Чернов А.П., Князев В.Н.
Вопросы разработки табличных процессоров

*Пензенский государственный университет
(Россия, Пенза)*

doi:10.18411/spc-16-01-2018-11

idsp: 000001:spc-16-01-2018-11

В последнее время формат языка XML, наряду с JSON, стал одним из наиболее распространённых форматов для хранения структурированных данных. XML (расширяемый язык разметки) вобрал в себя все преимущества языка SGML, правила формальной корректности и простоту разметки свободной формы. Однозначная структура и интуитивно понятный синтаксис — это те качества, которые позволяют использовать язык XML для обработки данных в программах. [1,2]

Язык XML применяется в большом числе программ, из которых можно выделить офисный пакет Microsoft Office, подразумевающий хранение данных документов в виде Office Open XML, представляющего из себя архив, в котором хранятся XML файлы с данными, графика и другие объекты. Именно задача эффективной работы с Office Open XML- файлами ставится в данной работе.

К разрабатываемому табличному процессору для операционной системы специального назначения предъявляются следующие требования:

- быстроедействие;
- оптимальное использование ресурсов;
- сохранение пользовательских данных: исходя из того, что на реализацию полного функционала, аналогичного MS Excel, потребуется время, необходимо организовать работу таким образом, чтобы работа с документом не приводила к потере данных из-за нереализованного функционала.

В процессе разработки табличного процессора возникла проблема с объёмом памяти, занимаемым объектным представлением XML- структуры документа. Так как разработка ведётся на языке C# с использованием уже существующих библиотек, определяющих специфику языка разработки, было установлено, что стандартные средства библиотеки System.XML не соответствуют требованиям по оптимальному использованию оперативной памяти. Дальнейшее решение проблемы требовало вникания в суть проблемы и рассмотрения возможных вариантов решения проблемы.

Для того, чтобы прочитать либо записать информацию на языке XML программе необходим соответствующий инструментарий. Программные модули, применяющиеся для чтения и записи XML, называются XML-процессорами. XML-процессоры можно классифицировать по способу применения: для чтения, для записи либо для чтения и записи одновременно.

Выделяют следующие типы XML-процессоров для чтения: событийные, потоковые, объектные. [3,4]

Событийные процессоры предполагают последовательное чтение текста XML, и разбор структуры XML заключается в вызове callback-функций при возникновении определённых событий, таких как появление открывающего тега, закрывающего тега, текстовой строки, атрибута и т.п.

К плюсам событийных процессоров можно отнести следующие: быстроедействие, низкие потребности в использовании оперативной памяти, простая реализация.

К минусам событийных процессоров можно отнести следующие: XML с перепутанным порядком тегов, которые можно считать «почти верными» будут интерпретированы как ошибочные; сложное применение на практике – необходимо

помнить, в каком месте документа происходит чтение и текущий контекст; если документ содержит много перекрёстных ссылок, приходится держать в памяти эти ссылки в строковом виде до тех пор, пока файл не будет считан до конца и станет возможно преобразовать строковые идентификаторы в указатели; при ошибке необходимо реализовать программное уничтожение уже созданной полузаполненной структуры, хранящейся в оперативной памяти; не получится использовать этот же алгоритм для записи.

Потоковые процессоры работают аналогично потокам ввода-вывода. Процессор этого типа действует как курсор, который размещается сразу после разобранный самой последней XML-лексемы и предоставляет методы для получения информации о ней. Этот подход очень эффективно использует память, так как не создает новых объектов.

К плюсам потоковых процессоров можно отнести следующие:

- высокое быстродействие;
- требуют мало памяти для работы;
- место в документе неявно задаётся местом в потоке выполнения.

К минусам потоковых процессоров можно отнести следующие:

- XML с перепутанным порядком тегов, которые можно считать «почти верными» будут интерпретированы как ошибочные;
- сложная реализация;
- если документ содержит много перекрёстных ссылок, приходится держать в памяти эти ссылки в строковом виде до тех пор, пока файл не будет считан до конца и станет возможно преобразовать строковые идентификаторы в указатели, возможно реализовать простановку ссылок при обратном проходе, но такой подход потребует дополнительных ресурсов;
- при ошибке необходимо реализовать программно освобождение ресурсов из-под уже созданной полузаполненной структуры, хранящейся в оперативной памяти;
- не получится использовать этот же алгоритм для записи.

Объектные процессоры (DOM – Document Object Model, объектная модель документа) воссоздают объектную модель по считанному текстовому представлению документа.

К плюсам объектных процессоров можно отнести следующие:

- относительная простота в программировании;
- в большинстве случаев, «почти верные» XML удастся корректно интерпретировать;
- в том случае, когда XML содержит много перекрёстных ссылок достаточно двух проходов: при первом происходит поиск ссылок, при втором их связывание;
- при ошибке в XML полусозданная модель в оперативной памяти будет автоматически очищена самой библиотекой;
- для чтения и записи используется один и тот же интерфейс.

К минусам объектных процессоров можно отнести следующие:

- более требовательны к памяти относительно других типов процессоров;
- довольно низкое быстродействие;
- ограниченный объём XML файла, с которым можно работать.

Для записи XML выделяют следующие типы процессоров: прямой записи, объектный. [3,4]

Процессоры прямой записи записывают содержимое XML последовательно, элемент за элементом.

К плюсам процессоров прямой записи можно отнести следующие:

- нет промежуточных объектов;
- высокое быстродействие.

К минусам процессоров прямой записи можно отнести следующие:

- при некоторых специфических задачах непригодны;
- примитивный процессор может генерировать неоптимальные пустые теги(<tag></tag>), а реализация более оптимального сложнее в программировании;
- при ошибке во время записи можно потерять данные;
- подходят только для записи;
- при ошибке программиста может получиться некорректный документ.

Объектная модель DOM соответствует аналогичной из пункта про запись.

К плюсам объектной модели можно отнести следующие:

- универсальность;
- единственный вариант, когда могут быть потеряны данные – это когда недостаточно места на диске, во всех других случаях исходный файл считывается в модель и данные хранятся в первоизданном виде.

К минусам объектной модели можно отнести следующие:

- объектные процессоры более требовательны к памяти относительно других типов процессоров;
- довольно низкое быстродействие.

Был проведён сравнительный анализ SAX-процессоров и объектных (DOM) процессоров. Результаты сравнения приведены в таблице 1.

Таблица 1

Сравнительный анализ XML-процессоров

Критерий	SAX	DOM
Минимизация потери данных	-	+
Оптимальное использование памяти	+	-
Быстродействие	+	-
Возможность как чтения, так и записи	-	+

Из анализа моделей XML-процессоров был сделан вывод, что для решения поставленной задачи необходимо использовать процессор, основанный на работе с DOM-представлением XML-документа.

Была рассмотрена реализация DOM модели из библиотеки System.XML. В результате анализа, был сделан вывод, что существующий алгоритм нуждается в доработке, так как расходует слишком много оперативной памяти. DOM-модель из библиотеки System.XML имеет такую особенность, что она хранит исходный текст узла на языке XML в памяти, что повышает скорость сохранения файла, так как для сохранения происходит простая последовательная запись узлов один за другим. Однако это не оптимальным образом расходует память и увеличивает время работы с документом.

Предлагается следующий алгоритм для построения DOM-модели документа. При чтении файла происходит его обход SAX-парсером, структура документа сохраняется в объектном виде без сохранения строкового представления элементов. Таким образом, DOM-дерево станет меньше занимать места в оперативной памяти, повысится быстродействие при работе с ним. Увеличится только время сохранения файла, так как придётся объектное представление конвертировать в текстовое.

Если документ состоит из 2 элементов: родительского и дочернего, то его текстовое представление в памяти будет содержать 4 тега (1 в тексте дочернего элемента и 3 в тексте родительского). Если эту иерархию расширить до 3

последовательно вложенных элементов, то результирующий текст будет содержать $1+3+5 = 9$ элементов.

Таким образом если документ состоит из 50 элементов, каждый из которых является потомком другого, то его объектная модель будет содержать в текстовом виде 502 тегов.

В предлагаемой модели это число можно уменьшить до 1 на каждый тэг: вместо дублирования этого текста в открывающем и закрывающем тегах и дублирования дочерних элементов в тексте родительских можно просто запоминать имя элемента.

Таким образом, представляется возможным сохранить больший объём памяти не занятым под хранение объектной модели, нежели этого требует реализация из библиотеки System.XML.

Предлагаемый комбинированный алгоритм подразумевает снижение используемого объёма оперативной памяти, увеличение быстродействия, что соответствует требованиям, налагаемым на разрабатываемые программные средства. Так как была использована модель DOM, при чтении файла происходит считывание всего текста файла, а при его сохранении никакие данные не теряются, что также соответствует заданным требованиям.

Предлагаемый алгоритм будет реализован при разработке табличного процессора для операционной системы специального назначения.

1. Рэй, Э. – Perl & XML. Библиотека программиста / Э. Рэй, Дж. Макинтош. — СПб.: Питер, 2003.— 208 с
2. Одиночкина, С.В. Основы технологий XML / С.В. Одиночкина. - СПб: НИУ ИТМО, 2013. – 56 с.
3. Сравнение XmlReader и SAX Reader. - URL: [https://msdn.microsoft.com/ru-ru/library/sbw89de7\(v=vs.110\).aspx](https://msdn.microsoft.com/ru-ru/library/sbw89de7(v=vs.110).aspx) (дата обращения 10.01.2018)
4. Модель DOM для XML. - URL: [https://msdn.microsoft.com/ru-ru/library/hf9hbf87\(v=vs.110\).aspx](https://msdn.microsoft.com/ru-ru/library/hf9hbf87(v=vs.110).aspx) (дата обращения 10.01.2018)