DOI: 10.34185/1991-7848.itmm.2024.01.046

ПЕРСПЕКТИВИ РЕФАКТОРИНГУ ОНТОЛОГІЙ

Карповський Д.О., Шинкаренко В. І.

Український державний університет науки і технологій, Україна

Анотація. *Існуючі онтології доповнюються, розширюються в обсязі, підвищується їх складність. З часом виникають проблеми інтеграції нових знань у якусь онтологію та підтримки вже існуючих онтологій. Проблема посилюється тим, що онтології зазвичай наповнені знаннями з баз даних і анотованих документів на природній мові, що значно збільшує обсяг і складність онтологій. Представлена робота присвячена обробці та аналізу існуючих онтологій і застосуванню до них певних методів рефакторингу з метою вдосконалення онтології та покращення читабельності представлених знань для розробників і користувачів. Виконується розробка відповідного програмного забезпечення та метрик, які будуть об'єктивно засвідчувати ефективність застосування методів рефакторингу. У процесі дослідження було виявлено певний перелік шаблонів, які можна використовувати для рефакторингу онтологій, і реалізовано один з них – шаблон «Property Raising» – перенесення властивості в суперклас. В результаті тестування було встановлено, що після застосування цього шаблону обсяг вихідного файлу з даними онтології зменшився на 3-5%, за умови, що класи-нащадки містили спільні властивості, які були передані суперкласу.*

Ключові слова. *онтологія, рефакторінг, SPARQL, Ruby, патерни*

Вступ. На сьогодні онтології у різних формах використовуються в різних галузях діяльності людини. Вони постійно доповнюються, розширюються в об'ємах, збільшується їх складність. З плином часу з'являються проблеми інтеграції в деяку онтологію нових знань і підтримки вже існуючих онтологій.

Проблема поглиблюється тим, що онтології зазвичай наповнюються знаннями із баз даних та анотованих документів природною мовою, що ще значно збільшує обсяги та складність онтологій.

Представлена робота присвячена обробці та аналізу існуючих онтологій і застосуванню до них певних методів рефакторингу задля покращення онтології та поліпшення читабельності представлених знань для розробників та користувачів. Крім того розглядається питання розробки відповідного програмного забезпечення та розробки метрик, які будуть об'єктивно засвідчувати ефективність застосування методів рефакторингу.

Основна частина. У процесі дослідження було розглянуті певні патерни, які можуть бути використані задля рефакторингу онтологій, та реалізовано один із них – патерн «Підняття властивості», тобто перенесення властивості до суперкласу.

Оскільки основними елементами онтологій є концепти, екземпляри, поняття, атрибути і відношення, то всі маніпуляції задля покращення онтологій будуть застосовуватись до них.

Передбачаються дослідження ефективності використання патернів для первинного рефакторингу онтологій [1]:

«Доповнення класу», «Доповнення класу нащадка», «Доповнення суперкласу» - додавання властивостей до вибраного класу;

«Скорочення ієрархії» – за необхідності виконати об'єднання суперкласу та класу нащадка задля зменшення кількості зв'язків;

«Доповнення ієрархії» – за необхідності створити суперклас та його нащадків задля сепарації атрибутів, зв'язків;

«Видалення класу» – за необхідності видалити певний клас, а всі його атрибути перенести до класу, який посилається на даний клас;

«Трансфер властивості» – перенесення властивості із одного класу до іншого;

«Підняття властивості» – перенесення властивості до суперкласу;

«Опускання властивості» - перенесення властивості до класу нащадка;

«Накладання обмежень» - зміна напрямку та властивостей зв'язків між класами.

Наразі розробляється програмний додаток рефакторингу логічних моделей, у якому відбувається завантаження оригінальної онтології, аналіз даної моделі на консистентність, зв'язків між об'єктами на предмет пошуку спільних властивостей та виділення цих властивостей в клас-нащадок, або базовий клас.

Додаток реалізується за допомогою мови програмування Ruby та фреймворку Ruby on Rails. У ході аналізу даних онтології та виконання запитів використовується бібліотека (gem) "ruby-rdf/sparkle". SPARQL – це мова запитів

RDF [2], тобто мова семантичних запитів до баз даних, здатна отримувати та маніпулювати даними, що зберігаються в Resource Description Framework (RDF)) форматі[3, 4].

Також виконується пошук спільних властивостей в класах-нащадках. За допомогою sparkle запитів та можливостей мови програмування Ruby проводиться аналіз атрибутів класів-нащадків певного суперкласу, і якщо кожен із них має однакові атрибути, то даний атрибут переноситься до суперкласу.

Після проведення тестування на наборі онтологій, структура яких мала спільні атрибути в класах нащадках, було виявлено, що об’єм оригінального файлу онтології було скорочено на 3-5% в залежності від об’єму файлу та обов’язкової наявності спільних атрибутів в класах-нащадках.

Також у процесі розробки знаходиться функціональність, яка дозволяє людині, яка працює із онтологією, краще зрозуміти її структуру. Ця функціональність розробляється за допомогою бібліотеки “railroady”, створеній для фреймворку Ruby on Rails [5, 6]. Після завантаження онтології до системи, користувач матиме можливість отримати зображення із UML-діаграмою структури онтології та зв’язками між її об’єктами.

Висновки. Наступним етапом даної роботи є програмна реалізація згаданих патернів рефакторингу та аналіз опрацьованих онтологій, розробка метрик, які будуть об’єктивно показувати ступінь покращення внаслідок використання патернів . В процесі аналізу можуть бути оцінені таки параметри як швидкість опрацювання запитів, доступність додавання нових елементів до онтології, можливість онтології до адаптації до певної бізнес моделі та т.і. [7].

Дана розробка допоможе підтримувати і доповнювати існуючі онтології, а також розробляти нові, застосовуючи методи рефакторінгу, щоб онтології можливо було підтримувати і доповнювати в майбутньому врівноважуючи обсяги і складність.

ЛІТЕРАТУРА / REFERENCE

1. Gruber, T. R. (1993). A Translation Approach to Portable Ontology Specifications. Knowledge Acquisition, 5(2), 199-220.

2. Fernández-López, M., Gómez-Pérez, A., & Juristo, N. (1997). METHONTOLOGY: From Ontological Art Towards Ontological Engineering. Proceedings of the AAAI97 Spring Symposium Series on Ontological Engineering.
3. Noy, N. F., & McGuinness, D. L. (2001). Ontology Development 101: A Guide to Creating Your First Ontology. Stanford Knowledge Systems Laboratory Technical Report KSL-01-05.
4. Uschold, M., & Gruninger, M. (1996). Ontologies: Principles, Methods and Applications. The Knowledge Engineering Review, 11(2), 93-155.
5. Bechhofer, S., van Harmelen, F., Hendler, J., Horrocks, I., McGuinness, D. L., Patel-Schneider, P. F., & Stein, L. A. (2004). OWL Web Ontology Language Reference. W3C Recommendation.
6. Horridge, M., & Bechhofer, S. (2011). The OWL API: A Java API for OWL Ontologies. Semantic Web, 2(1), 11-21.
7. Staab, S., & Studer, R. (Eds.). (2004). Handbook on Ontologies. Springer

PERSPECTIVES OF ONTOLOGY REFACTORING

Dmytro Karpovskyi, Viktor Shynkarenko

Abstract. *Existing ontologies are supplemented and expanded in scope, increasing their complexity. Over time, integration issues arise with adding new knowledge to an ontology and maintaining existing ones. The problem is exacerbated by the fact that ontologies typically incorporate knowledge from databases and annotated documents in natural language, significantly increasing the volume and complexity of ontologies. This paper focuses on processing and analyzing existing ontologies and applying specific refactoring methods to enhance ontology and improve the readability of presented knowledge for developers and users. The development of corresponding software and metrics is carried out to objectively demonstrate the effectiveness of refactoring methods. During the research process, a certain list of patterns was identified that can be used for ontology refactoring, and one of them was implemented – the "Property Raising" pattern – transferring a property to a superclass. As a result of testing, it was found that after applying this pattern, the volume of the original ontology data file decreased by 3-5%, provided that the subclass contained common properties that were passed to the superclass.*

Keywords: *ontology, refactoring, SPARQL, Ruby, patterns*.