2. Bobby Suryajaya Swiss German University, Charles Lim Swiss German University. «PRODML Performance Evaluation as SOT Data Exchange Standard! International Conference on Computer, Control, Informatics and its Applications 2013, At Jakarta, Indonesia [Электронный ресурс]. – Режим доступа: http://www.researchgate.net/publication/262493459_PRODML_Performance_Evaluation_as_SOT_Data_Exchange_Standard.

3. Data quality holding real time drilling back. Article from Digital Energy Journal, June 5, 2014 [Электронный ресурс]. – Режим доступа: http://www.digitalenergyjournal.com/n/Data_quality_holding_real_time_drilling_back_by_Jay_Hollingsworth/0a0d5704.aspx.

4. Energistics Oil and Gas Standards: The Next Generation. Article from Oil Council, May 2014 [Электронный ресурс]. – Режим доступа: http://www.oilcouncil.com/expert_insight_articles/energistics-oil-and-gas-standards-next-generation.

5. IT, standards can help aggregate well data for high-level automation. Article from Drilling Contractor, September 2013 [Электронный ресурс]. – Режим доступа: http://www.drillingcontractor.org/it-standards-can-help-aggregate-well-data-for-high-level-automation-25485.

6. Open standard protocol can improve real-time drilling surveillance. Article from Hart's E&P, September 2013 [Электронный ресурс]. – Режим доступа: http://www.energistics.org/Assets/petrolinkaramcoharts090513.pdf.

7. Design of an Automated Drilling-Prediction System. Article from JPT, September 2013 [Электронный ресурс]. – Режим доступа: http://www.energistics.org/Assets/petrolinkjptfinal090513.pdf.

УДК 004

РЕАЛИЗАЦИЯ КОМПОНЕНТА РАЗБОРА ДОКУМЕНТОВ ФОРМАТА WITSML

А.Ю. Черкашин, А.Ф. Тузовский
Научный руководитель: А.Ф. Тузовский, д.т.н., профессор каф. ОСУ, ИК, ТПУ
Томский политехнический университет, 634050, Россия, г. Томск, пр. Ленина, 30
E-mail: an0648766@mail.ru

The article is devoted to developing WITSML-parser component for processing and storage data system. WITSML is open standard based on XML designed for transfer well drilling data. It is described design of parser which implementation based on Reflection, advantages and disadvantages such approach.

Keywords: XML, WITSML, Reflection, .Net Framework, DOM-parser.
Ключевые слова: XML, WITSML, Reflection, .Net Framework, DOM-парсер.

Введение

В ходе разработки нефтяных месторождений участвует большое количество организаций, использующих разнообразным техническим и программным обеспечением. Для организации взаимодействия между ними по сбору и использованию большого количества разнообразных данных создаются стандарты. Одним из таких современных стандартов является стандарт передачи данных о бурении нефтяных скважин WITSML. Он регламентирует набор сущностей и структуру, которые могут быть использованы в сообщениях, структуру самих сообщений и непосредственно протокол передачи. Данный протокол основан на XML. Это связано с тем, что данные обрабатываются различными типами компьютеров и различными программными системами. Кроме этого XML формат предоставляет возможность любому специалисту понять информацию, которая содержится в WITSML-документе. Сообщения в формате WITSML по сети передаются с использованием протокола SOAP поверх HTTP. HTTP был выбран в связи с тем, что передача данных в корпоративных сетях с использованием протокола отличающегося от HTTP часто блокируется.

Реализация

ПО для работы с WITSML строится по архитектуре клиент-сервер, в которой используется два типа сетевых объектов: WITSML-клиенты и WITSML-серверы. WITSML-клиент выполняет сбор данных с датчиков. А услуги по обработке и хранению данных предоставляют WITSML-серверы. Периодически осуществляется установление соединения по инициативе клиентов, посылающих запросы на сервер (рис. 1).

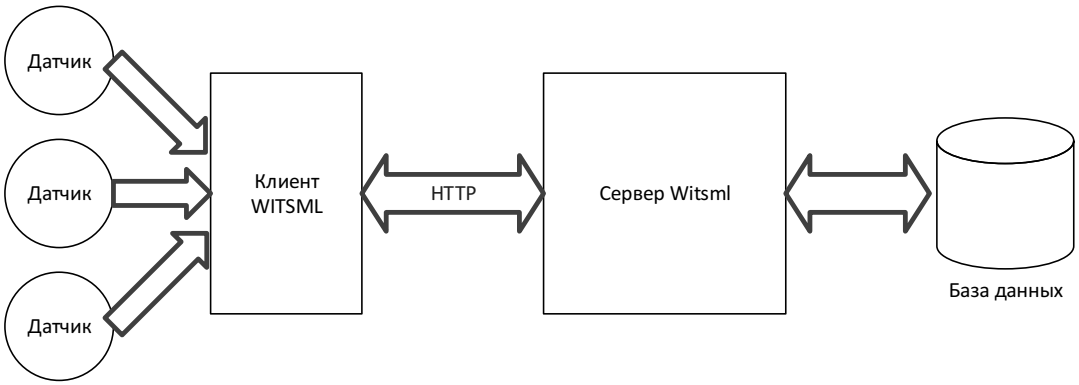

Рис. 1. Схема работы WITSML-клиента и WITSML-сервера

При разработке как сервера, так и клиента требуется реализовать грамматический анализ WITSML сообщений и проверку их синтаксической корректности сначала на соответствие правилам XML, а затем и структуре WITSML. Для выполнения такого вначале требуется извлечь содержание из SOAP сообщения в виде строки и затем передать на вход XML-парсеру.

Сотрудниками лаборатории ОСУ ведется разработка WITSML-сервера. Данный программный продукт реализуется на платформе Microsoft .Net Framework с использованием языка программирования C#.

В Framework Class Library начиная с версии 3.5 включены три набора классов для работы с XML. Два из них предназначены для полного разбора и работы с документом в памяти, также они называются DOM-парсеры, и один компонент для потоковой обработки. Потоковый парсер обычно используется, когда требуется экономное расходование памяти, поскольку не создает в памяти объектов для каждого элемента. DOM-парсер наоборот, создает полное дерево элементов в памяти. Поскольку предполагается работа со всеми элементами после разбора, был выбран DOM-парсер. А именно XDocument, который находится в пространстве имен System.Xml.Linq и поставляется в составе .Net Framework начиная с версии 3.5. Его API несколько проще, чем у устаревшего XmlDocument из пространства имен System.Xml.

Разбор XML осуществляется с помощью объекта класса XDocument. После разбора синтаксически правильного документа создается дерево элементов исходного документа в виде взаимосвязанных объектов. При это очень важно уделить особое внимание обработке ошибок, если документ было составлен неправильно.

На этапе разбора XML, если документ составлен синтаксически не корректно, дальнейшая обработка сообщения прекращается и клиенту возвращается код ошибки и сообщение, поясняющее и детализирующее ошибку, если таковое доступно для данного кода ошибки. Поскольку парсер стандартный, исключения, которые он выбрасывает, фильтруются по типу, соответственно типу присваивается код и извлекается сообщения об ошибке.

После успешного разбора XML требуется проверять соответствие передаваемых данных схеме WITSML. В стандартной библиотеке также имеются валидаторы для проверки XML документа на соответствия схеме. Однако, в WITSML существуют несколько особенностей, которые не позволяют применять стандартный валидатор. Одна из таких особенностей заключается в различии интерпретация некоторых элементов в зависимости от контек-

ста. В связи с этим возникла необходимость в разработке специальной реализации валидатора. Его принципиальное отличие от стандартного заключается в том, что он основывается на использовании средства рефлексии (Reflection) имеющейся в исполняющую среду платформы .Net Framework CLR (Common Language Runtime).

Для реализации данного валидатора генерируется набор классов CLR, соответствующих сущностям схемы WITSML, которые имеют набор свойств аналогичных набору элементов сущности. Кроме этого были созданы классы представляющие типы данных стандарта WITSML и содержащие методы проверки значения на соответствие данному типу. Для некоторых свойств и классов были заданы атрибуты для их декларативного описания. После построения DOM модели для WITSML-данных созданное дерево элементов передается на вход компонента проверки соответствия схеме.

Данный компонент начинает разбор с корневого элемента. По ходу спуска к листьям дерева для каждого XML-элемента исходя из названия тега и контекста создается соответствующий CLR-объект. Далее обрабатываются все дочерние XML-элементы и если очередной элемент является простым тегом или атрибутом, то создается объект соответствующего типа данных, вызывается метод проверки корректности значения, затем этот объект устанавливается в качестве значения соответствующего свойства ранее созданного CLR-объекта. Если очередной элемент оказывается сложным элементом, т. е. содержащим в себе дочерние элементы, то создается новый объект и производится обработка его дочерних узлов. Проверка сложных объектов осуществляется с помощью атрибутов классов.

При создании класса, с помощью механизмов рефлексии для него и всех его свойства извлекаются атрибуты, которые являются метаданными, на основе которых и делается вывод о соответствии схеме. Алгоритм повторяется до тех пор, пока не будет пройдено все дерево элементов. Если какому-либо из элементов присваивается значение неправильного типа, размера, или не соответствующее схеме, то генерируется исключение с кодом ошибки (сообщением, поясняющим ошибку), и краткой информацией о контексте ошибки, чтобы ее проще было локализовать. В случае отсутствия ошибок результатом работы компонента является полноценный CLR-объект.

Заключение

Использование механизма рефлексии, не смотря на некоторую сложности реализации компонента проверки, позволил избавится от императивного описания большого числа условий, что уменьшает количества ошибок. Классы для каждой сущности стандарта WITSML и классы для типов данных были созданы с помощью автоматической кодогенерации. Использование классов-атрибутов позволило сохранить декларативный стиль описания сущностей. Применение этих решений позволило сократить в целом время разработки и отладки за счет меньшего количества исходного кода, написанного вручную и подверженного ошибкам.

Проблема неоднозначной интерпретации некоторых элементов в зависимости от контекста, в данной реализации была эффективно решена с помощью механизма наследования и нескольких условных конструкций для определения того, какой из наследников должен обрабатывать входящий запрос.

Работы выполнены в рамках соглашения № 14.579.21.0023 по ФЦП.

Список литературы

1. Рихтер Дж., CLR via C#. – Питер, 2012, 929 с.
2. Скит Дж., C# программирование для профессионалов – Вильямс, 2011, 544 с.
3. Skonnard A., Gudgin M., Essential XML. Quick Reference – Addison-Wesley 2002, 440 с.
4. WITSML Standards [Электронный ресурс]. – Режим доступа: http://www.energistics.org/drilling-completions-interventions/witsml-standards.