

Языки XSLT и XLink и их реализация функциональными методами

Д.А. Лизоркин, К.Ю. Лисовский

Московский Государственный Университет, Институт Системного Программирования РАН

XSLT -- это язык преобразований XML-документов в другие XML-документы. XLink -- это язык описания межресурсных связей при помощи XML-аттрибутов и пространства имен. Данная статья дает обзор XSLT и XLink и рассматривает их применение для Электронных Библиотек.

Мощные процессоры языков XSLT и XLink могут быть реализованы с помощью функциональных методов. В статье рассматривается XSLT-процессор STX и XLink-процессор SXLink, реализованные на языке функционального программирования Схема. Эти процессоры базируются на модели данных SXML -- представлении Информационного Пространства XML Infoset в виде S-выражений. STX и SXLink могут применяться как эффективные инструменты трансформаций и связывания в основанных на XML Электронных Библиотеках.

1 Введение

XSLT [1] -- это язык преобразований XML-документов в другой формат, часто с целью представления. Выходными форматами обычно являются XML или HTML.

Использование XML-технологий затронуло практически все современные отрасли информатики. Электронные Библиотеки (ЭБ) не являются здесь исключением, и XML рассматривается как одна из ключевых технологий для построения современных Электронных Библиотек [2].

Несмотря на то, что XML является удобным форматом для внутреннего представления данных, он, однако, не предназначен для наглядного визуального представления. Для отображения информации пользователю Электронной

Библиотеки, XML-данные должны быть преобразованы в другую форму, например, в HTML-документ, который затем может быть просмотрен человеком через браузер. Язык XSLT, разработанный Консорциумом Всемирной Сети (World Wide Web Consortium), является декларативным языком для описания именно таких преобразований. Преобразования на XSL записываются в виде так называемого стиля (stylesheet), который представляется в формате XML.

Современные Электронные Библиотеки хранят разнообразные виды информации: текст, картинки, звук, видео и пр. В контексте приложений Электронных Библиотек нам часто приходится иметь дело с громадными ресурсами, которые характеризуются большим многообразием, многочисленными форматами представления, способами обработки и т.п. Эти обстоятельства делают затруднительным (или даже невозможным) адекватное представление таких данных в виде одной информационной единицы (например, в виде единого файла). Как правило, данные Электронных Библиотек должны представляться в виде совокупности многих ресурсов. Поскольку эти ресурсы по семантике связаны между собой, они не должны существовать как совершенно независимые друг от друга, и часто желательно специфицировать их родство с использованием связей. Язык XML Linking Language (XLink) [3], разработанный Консорциумом Всемирной Сети, может быть эффективно применен для данной цели: это язык описания ссылок между ресурсами с использованием XML и отдельного пространства имен.

Помимо объединения ресурсов в группу, XLink позволяет задавать отношение роли, которую играет каждый ресурс - участник связи. Дополнительно, если в Электронной Библиотеке предполагаются переходы между различными ресурсами (например, возможность посмотреть картинку, активизировав некоторый участок текста), ссылки языка XLink также могут задавать семантику для подобных переходов.

Разделы 2 и 3 дают обзор языков XSLT и XLink соответственно. Раздел 4 объясняет, почему функциональные методы великолепно подходят для реализации XSLT, и рассматривает STX -- реализацию XSLT на языке функционального программирования Схема (Scheme). Раздел 5 рассматривает реализацию языка XLink на Схеме -- SXLink. Раздел 6 обсуждает вопрос интеграции языков XSLT и XLink функциональными методами.

2 **Обзор языка XSLT**

Язык преобразований XSL (XSLT) -- это одна из наиболее мощных и активно используемых технологий в семействе XML. Язык XSLT получил широкое признание, и активно используется в индустрии, обычно в качестве языка для преобразования XML-документов в другой формат, часто для представления.

Хотя XSLT -- это полный с точки зрения Тьюринга язык программирования, он никогда не позиционировался как язык общего назначения для преобразования XML, равно как язык, в одиночку используемый для разработки законченных XML-приложений. Как явно указывается в Требованиях к XSLT [4], превращение XSLT в

язык программирования общего назначения не является явной целью.

Преобразование в языке XSLT выражается в виде правильно сформированного (well-formed) XML-документа [5]. XML-элементы, используемые языком XSLT, отличаются принадлежностью определенному пространству имен XML [6].

Преобразование, выраженное через XSLT (с помощью стиля), описывает правила для преобразования исходного дерева документа в конечное дерево. Конечное дерево отделено от исходного дерева. Структура конечного дерева может полностью отличаться от структуры исходного дерева. В процессе построения конечного дерева элементы исходного дерева могут подвергаться фильтрации и переупорядочению, также может быть добавлена новая структура.

Стиль содержит набор правил шаблона. Правило шаблона состоит из двух частей: *образца*, который сопоставляется с узлами в исходном дереве, и *шаблона* преобразований, который может быть обработан для формирования фрагмента конечного дерева. Преобразование производится за счет сопоставления образцов узлам дерева и вызова ассоциированных с образцами шаблонов:

- *Образец* -- это выражение на XPath, и он сравнивается с элементами исходного дерева. Узел соответствует конкретному образцу, если узел числится в наборе узлов, полученных в результате обработки данного образца как XPath-выражения в некоем возможном контексте. Возможные контексты -- это такие контексты, чьим узлом контекста является проверяемый узел или один из его предков. Для каждого фиксированного узла в исходном дереве, для него будет обрабатываться шаблон, ассоциированный с тем образцом, которому данный узел соответствует.
- *Шаблон* обрабатывается для данного конкретного узла, чтобы создать фрагмент в конечном дереве. Шаблон может содержать произвольные XML-данные, которые будут трактоваться как готовые фрагменты конечного дерева; а также элементы из пространства имен XSLT, определяющие инструкции по созданию фрагментов. При обработке шаблона каждая инструкция обрабатывается и заменяется на вычисленный ею фрагмент конечного дерева. Конечное дерево формируется как результат обработки шаблона, соответствующего корневому узлу исходного документа.

Для простых преобразований стиль часто образуется одним шаблоном, который используется как шаблон для всего конечного дерева. Данный подход особенно популярен для преобразования XML-документов, ориентированных на данные (data-centric XML documents).

Рисунок 1 дает пример стиля на XSLT. Данный стиль преобразует электронный документ, выраженный в виде XML-данных, в презентационный формат на XHTML. Стиль обозначается глобальным элементом документа `xsl:stylesheet` из пространства имен XSLT "<http://www.w3.org/1999/XSL/Transform>". Стиль обычно состоит из нескольких шаблонов XSLT, каждый из которых описывается своим собственным элементом `xsl:template`. Для краткости рисунок 1 показывает лишь один такой шаблон. У шаблона в общем случае есть образец, который задается

выражением на XPath и представляется как значение атрибута `match` данного элемента-шаблона. Образец используется для сопоставления с узлами исходного дерева. Шаблон на рисунке [1](#) соответствует всем элементам `title`, родителями которых является `doc`. Для каждого элемента, соответствующего образцу, шаблон обрабатывается (относительно данного элемента), чтобы создать фрагмент конечного дерева, в соответствии с содержимым `xsl:template`. В нашем примере на рисунке [1](#), преобразование рекурсивно применяется к дочерним вершинам элемента (эту работу выполняет инструкция XSLT `xsl:apply-templates`), и результат включается в заголовок первого уровня. Предполагая, что элемент `title`, родителем которого является `doc`, содержит название документа, данный шаблон создаст заголовок первого уровня, содержащий это название.

```
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"> <xsl:template
match="doc/title"> <h1> <xsl:apply-templates/> </h1> </xsl:template> <!--
Какие-то другие шаблоны --> </xsl:stylesheet>
```

Рисунок 1: Пример стиля для преобразования электронных документов в XHTML. Показанный шаблон преобразует название документа в заголовок первого уровня.

[3](#) Обзор языка XLink

XML Linking Language (XLink) -- это язык описания межресурсных связей с помощью XML.

XLink обеспечивает полную функциональность гиперссылок HTML, и гораздо большее: он позволяет устанавливать отношение связи между более чем двумя ресурсами, ассоциировать различные метаданные со ссылками, соединять ресурсы без их модификации [[7](#)].

Язык XLink позволяет соединять произвольные ресурсы, а не только XML-

документы. Понятие ресурса определяется в IETF RFC 2396 как любая адресуемая единица информации или сервиса. Если ресурс представляет собой правильно сформированный (well-formed) XML-документ, то спецификация XLink считает ресурсом также любую часть этого документа, определяемую идентификатором фрагмента на языке XML Pointer Language (XPointer). Подобный идентификатор фрагмента может дополнять адрес документа, задаваемый с помощью Унифицированного Идентификатора Ресурса (URI).

Язык XLink вводит два типа ссылок.

1.

Расширенная ссылка (extended link). Расширенные ссылки выражают полную функциональность языка XLink, такую как входящие (inbound) и сторонние (third-party) арки, а также объединяют произвольное количество участвующих в них ресурсов. Участвующие ресурсы могут быть любой комбинацией локальных и удаленных.

В терминах XLink, локальный ресурс -- это XML-элемент, который участвует в ссылке за счет того, что ссылочный элемент является для него родительским. Ресурс, который участвует в ссылке благодаря тому, что к нему адресуются с помощью унифицированного идентификатора URI, считается удаленным (remote), даже если он располагается в том же XML-документе, что и ссылка, или даже внутри ссылочного элемента.

Как результат, структура расширенных ссылок может быть достаточно сложной, включая в себя элементы, которые указывают на удаленные ресурсы; элементы, содержащие локальные ресурсы; элементы, которые определяют условия перехода по дугам; и элементы, обеспечивающие понимаемый человеком текст комментариев.

Обычно элементы типа "расширенная ссылка" располагаются отдельно от тех ресурсов, которые они соединяют (например, в совершенно разных документах). Таким образом, расширенные ссылки важны для ситуаций, когда соединяемые ресурсы доступны только для чтения; или когда модификация этих ресурсов является дорогостоящей и сложной операцией, тогда как модификация отдельно располагающейся ссылки достаточно проста; или когда ресурсы имеют форматы, не поддерживающие встроенные ссылки (как для многих мультимедийных форматов).

В языке XLink определяется способ, когда расширенной ссылке добавляется специальная семантика для указания на ссылочные базы (linkbases). Когда ссылка используется в таком назначении, она помогает XLink-приложению обрабатывать другие ссылки.

Рисунок [2](#) показывает пример использования расширенной ссылки языка XLink. В этом примере мы соединяем книгу из Электронной Библиотеки, автора книги и издательство. Мы даем описание автора в локальном

ресурсе языка XLink, т.е. описание целиком располагается внутри ссылочного элемента. Книга и издательство -- это удаленные ресурсы языка XLink, потому что на них ссылаются при помощи Унифицированных Идентификаторов Ресурсов. Каждый из трех ресурсов помечен собственной меткой (label). Эти метки используются при описании переходов. Мы определяем два перехода: от автора к книге и от книги к издательству. Каждый переход описывается своим собственным элементом типа дуга (arc). Заметим, что дуга, ведущая от книги к издательству, соединяет два удаленных ресурса.

```
<!--Расширенная ссылка языка XLink. Элемент может носить произвольное имя--> <MyExtendedLink xmlns:xlink="http://www.w3.org/1999/xlink" xlink:type="extended">
<!--Элемент типа ресурс. Специфицирует локальный (в терминах XLink) ресурс--> <author xlink:type="resource" xlink:label="A">
<!--Некоторая разметка, относящаяся к локальному ресурсу-->
<name>John</name> <surname>Smith</surname> </author>
<!--Элемент типа локатор. Адресуется к удаленному ресурсу по его URI--> <book xlink:type="locator" xlink:label="B"
xlink:href="http://library.com/book.xml"/> <publisher
xlink:type="locator" xlink:label="P" xlink:href="http://publisher.com"/>
<!--Элемент типа дуга. Определяет переход между ресурсами с метками "A" и "B"--> <MyArcElement xlink:type="arc" xlink:from="A"
xlink:to="B"/> <MyArcElement xlink:type="arc" xlink:from="B"
xlink:to="P"/> </MyExtendedLink>
```

Рисунок 2: Пример расширенной ссылки языка XLink. Ссылка устанавливает отношение связи между книгой (удаленный ресурс), ее автором (локальный ресурс) и издательством (удаленный ресурс).

2.

Простая ссылка (simple link) -- это ссылка, которая ассоциирует в точности два ресурса -- один локальный и один удаленный; с аркой, идущей от

первого ко второму. Таким образом, простая ссылка всегда является исходящей (outbound). Исходящая связь, в которой участвует ровно два ресурса, является наиболее распространенной (например, в эту категорию попадают ссылки `A` и `IMG` языка HTML). Поскольку простые ссылки предоставляют значительно меньше функциональных возможностей, чем расширенные ссылки, у простых ссылок нет какой-либо специальной внутренней структуры.

Хотя простые ссылки концептуально являются подмножеством расширенных ссылок, они синтаксически различны. В частности, для преобразования простой ссылки в расширенную ссылку требуется несколько структурных преобразований.

Предназначением простой ссылки является удобная короткая форма записи для эквивалентного случая расширенной ссылки. Один элемент вида "простая ссылка" объединяет в себе базовую функциональность элементов типа "расширенная ссылка", "локатор", "арка" и "ресурс". В том случае, когда реально требуется лишь подмножество свойств этих элементов, простая ссылка удобна как альтернатива расширенной ссылке.

Иллюстрация простой ссылки приведена на рисунке 3. Она ведет к содержанию некоторой книги, и напоминает гиперссылку `A` языка HTML. Однако, в отличие от гиперссылки HTML, простая ссылка языка XLink не требует именованного якоря для адресации к фрагменту документа, поскольку идентификатор фрагмента на языке XPointer предоставляет более гибкие возможности.

```
<!--Простая ссылка языка XLink. Элемент может носить произвольное имя--> <MySimpleLink xmlns:xlink="http://www.w3.org/1999/xlink" xlink:type="simple" xlink:href="http://library.com/book.xml#xpointer(doc/contents)"> Go to table of contents <!--Какая-то более сложная разметка может быть описана здесь--> </MySimpleLink>
```

Рисунок 3: Пример простой ссылки языка XLink

Следует отметить, что спецификация XLink в основном фокусируется на структурах данных для описания ссылок, и не фокусируется на модели их поведения. Любой вид сложной интерпретации и обработки ссылок является ответственностью приложения более высокого уровня, базирующегося поверх языка XLink.

4 STX: реализация XSLT на Схеме

Язык программирования Схема (Scheme) предоставляет практически неограниченные возможности по преобразованию XML/SXML-данных. Известным тому примером является Язык Семантики и Спецификации Стиля Документов (Document-Style Semantics and Specification Language, DSSSL) [8]; и недавно был разработан ряд новых систем, основанных на SXML [9, 10].

В данной статье мы рассмотрим STX -- инструмент для преобразования XML, основанный на XSLT и Схеме, и соединяющий в себе процессор для наиболее типичных стилей XSLT и систему для их расширения с помощью Схемы. STX обеспечивает среду для обобщенных преобразований XML-данных. STX интегрирует два функциональных языка -- Схему и основанный на XSLT язык трансформаций -- на базе общей модели данных SXML.

STX основан на SXSLT [11], и может рассматриваться как его специализация, направленная на совместимость с XSLT. В частности, STX использует алгоритм обхода дерева, разработанный в SXSLT, специализированный и оптимизированный для преобразования XML/SXML-данных в стиле XSLT.

4.1 Причины для реализации XSLT на Схеме

XSLT разработан главным образом для тех видов преобразований, которые требуются, когда XSLT используется как часть языка стилей XSL [1]. Для более сложной обработки XML-данных XSLT обычно используется совместно с некоторым языком программирования общего назначения, таким как Java или Python, что приводит к хорошо известной проблеме несоответствия импеданса, из-за разных парадигм программирования и моделей данных.

XSLT может рассматриваться как функциональный язык, и модель данных XML очень близка к S-выражениям Схемы [12]. Схема широко используется как язык расширения и как язык обработки XML, что делает Схему естественным кандидатом для расширения XSLT.

XSLT -- это полный с точки зрения Тьюринга язык программирования, и многие впечатляющие примеры доказывают его применимость для целого ряда самых разнообразных задач. Тем не менее, большинство "жизненных" приложений используют небольшое подмножество XSLT для относительно простых

преобразований XML-документов. Некоторые учебники по XSLT [13] даже утверждают, что лишь нескольких команд XSLT достаточно для большинства практических нужд.

STX предназначен для расширения типичных тривиальных стилей XSLT программным кодом на Схеме. В подобной системе, совместимые с XSLT шаблоны описывают представление, в то время как более сложные преобразования данных, анализ данных, или даже бизнес-логика выражаются на Схеме.

Данный подход позволяет повторно использовать существующие клиентские навыки в XSLT и защитить инвестиции в презентационные стили XSLT, и предоставляет развитые прикладные возможности по обработке данных, которые делают возможным реализовать законченное бизнес-решение в рамках предлагаемого инструментария.

4.2 Архитектура STX

XSLT использует слегка модифицированную модель данных XPath [14], которая может рассматриваться как представление Информационного Пространства XML Infoset [15] в виде помеченного дерева узлов. STX основан на модели данных SXML, поскольку SXML является реализацией Информационного Пространства XML в виде S-выражений.

SXML изначально разрабатывался с целью обеспечить эффективное вычисление запросов на XPath. Для SXML существует формальная спецификация [12], которая дополняется набором реализованных на Схеме основных XML-технологий. Две из этих технологий, а именно, XML-парсер SSAX и язык запросов к SXML SXPath, составляют ядро STX. И SXML, и модель данных XPath основаны на Информационном Пространстве XML, представленном как древовидная структура. Сходства между этими моделями данных очевидны [16], и их взаимное отображение друг на друга прямолинейно, что значительно упрощает интеграцию. По этим причинам STX основан на модели данных SXML.

Сопоставление образцов узлу производится в STX в соответствии с Рекомендацией XSLT. Правила шаблонов определяют вершины, к которым они применяются, с помощью образца, который представляет собой выражение на XPath. STX использует прямолинейный алгоритм сопоставления, основанный на SXPath -- реализации XPath на Схеме [17]. Сопоставление в STX производится за счет применения SXPath к образцу; преобразования образца в функцию, которая выбирает множество узлов; и последовательного применения этой функции к сопоставляемому узлу и к его родителям, до тех пор, пока сопоставляемая вершина не окажется членом результирующего множества узлов, или не будет достигнута корневая вершина. В первом случае узел соответствует образцу, во втором случае -- нет.

Целостная интеграция XSLT и Схемы была основной целью при разработке STX. Данная цель достигается использованием общей модели данных и вычислительной парадигмы для обоих компонентов STX.

Шаблоны XSLT являются, в сущности, чистыми функциями -- каждый шаблон определяет фрагмент конечного дерева как функцию от фрагмента исходного дерева, и не производит побочных эффектов [18]. STX преобразует шаблоны в списки функций Схемы. В каждом списке, первая функция конструируется с помощью XPath и используется для сопоставления шаблона узлам; вторая представляет собой функцию для преобразования в соответствии с шаблоном. Эта функция для преобразования может быть описана на Схеме (в случае `stx:template`), так же как и на XSLT (в случае `xsl:template`). В этом контексте, XSLT может рассматриваться как "синтаксический сахар" для кода на Схеме, в который она преобразуется. Идея иллюстрируется схемой [1](#).

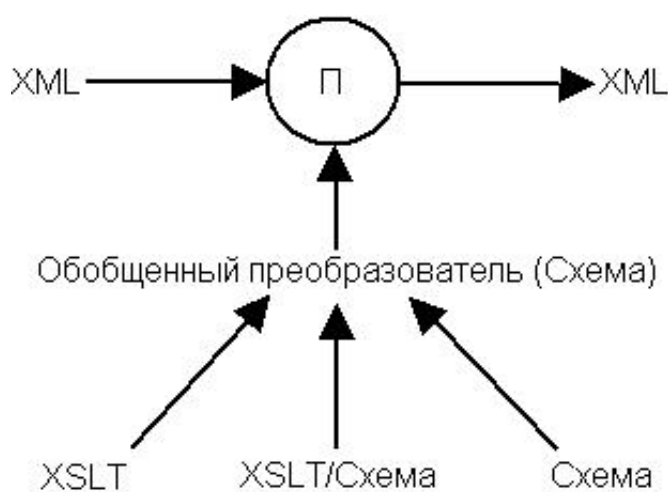


Схема 1: Модель обработки в STX. STX принимает стили, описанные на XSLT, на Схеме, а также на их комбинации.

STX подражает простому "презентационному" подмножеству XSLT. Менее популярные конструкции XSLT не поддерживаются, но могут быть заменены шаблонами на Схеме. Эта категория состоит, в основном, из "программистских" конструкций, таких как инструкции условной обработки, обработка переменных и параметров. Мы полагаем, что эти конструкции больше соответствуют языку программирования (Схема), чем языку стилей (XSLT).

XSLT может рассматриваться как чисто функциональный язык, однако у него есть серьезное ограничение: в XSLT функции не являются объектами первого класса. Данное ограничение является особенно жестким для языка XML-преобразований, потому что оно драматическим образом сужает класс реализуемых алгоритмов обработки древовидных структур. Естественно, подобного ограничения нет в Схеме, что делает функции и шаблоны, написанные на Схеме, гораздо более эффективными и гибкими в данной прикладной области.

[5](#) Реализация XLink на Схеме: SXMLink

SXMLink -- это реализация XLink и Интерфейс Прикладного Программирования (Application Program Interface, API) на языке функционального программирования Схеме. SXMLink полностью поддерживает синтаксис и семантику XLink, и предоставляет Интерфейс Прикладного Программирования для обработки множества документов, соединенных с помощью XLink.

[5.1](#) Причины для реализации XLink на Схеме

XLink не является языком программирования общего назначения, и поэтому не обладает достаточными возможностями для разработки большинства практических приложений. Для создания законченного XML-приложения XLink должен использоваться совместно с каким-то другим языком, что часто приводит к уже упоминавшейся проблеме несоответствия импеданса.

Такой проблемы нет в Схеме, так как S-выражения создают единую среду для представления документов с элементами XLink -- в виде SXML. Возможность использования единого языка высокого уровня для обработки XML-данных и реализации приложений является одним из важных достоинств подхода, использованного в SXMLink.

Кроме того, обработка документов, содержащих элементы XLink, является задачей обхода или преобразования деревьев. Данная задача прекрасно укладывается на парадигму обработки иерархических S-выражений, т.е. при этом могут использоваться типовые алгоритмы и инструментальные средства языка Схеме.

[5.2](#) Архитектура SXMLink

Важной частью SXMLink является специализированный парсер SSAX, который распознает все конструкции XLink, присутствующие в XML-документе, и преобразует его в расширенный SXML.

Поскольку ссылки языка XLink в общем случае вовлекают несколько соединяемых документов, SXMLink предлагает удобные средства для работы с множеством документов как с единым целым.

Как обсуждалось в разделе [3](#), спецификация XLink фокусируется в основном на

предоставлении структур данных для описания ссылок, и не предоставляет никакого Интерфейса Прикладного Программирования для их обработки. Мы разработали такой Интерфейс Прикладного Программирования и реализовали его в SXPath. Стоит отметить следующие операции, предоставляемые системой SXPath:

1.

Автоматическая валидация ссылок. Эта операция позволяет убедиться, является ли каждая ссылка XPath семантически осмысленной, т.е.

- все ресурсы, участвующие в ссылке, существуют;
- все части ресурсов, определяемые с помощью идентификаторов фрагментов на языке XPath, доступны;
- (если требуется) роли и/или роли дуг в XPath представляют собой

Валидация ссылок также идентифицирует ресурсы для набора ресурсов.

интересующих документов, таким образом предоставляя удобный механизм поддержания ссылок языка XPath действительными.

Автоматическая валидация ссылок, предоставляемая SXPath, наиболее важна для большого количества соединенных ресурсов, особенно для основанных на XML Электронных Библиотек.

2.

Разрешение ссылок. Данная операция преобразует все ссылки языка XPath в исходящие. Результирующий документ (называемый "разрешенным документом") эквивалентен исходному с точки зрения семантики XPath, но становится проще с точки зрения его последующей обработки другими приложениями, потому что для исходящей ссылки легко определить начало перехода. Например, разрешенный документ затем может быть легко преобразован в HTML и просмотрен с помощью браузера.

3.

Включение вершин. Данная операция является дальнейшим развитием разрешения ссылок. Включение вершин заменяет каждый узел в документе, являющийся исходным ресурсом (starting resource) на соответствующий этой ссылке целевой ресурс (ending resource). Данная операция особенно полезна, когда целевые ресурсы ссылок описывают некоторую детализированную информацию, и желательно создать новый документ, в котором вся эта информация включена в явном виде.

SXPath обеспечивает дополнительную гибкость рассмотренных операций за счет использования функций языка Схема в качестве объектов первого класса.

6 Функциональная интеграция XSLT и XPath

Данный раздел описывает несколько сценариев интеграции XSLT и XLink, целью которой является повышение выразительной мощи и гибкости средств преобразования и связывания XML-данных. Данный подход к интеграции XSLT и XLink значительно выигрывает при реализации его с использованием функциональных методов, как дальнейшее расширение STX и SXPath.

6.1 Соединение документа и его стиля

Как обсуждалось в разделе [2](#), один стиль XSLT может быть использован для большого класса документов, имеющих одинаковую структуру исходного дерева. Однако спецификация XSLT не предоставляет механизмов описания ассоциаций между стилем и XML-документом, к которому этот стиль может быть осмысленно применен.

XLink является естественным кандидатом для описания этих ассоциаций в виде ссылок. Поскольку ссылки XLink могут располагаться отдельно от ресурсов, которые они связывают [\[7\]](#), эти ссылки могут быть определены в наиболее удобном месте для каждого конкретного практического приложения:

- в стиле XSLT,
- или в XML-документе, для которого этот стиль разработан,
- или даже как отдельный XML-документ (называемый ссылочной базой).

Процессор XSLT, расширенный поддержкой XLink, сможет руководствоваться этими ссылками для выбора подходящего стиля XSLT для данного XML-документа. Подобный интеллектуальный процессор XSLT может быть удобно реализован как интеграция STX и SXPath, поскольку они оба базируются на общей модели данных (SXML) и общих алгоритмах (таких как XPath).

6.2 Преобразование XSLT в качестве представления целевого ресурса при переходе по ссылке языка XLink

При определении ссылки XLink иногда бывает желательно задавать представление (view) для ее целевого ресурса (получаемого как результат перехода по этой ссылке).

Хотя спецификация XLink не предоставляет данной возможности в явном виде, спецификация предоставляет механизм, который позволяет нам сделать такое расширение.

Спецификация XLink вводит глобальный атрибут `arcrole` в пространстве имен XLink, предназначенный для описания семантики целевого ресурса ссылки относительно ее исходного ресурса. Атрибут `arcrole` соответствует понятию свойства в Модели Описания Ресурсов [\[19\]](#), где роль может быть проинтерпретирована как утверждение о том, что *"исходный-ресурс имеет роль-дуги целевой-ресурс"*. В соответствии со спецификацией XLink [\[3\]](#), значением атрибута `arcrole` должен быть Унифицированный Идентификатор Ресурса, идентифицирующий некоторый ресурс, который описывает соответствующее

свойство.

Мы предлагаем одним из возможных применений атрибута XLink `arcrole` -- идентификация стиля XSLT. Если атрибут `arcrole` идентифицирует стиль, то этот стиль следует использовать для построения представления для целевого ресурса ссылки в момент перехода по ней.

Данная идея иллюстрируется рисунком 4. Ссылка XLink ведет к книге в Электронной Библиотеке, и атрибут `arcrole` идентифицирует стиль, определяющий представление для книги, которое будет получено при переходе по ссылке. Например, стиль может преобразовывать книгу в HTML. Необходимо отметить, что получаемое представление зависит от той ссылки, переход по которой был осуществлен. Другая ссылка, ведущая к той же книге, возможно, будет использовать другой стиль XSLT, и таким образом различные представления одной и той же книги будут получены при переходе по разным ссылкам. Подобное поведение полностью соответствует семантике атрибута `arcrole`, определенной в спецификации XLink.

```
<link-to-a-book xmlns:xlink="http://www.w3.org/1999/xlink" xlink:type="simple"
xlink:href="http://library.com/book.xml"
xlink:arcrole="http://library.com/transform.xsl"/>
```

Рисунок 4: Использование роли дуги для задания стиля XSLT, который служит в качестве представления для целевого ресурса (книги в Электронной Библиотеке)

SXLink может быть легко расширен предложенной функциональностью, поскольку SXLink распознает полный синтаксис XLink и, таким образом, атрибут `arcrole`, и поскольку интеграция SXLink и STX является естественной и прямолинейной.

[6.3](#) XSL-преобразование нескольких связанных XML-документов

Преобразование, выраженное через XSLT, обычно включает в себя единственный

обрабатываемый XML-документ. Это неудивительно, учитывая что Рекомендация XSLT появилась на два года раньше Рекомендации XLink.

Преобразование XML-документа, который связан с другими документами, с обеспечением должного внимания к этим связям, является сложной задачей. Полное распознавание разметки XLink в шаблонах XSLT -- это неуклюжее решение, т.к. синтаксис XLink является довольно громоздким, а семантика -- достаточно сложной. Вместо этого, мы предлагаем обрабатывать разметку, имеющую отношение к XLink, с помощью XLink-процессора, интегрированного с XSLT-процессором. В этом случае, лишь незначительно расширение синтаксиса XSLT требуется для преобразования связанных XML-документов.

А именно, мы предлагаем добавить одну дополнительную ось в язык XML Path (который используется как часть XSLT). Назовем ее ось `traverse` (перейти), и она будет содержать все узлы, к которым можно перейти по ссылкам языка XLink из контекстного узла. С использованием этой дополнительной оси мы в понятной и краткой форме сможем выражать XSL-преобразования, включающие в себя несколько соединенных документов.

Рисунок 5 иллюстрирует идею подобного расширенного шаблона XSL, который преобразует некоторую библиографию, выраженную на XML. Ось `traverse` может использоваться в образце шаблона, чтобы указать, что шаблон соответствует любому узлу (`node`) в библиографии, который является ссылкой XLink на книгу (`book`). Заметим, что книга может располагаться в другом XML-документе. В теле шаблона ось `traverse` позволяет нам сконструировать текст по информации из другого документа (`xsl:value-of`), и даже применять шаблоны к фрагменту удаленного документа (`xsl:apply-templates`).

```
<xsl:template match="node()[traverse::book]"> ... <xsl:value-of  
select="traverse::book/title"/> ... <xsl:apply-templates  
select="traverse::book/chapter/section"/> ... </xsl:template>
```

Рисунок 5: Шаблон XSL, расширенный поддержкой XLink

7 Заключение

Каждый из языков XSLT и XLink имеет свое уникальное предназначение в наборе XML-технологий. Эти два языка достаточно важны в контексте основанных на XML Электронных Библиотек.

Функциональные методы прекрасно подходят для обработки XML-данных, в частности, для их преобразования и связывания. В данной статье мы рассмотрели некоторые способы применения функциональных методов для реализации инструмента XML-преобразований STX, развивающего основные идеи языка XSLT, и XLink-процессора SXLink. STX и SXLink обеспечивают мощную и гибкую технологию для основанных на XML Электронных Библиотек. Языки XSLT и XLink могут значительно выиграть от их взаимной интеграции, и объединение STX и SXLink с использованием функциональных методов предоставляет практический подход к подобной интеграции.

Список литературы

[1]

Язык преобразований XSL (XSLT) Версия 1.0. Рекомендация W3C от 16 ноября 1999.

<http://www.rol.ru/news/it/helpdesk/xslt01.htm>

[2]

Cuneiform Digital Library Initiative to Use XML Encoding for Third Millennium Texts.

<http://xml.coverpages.org/ni2001-11-06-c.html>

[3]

XML Linking Language (XLink) Version 1.0. W3C Recommendation 27 June 2001.

<http://www.w3.org/TR/xlink/>

[4]

XSLT Requirements Version 2.0. W3C Working Draft, February 2001.

<http://www.w3.org/TR/2001/WD-xslt20req-20010214>

[5]

Extensible Markup Language (XML) 1.0 (Second Edition). W3C Recommendation 6 October 2000.

<http://www.w3.org/TR/REC-xml>

[6]

Namespaces in XML. World Wide Web Consortium 14-January-1999.

<http://www.w3.org/TR/REC-xml-names/>

[7]

XML XLink Requirements Version 1.0. W3C Note 24-Feb-1999.

<http://www.w3.org/TR/NOTE-xlink-req/>

[8]

Document Style Semantics and Specification Language (DSSSL). ISO/IEC 10179:1996(E).

<ftp://ftp.ornl.gov/pub/sgml/WG8/DSSSL/dsssl96b.pdf>

[9]

Kiselyov O. XML and Scheme. Workshop on Scheme and Functional Programming 2000, Montreal, 2000.

<http://www.okmij.org/ftp/Scheme/SXML-short-paper.html>

[10]

Oleg
Kiselyov,
Shriram Kris
hnamurthi.
SXSLT:
Manipulation
Language for
XML.
Practical

Aspects of
Declarative
Languages,
5th
International
Symposium,
PADL 2003.

[http://link
.springer.d
e/link/serv
ice/series/
0558/bibs/2
562/2562025
6.htm](http://link.springer.de/link/service/series/0558/bibs/2562/25620256.htm)

[11]

O. Kiselyov, K. Lisovsky. XML, XPath, XSLT Implementation as XML, XPath and SXSLT. International Lisp Conference ILC 2002, San Francisco. October, 2002.

<http://www.okmij.org/ftp/papers/SXs.pdf>

[12]

