УДК 004.4

ТРАНСФОРМАЦИЯ ПРОИЗВОЛЬНЫХ ТЕКСТОВЫХ ДАННЫХ В XML-ПРЕДСТАВЛЕНИЕ ПО ШАБЛОНУ НА ОСНОВЕ РЕГУЛЯРНЫХ ВЫРАЖЕНИЙ

А.А. Пантелеймонов

Институт кибернетики им. В.М. Глушкова НАН Украины
03680, Киев, проспект Академика Глушкова, 40,
тел.: (044) 522 4199; e-mail: pandrew@icyb.kiev.ua

Предлагается подход к решению задачи перевода произвольных текстовых данных в XML-представление, основанный на шаблонной обработке с помощью регулярных выражений. Приводится описание синтаксиса шаблона как XML-документа специального вида. Рассматривается возможность расширения синтаксиса, зависимого от языка реализации интерпретатора. Описываются особенности реализации, приводится алгоритм работы шаблонного интерпретатора и примеры трансформации с оценкой производительности.

The approach for arbitrary text transformation to XML notation is introduced. It is based on regular expressions and pattern processing. The pattern is represented by XML document with syntax declared. Syntax extension dependent on interpreter platform is discussed. Implementation issues along with recursive transformation algorithm and data processing examples with performance estimations are described.

Введение

Представление структурированных данных в нотации XML [1] позволяет широко использовать стандартизованные подходы, методы, средства обработки и передачи данных в гетерогенных средах. Применение XML упрощает процедуру интеграции программных компонентов, обеспечивает высокий уровень интероперабельности, облегчает интеграцию с другими технологиями, позволяет использовать готовые компоненты (модули, средства, библиотеки и т.п.) [2]. Как отмечается в [3], XML может претендовать на роль языка межпрограммного общения.

При построении компонентной программной системы, основанной на передаче и обработке данных в XML-представлении, естественным образом возникает задача перевода в XML текстовых данных других форматов, накопленных ранее либо приходящих в систему извне. Существует огромное разнообразие таких форматов (в том числе стандартизованных). И если задача трансформации XML-документа в другие отображения или представления обычно решается в рамках спецификаций и средств XSL/XSLT [4], то для задачи перевода текста произвольного формата в XML среди известных стандартов, спецификаций и рекомендаций не представлено столь эффективных и широко используемых решений.

Кандидатом на такую роль можно предложить Perl [5], как один из наиболее мощных и универсальных языков обработки текстов. Однако трудоемкость процесса создания, отладки и поддержки сценариев трансформации текстовых данных в XML на Perl в условиях частого обновления и добавления новых форматов данных, а также сложность обеспечения гибкой интеграции Perl-интерпретатора в гетерогенных системах не всегда компенсируется его универсализмом и богатством библиотек.

В докладе предлагается решение задачи трансформации текстовых данных в XML, основанное на обработке в режиме интерпретации входного текста известного формата с помощью XML-шаблона, настроенного на этот формат. Шаблон имеет структуру, близкую к структуре XML-документа, ожидаемого на выходе. В качестве средства лексического разбора (выделения фрагментов текста, используемых далее для наполнения структуры результирующего документа) предлагается использовать регулярные выражения [6] с сохраняемыми группами (capturing groups).

Основные положения синтаксиса языка

При ссылке на объектное представление конструктивных единиц языка XML будет использоваться терминология, принятая в спецификациях XML и Document Object Model (DOM) [7].

Текст интерпретируемой программы, описывающей трансформацию, записывается как XML-документ, отвечающий определенным соглашениям. Эта программа одновременно является XML-шаблоном для выходного XML-документа – результата трансформации.

Так как полная формальная запись синтаксиса в нотации, подобной форме Бэкуса-Наура (БНФ) [8, 9], выходит за рамки данной работы, приведем краткое упрощенное представление синтаксиса шаблона в более свободном неформальном виде с учетом следующих соглашений (табл. 1).

Таблица 1. Соглашения при описании синтаксиса шаблона

Конструкции и стили текста	Примеры	Описание
Italic text	match	Выделенные курсивом символы являются терминальными литералами
[…]	[.find]	Заключенное в квадратные скобки выражение является опциональным (может повторяться 0 или 1 раз)
[…]*	[super.]*	Выражение в квадратных скобках может повторяться ноль или более раз
expr1 | expr2	find | match	expr1 или expr2
Bold text	tag	Утолщенным шрифтом выделяется имя правила, заменяемое реальными терминальными последовательностями
rule ::= expr	substitution ::= $([super.]*i[,class])	rule выражается как expr

На выбор соглашений повлияли такие факты как использование угловых скобок в качестве терминальных символов синтаксиса языка XML, а также использование круглых скобок в синтаксисе шаблона для записи подстановок, во избежание двоякого толкования скобок этих видов в разных контекстах.

С учетом вышеприведенных соглашений основные положения синтаксиса XML-шаблона выглядят так:

tag-element ::= <**tag**
[*t2xrl:regex="***regex***"*]
[*t2xrl:group="***i***"*]
[*t2xrl:method="match"* | *t2xrl:method="find"*]
[*t2xrl:tag="***expr***"*]
[**attr**=*"***expr***"*]* >
[**expr** | **tag-element**]*
</**tag**>

tag ::= имя XML-элемента

regex ::= шаблон регулярного выражения

expr ::= **entry** [**entry**]*

entry ::= **substitution** | терминальный символ

substitution ::= *$(*[*super.*]***i)***

i ::= целое неотрицательное число, отвечающее индексу сохраняемой группы регулярного выражения, и где индекс 0 используется для текстового фрагмента, сопоставленного всему выражению.

В подстановке слово *super* означает, что значение сохраняемой группы нужно брать уровнем выше по дереву XML-шаблона (учитываются только элементы с регулярными выражениями).

Управляющие атрибуты. Для пространства имен (XML namespace), зарезервированного под управляющие атрибуты, была выбрана аббревиатура “t2xrl” (от англ. Text-to-XML Regular Expression Based Transformation Language). Управляющие атрибуты не включаются в XML-документ, являющийся результатом трансформации. Их назначение расшифровано в таблице 2.

Таблица 2. Управляющие атрибуты

Имя	Значение или действие по умолчанию (при отсутствии атрибута)	Описание
t2xrl:regex	На данном уровне сопоставление регулярному выражению не производится	Определяет шаблон регулярного выражения, который осуществляет лексический разбор фрагмента входного текста, определяемого атрибутами t2xrl:group и t2xrl:method. При отсутствии данного атрибута все выражения данного уровня используют значения сохраняемых групп из вышестоящего по дереву регулярного выражения
t2xrl:group	0	Определяет индекс сопоставленной сохраняемой группы от вышестоящего по дереву регулярного выражения, значение которой будет использоваться для сопоставления регулярному выражению текущего уровня. Если нет вышестоящего по дереву регулярного выражения, тогда весь входной текст выступает в роли тестируемого фрагмента

t2xrl:method	match	Метод “match” пытается сопоставить регулярному выражению текущего уровня весь входной тестируемый фрагмент, определяемый атрибутом t2xrl:group, тогда как метод “find” сканирует этот фрагмент на предмет содержания в нем подстроки, сопоставимой регулярному выражению. При нахождении более одной подстроки создается столько XML-элементов одного и того же уровня, сколько подстрок во фрагменте было найдено
t2xrl:tag	Замена имени XML-элемента не производится	Определяет имя XML-элемента, которое заменит текущее имя элемента, указанное в XML-шаблоне, описывающем трансформацию

Подробности обработки подстановок

Операция подстановки, являясь основой всей трансформации, требует более подробного пояснения. Как было вышеуказано, для любой подстановки закреплен следующий синтаксис: *$(*[*super.*]***i**)*. Вложенные регулярные выражения с сохраняемыми группами при сопоставлении с входным текстом формируют стек, каждый элемент которого является массивом размером по числу сохраняемых групп регулярного выражения соответствующего уровня, а элементы массива – переменные, хранящие сопоставленные сохраняемым группам фрагменты входного текста.

Для того чтобы извлечь из стека результат сопоставления вышестоящему по дереву регулярному выражению, а не текущему, должен применятся префикс *super*, причем столько раз, на сколько уровней выше находятся извлекаемые фрагменты.

Символ **i** в подстановке – это неотрицательное целое число, обозначающее номер сохраняемой группы (нулевая группа закреплена за всем сопоставленным фрагментом), т. е. нумерация сохраняемых групп, определяемых круглыми скобками, начинается с единицы (1, 2, 3...).

Примеры:

$(0) – результат наиболее недавнего сопоставления регулярному выражению (верхний элемент стека), либо весь входной текст, если стек пустой на момент обработки данной подстановки;

$(super.super.1) – значение первой сохраняемой группы реализованного сопоставления регулярному выражению на две позиции глубже по стеку.

Выражение может, как включать произвольное количество подстановок разных уровней, так и не содержать подстановок вообще. Для того, чтобы включить в выражение терминальный символ “$”, необходимо его удвоить (*$$*).

Составитель XML-шаблона ответственен за корректность (невыход за допустимый диапазон) номеров сохраняемых групп. Несоблюдение границ диапазона приведет к ошибке времени исполнения (runtime exception).

Расширение синтаксиса XML-шаблона для языка Java

При реализации интерпретатора также исследовалась возможность обеспечения дополнительной гибкости подстановок при трансформации в ущерб интероперабельности за счет введения пользовательских обработчиков подстановки. Цель такой модификации – обеспечение дополнительного преобразования фрагмента, сопоставленного группе, перед подстановкой по заданному пользователем алгоритму.

Пользовательские обработчики могут быть задействованы для особо сложных случаев, которые требуют каких-либо вычислений или условных подстановок, например, перевода в другие единицы измерения, смены формата, связанного с локализацией и т.п.

Сфера применения подобных обработчиков ограничена языком реализации интерпретатора.

Для поддержки пользовательских обработчиков на языке Java в синтаксис шаблона были внесены следующие расширения:

substitution ::= *$(*[*super.*]***i**[,**class**]**)**

class ::= имя Java-класса, выполняющего пользовательское преобразование перед подстановкой.

Пользовательские Java-обработчики обязаны иметь конструктор без параметров для обеспечения унифицированного инстанциирования обработчика по имени класса и реализовывать следующий интерфейс:

```
package org.t2xrl;
public interface Transformer {
    String transform(String src);
}
```

К обработчику также предъявляется такое требование как реентерабельность, так как один экземпляр обработчика может одновременно использоваться разными ветками интерпретатора текста.

Окончательный результат подстановки вычисляется как результат вызова метода *<обработчик>.transform()* со строкой, взятой по указанной в подстановке позиции из стека фрагментов исходного текста, сопоставленных сохраняемым группам регулярных выражений, в качестве параметра.

Байт-код скомпилированного класса обработчика должен быть выложен в место, доступное системному загрузчику классов виртуальной машины интерпретатора. Альтернативный способ загрузки обработчиков – передача при создании объекта трансформера в качестве параметров конструктора, помимо

шаблона, пользовательского загрузчика классов с переопределенными методами поиска и загрузки класса. Проблема «горячей» замены уже загруженного класса обработчика без перезапуска виртуальной машины не может быть решена базовыми средствами, поэтому предлагается вынужденный вариант замены путем создания обработчика с новым именем класса и замены на это имя всех его вхождений в шаблоне.

Особенности Java-реализации интерпретатора текста по XML-шаблону

XML-шаблон перед использованием по назначению (для трансформации входных текстов) должен пройти этап предварительной обработки, состоящий из нескольких фаз.

Сначала XML-шаблон преобразуется в объектное представление с помощью DOM-парсера.

Затем на основе полученного объектного представления строится дерево, дублирующее его структуру, но содержащее при узлах дополнительную информацию, ускоряющую последующую трансформацию текстов.

Все выражения проходят предобработку и разбиваются на список лексем двух типов: терминальных литеральных последовательностей и $-вхождений. Каждое $-вхождение представляется в виде пары индексов (номер сохраняемой группы и глубина стека), а также опциональной ссылки на пользовательский трансформер, который инстанциируется по имени класса на этапе предобработки в одном экземпляре.

Каждый узел, соответствующий XML-элементу, в зависимости от описания в шаблоне может хранить при себе следующую информацию:

- скомпилированный шаблон регулярного выражения и значения управляющих атрибутов, ассоциированных с этим шаблоном;
- препроцессированное в список лексем выражение для замены имени (tag) элемента;
- список остальных атрибутов, попадающих в результирующий XML-документ, с агрегированными препроцессированными выражениями, по которым вычисляются значения атрибутов;
- список дочерних узлов элемента двух типов: узлов-элементов и препроцессированных выражений текстовых блоков.

Интерпретатор входного текста по XML-шаблону осуществляет обход построенного шаблонного дерева, параллельно строя DOM-представление результирующего XML-документа, начиная с корневого узла по следующему рекурсивному алгоритму.

Вызывается процедура 1. В качестве начальных параметров процедуры берется тройка из корневого узла шаблонного дерева, узла пустого DOM-документа (наполняемого по мере обработки) и стека, состоящего из одного элемента – массива, единственная ячейка которого содержит весь текст. В результате выполнения процедуры получается заполненный XML-документ в DOM-представлении.

Процедура 1. Параметры: узел шаблона, узел результирующего XML-документа, стек результатов сопоставления регулярным выражениям.

Если текущий шаблонный узел содержит регулярное выражение, то из стека выбирается фрагмент исходного текста согласно указанных в шаблоне координат (глубина и номер группы). В цикле для каждой подстроки фрагмента, сопоставленной регулярному выражению (если указан метод поиска *find*) или один раз (если указан метод *match* и весь фрагмент успешно сопоставлен регулярному выражению) вызывается процедура 2. Процедура вызывается со следующей парой параметров: текущий узел шаблона и стек, в который вставлен результат текущего успешного сопоставления регулярному выражению. Результат выполнения процедуры 2 – DOM-элемент добавляется в качестве дочернего узла к текущему узлу DOM-представления результирующего XML-документа.

Если текущий узел не содержит регулярного выражения, то вызывается процедура 2 со следующей парой параметров: текущий узел шаблона и стек без изменений. Полученный из процедуры 2 DOM-элемент добавляется в качестве дочернего узла к текущему узлу результирующего XML-документа.

Процедура 2. Параметры: узел шаблона, стек результатов сопоставления регулярным выражениям.

Создается новый DOM-элемент, совершая все необходимые подстановки (значения атрибутов и имя узла). Затем в цикле вызывается процедура 1 для каждой шаблонного узла, дочернего по отношению к текущему, параметрами для которой будут выступать тройка из дочернего узла шаблона, только что созданного DOM-элемента и переданного стека без изменений. Результатом процедуры является созданный ею DOM-элемент.

Пример XML-шаблона и результатов трансформации по этому шаблону

Рассмотрим результаты работы алгоритма на примере упрощенного шаблона для обработки сообщений в формате протокола TL1 (Transaction Language 1) [10], широко используемого в телекоммуникационном оборудовании для его дистанционного мониторинга. Для наглядности приведем сокращенный вариант XML-шаблона, понимающий только команды ограниченного типа со статусом успешного завершения. Сокращены также малоинформативные имена устройств.

XML-шаблон:

```
<?xml version="1.0"?>
<tl1 xmlns:t2xrl="t2xrl"
        t2xrl:regex=
```

```
        "\s*(\S+) (\d\d-\d\d-\d\d) (\d\d:\d\d:\d\d)\s+M  (\S+) COMPLD\s+/\*\s*(([^:]+)\S+)\s*\*/\s+([^;]*);\s*"
        operation="$(6)" date="$(2)" time="$(3)" ctag="$(4)">
<command>$(5)</command>
<body>
  <record t2xrl:regex="&amp;quot;([^,]+),([^:]*):([^,]+),(\d+),,([^,]*),([^,]*),.*&amp;quot;"
        t2xrl:method="find"
        t2xrl:group="7"
        t2xrl:tag="$(super.1)"
        aid="$(1)" parameter="$(3)" locn="$(5)" dirn="$(6)">$(4)</record>
  </body>
</tl1>
```

Входное сообщение:

```
 N102082 02-02-02 02:02:02
M  00 COMPLD
 /*
RTRV-PM-ADSL:N102082:ADSL-1:&&::DHECV-I,,,,,,18-0;
 */
 "ADSL-1,ADSL:DHECV-I,537,,NEND,RCV,15-MIN,5-13,18-0"
 "ADSL-1,ADSL:DHECV-I,11,,NEND,TRMT,15-MIN,5-13,18-0"
;
```

Выходной XML-документ:

```
<tl1 ctag="00" date="02-02-02" operation="RTRV-PM-ADSL" time="02:02:02">
 <command>RTRV-PM-ADSL:N102082:ADSL-1:&amp;amp;amp;::DHECV-I,,,,,,18-0;</command>
 <body>
  <N102082 aid="ADSL-1" dirn="RCV" locn="NEND" parameter="DHECV-I">537</N102082>
  <N102082 aid="ADSL-1" dirn="TRMT" locn="NEND" parameter="DHECV-I">11</N102082>
 </body>
</tl1>
```

Скорость трансформации определяется многими факторами, включая глубину вложенности шаблона, общее число производимых подстановок, количество проходов по одним и тем же фрагментам входного текста, наличие ветвлений, определяемых взаимно исключаемыми регулярными выражениями одного уровня, сложность самих регулярных выражений. В среднем для сообщений формата протокола TL1, т. е. для шаблона и сообщений, аналогичных приведенным в примере, на процессоре Athlon 1830MHz (использовалась JRE версии 1.4.2) была получена оценка скорости трансформации в DOM-представление порядка 1 Мбайт/с.

Заключение

Представленная совокупность разработанных синтаксиса шаблона, алгоритма и средств трансформации произвольных текстовых данных в XML-представление по шаблону, основываясь на регулярных выражениях, позволяет эффективно решать задачу преобразования входных текстовых сообщений различных форматов в XML, что позволяет упростить и унифицировать их последующую обработку и хранение.

Предложенный подход позволяет обучать систему обработки текстовых сообщений, построенную с использованием предложенных средств, работе с новыми форматами сообщений путем загрузки XML-шаблона, их поддерживающего, без каких-либо модификаций исходного программного кода, его повторной сборки или перезапуска системы. В частности описанная технология использовалась для нормализации потока разнородных текстовых сообщений и их приведения к форме ассоциативного массива на общем пространстве ключей перед их корреляцией системой, описанной в [11].

В дальнейшем планируется развить язык шаблона и средства его интерпретации по пути обеспечения более полной поддержки языковых возможностей XML, таких как секции CDATA [1], комментарии, а также разработать средства валидации XML-шаблонов.

1. *Extensible* Markup Language (XML) 1.0. Second Edition. – W3C. – 2000. – 59 p. – URL: http://www.w3.org/TR/2000/REC-xml-20001006.
2. *Грищенко В.М.* Підходи та моделі взаємодії програмних компонентів на основі XML // Проблемы программирования. – 2000. – № 3-4. – С. 28–37.
3. *Sneed H.M.* Using XML to integrate existing Software Systems into the Web // Case studies of the 6th European Conference on Software Maintenance and Reengineering (CSMR'2002). – IEEE Computer Society Press. – 2002. – P. 25–34.
4. *Extensible* Stylesheet Language (XSL) Version 1.0. – W3C Recommendation. – 2001. – URL: http://www.w3.org/TR/xsl.
5. *Шварц Р., Кристиансен Т.* Изучаем Perl: Пер. с англ.. – К.: Издательская группа BHV, 1999. – 320 с.
6. *Regular* Expression Basic Syntax Reference. – URL: http://www.regular-expressions.info/reference.html.
7. *Document* Object Model (DOM) Level 1 Specification (Second Edition). Ver. 1.0. – W3C. – 2000. – 107 p.
8. *ISO/IEC* 14977:1996. Information technology – Syntactic metalanguage – Extended BNF. – 1996. – 12 p.
9. *Ахо А., Ульман Д.* Теория синтаксического анализа, перевода и компиляции. – М.: Мир, 1978. – т.2. – 486 с.
10. *TL1* Language Reference. – Micromuse. – 2002. – 89 p. – URL: http://www.tl1.com/pdfs/tl1/LanguageGuide.pdf
11. *Пантелеймонов А.А.* Вычисление выражений с повторяющимися и зависимыми фрагментами // Тез. докл. XLVIII научн. конф. МФТИ «Современные проблемы фундаментальных и прикладных наук». – Часть VII. – М.:–Долгопрудный, 2005. – С. 188