Московский государственный технический университет
имени Н.Э. Баумана

Кафедра «Системы обработки информации и управления»

к.т.н. профессор Э.Н. Самохвалов
к.т.н. доцент Г.И. Ревунков
к.т.н. доцент Ю.Е. Гапанюк

Методические указания
к лабораторным работам по курсу
XML – технологии
Часть 2
(4 семестр)

Москва
2013

СОДЕРЖАНИЕ

1 Теоретическая часть

1.1 Использование DTD для описания структуры документов XML

DTD является одним из способов проверки правильности структуры документа XML. Исторически он появился даже ранее, чем технология XML, так как DTD-описания перешли в XML из SGML.

DTD расшифровывается как Document Type Definition, описание типов документа.

В XML-документах DTD определяет набор используемых элементов, идентифицирует элементы, которые могут использоваться внутри других элементов, определяет возможные атрибуты для каждого элемента.

1.1.1 Пример внешнего DTD

Рассмотрим пример использования DTD для документа XML. В этом примере используется внешний DTD, который располагается в отдельном файле. В XML-документ встраивается ссылка на этот внешний файл.

Пример 1.

Файл XML:

```
<?xml version="1.0" encoding="Windows-1251"?>
<!DOCTYPE languages SYSTEM "languages.dtd" >
<languages>
     <language id="1">
          <name>HTML</name>
          <year>01.01.1990</year>
          <howold>19</howold>
     </language>
     <language id="2">
          <name>XML</name>
          <year>01.01.1998</year>
          <howold>11</howold>
```

```
    </language>
    <language id="3">
        <name>SGML</name>
        <year>01.01.1986</year>
        <howold>23</howold>
    </language>
    <empty attr1="1" attr2="текст"/>
    <CDATA_Example>
        <![CDATA[<<<<<<<<<<   >>>>>>>>>>]]>
    </CDATA_Example>
</languages>
```

Файл DTD:

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- В файле DTD могут быть использованы комментарии, если объявлена
инструкция обработки xml -->

<!ELEMENT name (#PCDATA)>
<!ELEMENT year (#PCDATA)>
<!ELEMENT howold (#PCDATA)>
<!ELEMENT CDATA_Example (#PCDATA)>

<!ELEMENT language (name, year, howold)>
<!ATTLIST language
        id CDATA #REQUIRED>

<!ELEMENT empty EMPTY>
<!ATTLIST empty
        attr1 CDATA #REQUIRED
        attr2 CDATA #REQUIRED>

<!ELEMENT languages (language+, empty, CDATA_Example)>
```

Проверить соответствие XML-документа файлу DTD можно с использованием XMLPad.

Для этого необходимо открыть документ XML и выбрать пункт меню «XML/Validate».

Если документ «валиден», то есть соответствует DTD, то выдается сообщение об отсутствии ошибок.

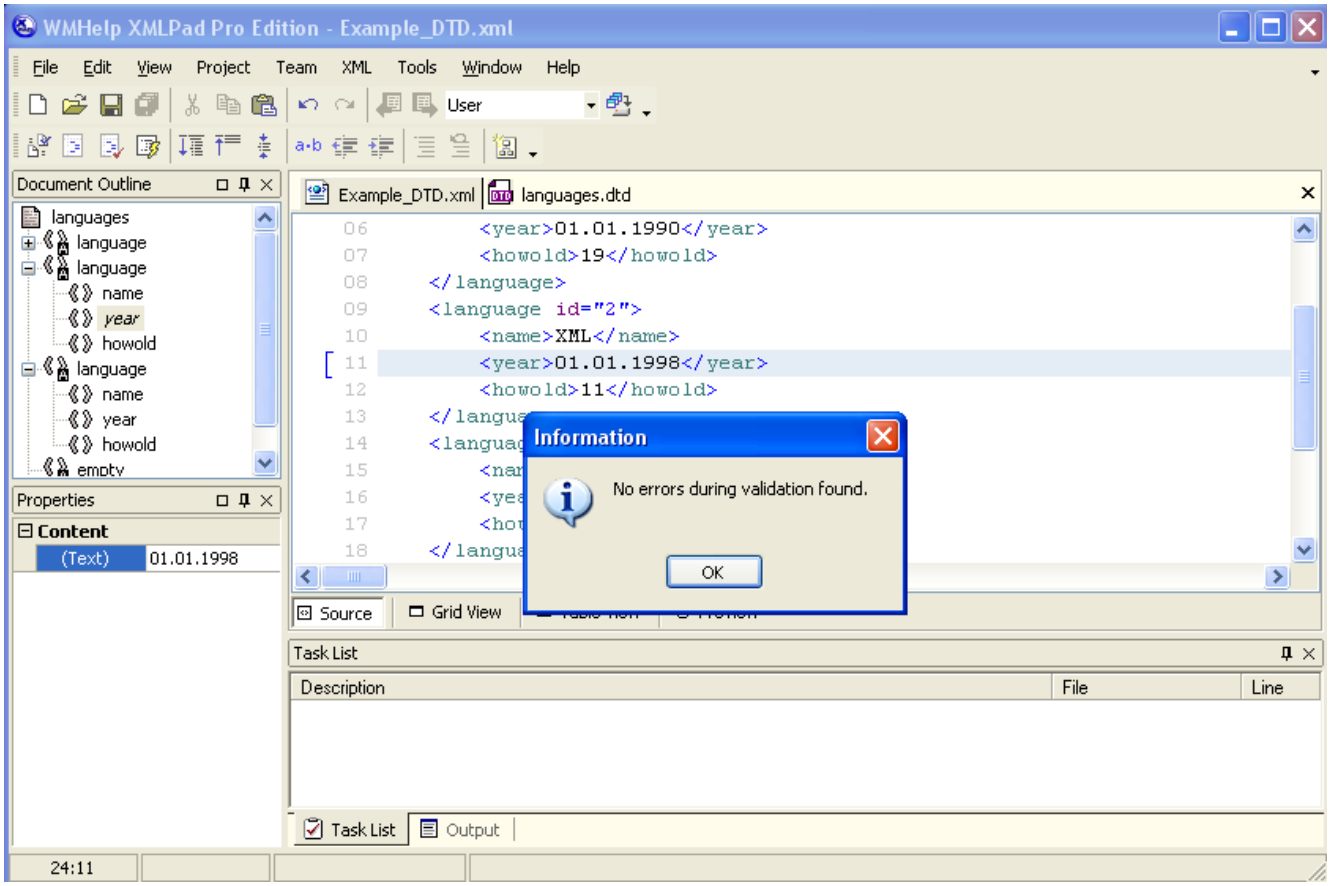


Рис. 1. Результат выполнения примера 1.

Присоединение DTD-файла к документу XML производится во второй строке XML-документа:

```
<!DOCTYPE languages SYSTEM "languages.dtd" >
```

Имя файла, который содержит DTD – «languages.dtd». К файлу DTD может быть указан полный путь, если он расположен в другом каталоге.

Рассмотрим более подробно текст DTD-описания.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- В файле DTD могут быть использованы комментарии, если объявлена инструкция обработки xml -->
```

DTD-документ может быть объявлен как XML-документ, то есть начинаться с инструкции обработки <?xml . . . ?>. Это дает возможность использовать в

документе XML-комментарии. Если комментарии не нужны, то инструкция обработки <?xml . . . ?> может быть пропущена.

```
<!ELEMENT name (#PCDATA)>
<!ELEMENT year (#PCDATA)>
<!ELEMENT howold (#PCDATA)>
<!ELEMENT CDATA_Example (#PCDATA)>
```

С помощью команды !ELEMENT объявляется элемент в документе. После команды !ELEMENT следует название элемента, в скобках указывается содержимое элемента.

#PCDATA расшифровывается как «parsed character data», разбираемые символьные данные. Это данные, которые будут разбираться анализатором, например, они могут включать секцию <![CDATA[ ]]>.

Элементы, обозначенные как #PCDATA, могут включать текст, но не могут включать вложенные элементы.

При объявлении атрибутов также используется обозначение CDATA – это данные, которые не будут разбираться анализатором.

```
<!ELEMENT language (name, year, howold)>
```

В элемент language могут быть вложены элементы name, year, howold. Они должны следовать именно в таком порядке, каждый элемент встречается один раз.

После названия элемента, а также после выражения в скобках могут встречаться символы «?», «+» и «*». Эти символы определяют количество вхождений элемента.

«?» – элемент встречается 0 или 1 раз.

«*» – элемент встречается 0 и более раз (итерация).

«+» – элемент встречается 1 и более раз (позитивная итерация).

То есть используется способ описания, похожий на тот, который применяется в дискретной математике для описания цепочек символов, допускаемых автоматом.

Примеры:

```
<!ELEMENT language (name?, year*, howold+)>
<!ELEMENT language (name, year, howold)*>
```

Если символ стоит после скобок, то он применяется ко всему выражению в скобках. Например

```
<!ELEMENT language (name?, year*, howold+)+>
```

означает, что элемент name должен встречаться 0 или 1 раз, за ним следуют элементы year, которые встречаются 0 или более раз, затем следуют элементы howold, которые встречаются один или более раз. Последовательность элементов name?, year*, howold+ может встречаться один или более раз, это указывает символ «+» после выражения в скобках.

Символ « , » между элементами означает строгое следование элементов друг за другом. Также может использоваться символ « | », который обозначает, что может встречаться один или другой элемент, например:

```
<!ELEMENT language ((name,year,howold)|(year,name,howold))>
```

означает, что в элемент language могут быть вложены элементы name,year,howold или year,name,howold.

```
<!ELEMENT language (((name,year)|(year,name)),howold)>
```

означает, что в элемент language могут быть вложены элементы name,year или year,name и элемент howold

```
<!ELEMENT language (name|year|howold)*>
```

означает, что в элемент language может быть вложен элемент name или year или howold 0 или более раз. Фактически это означает, что элементы name,year,howold могут быть вложены в любой последовательности любое количество раз.

```
<!ATTLIST language
          id CDATA #REQUIRED>
```

С помощью команды !ATTLIST определяются атрибуты элемента. После команды !ATTLIST следует название элемента, далее перечисляются атрибуты. Для определения атрибута используется три идентификатора.

Первый идентификатор – название атрибута, в нашем случае id.

Второй идентификатор определяет тип данных. Чаще всего для обозначения типа используется CDATA, то есть любые данные.

Также может использоваться идентификатор ID, который определяет уникальное значение атрибута:

```
<!ATTLIST language id ID #REQUIRED>
```

В качестве типа могут быть перечислены возможные значения атрибута (1 или 2 или 3):

```
<!ATTLIST language id (1|2|3) #REQUIRED>
```

Полный список типов данных приведен в спецификации.

Третий идентификатор определяет обязательность использования атрибута. #REQUIRED означает обязательный атрибут, #IMPLIED означает необязательный атрибут.

#FIXED "значение" определяет, что значение атрибута фиксировано и не может быть изменено: Например

```
<!ATTLIST language id CDATA #FIXED "1">
```

```
<!ELEMENT empty EMPTY>
<!ATTLIST empty
          attr1 CDATA #REQUIRED
          attr2 CDATA #REQUIRED>
```

Если при объявлении элемента вместо выражения в скобках используется EMPTY, то элемент объявляется как пустой, то есть не имеющий содержимого. В этом примере элемент empty объявляется как пустой элемент с двумя атрибутами: attr1и attr2.

```
<!ELEMENT languages (language+, empty, CDATA_Example)>
```

Элемент languages содержит один или более элементов language, элемент empty и элемент CDATA_Example.

1.1.2 Пример несоответствия документа XML и DTD

Внесем изменения в элемент <language id="1"> исходного документа. Изменения выделены полужирным шрифтом. В элемент language добавлены новый атрибут attr и новый элемент qwerty.

Пример 2.

```
<language id="1" attr="123">
    <qwerty>qwerty</qwerty>
    <name>HTML</name>
    <year>01.01.1990</year>
    <howold>19</howold>
</language>
```

При проверке документа на соответствие DTD возникают следующие сообщения об ошибках (в панели Task List):

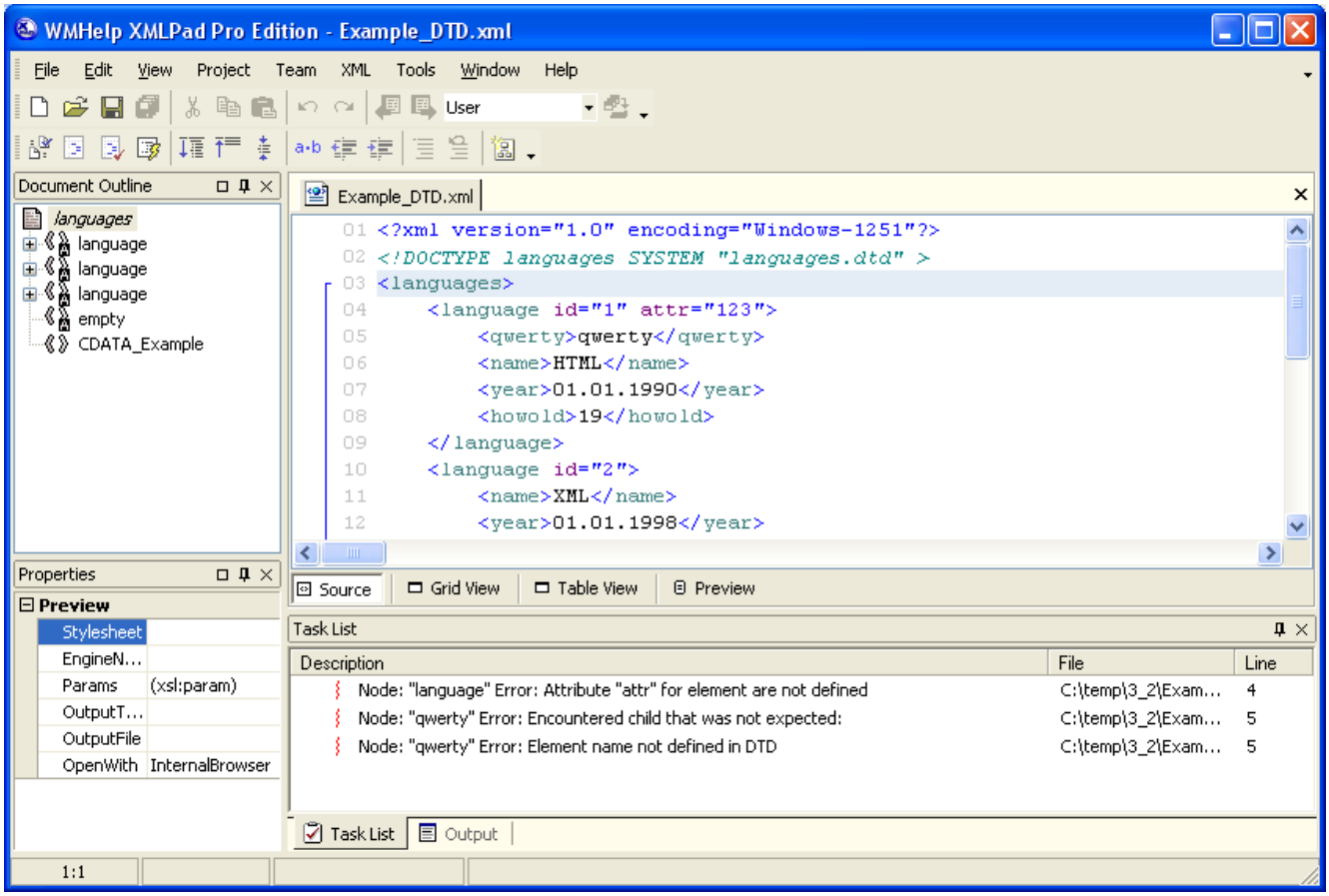


Рис. 2. Результат выполнения примера 2.

1.1.3 Пример встроенного DTD

DTD может быть встроен в документ XML.

Пример 3.

```
<?xml version="1.0" encoding="Windows-1251"?>
<!DOCTYPE languages [

<!ELEMENT name (#PCDATA)>
<!ELEMENT year (#PCDATA)>
<!ELEMENT howold (#PCDATA)>
<!ELEMENT CDATA_Example (#PCDATA)>

<!ELEMENT language (name, year, howold)>
<!ATTLIST language
          id CDATA #REQUIRED>

<!ELEMENT empty EMPTY>
<!ATTLIST empty
          attr1 CDATA #REQUIRED
          attr2 CDATA #REQUIRED>

<!ELEMENT languages (language+, empty, CDATA_Example)>
]>

<languages>
     <language id="1">
          <name>HTML</name>
          <year>01.01.1990</year>
          <howold>19</howold>
     </language>
     <language id="2">
          <name>XML</name>
          <year>01.01.1998</year>
          <howold>11</howold>
     </language>
```

```
    <language id="3">
        <name>SGML</name>
        <year>01.01.1986</year>
        <howold>23</howold>
    </language>
    <empty attr1="1" attr2="текст"/>
    <CDATA_Example>
        <![CDATA[<<<<<<<<<<    >>>>>>>>>>]]>
    </CDATA_Example>
</languages>
```

Результат валидации документа (пункт меню «XML/Validate»):

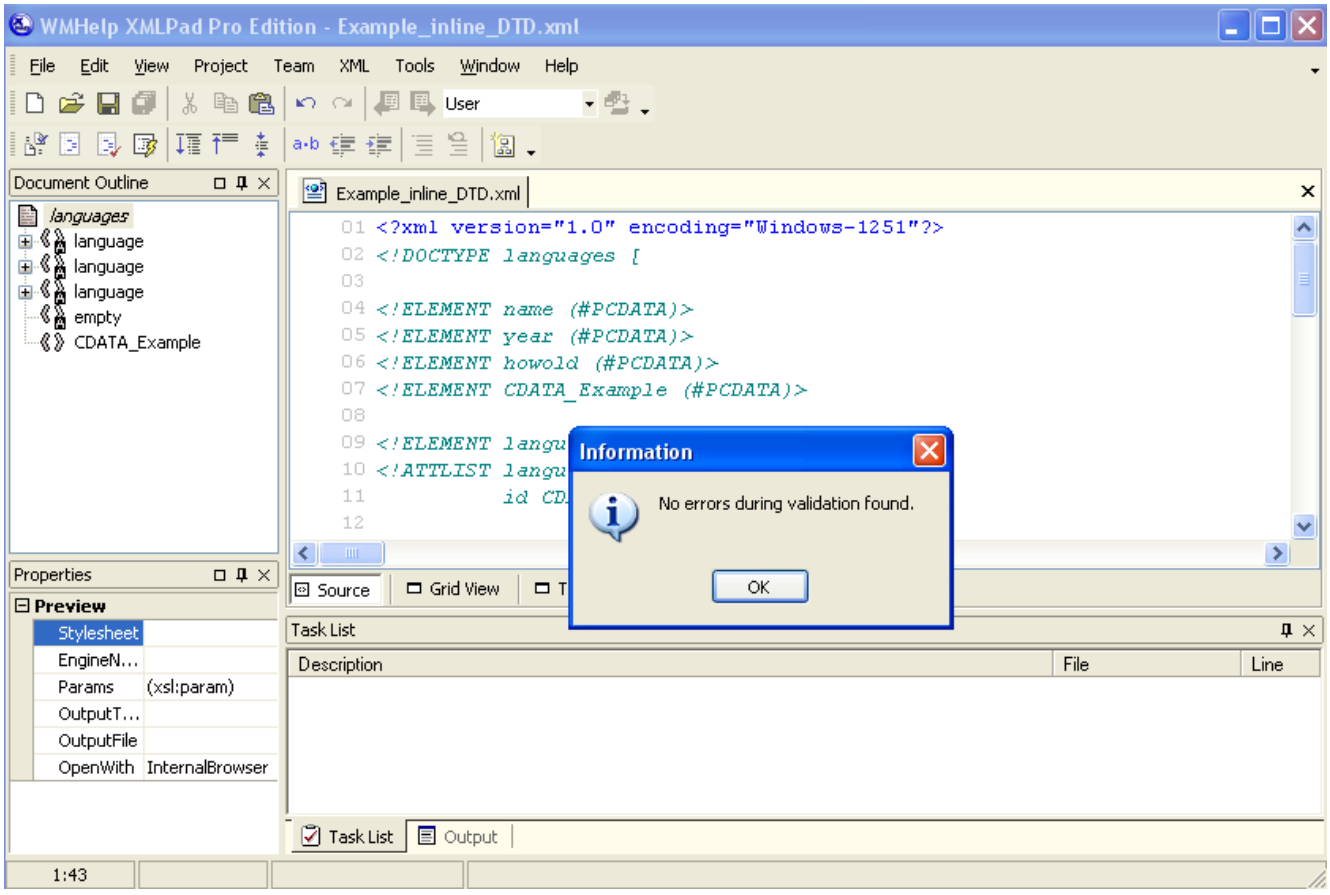


Рис. 3. Результат выполнения примера 3.

1.1.4 Графическое представление DTD

Редактор XMLPad позволяет представить внешнюю DTD в графическом виде. Для этого необходимо открыть файл DTD и выбрать вкладку «Diagram».

Для DTD и XML-схем используется одинаковая графическая нотация. Такая же графическая нотация для DTD и XML-схем используется в редакторе XML SPY.

Пример 4.

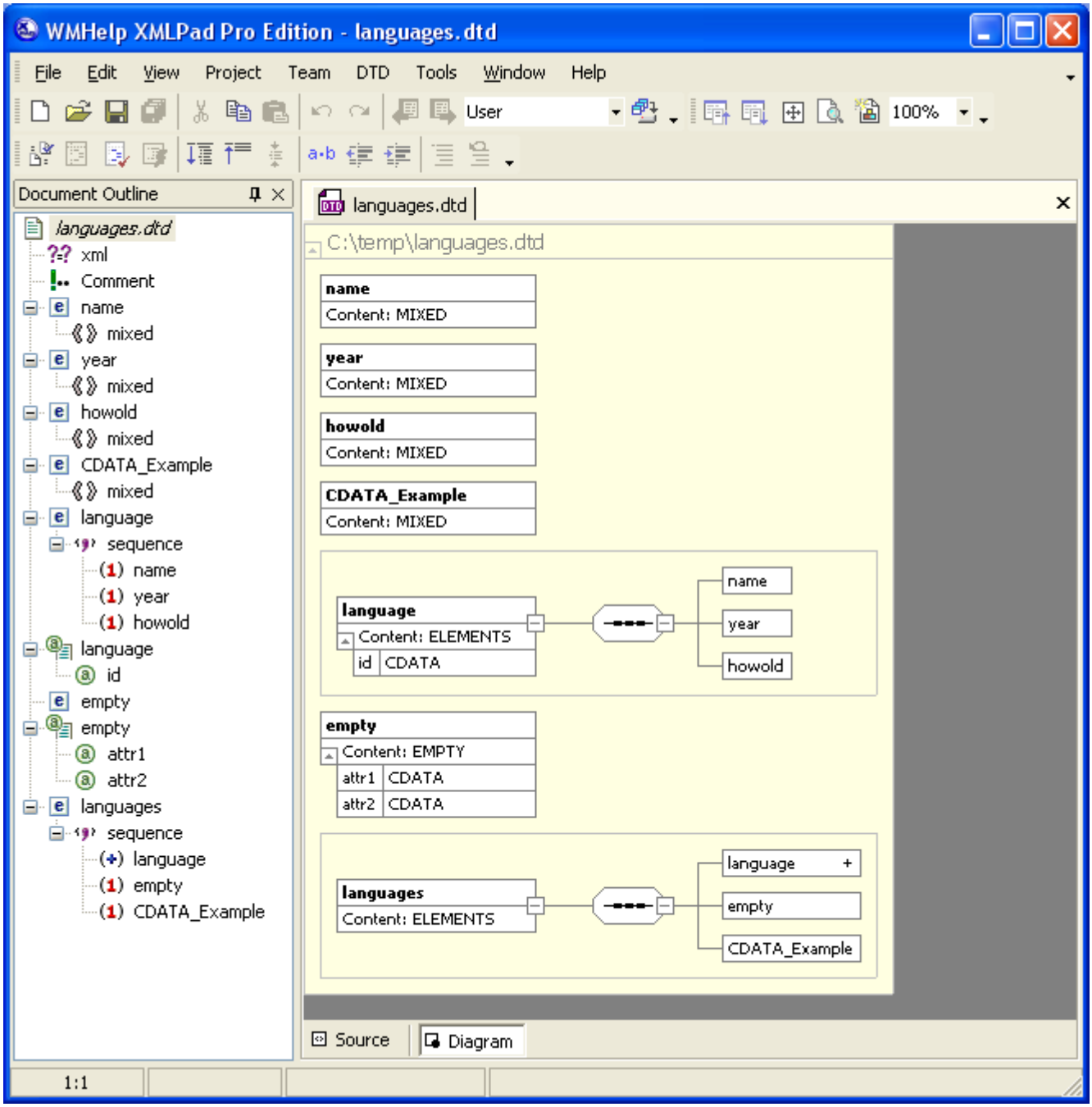


Рис. 4. Результат выполнения примера 4.

Если объявить элемент language следующим образом:

```
<!ELEMENT language (name? | year+ | howold*)+>
```

то вид диаграммы будет таким:

Пример 5.

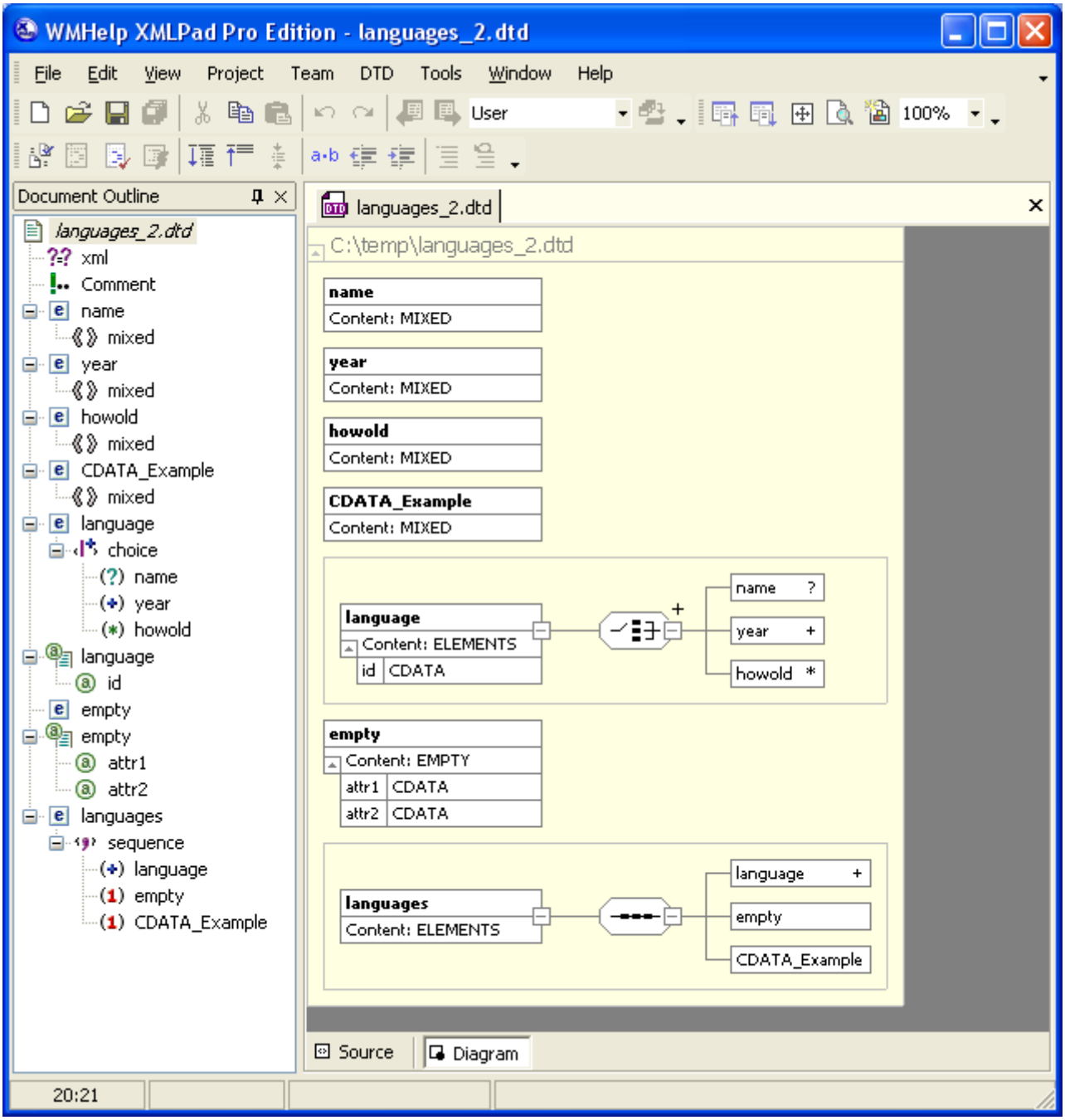
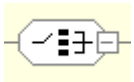


Рис. 5. Результат выполнения примера 4. Изменение 1.

На диаграмме используются следующие обозначения:

Последовательное соединение элементов, соответствует « , »



Выбор элементов, соответствует « | »

1.1.5 Генерация DTD и XML-схемы по XML-документу

XMLPad позволяет сгенерировать DTD или схему XML по документу XML.

Для этого необходимо открыть документ XML, который не связан с DTD или XML-схемой и выбрать пункт меню «XML/Create Schema».

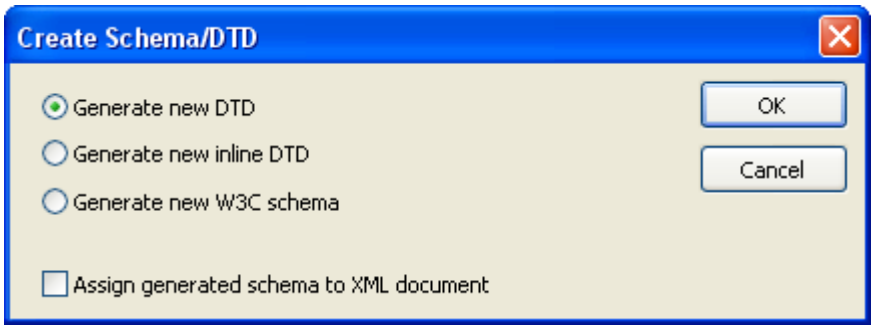


Рис. 6. Генерация DTD и XML-схемы по XML-документу.

Существует возможность автоматической генерации DTD, встроенного DTD или XML-схемы.

Также предусмотрена возможность привязки созданного DTD или XML-схемы к XML-документу, при этом в документ XML вставляется соответствующая инструкция.

Необходимо отметить, что полученный DTD или XML-схема почти всегда является «полуфабрикатом», который требуется дорабатывать вручную.

Это происходит по двум причинам. Во-первых, программа генерации может «ошибиться» и сгенерировать не совсем точный результат. Но главная причина в том, что в одном документе, как правило, не присутствуют все возможные варианты, которые должны быть учтены в DTD или XML-схеме.

Например, программа генерации может создать перечисление некоторых вариантов. Но это перечисление не учитывает все возможные случаи, и его необходимо заменить на итерацию.

1.1.6 Преобразование DTD в XML-схему и XML-схемы в DTD

В XMLPad существует возможность преобразования DTD в XML-схему (пункт меню «DTD/Convert to XSD»).

В режиме XML-схемы также существует возможность обратного преобразования «XSD/Convert to DTD».

Необходимо учитывать, что программа генерации может сгенерировать результат, требующий корректировки.

В обоих режимах существует возможность генерации примера документа XML, который соответствует DTD или XML-схеме. Пункт меню «DTD/Generate sample XML file» или «XSD/Generate sample XML file».

1.2 Использование схем XML для описания структуры документов XML

XML-схемы являются альтернативой DTD. Они, также как и DTD, определяют набор используемых элементов, идентифицируют элементы, которые могут использоваться внутри других элементов, определяют возможные атрибуты для каждого элемента. По сравнению с DTD, схемы обеспечивают более понятный способ описания документов. Схемы XML, в отличие от DTD, являются XML – документами.

Как отмечалось ранее, в DTD для описания содержимого элемента используется способ, похожий на описание цепочек символов, допускаемых автоматом. В схемах XML используется способ, более привычный для программиста: определяются типы данных и указывается принадлежность элементов XML к этому типу данных.

1.2.1 Пример XML-схемы

Пример 6.

Файл XML:

```
<?xml version="1.0" encoding="Windows-1251"?>
<languages xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="Example_schema.xsd">
     <language id="1">
          <name>HTML</name>
          <year>01.01.1990</year>
```

```
        <howold>19</howold>
    </language>
    <language id="2">
        <name>XML</name>
        <year>01.01.1998</year>
        <howold>11</howold>
    </language>
    <language id="3">
        <name>SGML</name>
        <year>01.01.1986</year>
        <howold>23</howold>
    </language>
    <empty attr1="1" attr2="текст"/>
    <CDATA_Example>
        <![CDATA[<<<<<<<<<<   >>>>>>>>>>]]>
    </CDATA_Example>
</languages>
```

Файл XSD:

```
<?xml version="1.0" encoding="utf-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">

    <xsd:complexType name="languageType">
        <xsd:sequence>
            <xsd:element name="name" type="xsd:string"/>
            <xsd:element name="year" type="xsd:string"/>
            <xsd:element name="howold" type="xsd:integer"/>
        </xsd:sequence>
        <xsd:attribute name="id" use="required" type="xsd:string"/>
    </xsd:complexType>

   <xsd:complexType name="emptyType">
     <xsd:attribute name="attr1" type="xsd:string" use="required"/>
     <xsd:attribute name="attr2" type="xsd:string" use="required"/>
   </xsd:complexType>
```

```
    <xsd:element name="languages">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="language" type="languageType"
maxOccurs="unbounded"/>
                <xsd:element name="empty" type="emptyType"/>
                <xsd:element name="CDATA_Example"
type="xsd:string"/>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>

</xsd:schema>
```

Проверить соответствие XML-документа схеме, как и в случае DTD, можно с использованием пункта меню «XML/Validate».

Если документ действителен («валиден»), то есть соответствует схеме, то выдается сообщение об отсутствии ошибок. Если документ содержит ошибки, то они выдаются в панели Task List.

Если схема XML набирается вручную в XMLPad, то для проверки правильности синтаксиса можно использовать пункт меню «XSD/Validate».

Также полезным может быть пункт меню «XSD/Change Document Namespace Prefix», который позволяет изменять префикс пространства имен.

Присоединение схемы к документу XML производится при объявлении корневого элемента документа XML.

```
<languages xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="Example_schema.xsd">
```

Для присоединения схемы используются пространства имен. Объявляется пространство имен со значением "http://www.w3.org/2001/XMLSchema-instance", это пространство имен используется для подключения в документ схемы XML.

Префикс пространства имен может выбираться произвольно, в нашем примере для префикса выбрано значение xsi.

Далее указывается атрибут xsi:noNamespaceSchemaLocation. Этот атрибут принадлежит указанному пространству имен, на что указывает префикс xsi перед названием атрибута. То есть этот атрибут является служебным, и относится не к документу, а используется для подключения XML-схемы. Значением атрибута является URI файла, содержащего схему XML.

Это наиболее простой способ подключения схемы, далее будут рассмотрены другие способы.

Рассмотрим более подробно текст XML-схемы.

```
<?xml version="1.0" encoding="utf-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
```

XML-схема является XML-документом, корневым элементом является элемент schema.

Элементы схемы принадлежат пространству имен "http://www.w3.org/2001/XMLSchema". В качестве префикса пространства имен обычно используется xsd или xs.

```
<xsd:complexType name="languageType">
    <xsd:sequence>
        <xsd:element name="name" type="xsd:string"/>
        <xsd:element name="year" type="xsd:string"/>
        <xsd:element name="howold" type="xsd:integer"/>
    </xsd:sequence>
    <xsd:attribute name="id" use="required" type="xsd:string"/>
</xsd:complexType>
```

Объявление сложного (составного) типа с названием languageType. Этот тип определяет последовательность элементов (xsd:sequence). В последовательность входят три элемента name, year, howold, они простого строкового (xsd:string) или целого (xsd:integer) типа.

Также определяется атрибут id, обязательный (use="required"), строкового типа (type="xsd:string").

Составной тип определяет содержимое какого-либо элемента, но название самого элемента в типе не указывается. Несколько элементов с разными названиями могут быть одного типа.

```
<xsd:complexType name="emptyType">
  <xsd:attribute name="attr1" type="xsd:string" use="required"/>
  <xsd:attribute name="attr2" type="xsd:string" use="required"/>
</xsd:complexType>
```

Объявление типа с названием emptyType. Этот тип соответствует пустому элементу, так как в нем не объявлено содержимое элемента. В этом типе объявлено два атрибута.

Типы, объявленные на уровне схемы, называются глобальными типами.

```
 <xsd:element name="languages">
      <xsd:complexType>
           <xsd:sequence>
                <xsd:element name="language" type="languageType"
maxOccurs="unbounded"/>
                <xsd:element name="empty" type="emptyType"/>
                <xsd:element name="CDATA_Example"
type="xsd:string"/>
           </xsd:sequence>
      </xsd:complexType>
 </xsd:element>
</xsd:schema>
```

В схеме также объявлен глобальный элемент languages. Обычно в качестве глобального объявляют корневой элемент документа.

В xsd:element вложен xsd:complexType. Это означает, что элемент languages составного типа. Поскольку на вложенный тип нет ссылок из других элементов, то атрибут name для xsd:complexType не используется, то есть тип является анонимным.

Составной тип содержит последовательность из трех элементов.

Элемент language типа languageType может встречаться неограниченное количество раз (maxOccurs="unbounded"). Атрибут maxOccurs указывает максимальное количество вхождений элемента, атрибут minOccurs указывает минимальное количество вхождений элемента. Атрибут может содержать число вхождений ("0", "1" и т.д.) или значение "unbounded" (неограниченное количество вхождений).

В последовательность также входят элемент empty типа emptyType и элемент CDATA_Example типа xsd:string. Каждый из этих элементов должен встречаться один раз.

1.2.2 Графическое представление схемы XML

Редактор XMLPad позволяет представить XML-схему в графическом виде. Для этого необходимо открыть файл со схемой и выбрать вкладку «Diagram». На диаграмме показаны глобальные типы и глобальный элемент.

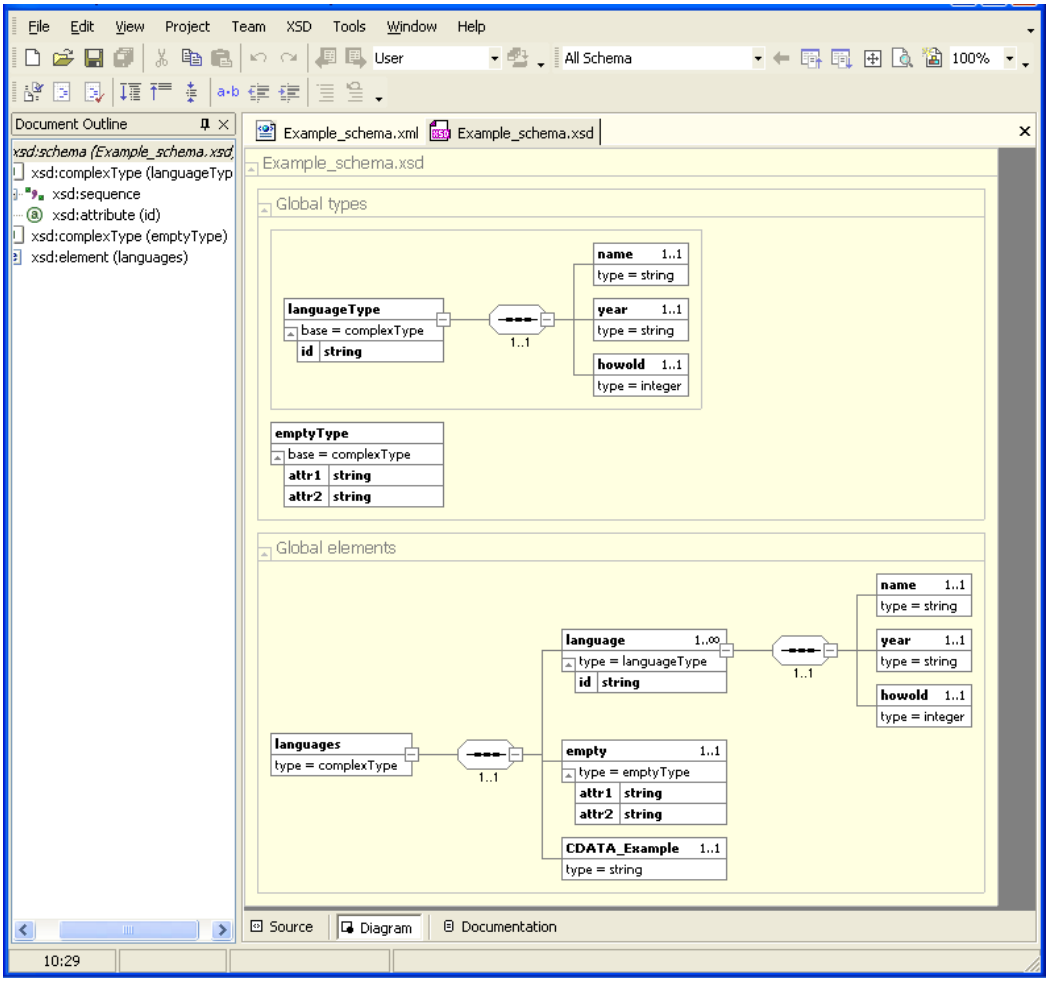


Рис. 7. Графическое представление схемы XML из примера 6.

Особенностью графического представления схемы является то, что некоторые фрагменты могут повторяться. Это связано с использованием типов. Например, элемент language типа languageType, поэтому содержимое составного типа languageType и элемента language одинаково.

Для генерации HTML-документа, содержащего подробное описание схемы необходимо выбрать вкладку «Documentation». При этом в текущем каталоге будет создан HTML-документ, содержащий документацию.

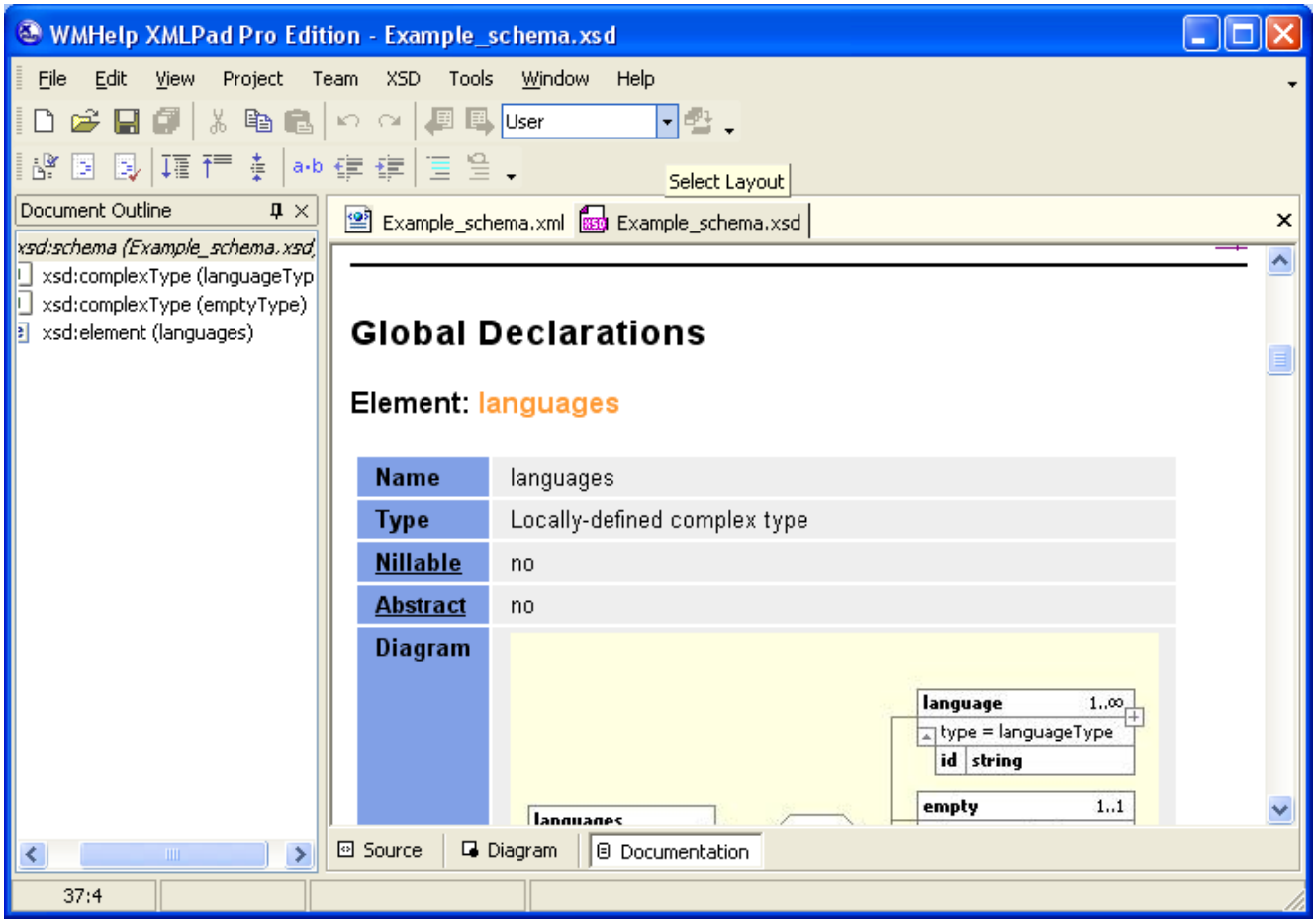


Рис. 8. Графическое представление схемы XML из примера 6. Формирование документации.

1.2.3 Использование простых типов и ограничений

Простыми (simple type) названются типы, которые определяют значение простого элемента или атрибута. Простой элемент содержит атомарное значение и не может иметь вложенных элементов.

Тип данных, который определяет элемент, в который вложены другие элементы, называется сложным или составным типом данных (complex type).

На следующем рисунке показана иерархия типов данных, приведенная в спецификации.

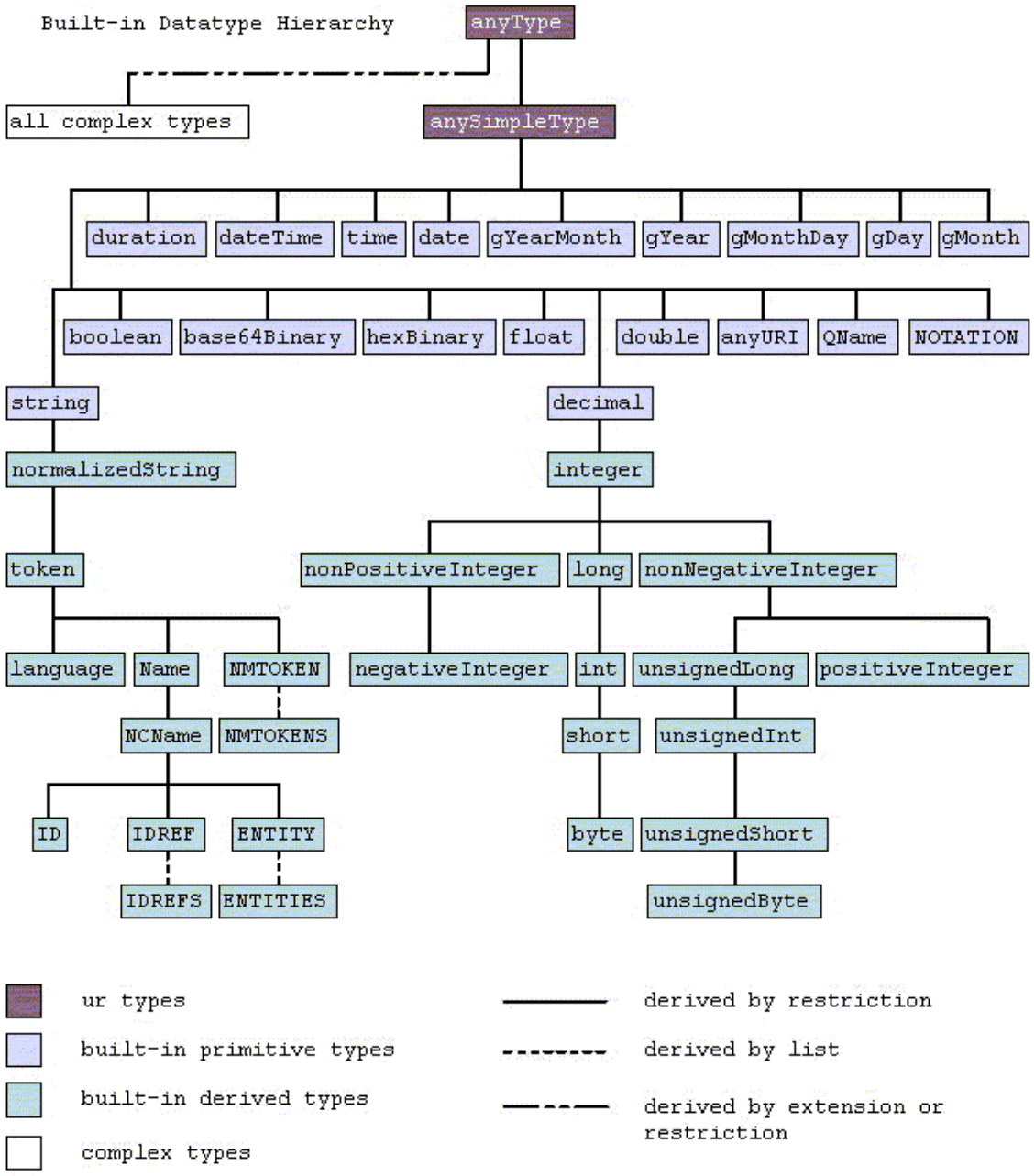


Рис. 9. Иерархия типов данных XML-схем.

Спецификация XML-схем позволяет вводить производные простые типы, накладывая ограничения на базовые простые типы.

Эти ограничения также назвают гранями, фасетами (facets).

Таблица 1. Ограничения строковых типов данных.

length	Ограничение на длину строки
minLength	Ограничение на длину строки снизу
maxLength	Ограничение на длину строки сверху
pattern	Задание шаблона строки регулярным выражением
enumeration	Перечисление элементов

Таблица 2. Ограничения числовых типов данных.

maxInclusive	Больше или равно
maxExclusive	Больше
minExclusive	Меньше
minInclusive	Меньше или равно
totalDigits	Количество цифр в числе
fractionDigits	Количество цифр в дробной части

Пример использования ограничений.

Пример 7.

Файл XML:

```
<?xml version="1.0" encoding="Windows-1251"?>
<!-- Примеры ограничений для простых типов -->

<Root_Element xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="facets.xsd">

   <Example_Length>1234</Example_Length>
   <Example_Length_MinMax>12345</Example_Length_MinMax>
   <Example_Length_MinMax>1234567</Example_Length_MinMax>

   <Example_Pattern>aaaaabbbbbcccccbbbbbaaaaaa</Example_Pattern>

   <domain>gov</domain>
   <domain>edu</domain>
```

```
   <Example_Inclusive>1</Example_Inclusive>
   <Example_Inclusive>5</Example_Inclusive>
   <Example_Inclusive>10</Example_Inclusive>

   <Example_Exclusive>2</Example_Exclusive>
   <Example_Exclusive>9</Example_Exclusive>

   <Example_Digits>123.45</Example_Digits>

</Root_Element>
```

Файл XSD:

```
<?xml version="1.0" encoding="Windows-1251"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">

    <xsd:simpleType name="Example_Length_Type">
       <xsd:restriction base="xsd:string">
          <xsd:length value='4'/>
       </xsd:restriction>
    </xsd:simpleType>
```

Простой тип с названием Example_Length_Type создается как ограничение (xsd:restriction), накладываемое на базовый строковый тип (xsd:string). В элемент xsd:restriction вкладываются элементы ограничений, в данном случае ограничение на общую длину строки.

```
    <xsd:simpleType name="Example_Length_MinMax_Type">
       <xsd:restriction base="xsd:string">
          <xsd:minLength value='5'/>
          <xsd:maxLength value='7'/>
       </xsd:restriction>
    </xsd:simpleType>

    <xsd:simpleType name="Example_Pattern_Type">
```

```
   <xsd:restriction base="xsd:string">
      <xsd:pattern value='(a|b|c)*'/>
   </xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name="Example_enumeration_Type">
   <xsd:restriction base="xsd:string">
      <xsd:enumeration value='gov'/>
      <xsd:enumeration value='edu'/>
      <xsd:enumeration value='com'/>
   </xsd:restriction>
</xsd:simpleType>

 <xsd:simpleType name="Example_Inclusive_Type">
   <xsd:restriction base="xsd:integer">
      <xsd:minInclusive value='1'/>
      <xsd:maxInclusive value='10'/>
   </xsd:restriction>
 </xsd:simpleType>

 <xsd:simpleType name="Example_Exclusive_Type">
   <xsd:restriction base="xsd:integer">
      <xsd:minExclusive value='1'/>
      <xsd:maxExclusive value='10'/>
   </xsd:restriction>
 </xsd:simpleType>

 <xsd:simpleType name="Example_Digits_Type">
   <xsd:restriction base="xsd:decimal">
      <xsd:totalDigits value='5'/>
      <xsd:fractionDigits value='2'/>
   </xsd:restriction>
 </xsd:simpleType>

 <xsd:complexType name="Root_Element_Type">
      <xsd:sequence>
```

```
                <xsd:element name="Example_Length"
type="Example_Length_Type" maxOccurs="unbounded"/>
                <xsd:element name="Example_Length_MinMax"
type="Example_Length_MinMax_Type" maxOccurs="unbounded"/>
                <xsd:element name="Example_Pattern"
type="Example_Pattern_Type"/>
                <xsd:element name="domain"
type="Example_enumeration_Type" maxOccurs="unbounded"/>

                <xsd:element name="Example_Inclusive"
type="Example_Inclusive_Type" maxOccurs="unbounded"/>
                <xsd:element name="Example_Exclusive"
type="Example_Exclusive_Type" maxOccurs="unbounded"/>
                <xsd:element name="Example_Digits"
type="Example_Digits_Type" maxOccurs="unbounded"/>
          </xsd:sequence>
     </xsd:complexType>
```

Составной тип, в котором определяются элементы для тестирования ограничений.

```
     <xsd:element name="Root_Element" type="Root_Element_Type"/>
</xsd:schema>
```

Определение корневого элемента документа.

1.2.4 Списки и объединения

Под списком в XML понимается возможность хранения списка значений в одном элементе XML. Значения в списке разделяются пробелами. Пример списка:

```
<Example_List>1 2 3 4 5 11 12 13 14 15</Example_List>
```

Объединение позволяет хранить в элементе значение одного или другого типа. Пример объединения, позволяющего хранить в элементе действительное число или список целых:

```
<Example_Union>1 2 3 4 5</Example_Union>
```

```
<Example_Union>123.123</Example_Union>
```

Списки и объединения относятся к простым типам. К спискам и объединениям можно применять ограничения.

Пример 8.

Файл XML:

```
<?xml version="1.0" encoding="Windows-1251"?>
<!-- Примеры списков и объединений -->

<Root_Element xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:noNamespaceSchemaLocation="list_union.xsd">

   <Example_List_Integer>1 2 3 4 5 11 12 13 14
15</Example_List_Integer>

   <Example_List_Integer_Length>1 2 3</Example_List_Integer_Length>

   <Example_List_String>строка1 строка2
строка3</Example_List_String>

   <Example_List_String_Enum>AAA BBB CCC</Example_List_String_Enum>

   <Example_Union>1 2 3 4 5</Example_Union>
   <Example_Union>123</Example_Union>

   <Example_Union2>333</Example_Union2>
   <Example_Union2>3.333</Example_Union2>

</Root_Element>
```

Файл XSD:

```
<?xml version="1.0" encoding="Windows-1251"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">

    <xsd:simpleType name="Example_List_Integer_Type">
       <xsd:list itemType='xsd:integer'/>
```

```
</xsd:simpleType>
```

Элемент xsd:list определяет список, в атрибуте itemType указывается тип элемента списка.

```
<xsd:simpleType name="Example_List_Integer_Length_Type">
   <xsd:restriction base="Example_List_Integer_Type">
      <xsd:length value='5'/>
   </xsd:restriction>
</xsd:simpleType>
```

Применяется ограничение на количество элементов в списке.

```
<xsd:simpleType name="Example_List_String_Type">
   <xsd:list itemType='xsd:string'/>
</xsd:simpleType>

<xsd:simpleType name="Example_List_String_Enum_Type">
   <xsd:restriction base="Example_List_String_Type">
      <xsd:enumeration value='AAA BBB CCC'/>
      <xsd:enumeration value='BBB'/>
      <xsd:enumeration value='CCC'/>
   </xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name="Example_Union1_Type">
   <xsd:union memberTypes="xsd:integer
Example_List_Integer_Type"/>
</xsd:simpleType>
```

Элемент xsd:union определяет объединение. В атрибуте memberTypes указывается список типов, которые могут входить в объединение. В этом примере в объединение могут входить целое число или список целых.

```
<xsd:simpleType name="Example_Union2_Type">
   <xsd:union memberTypes="xsd:integer xsd:double"/>
</xsd:simpleType>
```

Элемент может быть целым числом или действительным числом.

```
 <xsd:complexType name="Root_Element_Type">
      <xsd:sequence>
           <xsd:element name="Example_List_Integer"
type="Example_List_Integer_Type"/>
           <xsd:element name="Example_List_Integer_Length"
type="Example_List_Integer_Length_Type"/>
           <xsd:element name="Example_List_String"
type="Example_List_String_Type"/>
          <xsd:element name="Example_List_String_Enum"
type="Example_List_String_Enum_Type"/>
          <xsd:element name="Example_Union"
type="Example_Union1_Type" maxOccurs="unbounded"/>
          <xsd:element name="Example_Union2"
type="Example_Union2_Type" maxOccurs="unbounded"/>
       </xsd:sequence>
 </xsd:complexType>
```

Составной тип, в котором определяются элементы для создания списков и объединений.

```
    <xsd:element name="Root_Element" type="Root_Element_Type"/>
</xsd:schema>
```

Определение корневого элемента документа.

1.2.5 Простые элементы с атрибутами

В следующем примере создаются элемент с простым содержимым, содержащий атрибуты, и пустой элемент, содержащий атрибуты. Также показано использование типа xsd:anyType, который позволяет объявлять элементы произвольного типа с произвольным содержимым.

Пример 9.

Файл XML:

```
<?xml version="1.0" encoding="Windows-1251"?>
<!-- Примеры сложных типов -->

<Root_Element xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="compl_attrib.xsd">

   <Example_IntegerElement attr1="333"
attr2="string">333</Example_IntegerElement>

   <Example_EmptyElement attr1="333" attr2="string"/>

   <Example_anyType1 NotDefinedAttribute="NotDefinedAttribute">123
текст</Example_anyType1>
   <Example_anyType2 NotDefinedAttribute="NotDefinedAttribute">123
текст <child>123</child></Example_anyType2>

</Root_Element>
```

Файл XSD:

```
<?xml version="1.0" encoding="Windows-1251"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">

    <xsd:complexType name="Example_IntegerElement_Type">
       <xsd:simpleContent>
          <xsd:extension base="xsd:integer">
             <xsd:attribute name="attr1" type="xsd:integer"/>
             <xsd:attribute name="attr2" type="xsd:string"/>
          </xsd:extension>
       </xsd:simpleContent>
    </xsd:complexType>
```

Объявление типа элемента с двумя атрибутами, который содержит целое число. Этот тип объявляется как составной (xsd:complexType), но у него простое содержимое (xsd:simpleContent). Содержимое элемента базируется на целом числе (xsd:extension base="xsd:integer"). Элемент содержит описание атрибутов.

```
    <xsd:complexType name="Example_EmptyElement_Type">
       <xsd:attribute name="attr1" type="xsd:integer"/>
       <xsd:attribute name="attr2" type="xsd:string"/>
    </xsd:complexType>
```

Объявление типа, соответствующего пустому элементу с двумя атрибутами.

```
    <xsd:complexType name="Root_Element_Type">
          <xsd:sequence>
               <xsd:element name="Example_IntegerElement"
type="Example_IntegerElement_Type"/>
               <xsd:element name="Example_EmptyElement"
type="Example_EmptyElement_Type"/>

<xsd:element name="Example_anyType1" type="xsd:anyType"/>
<xsd:element name="Example_anyType2"/>
```

Использование типа xsd:anyType позволяет объявлять элементы произвольного типа с произвольным содержимым. Если тип не указан, то по умолчанию это также xsd:anyType.

```
          </xsd:sequence>
     </xsd:complexType>
     <xsd:element name="Root_Element" type="Root_Element_Type"/>
</xsd:schema>
```

1.2.6 Использование сложных (составных) типов

Сложные (составные) типы формируются с помощью следующих конструкций:

- xsd:sequence – определяет последовательность вложенных элементов. Соответствует символу « , » в DTD.
- xsd:choice – определяет выбор элементов. Соответствует символу « | » в DTD.

- xsd:all – определяет следование всех элементов в любом порядке. Каждый элемент может содержаться ноль или один раз. xsd:all используется редко.

Эти конструкции формируют «модель содержимого» элемента XML-документа.

Элементы xsd:sequence и xsd:choice могут быть вложены друг в друга, что соответствует комбинации « , » и « | » в DTD.

Аналогом символов «?», «*» и «+» из DTD в схемах XML являются атрибуты minOccurs и maxOccurs, которые определяют минимальное и максимальное количество вхождений элемента.

Каждый из атрибутов может содержать непосредственное число вхождений ("0", "1" и т.д.) или значение "unbounded" (неограниченное количество вхождений).

Пример 10.

Файл XML:

```
<?xml version="1.0" encoding="Windows-1251"?>
<!-- Языки разметки -->
<languages xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="compl_content_sc1.xsd">

     <language1>
          <name>HTML</name>
          <name1>HTML</name1>
     </language1>

     <language1>
          <year>01.01.1990</year>
          <howold1>14</howold1>
     </language1>
    <!-- ++++++++++++++++++++++++++++++++++++++++++++++++++ -->
     <language2>
          <name>SGML</name>
          <year>01.01.1986</year>
```

```
        <howold>18</howold>
    </language2>

    <language2>
        <name1>SGML</name1>
        <year1>01.01.1986</year1>
        <howold1>18</howold1>
    </language2>
   <!-- ++++++++++++++++++++++++++++++++++++++++++++++++++ -->
    <language3>
        <NAME>SGML</NAME>
        <year>01.01.1986</year>
        <NAME1>SGML</NAME1>
        <year1>01.01.1986</year1>
    </language3>

    <language3>
        <name>SGML</name>
        <howold>18</howold>
        <name1>SGML</name1>
        <howold1>18</howold1>
    </language3>
</languages>
```

Файл XSD:

```
<?xml version="1.0" encoding="Windows-1251"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">

    <!-- Последовательность выборов -->
    <xsd:complexType name="languageType_1">
        <xsd:sequence>

            <xsd:choice>
                <xsd:element name="name" type="xsd:string"/>
                <xsd:element name="year" type="xsd:string"/>
```

```
                    <xsd:element name="howold" type="xsd:integer"/>
               </xsd:choice>

               <xsd:choice>
                    <xsd:element name="name1" type="xsd:string"/>
                    <xsd:element name="year1" type="xsd:string"/>
                    <xsd:element name="howold1" type="xsd:integer"/>
               </xsd:choice>
          </xsd:sequence>
     </xsd:complexType>

    <!-- Выбор последовательностей -->
     <xsd:complexType name="languageType_2">
          <xsd:choice>

               <xsd:sequence>
                    <xsd:element name="name" type="xsd:string"/>
                    <xsd:element name="year" type="xsd:string"/>
                    <xsd:element name="howold" type="xsd:integer"/>
               </xsd:sequence>

               <xsd:sequence>
                    <xsd:element name="name1" type="xsd:string"/>
                    <xsd:element name="year1" type="xsd:string"/>
                    <xsd:element name="howold1" type="xsd:integer"/>
               </xsd:sequence>

          </xsd:choice>
     </xsd:complexType>

<!-- Последовательность выборов последовательностей -->
<!-- Возможно определить также выбор последовательностей выборов -->
     <xsd:complexType name="languageType_3">
          <xsd:sequence>

               <xsd:choice>
```

```
                <xsd:sequence>
                     <xsd:element name="NAME" type="xsd:string"/>
                     <xsd:element name="year" type="xsd:string"/>
                </xsd:sequence>

                <xsd:sequence>
                     <xsd:element name="name" type="xsd:string"/>
                     <xsd:element name="howold" type="xsd:integer"/>
                </xsd:sequence>
                </xsd:choice>

                <xsd:choice>
                <xsd:sequence>
                     <xsd:element name="NAME1" type="xsd:string"/>
                     <xsd:element name="year1" type="xsd:string"/>
                </xsd:sequence>

                <xsd:sequence>
                     <xsd:element name="name1" type="xsd:string"/>
                     <xsd:element name="howold1" type="xsd:integer"/>
                </xsd:sequence>
                </xsd:choice>
           </xsd:sequence>
     </xsd:complexType>

     <xsd:element name="languages">
          <xsd:complexType>
                <xsd:sequence>
                     <xsd:element name="language1"
type="languageType_1" maxOccurs="unbounded"/>
                     <xsd:element name="language2"
type="languageType_2" maxOccurs="unbounded"/>
                     <xsd:element name="language3"
type="languageType_3" maxOccurs="unbounded"/>
                </xsd:sequence>
          </xsd:complexType>
```

```
    </xsd:element>
</xsd:schema>
```

Более сложный пример, в котором используются атрибуты minOccurs и maxOccurs.

Пример 11.

Файл XML:

```
<?xml version="1.0" encoding="Windows-1251"?>
<!-- Языки разметки -->
<languages xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="compl_content_sc2.xsd">

    <language_1_1>
        <name>HTML</name>
        <year>01.01.1986</year>
        <howold>18</howold>

        <name>HTML</name>
        <year>01.01.1986</year>
        <howold>18</howold>
    </language_1_1>

    <language_1_2>
        <name>HTML</name>
        <name>HTML</name>
        <year>01.01.1986</year>
        <year>01.01.1986</year>
        <howold>10</howold>
        <howold>10</howold>
    </language_1_2>

    <language_1_3>
        <name>HTML</name>
        <name>HTML</name>
        <year>01.01.1986</year>
        <year>01.01.1986</year>
```

```
      <howold>10</howold>
      <howold>10</howold>

      <name>HTML</name>
      <name>HTML</name>
 </language_1_3>

<!-- ++++++++++++++++++++++++++++++++++++++++ -->

 <language_2_1>
      <name1>HTML</name1>
      <name>HTML</name>
      <year>01.01.1986</year>
      <year>01.01.1986</year>
      <howold>10</howold>
      <howold>10</howold>

      <name>HTML</name>
      <year>01.01.1986</year>
      <howold>10</howold>

      <year>01.01.1986</year>
      <howold>10</howold>
      <name>HTML</name>
 </language_2_1>

 <language_2_2>
      <name>HTML</name>
      <name>HTML</name>
 </language_2_2>

 <language_2_3>
      <name>HTML</name>
      <name>HTML</name>
      <year>01.01.1986</year>
      <year>01.01.1986</year>
```

```
      <howold>10</howold>
      <howold>10</howold>

      <name>HTML</name>
      <year>01.01.1986</year>
      <howold>10</howold>

      <year>01.01.1986</year>
      <howold>10</howold>
      <name>HTML</name>
 </language_2_3>

<!-- ++++++++++++++++++++++++++++++++++++++++ -->

 <language_3_1>
      <name1>HTML</name1>
      <name>HTML</name>
      <year>01.01.1986</year>
      <year>01.01.1986</year>
      <howold>10</howold>
      <howold>10</howold>

      <name>HTML</name>
      <year>01.01.1986</year>
      <howold>10</howold>

      <year>01.01.1986</year>
      <howold>10</howold>
      <name>HTML</name>
 </language_3_1>

 <language_3_2>
      <name1>HTML</name1>
      <name>HTML</name>
      <year>01.01.1986</year>
      <year>01.01.1986</year>
```

```
          <howold>10</howold>
          <howold>10</howold>

          <name>HTML</name>
          <year>01.01.1986</year>
          <howold>10</howold>

          <year>01.01.1986</year>
          <howold>10</howold>
          <name>HTML</name>
     </language_3_2>
</languages>
```

Файл XSD:

```
<?xml version="1.0" encoding="Windows-1251"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">

    <!-- Может повторяться вся последовательность -->
     <xsd:complexType name="languageType_1_1">
          <xsd:sequence minOccurs="1" maxOccurs="unbounded">
               <xsd:element name="name" type="xsd:string"/>
               <xsd:element name="year" type="xsd:string"/>
               <xsd:element name="howold" type="xsd:integer"/>
          </xsd:sequence>
     </xsd:complexType>

    <!-- Может повторяться элемент в последовательности -->
     <xsd:complexType name="languageType_1_2">
          <xsd:sequence>
               <xsd:element name="name" type="xsd:string"
minOccurs="1" maxOccurs="unbounded"/>
               <xsd:element name="year" type="xsd:string"
minOccurs="1" maxOccurs="unbounded"/>
               <xsd:element name="howold" type="xsd:integer"
minOccurs="1" maxOccurs="unbounded"/>
```

```
          </xsd:sequence>
     </xsd:complexType>

    <!-- Может повторяться вся последовательность и элемент в
последовательности -->
     <xsd:complexType name="languageType_1_3">
          <xsd:sequence minOccurs="1" maxOccurs="unbounded">
               <xsd:element name="name" type="xsd:string"
minOccurs="1" maxOccurs="unbounded"/>
               <xsd:element name="year" type="xsd:string"
minOccurs="0" maxOccurs="unbounded"/>
               <xsd:element name="howold" type="xsd:integer"
minOccurs="0" maxOccurs="unbounded"/>
          </xsd:sequence>
     </xsd:complexType>
    <!-- ++++++++++++++++++++++++++++++++++++++++++++++++++ -->
<!-- Любой элемент может повторяться произвольное количество раз -->
     <xsd:complexType name="languageType_2_1">
          <xsd:choice minOccurs="1" maxOccurs="unbounded">
               <xsd:element name="name" type="xsd:string"/>
               <xsd:element name="year" type="xsd:string"/>
               <xsd:element name="howold" type="xsd:integer"/>
          </xsd:choice>
     </xsd:complexType>

<!-- Выбор одного элемента из группы, элемент может повторяться
произвольное количество раз -->
     <xsd:complexType name="languageType_2_2">
          <xsd:choice>
               <xsd:element name="name" type="xsd:string"
minOccurs="1" maxOccurs="unbounded"/>
               <xsd:element name="year" type="xsd:string"
minOccurs="1" maxOccurs="unbounded"/>
               <xsd:element name="howold" type="xsd:integer"
minOccurs="1" maxOccurs="unbounded"/>
          </xsd:choice>
```

```
     </xsd:complexType>

<!-- Любой элемент может повторяться произвольное количество раз -->
     <xsd:complexType name="languageType_2_3">
          <xsd:choice minOccurs="1" maxOccurs="unbounded">
               <xsd:element name="name" type="xsd:string"
minOccurs="1" maxOccurs="unbounded"/>
               <xsd:element name="year" type="xsd:string"
minOccurs="1" maxOccurs="unbounded"/>
               <xsd:element name="howold" type="xsd:integer"
minOccurs="1" maxOccurs="unbounded"/>
          </xsd:choice>
     </xsd:complexType>
<!-- ++++++++++++++++++++++++++++++++++++++++++++ -->
    <!-- Избыточное описание -->
  <xsd:complexType name="languageType_3_1">
    <xsd:sequence minOccurs="1" maxOccurs="unbounded">
      <xsd:choice>
        <xsd:element name="name" type="xsd:string"/>
        <xsd:element name="year" type="xsd:string"/>
        <xsd:element name="howold" type="xsd:integer"/>
      </xsd:choice>
    </xsd:sequence>
  </xsd:complexType>

  <xsd:complexType name="languageType_3_2">
    <xsd:sequence>
      <xsd:choice minOccurs="1" maxOccurs="unbounded">
        <xsd:element name="name" type="xsd:string"/>
        <xsd:element name="year" type="xsd:string"/>
        <xsd:element name="howold" type="xsd:integer"/>
      </xsd:choice>
    </xsd:sequence>
  </xsd:complexType>

  <xsd:element name="languages">
```

```
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="language_1_1" type="languageType_1_1"
maxOccurs="unbounded"/>
        <xsd:element name="language_1_2" type="languageType_1_2"
maxOccurs="unbounded"/>
        <xsd:element name="language_1_3" type="languageType_1_3"
maxOccurs="unbounded"/>
        <xsd:element name="language_2_1" type="languageType_2_1"
maxOccurs="unbounded"/>
        <xsd:element name="language_2_2" type="languageType_2_2"
maxOccurs="unbounded"/>
        <xsd:element name="language_2_3" type="languageType_2_3"
maxOccurs="unbounded"/>
        <xsd:element name="language_3_1" type="languageType_3_1"
maxOccurs="unbounded"/>
        <xsd:element name="language_3_2" type="languageType_3_2"
maxOccurs="unbounded"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

1.2.7 Шаблоны проектирования схем

Так как в XML-схемах существуют различные варианты совместного использования составных типов и элементов, то появилось понятие «шаблон (паттерн) проектирования схем».

Существует четыре наиболее распространенных шаблона проектирования схем:

- Venetian Blind (венецианская штора).
- Russian Doll (матрешка).
- Salami Slice (ломтики салями).
- Garden of Eden (Райский Сад, идеальный шаблон).

Некоторые средства разработки (в частности NetBeans) умеют преобразовывать схемы из одного шаблона в другой.

Рассмотрим шаблоны более подробно.

В шаблоне Venetian Blind (венецианская штора) определен один глобальный элемент. Остальные вложены в глобальный с использованием именованных составных типов и групп. Составные типы и группы могут быть использованы несколько раз.

То есть в схеме определены глобальные типы xsd:complexType и xsd:simpleType с атрибутом name. Определен единственный глобальный элемент xsd:element, который ссылается на глобальные типы с помощью атрибута type.

Данный шаблон является наиболее распространенным. Все предыдущие примеры в этой главе (за исключением примера на ключи) разработаны именно по этому шаблону.

Пример 12.

Файл XSD:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">

    <!-- Определение типа для атрибута -->
    <xsd:simpleType name="howoldType">
       <xsd:restriction base="xsd:integer">
          <xsd:enumeration value='10'/>
          <xsd:enumeration value='15'/>
          <xsd:enumeration value='20'/>
       </xsd:restriction>
    </xsd:simpleType>

     <xsd:complexType name="languageType">
          <xsd:sequence>
               <xsd:element name="name" type="xsd:string"/>
               <xsd:element name="year" type="xsd:string"/>
               <xsd:element name="howold" type="howoldType"/>
          </xsd:sequence>
```

```
        <xsd:attribute name="id" use="required" type="xsd:string"/>
    </xsd:complexType>

    <xsd:element name="languages">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="language" type="languageType"
maxOccurs="unbounded"/>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
</xsd:schema>
```

Форма преобразования шаблона в NetBeans:

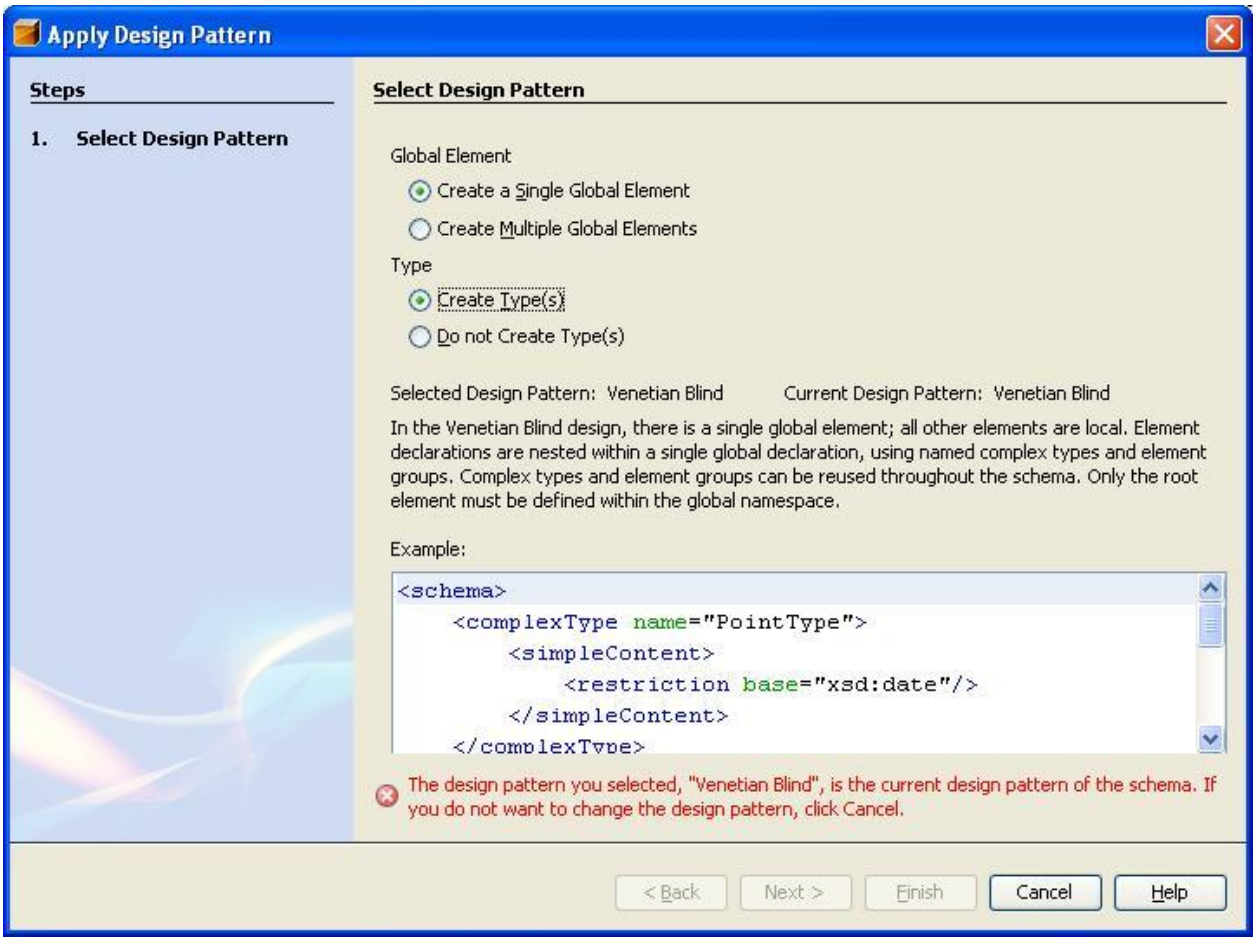


Рис. 10. Преобразование XML-схемы в шаблон «венецианская штора».

В шаблоне Russian Doll (матрешка) определен один глобальный элемент. Остальные вложены в глобальный с использованием неименованных составных типов. Составные типы могут быть использованы только один раз.

Пример 13.

Файл XSD:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<!-- Определение типа для атрибута -->
 <xsd:element name="languages">
  <xsd:complexType>
   <xsd:sequence>
    <xsd:element name="language" maxOccurs="unbounded">
     <xsd:complexType>
      <xsd:sequence>
       <xsd:element name="name" type="xsd:string"/>
       <xsd:element name="year" type="xsd:string"/>
       <xsd:element name="howold">
        <xsd:simpleType>
         <xsd:restriction base="xsd:integer">
          <xsd:enumeration value="10"/>
          <xsd:enumeration value="15"/>
          <xsd:enumeration value="20"/>
         </xsd:restriction>
        </xsd:simpleType>
       </xsd:element>
      </xsd:sequence>
      <xsd:attribute name="id" use="required" type="xsd:string"/>
     </xsd:complexType>
    </xsd:element>
   </xsd:sequence>
  </xsd:complexType>
 </xsd:element>
</xsd:schema>
```

Форма преобразования шаблона в NetBeans:

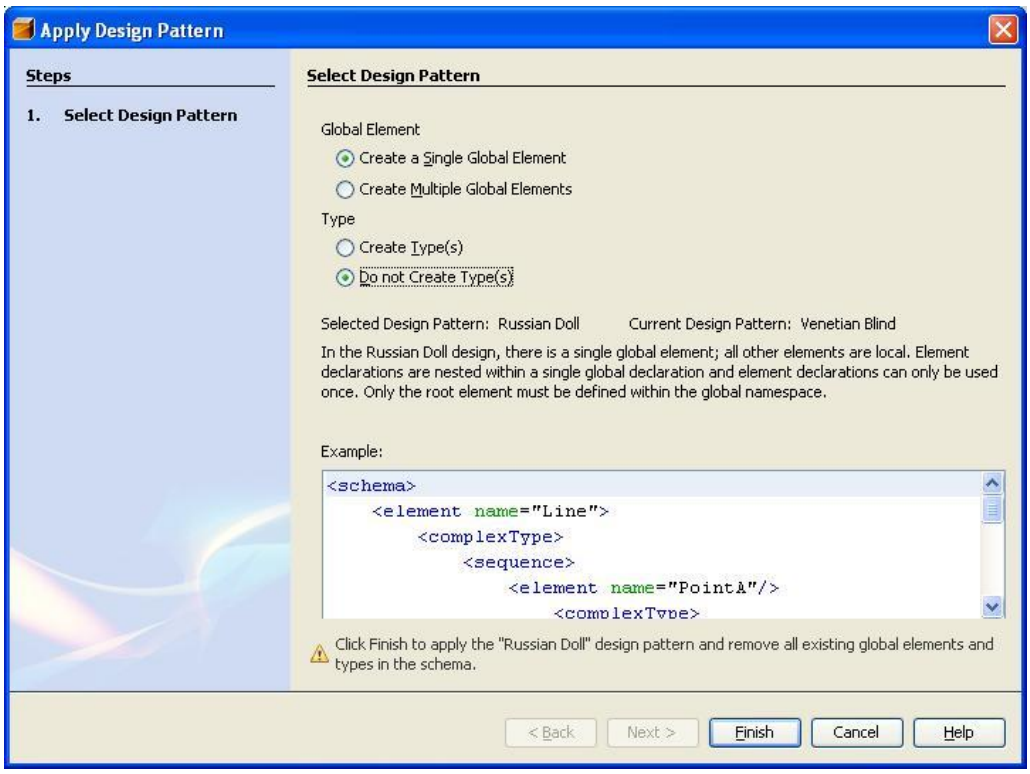


Рис. 11. Преобразование XML-схемы в шаблон «матрешка».

В шаблоне Salami Slice (ломтики салями) все элементы объявлены глобальными. Типы вложены в элементы. Используются ссылки на элементы (атрибут ref).

Пример 14.

Файл XSD:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">

    <xsd:element name="languages">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element ref="language" maxOccurs="unbounded"/>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>

    <xsd:element name="language">
        <xsd:complexType>
```

```
            <xsd:sequence>
                <xsd:element ref="name"/>
                <xsd:element ref="year"/>
                <xsd:element ref="howold"/>
            </xsd:sequence>
            <xsd:attribute name="id" use="required"
type="xsd:string"/>
        </xsd:complexType>
    </xsd:element>

    <xsd:element name="name" type="xsd:string"/>

    <xsd:element name="year" type="xsd:string"/>

    <xsd:element name="howold">
        <xsd:simpleType>
            <xsd:restriction base="xsd:integer">
                <xsd:enumeration value="10"/>
                <xsd:enumeration value="15"/>
                <xsd:enumeration value="20"/>
            </xsd:restriction>
        </xsd:simpleType>
    </xsd:element>
</xsd:schema>
```

Форма преобразования шаблона в NetBeans:

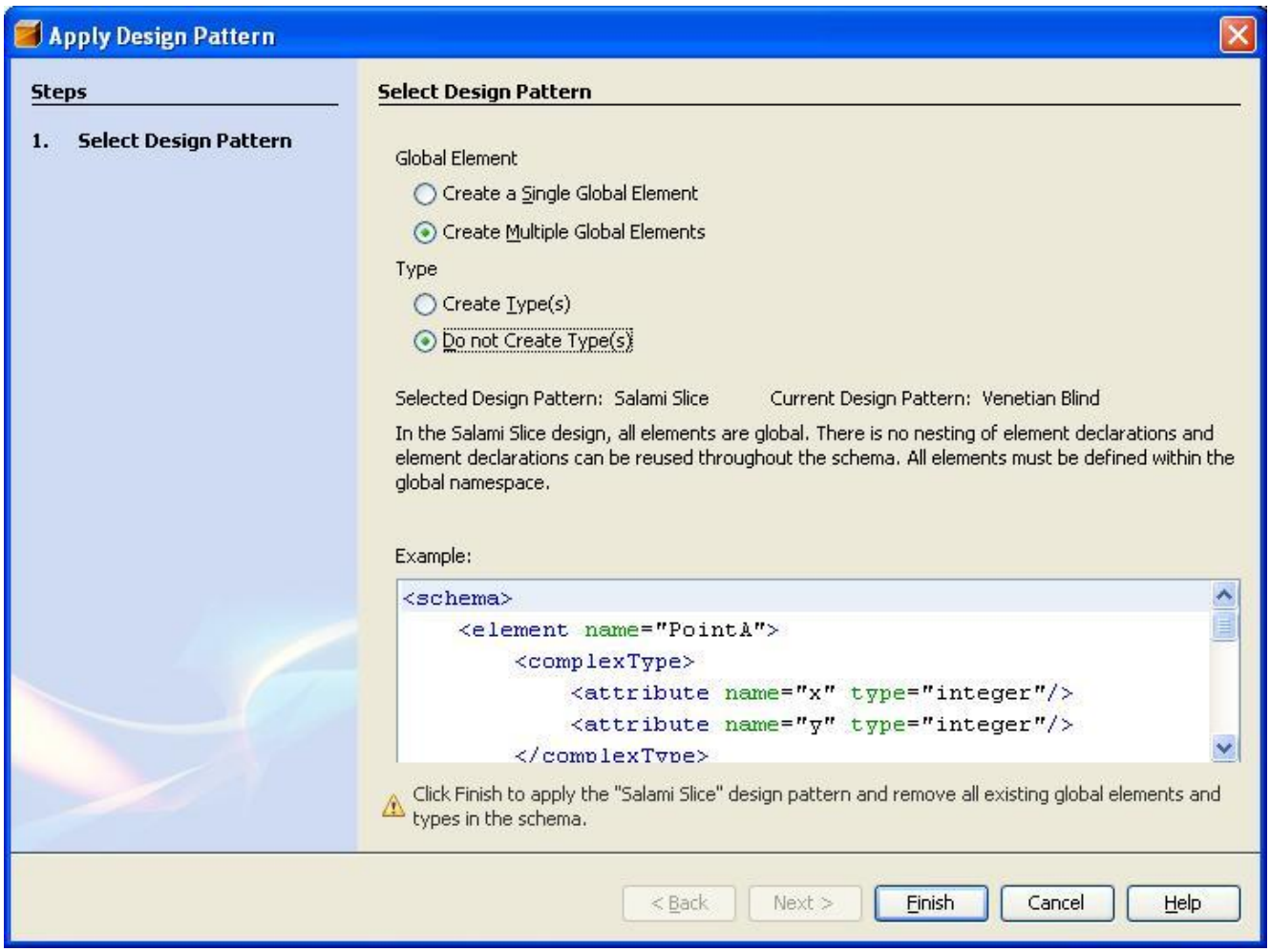


Рис. 12. Преобразование XML-схемы в шаблон «ломтики салями».

Шаблон Garden of Eden (Райский Сад, оптимальный шаблон) является комбинацией шаблонов Venetian Blind и Salami Slice. Элементы и типы объявляются глобальными, ссылки создаются по мере необходимости.

Пример 15.

Файл XSD:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">

    <xsd:element name="languages" type="languagesType"/>

    <xsd:complexType name="languagesType">
        <xsd:sequence>
            <xsd:element ref="language" maxOccurs="unbounded"/>
        </xsd:sequence>
```

```
    </xsd:complexType>

    <xsd:element name="language" type="languageType"/>

    <xsd:complexType name="languageType">
        <xsd:sequence>
            <xsd:element ref="name"/>
            <xsd:element ref="year"/>
            <xsd:element ref="howold"/>
        </xsd:sequence>
        <xsd:attribute name="id" use="required" type="xsd:string"/>
    </xsd:complexType>

    <xsd:element name="name" type="xsd:string"/>

    <xsd:element name="year" type="xsd:string"/>

    <xsd:element name="howold" type="NewHowoldTypeType"/>

    <xsd:simpleType name="NewHowoldTypeType">
        <xsd:restriction base="xsd:integer">
            <xsd:enumeration value="10"/>
            <xsd:enumeration value="15"/>
            <xsd:enumeration value="20"/>
        </xsd:restriction>
    </xsd:simpleType>

</xsd:schema>
```

Форма преобразования шаблона в NetBeans:

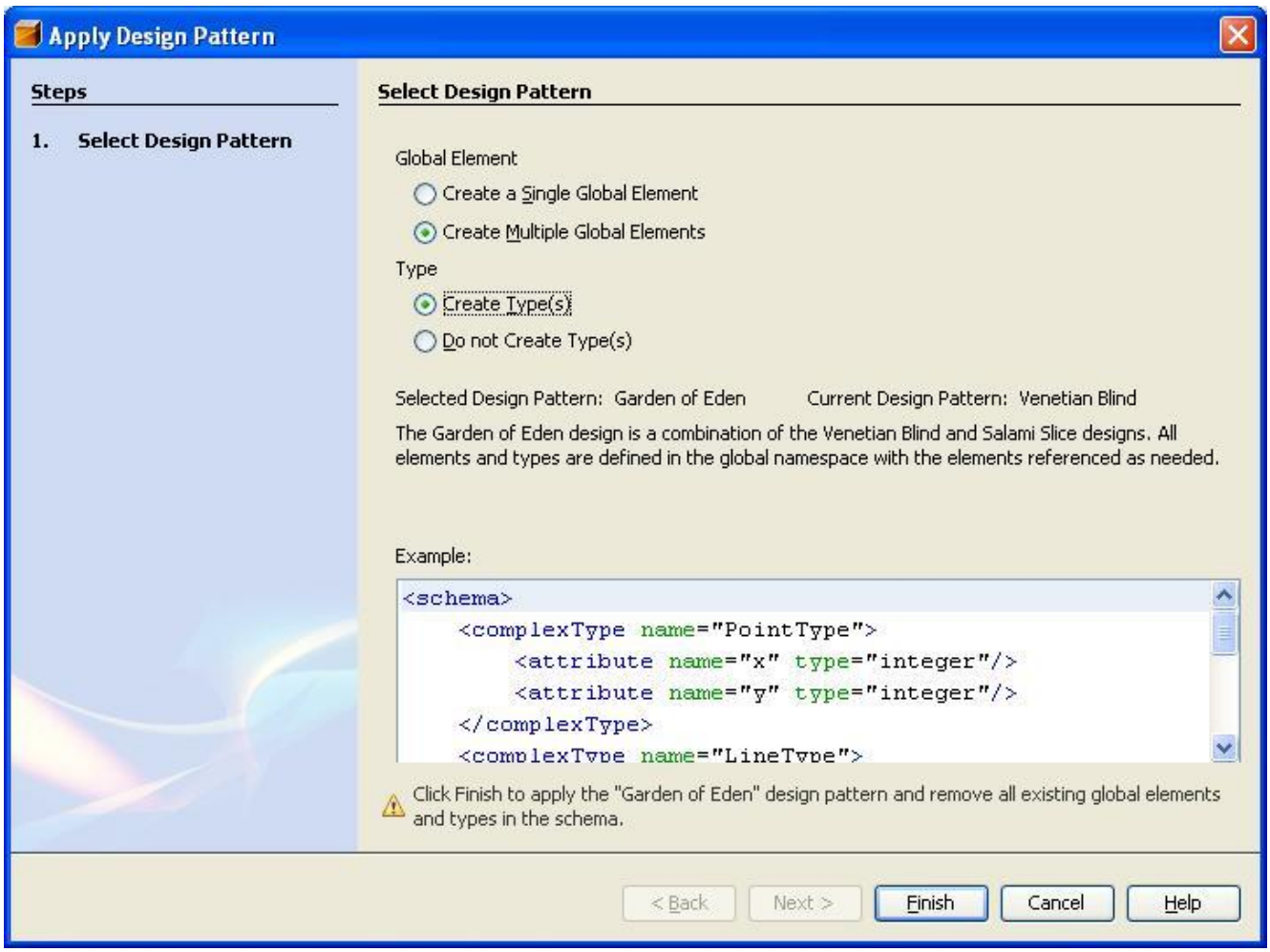


Рис. 13. Преобразование XML-схемы в оптимальный шаблон.

2 Условия лабораторных работ

2.1 Использование DTD для описания структуры документов XML

Разработайте пример описания выбранной Вами предметной области в виде документа XML. Документ должен содержать 5-7 различных типов XML-элементов.

Для разработанного документа создайте DTD-описание.

Для разработанного документа сгенерируйте DTD-описание с использованием средств XMLPad.

Сравните созданное Вами и сгенерированное DTD-описания.

2.2 Основы разработки схем XML

Для документа, разработанного в предыдущей лабораторной работе, сгенерируйте XML-схему с использованием средств XMLPad.

Внесите изменения в документ и схему для реализации следующих конструкций:

- списки;
- объединения;
- простые элементы с атрибутами;
- ограничения числовых типов данных;
- ограничения строковых типов данных.

2.3 Разработка схем XML. Использование составных типов

Основываясь на результатах предыдущей лабораторной работы, внесите изменения в документ и схему для реализации следующих конструкций:

- элемент «all»;
- элемент «sequence»;
- элемент «choice»;
- элемент «sequence», вложенный в элемент «choice»;
- элемент «choice», вложенный в элемент «sequence».

2.4 Разработка схем XML. Шаблоны проектирования схем

Модифицируйте схему, разработанную в предыдущей лабораторной работе, для реализации следующих шаблонов проектирования схем:

- венецианская штора;
- матрешка;
- ломтики салями;
- Райский Сад.

3 Требования к отчетам

Отчеты разрабатываются отдельно по каждой лабораторной работе. Отчет по каждой лабораторной работе должен включать:

- титульный лист;
- тексты XML-документов, DTD-описаний, XML-схем;
- результаты валидации.

4 Контрольные вопросы

1. Что такое DTD и для чего используется эта технология?
2. Как в DTD объявляются элементы XML-документа?
3. Как в DTD объявляется последовательность и выбор элементов?
4. Как в DTD объявляются атрибуты элементов XML-документа?
5. В чем разница между встроенными и внешними DTD?
6. В чем основное отличие в способах описания содержимого элементов в DTD и XML-схемах?
7. Как присоединить XML-схему к документу XML?
8. Как используются простые типы и ограничения (фасеты) в XML-схемах?
9. Как объявляются списки и объединения в XML-схемах?
10. Как объявляются сложные (составные) типы в XML-схемах? В чем отличие от DTD?
11. Как объявляется количество вхождений элемента в XML-схемах? В чем отличие от DTD?
12. Что такое шаблоны проектирования XML-схем? В чем их особенности?

5 Литература

1. Расширяемый язык разметки (XML) 1.0 (вторая редакция), 2000. [электронный ресурс] – Режим доступа: http://www.rol.ru/news/it/helpdesk/xml01.htm – Загл. с экрана.

2. [XML Schema Part 0 Primer, 2004] XML Schema Part 0: Primer Second Edition, 2004. [электронный ресурс] – Режим доступа: http://www.w3.org/TR/xmlschema-0/ – Загл. с экрана.
3. [XML Schema Part 1 Structures, 2004] XML Schema Part 1: Structures Second Edition, 2004. [электронный ресурс] – Режим доступа: http://www.w3.org/TR/xmlschema-1/ – Загл. с экрана.
4. [XML Schema Part 2 Datatypes, 2004] XML Schema Part 2: Datatypes Second Edition, 2004. [электронный ресурс] – Режим доступа: http://www.w3.org/TR/xmlschema-2/ – Загл. с экрана.