вом организме // В кн.: Теоретические и прикладные аспекты временной организации биосистем. – М.: Наука, 1976. – С. 174–186.
6. Ершов Ю.А., Карпов А.И., Костырин Е.В. Вектор состояния подсистем организма как основа автоматизации медицинской диагностики // Биомедицинские технологии и радиотехника. – 2004. – № 12. – С. 37–42.
7. Константинова Л.И., Кочегуров В.А., Шумилов Б.М. Параметрическая идентификация нелинейных дифференциальных уравнений на основе сплайн схем, точных на многочленах // Автоматика и телемеханика. – 1997. – № 5. – С. 15–20.
8. Степанова Е.И., Нарциссов Р.Т., Кочегуров В.А., Константинова Л.И. Прогнозирование здоровья детей раннего возраста. – Томск: Изд-во Том. ун-та, 1987. – 156 с.

Поступила 14.12.2010 г.

УДК 004.822:004.4

СИСТЕМА КОЛЛЕКТИВНОЙ ПОДДЕРЖКИ ОНТОЛОГИЧЕСКИХ МОДЕЛЕЙ

И.А. Заикин, А.Ф. Тузовский, В.З. Ямпольский

Томский политехнический университет
E-mail: i@tpu.ru

Рассмотрена модель жизненного цикла онтологии; сформулированы задачи, решаемые на каждом из его этапов. Предложена архитектура системы, обеспечивающей поддержку онтологических моделей на этапах анализа предметной области, реализации, тестирования, внедрения и корректировки несколькими пользователями, каждый из которых может использовать привычный для него редактор онтологий. Предложен вариант реализации системы с использованием компонентов OWL API, Pellet, RCO Fact Extractor SDK и OpenLink Virtuoso.

Ключевые слова:
Онтология, онтологическое моделирование, жизненный цикл онтологии, управление версиями, анализ текста, аксиома онтологии, OWL 2, SPARQL.

Key words:
Ontology, ontology modeling, ontology lifecycle, version control, text analysis, ontology axiom, OWL 2, SPARQL.

Введение

Современные информационные системы позволяют выполнять все более сложную обработку разнотипной, распределенной информации. Происходит переход от работы с синтаксисом документов к работе с их семантикой (смыслом). Интенсивное развитие семантических технологий привело к возможности создания более совершенных информационных систем на основе онтологических моделей. Увеличение количества и разнообразия информации привело к возрастанию сложности онтологических моделей. Современные онтологические модели являются модульными, то есть состоят из множества связанных между собой онтологий, каждая из которых описывает отдельную предметную область или задачу. Онтологические модели не являются статичными, они развиваются в связи с появлением новых данных, изменением предметной области или задачи. Онтологии, как и программное обеспечение, имеют свой жизненный цикл [1–4]. Для создания и поддержки модульных онтологических моделей требуется система, решающая задачи, возникающие на каждом из этапов жизненного цикла.

При создании онтологии от специалистов часто требуется анализ текстов по соответствующей предметной области или задаче, что занимает значительное время. В то же время, существуют инструменты [5], позволяющие автоматически выделять из текста на русском языке ключевые слова, которые могут быть использованы в качестве основы для создания онтологии. Для поддержки постоянно развивающейся модульной онтологической модели требуется совместная работа специалистов, для чего требуется механизм управления версиями. Использование с этой целью классических систем управления версиями, применяемых при разработке программного обеспечения, связано с несколькими проблемами. Онтологическая модель может быть представлена в виде списка рёбер графа, порядок которых не играет роли при интерпретации такого представления, в то время как классические системы управления версиями учитывают любое изменение файла. Кроме того, классические системы управления версиями не позволяют учитывать связь изменений с элементами онтологии и связи между онтологиями в модели.

1. Этапы жизненного цикла онтологии

Рассмотрим этапы жизненного цикла онтологии (рис. 1) и задачи, решаемые разработчиками на этих этапах.

На первом этапе выполняется **анализ осуществимости**, то есть анализ факторов, влияющих на успех онтологии: идентификация проблемных областей, возможностей и потенциальных решений. Выполняется также оценка трудозатрат и стоимости разработки на основе имеющихся знаний о

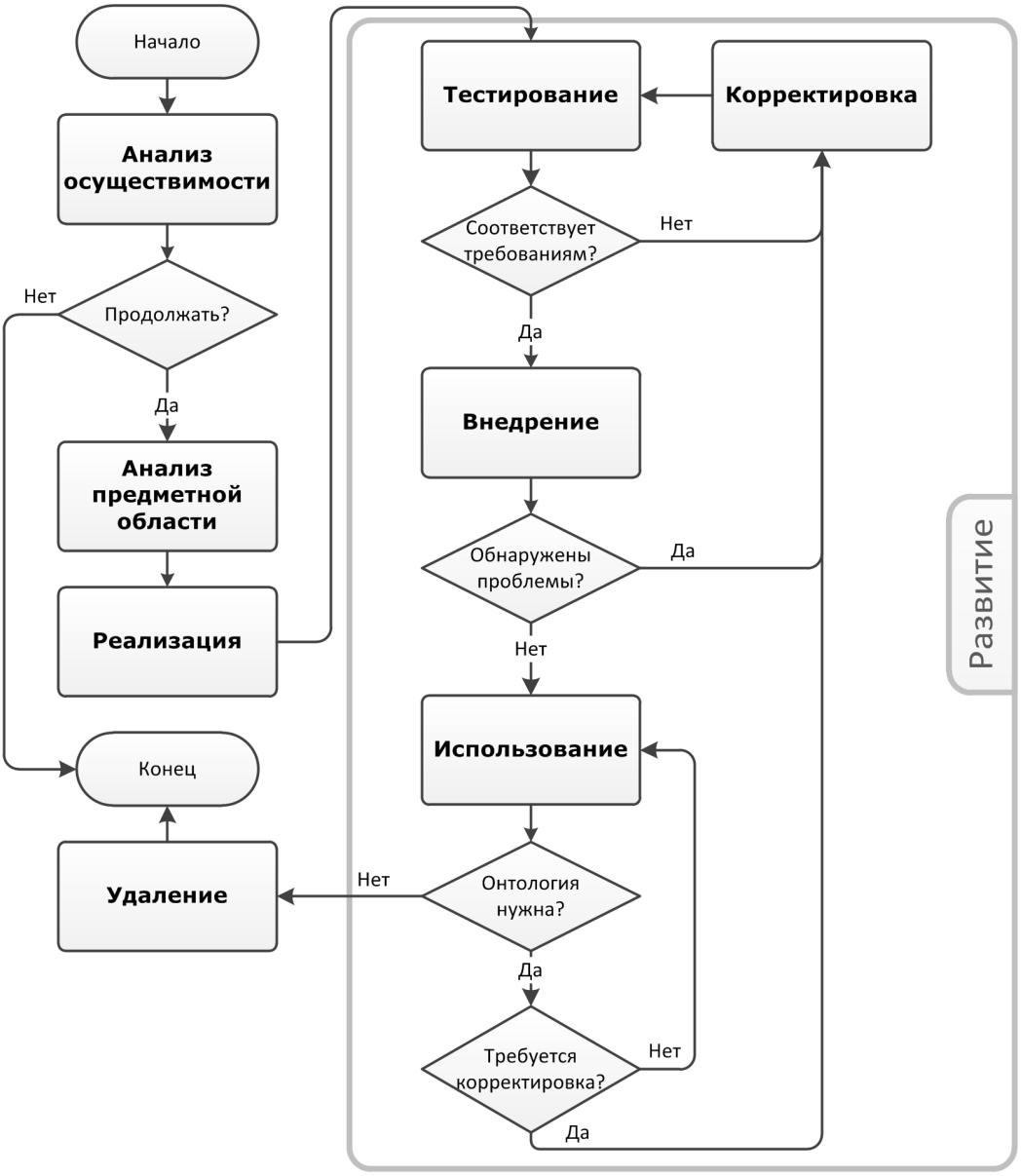

Рис. 1. *Этапы жизненного цикла онтологии*

предметной области, опыте использования инструментальных средств, предполагаемого размера и сложности онтологии.

На этапе **анализа предметной области** выявляются ключевые понятия и отношения предметной области и предназначение онтологии. Выявление понятий и отношений осуществляется разработчиками онтологии как на основе их собственного опыта и общения со специалистами в предметной области, так и на основе результатов анализа текста на естественном языке. На этом этапе также выполняется поиск уже существующих онтологий по данной предметной области с помощью поисковых машин по онтологиям, таких как Swoogle, SWSE или Watson, с использованием в качестве ключевых слов выявленных понятий и отношений.

На этапе **реализации** создаётся формальная онтология на конкретном языке представления, содержащая конструкции, специфичные для этого языка. Первый вариант онтологии может быть построен автоматически на основе выявленных понятий и отношений. Дальнейшее уточнение онтологии включает описание с помощью редактора онтологий специфичных для языка представления конструкций, таких как ограничения, перечисления, ключи, отношения несовместности и идентичности. Полученная онтология связывается с другими онтологиями, существующими в сети или на предприятии.

Тестирование модели включает проверку соответствия онтологий синтаксическим требованиям организации, семантике языка представления, а также нагрузочное тестирование. При успешном прохождении тестирования и соответствию требованиям, онтология готова к внедрению. В противном случае онтология отправляется на корректировку.

На этапе **корректировки** онтологии требуется устранить выявленные в процессе тестирования недостатки, либо внести изменения в связи с изменениями в предметной области; изменениями понимания предметной области разработчиками; связыванием с другими онтологиями; адаптацией к приложениям, использующим онтологию.

Для **внедрения** онтологий необходимо опубликовать протестированную версию онтологии. На этапе внедрения происходит обучение разработчиков приложений, использующих онтологическую модель. На этом этапе часто выявляются недостатки модели, поэтому может потребоваться корректировка отдельных онтологий с последующим тестированием всей модели.

На этапе **использования** онтологической модели приложения наполняют её. При обнаружении ошибок или неточностей в модели, требуется её корректировка с последующим тестированием. Приложения, использующие онтологию, обычно настраиваются на использование конкретной версии. При публикации новой версии, разработчики приложений знакомятся с изменениями и адаптируют к ней свои приложения.

В процессе использования может выясниться, что потребность в онтологии отпала. В этом случае онтология **удаляется**. На этапе корректировки онтологии может быть принято решение об объединении её с другой онтологией. В этом случае онтология также удаляется в пользу новой, более полной онтологии.

Таким образом, система для поддержки онтологической модели должна решать **задачи**, возникающие на каждом из этапов жизненного цикла. В их число входит выявление понятий и отношений предметной области, управление изменениями и версиями онтологий, поиск элементов онтологии, проверка и публикация онтологий. В данной работе наиболее полно рассматривается задача управления версиями онтологии, для решения которой применяются такие алгоритмы, как алгоритм сравнения версий и алгоритм применения изменений.

2. Базовые алгоритмы

Под онтологической моделью в данной работе понимается знаковая система $M=\langle O,A,R\rangle$, где $O=\{o_1,o_2,...\}$ – множество онтологий; A – множество аксиом $\{a_1,a_2,...\}$; R – функция, ставящая в соответствие каждому элементу множества O некоторое подмножество элементов из множества A.

Изменением c в онтологии o_i называется совокупность $\{o_i,a,p\}$, где a – аксиома, а p – операция, выполняемая над аксиомой: *add* (добавление) или *remove* (удаление). *Набором изменений* s называется множество $\{c_1,c_2,...\}$. *История* H_i изменений онтологии o_i состоит из множества записей $\{h_{i1},h_{i2},...\}$. *Запись в истории* $h_{ik}=\{s_{ik},r^h_{i1},r^h_{i2},...\}$ содержит набор изменений s_{ik}, и данные r^h_i, описывающие изменение (комментарий; дата и время; пользователь; инструмент, создавший набор изменений). Функция $b(o_i,\tau_1,\tau_2)$ называется *функцией истории изменений* и возвращает множество записей из истории изменений онтологии o_i, сделанных в период времени $[\tau_1,\tau_2]$. *Функция применения изменений* $f(\alpha,s)$ преобразует набор аксиом α в соответствии с набором изменений s. Она может быть реализована с использованием приведённого ниже алгоритма. Функция $ax(c)$, используемая в нём, возвращает аксиому, соответствующую изменению c.

```
1.  Для каждого изменения c∈s
2.      Если remove∈c, то
3.          Если ax(c)∈α, то
4.              α=α\ax(c)
5.      Если add∈c, то
6.          Если ax(c)∉α, то
7.              α=α∪ax(c)
```

Версией v_{ij} онтологии o_i называется фиксированный набор аксиом из множества A, который определяется как $f(v_{i,j-1},b(o_i,\tau_{i,j-1},\tau_{ij}))$, где $v_{i,j-1}$ – предыдущая версия онтологии, $\tau_{i,j-1}$ – время создания предыдущей версии онтологии, τ_{ij} – время создания версии v_{ij}.

Задача *сравнения версий онтологии* состоит в нахождении по двум данным версиям онтологии v_1 и v_2 такого набора изменений $s=\delta(v_1,v_2)$, что $f(v_1,s)=v_2$. В качестве функции δ предлагается использовать следующий алгоритм.

```
1.  s=∅
2.  Для каждой аксиомы a∈v1
3.      Если a∉v2, то
4.          s=s∪{o, a, remove}
5.  Для каждой аксиомы a∈v2
6.      Если a∉v1, то
7.          s=s∪{o, a, add}
```

3. Архитектура системы

Для поддержки онтологических моделей разработана модульная архитектура многопользовательской системы (рис. 2). Система является многопользовательской, так как с онтологиями могут одновременно работать несколько специалистов. Она состоит из таких модулей, как модуль анализа текстов на русском языке, модуль управления версиями, модуль поиска по модели, модуль тестирования и модуль публикаций. На рис. 2 также показан слой авторизации и уведомлений, включающий в себя модуль авторизации и модуль уведомлений.

Клиентская сторона содержит пользовательский интерфейс для доступа к серверным модулям системы. Доступен как синхронный режим работы, так и асинхронный. В синхронном режиме используется модуль отслеживания изменений, встраиваемый в существующие редакторы онтологий. В асинхронном режиме используется модуль сравнения версий, который генерирует наборы изменений по описанному выше алгоритму. Преимуществом асинхронного режима является то, что пользователь может редактировать онтологии в любом удобном для него редакторе. При этом даже необязательно иметь подключение к серверу. Преимуще-

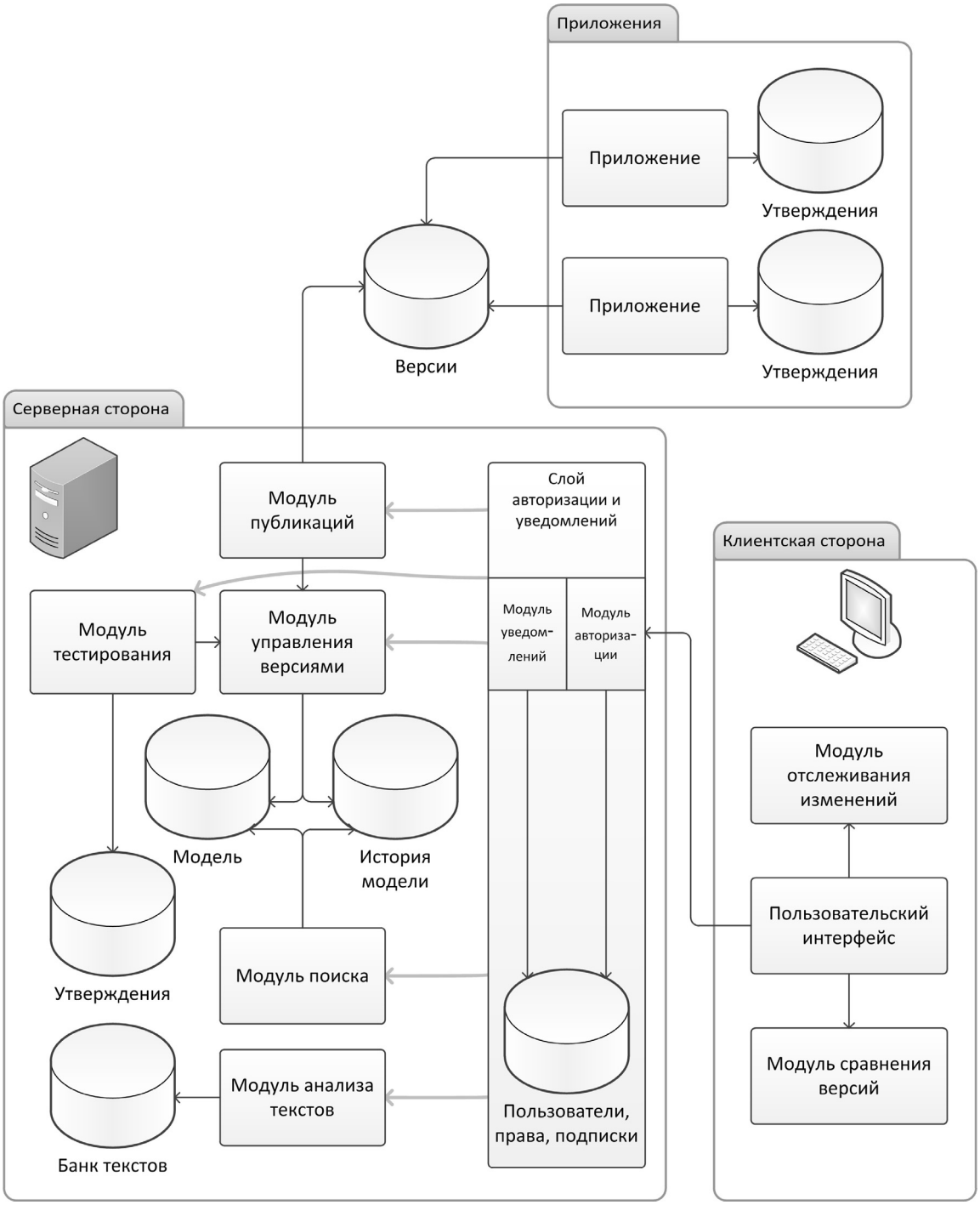

Рис. 2. *Архитектура системы*

ством синхронного режима является то, что пользователь сразу видит изменения, вносимые в модель другими пользователями.

Для авторизации пользователей в системе используется **модуль авторизации**. Для осуществления синхронного редактирования клиентская сторона подписывается на изменения. **Модуль уведомлений** осуществляет управление подписками и уведомляет клиентов об изменениях. Также он используется для уведомления о публикации новых версий, о результатах тестирования и о результатах анализа текстов.

С использованием **модуля анализа текстов** выполняется автоматическое выявление терминов предметной области, содержащихся в текстах на естественном языке. Результаты анализа заносятся в банк текстов для повторного использования, поскольку анализ текстов на естественном языке требует значительных временных затрат.

Модуль управления версиями напрямую взаимодействует как с хранилищем истории изменений модели, так и с хранилищем самой модели. Он позволяет вносить изменения в модель; создавать

предложения по внесению изменений; комментировать и оценивать предложения по внесению изменений и наборы изменений; получать состояние модели на заданный момент времени; получать историю изменений за определённый период времени; получать предложения по внесению изменений за определённый период времени. Одной из функций модуля является предотвращение конфликтов при коллективной работе над моделью с помощью *механизма предложений по внесению изменений*. Поддерживаются два режима работы: строгий и свободный.

В *строгом режиме*, прежде чем вносить какое-либо изменение в онтологию, необходимо создать предложение по внесению изменений и указать элементы онтологии, которые требуется изменить, добавить или удалить, а также исполнителей – пользователей системы, которым разрешено реализовать это предложение. С предложением может быть связан набор изменений, которые требуется применить. При создании предложения система определяет набор элементов, на которые повлияет изменение (содержащие ссылки на изменяемые элементы) и предупреждает пользователя о последствиях изменения. Предложение может обсуждаться и оцениваться, после чего оно либо принимается, либо отклоняется. Если существует несколько предложений по внесению изменений в одни и те же элементы онтологии, эти предложения должны рассматриваться совместно, и система предоставляет возможность поиска связанных предложений. Для успешного внесения изменений должны быть выполнены следующие условия: существует предложение по внесению изменений (при внесении изменений пользователь может выбрать реализуемое предложение); предложение одобрено; изменения вносит пользователь, назначенный исполнителем предложения; изменения вносятся в элементы онтологии, явно указанные в предложении.

В *свободном режиме* работы изменения могут быть внесены любым пользователем в любой элемент онтологии независимо от наличия предложения. Однако если существует недавнее изменение, затрагивающее те же элементы онтологии, что и вносимое изменение, последнее считается конфликтующим и не вносится в модель до разрешения конфликта пользователем, вносящим его.

Модуль поиска осуществляет поиск элементов онтологии, как в модели, так и в истории её изменений. Поиск осуществляется как по именам элементов и их текстовым меткам, так и по метаданным наборов изменений, связанных с данным элементом. Модуль поиска также позволяет искать предложения и наборы изменений, удовлетворяющие заданным критериям (дата, пользователь, онтология, инструментальное средство, связанные элементы онтологии).

Модуль тестирования выполняет автоматизированную проверку онтологической модели на соответствие синтаксическим требованиям организации, семантике языка представления, а также нагрузочное тестирование. Для выполнения проверки на соответствие синтаксическим требованиям используется набор сценариев, определяющих правила проверки. Проверка соответствия семантике языка представления осуществляется с помощью системы логического вывода. Современные системы логического вывода позволяют узнать и причину нарушения целостности онтологической модели. Для выполнения нагрузочного тестирования онтология наполняется тестовыми данными, а затем к ним выполняются множественные запросы с использованием машины логического вывода.

Модуль публикаций отвечает за публикацию версий онтологий на общедоступном хранилище. В качестве общедоступного хранилища может использоваться файловый сервер, веб-сервер или точка доступа SPARQL. Опубликованная версия модели может использоваться различными приложениями, как показано на рис. 2. Приложения используют опубликованную модель, однако не могут вносить изменения в неё, поэтому они используют отдельные хранилища утверждений. При публикации новой версии онтологии разработчики приложений знакомятся с изменениями, и при необходимости создают сценарии преобразования существующих фактов в формат, пригодный для работы с новой версией.

В рамках проекта «Семантическая Интеграционная Система» (№ 7734), выполняемого по программе «СТАРТ», разрабатывается система на основе предложенной архитектуры. В качестве языка представления онтологий используется язык OWL 2. Работа с онтологиями выполняется с использованием библиотеки OWL API [6], построенной в соответствии со структурной спецификацией OWL 2 [7]. Логический вывод осуществляется системой Pellet [8]. В качестве хранилища онтологической модели используется сервер OpenLink Virtuoso [9]. Обмен с хранилищем идёт по стандартному протоколу SPARQL [10], что позволяет использовать любое другое хранилище, поддерживающее этот протокол. Анализ текстов выполняется с использованием лингвистического анализатора RCO Fact Extractor SDK [5] из пакета Russian Context Optimizer.

Заключение

Выявление понятий и отношений предметной области с помощью автоматического анализа текстов на естественном языке уменьшает время и трудозатраты на создание исходного варианта онтологии. Управление изменениями даёт возможность разрешения конфликтов при многопользовательском редактировании, автоматического аннотирования изменений, а также поиска элементов онтологии по метаданным, связанным с соответствующим набором изменений. Автоматическое тестирование онтологической модели позволяет своевременно локализовать и устранить изменение, нарушившее целостность модели или пра-

вила её построения. Автоматическая публикация позволяет поддерживать в актуальном состоянии версии онтологий на общедоступном хранилище. Таким образом, проблема коллективной поддержки онтологических моделей может быть решена с помощью предложенной системы. Её эффективность обусловлена снижением затрат времени на создание и поддержку онтологической модели.

СПИСОК ЛИТЕРАТУРЫ

1. Du W. Corporate Semantic Web: Towards the Deployment of Semantic Technologies in Enterprises // Canadian Semantic Web: Technologies and Applications. – Berlin: Springer, 2010. – 217 p.
2. Noy N.F., Chugh A., Liu W., Musen M.A. A framework for ontology evolution in collaborative environments // The Semantic Web: Proc. 5th Intern. Conf. – Athens: Springer, 2006. – P. 544–558.
3. Sure Y., Staab S., Studer R. Ontology Engineering Methodology // Handbook on Ontologies. – Berlin: Springer, 2009. – 811 p.
4. Тузовский А.Ф., Чириков С.В., Ямпольский В.З. Системы управления знаниями (методы и технологии) / под общ. ред. В.З. Ямпольского. – Томск: Изд-во НТЛ, 2005. – 260 с.
5. RCO Fact Extractor SDK. Лингвистический анализатор текста. Общая информация // Технологии анализа и поиска текстовой информации. 2011. URL: http://www.rco.ru/product.asp?ob_no=5047 (дата обращения: 05.03.2011).
6. Horridge M., Bechhofer S. The OWL API: A Java API for Working with OWL 2 Ontologies // OWL Experiences and Directions: Proc. 6th Intern. Workshop. – Chantilly, 2009. – V. 529. – P. 53–62.
7. Motik B., Patel P.F., Parsia B. OWL 2 Web Ontology Language structural specification and functional style syntax // World Wide Web Consortium. 2009. URL: http://www.w3.org/TR/owl2-syntax/ (дата обращения: 05.03.2011).
8. Sirin E., Parsia B., Grau B.C., Kalyanpur A., Katz Y. Pellet: A practical OWL-DL reasoner // Journal of Web Semantics. – 2007. – V. 5. – № 2. – P. 51–53.
9. Orlink O. Implementing a SPARQL compliant RDF Triple Store using a SQL-ORDBMS. 2010. URL: http://virtuoso.openlinksw.com/dataspace/dav/wiki/Main/VOSRDFWP (дата обращения: 05.03.2011).
10. Torres E., Feigenbaum L., Clark K.G. SPARQL Protocol for RDF // World Wide Web Consortium. 2008. URL: http://www.w3.org/TR/rdf-sparql-protocol/ (дата обращения: 05.03.2011).

Поступила 09.03.2011 г.

УДК 004.415;551.46;551.52;553.361

WEB-РЕСУРС ДЛЯ АТМОСФЕРНОЙ КОРРЕКЦИИ СПУТНИКОВЫХ ДАННЫХ

М.В. Энгель[1], С.В. Афонин[1,2], В.В. Белов[1,2]

[1]Институт оптики атмосферы им. В.Е. Зуева СО РАН, г. Томск
[2]Томский государственный университет
E-mail: angel@iao.ru; afonin@iao.ru; belov@iao.ru

Дается описание Web-ресурса, позволяющего на основе физического подхода удаленно осуществлять атмосферную коррекцию спутниковых измерений. В качестве информационных источников для задания оптико-метеорологического состояния атмосферы используются локальные и пространственно распределенные информационные ресурсы. Web-ресурс на первом этапе ориентирован на обработку спутниковых данных EOS/MODIS и NOAA.

Ключевые слова:
Web-ресурс, атмосферная коррекция, спутниковые данные.

Key words:
Web-resource, atmospheric correction, satellite data.

Введение

Наблюдения земной поверхности с помощью спутниковых систем осуществляются через атмосферу, которая является многокомпонентной средой, искажающей результаты дистанционного зондирования Земли (ДЗЗ). Характер и степень атмосферных искажений зависит от спектрального диапазона и оптико-метеорологического состояния атмосферы в момент проведения зондирования. В этой связи атмосферная коррекция (АК) спутниковых измерений является необходимым условием успешного решения широкого спектра задач. Атмосферная коррекция используется в штатных алгоритмах тематической обработки спутниковых изображений системы глобального мониторинга EOS/MODIS, возможность её проведения для других спутниковых систем предоставляют различные коммерческие программные продукты (ERDAS, ENVI, FLAASH, ATCOR, ATREM, ACORN). В то же время, в большинстве случаев атмосферная коррекция производится приближенно, иногда с точностью, недостаточной для решения конкретной тематической задачи. Например, в задаче спутниковых измерений температуры поверхности Земли [1] учитывается только поглощение излучения водяным паром, но нет учета искажающего влияния аэрозоля и облачности. В задаче детектирования высокотемпературных источников (обнаруже-