Казекин Михаил Михайлович

ИСТОРИЯ ЯЗЫКОВ ПРЕДСТАВЛЕНИЯ ОНТОЛОГИЙ

Аннотация

В статье представлена история языков представления онтологий. Автор описывает основные этапы эволюции, дает краткую характеристику и обзор основных свойств языков. Также в статье представлены некоторые программные инструменты, с помощью которых можно осуществлять редактирование онтологий, записанных посредством одного из языков.

ВВЕДЕНИЕ

Как понятие в области искусственного интеллекта (ИИ) *онтология* возникает в начале 90-х годов в работах специалистов по инженерии знаний. Одно из первых определений, которое было дано этому понятию, было предложено в работе Т. Грубера [1]. Оно звучит так: «Онтология – это явная спецификация концептуализации»[1]. Таким образом, онтология – это формальное описание некоторой концептуальной модели предметной области в явном виде[2]. Мы приведем еще несколько определений:

1) *Онтология* – учение о бытие, в котором исследуются всеобщие основы, принципы бытия, его структура и закономерности (*Философия*).

2) *Онтология* – система понятий, используемых в качестве строительных блоков при построении систем обработки информации (*Системы, основанные на знаниях*) [2].

3) *Онтология* – соглашение[3] (контракт, договор) о разделяемых (shared) концептуальных моделях [1]. Примерами таких моделей могут быть концептуальные решения (framework) для моделирования знаний предметной области, специальные протоколы для коммуникации между взаимодействующими агентами, соглашения о представлении конкретных теорий в различных предметных областях. В самых простых случаях это может быть иерархия типов, определяющая классы и отношения включения между ними. Схемы реляционных баз данных тоже могут являться простейшим примером онтологий, так как они определяют отношения между понятиями, которые существуют в некоторой разделяемой базе данных, и ограничения целостности, которые должны выполняться для этой базы данных.

Заметим, что приведенные выше определения не конструктивны: они не определяют ни методов (методологий), ни меха-

[1] Ontology is an explicit specification of conceptualization.

[2] То есть, в виде, исключающем неявную, подразумеваемую, но не представленную информацию.

[3] Agreement.

низмов (конструкций) для построения онтологий. Поэтому мы дадим более конструктивное определение онтологии, которое устоялось в сообществе специалистов по ИИ:

4) *Онтология* – система, состоящая из понятий[1], отношений между ними[2] и аксиом для формализации понятий и отношений [2].

В соответствии с 4) можно дать следующее определение языка представления онтологий: это некая формальная (структурированная) система для выражения утверждений о понятиях, отношениях между ними и аксиом для формализации вышеупомянутых сущностей.

1. ИСТОРИЧЕСКАЯ ДИАГРАММА

История развития языков представления онтологий (см. историческую диаграмму на рис. 1) показывает[3], что создаваемые языки все больше соответствовали определению 4), фиксируя основные категории рассматриваемых сущностей (понятия, отношения, свойства) и предоставляя все более мощные, четкие и «тонкие» конструкции для концептуализации предметной области. Вместе с тем была сохранена формальность представления и обеспечена возможность логического вывода (reasoning support). В целом развитие языков представления онтологий можно представить как переход *от формы к содержанию*[4] [2]. Под этим переходом понимается концентрация теоретических и практических усилий не на формальных механизмах представления знаний, а на адекватности представления знаний наблюдаемой структуре предметной области и четкой фиксации создаваемого представления. Учитывая, что целью создания онтологии является стремление к созданию *разделяемых* моделей знаний [1], переход полностью соответствует определению «онтология», взятому из философии (рис. 1).

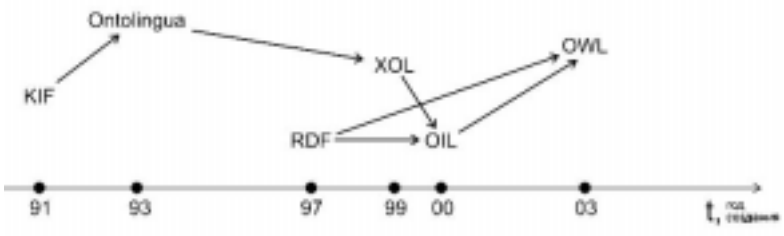

Рис. 1. История развития языков представления онтологий (стрелочками отмечено влияние языков)

2. ЯЗЫКИ ПРЕДСТАВЛЕНИЯ ОНТОЛОГИЙ

LISP-ПОДОБНЫЕ ЯЗЫКИ

В этом разделе рассмотрены два языка представления знаний: KIF и Ontolingua. Оба языка были разработаны Knowledge Systems Laboratory (KSL) отделения информатики Стэндфордского Университета. Исторически сложилось так, что сначала появились LISP-подобные языки, а потом, после появления XML, произошел переход от LISP-подобного представления к представлению в формате XML. Одним из недостатков KIF и построенного на его основе Ontolingua является огромная выразительная мощь без предоставления каких-либо средств для управления этой мощью. Это основная причина, почему для Ontolingua так и не было реализовано ни одного механизма вывода.

• ***KIF***

KIF [3] – компьютерный язык, который предназначен для обмена знаниями между различными программами. Создателями не предполагалось, что KIF будут использовать при взаимодействии людей с компьютерами. Создатели языка подчеркивают, что не предполагается также его использование для внутреннего представления знаний в пре-

[1] Часто представляющая собой иерархическую (отношение «is-a») организацию понятий.
[2] Вдобавок к отношению «is-a» также часто рассматривается «part-of» отношение.
[3] Этому выводу будет посвящена основная часть работы.
[4] Слову «содержание» соответствует англоязычное слово «content».

делах *одной* программы, хотя он также может быть использован и для этих целей. То есть, фактически, создателями KIF декларируется, что этот язык является языком-посредником (interlingua) между программными агентами.

• ***Ontolingua***

Ontolingua [4] позиционируется создателями как язык-посредник (interlingua) для представления онтологий, разработанный лабораторией KSL Стэндфордского университета. Ontolingua является расширением языка KIF, в который были добавлены фреймовые конструкции (в соответствии со стандартом OKBC (Open Knowledge Base Connectivity)) и механизмы перевода в KIF. Ontolingua может быть конвертирован в некоторые языки дескриптивной логики (Description Logics, DL), такие как Loom, Epikit и др. Ontolingua сам по себе не предоставляет возможности логического вывода. Впоследствии лабораторией KSL была разработана программная среда (в виде веб-интерфейса), которая позволяет просматривать, создавать, редактировать онтологии, и библиотека модульных и предназначенных для повторного использования онтологий.

XML-ПОДОБНЫЕ ЯЗЫКИ

Ключевые моменты развития языков представления онтологий на основе XML таковы:

1. Появляется язык RDF [5, 6, 7], основанный на XML. RDF выражает триплетную модель Subject – Predicate – Object (Resource – Property – Property Value)[1].

2. Появляется RDF Schema [8], которая является словарем, содержащим термины для описания и расширения RDF.

3. Разрабатывается XOL [9], язык, совместимый с OKBC и предлагающий фреймовый подход к представлению знаний.

4. Разрабатывается OIL (Ontology Inference Layer) [10], который построен на основе XOL и RDF Schema, а также использует элементы дескриптивной логики (Description Logics), которые призваны обеспечить формальную семантику и поддержку логического вывода (reasoning support).

5. На основе OIL группой проекта DAML организации DARPA создается язык семантической разметки DAML + OIL [11][2], который впоследствии стал доступен как набор документов, опубликованных консорциумом W3C. Этот язык обеспечивает намного более «тонкое» моделирование, чем RDF. Также аббревиатурой DAML обозначается набор (библиотека) онтологий, насчитывающий более 250 экземпляров.

6. Одной из последних инициатив W3C, которая завершает обзор языков, основанных на XML, является OWL (Web Ontology Language) [12], нацеленная на использование в проекте Semantic Web. OWL является переработанной версией DAML + OIL, в которой были учтены различные потребности, предъявляемые к мощности языка моделирования.

• ***RDF & RDF Schema***

RDF представляет собой технологическое решение (framework) для представления информации в World Wide Web.

Любое выражение RDF основано на коллекции так называемых триплетов, каждый из которых состоит из субъекта (subject), предиката (predicate) и объекта (object), что может быть изображено графически следующим образом (рис. 2).

Каждый триплет представляет собой утверждение о связи между понятиями (субъект и объект), обозначенными как узлы, которые эта связь соединяет. Направление связи важно: оно всегда указывает на объект.

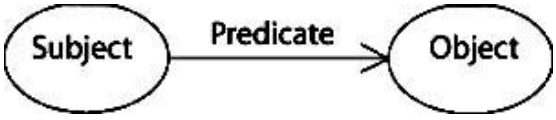

Рис. 2

[1] Также можно встретить название «Subject-Verb-Object» (SVO) – Подлежащее – Сказуемое – Дополнение.

[2] Не присутствует на диаграмме из-за схожести с прямым потомком – OWL.

Таким образом, RDF-граф образует семантическую сеть. Значением RDF-графа является логическое «И» всех утверждений-триплетов, формирующих этот граф.

Таким образом, RDF предоставляет модель (framework) для аннотации ресурсов Сети при помощи XML-синтаксиса и триплетной модели (Subject, Predicate, Object), которая имеет наглядное графическое представление (ориентированный размеченный граф).

RDF Schema является языком описания пользовательских словарей (наборов понятий) при помощи стандартизированного набора тегов в формате RDF-триплетов, который и составляет основу RDF Schema. Для простоты можно считать, что RDF Schema предоставляет систему типов для RDF. Сам по себе RDF не предоставляет никаких механизмов ни для описания свойств, ни для описания отношений между этими свойствами и другими ресурсами. Эту роль играет RDF Schema, который определяет понятия «класс», «свойство» и некоторые другие, позволяющие описывать классы, свойства и другие ресурсы.

• ***XOL***

Язык XOL (XML-based ontology exchange language) предназначен для обмена описаниями онтологий между различными агентами. XOL был одним из первых XML-языков, предназначенных для представления онтологий.

XOL предоставляет простой и наглядный XML-формат для фреймового представления в соответствии со стандартом OKBC для разработки фреймовых систем представления знаний.

• ***OIL***

Ontology Inference Layer [3] – это язык представления онтологий, который сочетает в себе:

а) широко используемый набор примитивов, заимствованный из фреймовых языков представления знаний (в основе языка лежит XOL),

б) формальную семантику и механизм вывода в рамках дескриптивной логики (Description Logics, DL),

в) инфраструктуру RDF(S), которая обеспечивает стандартизированное представление (синтаксис и набор примитивов).

• ***OWL***

OWL (Web Ontology Language) является ревизией DAML + OIL, и был разработан для использования приложениями, которым требуется обрабатывать информацию, а не просто предоставлять ее агентам. OWL предлагает большую способность к взаимодействию (interoperability), чем RDF(S), благодаря наличию дополнительного словаря вместе с поддержкой формальной семантики. У OWL есть три (по возрастанию выразительной мощности) подмножества: *OWL Lite*, *OWL DL* и *OWL Full*. OWL Lite содержит основные конструкции для описания классов и задания простых ограничений, OWL DL предлагает максимум возможностей, оставаясь при этом в рамках дескриптивной логики (все заключения являются вычислимыми и завершаются за конечное время). OWL Full не гарантирует возможности существования вычислительной поддержки и рассчитан на пользователей, которые хотят получить максимальную выразительную мощность и свободу от RDF.

1) *OWL Lite*

OWL Lite удовлетворяет основные потребности пользователей, которым нужна классификация (иерархия) и простые ограничения. Например, OWL Lite поддерживает ограничение на кардинальность, но с его помощью можно только задавать значения 0 или 1. Таким образом, ограничения OWL Lite проявляются не в мощности словаря, а в ограничениях на его элементы. Предполагается, что для OWL Lite разработка инструментария будет проще, чем для его более выразительных надмножеств, и OWL Lite предоставляет хорошую возможность по «миграции» тезаурусов и других таксономий. Вывод в OWL Lite также имеет меньшую вычислительную сложность, чем в OWL DL.

2) *OWL DL & OWL Full*

OWL DL и OWL Full используют один и тот же словарь, хотя OWL DL подчинен некоторым ограничениям. Грубо говоря, в

OWL DL класс не может также быть индивидом или свойством, и свойство не может быть также индивидом или классом. Таким образом, ограничения не могут налагаться на сами языковые элементы OWL.

Язык OWL предоставляет большой и выразительный словарь, позволяющий строить и адекватные модели знаний из различных предметных областей и верифицировать их. По мнению автора, недостатком языка (который был унаследован еще от RDF формата) является тесное сплетение модели и метамодели. Также хочется отметить, что следование триплетной модели приводит к значительному объему XML-представления, что увеличивает сложность разработки программных приложений, использующих OWL. Тем не менее, в настоящий момент OWL является общепризнанным стандартом для представления онтологий.

3. ИНСТРУМЕНТЫ ДЛЯ РАБОТЫ С ОНТОЛОГИЯМИ

• ***Protégé***

Один из самых популярных инструментов для работы с онтологиями Protégé (*http://protege.stanford.edu/*) был создан лабораторией KSL Стендфордского университета. Данный инструмент поддерживает два стандарта: разработанный в той же лаборатории OKBC, предлагающий фреймы в качестве способа представления знаний, и язык OWL.

Давайте рассмотрим интерфейс Protégé-OKBC подробнее (интерфейс Protégé-OWL повторяет рассматриваемый, за исключением элементов, специфических для OWL). Взаимодействие с Protégé (рис. 3) предполагает использование пяти окон, оформленных в виде закладок: Classes, Slots, Forms, Instances, Queries. Вверху располагается панель управления проектом. С ее помощью можно создавать модели, сохранять/загружать/экспортировать их и т. д. Там же располагается традиционное меню, которое содержит все основные функции управления программой.

Classes

Слева присутствует панель браузера иерархии классов (отношение subclass-of) (рис. 3). Посредством этой панели можно изменять иерархию, добавляя или удаляя классы. Справа от нее располагается панель Class Editor, в которой можно редактировать выбранный в браузере класс: описывать ограничения, слоты, роль класса (абстрактный, реально существующий) и создавать документацию к нему.

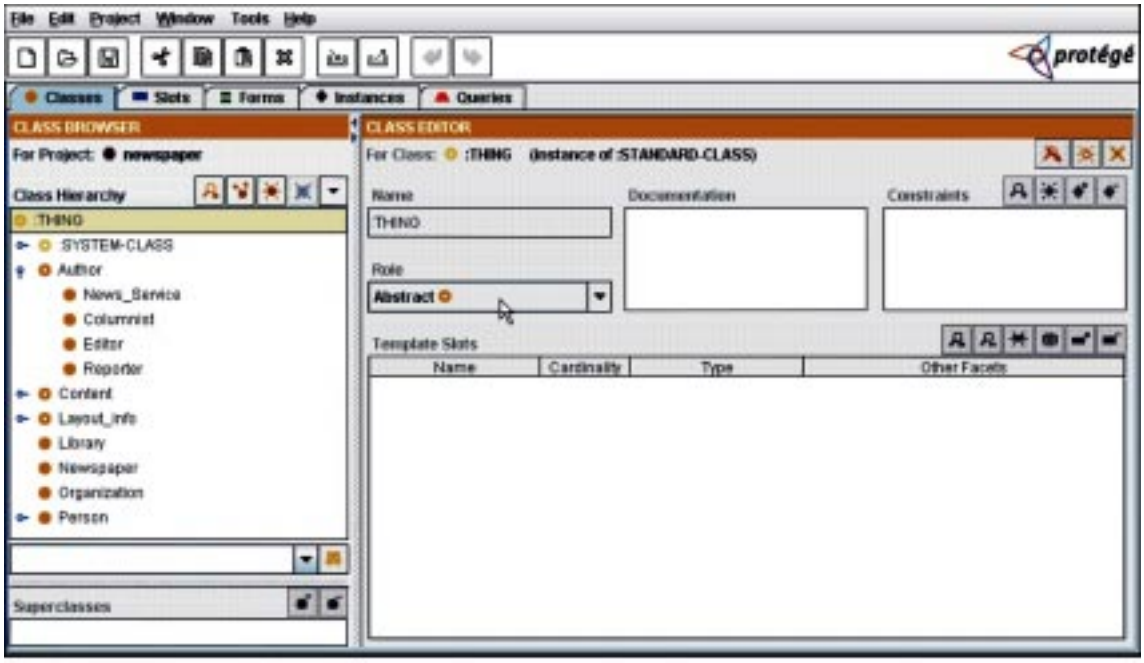

Рис. 3

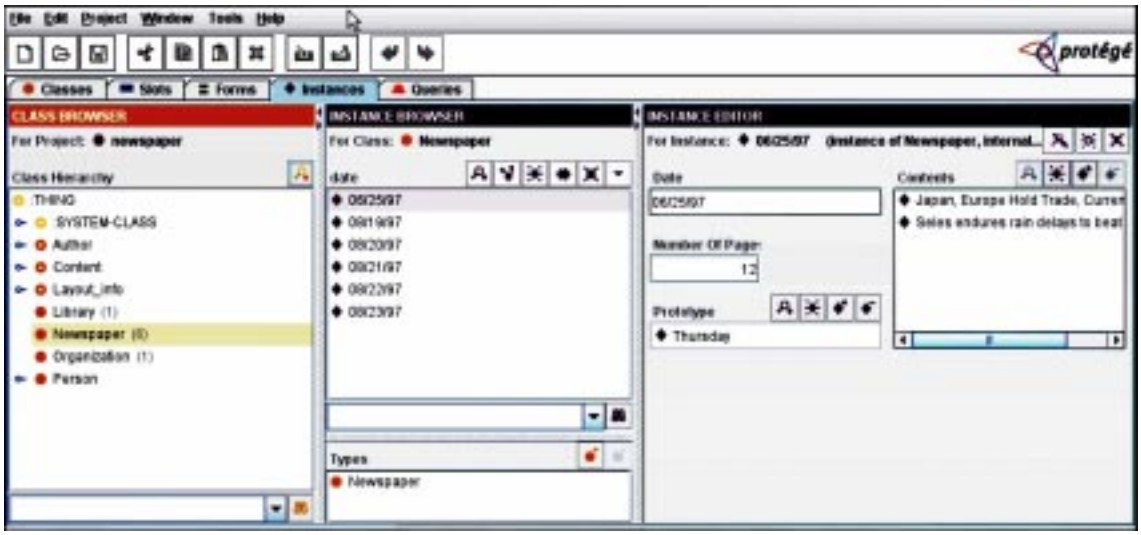

Рис. 4

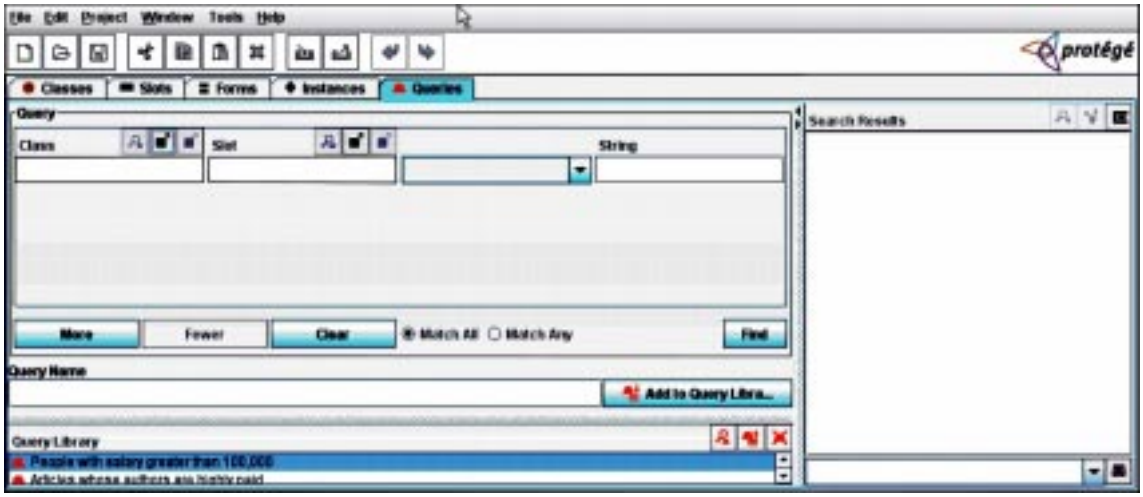

Рис. 5

Slots

По структуре вкладка Slots повторяет вкладку Classes. Слева расположен браузер слотов Slot Browser с иерархией слотов, а справа – редактор слота Slot Editor, в котором можно редактировать различные ограничения для выбранного в браузере слота: кардинальные ограничения, минимальное/максимальное значения, значения по умолчанию, домен слота.

Forms

Вкладка Forms предназначена для создания и редактирования форм ввода данных в модель. По структуре она также не отличается от предыдущих двух вкладок. С ее помощью можно создавать и редактировать формы, осуществляя привязку к создаваемой модели знаний.

Instances

Вкладка Instances (рис. 4) содержит уже знакомый браузер классов, браузер экземпляров (Instance Browser) и редактор экземпляров (Instance Editor). С ее помощью можно вводить данные посредством форм, созданных при помощи вкладки Forms, которые отображаются на панели Instance Editor при выборе класса и его экземпляра (посредством Class Browser и Instance Browser соответственно).

Queries

Последняя вкладка Queries (рис. 5), содержит инструменты для создания запро-

сов к наполненной базе знаний и выполнения уже созданных запросов (хранящихся в библиотеке Query Library). Слева расположена форма для редактирования запроса, справа отображаются результаты запросов.

В заключение обзора редактора Protégé отметим, что Protégé предоставляет удобные средства для поддержки модели знаний на всех этапах – от ее создания, до наполнения реальными данными и предоставления механизмов поиска по этим данным.

• ***OntologyEditor (Hozo)***

Второй рассматриваемый инструмент, HoZo (*http://www.hozo.jp/*), разработанный в Институте Научных и Промышленных Исследований (Университет г. Осака), также опирается на фреймовое представление данных, но, в отличие от Protégé, предлагает собственную метамодель, в которой присутствует понятие роли и ролевого свойства. В соответствии с этой моделью и организован пользовательский интерфейс редактора. В Hozo поддерживается экспорт в разные языки, в том числе в RDF(S), DAML + OIL, OWL.

В верхней части окна редактора располагается традиционное меню, содержащее основные функциональные возможности и привычные операции. Основная панель Editor Panel (рис. 6) располагается по центру приложения и разделена на три части: навигатор понятий (слева сверху), браузер свойств (слева внизу), и браузер понятий (область справа), не считая вкладки RDF(S), которая предполагает «моментальный снимок» модели в формате RDF(S). Рассмотрим основные части подробнее.

Навигатор понятий

С помощью навигатора понятий (при помощи двух закладок, WC-tree и RC-tree) осуществляется навигация по двум иерархиям, предлагаемым существующей в Hozo метамодельюю: иерархия «понятие-целое» (wholeness concept) и иерархия «понятие-отношение» (relation concept). Опираясь на то, что объекты состоят из частей, между которыми устанавливаются определенные

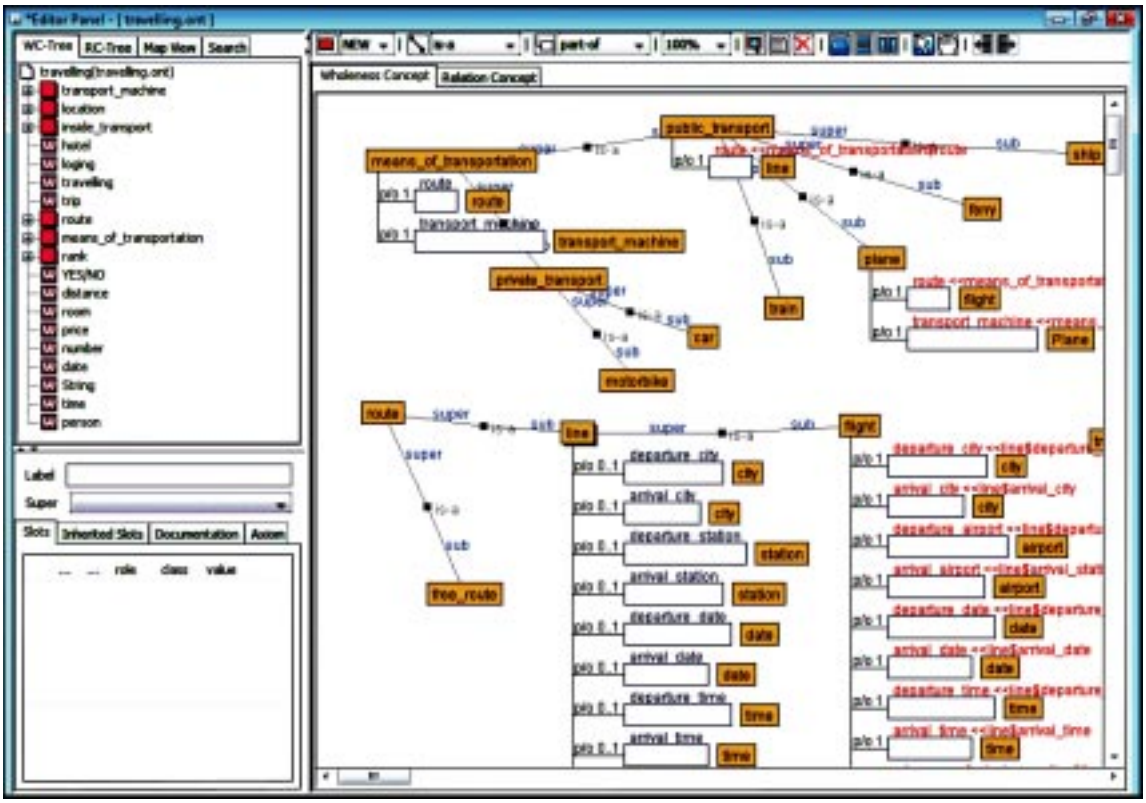

Рис. 6

отношения, авторы редактора (и метамодели) предложили описывать понятие с позиции объекта и его составляющих (wholeness concept tree) и с позиции устанавливаемых отношений (relation concept tree). В соответствии с таким подходом и навигатор понятий, и браузер понятий предлагают раздельное редактирование и просмотр соответствующих иерархий. Также с помощью навигатора понятий предлагают обзор иерархии «с высоты птичьего полета» (Map View) и текстовый поиск.

Браузер свойств

Браузер свойств предлагает просмотр и редактирование различных свойств для выбранных в навигаторе/браузере объектов (понятий и слотов), а также запись аксиом (ограничений на объекты) и составление документации к выбранному объекту. Так, с помощью браузера свойств можно задать значения для трех параметров, role concept (имя роли), role holder (носитель роли) и class constraint (ограничение на класс), с помощью которых разработчики моделируют концепцию «роль» в метамодели. Поясним, что потребность в концепции «роль» возникла из-за трудностей в адекватном моделировании таких отношений между понятиями, как, например, husband is-a person. Когда брак распадается, человек перестает быть мужем (это его роль), но не перестает быть человеком. Критерий идентичности, очевидно, нарушен (см. теоретическую часть настоящей работы). В данном примере разумно role concept присвоить значение «husband role», role holder присвоить «husband», a class constraint (к какому базовому классу принадлежит рассматриваемое понятие) назначить «person».

Браузер понятий

В браузере понятий осуществляется основной (визуальный) процесс редактирования онтологий. В верхней части браузера находится панель инструментов, на которой расположены инструменты создания понятий, добавления слотов (свойств) и установления различных отношений. Кроме общих для обеих вкладок кнопок «задать отношение is-a» и «задать отношение attribute-of», на панели инструментов располагаются инструменты «задать отношение part-of» (p/o для WC-tree) и «задать отношение participates-in» (p/i для RC-tree), в зависимости от вкладок Wholeness/Relation. Визуально part-of/participates-in выражены одним инструментом «part-of», присутствующим на обеих вкладках.

Отметим, что редактор Hozo существенно – и в положительную сторону – отличается от других известных автору редакторов наличием грамотной и продуманной метамодели, которая учитывает тонкости проектирования онтологий, и адекватной визуализацией этой метамодели.

ЗАКЛЮЧЕНИЕ

Если говорить о тенденциях, которые наблюдаются с момента появления первых подобных языков, то можно проследить переход от LISP-подобного представления модели к XML-представлению (см. рис. 7).

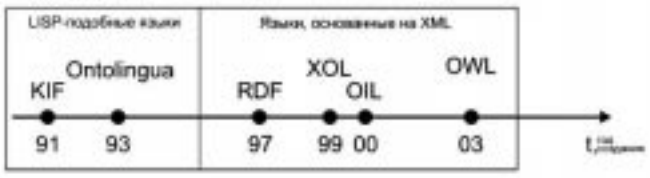

Рис. 7

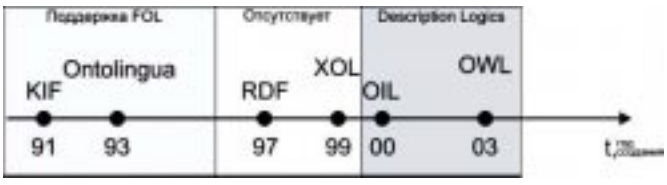

Рис. 8

При этом в целом этот переход происходил в рамках существующего фреймового (frame-based) подхода к представлению знаний (рис. 8).

В первых языках имелась возможность работы в рамках логики предикатов первого порядка (First Order Logics, FOL), а в более поздних – возможность моделирова-

ния в границах класса языков дескриптивной логики (Description Logics) (см. рис. 9).

Языки этого класса были созданы в качестве расширения таких инструментов представления знаний как фреймы и семантические сети, не снабженных формальными, основанными на логике семантическими конструкциями.

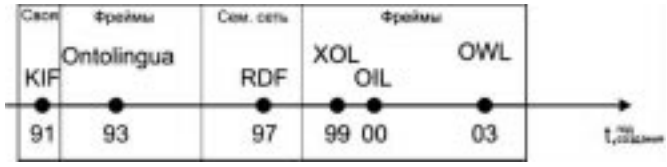

Рис. 9

Литература

1. *T.R. Gruber.* Toward principles for the design of ontologies used for knowledge sharing. *International Workshop on Formal Ontology*, Padova, Italy, 1992.

2. *R. Mizoguchi.* Tutorial on ontological engineering, *NEW GENERATION COMPUTING-TOKYO*, 2003. ISIR, Osaka University. *http://www.ei.sanken.osaka-u.ac.jp*

3. *M.R. Genesereth*, *R.E. Fikes*. Knowledge Interchange Format Reference Manual, *Technical Report Logic-92-1*, 1992. DARPA Knowledge Sharing Effort, CS Department, Stanford University.

4. *T.R. Gruber.* Ontolingua: A Mechanism to Support Portable Ontologies, *Technical Report KSL 91-66*, 1992. Knowledge Systems Laboratory, Stanford University.

5. *D. Beckett (ed).* RDF/XML Syntax Specification, *World Wide Web Consortium Recommendation*, 2004. *http://www.w3.org/TR/rdf-syntax-grammar/*

6. *F. Manola*, *E. Miller. (eds)* RDF Primer, *World Wide Web Consortium Recommendation*, 2004. *http://www.w3.org/TR/2004/REC-rdf-primer-20040210/*

7. *D. Beckett (ed).* RDF/XML Syntax Specification, *World Wide Web Consortium Recommendation*, 2004. *http://www.w3.org/TR/rdf-syntax-grammar/*

8. *D. Brickley (ed*). RDF Schema Specification, *World Wide Web Consortium Recommendation*, 2004. *http://www.w3.org/TR/rdf-schema/*

9. *P.D. Karp*, *V.K. Chaudhri*, *J. Thomere*. XOL: An XML-Based Ontology Exchange Language. Version 0.4, 1999.

10. *D. Fensel*, *I. Horrocks et al*. OIL In a Nutshell, *LECTURE NOTES IN COMPUTER SCIENCE, Springer-Verlag*, 2000.

11. *D. Connolly*, *F. van Harmelen et al*. DAML + OIL Reference Description, *World Wide Web Consortium Recommendation, 2001. http://www.w3.org/TR/daml+oil-reference*

12. *D.L. McGuinness and Frank van Harmelen. (eds)* OWL Web Ontology Language Overview, *World Wide Web Consortium Recommendation*, 2004. *http://www.w3.org/TR/owl-features/*

Abstract

The article presents the history of ontology representation languages. The author describes the main stages of evolution, gives a brief description and overview of the basic properties of languages. The article also presents some software tools that one can use to edit ontologies, represented by one of the languages.

Казекин Михаил Михайлович,
аспирант математико-механического факультета СПбГУ,
michael.kazekin@gmail.com