Серия «Математика»
2010. Т. 3, № 4. С. 65—79

Онлайн-доступ к журналу:
http://isu.ru/izvestia

ИЗВЕСТИЯ
Иркутского
государственного
университета

УДК 518.517

Объектные модели и распределенные системы знаний

А. В. Манцивода*
Иркутский государственный университет

Н. О. Стукушин
Иркутский государственный университет

Аннотация. В статье рассматриваются вопросы использования дескриптивных логик для построения моделей метаописаний. Предлагается метод построения таких моделей в рамках дескриптивной логики OO–проекций. Данный метод иллюстрируется на примере классической системы метаописаний – дублинского ядра.

Ключевые слова: онтологии; метаописание; дублинское ядро; язык запросов Libretto; спецификация.

1. Единое адресное пространство

При разработке формализованных систем знаний, ориентированных на определенную предметную область, возникает комплекс взаимоувязанных задач, которые, в частности, могут включать в себя:

1) Моделирование предметной области;
2) Манипулирование данными модели (создание баз данных/знаний, обновление информации и т.д.);
3) Аналитическую обработку данных с помощью «интеллектуализированных» методов (логический вывод);
4) Взаимодействие с внешним миром через программные и пользовательские интерфейсы,

и так далее. В любой достаточно сложной формализованной системе знаний перечисленные компоненты взаимодействуют в тех или иных сочетаниях. Одной из основных проблем является необходимость исполь-

* Работа выполнена при частичной финансовой поддержке Федеральной целевой программы «Научные и научно-педагогические кадры инновационной России» на 2009-2013 гг., государственный контракт No. 01201062602

зования разных технологий разработки для реализации каждой из этих компонент. И дело здесь не только в том, что приходится использовать несколько систем разработки. Структуры данных в этих технологиях, как правило, мало совместимы. Классический пример на информационном уровне – объектно ориентированные языки программирования и реляционные базы данных. Объектные модели весьма выразительны и удобны для работы в большинстве случаев, например, для описания предметных областей в образовательных системах. Поэтому объектно-ориентированные языки захватили командные высоты как основное средство разработки. С другой стороны, для хранения данных используется принципиально иная модель данных – реляционная. Таблицы баз данных намного более примитивны, чем объектные модели, но имеют значительные преимущества в организации долговременного хранения данных. При разработке сложных систем это приводит к одному – постоянной заботой разработчика становится перекачка объектных моделей в реляционные хранилища и обратно. Работа тонкая, требующая высокой квалификации и существенных дополнительных затрат.

В более широком контексте данная проблема формулируется как отсутствие единого адресного пространства, то есть:

1) Структуры данных в разных компонентах системы имеют разный и часто несовместимый формат;
2) Разработчику приходится явно управлять механизмами хранения данных, которые также могут быть совершенно различными.

Еще более остро проблема единого пространства стоит в системах обработки знаний. Имеется целый комплекс базовых понятий и концептов, о формализованном определении которых сообщество могло бы договориться, и совместно использовать всеми признаваемые определения. К сожалению, такие договоренности удается достичь только на синтаксическом (например, в рамках стандарта языка разметки XML) и презентационном (HTML, LaTeXи т.д.) уровнях. С семантикой дело обстоит намного хуже. Целый ряд попыток решить эту фундаментальную проблему так и не достиг желаемой цели. Наиболее амбициозным проектом в этом направлении, безусловно, стал семантический веб [13]. В своей концептуальной работе [5] Тим Бернерс-Ли рассматривал сематический веб как механизм, способный изменить ситуацию в мировом информационном пространстве в глобальных масштабах. В качестве ключевых инструментов реализации этой идеи рассматривались стандартизированные языки описания семантики RDF [12] и OWL [16] (аналогично тому, как XML является стандартизированным средством структурных описаний). К сожалению, ряд существенных промахов оставил данному подходу немного шансов на успешное продвижение.

Проблема 1. Тяжелейшим просчетом организаторов проекта стала низкая квалификация разработчиков базового языка описания ресур-

сов RDF. В результате стандартизированная в 1999 году спецификация RDF содержала элементарные логические ошибки (которые пришлось спешно править в процессе принятия намного более продуманного языка OWL). Совершенно не были продуманы алгоритмические вопросы, связанные с RDF (ряд базовых компонент RDF оказались просто алгоритмически неразрешимыми – поразительная небрежность со стороны людей, пытающихся автоматизировать соответствующие процессы). Наконец, RDF имеет крайне неудачный синтаксис: семантически значимая информация в описаниях RDF занимает 5-10% и тонет в структурах собственно языка.

Проблема 2. Для того, чтобы технология получила импульс к продвижению, она должна сопровождаться качественным программным обеспечением. К сожалению, основная масса разработок так и осталась на любительском и экспериментальном уровнях. Среди редких исключений следует отметить систему Protege [10], однако ее целевой аудиторией являются не разработчики информационных ресурсов, а скорее исследователи, работающие в определенных областях, например, биологии. Отсутствие значимых коммерческих разработок – еще один косвенный признак, свидетельствующий об отсутствии внешнего интереса к данному направлению.

Проблема 3. Для того, чтобы метод нашел широкое применение, был воспринят массовым разработчиком, он должен иметь низкий порог «входа», чтобы с методом начали работать. Общий уровень также должен соответствовать возможностям массового пользователя. Поскольку семантический веб основан на абстрактных и элитарных средствах математической логики, необходимы недюжинные усилия для обеспечения доступности предлагаемых средств широким кругам разработчиков информационных ресурсов.

Проблема 4. Высокая сложность методов семантического веба сочетается с отсутствием мотивации у пользователей в применении технологии. В этом направлении работа фактически не велась. Получается, что обычному разработчику не только сложно работать в данной технике, но и не понятно зачем.

Таким образом, хотя в настоящее время по семантическому вебу проводится большое количество международных конференций, работает значительное число исследовательских групп, защищается множество диссертаций, за десять лет, прошедших с момента опубликования «манифеста» семантического веба, ключевая цель данного направления так и не была достигнута, парадигма постепенно изменилась, исследования страдают мелкотемьем, потерей общего вектора развития. Существует большая опасность маргинализации всей области исследований. С нашей точки зрения, шансы столь многообещающего проекта были упущены.

С учетом критики, представленной выше, нами разработан подход к задаче представления знаний в распределенных информационных средах, направленный на решение перечисленных выше проблем. Данный подход реализуется на технологической основе системы Libretto, включающей одноименный язык, модуль управления объектными моделями Libretto Engine и пользовательский интерфейс для разработки программ и объектных моделей Libretto Notes. Наш подход основан на следующих принципах:

Принцип 1. Для описания предметных областей используются объектные модели – более простые и привычные массовому разработчику, чем модели логические. На самом деле нами предлагается значительно более гибкое решение, чем просто использование объектного моделирования. Благодаря результатам, представленным в [2], построены формальные отображения из логического аналога объектных моделей (простых дескриптивных логик, называемых OO-проекциями) как в произвольные дескриптивные логики (что обеспечивает использование объектных описаний в интеллектуализированных логических системах), так и в системы типов данных объектно-ориентированных языков программирования, а также реляционные модели баз данных. Эти результаты обеспечивают теоретическую и технологическую основу для построения единого пространства, и могут играть роль интеграторов.

Принцип 2. Формализованные описания предметных областей разрабатываются на языке (Libretto) – логическом языке, обладающем процедурной семантикой, что превращает их в исполняемый код, который может непосредственно использоваться при реализации информационных систем. Это важнейшее качество позволяет существенно удешевлять процесс разработки за счет исключения этапа реализации абстрактных компонент описаний, поскольку сами описания являются исполняемыми. Здесь решается еще одна проблема – промежуточные реализации часто по-разному интерпретируют описания, спецификации, стандарты, что приводит к проблемам несовместимости. Если же само описание непосредственно готово к применению, то эта сложность в значительной степени облегчается, если не исчезает полностью.

Принцип 3. Разработанные формальные описания через специальную среду доступны для распространения и повторного использования («реюзабилити») – как в «ручном» режиме, так и через программные интерфейсы. Заметим, что и сегодня объектные модели очень активно используются в качестве абстрактных описаний в рамках разнообразных спецификаций (достаточно упомянуть объектную модель документа, DOM). Но эти модели не исполняемые – для их использования в программных системах как правило разрабатываются специальные модули, на практике реализующие абстрактные модели спецификаций. Такие реализации могут повторяться множество раз, например, для таких спецификаций, как ISO 639 [8] и ISO 3166 [9]. Наша идея состо-

ит в том, что спецификации могут сопровождаться «каноническими» объектными моделями, непосредственно приспособленными для исполнения и включения в код в качестве готовых модулей и библиотек. Такие логические модели могут быть как частью самой спецификации, так и создаваться и одобряться соответствующим сообществом разработчиков.

Заметим, что создание среды обмена объектными моделями является довольно естественной идеей. Однако до сих пор такая среда не была создана. Одной из причин этого является отсутствие стандартизированного метода, в котором данная среда может функционировать. Очевидно, что единство адресного пространства является существенным требованием к данной среде, поскольку объектные модели могут работать и в рамках исполняемого кода, и для хранения структур данных. С нашей точки зрения язык Libretto вполне подходит для реализации подобной среды, поскольку позволяет организовать единое пространство и имеет строгую логическую семантику, необходимую для восприятия объектных моделей как формально корректных и однозначно понимаемых спецификаций. Для реализации принципа 3 мы воспользуемся методом LibrettoNetwork, разработанным в нашей исследовательской группе. Этот метод состоит в следующем. Объектные модели и их компоненты распространяются в виде онтологий, упакованных в специальном xml-формате (т.н. mvx-формате). Ключевым свойством здесь является то, что онтологии, в которых формируются объектные модели, имеют глобально уникальные имена (URI). Поиск онтологий осуществляется автоматически по их уникальным именам. Онтологии публикуются и хранятся на специальных серверах/репозиториях. Разработчику достаточно указать глобальное имя онтологии в команде `require` языка Libretto для того, чтобы она была автоматически импортирована из репозитория. Механизм require позволяет организовать распределенную сеть объектных моделей, которые могут автоматически использоваться (импортироваться) информационными системами.

Принцип 4. Организация единого семантического пространства. Здесь идея состоит в следующем. Если в процессе описания предметной области используется информация, для которой стандартизированы форматы представления (например, спецификация Dublin Core для метаданных, KML для представления трехмерных геопространственных данных, IMS QTI для представления образовательных тестов и т.д.), и эти спецификации обладают «каноническими» логическими моделями, то эти модели могут «импортироваться» и напрямую использоваться в процессе описания предметной области. Это очень похоже на операцию наследования в объектно-ориентированном подходе. Если эти канонические модели глобализованы (например, включены в соответствующие спецификации или признаются международны сообществом), то их использование при описании знаний глобализует и само это описание.

Например, в той части описания, которая основана на этих спецификациях, могут действовать стандартизированные обработчики, умеющие работать с объектными моделями спецификаций.

В данной работе на основе изложенных выше принципов исследуется возможность построения такого рода описаний. В основе нашего подхода лежит понятие онтологии как независимого модуля описния знаний, который может отчуждаться от своего разработчика, распространяться и многократно использоваться в разнообразных контекстах. В онтологиях языка Libretto информация описывается с помощью объектных моделей.

В качестве примера базовой спецификации нами будет использоваться Dublin Core (дублинское ядро, DC) [15], задающее правила построения метаописаний информационных ресурсов. Использование базовой модели будет продемонстрировано на такой предметной области как описание паспортов специальностей ВАК [3]. Благодаря стандартизированному подходу автоматически генерирутся соответствующий информационный ресурс.

2. Дублинское ядро

Дадим более подробную характеристику дублинского ядра. Дублинское ядро (Dublin Core, DC) [15] можно понимать как компактный язык для разработки формализованных утверждений определенных типов, описывающих ресурсы. В этом языке имеется два класса терминов – элементы («существительные») и квалификаторы («прилагательные»). Существительные и прилагательные в этом языке формируют простые предложения, предметом описания которых являются ресурсы. В многообразном информационном мире DC является «простейшим всеобщим языком для виртуальных путешественников» [15], на котором могут общаться почти все. И хотя он не всегда обеспечивает глубокое описание сложных концептов, зато он легок в понимании.

Структурно дублинское ядро состоит из двух уровней:

1) Simple Dublin Core – простой уровень, который еще называется неквалифицированным;
2) Qualified Dublin Core – квалифицированный уровень.

Простой уровень состоит из 15 базовых элементов, квалифицированный уровень содержит еще один элемент – Audience, а также группу квалификаторов (атрибутов) базовых элементов, позволяющих уточнять информацию из базовых элементов метаописаний.

Каждый элемент DC является опциональным (необязательным) и может повторяться в метаописании сколько угодно раз. Кроме того, порядок элементов в метаописании также никак не регулируется. Порядок, в котором несколько экземпляров одного и того же элемента

(например, Creator – создатель) входят в метаописание, может иметь значение в некоторых системах. Однако не гарантируется, что этот порядок будет сохраняться в других реализациях. Упорядочение зависит от синтаксиса, в котором реализуется дублинское ядро. Например, в синтаксисе RDF/XML порядок поддерживается, а в HTML нет.

3. OO-проекции

Базой для построения логической модели метаописаний будет служить OO-проекция – дескриптивная логика, с помощью которой логически формализуется процесс построения объектных моделей. OO-проекция является подлогикой значительно более выразительной логики $\mathcal{SHOIN}(D)$ [7], которая лежит в основе языка описания онтологий OWL-DL [16]. Это обеспечивает совместимость подхода, разрабатываемого нами, с базовыми конструкциями семантического веба.

Пусть

$$\mathcal{D} = \langle D_1, \ldots, D_k; \Omega \rangle$$

некоторая область данных. *Словарь* дескриптивной логики включает символы концептов, ролей, атрибутов и объектов и представляет собой четверку $\langle \mathcal{C}, \mathcal{R}, \mathcal{P}, \mathcal{O} \rangle$, где $\mathcal{C}$ – множество именованных концептов, $\mathcal{R}$ – множество ролей, $\mathcal{P}$ – множество атрибутов, $\mathcal{O}$ – множество имен объектов. Область данных $\mathcal{D}$ считается фиксированной, поэтому ее компоненты в словари не входят.

Определение 1 (Язык $\mathcal{SHOIN}(D)$)**.** *Если $c \in \mathcal{C}$, то c называется примитивным концептом $\mathcal{SHOIN}(D)$. Примитивные концепты являются концептами. Если a, b – концепты, $id \in \mathcal{O}$, $r \in \mathcal{R} \cup \mathcal{P}$, $p \in \Omega$, то $a \sqcap b$, $a \sqcup b$, $\neg a$, $\exists r.a$, $\forall r.a$, $\leqslant_n r$, $\geqslant_n r$, $\exists(x_1, \ldots, x_n).p$, $\forall(x_1, \ldots, x_n).p$, $\{id\}$ также концепты. Если $r \in \mathcal{R}$, то $\exists r'.a$ и $\forall r'.a$ – концепты, где $r' \in \{r^+, r^-, r^\pm\}$.*

Понятие OO-проекции введено в работе [2]. OO-проекции можно также рассматривать как модификацию и расширение простой дескриптивной логики $\mathcal{FL}_0$ [14]. Для построения OO-проекций вводится оператор типизации θ, который типизирует роли и атрибуты:

1) привязывает роли $R \in \mathcal{R}$ к парам $\theta(R) = [C_d, C_r]$, где $C_d, C_r \in \mathcal{C}$;
2) привязывает атрибуты P к парам $\theta(P) = [C_d, D_r]$, где $C_d \in \mathcal{C}$ и $D_r \in \mathcal{D}$.

C_d называется областью определения роли R (атрибута P), C_r (D_r) называется областью значений роли R (атрибута P).

Интерпретация I считается согласованной с θ, если

$$\forall R \in \mathcal{R}.(\theta(R) = [C_d, C_r] \;\rightarrow\; R^I \subseteq C_d^I \times C_r^I)$$
$$\forall P \in \mathcal{P}.(\theta(P) = [C_d, D_r] \;\rightarrow\; P^I \subseteq C_d^I \times D_r^I)$$

Интерпретация I согласована со словарем $\langle \mathcal{C}, \mathcal{R}, \mathcal{P}, \mathcal{O} \rangle$, если для любого объекта $o \in \top^I$ существует имя $O \in \mathcal{O}$ такое, что $o = O^I$ (т.е. все объекты выделенные). В дальнейшем будем рассматривать только интерпретации, согласованные с θ и словарем. $\mathcal{K} \models E$ будет обозначать, что E истинна в любой согласованной интерпретации $\mathcal{K}$. В отличие от этого $\mathcal{K} \vdash E$ означает, что E выводима из $\mathcal{K}$ в некотором полном исчислении (например, с помощью табличных алгоритмов), а значит, истинна в любой интерпретации.

Пусть $\mathcal{W} = \langle \mathcal{C}, \mathcal{R}, \mathcal{P}, \mathcal{O} \rangle$ – словарь.

Определение 2 (ОО-концепт)**.** *Концепт вида*

$$C_1 \sqcap \ldots \sqcap C_f \sqcap \forall R_1.E_1 \sqcap \ldots \sqcap \forall R_g.E_g \sqcap \forall P_1.D_1 \sqcap \ldots \sqcap \forall P_h.D_h$$

называется ОО-концептом, где E_i – ОО-концепты, $C, C_i \in \mathcal{C}$, $R_i \in \mathcal{R}$, $P_i \in \mathcal{P}$ и $D_i \in \mathcal{D}$. Если все $E_i \in \mathcal{C}$, то ОО-концепт называется примитивным.

Определение 3 (ОО-определение)**.** *Аксиома*

$$C \equiv E \tag{3.1}$$

где E – примитивный ОО-концепт, называется объектно-ориентированным определением (ОО-определением).

Говорим, что концепт C в ОО-определении *непосредственно наследует* от $C_1, \ldots, C_n$. Пусть $\mathcal{A} = \{C_i \equiv E_i\}$ – множество ОО-определений. C_i *наследует* от C_j, если он непосредственно наследует от C_j, или существует C_k такой, что C_i наследует от C_k и C_k наследует от C_j. $\mathcal{A}$ называется ацикличным, если оно не содержит C_i, наследующих от самих себя.

Схожим образом, если концепт C определен с помощью ОО-определения в множестве ОО-определений $\mathcal{A}$, то мы говорим, что концепт C *непосредственно зависит от* ролей $R_1, \ldots, R_k$ и атрибутов $P_1, \ldots, P_l$. C *зависит* от роли R (атрибута P), если он непосредственно зависит от роли (атрибута), или существует C' в $\mathcal{A}$, который непосредственно зависит от R (P), и C наследует от C'.

Пусть $\mathcal{L}$ – дескриптивная логика, содержащая $\mathcal{SHOIN}(D)$ как подлогику, и $\mathcal{K} = TBox_{\mathcal{K}} \cup ABox_{\mathcal{K}}$ – $\mathcal{L}$–описание предметной области в словаре $\mathcal{W} = \langle \mathcal{C}, \mathcal{R}, \mathcal{P}, \mathcal{O} \rangle$ (в дальнейшем такие описания мы называем онтологиями).

Пусть $\mathcal{W}_o = \langle \mathcal{C}_o, \mathcal{R}_o, \mathcal{P}_o, \mathcal{O}_o \rangle$ – подсловарь $\mathcal{W}$, $\mathcal{O}_o = \{O_1, \ldots, O_l\}$, а θ – оператор типизации на $\mathcal{W}_o$.

Определение 4 (ОО-проекция)**.**

$$\mathcal{K}_o = TBox_o \cup ABox_o -$$

ОО-проекция словаря $\mathcal{W}_o$ и оператора θ, если

1) $TBox_o$ *– ацикличное множество OO-определений, любая интерпретация которого согласована с* θ*;*
2) для каждой аксиомы $Ax \in TBox_o$ *выполняется* $\mathcal{K} \vdash Ax$*;*
3) любая роль и атрибут из словаря $\mathcal{W}_o$ *входит в OO-определения* $TBox_o$ *не более одного раза;*
4) $ABox_o \subseteq ABox_{\mathcal{K}}$ *и*
- *для любого* $C(O) \in ABox_o$*:* $O \in \{O_1, \ldots, O_l\}$ *и* $C \in \mathcal{C}_o$*, причем для каждого* O *такой* C *является единственным;*
- *для любого* $R(O, O') \in ABox_o$*:* $O, O' \in \{O_1, \ldots, O_l\}$*,* $R \in \mathcal{R}_o$ *такие, что* $C(O), C'(O') \in ABox_o$*, причем* $\mathcal{K}_o \models C \sqsubseteq C_1$*,* $\mathcal{K}_o \models C' \sqsubseteq C_1'$ *и* $\theta(R) = [C_1, C_1']$*;*
- *для любого* $P(O, v) \in ABox_o$*:* $O \in \{O_1, \ldots, O_l\}$*,* $P \in \mathcal{P}_o$*,* $C(O) \in ABox_o$ *и* $v \in |D|$*, где* $\theta(P) = [C_1, D]$ *и* $\mathcal{K}_o \models C \sqsubseteq C_1$*.*

Основная задача OO-проекций – логически определять форматы данных, описываемых в объектных моделях. В [2] было показано, что описания в OO-проекциях изоморфны структурам данных в объектно-ориентированных языках с множественным наследованием. Формализм OO-проекций реализован в системе Libretto, которая основана на идеях, предложенных в [11]. В этой системе логические описания управляются как объектные базы знаний. Запросы на языке Libretto являются представленными в специальном синтаксисе формулами логики $\mathcal{SHOIN}(D)$, а поиск ответа на запрос – логический вывод в соответствующей OO-проекции. Поскольку логика достаточно простая, поиск получается очень эффективным и позволяющим работать в реальном времени. Существенно то, что благодаря синтаксическим решениям составление логических запросов на языке запросов доступно даже тем, кто не знает логику. Например, формула логики $\mathcal{SHOIN}(D)$ («существует персона, являющаяся внуком сорокалетней супруги некоторого мужчины»):

$$Man \sqcap \exists hasChild^{-}.\exists hasChild^{-}.(\exists age.\{40\} \sqcap \exists spouse^{-}.Man)$$

эквивалентна следующему запросу на языке Libretto

```
Man/spouse[age = 40]/hasChild/hasChild[Man]
```

Обратим внимание на то, что хотя формула в Libretto–формате эквивалентна формуле в стандартном виде, ее читабельность намного выше.

4. Формализация дублинского ядра

Для включения дублинского ядра в нашу среду на первом этапе разрабатываем его логичскую объектную модель в виде кокретной OO-проекции. Данная модель состоит из двух онтологий

- `http://rdf.ontobox.org/` – онтология, описывающая элементы RDF (в той части, в которой они нужны для формирования объектной модели дублинского ядра)
- `http://basics.ontobox.org/` – формализация самой модели (на основе документа [6]).

Данные онтологии находятся в свободном доступе [1].

Ключевым понятием дублинского ядра является *ресурс*. Понятие ресурса в нашей формализации дублинского ядра строится из понятия ресурса языка RDF. В RDF ресурс имеет чрезвычайно общее и абстрактное определение: это любая сущность, обладающая глобальным именем, представленным в синтаксисе URI. Кроме этого RDF снабжает ресурс локальным идентификатором и меткой (label), служащей для хранения базовой информации о ресурсе, которая ориентирована на человека. На языке OO–проекций эта информация запишется с помощью следующего набора эквивалентностей (OO-определений):

$$HasURI \equiv \forall uri.string$$

$$HasID \equiv \forall ID.string$$

$$Identified \equiv HasURI \sqcap HasID$$

$$Resource \equiv Identified \sqcap \forall label.string$$

На Libretto эти аксиомы запишутся следующим образом

```
class rdf:HasURI {
    rdf:uri v:string[0,1]
};
class rdf:HasID {
    rdf:ID v:string[0,1]
};
class rdf:Identified extends rdf:HasURI, rdf:HasID;
class rdf:Resource extends rdf:Identified {
    rdf:label v:string
};
```

Здесь добавляется `rdf:` – приставка, фиксирующая принадлежность сущностей к онтологии (системе аксиом) `http://rdf.ontobox.org/`.

Теперь обратимся к понятию ресурса в дублинском ядре. Класс ресурсов в формализации дублинского ядра строится через массивное применение множественного наследования к понятию ресурса в языке

RDF. Ресурс «собирается» из классов, каждый из которых описывает определенное свойство ресурса в DC. На Libretto соответствующая аксмома выглядит следующим образом:

```
class Resource
    extends
        rdf:Resource,  # является частным случаем RDF--ресурса
        HasRelation, # связь с другими сущностями
        HasCreator, # создатели ресурса
        HasSubject, # топик/классификатор ресурса
        HasTitle, # имя/название ресурса
        HasDescription, # описание ресурса
        HasPublisher, # кто опубликовал ресурс
        HasContributor, # контрибуторы ресурса
        HasType, # вид ресурса
        HasFormat, # формат/носитель ресурса
        HasTemporal, # темпоральные данные о ресурсе
        HasSpatial, # географические данные о ресурсе
        HasLang, # языки ресурса
        HasIsReferencedBy, # контактные данные
        HasReferences # сущности, на которые указывает ресурс
;
```

Каждый дополнительный класс, специфицирующий понятие ресурса на уровне дублинского ядра, добавляет ему некоторую характеристику, определенную в спецификации. Например, класс `HasCreator`, задающий информацию о создателях описываемого ресурса, определяется в онтологии `http://basics.ontobox.org/` следующим образом:

```
class HasCreator {
    creator Resource
};
```

В этом определении создатели ресурса сами интерпретируются как ресурс.

На основе данного подхода нами была построена формальная логическая модель дублинского ядра (это модель доступна в [1]). Конечно, структура этой модели в целом следует значительно более сложным правилам, чем те, которые обсуждались здесь. Однако с концептуальной точки зрения эта модель решает именно те задачи, которые были сформулированы нами выше. В частности, данная модель может служить универсальной основой для более сложных и специализированных систем описаний, обладающих строгой логической семантикой. Рассмотрим пять базовых преимуществ предлагаемого нами подхода.

Логическая семантика. Подмножество Libretto, используемое для описания моделей, имеет строгую логическую семантику, поскольку

выражения языка – «закодированные» формулы. Таким образом с логической точки зрения описание на Libretto представляет собой набор логических аксиом. Это означает, во-первых, что данные описания имеют строгую и однозначную сематику (что позволяет, например, их применять как нормативные компоненты спецификаций). Во-вторых, данные описания можно напрямую использовать как базовые аксиомы предметной области в работе интеллектных систем автоматической обработки знаний. В частности, любое описание предметной области в модели дублинского ядра, разработанное на Libretto, является его логической формализацией.

Объектная ориентированность. С программистской точки зрения описания на Libretto представляют собой достаточно стандартные объектно-ориентированные описания с множественным наследованием и разделением свойств (полей) на о-свойства (значениями которых являются объекты модели) и т-свойства (значения являются элементы типов данных – строки, числа и т.д.). Это означает, во-первых, что модели может разрабатывать очень широкий круг людей, для которых привычен объектно-ориентированный стиль работы. Во-вторых, структуры данных в описаниях напрямую отображаются в структуры объектно-ориентированных языков программирования, что принципиально облегчает обработку хранимых данных модели.

Исполняемость. Эта позиция является следствием предыдущего пункта. Поскольку формализация дублинского ядра базируется на объектной модели, каждое конкретное описание будет представлять собой объектную модель. Очевидно, что эта модель может быть непосредственно погружена в систему данных объектно-ориентированного языка программирования (как раз в силу его объектной ориентированности). Это означает, что данные описания могут напрямую манипулироваться ОО-программами. Это принципиально отличает наш подход от вариантов, когда описания даны в RDF-формате или в виде баз данных – в этих случаях необходимы форматы-посредники, позволяющие представлять описания в формате, удобном для языков ООП.

Уточнение через наследование. Объектно-ориентированный подход очень хорошо сочетается с идеологией дублинского ядра как базового языка описания метаданных, обеспечивающего общую платформу для других более развитых и специфичных языков. Это означает, что сложные описания могут быть спроецированы на простой язык, который, хотя и не обладает большой выразительностью, зато понимаем всеми. Такой подход полностью соответствует механизму наследования. Беря в качестве основы объектную модель дублинского ядра мы можем через наследование надстраивать над ней сколь угодно сложные системы описаний, которые в любой момент можно спроецировать на лапидарный язык дублинского ядра. Например, описывая музыкальную предметную область, можно ввести класс композиторов как наследника класса

создателей `HasCreator`. Тогда те обработчики, которые не «понимают», что такое «композитор», спускаясь вниз по иерархии, могут получить, пусть менее точную, информацию о том, что это создатель ресурса (музыкального произведения).

Совместимость с RDF-формализмом. Стандартный формат представления описаний на дублинском ядре – язык RDF, поэтому возможность экспорта/импорта описаний из объектной модели в RDF и обратно является существенной. Эта проблема решается напрямую, поскольку объектная модель дублинского ядра строится на основе онтологии RDF `http://rdf.ontobox.org/`.

Следует также отметить синергетический эффект сочетания преимуществ. Например, строгая логическая семантика в сочетании с объектной ориентированностью дает нашему подходу два интерфейса – логический и программный. Это обеспечивает предлагаемому нами методу качества посредника между логическими и программными системами.

Нами была проведена серия экспериментов, в которых модель дублинского ядра применялась в качестве общей «платформы» для разработки серии объектных баз знаний. В частности, были формализованы ряд спецификаций ISO, справочная система по линейной алгебре. Следует отметить широкие возможности практического применения подхода. Например, были формализованы научные классификаторы, а также база паспортов специальностей ВАК РФ. Благодаря общей базовой модели удалось автоматизировать процесс генерирования информационных ресурсов из соответствующих конкретных моделей (см. [4]).

5. Заключение

В работе рассматриваются возможности построения объектных логических моделей для систем метаописаний. Исследуется идея использования этих формализаций в качестве инструментов «прямого действия», применяемых для интеллектуализации информационных систем и в качестве переносимых логических модулей многократного использования. Построена полноценная логическая модель дубленского ядра – классической системы метаописаний, используемой как минимальный общий язык метаописаний. Показано, что логически этот принцип минимальности может осуществляться через реализацию конкретных систем описаний как надмоделей базовой формализации. Возможности прикладного использования предлагаемого метода продемонстрированы на примере построения информационно-справочных систем, автоматически сгенерированных из логических моделей.

Список литературы

1. Логическая модель дублинского ядра [Электронный ресурс]. – URL: http://teacode.com/models/dc/ (дата обращения: 10.11.10).
2. Малых А.А. Логические архитектуры и объектно-ориентированный подход / А. А. Малых, А. В. Манцивода, В. С. Ульянов // Вестн. НГУ. Серия: Математика, механика, информатика. – 2009. – Т.9, вып. 3. — С. 64-85.
3. Сайт Высшей аттестационной комиссии (ВАК) [Электронный ресурс].– URL: http://vak.ed.gov.ru/ (дата обращения: 10.11.10).
4. Объектные базы, построенные в системе Libretto [Электронный ресурс].– URL: http://teacode.com/ (дата обращения: 10.11.10).
5. Berners-Lee T. The Semantic Web / T. Berners-Lee, J. Hendler, O. Lassila // Scientific American. – 2001. – № 5. – P. 34-43.
6. DCMI Specifications [Электронный ресурс]. – URL: http://dublincore.org/specifications/ (дата обращения: 10.11.10).
7. Ian Horrocks, and Peter F. Patel-Schneider. Reducing OWL entailment to description logic satisfiability. In Fensel, D., Sycara, K. and Mylopoulos, J., eds., Proc. of the 2003 International Semantic Web Conference (ISWC 2003), number 2870 in Lecture Notes in Computer Science, 17-29. Springer.
8. ISO 639 [Электронный ресурс]. – URL: http://www.loc.gov/standards/iso639-2/ (дата обращения: 10.11.10).
9. ISO 3166 [Электронный ресурс].– URL: http://www.iso.org/iso/country_codes.htm (дата обращения: 10.11.10).
10. Protégé: open source ontology editor and knowledge-base framework. [Электронный ресурс]. – URL: http://protege.stanford.edu/ (дата обращения: 10.11.10).
11. Malykh A. A Query Language for Logic Architectures / A. Malykh, A. Mantsivoda // Proceedings of 7th International Conference «Perspectives of System Informatics». – Springer-Verlag Berlin Heidelberg, Lect [Электронный ресурс]. ure Notes in Computer Science 5947. – 2010. – P. 294-–305.
12. Resource Description Framework (RDF) [Электронный ресурс]. – URL: http://www.w3.org/RDF/ (дата обращения: 10.11.10).
13. Semantic Web activity: [Электронный ресурс]. – URL: http://www.w3.org/2001/sw/ (дата обращения: 10.11.10).
14. The Description Logic Handbook: Theory, Implementation, Applications / F. Baader and others. – Cambridge. – 2003. – P.574.
15. The Dublin Core Metadata Initiative [Электронный ресурс]. – URL: http://dublincore.org/ (дата обращения: 10.11.10).
16. Web Ontology Language (OWL) [Электронный ресурс]. – URL: www.w3.org/2004/OWL (дата обращения: 10.11.10).

A. V. Mantsivoda, N. O. Stukushin
Object Models and Distributed Knowledge Systems

Abstract. In this paper the issues of description logic application to metadata model development are considered. A method based on the OO–projection, a simple description logic, is introduced and illustrated by the example of Dublin Core.

Keywords: ontology; metadata; Dublin Core; query language Libretto; specification

Манцивода Андрей Валерьевич, доктор физико-математических наук, профессор, Институт математики, экономики и информатики, Ир-

кутский государственный университет, 664000, Иркутск, ул. К. Маркса, 1 тел.: (3952)242210 (andrei@baikal.ru)

Стукушин Никита Олегович, аспирант, Институт математики, экономики и информатики, Иркутский государственный университет, 664000, Иркутск, ул. К. Маркса, 1 тел.: (3952)242210 (stukushin@gmail.com)

Andrei Mantsivoda, prof., Irkutsk State University, 1, K. Marks St., Irkutsk, 664003 Phone: (3952)242210 (andrei@baikal.ru)

Nikita Stukushin, PhD student, Irkutsk State University, 1, K. Marks St., Irkutsk, 664003 Phone: (3952)242210 (stukushin@gmail.com)