О принципах создания распределённых систем сбора данных на основе MDA-подхода*

А.Е. Гуськов
Институт вычислительных технологий СО РАН
e-mail: guskov@ict.nsc.ru

А.В. Васильков, Е.В. Новосёлов
Новосибирский государственный университет

В работе рассматривается новый подход к созданию распределенных систем сбора данных, особенностью которого является использование формальных онтологических описаний для автоматизированного построения компонентов системы.

1. Постановка задачи

Объектом данной работы являются системы сбора данных (ССД), которые представляют собой программно-технический комплекс, состоящий из нескольких узлов соединенных между собой каналами связи. Обычно узлом системы является один или несколько компьютеров, содержащих программное обеспечение для выполнения следующих основных задачи сбора:

- ручной ввод данных пользователем, либо автоматизированный опрос приборов;
- прием и передача данных между узлами системы по каналам связи;
- хранение данных;
- верификация данных;
- визуализация данных.

В перечисленных задачах можно выделить два основных аспекта — функциональный и предметный, — различие между которыми можно проиллюстрировать на примере задачи хранения данных. Для неё функциональный аспект выражается в реализации процедур управления данными, их размещения во внешней памяти и обеспечении безопасного доступа. Предметный же аспект состоит в определении структур и связей между элементами данными, соответствующими заданной предметной области. Для задачи хранения данных смешивание этих двух аспектов, без особых на то оснований, считается плохой практикой. Вместо этого используют СУБД как функциональную основу, которая не зависит от конкретной структуры данных. При этом структура данных является некоторой проекцией (моделью) предметной области, т.е. выражает тот самый предметный аспект. В этой работе исследуются возможности расширения такого подхода на остальные задачи (ввод, приём и передача, верификация, визуализация) и их использования для автоматизации создания систем сбора информации.

*Работа выполнена при поддержке президентской программы «Ведущие научные школы РФ» (грант № НШ-6068.2010.9) и гранта РФФИ № 09-07-00277-а.

Примечательно, что, как и в приведённом примере, функциональные аспекты могут реализовываться с помощью унифицированных решений, а предметные — путём автоматизированной генерации соответствующих программных артефактов. В конечном итоге это позволит не только создать формальную методику построения ССД, но и сократить время на их разработку и сопровождение, особенно при наличии факторов сложности и изменчивости предметной области. Безусловно, речь идёт не о полной автоматизации создания систем, готовых к эксплуатации, а об ускорении разработки их прототипов, которые в дальнейшем могли бы быть доработаны до полнофункциональных решений.

Целью работы является создание технологий автоматизации разработки распределенных ССД на основе онтологического описания предметной области. Для этого проводится исследование возможности генерации необходимых программных моделей из онтологии, выделяются ограничения на исходную онтологию, а предлагаемый подход реализуется в виде системы, позволяющей решать перечисленные выше задачи.

Таким образом, автоматизированное построение распределенной системы сбора информации будет состоять из следующих этапов:

1. описание предметной области с помощью онтологии;
2. настройка системы на специфику предметной области, описанной с помощью онтологии;
3. обеспечение выполнения задач ввода, хранения, отправки, получения и отображения данных.

Предложенное решение является реализацией активно развивающегося направления Ontology-Driven Information Systems Engineering [1, 2], при котором в основу информационной системы ложится онтология предметной области. Это направление получило развитие лишь в последние годы, хотя совершенно ясно, что онтологии обладают достаточным потенциалом для успешного использования их во всех стадиях разработки — начиная с моделирования предметной области и заканчивая конкретной реализацией.

2. Анализ программных моделей

Рассматривая архитектуру примитивной ССД, нетрудно прийти к выводу, что для её функционирования необходимы несколько программных моделей предметной области, которые были бы согласованы друг с другом:

- реляционная модель — для хранения данных в СУБД,
- объектно-ориентированная модель — для реализации логики обработки данных,
- иерархическая DOM-модель — для представления данных в XML-форматах при передаче данных от одного узла к другому.

Хотя перечисленные модели имеют различную структуру, они должны быть взаимно согласованы. Поэтому, кроме построения самих моделей, для функционирования системы необходимо также построить описания способов их связывания друг с другом. Примером решения такой задачи является технология ORM (Object-Relational Mapping),

Рис. 1. Соответствие моделей

Т а б л и ц а 1. Соответствие основных элементов моделей

Семантическая	**Объектная**	**Реляционная**	**Иерархическая**
Классы	Классы	Таблицы	XML Schema
Свойства	Поля	Столбцы	Узлы
Объекты	Экземпляры	Строки	Деревья

которая используется для связывания реляционной и объектно-ориентированной моделей.

Принципиальным для данной работы является тезис о том, что три перечисленные модели могут быть получены как некоторая проекция общей онтологической модели, основанной на дескриптивной логике (рис. 1). Более того, при определённых условиях такое проектирование может быть выполнено автоматически. Это позволяет говорить о том, что для реализации программных моделей предметной области достаточно получить её онтологическое описание и набор преобразователей, которые сформируют соответствующие проекции. С другой стороны, полученные проекции будут непосредственно использованы функциональными модулями ССД, которые не зависят от предметной области. Таким образом, для получения функционирующего прототипа ССД в рамках данного подхода требуется лишь разработка онтологической модели предметной области, которая встраивается в готовую систему.

В таблице 1 представлено соответствие основных элементов, которыми оперируют выделенные модели. Как видно, все модели, за исключением иерархической, работают со схожими сущностями. Однако специфика данных моделей не позволяет нам простым способом провести построение и связывание их друг с другом. Поэтому для каждого отдельного элемента модели необходимо разработать алгоритм его преобразования в соответствующие элементы других моделей.

3. Архитектура системы

Для исследования предложенного подхода была разработана система, схема которой изображена на рисунке 2. Каждый компонент системы отвечает за решение отдельной задачи, при этом можно выделить две основные подсистемы: подсистема построения моделей и подсистема работы с данными. В соответствии с вышесказанным, на вход первой подсистемы подаётся онтология некоторой предметной области. Она подвергается процедуре разбора и анализа, в течение которого проверяется выполнение всех необходимых требований и построение онтологической модели предметной области. Получен-

ная модель передаётся на вход генераторам соответствующих программных моделей, которые затем используются в подсистеме работы с данными для непосредственного выполнения задач ССД.

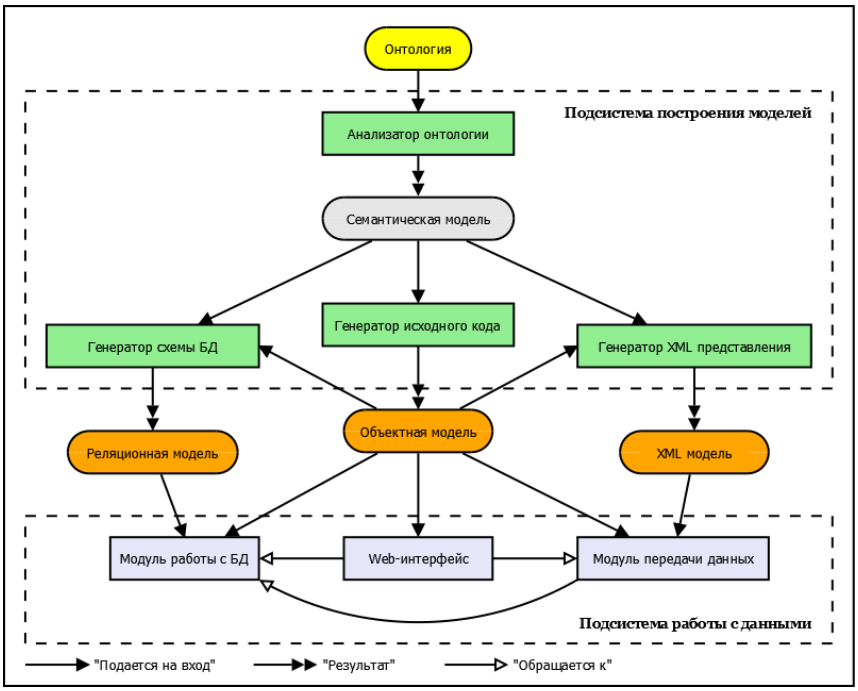

Рис. 2. Общая схема системы

На рис. 3. изображены схемы взаимодействия компонентов системы при вводе и передаче данных. Из них видно, что модуль работы с БД выполняет преобразование реляционной модели в объектную и наоборот. Это соответствует общепринятому использованию технологий ORM, которые могут быть применены и в данном случае.

Модуль передачи данных, кроме функций приёма и отправки XML-документов, обеспечивает преобразование между объектной и XML-моделью. А композиция этих двух модулей позволяет из реляционной модели получить иерархическую и наоборот.

4. Реализация системы

Реализация прототипа системы была выполнена на платформе Java с использованием ряда свободно распространяемых библиотек, в т.ч. CodeModel, Jena, Velocity.

Для программного доступа к онтологиям существуют два различных подхода:

1. Общий API, не зависящий от конкретной онтологии;
2. Специфический для каждой отдельной онтологии набор классов/методов.

К первому подходу относятся такие библиотеки как Jena и OWL API, позволяющие свободно оперировать RDF-графом онтологического описания. Второй подход реализуется с помощью генерации вспомогательных классов, которые значительно облегчают программный доступ к объектам онтологии. Данный подход используется, например,

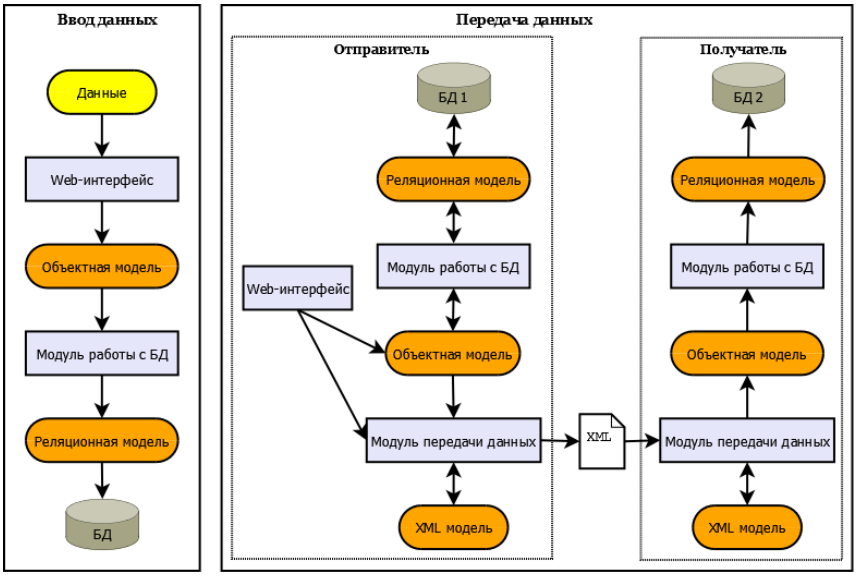

Рис. 3. Схема взаимодействия модулей системы при вводе и передаче данных

в RDFReactor и Owl2Java, которые позволяют, наряду с исходной онтологией, сохранять объекты в специализированных хранилищах (triplestore). В работе [3] был представлен более сложный инструмент, позволяющий справляться с такими проблемами, как множественное наследование. Однако различия между онтологической и объектной моделями не позволяют в полной мере отразить таким способом структуру исходной онтологии. Поэтому был выбран комбинированный подход — исходная онтология обрабатывается первым способом и формирует специфический набор классов, позволяющий в дальнейшем работать в соответствии со вторым подходом.

В качестве реализации технологии ORM была использована библиотека Hibernate. Следует отметить, что Hibernate предлагает несколько стратегий для отображения иерархии объектных классов в таблицы реляционной базы данных. В качестве основной была выбрана наиболее универсальная стратегия «one table per subclass», для использования которой требуется задать столбец-идентификатор, а для этого в корневой класс иерархии необходимо добавить соответствующее поле. Стоит отметить также отметить, что построенная на данном этапе реляционная схема находится в третьей нормальной форме.

В качестве основы для построения XML-модели был использован стандарт JAXB (Java Architecture for XML Binding), после чего построение иерархической модели свелось к выделению для каждого класса списка полей для преобразования. Опираясь на объектную модель, JAXB позволяет производить сериализацию (для передачи) и десериализацию (при приеме) объектов предметной области. Необходимым условием для этого является соответствие объектных моделей у посылающей и принимающей сторон, что достигается путем использования одной онтологии при построении обеих систем.

Для решения задач ввода и отображения данных на базе библиотеки Tapestry 5 был реализован минимальный пользовательский интерфейс, который позволяет продемонстрировать работоспособность разработанной системы. Интерфейс предоставляет

следующие возможности: просмотр списка доступных классов, просмотр списка объектов заданного класса, отправка выбранного объекта в другой узел системы, создание, просмотр, редактирование и удаление объекта заданного класса.

5. Заключение

В рамках данной работы были определены базовые технологии для автоматизированного создания распределенных ССД на основе онтологического описания предметной области. Было проведено исследование возможности генерации необходимых программных моделей на основе онтологии, выделены ограничения на исходную онтологию. В качестве подтверждения работоспособности предложенного метода создания ССД были реализованы модули генерации программных моделей, а также пользовательский веб-интерфейс, позволяющий на основе полученных моделей осуществлять ввод и отображение данных, их передачу в другой узел системы.

Предложенные методы и технологии могут быть эффективно использованы на начальных этапах построения распределённых ССД в различных предметных областях для ускоренного прототипирования.

Список литературы

[1] Yildiz B., Miksch S. Ontology-Driven Information Systems: Challenges and Requirements // Proceedings of the International Conference on Semantic Web and Digital Libraries, 2007. — c.11.

[2] Guarino N. Formal Ontology and Information Systems // Proceedings of the First International Conference on Formal Ontologies in Information Systems (FOIS), 1998. — c.3-15.

[3] Kalyanpur A. et al. Automatic Mapping of OWL Ontologies into Java // Proceedings of 16th International Conference on Software Engineering and Knowledge Engineering, 2004.