ного поповнення онтологічних баз знань типу WordNet новими семантичними відношеннями.

Треба також зазначити, що виявлення семантичних відношень не завжди є однозначним. Наприклад, пара слів <dog> і <black>, окрім стандартних значень цих слів (тварина собака і чорний колір), отримали під час пошуку найкоротших ланцюжків також інші значення:

<dog> – cad, bounder, blackguard, dog, hound, heel – (someone who is morally reprehensible) – грубіян, хтось, хто заслуговує на осуд;

<black> – black, disgraceful, ignominious, inglorious, opprobrious, shameful – ((used of conduct or character) deserving or bringing disgrace or shame) – ганебний, безчесний, заслуговує на немилість, сором чи ганьбу.

Коректні значення з'являлись під час тестування отриманих інструкцій набагато частіше, однак також потрапляли і результати, подібні до наведеного прикладу. Неприйнятні значення увійшли до результатів і відсіюватимуться у майбутній семантичній постобробці.

1. Miller G. Wordnet : An online lexical database / G. Miller // International Journal of Lexicography. – 1990. – №3 (4).
2. Miller, George A., Christiane Felbaum., J. Kegi, and K. Miller 1988. Wordnet : An electronic lexical reference system based on theories of lexical memory. 17. – P. 181–211.
3. Vossen P., L. Bloksma, H. Rodriguez, S. Climent, N. Calzolari, A. Roventini, F. Bertagna, A. Alonge, W. Peters. The EuroWordNet Base Concepts and Top Ontology. EuroWordNet (LE 4003) Deliverable D017, D034, D036. University of Amsterdam.
4. Temperley D, Lafferty J., Sleator D. 1995. Link Grammar Parser. – Режим доступу: http://www.link.cs.cmu.edu/link
5. Кормен Т., Алгоритмы : построение и аналіз / Т. Кормен, Р. Ривест. – Минск, 2005.

A. Anisimov, M. Glybovets, P. Kulyabko, O. Marchenko, K. Lyman

DEVELOPING METHODS FOR SEMI-AUTOMATIC EXTENDING AND REPLENISHMENT OF ONTOLOGICAL KNOWLEDGE BASES

Task of replenishment for WordNet type knowledgebases was investigated. Semi-automatic methods for adding correct links and relations between ontology elements were designed. These methods are based on fine heuristics and are quite robust.

УДК 004.4'23

Федорченко В. М.

БАЗА ЗНАНЬ КОМПОНЕНТІВ РЕПОЗИТОРІЯ NRECO

У роботі розглянуто основні аспекти побудови бази знань про програмні компоненти для репозиторія NReco. Визначено онтологію (RDFS/OWL) для опису компонентів різних рівнів абстракції з метою використання цих знань для організації ефективного пошуку в репозиторії. Особливу увагу зосереджено на видобуванні знань з XML-моделей та XSL-трансформацій; запропоновано евристичний механізм видобування знань про метамоделі XML-моделей в умовах відсутності повного формального визначення таких метамоделей.

Вступ

Сучасні підходи до побудови компонентної інфраструктури [1] та методи розробки програмних компонентів дають змогу забезпечити технічну можливість для їх ефективного повторного використання. На практиці ця можливість може бути реалізована лише за сприятливих умов, коли кожен програміст володіє повною інформацією про всі існуючі компоненти та способи їх використання. Сучасні платформи програмування, такі як Java чи Microsoft.NET, налічують тисячі стандартних компонентів, які добре задокументовані та загальновідомі; кількість до-

ступних сторонніх компонентів вимірюється десятками тисяч. При цьому ще існують корпоративні бібліотеки компонентів (зазвичай недостатньо документовані), які є в будь-якій розвинутій компанії, що розробляє програмні продукти. За цих умов ефективність повторного використання, яка передбачає пошук та інтеграцію наявних компонентів, безпосередньо залежить від особистих знань, досвіду та навичок кожного програміста в команді; внаслідок появи такого людського фактора спроби впровадження компонент-орієнтованої розробки досить часто зазнають невдачі або не виправдовують очікувань. Коли ж мова йде про модель-орієнтовану розробку та предметно-залежне моделювання [2], розв'язання проблеми релевантного пошуку та ефективного використання знань про програмні артефакти (моделі, метамоделі, трансформації) набуває першочергового значення.

Одним з можливих варіантів розв'язання цієї проблеми є створення бази знань про програмні компоненти та їх конфігурації, та побудова на основі бази знань спеціальної експертної системи. Основними функціями такої системи мають бути:

- індексація та видобування знань (knowledge extraction) з вихідних файлів;
- побудова внутрішнього представлення знань;
- обробка запитів та візуалізація результатів пошуку.

В цій роботі розглянуто основні аспекти побудови такої бази знань для компонентного середовища на основі платформи NET, а саме – в контексті репозиторія NReco [3]. Особлива увага присвячена альтернативним формам представлення компонентів (таким, як XML-моделі та XSL-трансформації), які в модель-орієнтованому середовищі NReco містять ключові знання. Основною базою знань можна використати низку стандартів RDF(S)/OWL/SPARQL [4, 5], оскільки ці стандарти пропонують уніфікований підхід до опису онтології, представлення фактів та механізмів обробки запитів.

Передусім визначимо форму і типи запитів, які повинні оброблятися такою базою знань. Найпоширенішою формою запиту є ключове слово, яке додатково може характеризуватись деяким контекстом – саме так працює контекстна довідка в MS VisualStudio (MSDN). Програміст має можливість викликати довідку, коли курсор знаходиться в межах деякого ідентифікатора; після цього інформація про цей ідентифікатор та його контекст (який дає змогу однозначно визначити семантичне навантаження) передається на обробку до бази знань. Відповідно бажаними результатами пошуку за таким запитом є максимально повна інформація про все, що стосується ключового слова (наприклад, якщо ключове слово визначає деякий клас, результатами може бути інформація про методи та поля класу, місце визначення класу, а також посилання на всі вихідні файли, де цей клас використовується).

Онтологія

В репозиторії NReco компоненти описуються у вигляді XML-файлів. База знань має містити інформацію про власне компоненти (ідентифікація, властивості), зв'язки між компонентами та місця (файли) їх використання. В середовищі NReco компонентами найнижчого рівня (базовими) є об'єкти (POCO, звичайний об'єкт CLR [6, 7]), які характеризуються такими власивостями:

- тип об'єкта (описується класом System.Type);
- строковий ідентифікатор компонента (не є обов'язковим). Цей ідентифікатор дає змогу організовувати посилання на компонент;
- залежності компонента (визначаються інтерфейсами залежностей та посиланнями на інші компоненти);
- інтерфейси компонента, які можуть використовувати інші компоненти;
- місцем опису (файл, розташування).

Складні компоненти визначаються як композиція базових компонентів; такі композитні компоненти зберігають основні властивості базових компонентів (ідентифікатор, залежності, інтерфейси). В термінах RDF(S)/OWL онтологія компонентів матиме такий вигляд (нотація N3, основні поняття):

@prefix c: <urn:schemas-nreco:metadata:component>.

@prefix owl: <http://www.w3.org/2002/07/owl#>.

@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>.

c:Type a owl:Class.

c:Component a owl:Class.

c:dependency a owl:ObjectProperty; rdfs:domain c:Component; rdfs:range c:Component.

c:componentType a owl:ObjectProperty; rdfs:domain c:Component; rdfs:range c:Type.

c:name a owl:DatatypeProperty; rdfs:domain c:Component; rdfs:range xsd:string.

c:implements a owl:ObjectProperty; rdfs:domain c:Component; rdfs:range c:Type.

c:dependsFrom a owl:ObjectProperty; owl: equivalentProperty.

c:dependency; a owl:TransitiveProperty; rdfs: range c:Component; rdfs:domain c:Component.

c:FileLocation a owl:Class.

C:fileName a owl:DatatypeProperty; rdfs:range xsd:string; rdfs:domain c:FileLocation.

C:fileStartIdx a owl:DatatypeProperty; rdfs: range xsd:integer; rdfs:domain c:FileLocation.

C:fileEndIdx a owl:DatatypeProperty; rdfs:range xsd:integer; rdfs:domain c:FileLocation.

c:definedIn a owl:ObjectProperty; rdfs:range c: Component; rdfs:domain c:FileLocation.

Наповнення бази знань

Всі базові та композитні компоненти описуються за допомогою XML-конфігурації IoC-контейнера [8] (є декілька винятків: ASP.NET застосування, де для опису компонентів інтерфейсу користувача використовуються метатеги, та WPF-застосування, де для тих самих цілей використовується XAML), отже, ця конфігурація є основним джерелом знань про компоненти нижнього рівня (приклад визначення об'єкта):

<component name=«uniqueName» type= «ClrType»>.

<property name=«Property1»><ref name= «uniqueName1»/>.

<property name=«Property2»><ref name= «uniqueName2»/>.

</component>.

З таких визначень задача формування фактів для бази знань є тривіальною задачею (для вищенаведеного прикладу, нехай тип ClrType реалізує інтерфейси ClrTypeInterface1 та ClrTypeInterface2, нотація N3):

c:#uniqueName a c:Component; c:name «uniqueName»; c:componentType c:#clrType.

c:#uniqueName c:dependency c:#uniqueName1; c:dependency c:#uniqueName2.

c:#uniqueName c:implements c:#clrTypeInterface1; c:implements c:#clrTypeInterface2.

Іншими важливими джерелами знань у репозиторії є XML-моделі, які трансформуються (низкою трансформацій відповідно до концепції багаторівневого налаштування [9]) до конфігурації IoC-контейнера та/або альтернативних форм опису компонентів (наприклад, метатеги ASP.NET), опис схем моделей (XSD) та власне опис трансформацій (XSL). В рамках XML-моделей (та їх трансформацій) теж можна виділити поняття компонента (тобто існування елементів, які ідентифікуються, мають чітко визначені залежності та інтерфейс або тип елемента). У цьому випадку відбувається трансформація (а отже, існує логічний зв'язок) між компонентами моделі та відповідними базовими компонентами, отже, для відображення цього зв'язку доповнимо онтологію такими поняттями:

c:generatedFrom a owl:ObjectProperty; rdfs: domain c:Component; rdfs:range c:Component.

c:generatedBy a owl:ObjectProperty; rdfs: domain c:Component; rdfs:range c:Component.

В загальному випадку XML-моделі мають довільну схему, тож ідентифікація компонентів у рамках таких моделей є нетривіальною задачею. На практиці поширеною є ситуація, коли метамодель визначена лише XML-схемою та (в неявному вигляді) наявними трансформаціями, а формального представлення онтології моделі (наприклад, у вигляді RDF(S)/OWL) немає.

Фактично в цьому випадку необхідно спочатку визначити метод формування онтології моделі (з XML-схеми та XSL-трансформації). Оскільки заздалегідь відомо, що схема та трансформації не містять достатньої інформації для формального визначення онтології, доведеться застосувати евристичні методи ідентифікації компонентів моделі.

Припустимо, що існує деяка XML-модель у вигляді дерева X={U,V} (де U – множина вузлів дерева), для якої достеменно відомо, що вона складається з деякої множини C компонентів. Оскільки компонент в XML-моделі ідентифікується деяким вузлом XML-дерева, то очевидно, що C ⊆ U. Звідси випливає постановка задачі для евристичного методу ідентифікації компонентів XML-моделі: визначити таку множину C_1, для якої б $|C_1 \Delta C|$ було мінімальне.

Основою XSL-трансформації є шаблон (xsl:template), який фактично являє собою правило перетворення деякого вузла XML-дерева. Очевидно, що деякі XSL-шаблони трансформують саме компоненти XML-моделі; також відомо, що XML-схема містить опис усіх можливих елементів моделі. Нехай $U_1 \subseteq U$ – множина вузлів, для яких існують XSL-шаблони, а $U_2 \subseteq U$ – множина вузлів, для яких існує опис в XML-схеми. Тоді шукану множину вузлів, що потенційно відповідають деяким компонентам, можна утворити таким чином: $C_1 = U_1 \cap U_2$. Насправді, множина C_1 буде містити всі елементи з множини C ($C \supseteq C_1$) з таких міркувань: характеристика компонента, як можливість повторного використання, буде вимагати наявності відповідного XSL-шаблону (оскільки шаблон передбачає можливість обробки будь-якої кількості елементів). Міра оптимальності ($|C_1 \Delta C|$) в цьому випадку буде залежати від структури XSL-трансформації та власне сутності самої моделі; будемо вважати, що евристичний метод є прийнятним (це можливо перевірити лише шляхом експерименту над реальними XML-моделями, схемами та їх трансформаціями), якщо буде зберігатися відношення:

$$|C|/|C_1| < 0.7.$$

Реалізація

Основою для бази знань було обрано бібліотеку SemWeb [10], яка реалізує необхідний мінімум компонентів для організації RDF-бази знань: сховище фактів, експорт-імпорт RDF XML/N3, обробку SPARQL-запитів та механізм виводу.

Наповнення фактами про базові компоненти (CLR-об'єкти) відбувається у кілька етапів:

- Індексуються наявні файли C# (ідентифікація – повне ім'я: простір імен та назва класу).

- Оскільки не всі класи можуть бути доступні у вигляді вихідних файлів, індексується також бінарний код (DLL) шляхом аналізу структури класів через механізм відображень (reflection).
- Індексуються файли конфігурації IoC-контейнера (на цьому етапі зв'язується інформація про CLR-типи та посилання на них в описах об'єктів).

Обробку XML-моделей, схем та трансформацій організовано так:

- Індексація XSL-трансформацій: формування фактів про компоненти трансформації (xsl: template).
- Індексація XML-схем з використанням інформації про трансформації, які можуть відноситись до відповідних елементів зі схеми (використання евристичних правил).
- Індексація власне XML-моделей: ідентифікація компонентів, визначення відношень c:generatedFrom та c:generatedBy шляхом аналізу трансформації за методом «чорної скриньки» (blackbox), коли до XSL-темплейтів додаються мітки, за якими визначаються згенеровані компоненти.

Побудована таким чином база знань для типового програмного проекту, що міститься в репозиторії NReco, налічує приблизно 15–20 тис. фактів. За такою схемою індексація відбувається в межах декількох хвилин (що цілком прийнятно, якщо проводити процедуру індексації як частину сценарію побудови проекту (build process)), а обробки запитів у межах декількох секунд.

Інтерфейсом доступу до бази знань став фасетний RDF-браузер, але завдяки сумісності за стандартами RDF(S)/OWL можливо використання будь-якого іншого RDF-сумісного фасетного браузера [11].

Висновки

Розроблений підхід до формування бази знань про програмні компоненти дає змогу ефективно генерувати проектну документацію та організувати ефективний семантичний пошук інформації про всі компоненти в репозиторії NReco. Запропонований механізм автоматичної генерації онтології для XML-моделей (в разі її відсутності) довів свою придатність на практиці (кількість помилкових ідентифікацій компонентів типових моделей в репозиторії NReco не перевищує 5 %); фактично цей механізм дає змогу повноцінно працювати з метамоделями, що не мають повного формального опису, а отже, суттєво економити час програмістів на підтримку такого опису (документування).

Подальший розвиток бази знань про програмні компоненти зосереджено на двох напрямах:

- деталізація бази знань шляхом введення додаткових властивостей компонентів та правил виводу (наприклад, визначення предикату «незалежний від», «довжина залежності» (шлях у семантичному графі тощо), що уможливлює програмістам швидко здійснити аналіз компонентної структури проекту;
- технологічна оптимізація механізмів індексації та обробки SPARQL-запитів. Якщо репозиторій буде вміщувати більш ніж 10–20 проектів (тобто кількіть фактів перевищить 100 тис.), то час, який треба затратити на індексацію і запити, виявиться неприйнятним.

1. Brown A. W., Large-Scale, Component-Based Development. Prentice Hall PTR, 2000.
2. Gasevic D., Djuric D., Devedzic V., Selic B. V. Model Driven Architecture and Ontology Development. Springer, 2006.
3. NReco : MDD Framework (open source project). – Режим доступу: http://nreco.googlecode.com/
4. Powers S. Practical RDF, O'Reilly Media, Inc., 2003.
5. Lacy L. W. Owl : Representing Information Using the Web Ontology Language, Trafford Publishing, 2005.
6. ISO/IEC 23271:2006 (ECMA-335). Common Language Infrastructure (CLI).
7. Miller J., Ragsdale S. The Common Language Infrastructure Annotated Standard, Addison-Wesley, 2004.
8. Федорченко В. М. Каркас для підтримки модель-орієнтованої розробки на основі спрощеної інфраструктури трансформації xml-моделей / В. М. Федорченко // Наукові записки. – 2008. – Т. 86.
9. Czarnecki K., Antkiewicz M., Kim P., Hwan C. «Multi-Level Customization in Application Engineering». Communications of the ACM (New York, USA: ACM Press) 49: 60–65.
10. Semantic Web/RDF Library for C#/.NET. – Режим доступу: http://razor.occams.info/code/semweb/
11. Oren E., Delbru R., Decker S., «Extending faceted navigation for RDF data». The 5th International Semantic Web Conference (ISWC) 2006, Athens, GA, USA, November 5–9, 2006. Proceedings: Springer Berlin.

V. Fedorchenko

NRECO REPOSITORY COMPONENTS KNOWLEDGE BASE

In this paper we address main aspects that concern building knowledge base of software components for NReco repository. Components ontology (RDFS/OWL) is defined for describing components of different levels of abstraction. This approach provides specifically designed means for organizing semantic search in the repository. Particular attention is paid to the knowledge extraction from XML-models and XSL-transformations. Besides, we have offered the heuristic mechanism of knowledge extraction about meta-models of XML-models with incomplete formal definition.