Математические структуры и моделирование
2013. № 2(28). С. 60–67

УДК 004.624:57.087.1

ФОРМАТ ДАННЫХ НА ОСНОВЕ XML-РАЗМЕТКИ ДЛЯ ОБМЕНА БИОМЕТРИЧЕСКИМИ ОБРАЗЦАМИ

А.В. Альтергот, О.А. Вишнякова, Д.Н. Лавров

В работе представлено описание формата XML-документа, содержащего биометрические образцы и выборки. При проектировании формата была заложена платформонезависимость, расширяемость формата. Формат используется для обмена данными между модулями разрабатываемой в настоящее время системы сбора биометрических данных.

Введение

Разбор формата данных можно проигнорировать при сборе данных, но рано или поздно встанет вопрос об интерпретации собранных данных. Командой разработчиков рассматривались два подхода: хранить данные в отдельных файлах в известных форматах или в уже разобранном виде, например, в виде строк вида [[2.0 3.0] [4.0 6.0]] [1]. Первый вариант был отвергнут из-за невозможности контролировать изменения форматов и их огромного разнообразия. Второй вариант был отвергнут из-за сильной избыточности и необходимости разбора строки, представляющей многомерный массив, а также «угрозы» реальной или мнимой потери точности при округлении.

Предложено компромиссное решение. Хранить бинарные данные в виде байтового потока (в XML в кодировке Base64, в базе данных в BLOB-полях можно хранить как набор байт) с указанием спецификатора байтового потока. В спецификаторе указывается тип ячейки и максимальное значение параметра, которое не должно превышать максимального числа ячейки.

1. XML-схема

При передаче биометрического образца нам требуется снабдить его множеством метаданных. В свою очередь метаданные могут в совокупности формировать некоторую сущность.

К примеру, сенсор, которым образец был снят, являясь частью описания образца, может быть представлен как самостоятельная сущность, а список сенсоров, используемых в конкретной системе хранения и обработки биометриче-

ООО «Люксофт»; Омский государственный университет им. Ф.М. Достоевского
E-mail: avaltergot@gmail.com, olga@infotekorg.ru, lavrov@omsu.ru

ских образцов, может быть передан самостоятельно как справочник. Структура элемента `<sensor/>` в стандарте XML Schema имеет следующий вид:

```
<xs:complexType name="Sensor">
  <xs:sequence>
    <xs:element name="name" type="String"
                minOccurs="0" maxOccurs="1"/>
    <xs:element name="description" type="String"
                minOccurs="0" maxOccurs="1"/>
  </xs:sequence>
  <xs:attribute name="id" type="Long" use="required"/>
</xs:complexType>
```

Сенсоры могут быть различных типов. В частности:

```
<xs:complexType name="Microphone">
    <xs:complexContent>
        <xs:extension base="Sensor">
            <xs:sequence minOccurs="0"/>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
```

В данном случае тип сенсора «микрофон» не имеет каких-либо дополнительных характеристик, но в будущем они могут появиться, при этом базовый тип Sensor меняться не будет.

Следовательно, элемент `<sensor/>` в XML файле имеет вид (в данном случае микрофон):

```
<sensor xsi:type="bio:Microphone" id="1">
    <name>Genius</name>
    <description>10Hz-22kHz</description>
</sensor>
```

Для элемента `<sensor/>` обязательным является лишь атрибут `@id`, которого достаточно для определения сенсора, при условии, что он известен системе. Мы предполагаем, что все известные системе сенсоры будут вноситься в неё заранее, с присвоением соответствующего `id`. Элемент `<sensor/>` может быть частью описания образца и элементом списка справочника сенсоров в случае его выгрузки в XML.

Аналогично, данные о субъекте выделяются в элемент `<subject/>`. Описание типа:

```
<xs:complexType name="Subject">
   <xs:sequence>
     <xs:element name="hash" type="String"
                 minOccurs="1" maxOccurs="1"/>
     <xs:element name="gender" type="Gender"
```

```
                minOccurs="0" maxOccurs="1"/>
     <xs:element name="birthdate" type="Date"
                minOccurs="0" maxOccurs="1"/>
     <xs:element name="document" type="String"
                minOccurs="0" maxOccurs="1"/>
     <xs:element name="comment" type="String"
                minOccurs="0" maxOccurs="1"/>
   </xs:sequence>
   <xs:attribute name="id" type="Long" use="optional"/>
   </xs:complexType>
```

Обязательным у <subject/> является наличие hash'а. Мы накладываем уникальность на hash в рамках системы, поскольку биометрические образцы хранятся и передаются обезличенными, в то время как hash вычисляется по персональным данным испытуемого. Таким образом, мы всегда знаем, что образцы, будучи сняты в разное время разными способами, принадлежат одному и тому же человеку, но неизвестно кому именно. Пример <subject/> в XML:

```
    <subject>
        <hash>87cc535ab17...ea286</hash>
        <gender>MALE</gender>
        <birthdate>2013-01-01T00:00:00.000Z</birthdate>
        <comment>Comment</comment>
     </subject>
```

Отметим, что типы String, Long, Date, Gender и другие простые типы описаны в схеме, их вид интуитивно понятен, не требует отдельных пояснений.

Как уже было сказано, передаётся и хранится биометрический образец + метаданные. Описав отдельные сущности, мы можем перейти к «центральному» элементу биометрической схемы на основе XML.

Структура его следующая:

```
<xs:complexType name="Sample">
  <xs:sequence>
     <xs:element name="timeStamp" type="Date" minOccurs="1"
                maxOccurs="1"/>
     <xs:element name="binaryData" type="BinaryData"
                minOccurs="1" maxOccurs="1"/>
     <xs:element name="format" type="String" minOccurs="1"
                maxOccurs="1"/>
     <xs:element name="hash" type="String" minOccurs="1"
                maxOccurs="1"/>
     <xs:element name="subject" type="Subject" minOccurs="1"
                maxOccurs="1"/>
     <xs:element name="sensor" type="Sensor" minOccurs="1"
                maxOccurs="1"/>
   </xs:sequence>
   <xs:attribute name="id" type="Long" use="optional"/>
</xs:complexType>
```

Элемент `<binaryData/>` типа `BinaryData` содержит снятый биометрический образец. На данном этапе развития системы под этим типом подразумевается обыкновенный массив байт в формате Base64. В будущем, он может содержать ссылку на файл в файловой системе или выполнен в виде attachment. В любом случае, снятый образец представляет собой набор байт, а информацию о том, как интерпретировать этот набор, содержит элемент `<format/>`. Атрибут `@id` не является обязательным для элемента `<sample/>`, присутствует в служебных целях. Для `hash` требование аналогичное таковому для элемента `<subject/>` (см. выше).

Аналогично сенсорам образцы имеют различные типы. В частности, описание образца типа «речевой образец»:

```
<xs:complexType name="VoiceSample">
   <xs:complexContent>
      <xs:extension base="Sample">
      <xs:sequence>
         <xs:element name="text" type="Text" minOccurs="1"
                     maxOccurs="1"/>
         <xs:element name="handMarking" type="String"
                     minOccurs="0" maxOccurs="1"/>
      </xs:sequence>
      </xs:extension>
   </xs:complexContent>
</xs:complexType>
```

Заметим, что речевой образец содержит дополнительные характерные свойства, как то: прочитанный текст (элемент `<text/>`), ручная разметка фонем (`<handMarking/>`). Другие типы образцов могут иметь индивидуальные характерные свойства.

Исходя из вышесказанного, XML-документ, содержащий речевой биометрический образец, снятый микрофоном, будет иметь следующий вид:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<bio:samples xmlns:bio="http://omsu.ru/fkn/ctn/bio/sample/schema">
  <sample xsi:type="bio:VoiceSample"
            xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <timeStamp>2013-08-30T00:44:30.143Z</timeStamp>
    <binaryData>
      <data>
         EuwVAhKgFC8TZx ...<omitted output>...
            A3ADMAGAAXAAH/0/+3/4X/Vv9e/3n/jf+d/3j/XP9K/yb/Qv9v
      </data>
    </binaryData>
    <format>84000:s:48000</format>
    <hash>1b7040708946478db8d834f ... eff61</hash>
    <subject>
       <hash>c65cdec4aa512f67fd75 ... 3701a</hash>
```

```
    <gender>MALE</gender>
    <birthdate>2013-08-30T00:00:00.000Z</birthdate>
    <comment>comment</comment>
  </subject>
  <sensor xsi:type="bio:Microphone" id="1">
  <name>Genius</name>
     <description>10Hz-22kHz</description>
  </sensor>
  <text>
    <hash>3b44a07140e76ddb26ae8 ... c7d82</hash>
         <text>абвгд</text>
     </text>
 </sample>
```

Достоинства

- Описанный подход делает возможным взаимодействие с биометрическим образцом, как с объектом в рамках привычной объектно-ориентированной модели. Одновременно формат XML является текстовым, что позволяет человеку читать и редактировать метаданные без использования специальных средств.
- Технология XML Schema позволяет валидировать XML-документ по описанным правилам.
- Технология XML Schema позволяет применять инструменты кодогенерации для объектно-ориентированных языков программирования, что ускоряет процесс разработки приложений. В комплексе с каркасами XML-парсинга подход позволяет быстро и удобно инстанциировать в памяти объекты биометрических образцов.

Недостаки

- Формат XML, в силу своих особенностей, вносит приличную долю служебных тегов в передаваемые данные.
- Используемый на данном этапе подход передачи бинарной информации — формат Base64 — является избыточным, добавляет 1/3 от объёма данных.
- В текущей редакции схема данных подразумевает дублирование некоторых элементов. Например, конструкция
  `<subject><hash>ABC<hash/><subject/>`
  будет обязательно присутствовать в каждом образце, снятого с человека с `hash=ABC`.

2. Формат спецификатора

Формат бинарных данных описывается тэгом `<format>` содержащим спецификатор формата. Его структура описывается следующим образом:

$$\underbrace{N_1 \times \ldots \times N_i \times \ldots \times N_R}_{\text{размерность}} : T_1 M_1 \ldots T_L M_L : F_s,$$

где R — размерность данных;

N_i — количество точек на соответствующей i-й оси;

L — число полей в ячейке;

$T_j M_j$ — тип ячейки (T_j) и максимальное значение (M_j), которое может быть опущено, что будет означать, что оно совпадает с максимальным, определяемым типом; $T_j = \{b, s, i, l, B, S, I, L, F, D\}$;

B — байт без знака (8 бит),

S — слово без знака (16 бит),

I — двойное слово без знака (32 бита),

L — четверное слово без знака (64 бита), в нижнем регистре те же размерности со знаком (соответствуют в точности типам java:

b — byte,

s — short,

i — int,

l — long,

F — соответствует java float (32 бита),

D — соответствует java double (64 бита).

F_s — частота дискретизации в герцах (для звука, сигналов акселерометров и графического планшета), разрешение при печати в точках на дюйм (для изображений), в кадрах в секунду для видеопотока;

: — разделитель между группами однотипных полей (:: — отсутствующее поле, значение по умолчанию в соответствии с типом);

$\times$ — разделитель внутри группы размерность ($\times\times$ — значение по умолчанию в соответствии с типом); в реальном XML-файле вместо $\times$ используется `x`).

2.1. Примеры спецификаций формата

Изображение размером 320×200 в формате RGB с разрешением 300 dpi:

`320x200:BBB:300` — сокращённая форма;

`320x200:B255B255B255:300` — полная форма.

Изображение размером 320x200 в формате CMYK с разрешением 600 dpi:

```
320x200:BBBB:600
```

Монозвуковая дорожка с частотой дискретизации 48 кГц и 32-битной оцифровкой (4 байта), 1000 отсчётов:

`1000:I:48000` — нецентрированные или

`1000:i:48000` — центрированные относительно нуля отсчёты.

Стереозвуковая дорожка с частотой дискретизации 22 кГц и 16-битной оцифровкой, 32000 отсчётов:

```
32000:II:22000
```

Данные графического планшета A6 (4,13x5,83 дюйма) с частотой дискретизации по времени 50 Гц и разрешением по горизонтали и вертикали 2540 dpi (lpi) и 512 уровнями давления, 2000 отсчётов:

`2000:FFS512:xxx:50`, если разрешение сразу переводится во float

или, если не переводится, — используем формат без сокращений (4,13*2540=10490,2 точек; 5,83*2540= 14808,2):

`2000:S10491S14809S512:50`.

Пример формат-строки видеопотока в 1000 кадров размером 320x200 в формате RGB и частотой 30 кад. / сек.

```
320x200x1000:BBI:30.
```

2.2. Порядок байт

Порядок байт для записи чисел будем использовать от старших байт к младшим. Этот порядок не совпадает с принятыми соглашениями для архитектуры процессоров Intel, но является более естественным и, кроме того, `BIG_ENDIAN` используется по умолчанию в Java. Преобразования из `BIG_ENDIAN` в `LITTLE_ENDIAN` в языке Java не представляет сложности[1].

2.3. Регулярное выражение для проверки формат-строки

Только для удобства представления регулярное выражение разбито ниже на три строки с отступами:

```
((?:[1-9][0-9]*)(?:x[1-9][0-9]*)*):
 ((?:[B,b,I,i,L,l,S,s,F,D](?:[1-9][0-9]*)?(?:\\{\\w+?\\})?)+):
  ([1-9][0-9]*(?:\\{\\w+\\})?)
```

Регулярное выражение представляет собой единую строку без каких-либо пробелов.

Регулярное выражение сформировано так, чтобы стандартные группы `group(1)`, `group(2)`, `group(3)` содержали поля, разделённые двоеточиями, которые в последующем можно будет разбирать отдельно.

Для проверки работы регулярного выражения созданы JUnit-тесты.

[1]Примеры смены порядка байт можно найти в статье `http://javainception.ru/index.php/sistemavvodavivodajava/sistemavvodavivodajava/o-poryadke-baytov.html`

2.4. Критика

Присутствует изменчивость единиц измерения Fs: то это может быть dpi, а может быть Hz или fps (кадров в секунду). Для улучшения читаемости и лучшего понимания контекста решено ввести комментарии в формат-строку `(?:\{\w+\})?)`. Таким образом, формат-строки с комментариями теперь выглядят так:

```
1000:B:100{Hz}
123:L{x}L{y}L{p}L{t}:60{Hz}
1000:L{x}L{y}L{p}L{t}:60{Hz}
320x200:B255{Red}B255{Green}B255{Blue}B255{null}:300{dpi}
```

3. Заключение

В результате длительных дискуссий в коллективе разработчиков выработан формат, который удовлетворяет большинство разработчиков проекта. Спецификация формат-строки представлена в данной статье. Разработано регулярное выражение для быстрой проверки корректности формат-строки и её последующего анализа. Созданы JUnit-тесты для модульного тестирования использования этого регулярного выражения.

В проектах, в которых используется текстовое представление собираемых данных, формат и подход, описанные в [1], могут быть удобнее в использовании и в действительности активно продолжают использоваться [3]. Тем не менее, представленный в данной работе формат более универсален. В настоящее время разрабатывается вторая версия JSON-формата, которая будет использовать идеи и подходы, описанные здесь для XML.

Описанная в данной статье схема XML-документа в настоящее время используется в разрабатываемой системе сбора биометрических данных, код которой размещается на сервере bitbucket по адресу: `https://bitbucket.org/dlavrov/biometricsmodelling`. Результаты работы докладывались на научной конференции [2].

Литература

1. Вишнякова О.А., Лавров Д.Н. Формат обмена данными в системе сбора и обработки биометрических образцов // Международная научно-практическая конференция «Информационные ресурсы в образовании». Нижневартовск : НГГУ, 17-19 апреля 2013. С. 146–148.
2. Лавров Д.Н., Альтергот А.В., Вишнякова О.А., Долгополов В.П. Спецификатор бинарных данных для XML-формата обмена биометрическими данными // Математическое и компьютерное моделирование : сборник материалов научной конференции (Омск, 18 октября 2013 г.). Омск : Изд-во Ом. гос. ун-та, 2013. С. 49–52.
3. Казанцева А.Г. Архитектура для сбора биометрических образцов походки человека // Математическое и компьютерное моделирование : сборник материалов научной конференции (Омск, 18 октября 2013 г.). Омск : Изд-во Ом. гос. ун-та, 2013. С. 58–64.