УДК 004.822

ПРОБЛЕМИ І МОЖЛИВОСТІ ПРОГРАМУВАННЯ В СЕРЕДОВИЩІ SEMANTIC WEB

П.І. Андон, Л.П. Бабенко

03680, Киев, Інститут программних систем НАН України, проспект Академіка Глушкова, 40

Широке розповсюдження Інтернету і створення глобальної світової мережі даних (WWW) створило нові засади розробки програм, обумовлені появою, з одного боку, нових викликів і проблем, які ускладнюють розробку, а з іншого, нових можливостей і бонусів, які її збагачують. У статті аналізуються згадані вище проблеми, підходи, які натепер допомагають їх вирішувати, зокрема напрацьовані для відомого середовища Semantic Web, та перспективні зміни парадигми програмування, які обіцяє це середовище.

Wide spread of the Internet and the creation of a global World Wide Web (WWW) has created a new basis for developing programs, due to the advent of, on the one hand, new challenges and problems that make it difficult to develop, and on the other, new promises and bonuses to enrich it. This article analyses the problems mentioned above, approaches that currently help to decide them, in particular, which are proposed in a well-known Semantic Web environment, and perspectives to changes in the current paradigm of programming..

Вступ

Інтернет створив нові обставини розробки програмних систем:

- Інтернет надав потужні засоби зберігання великих обсягів інформації.
- Інтернет надав потужні засоби транспортування інформації.
- Вказані обставини призвели до створення всесвітньої мережі World Wide Web (WWW), яка є одним з найбільших громадських репозиторіїв (сховищ) інформації у світі.

Всесвітня мережа (будемо називати її *вебом*) поєднує у собі величезну кількість інформації, більшість якої становлять ресурси вільного доступу, які можливо і доцільно ефективно використовувати при створенні та функціонуванні нових інформаційних систем, тобто ресурси повторного використання. Це обумовило реальні потреби у використанні програмних агентів при створенні та функціонуванні автоматизованих інформаційних систем, підвищені ефективності та розширенні сфери застосування таких агентів. Але для автоматичного здійснення своїх функцій агент має розуміти інформацію, з якою він має працювати. Зміст цієї інформації або можливість зрозуміти зміст йому може надати людина, яка створює агента та інформацію, але для цього вона має навчитися самій розуміти інформацію, яка створена іншими.

Інтернет створив принципово нове комп'ютерне середовище з невичерпною пам'яттю, вміст якої зберігається неструктурованим. Він дає змогу отримати відповідь на будь-який запит, але як правило, відповідь буде неосяжною або неточною. Тому можна стверджувати, що для того, щоб ефективно використовувати переваги Інтернету, необхідно вирішити такі проблеми:

- створення ефективних методів пошуку та ідентифікації потрібної людині інформації (з огляду на обсяги вебу та обсяги множини відповідей) і відповідне навчання цим методам програмних агентів,
- створення методів визначення програмними агентами змісту інформації, яку при наповненні вебу було орієнтовано на розуміння людиною,
- інтеграція програмними агентами знайденої інформації у потрібне прикладне застосування.

Вирішення цих проблем має чималі труднощі:

- дані, які є у вебі, зібрані з різних джерел, випадковим чином, в різний час,
- зміст та призначення даних вебу залишилися у головах їх творців,
- дані у вебі можуть бути розпорошені у різних місцях,
- структура і особливості представлення даних вебу відомі тільки їх творцям,
- творці даних можуть змінювати їх без попереднього повідомлення.

Дані вебу, призначені при створенні для використання людиною, для програмних агентів потребують спеціальних коментарів – знань щодо змісту і представлення даних, при цьому обсяги вебу унеможливлюють змінювати дані у відповідності з потребами програмних агентів, мова може тільки вестись про адитивне супроводження даних знаннями, які описують дані – такі знання отримали назву *метаданих*. Метадані можуть створюватись окремо від даних і в просторі, і в часі, але мають мати прямі посилання на дані, які вони описують.

Виходячи з вищезгаданого, на часі створення концепцій, мов та інструментів представлення знань щодо метаданих, без яких єфективне використання усього багатства вебу стає проблематичним.

На сьогодні відомий консорціум W3C інтегрує міжнародні зусилля, спрямовані на забезпечення даних вебу в форматах, доступних для автоматизованої обробки, інтеграції і розмірковувань. Мета цих зусиль полягає у перетворенні вебу даних у так званий *семантичний веб* (Semantic Web) [1], в якому дані супроводжуються поясненнями, що визначають їх семантику або зміст. Дані стають однозначно позначеними та адресованими.

Семантичний веб має на меті створити умови для використання усієї множини розподіленої інформації та послуг (*сервісів*), які існують у с віті глобальної мережі, як готові ресурси для новостворених програм.

Зазначені засобі поки що орієнтовані на процеси накопичення, пошуку інформації, визначення її змісту, а також її інтеграції та використання в нових умовах. Інакше кажучи, ми говоримо про повторне використання інформації в семантичній мережі. Зверніть увагу, що традиційні питання програмування (наприклад, формулювання вимог, алгоритмізація проблеми, архітектура, дизайн, тощо.) у рекомендаціях від W3C не розглядаються, ймовірно, вважається, що вплив на них специфіки семантичного вебу недостатньо досліджено. Однак, з нашого погляду, чітко проглядаються можливості для інформаційної підтримки та онтологічного управління всіх етапів життєвого циклу програмної інженерії..

Далі розглядаються напрацьовані дотепер моделі, методи та засоби семантичного вебу та їх вплив на моделі, методи та засоби розробки програмного забезпечення в семантичній мережі. Результати, напрацьовані для семантичного вебу, відкривають цілком нові можливості для розробників програмних систем, не використовувати які означало би професійне відставання.

1. Концептуальні основи семантичного вебу

Існують два основних аспекта програмування в семантичному вебі: представлення знань та інтеграція прикладних застосувань. Представлення знань повинне відображати семантику знань. Під семантикою будемо розуміти зміст або сенс. Знання щодо змісту даних дозволяють їх використовувати більш ефективно. У більшості джерел інформації значення змісту часто відсутнє, при їх використанні мається на увазі, що зміст знають користувачі, або його було закладено у логіку програми обробки даних. В обох випадках зміст розподілено у багатьох головах або програмах і повторне використання цього змісту потребує їх участі. Для подолання практики, яка склалась, при створенні семантичного вебу декларуються наступні настанови (парадигми) керування даними:

- Прикладні застосування семантичного вебу розглядають дані як центральнее, ключове питання розробки. Зміст даних (їх семантику) треба явно визначати. а не втілювати у алгоритми обробки або інструкції з використання. Метадані можуть створюватись окремо від даних і в просторі, і в часі, але мають мати прямі посилання на дані, які вони описують, тобто забезпечувати прямий доступ до даних від опису змісту і прямий доступ до опису змісту від даних.
- У вебі контент пов'язується з іншими контентами через універсальний ідентифікатор ресурсу - Universal Resource Identifier або URI [2]. Такий же механізм може бути застосовано для зв'язку інформації з описом її семантики (змісту).
- Дані можуть надходити з багатьох джерел, у тому числі рзподілених у вебі. Інтеграція даних з різних джерел потребує узгодження: а) відмінності у представленні даних, б) відмінності між моделями їх опису (метаданих), в) відмінності словників і відношень, використаних при моделюванні домену або проблемної області (онтологій).
- Дані є динамічними, тобто оперативно можна змінити структуру та значення інформації, що використовується в певному прикладному застосуванні, що не повинно приводити до надзвичайних ситуацій у інших прикладних застосуваннях.
- Дані відокремлені від застосування, їх представлення та опис можуть бути використані в будь-якому застосуванні.

Підхід семантичного вебу до моделювання знань наступний: знання з області джерела знань треба описати в термінах понять області призначення знань, при цьому перетворення контенту (і, відповідно, втрата інформації) не відбувається, змінюється тільки опис. Навіть якщо застосування не підтримує зовнішні поняття, воно може використовувати зв'язки між локальними та зовнішніми поняттями як основу для тлумачення.

Ядро моделі семантичного вебу складають п'ять понять.

1. ***Твердження***. В семантичному вебі інформація представлена як набір тверджень, кожне з яких складається з трьох частин: *суб'єкт, предикат, об'єкт*. Тому твердження іноді називають трійкою або триплетом. Три елементи твердження мають зміст, аналогічний їх змісту у граматиці природної мови.. С*уб'єктом твердження* є те, що воно описує, а предикат описує взаємозв'язок між суб'єктом та об'єктом. Наприклад, у твердженні «*Волга впадає у Каспійське море» Волга* є суб'єктом твердження, *Каспійське море* є об'єктом, а *впадає* є відношення між ними. Дві важливі особливості твердження спрямовані на спрощення інтеграції даних із різних джерел і можливості динамічного включення нових даних і побудови композицій даних з різних джерел: зміст тверджень не залежить від порядку їх появи у документі та суб'єкт, предикат та об'єкт твердження можуть бути представлені як URI.

2 ***URI***. URI (Uniform Resource Identifier) [2] – забезпечує унікальні для всього Інтернету імена. Кожний елемент твердження – суб'єкт, предикат, об'єкт може бути представлений своїм URI, і таким чином однозначно ідентифікуватися по всьому вебу. Розширенням URI є IRI – інтернаціоналізований ідентифікатор ресурсів, що забезпечує кодування за допомогою набору символів Юнікод. URI дозволяє визначити розширюваний простір імен, незалежний від масштабу адресування. URI може бути представлено як URL – Uniform Resource Locator (уніфікований покажчик місця розташування ресурсу), який може бути перенаправлений на додаткову інформацію. URI може бути представлено як абстрактне унікальне ім'я ресурсу URN (Uniform Resource Name).

3. ***Мова семантичного вебу.*** Твердження має бути висловлене мовою семантичного вебу. Мова складається з набору ключових слів, які дозволяють описувати інструкції для різних інструментів семантичного вебу. Користувач має на вибір кілька мов, що відповідають різноманітності і динаміці Інтернету, з різним ступенем складності та семантичної виразності, так що користувач може збалансувати свої вимоги до продуктивності і виразності. Вищі рівні виразності часто вимагають додаткових ресурсів пам'яті і обробки.

4. ***Онтологія***. Онтологія складається з тверджень, які визначають поняття, відношення і обмеження. Це схоже на діаграми бази даних або діаграми об'єктно-орієнтованих класів. Онтологія дозволяє для конкретної області інтересів (яку прийнято називати *доменом або проблемною областю* – ПрО) об`єднати попередньо визначений зарезервований словник термінів, які позначають властиві ПрО поняття та існуючі між ними відношення, в один документ, що служить як модель семантичного опису знань щодо ПрО. Існує чимало багатих онтологій, які можна безпосередньо включати до свого прикладного застосування або адаптувати для своїх потреб.

5. ***Дані екземпляра.*** Дані екземпляра є твердження, що містять інформацію про конкретні екземпляри, а не загальне поняття. *Джон* є екземпляр, в той час як *особистість* є поняття або клас. Це аналогічно до поняття *екземпляр* в об'єктно орієнтованій програмі. Семантичний веб забезпечує гнучкість в роботі з екземплярами. Екземпляр не прив'язаний постійно до будь-якого класу або набору класів. Екземпляр може взагалі не мати класу і існувати самостійно (як, наприклад, екземпляр твердження) або бути пов'язаним з декількома класами. Це дозволяє програмі додавати екземпляри перш, ніж вона зрозуміє їх відносини з класами. Застосування може динамічно змінювати асоціації екземпляра зі своїм класом або призначати декілька класів окремому екземпляру, при тому і під час виконання. Це дозволяє створювати і збирати інформацію незалежно від визначення класу. Дані екземпляра складають основну масу семантичного вебу.

Семантика даних виражається через поняття і відношення між поняттями. Семантичні відношення формують семантичний веб. Відношення у семантичному вебі існують окремо від понять, які вони пов`язують, і можуть вільно приєднуватися до будь-якого з тверджень. Це дозволяє підтримувати для відношень успадкування та правила обмежень. Колекції тверджень візуалізуються за допомогою графа.

2. Модель представлення знань у семантичному вебі

Проблему спільного використання даних декількома системами можна розділити на дві: проблему синтаксичного спільного використання та проблему семантичного спільного використання. Синтактичний аспект пов'язаний з отриманням доступу до спільних даних, тоді як семантичний аспект передбачає включення інформації у структури даних системи використання.

Зусилля по розробці нових практик управління інформацією, які би відповідали новому середовищу її зберігання та доставки до споживача, пройшли декілька стадій розвитку. Першою з них була поява так званих мов розмітки, серед яких XML [3], визнана на сьогодні базової мовою для розширень. У XML втілено ідею супроводжувати будь-яку інформацію в безмежній мережі Інтернет коментарями щодо її змісту. Власне елементи інформації (так званий *контент*) оточуються відповідними їм ярликами. які отримали назву *тегів* (*tag*), що пояснюють її призначення. Спочатку тегів було небагато і вони відносились переважно до засобів редагування документу. Однак, було введено два важливі інструменти: засіб визначення ієрархії вкладення контентів у документі та поняття простору імен (*namespace*), що описує іменовану сукупність імен елементів і атрибутів, призначених для забезпечення їх унікальності в XML-документі. Кожен раз при використанні деякого ідентифікатора треба вказувати простір імен, до якого він належить (як правило,URL документа, у якому визначено зміст цього ідентифікатора).

Механізм тегів дозволяє спеціалізувати XML на певні професійні спільноти, які змогли домовитися щодо стандартів на теги, які вони хочуть використовувати. Так з`явилися спеціальні розширення XML для математиків, хіміків тощо.

Наступним кроком у розвитку універсальних засобів специфікації інформації у документах стало розширення XML Schema [3], в якому з`явилися теги для визначення типів елементів та типів їх атрибутів. Але контент залишився неструктурованим.

Розширення RDF [4] дозволило структурувати контент, представивши його у вигляді множини тверджень, які пов`язують поняття та дані, що використовуються в документах, складною мережею відношень, які відображають їх семантику. Окрім додавання нових стандартних тегів, у RDF запропоновано кілька типових відношень, притаманних об'єктам більшості ПрО, а також дозволено кожному застосуванню мати специфічні тільки для нього відношення. Подалі з`явилися нові розширення RDF у напрямку зміцнення семантичного аспекту опису шляхом додавання стандартизованих тегів і відношень, зміст яких має бути узгодженим і фіксованим спільною угодою певного співтовариства фахівців, що працюють у певних ПрО або в області інформаційних технологій в цілому (зокрема мов побудови онтологій).

Ефективні рішення синтаксичної проблеми обміну даними може надати XML.

Семантична інформація в семантичному вебі моделюється переважно трьома мовами: Resource Description Framework (RDF) [4], RDF Schema – RDFS [5] та мови опису веб-онтологій OWL.2 [6], причому останні дві (RDFS і OWL.2) є розширенням першої (RDF). Синтаксична структура кожної з мов має вигляд композиції тегів і тверджень, але послідовно нарощується склад тегів і відношень, семантика (зміст) яких вважається наперед визначеною в просторі імен конкретної мови. Завдяки цьому вони виступають як елементи конструювання метаданих опису семантики знань. Співвіднесення даних та метаданих спрощує проблему

спільного використання даних на семантичному рівні. Разом дані та метадані роблять інформацію переносною, тому що відношення між значеннями даних не залежать від їх зберігання.

Мова RDF. Як зазначалося вище, в семантичній мережі інформація представлена як набір тверджень, що складається з трьох частин (трійки або триплети): суб'єкт, предикат та об'єкт. Твердження такого роду природно формують орієнтований граф, чиї вузли є суб'єкти і об'єкти кожного твердження і дуги, що їх поєднують – це предикати. Така модель даних формалізована мовою Resource Description Framework (RDF).

Графи не мають коріння. Об'єднання графів концептуально є те саме, як розміщення їх поруч один з одним.

Триплети самі є потужним інструментом для об'єднання даних. Триплети є тільки колекції URI і літералів, і кожен URI або літерал має глобальну область дії. Використання глобального простору імен має вирішальне значення, оскільки це означає, що триплети завжди можуть бути об'єднані без перекладу імен. Оскільки кожен компонент графа може бути використаний без перекладу, всі графи можна транспортувати і комбінувати без перекладу, що є величезною перевагою при обміні. Оскільки твердженя RDF не треба перекладати при передачі від однієї системи до іншої, вони діють у будь-якому контексті. Вони є повністю самодостатні і, як такі, вони не залежать один від одного. Ця незалежність означає, що порядок, в якому вони наведені, не має значення.

Як саме поняття графа, RDF як таке не є мовою, але є абстрактною моделлю, яка може бути серіалізованна (конкретизована) багатьма способами. Деякі з реалізованих форматів серіалізації сьогодні є досить популярні, наприклад, *Turtle* (Terse RDF Triple Language) [7].

Існує два типи вузлів: ресурси та літерали. Літерали представляють конкретні значення даних і можуть виступати тільки як об'єкти тверджень. Ресурси, навпаки, це все, що можна назвати, вони можуть бути і суб'єктами, і об'єктами. Ресурс – це ім'я, яке представляє об'єкт або поняття. Імена ресурсів мають форму згаданих вище URI.

Існують особливі вузли, що називаються пустими вузлами. Пусті вузли є змінні, які використовуються для узагальнення понять. Вони дозволяють пов`язувати триплети, не турбуючись про пов`язування ресурсів.

Предикати, які також називають властивостями, у свою чергу є ресурси і можуть бути представлені як URI. Розглянемо предикати RDF, які є наперед визначеними.

Предикат *RDF: type* дозволяє віднести ресурс до певного типу, тобто ствердити, що *ресурс-1* має тип, визначений ресурсом *ресурс-2*. Будь-який ресурс може мати багато тверджень щодо нього, у тому числі мати багато тверджень щодо його типу або взагалі не мати жодного.

RDF є розширенням XML, він використовує стандартні угоди щодо введення скорочень (так звані префікси для URI простору імен).

Літерали можуть представляти предикат. Атрибут *rdf:datatype* може вказувати, як інтерпретувати значення літералу. Літералу може бути призначений попередньо визначений тип даних XML Schema, або тип даних користувача, визначений URI. Для строкових літералів може бути вказано мову (наприклад, англійська, російська тощо.).

Твердження в RDF може бути про що завгодно, навіть про інше твердження. За допомогою твердження про твердження зручно проводити кваліфікацію або анотування інформації. Наприклад, одне твердження може містити теги джерел інформації або часові позначки додавання інформації до системи.

RDF має кілька конструкцій для групування інформації. *Контейнери* є групи ресурсів, що можна поповнювати або зменшувати під час виконання. В RDF можна визначати три типи ресурсів, які є контейнерами: *rdf:Bag* використовується для представлення групи невпорядкований ресурсів, rdf:Seq представляє впорядковану множину ресурсів, rdf:Alt, як і Bag, визначає невпорядковану колекцію, але її екземпляри є еквівалентними альтернативами.

RDF дозволяє групувати ресурси як списки, які не змінюватимуться навіть при злитті RDF-графів. Список можна обходити: предикат *rdf:first* посилається на перший елемент у списку, предикат *rdf:rest* відноситься до решти списку, яка першим елементом має другий ресурс первинного списку. Цей процес продовжується рекурсивно, поки список *rdf:rest* не стане *rdf:nil* (пусто).

RDF Schema (RDFS). Першим кроком визначення змісту інформації RDF є розробка спільного словника або колекції ресурсів, які мають наперед визначений зрозумілий зміст і завдяки цьому використовуються для опису інших ресурсів. Цей словник визначається у RDFS [5] за допомогою тегів з попередньо визначеним змістом. RDFS дозволяє організувати класи та властивості в ієрархії узагальнення та спеціалізації, визначити домен і очікувану область визначення властивостей, визначити членство в класі, визначити та інтерпретувати типи даних. Всі ресурси RDFS вважаються членами класу всіх ресурсів RDF і те ж саме вірно для всіх екземплярів. Далі можна описати конкретні екземпляри класів, робити твердження щодо їх властивостей або приписувати їм членство в інших класах, визначених в RDFS.

У семантичному вебі RDFS є фундаментальним інструментом для побудови онтологій, який виключає конфлікти імен під час інтеграції даних. Інструкції для програмування можуть зосередитися на програмній роботі, залишаючи осторонь складні питання узгодження представлення інформації у семантичному вебі під час інтеграції.

Мова опису онтологій для веба OWL (Ontology Web Language). Наступним кроком на шляху семантичного опису інформації стала поява мови для опису онтологій OWL.2 [6], яка розширює RDF і RDFS

додаванням у словник властивостей і класів, що мають асоційовану з ними семантику, використовувану для надання змісту даним, які вони описують.

Онтологія дозволяє зв'язати для конкретної області інтересів (називаної доменом чи проблемною областю – ПрО) визначений зарезервований словник термінів (відповідних поняттям ПрО) і існуючі між ними відношення в єдиний документ, який може слугувати цільною моделлю семантичного опису знань щодо ПрО. Це дозволяє повторно використовувати такі знання при створенні моделі іншої ПрО, встановлюючи відношення між поняттями ПрО-джерелом та ПрО-призначенням.

Специфічними для онтології типами ресурсів є індивід, клас, тип даних, властивість об'єкта, властивість типу даних і властивість анотації. Клас є колекція індивідів. Властивість є відношення або між двома індивідами, або між індивідом і значенням літерала. Індивід є екземпляр (також відомий як член) класу.

Онтологія в OWL.2 представляється як веб-документ, що містить заголовок і ряд визначень класів і властивостей, описів індивідів, а також описів області значень даних. Заголовок онтології є ресурс, що представляє саму онтологію. Заголовок описує онтологію і звичайно містить коментарі, етикетки, інформацію про версію, а також твердження імпорту онтології. Твердження імпорту онтології є важливий засіб інтеграції і повторного використання онтологій.

Онтології як засіб інтеграції знань. Технології семантичного веба дозволяють користувачам спільно розділяти і ревикористовувати (повторно використовувати) ті самі набори даних на різних платформах і в різних прикладних застосуваннях. Вони досягають цього завдяки двом найважливішим принципам дизайну: відокремлення моделі знань від прикладного застосування й інтеграція моделей знань шляхом повторного використання і розширення.

Спільне використання знань через границі проблемних областей і прикладних застосувань стає можливим, якщо встановити відношення між їхніми поняттями. Обсяг робіт, необхідних для інтеграції двох областей знань (чи прикладних застосувань) прямо пропорційний числу різних, неузгоджених концепцій у кожній. Процес може включати просте відображення між ідентичними поняттями, створення відношень підкласу чи підвластивості, операції такої ж складності, як математичні обчислення або маніпулювання зі строками. Деякі з цих відношень можуть бути встановлені з використанням конструкцій онтологій, а інші вимагають використання мов на правилах (див. далі).

Щоб полегшити інтеграцію доменів, треба прагнути звести до мінімуму число понять, що неузгоджені між доменами. Один із простих шляхів – ревикористання чи розширення понять існуючих онтологій.

Ревикористання онтологій базується на ревикористанні їхніх описів, для чого в OWL.2 визначені профілі або підмови. Основною метою профілю OWL є утворення підмножини OWL, яка дозволяє одержати поліпшені обчислювальні характеристики за рахунок зниження виразності, що робить профіль більш зручним для конкретних співтовариств користувачів і підвищує продуктивність конкретних технологій.

У рамках стандарту мови OWL.2 виділено три його стандартних розширення.(профіля): для моделювання багаторівневих ієрархій класів багатих таксономій; для моделювання інформації в існуючих базах даних і їхньої інтеграції і для використання в прикладних застосуваннях, що застосовують правила логічного виводу, різонери й інші системи обробки правил.

Коли ви використовуєте чи розширюєте онтологію, вона є основою для конкретної онтології вашого домена. Таку онтологію прийнято називати основною чи верхньою. Основна онтологія містить об'єкти і поняття, що виходять за границі однієї області знань. Ці онтології можуть спростити обмін інформацією шляхом створення середовища, у якому термінології розрізнених областей знань ідуть коренями в загальний простір. Дотепер створена представницька множина таких онтологій, ведуться роботи зі створення реєстрів і репозиторіїв онтологій, чимало яких вже сьогодні широко застосовуються, як наприклад:

BFO (Basic Formal Ontology) [8] підтримує онтологію, орієнтовану на наукові дослідження і складається з ряду підонтологий, що можуть бути віднесені до категорій або SNAP або SPAN онтологій. SNAP онтології виражають поняття, корисні для опису знімків (snapshots) речей, що або тривалі (вони не піддаються плину часу) або миттєві. SPAN онтології виражають поняття для опису речей, що тривають у період часу і для яких важливий часовий контекст.

Cyc **[9]** – це великий проект по штучному інтелекту, що розробляє всеосяжну онтологію і зв'язану з нею базу знань об'єктів і понять повсякденного життя. Метою проекту є створення баз знань, які можна використовувати у прикладних застосуваннях для розмірковувань про світ, що імітують роздуми людини. Open Cyc є підпроектом Cyc з відкритим вихідним кодом і публічною базою знань.

DOLCE [10] підтримує онтологію, що виражає об'єкти і поняття, релевантні природній мові і когнітивній науці.

Dublin Core Metadata Initiative [11] є проектом стандарту метаданих для широкого спектра прикладних застосувань, включаючи опис документів і мультимедіа, що дозволяє визначити поняття *назва, тип, опис, авторство* та *інформація про час створення,* що часто використовуються у властивостях анотації в OWL.

FOAF [12] підтримує онтологію *Friend of a Friend (друг мого друга), що* виражає інформацію про друзів на World Wide Web. Онтологія містить класи і властивості для збору особистої інформації, адреси електронної пошти, онлайн-рахунки й обмін миттєвими повідомленнями, а також інтернет-документи і зображення.

GeoRSS [13] є словник термінів, використовуваних у документах RDF для представлення геопросторової інформації.

Для пошуку і використання онтологій, що відповідають потребам конкретного прикладного застосування, розроблені реєстри і репозиторії онтологій, і навіть машини пошуку онтологій у Web, наприклад, Swoogle [14], що дозволяє переглядати індекси онтологій, використовуючи простий текстовий запит, як і при роботі з будь-якою інформаційно-пошуковою системою типу Google чи Yahoo.

3. Інструменти семантичного веба

OWL є інструмент тільки для визначення семантики моделей бази знань. По суті, база знань є колекція фактів (тверджень), створених під керуванням онтології відповідної проблемної області. Для побудови прикладних застосувань, що використовують інформацію семантичного веба, потрібна інтегрована робота спеціальних інструментів. Зупинимося коротко на основних з них.

Сховище триплетів повинне бути засноване на графовій моделі і надавати ефективні способи розміщення тверджень, забезпечуючи механізм для швидкого шляху до кожного твердження, що містить конкретний ресурс як його суб'єкт, предикат, чи об'єкт. Прикладом широко розповсюдженого інструмента роботи з онтологіями можна назвати Protégé [15].

Механізми виявлення інформації. Під виявленням (discovery) інформації будемо розуміти різні способи пошуку інформації, що зберігається у твердженнях RDF. Для пошуку інформації в семантичному вебі не існує єдиного методу, що працював би у всіх ситуаціях. У залежності від того, чи точно ми знаємо, що ми шукаємо, і де дані можуть існувати, і чи усвідомлюємо ми, наскільки дані структуровані, для одержання відповіді доцільно застосовувати навігацію, пошук (searching) чи запитування (quering). Інструментом реалізації запитів W3C рекомендує мову запитів SPARQL [16]**.** SPARQL підтримує чотири різних форми запиту:

SELECT – визначає кінцеві точки зв'язування термів RDF (порожні вузли, IRI, чи літерали) зі змінними на основі заданого шаблона запиту. Зв'язування просто використовуються для результатів, що повертаються, і не є частиною графа RDF. Множину результатів SELECT зручно відображати в табличну форму.

CONSTRUCT – дозволяє переформулювати зв'язані (bound) змінні в будь-який вид графа RDF, який можна створити, якщо вірний кожен триплет (наприклад, жоден літерал не використовується на позиції суб'єкта чи предиката). Ця форма запиту дозволяє легкий і потужний спосіб перетворення даних з одного графа RDF чи OWL онтології в інші. Графи, що повертаються у відповідь на запити CONSTRUCT, можуть бути додані в RDF репозиторії чи скомбіновані з іншими графами RDF.

ASK дозволяє довідатися, чи існує конкретний граф, видаючи булевску змінну зі значеннями *істина* або *неправда*. Клієнти можуть прозондувати цю кінцеву точку для отримання інформації без необхідності використовувати потенційно дорогі SELECT чи CONSTRUCT.

DESCRIBE – повертає RDF граф, обумовлений тільки процесором при введенні обмеженого запиту клієнта. При цьому клієнт не повинен знати, як дані структуровані. У кінцевому рахунку кінцева точка вирішує, які RDF дані повертаються клієнту.

Механізми розмірковування за допомогою логіки і правил над даними семантичного веба.

Факти можуть бути явними чи неявними. Явними фактами є ті, для яких є явні твердження в базі знань. Неявні факти є виводи, що представляють факти, існування яких випливає зі сполучення явних фактів, семантики онтології і правил у базі знань. Виводи отримують за допомогою компонента розмірковувань над базою знань, називаного різонером**.** У залежності від реалізації, виводи можуть додаватися безпосередньо до тверджень бази знань або генеруватися в міру необхідності з тверджень бази знань. Реалізації баз знань можуть виконувати вивід автоматично чи як зовнішній процес. Прикладом реалізації такої системи є Pellet [17].

Каркаси семантичного веба. Для забезпечення спільної роботи приведених вище інструментів застосовуються спеціальні інструменти, так називані *каркаси (framework*) інтегрирації інструментів для збереження і пошуку RDF-інформації, а також інтерпретації семантики OWL. Прикладом реалізованого каркаса семантичного веба може служити Jena Semantic Web Framework [18], що забезпечує спільну роботу згаданих вище Protégé, SPARQL, Pellet і ін.

4. Розмірковування над знаннями у семантичному вебі

У семантичному вебі передбачені механізми розмірковувань, що застосовують інтерпретацію семантики OWL до інформації в базі знань. Ці механізми, називані *різонерами*, комбінують твердження, що містяться в базі знань, з набором правил логіки, щоб одержувати висновки чи виконувати відповідні дії. Правила моделюють умовну конструкцію if – then (*якщо – то)* і складаються з двох частин. Перша частина моделює умову для правила і називається *body* (посилкою), а друга частина – head (голова) є висновком правила. Правило встановлює, що щораз, коли набір тверджень бази знань відповідає його умовам (посилці), у базі знань неявно припускається поява нового твердження, заданого висновком правила. За допомогою правил із заданої бази знань шляхом логічного виводу отримують нові, додаткові знання.

Правила можуть бути використані для вираження більшої частини семантики OWL і як інструмент користувачів для вираження довільних відношень, які не можуть бути змодельовані в OWL.

У деяких прикладних застосуваннях вивід реалізований як зовнішній компонент, що ініціюється вручну і висновки якого додаються в базу знань вручну. Останній підхід може бути використаний як метод зменшення обчислювальних витрат на виконання виводу, якщо це негативно позначається на загальній продуктивності бази знань.

Є два основних методи виконання різонером виводу, що базується на правилах:

- пряма побудова ланцюжка виводу чи прямий вивід. При прямому виводі усі висновки (маються на увазі припущення щодо фактів) заносяться прямо до репозиторію. Прямий вивід застосовується всякий раз, коли додаються нові факти, і в рамках тієї ж операції виведені висновки негайно додаються в базу знань. Як результат, база знань завжди містить усі явні твердження щодо фактів, а також усі неявні твердженя щодо фактів. Прямий вивід названий так тому, що висновок здійснюється роботою уперед від даних і правил у базі знань до висновків, що випливають з них;
- зворотна побудова ланцюжка виводу чи зворотний вивід. У цьому методі за допомогою розмірковувань здійснюється спроба вивести умови з цільового набору фактів (умова чи правила паттерна запиту), застосовуючи логіку системи в зворотному напрямку, поки умови не будуть задоволені на явних фактах бази знань.

Для завдання правил у семантичному вебі разроблені так називані мови на правилах. Серед них широке визнання отримала мова SWRL [19], орієнтована на взаємодію з онтологіями, представленими в OWL.2. У SWRL тіло і голова правила виражаються у виді кон`юнкції атомів, кожний з який є один із предикатів включення в клас, бінарних предикатів (властивостей об'єкта чи типу даних будь-якого типу), рівності, нерівності чи вбудованих функцій.

Оскільки розроблений і продовжує розроблятися цілий ряд мов на правилах, у W3C визнали за необхідне розробити спеціальний стандарт, орієнтований на забезпечення інтеграції, інтероперабельності та уніфікації декларативних програм, представлених у різних мовах і системах на правилах. У 2010 році W3C прийняв стандарт RIF (Rule Interchange Format) [20]. Варто підкреслити, що даний стандарт не є пропозицією «найкращої» мови на правилах. Його ціль – надати засоби для уніфікованого, інтероперабельного використання специфікацій, виконаних на різних мовах на правилах. Його роль – служити проміжною мовою, що забезпечує можливість відображення, що зберігає семантику, у різні мови на правилах. Для реалізації зазначених вище мети і ролі RIF представлене як сімейство мов, називаних діалектами, зі строго визначеними синтаксисом і семантикою. Діалекти повинні відбивати якнайбільше особливостей синтаксису і семантики існуючих мов на правилах і в той же час бути розширюваними мовами, що дозволяють визначати нові діалекти RIF як синтаксичні розширення існуючих з необхідною додатковою функціональністю. Такі розширення надалі можуть бути стандартизовані.

У рамках RIF розроблений каркас розширення логічних діалектів RIF-BLD, що надає засоби визначення синтаксису і семантики діалектів при завданні ряду параметрів. Дотепер визначений також діалект RIF-PRD для роботи з мовами на продукційних правилах.

5. Композиція даних у семантичному вебі

Один з основних варіантів використання семантичного веба полягає у віртуальній інтеграції інформації з декількох розрізнених джерел. Цей процес інтеграції може бути розкладений на два основних кроки: приведення даних до загальної моделі даних (RDF у випадку семантичного веба) і опис даних, використовуючи єдину модель знань. Як тільки дані будуть об'єднані в загальну модель, доступ до агрегованої інформації і маніпулювання нею стають можливими в одній моделі, однак, дані, як і раніше, будуть описуватися за допомогою різних словників. Щоб дані були цілком інтегровані, вони повинні бути об'єднані в загальну модель даних і описані за допомогою єдиної моделі знань.

6. Семантичні веб-сервіси

Сервісно-орієнтована модель програмування. Сьогодні у всесвітній глобальній мережі існують тисячі традиційних веб-сервісів.

Веб-сервіс є програмна система, призначена для підтримки взаємодії в мережі інтероперабельних машин. Зазначимо, що веб-сервіс є також повторно використовуваним ресурсом, але з наступними особливостями:

- сервіс призначений для використання під час виконання;
- сервіс призначений для використання в середовищі веб;
- сервіс може ревикористовуватися, якщо контекст його використання визначається незалежно від нього;
- сервіс може ревикористовуватися одночасно в багатьох контекстах (багатьма прикладними застосуваннями), тобто відповідати на виклики інших сервісів, взаємодіяти шляхом комунікацій так, щоб кожна взаємодія не руйнувала інші взаємодії і не заважала активізації нової взаємодії. Іншими словами, кожна інформаційна система може використовувати сервіс незалежно від того, де він оперує і хто його контролює;
- користувач сервісу контролює тільки життєвий цикл активності сервісу, тобто взаємодії з ним, водночас, як життєвий цикл самого сервісу, його зміни знаходяться поза контролем користувача;
- виходячи з попередньої особливості, будь-які зміни сервісу повинні породжувати версії, функціонально і синтаксично сумісні з попередніми версіями.

Здатність інтегрувати й об'єднати сервіси в корисні збірні прикладні застосування вимагає великої ручної роботи. Розроблювач повинен вивчити кожен сервіс стосовно його змісту або семантики, а потім

визначити, як використати цей зміст за допомогою правильного синтаксису і протоколу. Крім того, існують обмежуючі "договірні угоди" для різних інтерфейсів веб-сервісів. Інтерфейси можуть бути змінені чи вилучені їх постачальниками в будь-який час, і розроблювачі, що використовують ці сервіси, ніколи не зможуть довідатися, що зміни відбулися, поки залежні від них прикладні застосування не перестануть працювати.

Проблеми специфікації семантичних веб-сервісів. Як було відзначено раніше, семантичний веб націлений на використання безлічі розподіленої інформації і сервісів, що існують як готові ресурси для нового прикладного застосування. Веб-сервіси, створювані та використовувані автоматично програмними агентами, будемо називати *семантичними веб-сервісами* чи семантичними сервісами. Такі сервіси мають забезпечити машині можливість динамічно виконувати процеси виявлення, виклику, узгодження і композиції веб-сервісів для досягнення користувачами деякої кінцевої мети. Виявлення сервісу вимагає опису його семантики. Виклик описує, як сервіс запускається чи виконується. Узгодження охоплює широкий набір питань, від узгодження интерфейсів до угод щодо вартості, частоти використання, часу чекання реакції й ін. Композиція сервісів вимагає побудови з них потоку робіт.

Дотепер були розглянуті моделі і методи організації, представлення, збереження, пошуку й інтеграції інформації в семантичному вебі. Щоб мінімізувати участь людини у використанні засобів обробки веб-информации – веб-сервісів – потрібні також спеціальні моделі і методи організації, представлення, збереження, пошуку й інтеграції веб-сервісів.

Для декларованих концепцією семантичного веба задач взаємодії з мінімальною участю людини ідеальними були би послуги ряду чітко визначених інформаційних сервісів, що дозволяють самостійно зареєструватися, запросити в глобальній мережі реєстрації сервісів потрібні йому доступні сервіси і зв'язатися з ними для спільної роботи за допомогою програмних агентів. Для всіх цих дій необхідно чітко й однозначно визначити, які дані він повинен повідомити при своїй реєстрації в деяких репозиторіях чи сховищах сервісів, щоб інші сервіси чи користувачі могли його знайти, у які теми чи індекси репозиторіїв збереження сервісів він повинен бути включений, як знайти потрібні йому сервіси і як з ними зв'язатися, нарешті, як інтегрувати його в нове прикладне застосування, що створюється програмним агентом.

Перелічені проблеми перетворення веб-сервісів у семантичні веб-сервіси подібні з аналогічними проблемами інформації семантичного веба, розглянутими раніше. Тому і рішення подібні – супроводжувати сервіс описом (який здатні зрозуміти потенційні користувачі сервісу, у тому числі програмні агенти) його функціональності, інтерфейсів, шляхів доступу й ін. При цьому зростає роль стандартизації засобів опису і реалізації сервісів – сервіс може повторно використовуватися, тільки якщо інші сервіси можуть з ним взаємодіяти й обмінюватися інформацією. Якщо усі вони розроблені незалежно, імовірність такої взаємодії дорівнює нулю. Дотепер ряд стандартів взаємодії веб-сервісів прийнято як рекомендації W3C, інтенсивна робота ведеться над розширенням їхнього кола в ряді інших професійних об'єднаннях. Короткий огляд підходів до проблеми семантичного опису веб-сервісів подано у [21]. Зупинимося на аспектах цієї проблеми.

Аспект функціональності. Даний аспект є ключовим для розуміння функцій і призначення веб-сервіса. Водночас він гірше піддається формалізації, оскільки є продуктом неформальної системи, якою є людина, і виражає звичайно її бачення змісту програми. Природнім прагненням при цьому є використання термінів ПрО, до якої відноситься описувана програма. У розглянутих у [21] проектах UDDI, WSMO, SAWSDL характеристики аспекту функціональності представлені послугами, що користувач може отримати при зверненні до сервісу – це аналог поняття *Use Case,* вперше введеного в UML. Зміст послуг виражається за допомогою онтологій. Онтологія представляє систему класифікації, у якій зміст послуги може бути виражений вузлом цієї онтології. На кожнім з рівнів ієрархії в специфікації аспекту функціональності можна одночасно застосовувати кілька класифікацій, тобто характеризувати функції об'єкта специфікації з різних точок зору. Такий підхід дозволяє застосувати для виявлення сервісу засоби навігації, пошуку і запитів вище згаданої мови SPARQL.

Аспект інтерфейсу виділений як самостійний у більшості запропонованих стандартів. У ньому визначаються правила звертання до програми за отриманням оголошених у ній послуг – так звані кінцеві точки встановлення зв'язку з послугою (наприклад, мережні адреси зв'язування), виконувані в рамках послуги операції, їх вхідні і/чи вихідні параметри, необхідність взаємодії з іншими програмами, із зовнішніми факторами й ін.

У стандарті **WSDL** [22] інтерфейс описується на двох рівнях: абстрактному і конкретному. На абстрактному рівні (синтаксичному) для компонента опису, називаного *інтерфейс (interface),* вказуються абстрактні *операції,* що може виконати сервіс (під операцією розуміється простий обмін повідомленнями з клієнтом). Для операцій указуються типи повідомлень і так називаний *паттерн обміну повідомленнями (*характер обміну – уведення, чи видача, чи і те й інше*)*. Визначаються також дії при виняткових ситуаціях. На конкретному рівні (рівні реалізації) елемент *binding* (зв'язування) визначає точки доступу до сервісу (мережні адреси і протоколи зв'язування), а також унікальне ім'я сервісу, що дозволяє створювати однозначні посилання на компоненти опису сервісу у відповідних сховищах.

У стандарті **SAWSDL** [23] пропонується набір атрибутів розширення WSDL і XML Schema, призначених для опису семантичних властивостей компонентів WSDL. Ці атрибути дозволяють посилатися на поняття семантичної моделі, визначеної поза специфікаціями WSDL: додатковий атрибут *category* для елемента *interface* задає інформацію про категоризацію функціональності сервісу за допомогою вузлів відповідних онтологій, як вищезазначено.

У проекті **WSMO** [24] крім характеристик, запропонованих WSDL і використовуваних у проекті за допомогою механізму нефункціональних властивостей, додаються дві спеціальних характеристики інтерфейсу:

Хореографія (choreography) – описує взаємодію веб-сервіса і його клієнта. Клієнтом може бути людина, інший веб-сервіс чи інше прикладне застосування. Концепція хореографії базується на абстрактній машині станів. Її складовими є: *стан (state)* (описуваний як множина явно зазначених екземплярів понять, чи відношень функцій і значень їхніх атрибутів) і *перехід (guarded transitions), що* визначає умову зміни стану, задану у формі аксіом спеціальної мови WSML (розробленої в рамках того ж проекту), а також необхідну модифікацію стану при її істиності.

Оркестровка (Orchestration) визначає послідовність і умови викликів інших сервісів, необхідних даному для реалізації його функціональності. Ця характеристика також базується на абстрактній машині станів. Її складові: *стан (state)* (описуваний як множина явно зазначених екземплярів понять, чи відношень функцій і значень їхніх атрибутів) і *переходи (guarded transitions),* але *перехід* визначає умову виклику необхідного веб-сервісу і посилання на використовуваний медіатор (посередник).

Зазначимо, що вищеприведені характеристики використовують спеціально задану онтологію інтерфейсу, що відбиває зовнішні фактори, які взаємодіють з веб-сервісом, і характерні для них події.

Кожне з вищерозглянутих рішень з специфікації семантичних сервісів дає деяке розуміння їх проблем, але жодне з рішень не стало незаперечним переможцем цих проблем.

7. Особливості життєвого циклу розробки програм у семантичному вебі

Керована знаннями розробка програм вимагає спеціальних засобів накопичення знань про моделі проблемих областей і ассоційовані з ними колекції повторно використовуваних готових ресурсів (ГОР), корисних на визначених етапах ЖЦ розробки. У семантичному вебі, як було вищевідзначено, такими засобами є онтології і мови на правилах.

Будемо розрізняти три категорії знань:

1. Прикладні знання, що відносяться до визначеної проблемної області (ПрО), що відбивають виробничі інтереси фахівців виділеного визначеного профілю у рамках виділеного людського співтовариства, наприклад: визначені ролі в бізнесі (бухгалтер, менеджер по кадрам і т.п.), визначена область науки, визначене регіональне співтовариство, визначена фірма.

2. Загальносистемні знання, що відносяться до окремих процесів програмної інженерії на окремих етапах ЖЦ (визначення вимог, функціональне й архітектурне проектування, кодування, тестування, документування, навчання користувачів тощо) і/чи до організаційних процесів (зокрема, що стосуються розробки і використання програмних систем, наприклад, ліцензійні проблеми, антивірусний захист і авторизація доступу, взаємні трансформації форм представлення об'єктів тощо). Будемо вважати, що такі знання відносяться до домену програмної інженерії. Надалі будемо посилатися на них як SEN (Software Engineering) на відміну від інших прикладних областей, за якими збережемо позначення ПрО. Система знань відносно SEN відповідає [29] і відіграє подвійну роль: по-перше, відбиває знання щодо процесів ЖЦ розробки, незалежні від прикладних застосувань, і служить для класифікації універсальних методів, інструментів і ГОР, доступних системі на конкретних етапах ЖЦ. По-друге, використовується для анотування ГОР конкретного прикладного застосування на конкретному етапі ЖЦ.

3. Область знань, що відноситься до засобів специфікації різних типів ГОР, відомих натепер у визначених колах професійних розробників, у тому числі припустимі форми варіантності, що надає даний тип ГОР.

Таким чином, ГОР у системі атестуються за трьома головними вимірами: визначеного прикладного застосування; визначеного етапу ЖЦ розробки; визначеного типу ГОР. Вищим рівнем класифікації будемо вважати область прикладного застосування.

Кожному з перелічених типів знань відповідає своя онтологія, що в ідеалі підтримується в актуальному стані відповідним професійним співтовариством.

Запропонована категоризація знань дозволяє розроблювачу ооримувати наступні інформаційні послуги: 1) пошук знань, що відповідають ПрО, у сфері якої він веде розробку; 2) пошук ГОР, що відомі в даній ПрО; 3) пошук ГОР, що можуть бути використані на визначеному етапі ЖЦ; 4) пошук відомих ГОР заданого типу.

При наявності керування знаннями вищеперерахованих трьох категорій традиційний ЖЦ розробки може бути модифікований у такий спосіб.

Етап 1. Інженерія вимог на розробку під керуванням онтології знань щодо ПрО. Онтологія дозволяє утримувати користувача в максимально можливому просторі визначених можливостей, зміст яких зафіксований і зрозумілий як розроблювачу, так і замовнику. Очевидно, що перебування в такому просторі гарантує обох сторін договору (який матеріалізований у виді вимог) від взаємних непорозумінь.

Етап 2. Пошук ГОР, корисних для використання в плановаій розробці на поточному етапі життєвого циклу. Вивчення знайдених кандидатів на використання й інтеграція придатних з них у створювану розробку.

Етап 3. При відсутності чи недостатності придатних ГОР консультації під керуванням онтології SEN про інструменти, які можуть бути надані середовищем на даному етапі ЖЦ і застосування їх для розробки відсутніх ресурсів. Інтеграція їх у створювану розробку.

У процесі керування розробкою знання, представлені онтологіями, можуть відбивати різні артефакти, такі як:

- виконавці, їхні атрибути, їхнє завантаження, стадії готовності доручених їм робіт;
- готові компоненти, їхні версії;
- конфігурації окремих випусків розроблювального програмного продукту чи його складових;
- інструменти тестування, тестові дані, протоколи тестування;
- навчальні матеріали та інструкції;
- протоколи виміру якості програмного продукту.

З вищеописаного випливає, що середовище розробки повинне містити методичну, інструментальну, довідкову і навчальну підтримку інформаційних потреб розроблювача. Інакше кажучи, мати у своєму складі керовану знаннями систему консультування розроблювача на всіх етапах ЖЦ розробки щодо доступних йому на поточній стадії його роботи ГОР, у тому числі методичних правил і стандартів, дотримувати яким він зобов'язаний чи зацікавлений. Успіх такого консультування базується на вдалих моделях вищезгаданих категорій знань.

Висновки

Семантичний веб – це нове інформаційне середовище, що змінює суть, стиль, інструменти розробки програм, що дозволяє застосувати знання для реального керування розробкою, завдяки чому фактор повторного використання може бути багаторазово посилений. Це відкриває зовсім нові можливості у використанні багатства розподіленої інформації і сервісів, що існують у всесвітній глобальній мережі, на комерційному підприємстві і як особисті ресурси, не використовувати які означало б утрату своєї конкурентноздатності. Семантичний веб забезпечує принципово нові процеси в життєвому циклі розробки програм:

- інформаційну підтримку керування розробкою на всіх етапах ЖЦ;
- інженерію вимог під керуванням моделі знань;
- пошук готових ресурсів розробки і доступних інструментів їхньої інтеграції;
- конфігурацію продукту під керуванням онтології версій і змін;
- верифікацію продукту під керуванням онтології тестування;
- навчання користувачів під керуванням онтології знань про функціональні можливості продукту;
- керування віртуальними колективами розроблювачів.

Основні напрямки робіт із застосування і розвитку ідей семантичного веба:

- розробка моделей, методів і засобів доступу до світових інформаційних ресурсів;
- створення моделей опису готових ресурсів розробки як повторно використовуваних знань;
- створення і стандартизація онтологій верхнього рівня (універсальних відносно ПрО);
- створення бібліотек онтологій за профілями знань;
- розробка методів і засобів інтеграції світових онтологій в украиномовний простір;
- створення комплексу інструментів ведення онтологій як робочого місця онтолога;
- реєстри програм;
- розробка методичних і навчальних матеріалів по застосуванню семантичного веба.

1. *W3C* Semantic Web Activity home page /URL: http://www.w3c.org/
2. *Berners-Lee*, *R. Fielding, and L. Masinter*: RFC 3986 - Uniform Resource Identifiers (URI): Generic Syntax, *IETF,* January 2005 /URI:http://www.isi.edu/in-notes/rfc3986.txt.
3. *Дейтел Х.М.* і ін. Як програмувати на XML. – «Біном». – 2001. – 874 с.
4. *Resource* Description Framework (RDF) */URL:* http://www.w3.org/RDF.
5. *Resource* Description Framework Schema (RDFS)/URL: http://www.w3.org/2000/01/rdf-schema#.
6. *OWL.2* Web Ontology Language/URL: http://www.w3.org/2002/07/owl#.
7. *Terse* RDF Triple Language (Turtle)/URL*:* **http://www.w3.org/TeamSubmission/turtle/**.
8. *BFO* (Basic Formal Ontology) /URL: http://www.infomis.org/bfo.
9. *OpenCyc* /URL: http://sw.opencyc.org,.
10. *DOLCE* /URL: http://www.loa-cnr.it/DOLCE.html.
11. *Dublin* Core Metadata Initiative/URL: http://dublincore.org.
12. *FOAF* /URL: http://www.foaf-project.org .
13. *GeoRSS* /URL: http://georss.org.
14. *Swoogle* /URL: http://swoogle.umbc.edu.
15. *Protégé* Protégé Ontology Editor 4.0 Alpha /URL: http://protege.stanford.edu/download/protege4/installanywhere/.
16. *SPARQL* Protocol and RDF Query Language W3C Recommendation 2008. URL: http://www.w3.org/TR/2008/REC-rdf-sparql-query-20080115/.
17. Pellet ReasonerFactory Pellet /URL: http://pellet.owldl.com.
18. *Jena* Semantic Web Framework /URL: http://jena.sourceforge.net.
19. *Semantic* Web Rule Language (SWRL) /URL:
20. Rule Interchange Format overview//W3C Working DRAFT.-URL;http://www.w3c.org/TR/2010/WD-rif-overview-2010511-2010.
21. *Бабенко* Л.П. Характеристический анализ современных подходов к спецификации семантики программ // Труды второго симпозиума «Онтологическое моделирование». – Казань, 11-12 октября 2010. – М: ИПИ РАН, 2011 . – С. 270–285.

22. *Web* Service Description Language. W3C Recommendation /URL: http://www.w3.org/TR/2007/REC-wsdl20-20070626.
23. *Semantic* Annotations for WSDL and XML Schema. W3C Recommendation. /URL: http://www.w3.org/TR/sawsdl/.
24. *Web* Service Modeling Ontology – WSMO /URL: http://www.w3.org/Submission/WSMO/,
25. *Andon P., Deretsky V.* Control Oriented Ontology and Process Description for Cooperation Agents in Information Retrieval / Sixth International Scientific Conference „Electronic Computers and Informatics ECI'2004" . September 22-24, 2004. – Kosice – Herlany, Slovakia.
26. *Andon P., Deretsky V.* The Semantic Web Technology for Improving Existent Information Retrieval Systems/ 10TH Panhellenic Conf/ on Informatics. The Proceedings LNCS. –2005. – P. 367 – 373.
27. *Andon P., Deretsky V.* Approach to Automatic Creation of Ontology from Documents for Improving Existent Information Retrieva l /2 nd Balkan Conference in Informatics (BCI'2005) November 17-19.– 2005.– P. 236 – 241.
28. *Hebeler J, Fisher M., Blace R., Perez-Lopez A.:* Semantic Web Programming/ Wiley Publishing, Inc. – Indianapolis, Indiana. – 2009 . – 651 p.
29. *Software* Engineering Body of Knowledges (SWEBOK): URL: http://www.swebok.org