Архитектура RDFS-системы.

Практика использования открытых стандартов и технологий Semantic Web в системе ИСИР.

Бездушный А.А. alix@7ka.mipt.ru
Бездушный А.Н. bezdushn@ccas.ru
Нестеренко А.К. alexn@ccas.ru
Серебряков В.А. serebr@ccas.ru
Сысоев Т.М. tim@ ccas.ru

Вычислительный центр им. А.А. Дородницина РАН

Аннотация

В работе анализируются основные концепции Semantic Web и перспективы его использования. Приводится сопоставление парадигмы Semantic Web с традиционными парадигмами программирования. Рассматривается общая архитектура новой версии системы ИСИР, опирающейся на открытые стандарты W3C: Semantic Web, XML технологии, и на применение open-source решений. Архитектура позволяет разрабатывать распределённые объектно-ориентированные цифровые библиотеки, информационные и корпоративные порталы на базе различных типов хранилищ информации, таких как объектные и реляционные базы данных, LDAP-каталоги. Система параметризуется описанием объектной схемы данных конкретной предметной области и легко адаптируется к её изменениям. Для описания схемы используется RDFS. Архитектура имеет многоуровневую модульную организацию, каждый уровень имеет собственные цели и абстракции. Фундамент архитектуры - ядро ИСИР - унифицирует механизмы работы с объектными данными, хранимыми в РСУБД, а также предоставляет ряд услуг управления ими, в частности, разграничение прав доступа, журнализация изменений. На базе ядра строятся более высокоуровневые сервисы, такие как RDF/XML-обмен данными, репликация информации между репозиториями, атрибутно-полнотекстовая индексация данных, управления потоками работ и др. Web-архитектура ИСИР предоставляет средства для простой и эффективной разработки корпоративных порталов. Средства публикации информации и построения отчётов применяют механизмы XSLT и поддерживают широкий спектр целевых форматов. Служба управления потоками работ следует стандартам WfMC - канонической модели и языку спецификации потоков работ. Описываются реализации служб, их роль и место в новой версии системы ИСИР.

Труды 5ой Всероссийской научной конференции «Электронные библиотеки: перспективные методы и технологии, электронные коллекции» - RCDL2003, Санкт-Петербург, Россия, 2003.

1 Resource Description Framework

"Semantic Web – это расширение Web, в котором информации придаётся определённая семантика, позволяя людям и машинам работать вместе" – примерно такое определение дают своим работам члены W3C Semantic Web Activity [8]. Целью этого проекта является внедрение в Web таких технологий, которые позволят существенно повысить уровень интеграции информации, обеспечить развитую машинную обработку данных, дадут возможность выдавать более адекватные ответы на поисковые запросы и т.д.

Текущее состояние Web характеризуется слабой структурированностью данных, низким уровнем их взаимосвязи. Распространение XML-технологий дает возможность структурировать информацию, обеспечить синтаксическую интероперабельность приложений. Semantic Web является логическим продолжением развития Web – от гипертекстовых страниц к XML-данным, а от XML – к смысловому содержанию и объединению разбросанной в Web информации.

Semantic Web базируется на модели данных Resource Description Framework (RDF), которая позволяет объединить информацию из различных источников, включая базы данных и системы инженерии знаний. RDF может быть наиболее полезен в обеспечении совместного использования информации, смысл которой может одинаково интерпретироваться различными программными агентами. Второй базовый компонент Semantic Web – это RDF/XML-синтаксис, который позволяет представить RDF-данные в XML-виде. Следующий уровень в пирамиде технологий Semantic Web занимает язык RDF Schema – язык описания словарей RDF-терминов (классов и свойств Web-ресурсов). RDFS служит фундаментом для более богатых языков описания онтологий предметной области, которые

позволяют адаптировать к Web системы логики и обеспечить семантическую обработку данных.

RDF модель данных, составляющая основу методики Semantic Web, является представителем семейства ER-моделей данных, специфика которой состоит том, что ресурсы и свойства идентифицируются с помощью глобальных идентификаторов - URI. RDF описывает предметную область в терминах ресурсов, свойств ресурсов и значений свойств. RDF-данные можно расценивать как совокупность утверждений – субъект, предикат (свойство) и объект утверждения, представлять в виде направленного графа, образуемый такими утверждениями.

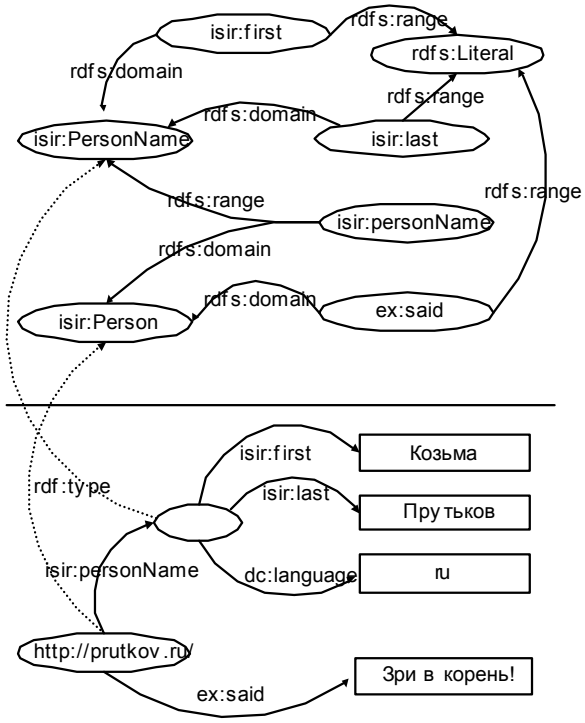
RDF/XML-синтаксис [14] позволяет записать граф в последовательной форме, пригодной для обмена данными. Этот синтаксис достаточно гибок – он допускает различные формы записи одного и того же графа, различные сокращенные формы.

RDF-схема (RDFS, [7]) представляет собой систему типов для Semantic Web. RDFS позволяет определить классы ресурсов и свойства как элементы словаря, и специфицировать, какие свойства, с какими классами могут быть использованы. RDFS выражает эти словари средствами RDF, предоставляя набор предопределённых ресурсов и свойств с обозначенной для них смысловой нагрузкой, которые могут быть использованы для описания новых RDF-словарей.

Таким образом, любое RDFS-описание представляет собой «обычные» RDF-данные – данные о классах и свойствах. RDFS позволяет определить уникальные (идентифицируемые URI) классы ресурсов, представляющие концептуальную модель конкретной предметной области, и уникальные (идентифицируемые URI) свойства, интересующие нас в этой области. Принадлежность ресурса к конкретному классу задается с помощью свойства rdf:type. Описываемые в словаре классы сами являются экземплярами предопределённого класса rdfs:Class, свойства же являются экземплярами rdf:Property. RDFS позволяет указать, каким классам пРИСущи заданные свойства, и ресурсы какого класса могут появиться в качестве значения заданного свойства. Эта информация указывается в словаре с помощью свойств rdfs:domain и rdfs:range соответственно. RDFS позволяет связать классы (rdfs:Class) отношениями множественного наследования (rdfs:subClassOf). В RDFS-модели, как и в обычном объектном подходе, классам свойственен полиморфизм. То есть, экземпляр подкласса всегда может сыграть роль экземпляра своего суперкласса, и появиться как субъект или объект свойства, для которого в качестве соответственно range или domain был указан суперкласс. Свойства также могут быть связаны отношениями множественного наследования (rdfs:subPropertyOf). Наследование свойства означает более узкую специализацию этого свойства, уточнение смысла и сужение границ использования.

На рисунке приведен пример двух RDF-графов, один из которых соответствуют RDF-схеме некоторой предметной области, а второй - конкретным данным.

С введением механизмов определения словарей, деятельность Semantic Web выходит на новый уровень. На данный момент различные организации по стандартизации предлагают стандартные словари для описания ряда предметных областей. Использование таких публичных словарей (или сопоставление с ними) позволяет «незнакомым» приложениям обмениваться информацией друг с другом, точно так же, как человек, попавший в чужую страну без знания языка, всё равно сможет в ней общаться - с помощью «стандартного» языка жестов, или, например, угадывая латинские, греческие, славянские корни в словах. В качестве примера таких инициатив стандартизации можно упомянуть инициативу Dublin Core [2], предоставляющую минимальный набор свойств для идентификации ресурсов Web, Publishing Requirements for Industry Standard Metadata (PRISM), определяющую словарь метаданных для издательских организаций, Electric Power Research Institute Common Information Model, указывающую общую семантику для энергетических систем, RDF Site Summary для описания каналов новостей Web-порталов и многие другие инициативы.

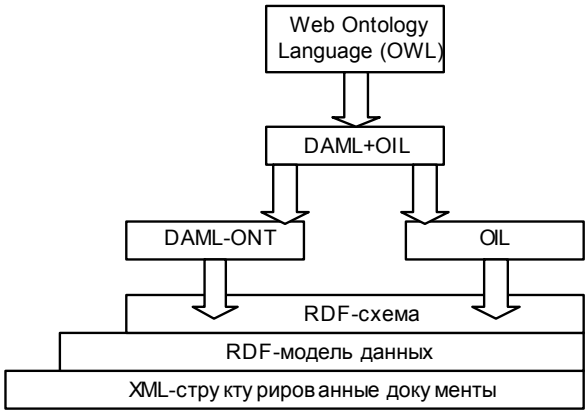


Язык RDFS предоставляет лишь базовые возможности для описания словарей предметных областей, но он легко может быть расширен дополнительными примитивами моделирования, более детально и специализировано описывающими нужные аспекты классов и свойств. Механизм расширения внутренне присущ RDFS, поскольку для описания схем используется модель данных RDF, которая позволяет расширить описание любых ресурсов дополнительной информацией. Предопределённый словарь «мета-типов» RDFS также может быть рас-

ширен под нужды приложения, благодаря чему появляется возможность добавлять в язык новые примитивы.

Расширяемость позволяет RDFS стать фундаментом для более богатых языков концептуального моделирования – языков описания web-онтологий предметных областей. Цель таких языков – указать дополнительную машинно-интерпретируемую семантику ресурсов, то есть сделать машинное представление данных более похожим на положение вещей в реальном мире. Использование богатых языков концептуального моделирования позволит адаптировать к Web большое количество наработок в области систем инженерии знаний и баз знаний. Привлечение к Web систем логики и искусственного интеллекта составляет вершину «пирамиды Semantic Web», обеспечивая адекватный поиск информации и её машинную интерпретацию.

Первыми предложениями по описанию онтологий на базе RDFS были DAML-ONT (DARPA Agent Markup Language) [11] и European Commission OIL (Ontology Inference Layer) [13]. На базе этих двух предложений возникло совместное решение – DAML+OIL, которое привело к созданию в рамках инициативы Semantic Web отдельной группы, ответственной за пересмотр этого решения и стандартизацию языка описания Web-онтологий (OWL – Web Ontology Language) [5].



Однако ориентированность языков описания онтологий на системы математической логики делает их слишком тяжеловесными для огромного количества приложений, которым достаточно простого языка описания словарей – RDFS. И это правильно, каждая ступень в пирамиде Semantic Web – это ступень, на которой многие приложения могут остановиться, согласно своим собственным требованиям к данным и их использованию.

2 Сопоставление RDF(S) с другими парадигмами

Система типов RDFS похожа на многие общепринятые системы типов, как в ER-моделировании, объектно-ориентированном программировании и UML [9], и т.п. Инициатива Semantic Web не ставит перед собой цели создать новую модель данных, напротив, она ориентируется на интеграцию различных моделей данных с целью получения информации из соответствующих источников. RDFS отличается от этих стандартных систем типов в нескольких существенных аспектах, которые являются следствием глобализации и децентрализации информационной системы, к которой мы приходим, «выходя» в Web из установленных моделью данных рамок. В каком-то смысле RDF(S) есть адаптация этих моделей к Web. Рассмотрим сопоставление примитивов RDFS и модели данных объектно-ориентированного программирования (согласно RDF Primer [7]).

Один из архитектурных принципов Web состоит в том, что кто угодно может расширить описание существующих ресурсов [16], то есть «кто угодно может сказать, что угодно, о чём угодно». Это означает, что отношение между двумя объектами может храниться отдельно от любой другой информации об этих объектах. Это сильно отличается от того, к чему мы привыкли в обычных объектно-ориентированных системах, в которых считается, что информация об объекте хранится внутри объекта: определение класса объекта подразумевает указание места хранения его свойств. Такое отличие является следствием децентрализации и адаптации к положению вещей в реальном мире. Например, один человек может определить автомобиль, как нечто, имеющее колёса, вес и размер, но не предвидеть цвет. Это не остановит другого человека от утверждения, что его машина – красная, используя некоторый словарь цветов.

Из этого архитектурного принципа Web следует основное отличие парадигмы RDFS от объектной парадигмы – это ее свойство-центричность. Свойства (отношения, предикаты) в RDFS являются объектами первого уровня, как и классы: они идентифицируются URI и определяются независимо от классов, тогда как в объектной и ER парадигмах свойства (атрибуты) указываются в «теле» класса, смысл свойств с одинаковыми названиями в разных классах может быть различен. Впрочем, такой подход уже использовался, например, в X.500, LDAP, где свойства и их характеристики описываются отдельно от класса, а потом «привязываются» к нужным классам. Он оправдывает себя в системах, ориентированных именно на хранение разнообразной слабоструктурированной информации.

Вместо того, чтобы описывать классы в терминах свойств (структуры), имеющихся у него, как это делается в объектно-ориентированных системах, RDFS описывает свойства в терминах классов, к которым они применимы, указывая rdfs:domain (область применения свойства) и rdfs:range (область значений свойства). Различие между этими подходами может показаться только синтаксическим, но на самом деле есть существенная разница, которая связана как раз с глобализацией информационной системы при адаптации её к Web, где «кто угодно может сказать, что угодно, о чём угодно». Например, если кем-то определен класс ex:Book со свойством ex:author, принимающим значения типа

ex:Person, то это не запрещает другим разработчикам придать классу ex:Book дополнительное свойство my:publisher, достаточно лишь указать этот класс в rdfs:domain нового свойства my:publisher. Это не требует переопределения класса, причем создатели класса могут быть в неведении данного факта. В то же время в ООП потребовалось бы переопределить и перекомпилировать класс.

Кроме того, RDFS вообще не требует, чтобы у свойства была задана область применения – свойство без domain может быть использовано для описания любого ресурса, независимо от его класса. Определение свойства без указания области применения позволяет использовать его в будущем в ситуациях, которые не могли быть предвидены в момент разработки схемы. Именно так поступает Dublin Core[2], предоставляя словарь стандартных свойств, пригодных для описания любого Web-ресурса, для которого они окажутся полезными.

Другое важное отличие в семантике RDFS-описаний – это то, что они носят описательный, а не «предписывающий» характер, то есть, они могут использоваться не для того, чтобы наложить ограничения на применение свойств, а просто чтобы предоставить дополнительную информацию приложению, обрабатывающему эти данные. Если ОО язык программирования объявляет класс Book с атрибутом author типа Person, это обычно интерпретируется как набор ограничений (условий применения). ОО язык не позволит создать экземпляр класса Book без атрибута author или указать в качестве значения author объект, не являющийся экземпляром Person. Наконец, он не позволит создать экземпляр Book с каким-то другим атрибутом.

RDF Schema, напротив, предоставляет информацию о схеме как дополнительное описание ресурсов, но не указывает, как это описание должны использоваться приложениями. Приложение вольно по своему усмотрению считать RDF-данные соответствующими схеме или нет, если в описании отсутствует некоторое свойство, требуемое схемой, либо присутствуют свойства, не указанные в схеме. Одно приложение может интерпретировать RDFS-описания как шаблон для генерации данных, и проверять соответствие данных областям значений свойств, то есть интерпретировать описания схемы так же, как они интерпретируются в ОО языке программирования. Другое приложение может интерпретировать RDFS-описания как дополнительную информацию о данных, которые оно получает. Например, если оно получит RDF-данные с указанием свойства ex:author, содержащим значение без указания типа, то может заключить на основе описания схемы, что это значение является ex:Person. Третье приложение может получить данные, в которых свойство ex:author содержит ресурс типа ex:Student, и использовать информацию схемы как базис для предупреждения, что данные могут содержать ошибку. Хотя, возможно, где-то существует RDFS-описание, решающее эту проблему, например, указывающее, что ex:Student подкласс ex:Person.

Итак, RDFS утверждения всегда описательны. Они могут, конечно, интерпретироваться как «предписывающие» (ограничения), но только если приложение желает их так интерпретировать. Всё, что делает RDFS-описание – это предоставляет приложениям дополнительную информацию «для размышления».

Интеграция в Semantic Web систем баз знаний, математической логики и инженерии знаний в состоянии принести двойную прибыль. Во-первых, для механизмов Web (таких как поиск информации) становится доступным большое количество баз знаний, созданных в области медицины, биологической химии и пр. С другой стороны, появляется возможность адаптации к Web самих технологий, наработанных в области математической логики и инженерии знаний, что позволит поисковым системам и программным агентам самостоятельно анализировать предоставленную информацию.

Однако прямой перенос этих технологий невозможен в силу глобальности и децентрализации, которую мы наблюдаем в Web и не наблюдаем в имеющихся системах инженерии знаний. Многие решения (например, [12]) в силу своей концептуальной и физической централизации требуют глобальной целостности ссылок, исходят из предположения об «ограниченности мира». Это же было и с гипертекстовыми системами 70-90 годов, до появления Web. Обобщение KIF для Web должно быть во многом аналогично тому, как были обобщены первые гипертекстовые системы – следует «заменить» локальные идентификаторы на URI и убрать требование глобальной целостности. Кроме того, адаптированные к Semantic Web системы логики должны быть хорошо расширяемы и приспособлены к различными типами «противоречивости» данных.

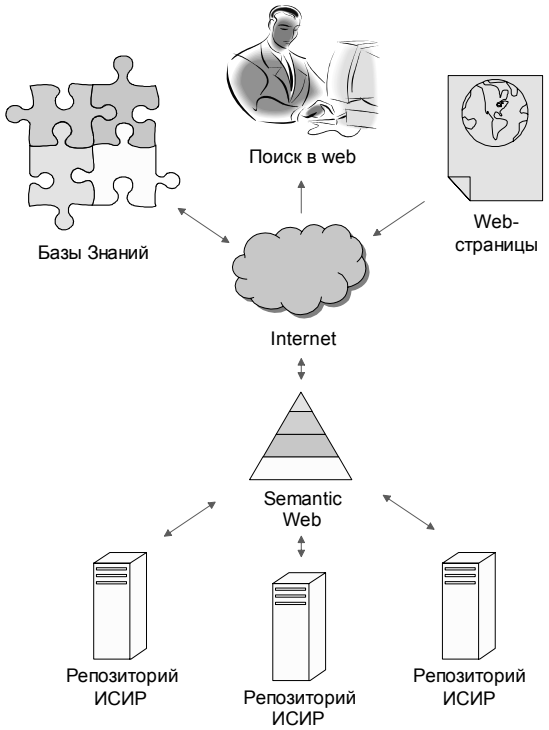
Semantic Web – это то, что мы получим, применив ту же глобализацию к представлению знаний, какая изначально была применена к гипертексту. Достаточно убрать централизованные понятия абсолютной истины, абсолютного знания, полной доказуемости, и посмотреть, что мы можем сделать с ограниченным знанием.

3 Соотношение ИСИР и Semantic Web

ИСИР-технологии [10] ориентированы на формирование единой информационной среды из разнородных и распределенных источников информации, называемых репозиториями, содержащих ресурсы в реляционных и объектных базах данных, LDAP-каталогах, XML и RDF хранилищах, Z39.50-системах и т.п. ИСИР предоставляет ряд служб по поддержке репозиториев, например, репликацию и обмен данными, индексирование и поиск ресурсов, технологию построения Web-порталов для доступа к данным и манипулирования ими. В плане интеграции разнородных информационных источников ИСИР пытается в меру своих возможностей двигаться в направлении целей, намеченных Semantic

Web, используя его технологии. Semantic Web предлагает стандартный механизм и унифицированную модель данных для интеграции информационных источников Web, и ожидает от них поддержки этого механизма. ИСИР предоставляет некоторое техническое решение для построения таких информационных источников, для организации их взаимодействия, как между собой, так и с другими независимыми источниками. ИСИР стремится к реализации идей Semantic Web в своей более узкой и более специализированной области – объектно-ориентированные информационные порталы и цифровые библиотеки.

ИСИР и Semantic Web могут и должны работать вместе. С одной стороны, ИСИР-репозитории служат источниками RDF-знаний для сторонних программных агентов Semantic Web, таких как поисковые системы. С другой стороны, Semantic Web может помочь ИСИР «добывать» информацию из сторонних источников - доступных в Web тезаурусов и классификаторов, баз знаний и пр. систем, с которыми может быть связана информация в информационном портале или цифровой библиотеке. Использование RDF для обмена данными позволяет обмениваться информацией как между репозиториями ИСИР, так и со сторонними заинтересованными системами.



Основой Java-архитектуры ИСИР является объектно-ориентированный подход к представлению данных. Для каждого типа хранилища поддерживается механизм отображения объектной модели данных во внутреннюю модель (реляционную, LDAP…). ИСИР-репозиторий не универсален – он не может, и не должен хранить, «что угодно». Каждый репозиторий способен хранить данные, соответствующие жёсткой замкнутой объектной схеме, описывающей допустимые классы, их свойства, которой сопоставлена система хранимых Java-классов и, например, система таблиц в реляционной БД. Такая объектная схема соответствует традиционной парадигме объектно-ориентированного программирования и объектных баз данных, то есть исходит из понятия «самодостаточности» схемы. Это естественно, так как пока мы находимся в контексте одного репозитория, нет никакой необходимости в «децентрализации» схемы, более того, как правило, модель данных расположенного под объектным уровнем хранилища требует замкнутости схем (реляционная, объектная БД, LDAP..).

Когда же мы выходим на Web-уровень и задаёмся задачей интеграции разнородных репозиториев в Web, мы сталкиваемся с «глобализацией» и «децентрализацией» информации, и логично воспользоваться в этой области парадигмой Semantic Web и языком RDFS. Элементы схемы репозитория становятся глобально-идентифицируемыми (с помощью URI), схемы разных репозиториев перестают быть независимыми. Например, может оказаться так, что некоторые репозитории хранят в себе объекты одного и того же класса, но разные наборы свойств, в соответствии с теми аспектами этого класса, которые им нужны в их специфической предметной области. Может выясниться, что репозитории хранят схожие (но не одинаковые) классы, и при их интеграции часть информации из одного репозитория может использоваться противоположными, например, для пополнения имеющейся у них информации или для установления требуемых взаимосвязей между ресурсами.

Возникает естественное желание как-то интегрировать эти две аспекта в рамках одной схемы данных. С этой целью в ИСИР сформировано расширение языка RDFS, пригодное для описания локального и глобального аспектов объектной схемы данных. Мы используем понятия RDFS-классов и свойств, идентифицируемых URI, понятия наследования классов и свойств, rdfs:range свойства, но добавляем к этим примитивам моделирования возможность четко описать класс в терминах его атрибутов, как это делается в ОО языках программирования. С этой целью введено специальное свойство schema:property - отношение, связывающее класс с его свойствами (задающее «атрибутику» класса).

Каждый репозиторий рассматривает свою схему ресурсов как замкнутое описание, по которому можно получить, например, соответствующее ODL-описание или сгенерировать систему bean-подобных Java-классов. Он интерпретирует rdfs:range как ограничение на тип значений свойства, которому должны удовлетворять хранимые данные, schema:property - как ограничение, чётко задающее все присутствующие в классе атрибуты (которым будут соответствовать get/set методы в java-классах, поля в реляционной БД и т.п.). Репозиторий интерпретирует свою схему ресурсов в рамках традиционной объектной парадигмы, как требуется ИСИР при хранении данных. Это не противоречит идеям Semantic Web, поскольку приложения могут интер-

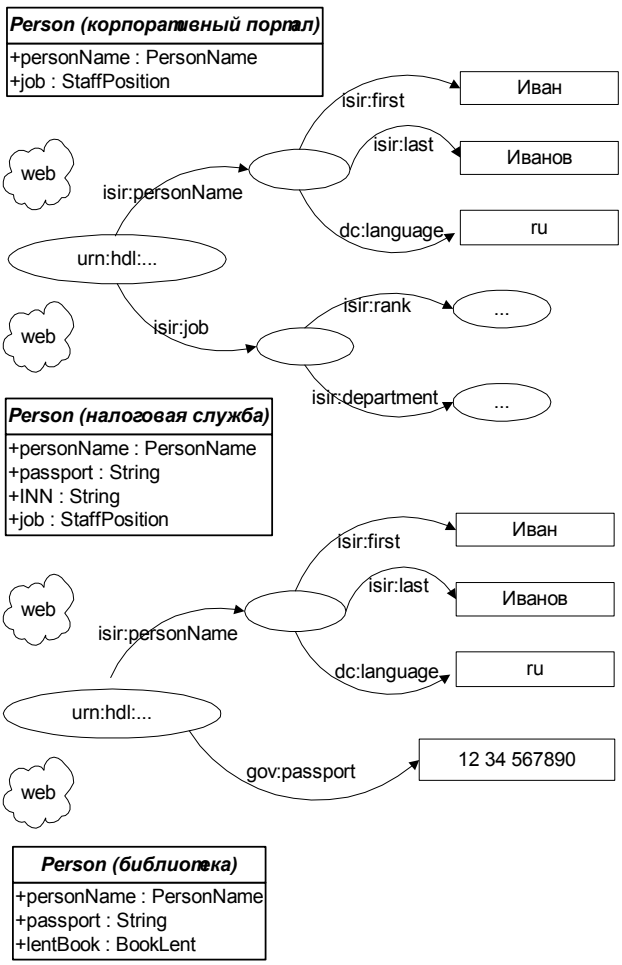
претировать RDFS-описания по собственному усмотрению.

С другой стороны, при обмене данными между репозиториями и интеграции их информации, RDFS-описание объектной схемы каждого репозитория расценивается с позиции децентрализации, распределённости информации в Web. При загрузке сторонних RDF-данных, из них отбирается только та информация, которую репозиторий способен сопровождать согласно своей замкнутой схеме и своим ограничениям. При формировании результатов поисковых запросов поисковый сервис может возвращать пользователям описания ресурсов, интегрирующие свойства разных репозиториев, а, возможно, к тому же отфильтрованные в соответствии с предпочитаемой пользователем схемой ресурсов.

Рассмотрим пример совместной работы трёх информационных источников – корпоративного портала некоторой организации, базы данных налоговой службы и портала библиотеки. Схема объектных данных каждого из этих ИСИР-репозиториев содержит класс isir:Person, представляющий нужную в репозитории информацию о человеке. Пространство имён ‘isir’ соответствует базовой схеме метаданных ИСИР, находящей применение в большинстве приложений. В той же базовой схеме определено свойство isir:personName, указывающее ФИО человека на нужном языке.

Однако каждому репозиторию нужна более специфичная информация о данной личности – корпоративный портал содержит данные о должности человека и контактной информации, которая может быть полезна пользователю портала. Налоговая служба заинтересована, помимо этого, в паспортных данных этого лица, его ИНН и сведениях о доходах. Библиотечная система хранит лишь паспортные данные и информацию о взятых книгах. Соответственно, каждый репозиторий специализирует класс isir:Person под свои нужды, и связывает с ним только те свойства, которые представляют интерес в данной предметной области.

Таким образом, структура класса Person различна для разных репозиториев (на рисунке эти различные «взгляды» на isir:Person изображены в стиле UML). Тем не менее, часть информации в этих источниках пересекается, и один репозиторий может реплицировать нужную информацию из другого. Так, налоговая служба может позаимствовать информацию о месте работы непосредственно из корпоративного портала организации-работодателя, а библиотечная система – паспортные данные из налоговой. Репозитории обмениваются информацией в RDF/XML-представлении, что обеспечивает интероперабельность не только между собой, но и со сторонними системами Semantic Web.



Итак, ИСИР использует RDFS как фундамент для описания объектных схем за простую объектную модель, удобство расширения новыми примитивами и адаптированность к Web, а следовательно и к обмену RDF и XML данными. Онтологические языки, в частности OWL, слишком сложны в силу своей ориентации на системы логики, и не имеет смысла использовать их лишь для описания объектной схемы репозитория и простых ограничений на неё. Но многие онтологии могут быть легко адаптированы под требования ИСИР путём их упрощения, что позволяет создавать цифровые библиотеки, хранящие информацию в соответствии с системой классов и свойств данной онтологии. Приоритетное положение RDFS в ИСИР не мешает изначально описать объектную схему репозитория на ODL [4], смоделировать на UML [9], либо получить по существующей системе Java-классов. ИСИР предоставляет некоторые механизмы по отображению таких описаний объектной схемы друг в друга и их генерации.

ИСИР-технологии не поддерживают средства для хранения произвольных RDF-данных. Каждый репозиторий жёстко ограничен своей схемой предметной области, которая выражается не только в структуре хранимых Java-классов, но и в схеме скрытого под ним хранилища, например в структуре таблиц реляционной БД. Существуют продукты,

нацеленные именно на хранение произвольного RDF [15], которые хранят в БД RDF-тройки (утверждения), и позволяют делать RDF-специфичные запросы к этому множеству троек. Эти системы следуют абсолютно иным целям, чем ИСИР, мало эффективны в работе с большими объёмами данных, более приближены к системам баз знаний, нежели к цифровым библиотекам. При необходимости данные ИСИР-репозитория посредством их RDF-представления могут быть перегружены в подобную систему.

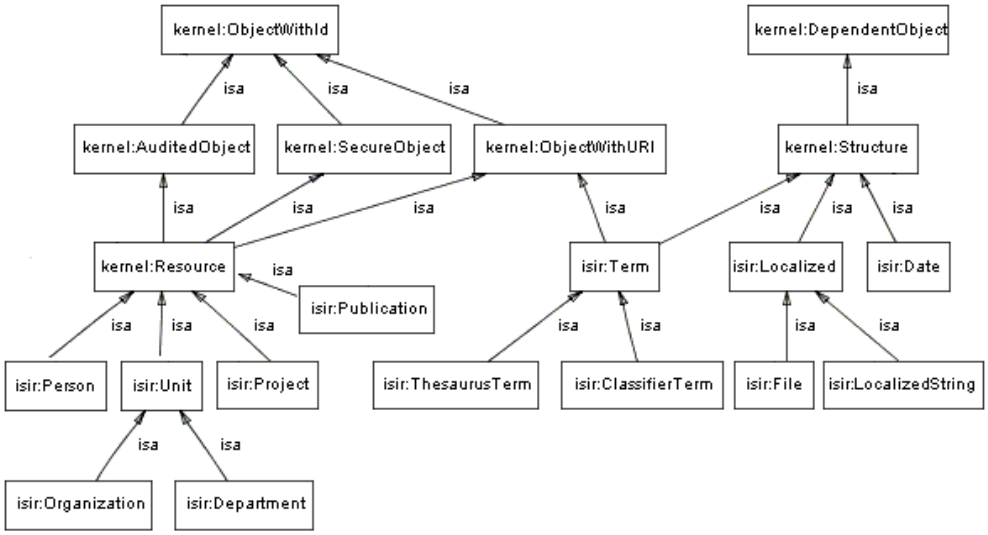
4 Примитивы моделирования RDF Schema в ИСИР

ИСИР расширяет язык RDFS возможностью четкого указания допустимых атрибутов класса (в соответствие с традиционной парадигмой ООП), возможностью выделения линии одиночного наследования в наследовании классов (понятие «абстрактных классов») и некоторыми простыми ограничениями на множество значений свойства.

- schema:Class и schema:Property – подклассы соответственно rdfs:Class и rdf:Property. С этими метаклассами связаны дополнительные свойства.
- schema:AbstractClass – подкласс schema:Class, позволяющий выделить линию одиночного наследования во множественном наследовании классов rdfs:subClassOf. Понятие «абстрактного класса» используется при сопоставлении схемы с теми языками программирования, в которых запрещено множественное наследование, в частности Java. При генерации Java-кода, каждому schema:Class соответствует класс Java, а каждому schema:AbstractClass – Java-интерфейс. Соответственно, генератор не допускает наследования schema:Class от более чем одного неабстрактного класса.
- schema:property – свойство класса schema:Class, сопоставляющее со schema:Class набор допустимых атрибутов schema:Property, таким образом определяя структуру хранимых Java-классов репозитория (согласно традиционной парадигме ООП).
- schema:inverseOf – свойство schema:Property, задающее «обратное отношение» к данному свойству. Например, “isir:job” имеет обратное отношение “isir:hired”, а “isir:author” -обратное отношение “isir:authorOf”. Аналогичное свойство имеется в языках описания онтологий – DAML+OIL, OWL.
- schema:minCardinality, schema:maxCardinality – минимальная и максимальная мощность свойства, то есть ограничения на минимальное и максимальное количество значений этого свойства. В DAML+OIL и OWL эти ограничения указываются в виде локальных ограничений на свойство в классе.
- schema:Attribute – подкласс schema:Property. Свойства, являющиеся экземплярами schema:Attribute, называются «ИСИР-атрибутами» и используются для связывания с объектами (ресурсами или зависимыми объектами) его составных частей, которые не могут существовать независимо от объекта. Такие свойства не могут принимать значения только примитивных типов данных или типов структур (kernel:DependentObject – см. ниже). Понятие «ИСИР-атрибута» соответствует UML-понятиям «атрибута» и «агрегации».

Ядро ИСИР предоставляет различные встроенные услуги по управлению хранимыми объектами, такие как автоматическая проверка прав доступа, аудит и пр. Для того, чтобы к хранимому классу была подключена некоторая услуга, он должен расширить соответствующий предопределённый «абстрактный класс». Так, для подключения системы безопасности и возможности конфигурации персональных прав доступа к объектам класса, достаточно сделать класс наследником kernel:SecureObject. Благодаря использованию множественного наследования, к классу можно подключить несколько услуг одновременно (например, «зависимый объект» и персональные права доступа).

- kernel:ObjectWithId – такой объект имеет уникальный в пределах хранилища числовой идентификатор - id. Большинство сервисов ядра, такие как security, audit и пр., нуждаются в том, чтобы объект обладал таким id.
- kernel:SecureObject – такой объект обладает персональными правами доступа – Access Control List, указывающий, каким пользователям какие привилегии над этим объектом даны. Достаточно наследовать свой класс от SecureObject – и он попадёт под политику безопасности ядра.
- kernel:AuditedObject – наследование от этого «абстрактного класса» подключает автоматический аудит изменений объекта.
- kernel:DependentObject – экземпляры подклассов этого «абстрактного класса» расцениваются ядром как зависимые объекты. Такой объект имеет неявное обязательное поле scope – «контекст», связывающее его с объемлющим хранимым объектом. Зависимый объект не может существовать без контекста. При создании зависимый объект должен быть проассоциирован с допустимым объектом-контекстом, иначе он не получит статус хранимого. При удалении объекта-контекста автоматически удаляется и зависимый объект. Ассоциация происходит при присвоении зависимого объекта некоторому свойству объекта-контекста, принимающему только одно значение, либо при добавлении его в коллекцию значений многозначного свойства объекта-контекста. Свойства, проводящие ассоциацию зависимого и объемлющего объектов, называются ИСИР-атрибутами (schema:Attribute). На тип объекта-контекста не накладывается ограничений – он и сам может быть зависимым объектом от некоторого



третьего хранимого объекта. Для чтения, модификации, удаления зависимого объекта необходимо иметь аналогичное право для объекта-контекста (чтение контекста для чтения зависимого, модификация контекста для модификации зависимого, модификация или удаление контекста для удаления зависимого). При изменении зависимого объекта меняется и дата модификации объекта-контекста.

- kernel:ObjectWithURI – такой объект может обладать URI – строкой, однозначно идентифицирующей его в контексте данного хранилища, а иногда (в зависимости от схемы URI) и в глобальном аспекте. URI используется в RDF-сериализации для того, чтобы ссылаться на этот объект.

Помимо указанных абстрактных классов, вводится два типичных неабстрактных базовых класса, удобных для использования в большинстве информационных систем – это «ИСИР-ресурс» и «структура».

- kernel:Resource - логическая единица хранения, условная единица информации, наделенная «самостоятельностью», «индивидуальностью» в том смысле, что она может существовать вне зависимости от других информационных объектов, представлять первичный интерес пользователей системы. Каждый ресурс относится к некоторому типу ресурсов, который определяет некоторую совокупность сведений, внутренне присущих ресурсам этого типа, но не рассматривающих взаимосвязи с другими ресурсами. Ресурс обладает глобально уникальным идентификатором, относительно которого известен и поддерживается механизм разрешения идентификатора в протокол и адрес доступа к ресурсу, например, Handle System.
- kernel:Structure - структурное значение («зависимый объект»), существующее только в контексте логических единиц хранения (ресурсов). Простой не-абстрактный подкласс kernel:DependentObject.

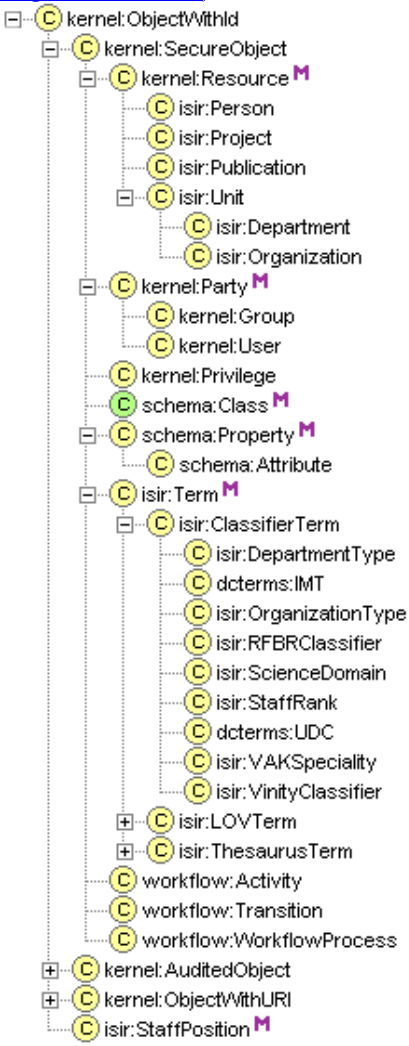
5 Базовая схема метаданных ИСИР РАН

При описании RDFS-схемы прикладных классов для ИСИР РАН был сформирован набор прикладных классов и свойств, характерных для многих информационных систем. Примеры включают базовую информацию о документах, организациях, людях и их деятельности.

Все типы информационных объектов в базовой схеме ИСИР делятся на «ресурсы» (kernel:Resource), такие как организации и подразделения, персоналии, проекты, публикации, и «структуры» - зависимые объекты, представляющие собой часть информации некоторого ресурса. Структуры используются для группировки атрибутов ресурсов в отдельные объекты, они связываются с ресурсами свойствами типа schema:Attribute. Отдельный тип информационных объектов – это «термы», объединяющиеся в «таксономии» - централизованные словари, классификаторы, тезаурусы ИСИР.

В базовой схеме вводится механизм для выражения многоязычной информации – такая информация выражается в виде отдельных «локализованных» объектов, содержащих текстовую информацию на конкретном языке. Класс isir:File базовой схемы предназначен для отражения длинной текстовой или бинарной информации. Конкретный способ хранения этой информации зависит от конфигурации системы и указанного «режима». Файл может представлять собой как web-ссылку (URL), по которой может быть получено его содержимое, так и CLOB или BLOB в реляционной БД (когда хранилищем является RDBMS), либо файл локальной файловой системы. С файлом может быть связана различная метаинформация, такая как размер, MIME-тип и пр. Описанная функциональность класса File обеспечивается за счёт скрытой в нём прикладной логики. «Файловая» информация широко используется в конкретных прикладных системах (документы). Компоненты Web-архитектуры в ИСИР обеспечивают загрузку файлов на сервер, организацию работы с zip-архивами, например, для хранения html-документа с картинками.

Полное описание RDF-схемы ИСИР весьма объёмно и здесь не приводится. Это описание может быть найдено в проектной документации. На следующем рисунке показаны фрагменты иерархии классов и свойств ИСИР РАН, визуализированные редактором онтологий Protégé (http://protege.stanford.edu/).



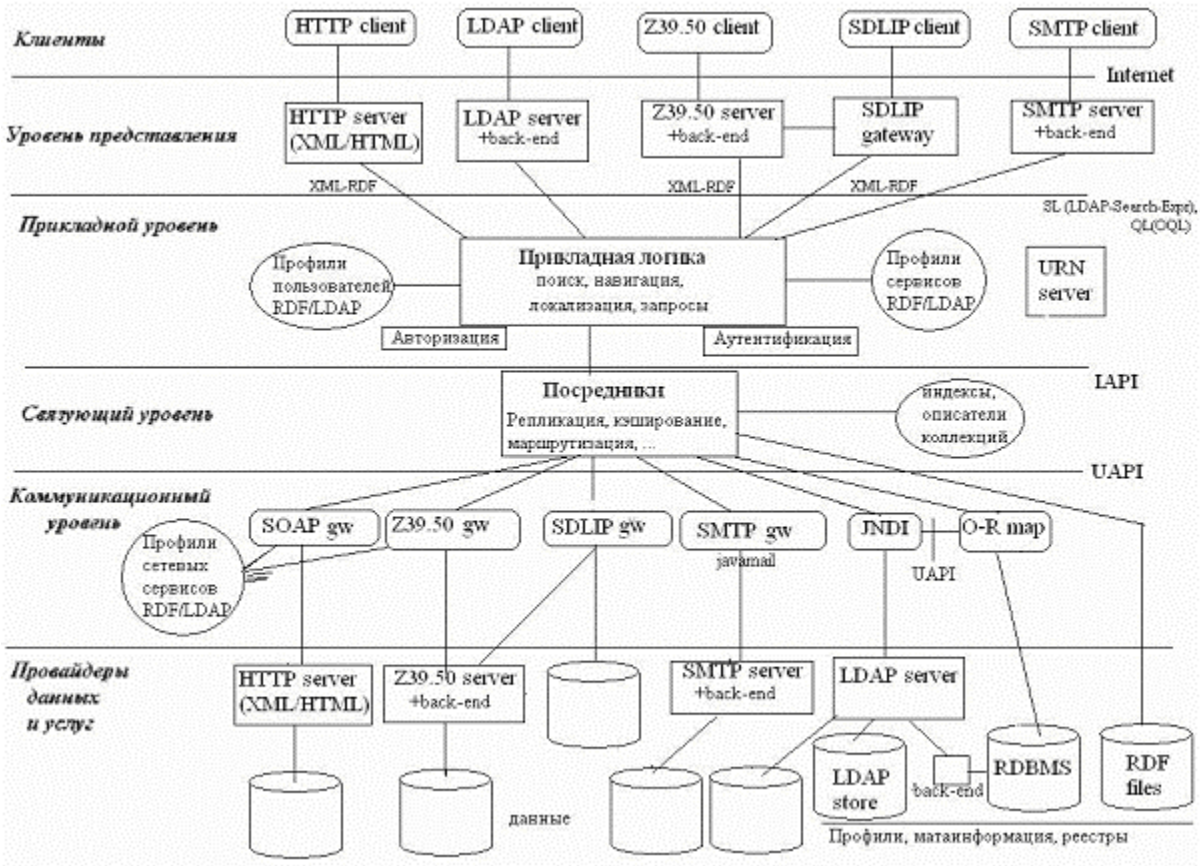
6 Абстрактная архитектура

Технологии ИСИР [10] ориентируются на реализацию ключевых средств корпоративных порталов. Корпоративный портал выступает в роли посредника, направляющего обращения пользователей к совокупности сервисов, релевантных данной тематический области, используя открытые прикладные протоколы, такие как HTTP, SOAP, Z39. 50, SDLIP, LDAP и д.р. Среди основных задач портала обеспечение операций обнаружения ресурса, локализации его местоположения, запроса и доставки ресурса.

Реализация исходит из общей многоуровневой архитектуры [3]. Цель выделения уровней, в которые в свою очередь содержат компоненты и модули, состоит в том, чтобы обеспечить локальное упрощение при поддержке сложных функциональных возможностей. Каждый уровень имеет собственные цели и абстракции, взаимодействие между уровнями ограничено. Модульная организация обеспечивает возможности простого расширения функциональных возможностей системы, простую интеграцию новых высокоуровневых сервисов с существующими. Общая архитектура определяет согласованные интерфейсы между уровнями, интерфейсы для поддерживаемой совокупности протоколов, используемых как клиентами системы (например, HTTP, SOAP, Z39.50, LDAP), так и провайдерами данных и услуг (например, HTTP, OAIP, SOAP, Z39.50, SDLIP, LDAP, FTP), интерфейсы для базового множества операций таких, как поиск, локализация, запрос, доставка. Общая архитектура выделяет следующие уровни.

Уровень представления. Этот уровень отвечает за представление информации пользователям и обеспечение пользовательского ввода. Информация, вводимая пользователем, передается вниз на прикладной уровень, что может осуществляться посредством создания программных объектов для передачи их через согласованные API/интерфейсы или с помощью кодирования данных в специфическом транспортном формате (сериализации), например XML.

Прикладной уровень. Прикладной уровень обеспечивает реализацию прикладных операций, необходимых пользователям или программным агентам в рамках системы. Он отвечает за поддержку логики приложений системы. Прикладные операции - это высокоуровневые сервисы, реализуемые на основе низкоуровневых функций связующего программного обеспечения (промежуточного уровня). Прикладной уровень может поставлять на уровень представления и собственные сервисы, не зависящие от нижних



уровней, например, сервис персонификации. Уровень отвечает за поддержку пользовательских профилей, за ведение сессий пользователей. Может приспосабливать информационное окружение под определенные сообщества пользователей. В его задачу входит адаптация запросов и их результатов к текущей ситуации, определяемой событиями сессии и профилем пользователя.

Связующий уровень. Этот уровень (уровень промежуточного связующего программного обеспечения - посредника) ответственен за понимание значения сервисов, запрашиваемых прикладным уровнем, и сервисов, предоставляемых провайдерами. Связующий уровень, получая запросы от прикладного уровня, должен определить, какие из провайдеров услуг могут удовлетворить запрос, возможно, сложная комбинация из них. Связующий уровень может обеспечивать управление авторскими правами.

Коммуникационный уровень обеспечивает единообразное представление провайдеров данных, использующих разные сетевые возможности. Он ответственен за коммуникации с внешними сервисами, скрывает от связующего уровня такие подробности, как коммуникационные протоколы, расположение внешних сервисов. Он может обеспечивать отображение между словарями метаданных, чтобы поддержать термины, понятые связующему уровню. В некоторых случаях сервисы провайдеров (внешних сервисов) могут «непосредственно» взаимодействовать со связующим уровнем, в таких случаях действие, выполняемое коммуникационным уровнем, тривиально. Коммуникационный уровень обеспечивает связь между связующим уровнем и провайдерами, основываясь на профиле сетевого сервиса, связанным с каждым сервисом, предоставляемым поставщиками. Профиль сетевого сервиса предоставляет информацию о расположении, протоколе, языке запросов и форматах ответов и словарях метаданных, требуемых для осмысленного доступа к внешнему сервису.

Уровень провайдеров данных и услуг. Уровень провайдеров данных и услуг включает все внешние сервисы, предоставляемые провайдерами в соответствии с профилями сетевых сервисов. Уровень включает "первичные" сервисы, на основе которых функционирует система. Он также включает "вторичные" сервисы, используемые системой при предоставлении "первичных" сервисов. Это, например, могут быть реестры схем метаданных, сервисы аутентификации, пользовательские профили.

Ключевую роль в архитектуре играют репозиторный сервис и механизмы глобальной идентификации и контроля доступа. Основываясь на услугах репозиторного сервиса, сервисы следующих уровней – репликационный, индексный, поисковый, метаинформации, составляющие совместно с базовыми стек сервисов сервера, обеспечивают обмен данными и поиск в распределенной среде.

7 Ядро ИСИР

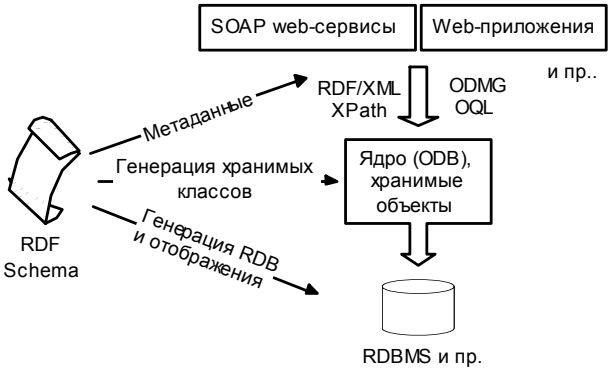
Технологии ИСИР ориентируются на формирование единой корпоративной информационной системы из разнородных хранилищ и источников информации в распределенной среде, включая объект-

ные и реляционные базы данных, LDAP-каталоги и пр. Каждое полнофункциональное хранилище называется репозиторием. ИСИР предоставляет многочисленные службы по поддержке репозиториев, например репликацию и обмен данными, индексирование и поиск, технологию построения web-порталов для доступа к данным.

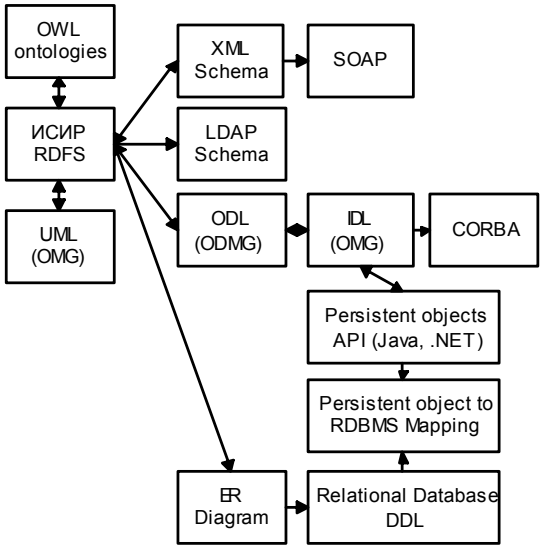
Основой новой версии ИСИР является объектно-ориентированный подход к представлению данных. Использование такого подхода унифицирует модель хранимых данных, облегчает процесс интеграции с Semantic Web, XML-технологиями, стандартами OMG и ODMG. Использование понятия «хранимых объектов» в объектно-ориентированном языке программирования (на платформах Java, .NET) позволяет разработчикам прикладных приложений на базе ИСИР-технологий абстрагироваться от ненужных деталей и сконцентрировать своё внимание собственно на логике приложения.

Мы исходим из того, что репозиторий не должен быть универсальным - он не может, и не должен хранить произвольную структурированную информацию. Репозиторий специализируется на своей предметной области и способен хранить данные, соответствующие собственной объектной схеме, которая описывает используемые классы и свойства. Ядро ИСИР обеспечивает механизм отображения объектной модели данных во внутреннюю модель данных используемого хранилища (реляционную, LDAP…). Так, в реляционной базе данных объектной схеме соответствует структура таблиц. С помощью такого механизма отображения ядро фактически превращает низлежащее хранилище в объектную базу данных и позволяет прикладному коду работать с «хранимыми объектами», изменения в которых прозрачно отображаются в хранилище. На данном этапе подобный механизм отображения поддерживается для всех распространённых реляционных СУБД, разрабатывается отображение в LDAP. Кроме того, Sun Microsystems предлагает стандартную архитектуру Java Data Objects для прозрачного хранения Java-объектов в различных базах данных и иных системах. Любая сторонняя реализация этой спецификации может быть подключена в Ядро ИСИР.

Как фундамент для описания объектных схем ИСИР использует язык RDF Schema - за простую объектную модель, удобство расширения новыми примитивами, адаптацию к Web и преимущества, приносимые при обмене RDF/XML данными. ИСИР расширяет RDFS необходимыми примитивами, которые позволяют эффективно использовать этот язык для описания объектной схемы данных репозитория ИСИР. Каждый репозиторий рассматривает свою RDF-схему как замкнутое описание собственной объектной схемы данных, которой соответствует структура «хранимых» java-классов и схема хранилища, например реляционной БД. С другой стороны, использование RDFS и технологий Semantic Web приносит существенные выгоды при интеграции различных репозиториев в единую информационную среду, взаимодействии со сторонними информационными источниками.



Приоритетное положение RDFS в ИСИР не мешает изначально описать объектную схему репозитория на ODL [4], смоделировать на UML [9], либо получить по существующей системе Java-классов. ИСИР предоставляет некоторые механизмы по отображению таких описаний друг в друга и их генерации.



Генератор Java-классов позволяет получить по ИСИР RDFS-описанию исходный код bean-подобных «хранимых классов». В эти классы вручную может быть заложена любая бизнес-логика, заменяющая или дополняющая исходное поведение. При изменении схемы (например, добавлении свойств) будет произведена инкрементная перегенерация классов – внесённые в код изменения будут сохранены. Таким образом, не ограничивая функциональных возможностей системы, RDFS позволяет автоматизировать большинство операций. Ядро и сервисы ИСИР параметризуются объектной схемой, и способны работать с любой нужной предметной областью.

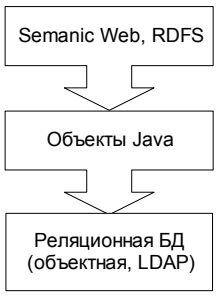
Генератор реляционной БД позволяет получить по ИСИР RDFS-описанию SQL DDL-скрипт для создания таблиц, в которых будут храниться данные, и описание объектно-реляционного отображения. Процесс генерации является настраиваемым, и дизайнер может явно указать, какие примитивы реляционной БД необходимо использовать для задан-

ных примитивов объектной схемы в том или ином случае, например, какое решение применить для поддержки наследования классов. Настройки генератора указываются в RDF-формате вместе с объектной схемой, благодаря расширяемости RDFS. В случае, когда необходимо настроить ИСИР-систему на имеющееся унаследованное хранилище, необходимо вручную описать аналогичное отображение. Благодаря настраиваемому отображению объектов в модель данных хранилища, ИСИР позволяет вывести унаследованные базы данных на новый уровень, открывая к ним доступ ведущим Web-технологиям и Semantic Web.

Как уже упоминалось, ИСИР-приложения, фактически, работают с объектной базой данных, надстроенной над реальным хранилищем. Доступ к такой объектной БД предоставляется репозиторным сервисом ядра. Интерфейс этого сервиса является подмножеством стандартного интерфейса объектных баз данных ODMG, который позволяет извлекать нужные объекты OQL-запросами, управлять транзакциями. Приложение работает с полученными из репозиторного сервиса «управляемыми» объектами как с обычными объектами языка Java - вызывает их методы, меняет свойства - и при успешном завершении транзакции изменения отражаются в хранилище.

Ядро предоставляет различные встроенные услуги по управлению хранимыми объектами, такие как автоматическая проверка прав доступа, аудит и пр. Для того чтобы к хранимому классу была подключена некоторая услуга, он реализовывать соответствующий маркер-интерфейс. К одному классу могут быть подключены несколько услуг одновременно (например «зависимый объект» и персональные права доступа). В ИСИР RDFS подключению услуг соответствует наследование классов от предопределённых «абстрактных классов» ядра.

8 Высокоуровневые сервисы



Ядро ИСИР позволяет разработчику абстрагироваться от конкретного устройства хранилища и работать с хранимыми Java-объектами, соответствующими объектной схеме репозитория. Многие службы, тем не менее, нуждаются в более общей абстракции – они не должны зависеть от конкретной объектной схемы, а должны быть способны работать с разными репозиториями и схемами. Такие службы работают не напрямую с репозиторным сервисом ядра и хранимыми Java-объектами, а с абстрактным программным интерфейсом Unified API (UAPI).

UAPI представляет собой набор простых Java-интерфейсов, ориентированных на работу с RDF-совместимыми данными. Схемонезависимый прикладной код использует переданное ему описание RDF-схемы и Unified-интерфейс к репозиторию для работы с хранилищем.

UAPI предельно прост в реализации, что позволяет применить часть архитектуры ИСИР ко внешним источникам информации. Это позволяет организовать обмен данными, репликацию, индексирование и поиск не только для репозиториев, построенных на базе ядра ИСИР, но и для внешних источников, над которыми реализован UAPI.

На базе ядра и UAPI строятся более высокоуровневые службы ИСИР:

- Сервис сериализации обеспечивает возможность выгрузки хранимых объектов из репозитория в XML/RDF-файл.
- Сервис десериализации, наоборот, позволяет загрузить в репозиторий XML/RDF-файл. Поддерживаются гибкие возможности по модификации уже имеющихся в репозитории ресурсов пришедшими в файле данными, по интеграции пришедших объектов с имеющимися.
- На базе сервисов сериализации/десериализации строится сервис репликации данных между репозиториями, распределенными в Интернет. Этот сервис обеспечивает обмен XML/RDF-данными измененных ресурсов на базе протокола CIP. Используются следующие типы взаимодействия репозиториев: «рассылка» данных при наступлении заданного события (Push), «вытягивание» данных при наступлении заданного события (Poll), обмен на основе согласования действий уведомлениями (Notify).
- Сервис глобальной идентификации, базирующийся на Handle System, обеспечивает глобальную идентификацию некоторых ресурсов репозитория в Интернет с помощью handle-строки, позволяет найти репозиторий-владелец ресурса, владельцев его копий и реплик, получить доступ к ресурсу через web и пр.
- Сервис индексирования обеспечивает построение атрибутно-полнотекстового IDF-индекса (инвертированные списки, хранимые и обслуживаемые РСУБД) по хранимым объектам репозитория, а также RDF/XML данным, предоставленным сервисом сериализации для внешних информационных источников, либо по текстовым и HTML-файлам. Индексатор использует нормализацию индексируемых слов, поддерживает русский и английский языки. Служба индексации параметризуется UPI или файлом с RDF-данными и RDFS-описанием схемы данных, указывающей также, какие свойства объектов следует индексировать.
- Сервис поиска по атрибутно-полнотекстовому индексу, принимая LDAP-подобное логическое выражение, преобразует его в SQL-запрос к таблицам индекса и ранжирует выбранные запросом объекты в соответствие с частотными характеристиками вхождений слов.

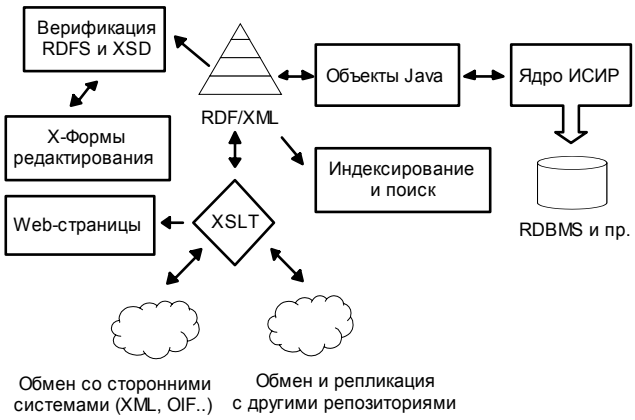
9 XML-технологии в ИСИР

Архитектура ИСИР повсеместно использует XML и сопутствующие технологии. XML-представление данных является такой же неотъемлемой частью ИСИР, как и объектное представление. XML-представление объектных данных позволяет применить к ним всю мощь XML-технологий и интегрировать в архитектуру различные разработки, связанные с XML-технологиями.

Представление объектной информации в виде XML в ИСИР опирается на идеи и механизмы Semantic Web. Этот W3C-проект являет собой логическое продолжение развития Web – от гипертекстовых страниц к структурированным XML-документам, а затем к смысловому содержанию и адекватной машинной интерпретации данных. Модель данных Sematic Web представляет собой выражение объектной и ER-моделей данных в глобальном аспекте Web – объекты и их свойства идентифицируются здесь с помощью URI. Semantic Web определяет принципы записи таких данных в XML (RDF/XML-синтаксис), и ИСИР использует эти принципы для сопоставления XML-схемы и XML-представления объектной схеме и объектному представлению данных.

Представление объектной информации в XML в согласии с принципами Semantic Web приносит ряд существенных преимуществ при обмене этой информацией для интеграции данных различных репозиториев. Структура такого (RDF/XML) файла чётко соответствует объектной схеме репозитория ИСИР, объекты которого представлены в данном XML. Таким образом, RDF-файл несёт в себе семантику сериализованных данных, а не только данные. Получатель сможет правильно сопоставить сериализованные данные с собственной системой классов и свойств, и адекватно их обработать. Это как раз то преимущество, которое несёт Semantic Web – семантическая интероперабельность.

ИСИР использует XML для представления хранимых данных, когда необходимо воспользоваться услугами технологий, базирующихся на XML (XSLT и XForms, RDF/XML для обмена данными, SOAP). В многоуровневой архитектуре ИСИР XML-представление данных стоит на следующей ступеньке после Semantic Web и объектного представления, которое в свою очередь стоит над представлением данных в соответствие с типом конкретного хранилища (ODBMS, RDBMS, LDAP..). Уровень Semantic Web представлен интерфейсом UAPI, а с XML-представлением данных имеют дело сервисы сериализации и десериализации.



На базе этих сервисов строится сервис репликации данных между репозиториями, распределенными в Web. Служба индексации ИСИР также связана с XML и Semantic Web – она обеспечивает построение атрибутно-полнотекстового индекса по RDF/XML-данным индексируемых объектов. ИСИР поддерживает возможности доступа к репозиторию по протоколу SOAP, то есть позволяет организовать Web-сервис на базе репозитория. Среди функций такого web-сервиса – сериализация в RDF/XML хранимых объектов, выбранных поисковым запросом, и модификация хранимых объектов согласно предоставленному клиентом RDF/XML.

10 Web-публикация

Упомянутые выше службы ориентированы в большей степени на обмен или обработку данных, и используют XML для представления этих данных в сериализованной форме. Другой аспект использования XML-технологий в ИСИР – это web-публикация информации с применением XSLT. Использование XML/XSLT для построения ИСИР Web-порталов позволяет разделить логику выборки данных и стиль их представления, получить возможность представлять результат в различных форматах помимо HTML (WML для сотовых телефонов, PDF для печати, Excel для отчётов, SVG для графиков, RDF для метаданных). Для каждой «активной страницы» портала необходимо отдельно описать:

- Какие данные будут показаны на странице (логику выборки).
- Как оформить эти данные, чтобы получить требуемый HTML, PDF (стиль представления). При желании у страницы может быть несколько стилей, рассчитанных на разных пользователей, разный контекст.

При разработке всех компонентов архитектуры ИСИР мы придерживаемся общей стратегии следования открытым стандартам и общепризнанным решениям. В плане XML-изации web-сервера с применением XML/XSLT-технологий и разделением данных, логики и представления в Java Community отлично зарекомендовал себя Apache-проект Cocoon [1]. Cocoon опирается на простую конвейерную модель обработки: XML-документ проходит через конвейер, который представляет

собой несколько фаз преобразования данных. Каждый конвейер начинается генератором (точка средоточения логики и данных), содержит несколько преобразователей (например, XSLT), и завершается сериализатором. По конвейеру происходит постепенное перемещение потока XML-информации, с преобразованием её от исходного XML данных (content) к нужному XML представления.

Генератор Преобразователь Сериализатор

Простейший сериализатор преобразует SAX-поток в данные в формате HTML, WML, SVG, VRML, PDF, PostScript, Microsoft Word и т.п. Cocoon обобщает понятие «активных серверных страниц» на случай XML/XSLT-публикации данных, вводя механизм XML-серверных страниц (Extensible Server Pages - XSP). Структура «XSP-страниц ИСИР» чётко соответствует ИСИР RDF-схеме репозитория – это залог семантической интероперабельности распределённых в web приложений. Для иллюстрации ниже приведён небольшой пример

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="..."
         xmlns:isir="urn:hdl:1016.1/core/">
<isir:Organization>
  <!-- описание объекта - организации -->
  <!-- свойство организации - название -->
  <isir:unitName>
    <isir:full>
      НИИ Чародейства и Волшебства
    </isir:full>
    <isir:abbrev> НИИ ЧАВО </isir:abbrev>
  </isir:unitName>
  <isir:department>
  <!-- описание подразделения -->
    <isir:unitName>
      <isir:full> Дирекция </isir:full>
    </isir:unitName>
    <isir:staff>
      <!-- описание штатной единицы -->
      <isir:rank>
        <isir:name>
          <isir:full> директор </isir:full>
        </isir:name>
      </isir:rank>
      <isir:hired>
        <!-- информация о нанятом человеке -->
        <isir:personName>
          <isir:first> Янус </isir:first>
          <isir:middle>
            Полуэктович
          </isir:middle>
          <isir:last> Невструев </isir:last>
        </isir:personName>
      </isir:hired>
    </isir:staff>
    <isir:staff>
      <!-- описание штатной единицы -->
      <isir:rank>
        <isir:name>
          <isir:full>
            заместитель директора по

            административно-хозяйственной части
          </isir:full>
        </isir:name>
      </isir:rank>
      <isir:hired>
        <!-- информация о нанятом человеке -->
        <isir:personName>
          <isir:first> Модест </isir:first>
          <isir:middle>Матвеевич</isir:middle>
          <isir:last> Камноедов </isir:last>
        </isir:personName>
      </isir:hired>
    </isir:staff>
  </isir:department>
</isir:Organization>
</rdf:RDF>
```

«XSP-страницы ИСИР» соотносятся с RDF/XML-данными точно так же, как JSP соотносятся с HTML. «XSP-страницы ИСИР» отвечают за динамическую генерацию RDF/XML-документов, то есть за выборку необходимой объектной информации из хранилища и представление её в RDF/XML-виде. В ИСИР XSP-страницы позволяет максимально наглядно и декларативно описать, какие данные следует выбрать. Такая страница похожа на шаблон XML-документа, в который осталось лишь подставить значения из хранилища.

11 Управление потоками работ

В наше время происходит стремительный рост числа информационных Web-порталов и цифровых библиотек, в которых все большее и большее применение находят автоматизированные системы управления потоками работ. С их помощью становится возможным увеличение качества и объемов выполняемых работ по манипулированию информационным наполнением и сервисами Web-порталов и цифровых библиотек. С появлением сложных Web-систем, цифровых библиотек, корпоративных порталов, поддерживающих и управляющих большими объемами и потоками информации, сформировалось категория подсистем, называемых системами управления содержанием/контентом Web-систем (WCMS - Web Content Management System). Первые WCMS представляли совокупности Web-форм для управления данными – как несвязанных Web-форм, так взаимосвязанных web-форм, но с фиксированным в программном коде порядком обработки. Сейчас актуальной становится потребность в более гибкой организации управления потоками работ, в поддержке декларативных форм определения и оперативного изменения потоков работ. WCMS должны управлять созданием и модификацией информационных, сильно взаимосвязанных ресурсов произвольных типов. Потоки работ могут быть связаны не только с поддержкой информационных ресурсов, но и с выполнением полностью автономных регламентных процедур системы в ответ на, например, программные обращения, на наступление некоторого события и т.п.

Репозитории ИСИР должны обладать гибкими средствами для поддержания актуальности имеющихся данных: пакетной загрузки объемов информации, автоматизированными средствами создания и редактирования ресурсов. Редактирование и создание отдельных ресурсов с помощью форм требует наличия системы, управляющей декларируемыми

потоками работ пользователей. XML и RDF загрузка решают задачу о внесении в базу данных больших объемов информации, но после нее требуется участие администраторов данных, отвечающих на запросы системы по нарушениям форматов, разрешению неоднозначности, выявлению дубликатов, которое также должно быть осуществляться в рамках ролевых, контролируемых потоков работ. Поэтому возникла задача создания службы управления потоками работ.

Существующие на данный момент модели описания потоков работ разбиваются[18] на две категории:

- Графовые модели (XPDL[19,20], WSFL[21]), наглядно отражающие древовидную структуру процесса.
- Блочные модели (BPML[17], XLANG[22]), наиболее приближенные к блочной структуре языков программирования.

Проведенный сравнительный анализ двух видов моделей на примере их основных представителей (XPDL и BPML) с точки зрения организации в них основных конструкций, определяющих модель потоков работ, показал, что все основополагающие конструкции блочной модели BPML могут быть выражены через их прямые аналоги модели потоков работ XPDL. Между тем спецификация XPDL обладает рядом очевидных преимуществ.

На основании проведенного анализа за базовый стандарт описания модели рабочих процессов по редактированию ресурсов репозитория ИСИР выбрана спецификация XML Process Definition Language (XPDL) ввиду наиболее полного соответствия требованиям к функциональности создаваемого решения. Была создана служба управления потоками работ по редактированию ресурсов репозиториев ИСИР, ориентирующаяся на это предложение. Данная служба использует сервисы ИСИР для хранения объектной модели рабочих процессов в базе данных, а также службу аутентификации и управления пользовательскими правами доступа к объектам процессов. Для декларативного описания новых типов потоков работ используется часть спецификации XPDL, обеспечивающая необходимую функциональность. Пользовательские интерфейсы системы управления потоками работ построены на базе Java-технологий ИСИР, автоматизирующих процесс построения Web-форм для редактирования ресурсов.

Общая объектная модель процесса представлена на следующей диаграмме:

Поддержка и следование службой управления рабочими процессами по манипулированию ресурсами репозитория ИСИР ряду соответствующих стандартов делает возможным ее применение в системах управления контентом Web-порталов, системах электронного документооборота и других сервисах, автоматизирующих управление потоками работ, распределенных между разнообразными участниками производственных процессов. Широчайшая область применения служб подобного типа дает не менее широкие предпосылки к дальнейшему развитию и модернизации данного проекта. К ближайшим шагам по дальнейшей разработке и модернизации компонентов системы можно отнести следующее:

- Дополнение выбранного подмножества элементов XPDL конструкциями, поддерживающими рекурсивные процессы и задания (например, группы заданий BlockActivity и ActivitySet, а так же подпроцессы SubFlow). Расширение набора управляющих атрибутов созданной объектной модели.
- Реализация модели пакетов XPDL с возможностью экспорта и импорта элементов описания.
- Добавление остальных компонентов объектной модели XPDL с поддержкой их хранения в реляционной базе данных.
- Поддержка исключительных ситуаций и их обработчиков с возможностью оповещения как исполнителей, так и администратора системы. Реализация как системных, так и пользовательских обработчиков исключений.
- Реализация некоторого унифицированного интерфейса для общения системы с внешними Web-приложениями произвольного типа (возможно, базирующегося на описании интерфейсов Web-служб с помощью WSDL).
- Поддержка условных переходов между заданиями и условных циклических конструкций языка описания рабочих процессов XPDL.

- Поддержка множественных переходов с одного задания (параллельный запуск нескольких заданий с возможностью их синхронизации).
- Реализация подсистемы управляющей версиями хранимых объектов, поддерживающей как ведение версий отдельных атрибутов объекта, так и объекта целиком с учетом связей с другими ресурсами.
- Реализация подсистемы управления транзакциями заданий на базе службы ведения версий хранимых объектов.
- Усовершенствование подсистемы протоколирования операционных состояний объектов рабочего процесса. Добавление возможности ведения протоколов на различных физических носителях. Интеграция со службой аудита хранимых объектов «ядра».
- Модернизация Web-интерфейсов системы управления потоками работ.
- Выбор и применение одного из существующих средств визуального проектирования для создания XPDL-описаний рабочих процессов.

К задачам второстепенной важности можно отнести поддержку возможности экспорта XPDL-описаний с предварительным преобразованием к другим существующим стандартам декларации потоков работ и реализацию распределенной системы управления рабочими процессами.

Литература

[1] Apache Cocoon Project. http://cocoon.apache.org/2.0/
[2] Dublin Core Activity. http://dublincore.org
[3] MIA Development: Architecture and Functional Model, Tracy Gardner UKOLN, University of Bath.
[4] Object Database Management Group. http://www.odmg.org
[5] OWL Web Ontology Language 1.0 Reference. W3C Working Draft. http://www.w3.org/TR/owl-ref/
[6] RDF Primer. W3C Working Draft. http://www.w3.org/TR/rdf-primer
[7] RDF Vocabulary Description Language 1.0: RDF Schema. W3C Working Draft. http://www.w3.org/TR/rdf-schema
[8] Semantic Web Activity. http://www.w3.org/2001/sw
[9] OMG UML Resource Page. http://www.omg.org/uml/
[10] Бездушный А.Н., Жижченко А.Б., Кулагин М.В., Серебряков В.А. Интегрированная система информационных ресурсов РАН и технология разработки цифровых библиотек. Программирование V 26, N 4, 2000, pp. 177-185
[11] DAML Language. http://www.daml.org/about.html
[12] Knowledge Interchange Format, Genesereth M. draft proposed American National Standard NCITS.T2/98-004. http://logic.stanford.edu/kif/dpans.html
[13] Ontology Inference Layer. http://www.ontoknowledge.com/oil
[14] RDF/XML Syntax Specification (Revised). W3C Working Draft. http://www.w3.org/TR/rdf-syntax-grammar/
[15] Dave Beckett, Jan Grant. Semantic Web Scalability and Storage: Mapping Semantic Web Data with RDBMSes. http://www.w3.org/2001/sw/Europe/reports/scalable_rdbms_mapping_report
[16] Tim Berners-Lee. What the Semantic Web can represent., 1998. http://www.w3.org/DesignIssues/RDFnot.html
[17] BPML working draft March 25, 2002. http://www.bpmi.org, http://xml.coverpages.org/bpml.html
[18] Robert Shapiro. A Comparison of XPDL, BPML and BPEL4WS. http://xml.coverpages.org/Shapiro-XPDL.pdf
[19] Workflow Management Coalition standards. http://www.wfmc.org/standards/standards.htm
[20] Workflow Process Definition Interface – XML Process Definition Language. http://www.wfmc.org/standards/TC-1025_10_xpdl_102502.pdf
[21] Web Services Flow Language. http://xml.coverpages.org/wsfl.html
[22] Web Services for Business Process Design. http://xml.coverpages.org/xlang.html

RDFS-system architecture. Practical usage of opensource standards and technologies in ISIR system.

Bezdushny A.A., Bezdushny A.N., Nesterenko A.K., Serebryakov V.A., Sysoev T.M.

The article focuses on the latest solutions of the Semantic Web W3C activity. The authors make a thorough analysis of the underlying technologies, and show a comparison of RDFS and the traditional object-oriented programming paradigm. The article describes the practice of building a distributed information system in terms of Semantic Web. It shows the usage of RDF and RDFS in the new version of ISIR. The article focuses a number of major solutions of ISIR architecture, which provides means for automated development of corporate information portals and web-enabled digital libraries. The architecture employs the most recent Java and XML technologies. It integrates smoothly into Semantic Web and uses RDFS to describe information system schema. ISIR builds an object kernel on top of any of the most widespread RDBMS, provides object-level security, auditing, data exchange and replication, full-text attributive indexing and search, and web-management of information resources.