Серия «Математика»
2009. Т. 2, № 2, С. 94—104

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

ИЗВЕСТИЯ
Иркутского
государственного
университета

УДК 510.62:004.82

Онтобокс: онтологии для объектов *

А.А. Малых, А.В. Манцивода,
Иркутский государственный университет

Аннотация. Семантический веб предлагает элегантную концепцию модернизации Интернета, основанную на логических средствах. Очевидно, что заявленная цель не может быть достигнута без существенного влияния предлагаемых подходов на массового веб-разработчика. Разработчики должны воспринять семантический веб как среду для своей практической деятельности. Чтобы добиться этого, необходимо достигнуть ряда целей, включая эффективную работу с онтологиями, наведение мостов между методами семантического веба и стандартными методами веб-разработки, инкапсуляцию тяжелых логических формализмов. В данной работе, базирующейся на идее логической архитектуры, мы предлагаем ряд подходов к решению этих проблем.

Ключевые слова: онтологии, Онтобокс, базы знаний

1. Введение

Семантический веб (СВ) [1][2] предлагает логический подход к обработке данных и знаний в Интернете. Этот подход ориентирован на развитие WWW как более «интеллектуальной» среды. Очевидно однако, что ощутимые результаты не могут быть получены без влияния на ежедневную практику людей, работающих в Интернете, в первую очередь, веб-разработчиков. СВ базируется на логических методах со строгой семантикой и элегантными методами обработки данных, и это большое достижение. Но именно его логическая природа и воздвигает определенный барьер между СВ и обычными разработчиками. В основном люди используют значительно более понятные средства. И онтологии, на которых СВ основан, никогда не станут достаточно простыми для этого.

* Работа выполнена при частичной финансовой поддержке программ «Фундаментальные исследования и высшее образование» (проект НОЦ-017 «Байкал») и «Развитие научного потенциала высшей школы (2009-2010 гг.)» (проекты РНП.2.2.1.1/5901 и 3.2.3/3488).

Некоторые специалисты также полагают, что онтологии хороши для обработки знаний на общем терминологическом уровне (уровне т.н. Т-бокса), но слишком тяжелы и неэффективны на уровне объектов и конкретных данных (в А-боксе). И поскольку конкретные данные составляют более 95% содержимого Интернета, то получается, что СВ оказывается здесь в аутсайдерах. Нам представляется, что неэффективность является проблемой не самих онтологий, а их конкретных реализаций. Одни и те же системы вывода не могут одновременно хорошо работать как не терминологическом, так и на объектном уровне – из-за хорошо известного противоречия между выразительностью и эффективностью. Для Т-бокса необходимы более выразительные и мощные, а значит, и более тяжелые системы вывода. Но это совершенно не означает, что онтологии нехороши для хранения конкретной информации. Просто в Т-боксе и А-боксе должны работать *разные* хотя и логически совместимые формальные системы. И, к счастью, дескриптивные логики достаточно гибки для того, чтобы обеспечить нас инструментами как первого, так и второго рода.

В [5] нами предлагается подход к стратификации («расслоению») дескриптивных логик, который там назван *логической архитектурой*. Логическая архитектура базируется на идее, что не только сама логика должна быть расслоена. Каждая логическая страта должна быть обеспечена своими собственными интерфейсами и обработчиками. В частности, такой подход позволяет развивать очень эффективные и практичные методы обработки объектов и конкретных данных, оставаясь при этом в том же логическом контексте, что и при работе на терминологическом уровне. Работая с онтологиями на разных уровнях, мы можем одновременно пользоваться преимуществами простых, но быстрых и масштабируемых техник, и возможностями сложных, но выразительных методов. И поскольку все эти уровни уложены в рамках унифицированной логической среды, то результаты, полученные на каждом из уровней, могут свободно использоваться в рамках среды в целом.

В данной работе рассматриваются возможности, связанные с реализацией подлогик дескриптивных логик, достаточно «простых» для того, чтобы допускать эффективные реализации, и быть доступными для массового пользователя.

2. Логика для объектов

Пусть $\mathcal{KD}$ – предметная область и $\mathcal{L} = \langle L, \vdash \rangle$ – дескриптивная логика, в которой $\mathcal{KD}$ формально описана. Здесь L – язык $\mathcal{L}$ и $\vdash$ – ее система вывода. Вопрос заключается в том – как работать с объектами в $\mathcal{KD}$? Конечно, мы бы предпочли не использовать внешние технологии (вроде

баз данных), а работать с ними в рамках онтологий. Тогда нам нужна специальная «объектная» подлогика $\mathcal{L}_{Obj}$ логики $\mathcal{L}$ – достаточно простая для того, чтобы допускать конкурентноспособные реализации и быть понятной людям. Для совместимости необходимо также, чтобы результаты обработки на уровне логики $\mathcal{L}_{Obj}$ допускали корректное использование на более высоких уровнях $\mathcal{L}$. Кроме того, $\mathcal{L}_{Obj}$ должна быть достаточно «дружелюбной» к популярным сегодня технологиям хранения информации, чтобы навести мосты между методами семантического веба и уже имеющимися веб-технологиями и хранилищами данных. Для удовлетворения этих требований, необходимо снабдить $\mathcal{L}_{Obj}$ специальными методами обработки и интерфейсами, что сделает $\mathcal{L}_{Obj}$ частью логической архитектуры $\mathcal{L}$ [5].

Вопрос в том, какова та минимальная логика, в которой можно успешно и эффективно работать с объектами в онтологиях? С нашей точки зрения это должен быть логический эквивалент хорошо известного *объектно ориентированного подхода*. В [5] нами вводится понятие ОО-проекции, которая представляет собой расширение простой логики $\mathcal{F}L_0$ [3], моделирует объектно ориентированные описания, и является хорошим кандидатом на роль абстрактной логики $\mathcal{L}_{Obj}$. В частности, ОО-проекции позволяют работать с типами данных вроде целых чисел и строк.

Рассмотрим, как объектно-ориентированные структуры могут моделироваться в рамках дескриптивных логик. Пусть $\mathcal{L}$ дескриптивная логика с областью типов данных $\mathcal{D}$. *Словарем* $\mathcal{L}$ назовем $\mathcal{V} = \langle \mathcal{C}, \mathcal{R}, \mathcal{P}, \mathcal{O} \rangle$, где $\mathcal{C} = \{C_1, \ldots, C_n\}$, $\mathcal{R} = \{R_1, \ldots, R_m\}$, $\mathcal{P} = \{P_1, \ldots, P_k\}$ и $\mathcal{O} = \{O_1, \ldots, O_p\}$ конечные подмножества *имен концептов*, *ролей*, *атрибутов* и *имент* логики $\mathcal{L}$, соответственно. Оператор типизации θ привязывает каждую роль $R \in \mathcal{R}$ к паре концептов $\theta(R) = [C_d, C_r]$, где $C_d, C_r \in \mathcal{C}$. Аналогично, для каждого атрибута P имеем $\theta(P) = [C_d, D_r]$, где $C_d \in \mathcal{C}$ и $D_r \in \mathcal{D}$. Концепт C_d назовем областью определения R (или P), а концепт C_r (тип данных D_r) назовем областью значений R (или P). Интерпретация I в $\mathcal{L}$ *подтверждает* оператор типизации θ, если

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

Определение 1. [OO-concept] 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 $E_i \in \mathcal{C}$, тогда ОО-концепт называется *примитивным*.

Определение 2. [ОО-определение] Аксиома

$$C \sqsubseteq E, \tag{2.1}$$

где E примитивный ОО-концепт, называется *объектно-ориентированным определением.*

Пусть $\mathcal{A} = \{C_i \sqsubseteq E_i\}$ – множество ОО-определений. Говорим, что C_i зависит от C_j в $\mathcal{A}$, если $C_i \sqsubseteq C_j \sqcap E' \in \mathcal{A}$, или существует C_k такой, что C_i зависит от C_k и C_k зависит от C_j. $\mathcal{A}$ называется ацикличным, если оно не содержит C_i, зависящих от самих себя.

Язык ОО-проекций состоит из ОО-концептов, и только ОО-определения допускаются в их Т-боксе. ОО-определения имеют непосредственную объектно-ориентированную интерпретацию, в которой C рассматривается как объектно-ориентированный класс, наследующий от $C_1, \ldots, C_f$, и имеющий поля $R_1, \ldots, R_g$, $P_1, \ldots, P_h$.

Теперь обратимся к определению ОО-проекции. 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 $\mathcal{V} = \langle \mathcal{C}, \mathcal{R}, \mathcal{P}, \mathcal{O} \rangle$ (в дальнейшем такие описания мы называем онтологиями). Пусть $\mathcal{V}_o = \langle \mathcal{C}_o, \mathcal{R}_o, \mathcal{P}_o, \mathcal{O}_o \rangle$ – подсловарь $\mathcal{V}$, $\mathcal{O}_o = \{O_1, \ldots, O_l\}$, а θ – оператор типизации на $\mathcal{V}_o$.

Определение 3. [ОО-проекция]

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

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

1. $TBox_o$ – ацикличное множество ОО-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 $\mathcal{V}_o$ входит в ОО-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 $\mathcal{K}_o \models C \sqsubseteq C_1$.

В [5] нами вводится абстрактный объектно-ориентированный язык (AOOL) с множественным наследованием, который базируется на идеях из [7] и др. Показывается, что любая ОО-проекция $\mathcal{K}_o$ может быть транслирована в AOOL программу $prog(\mathcal{K}_o)$ таким образом, что иерархия ОО-концептов из $\mathcal{K}_o$ (порожденная включением $\sqsubseteq$) и иерархия типов данных $prog(\mathcal{K}_o)$ являются изоморфными алгебрами. Этот результат показывает, как объектные модели могут быть погружены в

онтологии общего вида. На самом деле, вложенные объектные модели могут играть роль ядра и наиболее «осязаемой» составляющей более общих логических описаний, которая содержит базисную информацию о соответствующем предметной области. Поскольку структура ОО-проекций весьма проста, они допускают очень эффективные реализации. В пункте 3 мы рассматриваем Онтобокс – очень эффективную реализацию ОО-проекций.

3. Онтобокс

В этом пункте рассматривается Онтобокс (OntoBox) – реализация ОО-проекций, разработанная на Java. Онтобокс оперирует с такими стандартными понятиями, как концепты (классы), типы данных, роли (объектные свойства), атрибуты (свойства в типах данных) и объекты. Любая онтология в Онтобоксе обрабатывается как отдельное пространство имен, в котором собрана определенная информация об иерархии классов и объектах. Можно сказать, что онтология в Онтобоксе играет роль «пакета/модуля», в котором хранится достаточно замкнутая порция описания предметной области. Например, терминологические описания предметной области и информация об ее конкретных объектах могут храниться в разных онтологиях. Таким образом, ОО-проекции могут хранить несколько онтологий (или сегментов нескольких онтологий).

Объектная модель Онтобокса (ООМ). Реализация Онтобокса базируется на его абстрактной объектной модели [8]. ООМ определяет механизмы взаимодействия базовых компонентов системы. Она также используется как интерфейс взаимодействия с внешними модулями, которые могут подключаться к Онтобоксу в виде «плагинов». Например, через объектную модель с Онтобоксом работает его язык запросов. Другим потенциальным примером такого «плагина» может стать система автоматического доказательства в дескриптивных логиках, для которой Онтобокс будет служить очень эффективным обработчиком объектных подмоделей. С реализационной точки зрения Онтобокс состоит из Java-интерфейсов и классов, которые разделяют абстрактную структуру ООМ и ее конкретные реализации. Java-интерфейсы описывают базовые операции чтения, создания и изменения в обрабатываемых онтологиях.

Имеется одно важное отличие ООМ от стандартного подхода, принятого в дескриптивных логиках. В дескриптивных логиках значения свойств (как ролей, так и атрибутов) объекта неупорядочены и рассматриваются как множества. ООМ базируется на концепции *семантического программирования* [4], в котором неупорядоченные множества в данном контексте заменяются на упорядоченные списки.

Реализация языка запросов. Выше упоминалось, что язык запросов работает с данными, хранящимися в Онтобоксе через его объект-

ную модель. На самом деле, язык запросов реализован в отдельной Java-библиотеке взаимодействующей с внешним миром через интерфейс OOM. Текущая версия языка запросов поддерживает широкую гамму возможностей, позволяющих получать из онтологий необходимую информацию, а также манипулировать их содержимым.

Хранение данных. Текущая реализация Онтобокса является третьей по счету. Первая реализация была построена над реляционной базой данных. Структуры OntoBox отображались в таблицы базы данных, и генерировался переносимый код, который мог работать с различными СУБД, включая MySQL, HSQLDB, Apache Derby/IBM DB2. К сожалению, этот подход не позволил получить удовлетворительную эффективность, поскольку затраты на работу самого Онтобокса были помножены на медленную работу баз данных. Некоторые оптимизации (например, кэширование в оперативной памяти) несколько улучшили ситуацию, но этого оказалось недостаточно.

Во второй версии мы отказались от баз данных и реализовали «легкую» версию системы, которая полностью работала в оперативной памяти компьютера. Такое решение основывается на феноменально быстром росте объемов оперативной памяти в современных компьютерах. Вторая версия использовала в качестве основного строковое представление данных. Третья версия базируется на числовом кодировании данных и сокращенном комплекте базовых методов OOM (остальные методы определяются через базовые). Это повышает эффективность работы на 50%. Теперь компоненты Онтобокса сохраняются в специальных структурах, базирующихся на двусторонних отображениях (bi-map), и гибридах отображений и списков (близких к multi-map). Полные имена (URI) сущностей и онтологий индексируются. Информация, сохраненная в Онтобоксе, запрашивается и изменяется с помощью операций двух типов - *ask* (операции чтения) и *tell* (операции редактирования). Сохранность данных, находящихся в Онтобоксе между сессиями работы, обеспечивается дисковым логом, который содержит последовательность операций редактирования.

Атрибуты объектов могут содержать большие объемы пассивных данных (например, изображения или тексты книг), которые не вовлечены в активную обработку. Для таких случаев реализована технология DMap. Она разделяет атрибуты и значения атрибутов большого объема, сохраняя в атрибутах уникальные идентификаторы, играющие роль ключей в DMap, и использующиеся в тот момент, когда содержимое атрибута должно быть получено. DMap работает на нижнем реализационном уровне, и не виден на уровне OOM. Это означает, что обычные атрибуты, и крупные данные, сохраняемые на диске, логически неотличимы. DMap базируется на собственной объектной модели, которая позволяет проводить эксперименты с различными вариантами реализации такого хранилища. В настоящее время Онтобокс работает с

простой, но достаточно эффективной версией DMap. Также проводятся эксперименты с реализациями, базирующимися на BerkleyDB и на нашем собственном подходе, использующем специфические механизмы доступа к информации в Онтобоксе.

Транзакции. Механизм транзакций также реализован в Онтобоксе. Он играет роль, аналогичную транзакциям в базах данных. Транзакции весьма полезны для эффективной обработки ошибок, для параллельной обработки данных, и позволяют избегать возникающие в подобных ситуациях конфликты. Механизм транзакций также необходим для организации версионности.

Транзакция в Онтобоксе состоит из последовательности операций, которые выполняют один логический шаг обработки данных. Транзакцию можно «откатить» (roll back) с полным восстановлением информационной базы (например, в случае некорректных операций), либо «принять» (commit), если все в порядке. Режим «auto-commit», когда каждая операция рассматривается как мини-транзакция, также поддерживается Онтобоксом. Транзакции в Онтобоксе полностью изолированы, то есть результаты выполнения промежуточных операций транзакции недоступны снаружи.

Обработка событий. Чтобы обеспечить эффективную интеграцию Онтобокса с разнообразными внешними системами и модулями, в нем реализована система обработки событий. В частности, обработка событий полезна для формирования стратегий поиска решения, для создания пользовательских интерфейсов, для того чтобы информировать решатели и интерфейсы об изменениях, происходящих в Онтобоксе. Обработка событий реализована на уровне ООМ, и поэтому любой внешний модуль, который использует для взаимодействия с Онтобоксом его объектную модель (включая язык запросов), также может управлять системой обработки событий. События обрабатываются в рамках отдельного треда. Этот тред работает со специальным буфером/очередью tell-операций, который оповещает зарегистрированных «слушателей» о пойманных событиях.

Переносимость. Онтобокс поддерживает несколько сервисов экспорта/импорта. Во-первых, Онтобокс обладает интерфейсом OWL API [9], который обеспечивает экспорт в несколько стандартных форматов, например, в OWL/RDF. Во-вторых, онтологии могут экс/импортироваться в виде скриптов языка запросов. Кроме того, онтологии могут портироваться в облегченном XML формате (напоминающем OWLX), настроенном на представление информации из ОО-проекций. Доступен также конвертор из дампов БД в формате DDL/SQL в плоские онтологии. Еще один модуль транслирует содержимое Онтобокса в Java-программу, работающую в ООМ. Выполнение этой программы позволяет заново построить закодированную базу знаний в новой копии Онтобокса. Наконец, Онтобокс способен сгенерировать дамп своего внутрен-

него состояния, который затем может быть загружен в любую другую копию Онтобокса в виде отдельного проекта.

4. Онтосервер

Онтобокс может играть роль «продвинутого» аналога систем управления базами данных при решении стандартных задач разработки сайтов, порталов и веб-сервисов. Для многих приложений решения, базирующиеся на онтологиях и Онтобоксе, работают значительно эффективнее, чем решения, основанные на БД. Но для того, чтобы сделать Онтобокс конкурентноспособным игроком на поле веб-разработки, ему необходим комплект сетевых возможностей. Выше рассматривался способ прямого взаимодействия с Онтобоксом через объектную модель и язык запросов. Нами была также разработана клиент-серверная технология (OntoBox client/server technology, OCST), которая поддерживает удаленный доступ к онтологиям Онтобокса, а также доступ из приложений, разработанных на других языках программирования. Эта технология работает поверх HTTP как базового протокола, что позволяет вовлечь стандартную инфраструктуру HTTP, включая прокси-серверы и брэндмауэры. Тело запросов в OCST представляет собой последовательность выражений на языке запросов, которые могут включать как ask-, так и tell-операции. Онтобокс оценивает запросы и возвращает результаты также по OCST. Чтобы передать результаты, OCST использует простой бинарный протокол обмена данными, базирующийся на списках и отображениях. Существует опциональная возможность сжатия передаваемых данных, которая полезна в случае медленных каналов обмена.

OCST поддерживается в текущей версии Онтобокса в полном объеме, превращая его в *«онтологический сервер»* с огромным спектром возможных приложений. В частности, разработаны клиентские модули для C, C#, Ruby и Javascript. Это означает, что Онтобокс может использоваться как в веб-, так и автономных приложениях, разрабатываемых на этих языках. В настоящее время развивается альтернативная версия OCST, которая базируется на текстовых форматах XML и JSON. Они менее эффективны, чем бинарный формат, но проще в отладке, и кроме того, многие языки программирования обладают специальными библиотеками для работы с XML и JSON.

5. Язык запросов

Онтобокс предоставляет два способа доступа к хранящейся в нем информации. Первый способ низкоуровневый и основан на абстрактной объектной модели OOM. Для большинства приложений этот способ,

хотя и обеспечивает высокий уровень эффективности, является слишком громоздким и затратным с точки зрения как разработки, так и поддержки. ООМ можно рассматривать как своего рода «ассемблер», предназначенный для разработки других более удобных средств доступа к информации, хранящейся в онтологиях. Среди таких средств выделяется встроенный *язык запросов* Онтобокса, который играет в нем роль, аналогичную роли SQL в базах данных. Этот язык навигационного типа. Запросы, написанные на нем, образно говоря, описывают путь, пройдя который, можно добраться до искомых данных. С логической точки зрения каждый запрос эквивалентен некоторой формуле в логике $\mathcal{SHOIN}(D)$. Более подробно сам язык запросов и его семантики описаны в [6].

6. Апробация

Предварительные оценки показывают, что Онтобокс может использоваться в большом спектре самых разнообразных приложений. Нами разработан ряд пилотных систем (телефонная книга, справочные системы по линейной алгебре и математическому анализу, междисциплинарная онтология по географическим объектам озера Байкал, 3D модель озера, репозиторий мультимедийных ресурсов и т.д.). Онтобокс свободно работает с онтологиями, содержащими сотни тысяч объектов, и во многих случаях может эффективно заменить базы данных. Особенно этот подход полезен, когда мы работаем с иерерхически структурированными данными и данными, представленными в виде объектных моделей. Базы данных плохо приспособлены для обработки иерархических структур (иногда для того, чтобы преодолеть эту трудность, приходится использовать специальные системы типа [10] and [11]), в то время, как Онтобокс предлагает весьма эффективный и естественный стиль работы с объектными моделями. Например, в [6] рассматривается таксономия NCBI [12], которая описывает имена всех живых организмов, содержащих хотя бы одну нуклеотидную или протеиновую цепочку. Таксономися содержит 482960 объектов. Работая с этой таксономией Онтобокс со своим языком запросов демонстрирует семикратное превышение скорости по сравнению с первичной базой данных с запросами, представленными на SQL. Важно также, что Онтобокс имеет эффективную связь с объектно-ориентированными языками программирования как Java, C#, Ruby. Взаимодействие Онтобокса с языками ОО-программирования, когда объекты Онтобокса напрямую отображаются во внутренние объекты этих языков, формирует весьма элегантную и сбалансированную технику работы, позволяющую быстро и эффективно разрабатывать соответствующие приложения.

7. Заключение

Мы надеемся, что использование объектно-ориентированных подлогик дескриптивных логик является полезной идеей для продвижения элитных логических средств в направлении массовых задач веб-разработки. В частности, Онтобокс, эффективно реализуя достаточно простую подлогику дескриптивной логики $\mathcal{SHOIN}(D)$, вполне доступен для применения даже неопытными пользователями – и это неоднократно проверялось в рамках обучения студентов. Представляется, что гибкое сочетание мощных логических средств (с большими выразительными возможностями, но при этом тяжелых алгоритмически и трудных для понимания обычным пользователем) с легкими логическими средствами (не столь выразительными, но очень эффективными и понятными широкому кругу разработчиков) является интересным направлением использования логических структур.

Список литературы

1. Berners-Lee T., Hendler J., Lassila O. The Semantic Web // Scientific American. – 2001. – №5. – P.34-43.
2. The Semantic Web. http://www.w3.org/2001/sw/
3. Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, Peter F. Patel-Schneider (Eds.): The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press 2003.
4. 4. Goncharov S., Ershov Yu., Sviridenko D. Semantic Programming // 10th World Congress Information Processing'86. Dublin, 1986. P. 1093–1100.
5. А.А. Малых, А.В. Манцивода, В.С. Ульянов / Логические архитектуры и объектно-ориентированный подход // Вестник НГУ. Серия: Математика, механика, информатика. 2009. Т9. Вып. 3. С. 64-85
6. A. Malykh and A. Mantsivoda. A Query Language for Logic Architectures. Proceedings of PSI'09, p.212-220. http://meta2project.org/ru/psi09.pdf
7. Luca Cardelli. A Semantics of Multiple Inheritance. Information and Computation, v.76 n.2-3, p.138-164, February/March 1988.
8. OntoBox's Object Model. http://teacode.com/meta2/javadoc/common/.
9. The OWL API. http://owlapi.sourceforge.net/
10. Hibernate: mapping an object-oriented domain model to a relational database. https://www.hibernate.org/
11. ADO.NET Entity Framework. http://www.microsoft.com/sqlserver/2008/en/us/ado-net-entity.aspx
12. The NCBI Entrez Taxonomy. http://www.ncbi.nlm.nih.gov/sites/entrez?db=taxonomy

---

A.A. Malykh, and A.V. Mantsivoda

Ontobox: Ontologies for Objects

Abstract. The Semantic Web offers the elegant conception of Web modernization, which is based on logical means. It is evident that this goal can not be achieved without the significant influence of the approach on the mass of 'conventional' web developers. The developers must accept the Semantic Web as an environment for everyday activities. To get this, it is necessary to solve a number of problems including the efficient work with ontologies, bridging gaps between the SW methods and standard web development techniques, incapsulation of heavy logical formalisms. In this paper we propose an approach to these problems, which is based on the ideas of logical architectures.

Keywords: ontologies, Ontobox, knowledge bases

Малых Антон Александрович, Институт математики,экономики и информатики, Иркутский государственный университет, 664003, Иркутск, ул. К. Маркса, 1 тел.: (3952)242210 (malykh@baikal.ru)

Манцивода Андрей Валерьевич, Институт математики,экономики и информатики, Иркутский государственный университет, 664003, Иркутск, ул. К. Маркса, 1 тел.: (3952)242210 (andrei@baikal.ru)

Anton Malykh, Irkutsk State University, 1, K. Marks St., Irkutsk, 664003 Phone: (3952)242210 (malykh@baikal.ru)

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