо запису інформації. Робляться відповідні, але різні припущення щодо інтерпретації мовних висловів чи заяв. Безліч припущень впливають на семантичну відповідність між мовними конструкторами і їх нотаціями. Якщо в OWL такі невідповідності можуть бути усунені і вирази є більш однозначними, то в UML вони більш різноманітні. UML дозволяє різні інтерпретації мовних конструкцій залежно від точки зору.

Існують суттєві відмінності між UML та OWL [18].

У діаграмі класів UML виходять з припущення про закритість світу. Тобто всі твердження, що не були явно вказані, є хибними. Натомість OWL використовує припущення відкритого світу. Припущення ж про відкритість світу говорить, що деяке твердження є не істинним, не хибним.

UML має поняття профілів, що уможливлюють розширювати мета-моделі елементів UML. Водночас OWL не має відповідної конструкції. У більшості випадків профілі UML використовуються для визначення стереотипів, щоб розширити класи. Представлення цих стереотипів може бути відображено в OWL через створення низки нових класів та узагальнення тверджень. Однак більша частина профілю UML доволі специфічна та вимагає відповідних правил перетворення, адаптованих для певного профілю.

Абстрактні класи UML не можуть бути перетворені в OWL. Якщо клас визначається як абстрактний в UML, то це означає, що екземпляри цього класу (об'єкти) не можуть бути створені. В OWL не закладено функцію, яка вказувала б на те, що клас не повинен містити екземплярів. Одним із принципів до збереження семантики абстрактного класу є використання DisjointUnion. Це гарантуватиме, що будь-який екземпляр, який належить до підкласу, також відноситься до абстрактного суперкласу. Та це не забороняє створювати екземпляри абстрактного суперкласу.

В UML видимість елементів моделі може бути зменшена шляхом маркування їх як public, private і т. д. Також можна оголосити елементи UML моделі, які доступні тільки для читання. В OWL відсутній механізм управління, для обмеження доступу до елементів моделі. OWL онтології також не можуть містити будь-яких операцій.

В OWL можливо визначати властивості об’єкта на рівні онтології, які не обов’язково мають бути прив’язані до класу. Водночас, UML пропонує два способи прив’язування атрибутів: через атрибут класу та через асоціації. Як випливає з назви, у першому випадку атрибут класу відноситься до класу. Асоціації визначені на рівні пакету діаграми класів. Щоправда, такі асоціації повинні мати на полюсах класи як типи. Тому UML-асоціації не повністю підходять для представлення загальних властивостей об'єкта.

Внаслідок цих концептуальних розбіжностей між UML та OWL неможливо побудувати однозначне відображення. У деяких роботах є спроби розширити OWL до відповідного діалекту, що відповідав би UML. В цій роботі запропоновано розширення UML через додаткові нотації та стереотипи, для збільшення повноти відображення.

Здійснене відображення базується на таких припущеннях:

– розглядаються тільки бінарні зв'язки між класами/концептами; зазначимо, що це ніяк не обмежує узагальненості, оскільки відомо, що n-арні зв'язки можуть бути представлені у вигляді сукупності бінарних зв'язків;

– не розглядаються питання опису відображення операцій (методів) класів UML.

Схема відображення діаграми класів UML у конструкції DL базується на таких принципах:

– клас UML представляється атомарним концептом DL;

– атрибут класу представляється роллю;

– бінарна асоціація представляється роллю;

– асоціативний клас бінарної асоціації представляється класом і двома асоціаціями;

– n-арна асоціація представляється за допомогою процедури реїфікації у вигляді додаткового концепту і його рольових зв'язків з концептами, відповідними класами, що входять до складу n -арної асоціації;

– відношення узагальнення/спеціалізації представляються термінологічними аксіомами включення DL.

Для повноти відображення UML в OWL-DL через ДЛ пропонується використовувати діалект SHOIQ, на якому базується OWL-DL. Це відображення описане в роботі [18].

Побудова та перевірка еталонної моделі інформаційного об'єкту ЕБ

В основу моделі ІО покладено ідею абстрагування ресурсу від його реалізації та уніфікації з сервісом. Для нашої моделі використаємо ресурсо-орієнтований погляд, що означає дуалізм при взаємодії з ІО. Тобто будь-який сервіс намагатиметься представитися у вигляді ІО, при цьому кожного разу такий сервіс буде обчислювати своє представлення у вигляді ІО.

У контексті зв'язаних даних будь-яке звернення до ресурсу має бути представлено як набір даних у вигляді певного графу. При цьому повинна забезпечуватися сумісність з такими мовами запитів, як SPARQL. Відповідна взаємодія може покритися архітектурним стилем REST (Representational State Transfer). У цій роботі пропонується об'єднати концепцію зв'язаних даних та концепцію сервіс-орієнтованої архітектури до побудови моделі ІО. Це дозволить маніпулювати динамічними інформаційними ресурсами в межах однієї моделі. Концептуально REST є спробою побудови архітектури, в якій сучасній архітектурі мережі Інтернет надається можливість перейти на новий рівень взаємодії між вузлами. В основі REST передбачені такі обмеження до моделювання ІО:

– масштабованість окремих компонентів та їх взаємодія;

– клієнти відокремлені від сервера єдиним інтерфейсом – розділення відповідальності означає, що клієнти не відповідають за сховище даних, що є внутрішнім для кожного сервера;

– взаємодія клієнт-сервера обмежується відсутністю збереження контексту клієнта на сервері між запитами. Кожен запит від будь-якого клієнта містить усю інформацію, необхідну для його обслуговування, а будь-який стан сесії зберігається в клієнті. Сервер може бути в певному стані. Це обмеження вимагає, щоб стан на стороні сервера ідентифікувався через URL. Це не тільки робить сервери більш видимими для моніторингу, а й робить їх більш надійними в разі часткової відмови мережі. Тобто клієнти не пов'язані із зберіганням даних, ця функція залишається за сервером. Сервери в свою чергу не пов'язані з інтерфейсом або станом користувача;

– можливість кешування – клієнти можуть кешувати відповіді;

– багаторівневість системи;

– легкість у розгортанні сервера;

– єдиний інтерфейс між клієнтами і серверами спрощує й розділяє архітектуру, даючи змогу кожній частині розвиватися самостійно.

Проте Semantic Web, зокрема реалізація мережі зв'язаних даних, вносить певні зміни в проектування REST взаємодії між серверами та клієнтами. Оскільки LD є розподіленою мережею, то нівелюються розділення на клієнт-серверну частину, тому що кожен вузол може одночасно виконувати роль і клієнта і сервера. Це пов'язано з тим, що набори даних на різних серверах можуть легко об'єднуватися між собою, утворюючи певну мережу. Також дані в LD є відкритими – це означає: крім того, що кожен може ввести певні зміни в LD, у будь-який момент часу в онтологіях, на яких базується LD, можуть бути додані чи видалені відношення чи твердження, які будуть пливати на мережу LD+REST. З поміж них можна виділити такі, що подано у таблиці.

Таблиця

REST	REST + LD
Клієнт-серверна архітектура передбачає незалежність запитів між собою, причому як клієнт, так і сервер може мати власні стани	Виконання запитів SPARQL може призводити до зміни станів будь-якого вузла мережі, отже, це суперечить принципам REST
Головною особливістю, яка відрізняє REST архітектурний стиль від інших мережевих стилів, є акцент на єдиний інтерфейс між компонентами. Ця особливість є найбільшою відмінністю. Завдяки застосуванню під час розроблення програмного забезпечення принципу уніфікованості інтерфейсу загальна архітектура системи спрощується і видимість взаємодії поліпшується. Реалізація відокремлених послуг сприяє незалежному розвиткові. Компроміс полягає в тому, що універсальний інтерфейс знижує ефективність, бо інформація передається в стандартизованій формі, а не у специфічній для потреб програми	В LD визначено фактично декілька інтерфейсів взаємодії з даними. Усі вони базуються на протоколі HTTP, але мають різне призначення. Зокрема, RDF може мати різний тип спеціалізації (JSON/XML). Водночас доступ до даних може відбуватися або через пряме звернення до ресурсу, або через точки доступу SPARQL

На модель інформаційного об'єкта, що пропонується (рис. 2), накладаються деякі обмеження: інформаційний об'єкт (ІО) функціонує у веб-середовищі. Це означає, що доступ до ресурсів відбувається за протоколом HTTP. Центральним об'єктом моделі є ІО, який фактично є контейнером для інформаційного наповнення. Тобто контент інформаційного ресурсу міститься в ІО. Кожен ІО унаслідується від інформаційного ресурсу. Інформаційний ресурс необхідний, щоб надати можливість ІО отримати низку додаткових

властивостей. Основна функція інформаційного ресурсу – це забезпечення доступу до ІО за протоколом HTTP. У межах моделі передбачається віддалений доступ за ідентифікатором URI. Між ІО можуть бути бінарні відношення, предикат для цих відношень задається в зовнішніх онтологіях. Ці бінарні відношення дозволяють розкривати семантику ІО через визначення взаємозв'язків з іншими ІО. Наприклад, вони можуть визначати видавця ІО. Бінарні відношення моделюють RDF-трійки.

Проте виділено окремий тип відношень, які агрегують ІО в колекції. Це пов'язано з тим, що в ЕБ ресурси, як правило, зв'язані між собою певними визначеними сутностями. Наприклад, у багатьох ресурсах може бути один і той же автор, або ресурси можуть відноситися до одного і того ж предметного класифікатора [20].

Тому виникає необхідність групувати ресурси за певними суттєвими спільними ознаками. Ці групи називаються колекції [20]. Один і той же ІО може входити до різних колекцій.

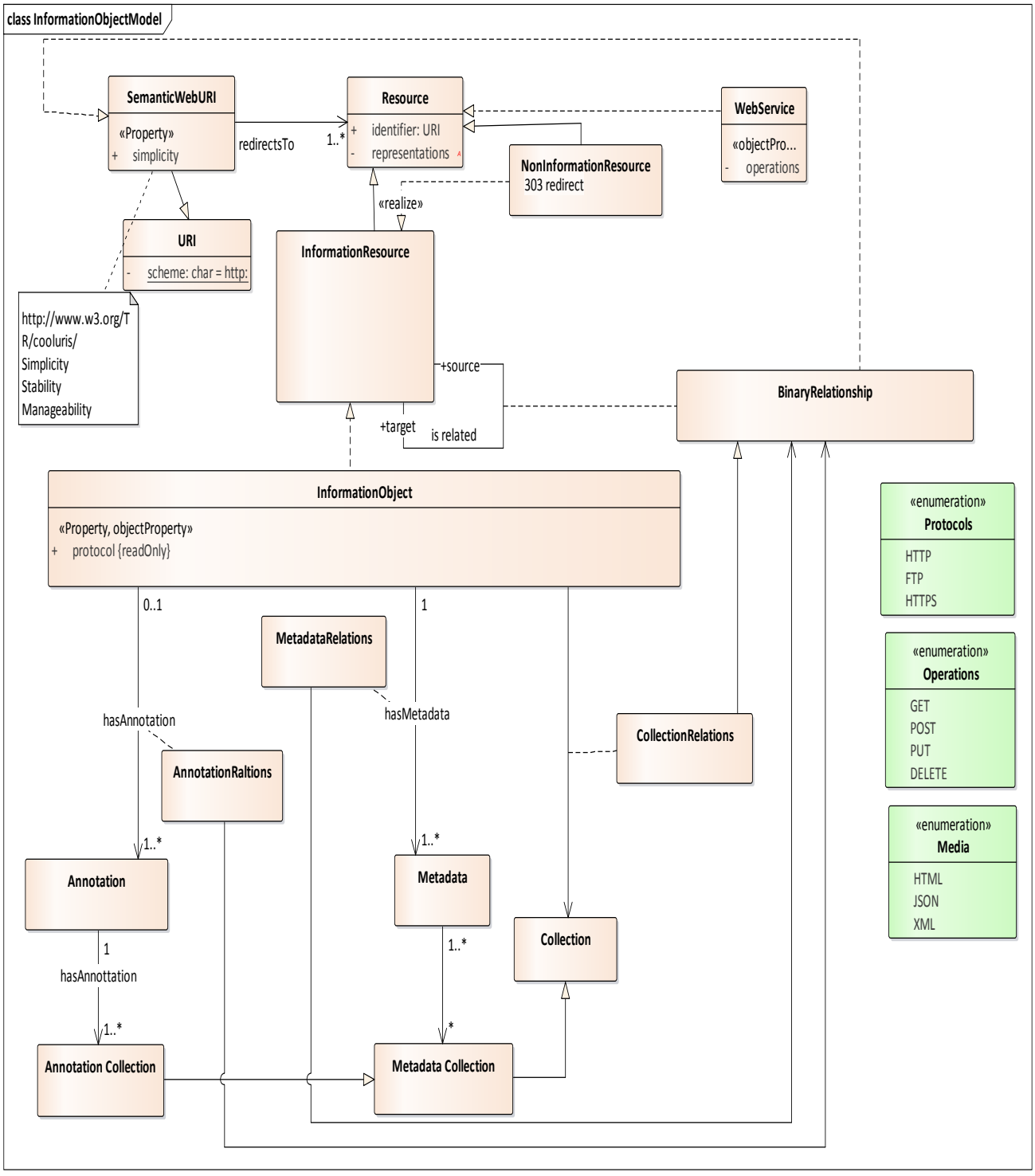

Рис. 2. Модель інформаційного об'єкта в UML

В ЕБ представлено множину інформаційних ресурсів, які описують певні публікації, кожна з яких описується певним набором ІО. Модель також покриває сервіс-орієнтовану архітектуру в межах REST. Кожен стан веб-сервісу в моделі розглядається як певний інформаційний ресурс. ІО можуть поєднуватися в певні набори або так звані колекції. Тобто оболонкою для ІО в моделі є інформаційний ресурс і саме наповнення інформаційного ресурсу міститься в ІО. Кожен ІО унаслідується від інформаційного ресурсу.

Відповідне відображення до OWL має вигляд, показаний на рис. 3.

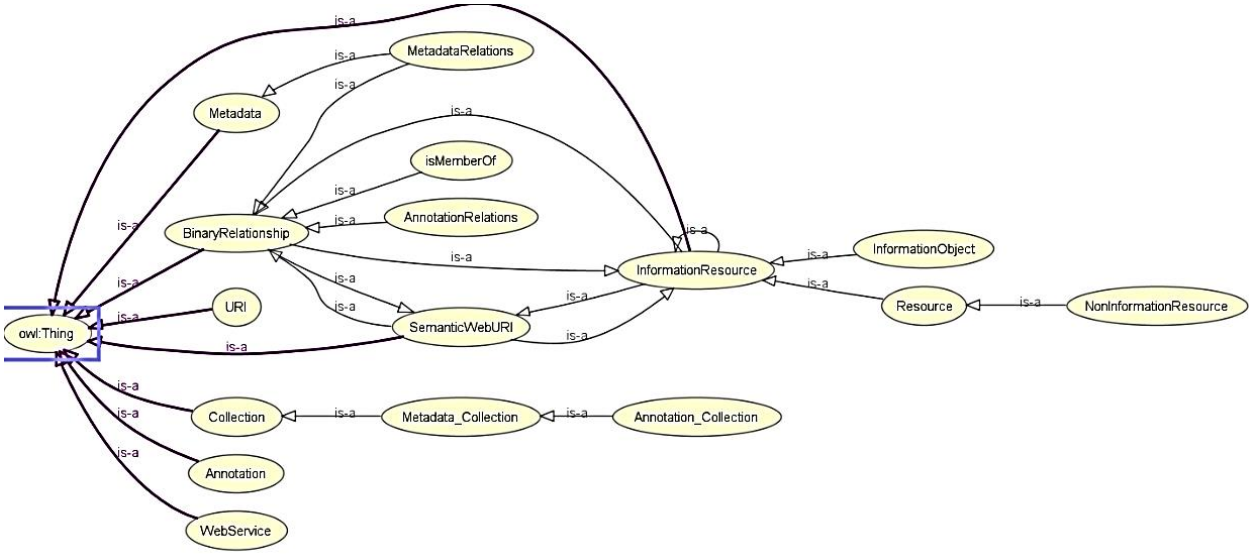

Рис. 3. Модель інформаційного об’єкту в OWL

Для верифікації розробленої моделі запропоновано метод, суть якого полягає у наступному. Спочатку кожен клас UML відображаються в концепт ДЛ SHOIQ. На наступному етапі відбувається відображення для всіх відношень та атрибутів діаграми класів UML в ролі ДЛ. Згодом всі вирази ДЛ переносяться в Protégé. Розроблена модель пройшла перевірку на когерентність і послідовність такими машинами виводу, як Pellet [21] та HermiT [22, 23]. Обидві машини виводу використовують подібні алгоритми, засновані на методі таблиць (tableau based reasoner та hypertableau calculus). Тому можна стверджувати, що розроблена модель ІО ЕБ пройшла формальну верифікацію (рис. 4).

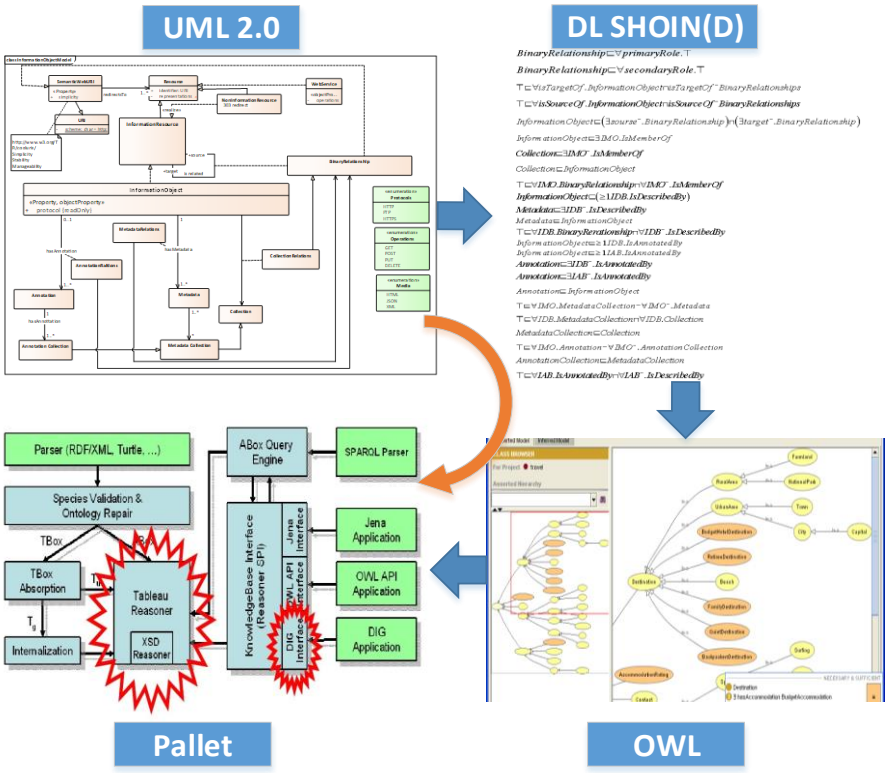

Рис. 4. Схема верифікації розробленої моделі ІО

Висновки

Одним із шляхів розв'язання проблеми інтеграції даних в ЕБ є використання моделі зв'язаних даних Linked Data (LD). Перевага зв'язаних даних полягає у тому, що цінність і корисність даних збільшується відповідно до того, наскільки більше вони пов'язані з іншими даними. Більшість класичних ЕБ побудовані з використанням реляційної БД. Проте сучасні підходи до відображення реляційних баз даних до зв'язаних даних не враховують специфіку схем бази даних класичних ЕБ. Для вирішення цієї задачі необхідно розробити відповідний метод відображення. Окрім цього актуальною є проблема побудови моделі ІО та її верифікації. Для вирішення даних задач, було отримано наступні результати. Розроблено метод формальної верифікації UML 2.0 через відображення OWL-DL в UML 2.0. У результаті розроблено оригінальний метод до відображення OWL-DL в UML 2.0 через ДЛ. Забезпечено повноту відображення з OWL-DL через стереотипи та мічені значення UML 2.0 на рівні M0, M1 метамоделі MOF. Запропоновано модель інформаційного об'єкта для СЕБ, яка побудована з використанням мови UML. Також здійснено перевірку запропонованої моделі шляхом відображення моделі UML в OWL з метою валідації цієї моделі засобами машин-суджень.

Література

1. Linked Dat, 07 2015. Available from: http://www.w3.org/standards/semanticweb/data.
2. Berners-Lee T. Linked Data. 2009. Available from: http://www.w3.org/DesignIssues/LinkedData.html. [Accessed: 07 2015].
3. Sure Y., Studer R. Semantic Web technologies for digital libraries. *Library Management*.2005. Vol. 4/5, N 26. P. 190–195.
4. Alotaibi S. The 4th Saudi International Conference. In Semantic Web Technologies for Digital Libraries: From Libraries to Social Semantic Digital Libraries (SSDL), Over Semantic Digital Libraries (SDL). Manchester. 2010.
5. Daniela B., Diego C., Giuseppe D. Reasoning on UML class diagrams. *Artificial Intelligence*. 2005. Vol.168, N 1–2.
6. Макгрегор Дж. Сайкс Д. Тестирование объектно-ориентированного программного обеспечения. Практическое пособие: Пер. с англ. / Джон Макгрегор, Девид Сайкс. К.: ООО «ТИД «ДС». 2002. 432 с.
7. Sturm A., Balaban M., A. Maraee. Management of Correctness Problems in UML Class Diagrams Towards a Pattern-Based Approach. *International Journal of Information System Modeling and Design*. 2010. Vol. 1. N 4. P. 24–47.
8. Thalheim B. Foundations of entity-relationship modeling. Annals of Mathematics and Artificial Intelligence. 1993. Vol. 7. N 1–4.
9. OMG's Meta Object Facility. Available from: http://www.omg.org/mof/.
10. Волович М.Е., Дерюгина О.А. Верификация UML-моделей программных систем. *Cloud of Science*. 2015. Vol. 2. N 1. P. 138–146.
11. Новицький О. Відображення класів UML до дескриптивної логіки для семантичного моделювання об'єктів. В Інформація, комунікація, суспільство 2016: матеріали 5-ої Міжнародної наукової конференції ІКС-2016, 19–21 травня 2016 року, Україна, Львів, Славське. 2016. С. 62–63.
12. Andrea C., Diego C., Giuseppe D.G., Maurizio L. A Formal Framework for Reasoning on UML Class Diagrams. In Foundations of Intelligent Systems, T. 2366, Springer Berlin/Heidelberg, 2012.
13. Cranefield D., Purvis A. 16th International Joint Conference on Artificial Intelligence (IJCAI-99) In Uml as an ontology modelling language, 1999.
14. Brody T., Stamerjohanns H., Vallieres F., Harnad S., Yves G., Oppenheim C. National Policies on Open Access (OA) Provision for University Research Output: an International meeting. In The effect of Open Access on Citation Impact, Southampton, 2004.
15. Object Management Group. Available from: http://www.omg.org/spec/OCL/.
16. Object Management Group, «Documents associated with Ontology Definition Metamodel (ODM) Version 1.0,» 05 2009. Available from: http://www.omg.org/spec/ODM/1.0/. [Accessed: 07 2015].
17. Simmonds J. Consistency Maintenance of UML Models with Description Logics, 2003.
18. Новицький О.В. Розширення UML специфікації для моделювання семантичних об'єктів. *Проблеми програмування*. 2016. N 2–3. С. 211–219.
19. OMG Unified Modeling LanguageTM (OMG UML), Superstructure Version 2.4.1. Available from: https://www.omg.org/spec/UML/2.4.1/About-UML/.
20. Наукові засади та теоретико-методологічні принципи створення сучасних енциклопедій: колективна монографія / За ред. д-ра іст. наук, проф. Киридон А. М. – К. : Державна наукова установа «Енциклопедичне видавництво». 2015. 160 с.
21. Sirin E., Parsia B., Cuenca Grau B., Kalyanpur A., Yarden K. Pellet: A practical OWL-DL reasoner. *Journal of Web Semantics*. 2007. T. 5, N 2. P. 51–53.
22. Shearer R., Motik B., Horrocks I. HermiT: A Highly-Efficient OWL Reasoner. *OWLED*. 2008. V. 432. P. 91.
23. Новицький О.В. Сервіс-орієнтовані, розподілені системи реального часу в електронних бібліотеках (ЕБ). *Проблеми програмування*. 2013. № 2. С. 87–94.

References

1. Linked Dat, 07 2015.: Available from: http://www.w3.org/standards/semanticweb/data.
2. Berners-Lee T. Linked Data. 2009. Available from: http://www.w3.org/DesignIssues/LinkedData.html. [Accessed: 07 2015].
3. Sure Y., Studer R.. Semantic Web technologies for digital libraries. Library Management.2005. Vol. 4/5, N 26. P. 190–195.
4. Alotaibi S. The 4th Saudi International Conference. In Semantic Web Technologies for Digital Libraries: From Libraries to Social Semantic Digital Libraries (SSDL), Over Semantic Digital Libraries (SDL). Manchester. 2010.
5. Daniela B., Diego C., Giuseppe D. Reasoning on UML class diagrams. Artificial Intelligence. 2005. Vol.168, N 1–2.
6. McGregor John D., Sykes David A. A Practical Guide to Testing Object-Oriented Software. Addison-Wesley Longman Publishing Co., Inc. 2001. 416 p.
7. Sturm A., Balaban M., A. Maraee. Management of Correctness Problems in UML Class Diagrams Towards a Pattern-Based Approach. *International Journal of Information System Modeling and Design*. 2010. Vol. 1, N 4. P. 24–47.
8. Thalheim B. Foundations of entity-relationship modeling. Annals of Mathematics and Artificial Intelligence. 1993. Vol. 7, N 1–4.
9. OMG's MetaObject Facility. Available from: http://www.omg.org/mof/.

10. Volovich M.E., Deryugina O.A. UML models of software systems verification. *Cloud of Science*. 2015. Vol. 2, N 1. P. 138–146.
11. Novitskyi O.V. Mapping UML classes to the description logic for object semantic modelling. Information, Communication, Society 2016: Proceedings of the 4th International Scientific Conference ICS-2015, May 20-23, 2015, Ukraine, Lviv, Slavske, 2016. P. 62 – 63.
12. Andrea C., Diego C., Giuseppe D.G.. Maurizio L. A Formal Framework for Reasoning on UML Class Diagrams. In Foundations of Intelligent Systems, т. 2366, Springer Berlin/Heidelberg, 2012.
13. Cranefield D., Purvis A. 16th International Joint Conference on Artificial Intelligence (IJCAI-99) In Uml as an ontology modelling language, 1999.
14. Brody T., Stamerjohanns H., Vallieres F., Harnad S., Yves G., Oppenheim C. National Policies on Open Access (OA) Provision for University Research Output: an International meeting. In The effect of Open Access on Citation Impact, Southampton, 2004.
15. Object Management Group. Available from: http://www.omg.org/spec/OCL/.
16. Object Management Group, «Documents associated with Ontology Definition Metamodel (ODM) Version 1.0,» 05 2009. Available from: http://www.omg.org/spec/ODM/1.0/. [Accessed: 07 2015].
17. Simmonds J. Consistency Maintenance of UML Models with Description Logics, 2003.
18. Novitskyi O.V. Extending UML specification for semantic objects modeling. *Problems in programming*. 2016. N 2–3. P. 211–219.
19. OMG Unified Modeling LanguageTM (OMG UML), Superstructure Version 2.4.1. Available from: https://www.omg.org/spec/UML/2.4.1/About-UML/.
20. Scientific bases and theoretical and methodological principles of creation of modern encyclopedias: collective monograph / Ed. Dr. hist. sciences, prof. Kirydon A.M. - K.: State Scientific Institution "Encyclopedic Publishing House". 2015. 160 p.
21. Sirin E., Parsia B., Cuenca Grau B., Kalyanpur A., Yarden K. Pellet: A practical OWL-DL reasoner. *Journal of Web Semantics*. 2007. T. 5, N 2. P. 51–53.
22. Shearer R, Motik B. Horrocks I. HermiT: A Highly-Efficient OWL Reasoner. *OWLED*. 2008. V. 432. P. 91.
23. Novitskyi O.V. Service-oriented distributed real-time systems in digital libraries (DL). *Problems in programming*. 2013. N 2. P. 87–94.

Одержано 02.03.2020

Про авторів:

Новицький Олександр Вадимович,
кандидат технічних наук,
науковий співробітник.
Кількість публікацій в українських виданнях – 31.
Кількість публікацій в зарубіжних виданнях – 5.
Індекс Хірша – 7,
http://orcid.org/0000-0002-9955-7882,

Резніченко Валерій Анатолійович,
кандидат фізико-математичних наук,
заст. зав. відділом.
Кількість публікацій в українських виданнях – 61.
Кількість публікацій в зарубіжних виданнях – 4.
Індекс Хірша – 12,
http://orcid.org/0000-0002-4451-8931.

Місце роботи авторів:

Інститут програмних систем НАН України,
03187, м. Київ-187, проспект Академіка Глушкова, 40.
Тел.: +38(044) 526-51-39.
E-mail: reznich@isofts.kiev.ua