УДК 004.82
DOI: 10.32820/2074-8922-2019-65-135-146

ПЕРЕТВОРЕННЯ ДАНИХ ІЗ ФОРМАТУ ДАНИХ МОВИ OWL ДО XML ТОДОС ІЗ ЗБЕРЕЖЕННЯМ ОСНОВНОГО ЛОГІЧНО-СТРУКТУРНОГО ЗМІСТУ ІНФОРМАЦІЇ

©Гуралюк А. Г., Ростока М. Л.
«Мала академія наук України» МОН України і НАН України

Інформація про авторів:

Гуралюк Андрій Георгійович: ORCID: 0000-0002-7497-5746; ss_variant@bigmir.net, кандидат педагогічних наук, старший науковий співробітник відділу створення та використання інтелектуальних мережних інструментів Національного центру «Мала академія наук України» МОН України і НАН України, вул. Дегтярівська, 38/44, м. Київ, 04119, Україна.

Ростока Марина Львівна: ORCID: 0000-0002-1891-5482; marilvross@gmail.com, кандидат педагогічних наук (доктор філософії), старший науковий співробітник відділу інформаційно-дидактичного моделювання Національного центру «Мала академія наук України» МОН України і НАН України, вул. Дегтярівська, 38/44, м. Київ, 04119, Україна.

З появою Semantic Web та семантичних технологій виникли онтології, які стали однією з найвизначніших парадигм представлення знань. Основою подальшого розвитку Semantic Web на сьогодні вважається онтологізація представлених у ній знань. Онтологія – це спроба всеосяжної і детальної формалізації деякої області знань за допомогою концептуальної схеми. Зазвичай така схема складається зі структури даних, що містить всі релевантні класи об'єктів, їх зв'язку і правила (теореми, обмеження), прийняті в цій галузі.

Однією із найбільш часто використовуваних мов описання веб-онтологій у світовій практиці є мова OWL (Web Ontology Language) призначена для надання засобів, які можна використовувати для опису класів і відносин між ними, що притаманні веб-документам і додаткам.

Проте, існує ряд задач для яких використання мови OWL є недоцільним. Для розв’язання частини таких задач була створена вітчизняна розробка ТОДОС (Трансдисциплінарні Онтологічні Діалоги Об'єктноорієнтованих Систем), що використовує мову, створену на основі стандарту представлення даних XML. Актуальність статті обумовлена недостатньою розробленістю взаємозв’язку між різними програмними засобами призначеними для створення, описання, візуалізації та редагування онтологій. А саме, відсутністю програмних додатків, що забезпечували б конвертацію даних із формату OWL до формату системи ТОДОС та у зворотному напрямі.

У запропонованій статті описується метод двостороннього перетворення даних між форматами мови OWL та системи ТОДОС. Проводиться аналіз базових конструкцій мов OWL та XML (ТОДОС). Розглядаються питання функціональної еквівалентності конвертованої і вихідної систем. Визначаються проблеми із збереженням основного логічно-структурного змісту інформації, що пов’язані із різною потужністю мов тощо. Пропонується авторська розробка (програмний додаток мовою java) для реалізації конвертації згаданих форматів даних.

У якості результату дослідження, схематично викладеного у запропонованій статті, робиться висновок, що неможливо встановити повну взаємовідповідність між мовами OWL та XML (ТОДОС). Не зважаючи на те, що OWL є більш потужною семантичною системою, за деякими можливостями візуалізації XML ТОДОС перевищує OWL, і відповідно функція еквівалентності перетворення базових конструкцій заданих мов не володіє властивістю транзитивності. Проте, із певними допущеннями можна створити модель часткової конвертації, де втрати даних були б мінімальними.

Ключові слова: OWL, XML, ТОДОС, RDFS, конвертор, перетворення форматів, онтології, семантика.

Гуралюк А. Г., Ростока М. Л. **«**Преобразования данных из формата данных языка OWL к XML ТОДОС с сохранением основного логически-структурного содержания информации».

С появлением Semantic Web и семантических технологий возникли онтологии, которые стали одной из самых выдающихся парадигм представления знаний. Основой дальнейшего развития Semantic Web сегодня считается онтологизация представленных в ней знаний. Онтология – это попытка всеобъемлющей и детальной формализации некоторой области знаний с помощью концептуальной схемы. Обычно такая схема состоит из структуры данных, содержащей все релевантные классы объектов, их связи и правила (теоремы, ограничения), принятые в этой области.

Одним из наиболее часто используемых языков описания веб-онтологий в мировой практике является язык OWL (Web Ontology Language) предназначенный для предоставления средств, которые можно использовать для описания классов и отношений между ними, присущих веб-документам и приложениям.

Однако, существует ряд задач, для которых использование языка OWL нецелесообразно. Для решения части таких задач была создана отечественная разработка ТОДОС (Трансдисциплинарные Онтологические Диалоги Объектноориентированных Систем), использующая язык, созданный на основе стандарта представления данных XML.

Актуальность статьи обусловлена недостаточной разработанностью взаимосвязи между различными программными средствами предназначенные для создания, описания, визуализации и редактирования онтологий. А именно, отсутствием приложений, которые обеспечивали бы конвертацию данных из формата OWL к формату системы ТОДОС и обратно.

В предлагаемой статье описывается метод двустороннего преобразования данных между форматами языка OWL и системы ТОДОС. Проводится анализ базовых конструкций языков OWL и XML (ТОДОС). Рассматриваются вопросы функциональной эквивалентности, конвертируемой и исходной систем. Определяются проблемы с сохранением основного логически-структурного содержания информации, связанные с разной мощностью языков и т.п. Предлагается авторская разработка (программное приложение языке java) для реализации конвертации упомянутых форматов данных.

В качестве результата исследования, схематично изложенного в предлагаемой статье, делается вывод, что невозможно установить полное взаимное соответствие между языками OWL и XML (ТОДОС). Несмотря на то, что OWL является более мощной семантической системой, по некоторым возможностями визуализации XML ТОДОС превышает OWL, и соответственно функция эквивалентности преобразования базовых конструкций, заданных как не обладает свойством транзитивности. Однако, с определенными допущениями можно создать модель частичной конвертации, где потери данных были бы минимальными.

Ключевые слова: OWL, XML, ТОДОС, RDFS, конвертор, преобразование форматов, онтологии, семантика.

A.Huraliuk, M.Rostoka "Data conversion from the OWL data format to the XML TODOS format while saving the logical and structural information content.*"*

With the introduction of the Semantic Web and semantic technologies, ontologies have emerged and become one of the most prominent paradigms for representing knowledge. The ontologization of the knowledge presented in it is considered the basis for the further development of the Semantic Web. Ontology is an attempt of comprehensive and detailed formalization of a certain field of knowledge using a conceptual scheme. Typically, such a scheme consists of a data structure containing all relevant classes of objects, their relationships and regulations (theorems, restrictions) adopted in this area.

One of the most commonly used languages for describing web ontologies in the world practice is OWL (Web Ontology Language), which is designed to provide tools that can be used to describe classes and the relationships between them that are common to web documents and applications.

However, there are a number of tasks for which the use of the OWL language is impractical. The domestic development of TODOS (Transdisciplinary Ontological Dialogues of Object-Oriented Systems) was created to solve some of these problems using a language created on the basis of the XML data presentation standard.

The relevance of the article is conditioned by the insufficient development of the relationship between various software tools designed to create, describe, visualize and edit ontologies; namely by the absence of applications that would ensure the conversion of data from the OWL format to the TODOS system format and vice versa.

This article describes a method for two-way data conversion between the OWL and the TODOS system language formats. The analysis of the basic constructions of the OWL and XML (TODOS) languages is carried out. The issues of functional equivalence of the convertible and original systems are considered. The problems are identified with the preservation of the main logical and structural content of information associated with different power languages, etc. Authoring is proposed (a java language software application) for implementing the conversion of the mentioned data formats.

As a result of the research, outlined schematically in the present article, it is concluded that it is impossible to establish complete mutual correspondence between the OWL and XML (TODOS) languages.

Despite the fact that OWL is a more powerful semantic system, according to some of the visualization capabilities of XML, TODOS exceeds OWL, and, accordingly, the equivalence function of the transformation of basic constructions of specified languages does not have the transitivity property. However, with certain assumptions, a partial conversion model can be created where data loss would be minimal.

Keywords: OWL, XML, TODOS, RDFS, converter, format conversion, ontology, semantics.

Актуальність. З появою семантичної мережі та семантичних технологій виникли онтології, які стали однією з найвизначніших парадигм представлення знань [12]. Онтологія – це термін, запозичений з філософії, що відноситься до науки опису видів сутностей різноманітних об'єктів та зв'язків між ними. Використання комп'ютерних онтологій розглянуто в працях Т. Ґрубера (T.Gruber), Т. Джефрі (T. Jeffrey), Ю. Дінґа (Y. Ding), О.Г. Євсеєвої, В.А. Лапшина, В.В. Литвина, В.В. Любченка, Л.В. Найханова, О.С. Нарiньяні, С. Ніренбурґа (S. Nirenburg), Н. Ноя (N. Noy), М.А. Попової, А.В. Смірнова, О.Є.Стрижака, С.О. Субботіна, Й. Зуре (Y. Sure), Д. Фора (D. Faure) та багатьох інших.

Однією із найбільш часто використовуваних мов описання веб-онтологій у світовій практиці є мова OWL (Web Ontology Language) призначена для надання засобів, які можна використовувати для опису класів і відносин між ними, що притаманні веб-документам і додаткам [5,6,11].

Вітчизняною розробкою є система ТОДОС – практикоорієнтований засіб опису веб-онтологій, призначений для обробки контенту мережевих інформаційних ресурсів. Ця система використовує власну мову, описану форматом XML [7,8].

Існує ряд задач, які потребують взаємодії між цими двома мовами. Для забезпечення такої взаємодії на основі визначених вимог була побудована інформаційна модель та здійснена практична реалізація програми двосторонньої конвертації між мовами OWL та XML (ТОДОС).

Виклад основного матеріалу. Основою подальшого розвитку Semantic Web на сьогодні вважається онтологізація представлених у ній знань. Онтологія – це спроба всеосяжної і детальної формалізації деякої області знань за допомогою концептуальної схеми. Зазвичай така схема складається зі структури даних, що містить всі релевантні класи об'єктів, їх зв'язку і правила (теореми, обмеження), прийняті в цій галузі.

Онтологічні моделі відображають концептуальний погляд дослідника на деяку предметну область і дають можливість однозначно визначати її поняття, структурувати, накопичувати і неодноразово використовувати знання. Потреба такого способу представлення знань постійно відчувалася в багатьох галузях науки і, як відповідь на актуальну потребу, онтологічний підхід став використовуватися практично одночасно в декількох сферах людської діяльності. Сьогодні можна говорити про формування наукового напрямку, що займається дослідженням онтологій знань, розробкою як теоретичної бази, так і практичної технології застосування онтологій в житті людини.

OWL-онтологію можна розглядати як сукупність тверджень дескрипційної логіки, що задають обмеження на множині інтерпретацій класів і відносин.

В основу OWL покладено мову RDFS (RDF Schema). Виразних можливостей RDFS досить для опису класів і властивостей RDF-ресурсів, а також ієрархій на них. Синтаксис RDF / XML гарантує підтримку обміну онтологіями між гетерогенними системами, які можуть навіть не підтримувати саму мову OWL як такої (чисті RDF / RDFS системи) – серіалізація в даний синтаксис визначає правила приведення елементів мови OWL до абстрактних конструкцій RDF, записаним у вигляді ієрархії XML тегів. Проте, OWL надає додаткові можливості: визначення класу через обмеження на його властивості, через теоретико-множинні операції над класами, визначення характеристик (симетричність, транзитивність і інших) і кардинальних чисел властивостей [1].

Класи використовуються для моделювання понять (концептів). Під класом ми розуміємо абстрактний механізм для групування ресурсів зі схожими характеристиками. Кожен OWL-клас асоціюється з множиною екземплярів (індивідів), які називаються екстенсіоналом класу. У той же час, клас має власний особливий сенс (що лежить в основі поняття), який пов'язаний, але не дорівнює екстенсіоналу класу. Два класи можуть мати один і той же екстенсіонал, але при цьому залишатися різними класами. Між класами

можна встановлювати ієрархічні відносини типу «загальне-часткове».

Відповідно семантиці OWL, це відношення має теоретико-множинне трактування: екстенсіонал підкласу повністю входить в екстенсіонал надкласу.

Класи складаються з екземплярів. Що моделювати за допомогою класів, а що за допомогою екземплярів – важливе питання онтологічного моделювання.

Властивості служать для моделювання атрибутивної інформації, що характеризує членів класів, а також для моделювання відносин між ними. У термінології OWL властивості, що моделюють атрибутивну інформацію, називаються властивостями-значеннями (datatype properties), а ті, що моделюють відносини – об'єктними властивостями (object properties).

Властивість є бінарним відношенням. Властивості одночасно використовуються і для декларації узагальнених тверджень про члени класів, і для специфікації тверджень про конкретні екземпляри. Одним із суттєвих недоліків OWL є відсутність можливості природним чином визначати властивості у властивостей. Це не дозволяє моделювати атрибути у предметних відносин, n-арні відносини і атрибути у атрибутів.

Існує два шляхи вирішення цієї проблеми. Один з них – вийти за рамки OWL і вдатися до механізму реіфікації (від англ. Reify – матеріалізовувати). Реіфікації є засобом RDF [14].

Її суть полягає в тому, щоб розглядати RDF-затвердження як окремі об'єкти. У RDF твердженням вважається триплет виду <об'єкт, предикат, суб'єкт>. Даний триплет може бути визначений як екземпляр класу.

Альтернативний шлях – декларувати відносини як класи, а не як властивості [15]. Автори цього рішення критикують використання реіфікації для визначення властивостей у властивостей, наводячи такі аргументи. «У n-місцевих відносинах додаткові аргументи, зазвичай, характеризують не саме твердження (statement), а екземпляр відносини». Тому було б більш природним оперувати безпосередньо екземплярами відносин, ніж твердженнями про ці екземпляри.

Суть другого підходу в наступному. Ставлення визначається як клас, а екземпляри цього класу будуть виступати в ролі екземплярів відносини. У свою чергу, аргументи цього відносини зв'язуються з ним за допомогою механізму властивостей і їх кількість вже не обмежена.

Для того, щоб використовувати будь-яку OWL-онтологію, в програмних розробках або дослідженнях слід чітко уявляти інтерпретації її класів і відносин. В іншому випадку будь-яка її модифікація може призвести до появи суперечливих аксіом і, як наслідок, до можливості отримання невірних результатів її логічного аналізу, а також до помилок або збоїв в роботі заснованих на ній програм. У свою чергу, проблема розуміння онтології (Ontology Comprehension), як це зазначено в роботі [13], є на даний момент досить новою і потребує подальшого дослідження.

RDF і OWL – мови представлення інформації (знань), які можуть використовуватися для опису онтологій як в семантичній павутині, так і в різних прикладних інформаційних системах. RDFS є надбудовою над RDF і визначає його базові конструкції (ресурс, клас, підклас, тип даних, домен і т.п.). Якщо проводити аналогію з базами даних, то RDFS дозволяє задати структуру БД, а RDF наповнити її вмістом. OWL повністю включає в себе RDF і розширює його можливості. У прикладних OWL-онтологіях значна частина опису виконана за допомогою конструкцій RDF і RDFS [19].

Базовим елементом мови RDF є трійка (триплет, аксіома), яка складається з суб'єкта, предиката і об'єкта (рис. 1).

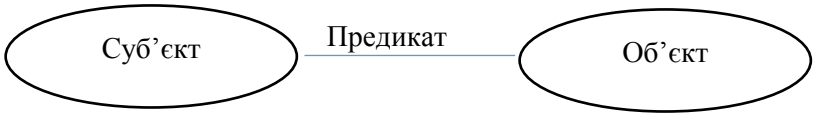

Рис.1. RDF-трійка «суб'єкт – предикат – об'єкт»

Предикат (властивість) являє собою бінарне відношення між суб'єктом і об'єктом. Залежно від контексту одна і та ж сутність онтології може виступати в будь-якому з перерахованих якостей. Тобто в одній трійці сутність може бути суб'єктом, в іншій – предикатом, а в третій – об'єктом.

Перші два елементи RDF-трійки (суб'єкт і предикат) обов'язково ідентифікуються за допомогою URI. Об'єкт може бути як сутністю,

ідентифікованої за допомогою URI, так і RDF-літералом (рядком, числом, датою і т.п.).

У свою чергу, мова XML ТОДОС має дещо обмежений набір можливостей порівняно з OWL, тому повноцінна конвертація неможлива. Проте можна виконати часткову конвертацію із збереженням зворотної конвертації шляхом встановлення умовної еквівалентності окремих об'єктів і збереження надлишкових даних у технічній вершині (для ТОДОС) або у коментарях (для Protégé). У системі ТОДОС файл починається тегом <Graph>, котрий вказує на те, що даний файл описує онтологічну структуру [2].

У кожному файлі обов'язково присутні два блоки: блок, обмежений тегом <Nodes></Nodes>, у якому розташовані вершини (вузли) онтології (онтографа), та блок, обмежений тегами <Edges></Edges> – тут знаходиться описання зв'язків між різними вузлами.

Блок <Nodes> містить опис вершин, обмежених тегами <Node></Node>. Кожен вузол має ряд атрибутів: унікальний ідентифікаційний код guid, ім'я (атрибут nodeName), а також ряд параметрів, відповідних за відображення вершини в різних редакторах. Атрибут nodeName для онтологій, створених за допомогою системи ТОДОС є утворюючим, тобто зв'язки між вузлами однозначно задаються за допомогою вузлів цих вершин. Таким чином, для запобігання невизначеностей всі вузли мають унікальне ім'я. Кожному із вузлів може бути поставлене у відповідність певне контекстне наповнення, яке описується всередині тегів <data> </data>.

Блок <Edges> містить теги <Edge>, що описують зв'язки між вершинами. Атрибут node1 відповідає дочірньому, а node2 – батьківському вузлу.

У форматі ТОДОС відсутня різниця між класом і екземпляром.

Перетворення даних із формату даних мови OWL до XML ТОДОС зі збереженням основного логічно-структурного змісту інформації потребує встановлення функціональної еквівалентності конвертованої і вихідної систем, тобто формалізації відповідності елементів мов OWL та XML ТОДОС. Кожна сутність (ресурс), описувана в онтології, повинна мати унікальний ідентифікатор. Для цього використовуються URI або IRI. Відмінність останнього полягає в можливості використання символів національних алфавітів при вказані імені ресурсу за рахунок підтримки Unicode.

Моделювання конвертеру між OWL та XML ТОДОС ускладнено тим, що не існує повної взаємовідповідності між цими мовами.

Не зважаючи на те, що OWL є більш потужною семантичною системою, за деякими можливостями візуалізації XML ТОДОС перевищує OWL, і відповідно функція еквівалентності перетворення базових конструкцій заданих мов не володіє властивістю транзитивності. Проте, із певними допущеннями можна створити модель часткової конвертації, де втрати даних були б мінімальними

2. Модель функціональної еквівалентності структур мов OWL та XML ТОДОС визначає, що основними об'єктами для ТОДОС є node (вершина, вузол) і edge (зв'язок). У відповідність вузлу можна поставити такі елементи OWL як клас і екземпляр. Клас може містити індивіди (екземпляри класу- individuals). Індивіди визначаються за допомогою аксіом індивідів (фактів), яких може бути два види:

1) факти членства індивідів в класах;

2) факти ідентичності / відмінності індивідів.

Модель функціональної еквівалентності структур мов OWL та XML ТОДОС визначає, що основними об'єктами для ТОДОС є node (вершина, вузол) і edge (зв'язок).

У відповідність вузлу можна поставити такі елементи OWL як клас і екземпляр

OWL-клас може бути описаний шістьма способами:

1) ідентифікатором класу (URI);

2) перерахуванням всіх екземплярів класу;

3) обмеженням на значення властивості;

4) перетином 2-х і більше визначень класів;

5) об'єднанням 2-х і більше визначень класів;

6) доповненням (логічним запереченням) визначення класу.

Клас може містити індивіди (екземпляри класу- individuals). Індивіди визначаються за допомогою аксіом індивідів (фактів), яких може бути два види

1) факти членства індивідів в класах;

2) факти ідентичності / відмінності індивідів.

Між мовами OWL та XML ТОДОС можлива лише часткова конвертація, доцільна в деяких прикладних задачах, оскільки вони хоч і призначені для описання онтологій, проте мають достатньо багато принципових відмінностей.

Таблиця 1.

Деякі відповідності між OWL та XML ТОДОС

<table>
<tr><td colspan="3">Заголовок (простір Імен)</td></tr>
<tr><td colspan="2"><?xml version="1.0"?>
<rdf:RDF
xmlns ="URI"
……..
<owl:Ontology rdf:about="URI">
…….
</owl:Ontology></td><td>Graph
(nodesize, vspacing, hspacing, padding, guid, searchurl)</td></tr>
<tr><td>Коментар</td><td>Розміщуємо в ТОДОС за допомогою технічної вершини
URI можуть співпадати або бути відсутніми; prefix – різні префікси, в залежності від потреб
При імпорті з ТОДОС, у разі відсутності запису в технічній вершині використовуємо «Стандартні простори імен»</td><td>При першій конвертації до ТОДОС автоматично створюється, властивості встановлюємо «за замовчуванням»: наприклад 10, 3, 20, 4, ********-****-****-****-****-************,
https://www.google.com.ua/search?q=###SEARCH###"
* – випадкове 16-е число
При конвертації до owl заносимо в коментар</td></tr>
<tr><td colspan="3">Вузол (вершина)</td></tr>
<tr><td colspan="2">Головна вершина:
Thing – універсум</td><td>Node
(guid, nodeName, nclass, shape, color, xPos, yPos, font, fontsize)</td></tr>
<tr><td>Коментар</td><td>Кожен індивід в OWL є членом класу owl: Thing. Явно не задається, присутня «за замовчуванням»
Якщо онтограф ТОДОС має єдину вершину, то її дані можуть бути збережені в коментарях до Thing</td><td>Відсутня як універсум, не обов’язкова. як правило є в якості звичайної вершини – назви онтології.
При конвертації з OWL: nodeName= " Thing "</td></tr>
<tr><td colspan="2"><i>Клас:</i>
<owl:Class rdf:about="URI# Ім’я ">
……..
</owl:Class></td><td><i>Node</i>
(guid, nodeName, nclass, shape, color, xPos, yPos, font, fontsize)
nodeName= " Ім’я"</td></tr>
<tr><td>Коментар</td><td colspan="2">Задання класу ідентифікатором</td></tr>
<tr><td colspan="2"><i>Individuals:</i>
<owl:Thing rdf:about="URI# Ім’я ">
<rdf:type rdf:resource="URI(3)#NamedIndividual"/>
<rdf:type rdf:resource="URI#Клас"/>
</owl:Thing></td><td><i>Node</i>
(guid, nodeName, nclass, shape, color, xPos, yPos, font, fontsize)
nodeName= " Ім’я"
nclass= "Клас"</td></tr>
<tr><td>Коментар</td><td></td><td>Можливі значення «за замовчуванням»:
shape="square" color="13421772" font="Verdana" fontsize="14" xPos="" yPos="" – вираховувати недоцільно, автоматично в едіторі</td></tr>
<tr><td colspan="3">Зв’язок (дуга)</td></tr>
<tr><td colspan="2"><SubClassOf>
<Class abbreviatedIRI="URI#Parent"/>
<Class abbreviatedIRI="URI#Child"/>
</SubClassOf></td><td>Edge
(guid, edgeName, node1, node2, group="" istwoway="")</td></tr>
</table>

Більш детально зупинимось на типах зв’язків між об’єктами. Система ТОДОС дозволяє візуалізувати зв’язки за допомогою ребер онтографу, виокремивши їх різні типи за допомогою кольорів. Для зручності задамо такі типи у таблиці 2.

Таблиця 2.

Типи зв'язків

№	Назва зв'язку	Колір для ТОДОС	Приклад OWL
1)	Subclass (default)		<SubClassOf> <Class abbreviatedIRI="Ontology:untitled-ontology-104#Doctor"/> <Class abbreviatedIRI="Ontology:untitled-ontology-104#Person"/> </SubClassOf>
2)	Individual		<ClassAssertion> <Class abbreviatedIRI="Ontology:untitled-ontology-104#Patient"/> <NamedIndividual IRI="#Patient1"/> </ClassAssertion>
3)	Domain-Range		<ObjectPropertyDomain> <ObjectProperty IRI="#hasBloodSugar"/> <Class abbreviatedIRI="Ontology:untitled-ontology-104#Patient"/> </ObjectPropertyDomain>
4)	Property		<ObjectPropertyAssertion> <ObjectProperty IRI="#hasBloodSugar"/> <NamedIndividual IRI="#Patient1"/> <NamedIndividual IRI="#BloodSugar001"/> </ObjectPropertyAssertion>

Отже, можна відзначити, що конвертація без втрат даних неможлива, проте в межах деяких задач вона є доцільною.

Для розробки конвертеру була обрана мова програмування Java. Вибір Java у середовищі Eclipse зумовлено тим, що інструменти для програмування на Java є безкоштовними та мають широке розповсюдження. Java володіє великою множиною стандартних бібліотек і класів та детальною документацією, що полегшує написання найпростіших, але широко використовуваних методів. Саме такий підхід дозволяє писати код, слідуючи найкращим практикам програмування.

Ще однією із особливостей Java є повна кроссплатформенність з підтримкою як Windows, так і MacOS, Linux і багатьох інших операційних систем.

Для створення додатку для конвертації мов OWL та XML (ТОДОС) було використано API OWL – це Java API для створення, маніпулювання та серіалізації OWL онтологій [4] , яке постійно оновлюється та автоматично інтегрується до середовища Eclipse.

Для створення конвертору був використаний фреймворк Maven. В кореневому каталозі знаходиться файл pom.xml. Це файл опису проекту, на основі якого працює Maven. Він написаний на мові POM, із сімейства XML.

У верхній частині додатку розташована панель із назвою конвертера, а під нею знаходяться кнопки, натискання на які забезпечує виклик основних модулів програми. Кнопка «Завантажити» викликає діалог вибору завантаження файла-джерела із онтологією. Користувач може вибрати файл у форматі OWL або у форматі XML.

У залежності від формату файлу джерела можна здійснити конвертування, натисканням відповідної кнопки «Конвертувати до XML» чи «Конвертувати до OWL», після чого результуючий файл відобразиться у панелі «Результат», яка знаходиться у правій частині додатку (рис. 2).

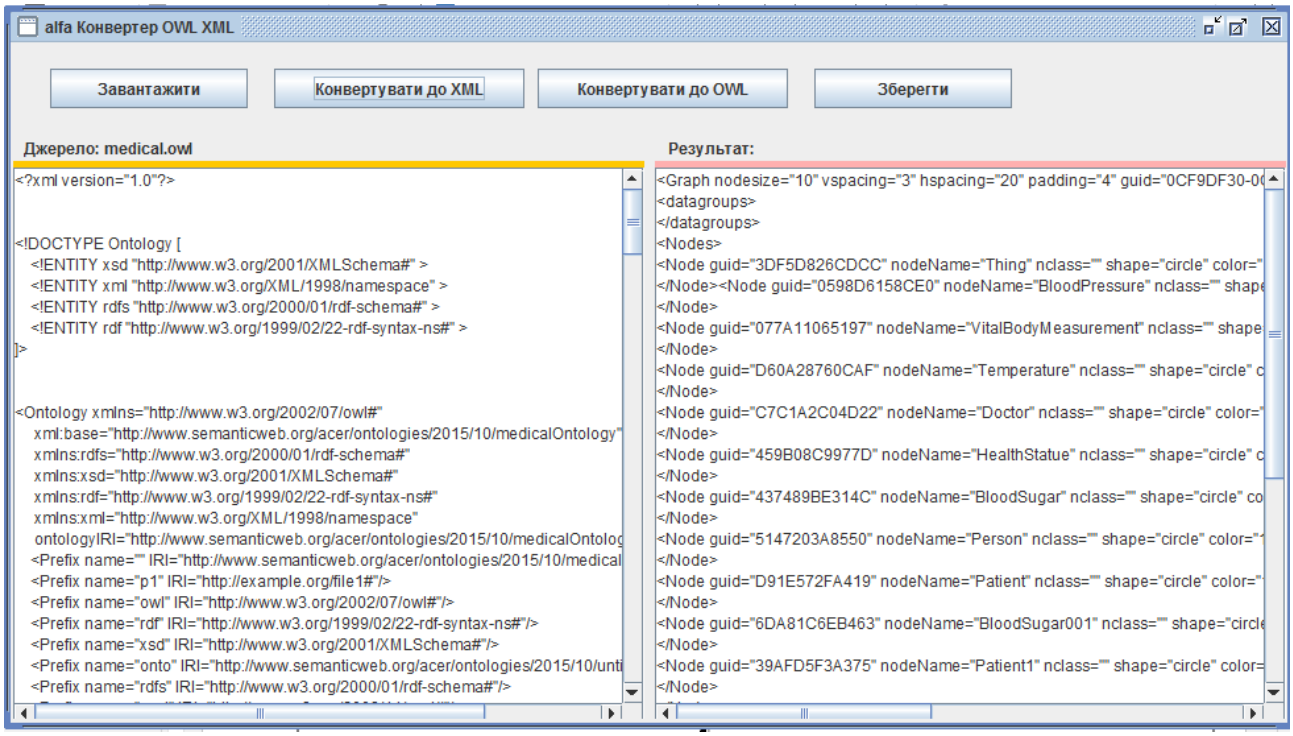

Рис. 2. Відображення файла-результату для конвертації до XML

Формат даних OWL, хоч у цілому має вищу потужність, ніж XML ТОДОС, проте існують деякі дані, що присутні в XML і не мають аналогів для OWL онтології. Наш конвертер такі дані генерує автоматично, виходячи із усередненого розуміння їх значень, або залишає незаповненими.

Так заголовок файлу у форматі XML ТОДОС містить наступний тег: <Graph nodesize="10" vspacing="3" hspacing="20" padding="4" guid="…" searchurl="https://www.google.com.ua/search?q=###SEARCH###">

Параметри vspacing="3", hspacing="20", padding="4" відповідають за взаємне розташування вершин при побудові онтографа у системі ТОДОС. Вони автоматично генеруються виключно із ергономічних міркувань.

Ідентифікатором об’єкта є параметр guid="…", так само як і параметри guid для кожного із об’єктів онтології є псевдовипадковим числом, частина якого формується за допомогою генератора випадкових чисел і з деякою ненульовою імовірністю (порядку 2,3·10-10) може повторитись , а частина генерується в залежності від поточного системного часу і в межах одного додатку є унікальною.

Опишемо більш детально алгоритм конвертації онтології із формату OWL.

1) Необхідно створити машинне представлення онтології – задати менеджер OWL онтології. Існують різні синтаксиси відображення онтологій у межах OWL/XML.

Використання OWL API дозволяє отримати онтологію з 16 типів таких синтаксисів. Серед них Functional, Manchester, Latex, Turtle, RDF/XML, OBO, DL, KRSS та інші, а також їх комбінацій.

```
OWLOntologyManager man = OWLManager.createOWLOntologyManager();
```

Веб-онтологія у форматі OWL завжди має вершину «Thing», яка є своєрідним універсумом, тобто суперкласом для всіх об’єктів, що містяться в онтології. В системі ТОДОС така вершина за замовчуванням відсутня, тому є сенс її штучного створення для імпортованих онтологій. Враховуючи, що ТОДОС не передбачає наявності коментарів у тексті, можна необхідні коментарі представити контентом цієї вершини. Також її контент можна подати у вигляді блоків, що відповідатимуть за збереження вихідних даних тим самим ми приберемо одну з ключових відмінностей – а саме, уникнемо можливої незв'язності графа.

```
OWLAPIFirst.jTextArea2.append("<Node guid=\""+setGUID() + "\" nodeName=\"" + "Thing" +
"\" nclass=\"" + "\" shape=\""+ "circle"+ "\" color=\""+ "13421772"+ "\" xPos=\""+ "20"+ "\" yPos=\""+
"20"+ "\" font=\""+ "Verdana"+ "\" fontsize=\"" + "14"+ "\">\n</Node>" );
```

2) Отримання IRI онтології. Онтологія у форматі ТОДОС не містить ідентифікатора IRI, тому це значення зберігається у контенті вершини «Thing» на випадок зворотного перетворення.

```
IRI documentIRI = man.getOntologyDocumentIRI(ont);
```

3) Наступний крок – отримання масиву класів

```
Set <OWLClass> classes = ont.getClassesInSignature();
……
String sClasses = classes.toString().substring(1,classes.toString().length()-1);
String[] myOWLClass = sClasses.split(", ");
```

4) Створення вершин (Nodes) на основі списку класів та запис їх до файлу-результату;

5) Визначення дочірності класів та встановлення відповідних зв'язків;

6) Встановлення класів без батьківських зв'язків і задання їм у якості батьківського об'єкту Thing;

7) Визначення відповідних класу індивідуумів та встановлення зв'язків із батьківськими класами;

8) Запис зв’язків до ТОДОС та задання їх типів.

Зворотне перетворення, тобто переконвертація із формату ТОДОС до файлів OWL дещо відрізняється за ідеологією. В межах свого дослідження я не ставила за мету забезпечити конвертацію із формату XML до будь якого синтаксису. Тому був обраний мінімально необхідний для описання онтології синтаксис RDF/XML.

Модуль конвертації з XML ТОДОС до OWL має низку особливостей.

1) Необхідність врахування мовних особливостей. Мова OWL у якості ідентифікаторів використовує URI (IRI), які не підтримують кирилицю. Тому було створено toTranslit, яка перетворює кирилічні символи (здійснює транслітерацію) в латиницю або у сполучення латинських символів (наприклад, відбувається транслітерація "щ" до "sch").

2) Доцільно виявилось уявити онтологію в форматі XML як «Document Object Model», коротко DOM), а параметри представити як HTML-теги.

Відповідно до об'єктної моделі документа, кожен HTML-тег є об'єктом. Вкладені теги використовуються «дітьми» батьківських елементів. Текст, який знаходиться всередині тегу, також є об'єктом.

Таким чином для зчитування з формату XML необхідно реалізувати простий DOM-парсер для представлення онтології у вигляді дерева DOM-документу.

```
// Створюємо DOM-парсер
DocumentBuilder documentBuilder =
DocumentBuilderFactory.newInstance().newDocumentBuilder();
// Створюємо дерево DOM документу
Document document = documentBuilder.parse(new
ByteArrayInputStream(xml.getBytes(Charset.forName( "UTF-8" ))));
```

Декларування вершин залежить від визначення наперед заданих в XML ТОДОС груп. При конвертації з OWL декларувались зв’язки 4-х типів (Таблиця 2). При зворотній конвертації відбувається їх врахування при створенні відношень між класами та індивідами.

У випадку, коли таке декларування не задане, екстенсіонал класу вважається множиною сабкласів (це пов’язано із тим, що система ТОДОС не розрізняє поняття класу і індивіду і, фактично, всі вузли онтографу в системі ТОДОС є класами).

Таким чином використання розбиття вихідного XML-документу на теги, побудова ієрархії цих тегів вже як DOM-документу та запис їх в синтаксисі RDF/XML забезпечує приведення елементів мови OWL до абстрактних конструкцій RDF, записаним у вигляді ієрархії XML тегів, тобто будується веб –онтологія мовою OWL.

Для унаочнення представлення онтологій у світовій практиці використовуються онтологічні графи (онтографи).

Представлення інформації у вигляді онтографу дозволяє вивчати не лише окремий термін (поняття), але й отримувати всі його семантичні зв'язки з іншими поняттями, тим самим осмислюючи його роль у даній системі знань чи в ході розв'язання задачі.

Однак мережевий онтограф може виступати не лише засобом організації знань. Розширюючи його традиційні функції, онтограф можна перетворити на середовище, в якому забезпечується активна робота зі знаннями, а також оригінальним чином вирішуються навчальні завдання.

У процесі виділення ієрархій між термінами-поняттями виконується локалізація таксономічних невизначеностей на рівні входу й виходу певних вершин онтографу, що визначають неможливість безпосередньо визначити певне відношення між об'єктами чи процесами. Розгляд цих вершин, як таксономічних невизначеностей, забезпечує синхронізацію смислових сутностей, що, в свою чергу, дозволяє оптимізувати умови переходу й дозначити класифікацію об'єктів у онтологічної моделі [3]

На рис. 3 і 4 відображення онтології «medical» до та після конвертації у вигляді графа із використанням відповідних формату засобів візуалізації.

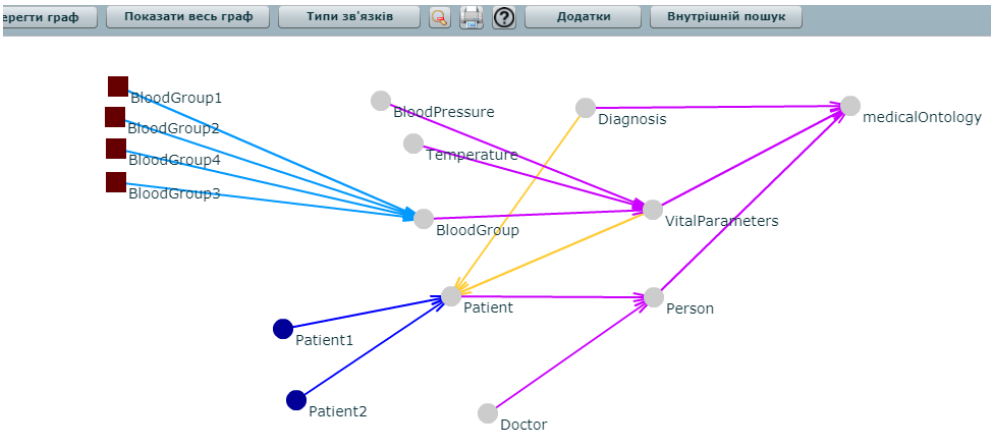

Рис. 3. Файл medical.xml у вигляді графа ТОДОС

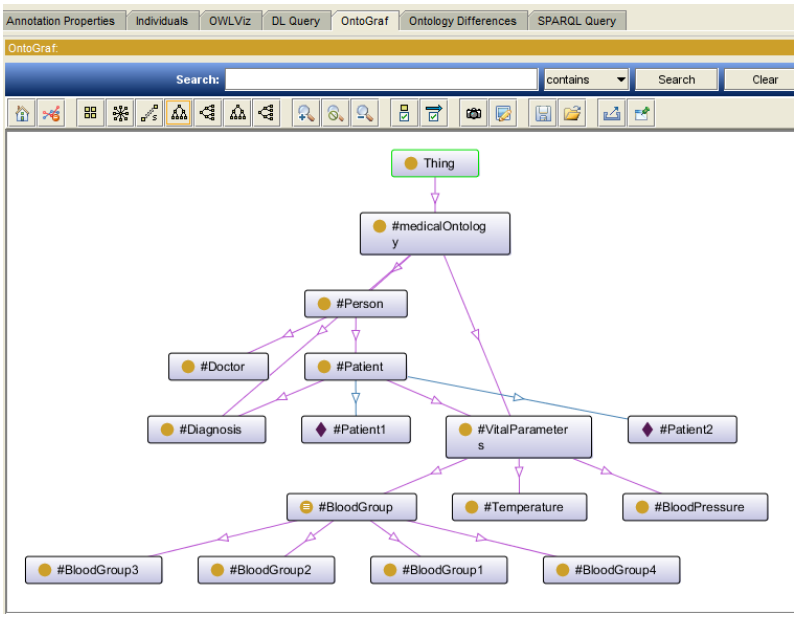

Рис. 4. Конвертований файл medical.owl у вигляді графа (OntoGraf)

Висновки. Таким чином, можемо відзначити, що візуально відображення онтографу в різних засобах співпадає з точністю до інтерфейсу засобу візуалізації.

Моделювання конвертеру між OWL та XML ТОДОС ускладнено тим, що не існує повної взаємовідповідності між цими мовами. Не зважаючи на те, що OWL є більш потужною семантичною системою, за деякими можливостями візуалізації XML ТОДОС перевищує OWL, і відповідно функція еквівалентності перетворення базових конструкцій заданих мов не володіє властивістю транзитивності. Проте, із певними допущеннями можна створити модель часткової конвертації, де втрати даних були б мінімальними.

Список використаних джерел

1. Анисимов В. В. Интеллектуальные информационные системы : курс лекций об онтологических системах [Электронный ресурс] / В. В. Анисимов. – Режим доступа : https://sites.google.com/site/anisimovkhv/ learning/iis/ lecture/tema11. – Дата звернення 15.05.2019р.

2. Гуралюк А. Г. Візуалізація предметних онтологій / А. Г. Гуралюк // Розбудова єдиного інформаційного простору української освіти – вимога часу : збірник матер. Всеукр. наук.-практ. WEB-форуму. – Кропивницький. Вид-во Льотної академії Національного авіаційного університету, 2018. – С. 135–143.

3. Клещев А. С. Математические модели онтологий предметных областей. Ч. 1. Существующие подходы к определению понятия «онтология» / А. С. Клещев , И. Л. Артемьева // Информационные процессы и системы. – 2001. – № 2. – С. 20–27.

4. Клишин А. П. Среды разработки Java-приложений / А. П. Клишин, С. А. Казарин, А. А. Мытник. – Томск : ТГПУ, 2013. – 108 с.

5. Михайлюк А. В. OWL как стандартная модель представления трансдисциплинарных знаний в Semantic Web / А. В. Михайлюк // Information Content and Processing. – 2014. – Vol. 1. № 3. – Pp. 249–261.

6. К интеграции онтологий предметных областей / А. Палагин, А. Михайлюк, В. Величко [и др.] // Information Models of Knowledge. ITHEA. – Kiev, Ukraine – Sofia, Bulgaria, 2010. – С. 69–85. – Режим доступу : http://citeseerx.ist.psu.edu/viewdoc/ download? doi=10.1.1.386.7791&rep=rep1&type=pdf. – Дата звернення 20.05.2019р.

7. Стрижак О. Онтологічні електронно-освітні ресурси – інформаційний базис підтримки розвитку обдарованості [Електронний ресурс] / О. Стрижак // Навчання і виховання обдарованої дитини. – 2013. – Вип. 2. – С. 79–87. – Режим доступу : http://nbuv.gov.ua/UJRN/Nivoo_2013_2_12. – Дата звернення 30.05.2019р.

8. Стрижак О. Є. Засоби онтологічної інтеграції і супроводу розподілених просторових та семантичних інформаційних ресурсів [Електронний ресурс] / О. Є. Стрижак // Екологічна безпека та природокористування. – 2013. – Вип. 12. – С. 166–177. – Режим доступу : http://nbuv.gov.ua/UJRN/ ebpk_2013_12_18.- Дата звернення 15.05.2019р.

9. Katifori A. Ontology visualization methods – A survey / A. Katifori // ACM computing surveys. – 2007. – Vol. 39, Iss. 4. – pp. 10–es, Nov. 2007.

10. Ontology-based Information Visualization [Electronic resource]. – Access mode : http://cciweb.uncc.edu/~ras/ RS/OBIV.pdf. – Дата звернення 30.09.2019р.

11. OWL 2 Web Ontology Language. Document Overview [Electronic resource]. – Access mode : http://www.w3.org/TR/owl2-overview/.- Дата звернення 27.10.2009р.

12. Plenary at WWW Geneva 94. International World Wide Web Conference, at CERN, Geneva, Switzerland, in September 1994. [Electronic resource]. – Access mode : http://www.w3.org/Talks/WWW94Tim/ Overview.html. – Дата звернення 10.09.2019р.

13. Bergh J. R. Ontology comprehension / J. R. Bergh. – Matieland : University of Stellenbosch, 2010. – 91 p.

References

1. Anisimov, VV n.d., *Intellektualnye informacionnye sistemy* : lecture course on ontological systems [Intelligent Information Systems] [Electronic resource] – viewed 15 May 2019, <https://sites.google.com/ site/anisimovkhv/learning/iis/ lecture/tema11>.

2. Huraliuk, AH 2018, ‘Vizualizatsiia predmetnykh ontolohii’[Visualization of subject ontologies], *Rozbudova yedynoho informatsiinoho prostoru ukrainskoi osvity – vymoha chasu*, Vydavnytstvo Lotnoi akademii Natsionalnoho aviatsiinoho universytetu, Kropyvnytskyi, pp. 135-143.

3. Kleshhev, AS & Artemeva, IL 2001, ‘Matematicheskie modeli ontologij predmetnyh oblastej,[Mathematical models of domain ontologies] Part 1 Sushhestvujushhie podhody k opredeleniju ponjatija ontologija’, *Informacionnye processy i sistemy*, no. 2, pp. 20-27.

4. Klishin, AP, Kazarin, SA & Mytnik, AA 2013, *Sredy razrabotki Java-prilozhenij*,[Java Application Development Environments] Tomskij gosudarstvennyj pedagogicheskij universitet, Tomsk.

5. Mihajljuk, AV 2014, ‘OWL kak standartnaja model predstavlenija transdisciplinarnyh znanij v Semantic Web’[OWL as a standard model for the presentation of transdisciplinary knowledge in the Semantic Web], *Information Content and Processing*, vol. 1, no. 3, pp. 249-261.

6. Palagin, A, Mihajljuk, A, Velichko, V et al. 2010, ‘K integracii ontologij predmetnyh oblastej’[Toward Integration of Domain Ontologies], *Information Models of Knowledge. ITHEA*, Kiev, Sofia, pp. 69-85, viewed 20 May 2019: http://citeseerx.ist.psu.edu/viewdoc/download? doi=10.1.1.386.7791&rep=rep1&type=pdf.

7. Stryzhak, O 2013, ‘Ontolohichni elektronno-osvitni resursy – informatsiinyi bazys pidtrymky rozvytku obdarovanosti’[Ontological e-education resources are an information base for supporting the development of giftedness], *Navchannia i vykhovannia obdarovanoi dytyny*, iss. 2, pp. 79-87, viewed 30 May 2019: <http://nbuv.gov.ua/ UJRN/Nivoo_2013_2_12>.

8. Stryzhak, OIe 2013, ‘Zasoby ontolohichnoi intehratsii i suprovodu rozpodilenykh prostorovykh ta semantychnykh informatsiinykh resursiv’[Means of ontological integration and maintenance of distributed spatial and semantic information resources], *Ekolohichna bezpeka ta pryrodokorystuvannia*, iss. 12, pp. 166-177, viewed 15 May 2019 <http://nbuv.gov.ua/UJRN/ebpk_2013_12_18>.

9. Katifori, A 2007, ‘Ontology visualization methods – A survey’, ACM computing surveys, vol. 39, iss. 4. – pp. 10–es, Nov. 2007.

10. Ontology-based Information Visualization, viewed 10 September 2019, <http://cciweb.uncc.edu/~ras/ RS/OBIV.pdf.

11. OWL 2 Web Ontology Language. Document Overview. W3C Recommendation :W3C, viewed 27 October 2009, <http://www.w3.org/TR/owl2-overview/>.

12. Plenary at WWW Geneva 94. International World Wide Web Conference, at CERN, Geneva, Switzerland 1994, viewed 10 September 2019: <http://www.w3.org/Talks/WWW94Tim/ Overview.html>.

13. Bergh, JR 2010, *Ontology comprehension*, University of Stellenbosch, Matieland.

Стаття надійшла до редакції 01.10.2019 р.