ASN.1 <-> XML TRANSLATION

Management Verteilter Systeme

Patrick Gerdsmeier, Pierre Smits

Wintersemester 2003/2004

Lehrstuhl für Systeme

Prof. Dr. Kraemer

BTU Cottbus

ASN.1 <-> XML TRANSLATION

Gliederung des Vortrags

<> Was ist ASN.1?

<> Was ist XML?

<> Konvertierungskonzepte

<> Design eines Translators

<> Weitere Beispiele

<> Fazit

<> Literatur

ASN.1 (Abstract Syntax Notation)

Was ist ASN.1?

<> ASN.1 - Abstract Syntax Notation One

<> Framework für Daten und Informationen

<> Daten sind in einem Baum strukturiert

<> Unabhängig von der Umgebung

<> Häufig in Kommunikationsprotokollen verwendet

<> ASN.1: *ISO 8824 (CCITT/ITU X.680-683 früher 208)*

<> Encoding Rules: *ISO 8825 (CCITT/ITU X.690-693 früher 209)*

ASN.1 (Abstract Syntax Notation)

ASN.1 Abstract Syntax Definition

<> Definition der Datenstrukturen

```
Record ::= SEQUENCE {
  number Number,
  name   Name }

Number ::= INTEGER
Name ::= OCTET STRING
```

XML

Was ist XML?

<> XML - eXtensible Markup Language

<> ASCII-basierte Sprache zur Beschreibung baumstrukturierter Daten

<> Daten werden in Tags eingeschlossen

<> Deklarationen (Version, Character Encoding, ...)

<> Wohlgeformt :

Dokument hat ein Wurzelelement

Jedes Tag wird geschlossen

Verschachtelung der Tags ohne Überschneidungen

XML Beispiel

"bsp.xml"

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE root SYSTEM "bsp.dtd">
<?xml-stylesheet type="text/xsl" href="bsp.xsl" ?>
...
<root>
   <child ID="1" Name="Hallo">Text</child>
   <child ID="2" Name="Welt">nochmal Text</child>
</root>
...
```

DTD

<> XML-Dokumente sind gültig, wenn sie einer DTD entsprechen

<> Angaben über die (logische) Struktur einer XML-Datei

"bsp.dtd"

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT root (child*)>
<!ELEMENT child (#PCDATA)>
<!ATTLIST child
   ID CDATA #REQUIRED
   Name CDATA #REQUIRED
>
```

XSL

<> Formatierung von XML-Daten

<> Transformation von XML-Daten in andere XML-Daten (z.B XHTML)

"beispiel.xsl"

```
...
<xsl:template match="child">
  <p><xsl:value-of select="@ID" /></p>
  <p><xsl:value-of select="@Name" /></p>
  <p><xsl:value-of select="." /></p>
</xsl:template>
...
```

Konvertierungskonzepte

<> BER - Basic Encoding Rules

<> Konvertierung zwischen binärer und ASCII-Darstellung

```
record Record:=={                    30 09
  number   12                          02 01 0C
  name     '1AA2FFFF'H }               04 04 1AA2FFFF
```

Konvertierungskonzepte

<> XER - XML Encoding Rules

<> Konvertierung zwischen BER und XML-Darstellung

```
<_SEQUENCE>
  <_INTEGER>12</_INTEGER>
  <_OCTET_STRING>1AA2FFFF</_OCTET_STRING>
</_SEQUENCE>
```

Konvertierungskonzepte

<> Konvertierung der ASN.1 Abstract Syntax Definition in eine DTD

<> Benötigt zur aussagekräftigen Benennung der XML-Tags

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT record (number,name)>
<!ELEMENT number (#PCDATA)>
<!ELEMENT name (#PCDATA)>
```

Konvertierungskonzepte

<> Mit Hilfe der DTD lassen sich die XER Daten sinnvoll benennen

<> Ergebnis ist eine verständlich lesbare XML-Datei

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE Record SYSTEM "record.dtd">
<?xml-stylesheet type="text/xsl" href="record.xsl" ?>
<record>
  <number>12</number>
  <name>1AA2FFFF</name>
</record>
```

Design eines ASN.1/XML Translators

Ziele:

<> Java Bibliothek (entwickelt für Java >=1.1)

<> Übersetzung von ASN.1 in XML und umgekehrt

<> Ausdrucksmöglichkeiten in XML und ASN.1 nicht gleichmächtig

<> Informationsverluste sollen vermieden werden

Datentypen

Grundsätzliche Regeln der Übersetzung ASN.1/XML:

<> Namensgebung der Elemente:

Name_des_Elterelements..Name_des_Elements

<> Punkt für ASN.1 Namen nicht zulässig, somit keine Konflikte

<> Weitere Informationen werden als Attribute angegeben

Datentypen

Einfache Typen:

ASN.1: `A::=INTEGER`

DTD:
```
<!ELEMENT root..A (#PCDATA)>
<!ATTLIST root.A typeRef CDATA #FIXED "A"
                 type CDATA #FIXED "INTEGER">
```

Komplexe Typen

Sequenzen (Sequence type):

ASN.1:
```
B::=SEQUENCE {
        a A DEFAULT 10,
        c C }
```

DTD:
```
<!ELEMENT root..B (B.a?, B.c)>?, B
<!ATTLIST root.B typeRef CDATA #FIXED "B"
                 type CDATA #FIXED "SEQUENCE">
<!ELEMENT B.a (#PCDATA)>
<!ATTLIST B.a typeRef CDATA #FIXED "A"
              type CDATA #FIXED "INTEGER"
              type CDATA #FIXED "10">
...
```

Komplexe Typen

Mengentypen (Set type):

ASN.1:
```
C::=SET OF A
```

DTD:
```
<!ELEMENT root..C (C._)*>
<!ATTLIST root.C typeRef CDATA #FIXED "C"
                 type CDATA #FIXED "SET OF">
<!ELEMENT C._ (#PCDATA)>
<!ATTLIST C._ typeRef CDATA #FIXED "A"
              type CDATA #FIXED "INTEGER">
```

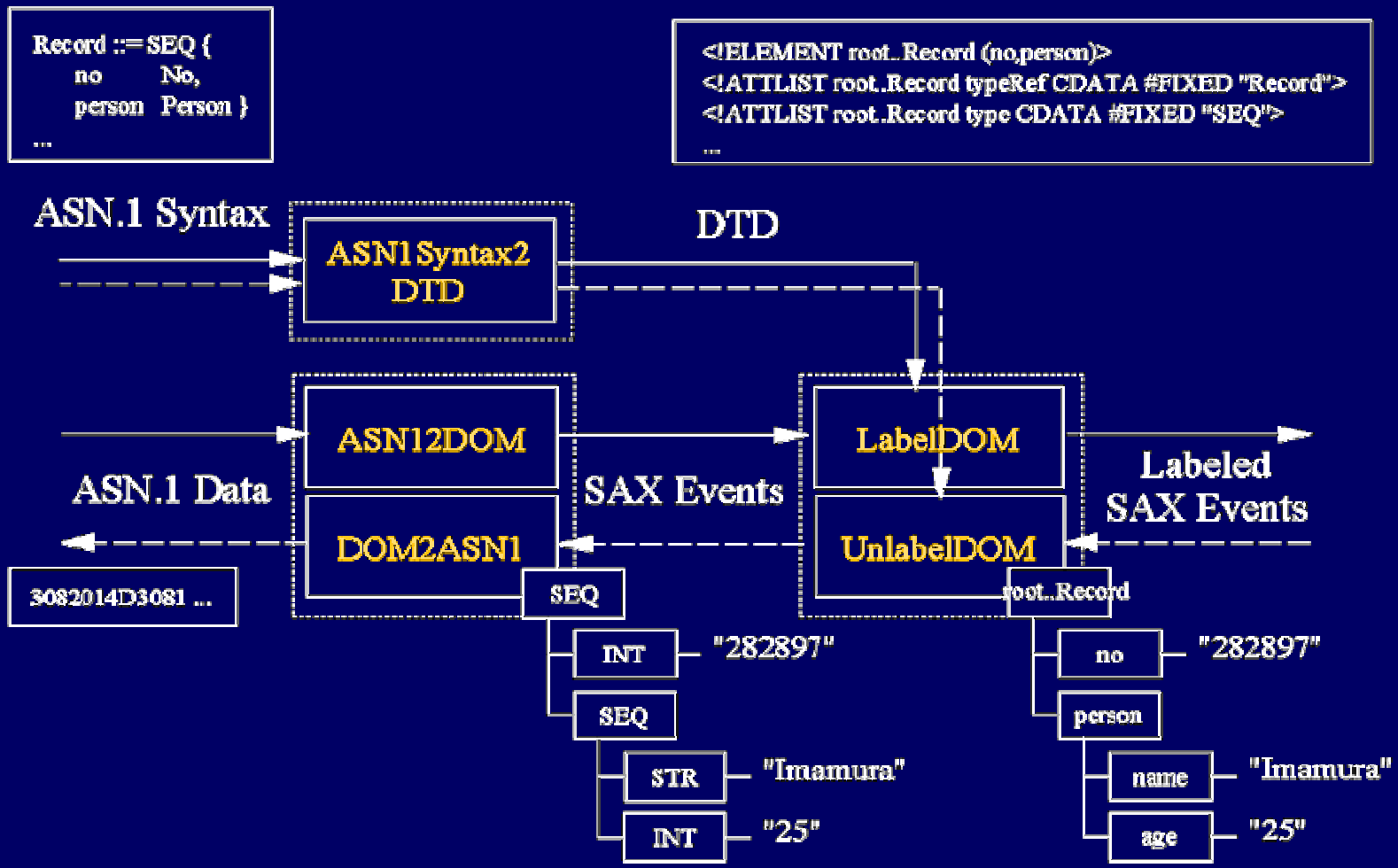
Komplexe Typen

Auswahltypen (Choice type):

ASN.1:
```
D::=CHOICE {
        a A,
        c C }
```

DTD:
```
<!ELEMENT root..D (D.a|D.c)>
<!ATTLIST root.D typeRef CDATA #FIXED "D"
                 type CDATA #FIXED "CHOICE">
<!ELEMENT D.a (#PCDATA)>
<!ATTLIST D.a typeRef CDATA #FIXED "A"
              type CDATA #FIXED "INTEGER">
```

Architektur der Implementierung



Implementierung

Fünf Module:

<> ASN1Syntax2DTD:

Übersetzung einer ASN.1 abstract syntax definition in eine DTD

<> ASN12SAX (Rückwärts: SAX2ASN1):

Übersetzung von BER in XER (SAX respektive).

<> LabelSAX (Rückwärts: UnlabelSAX):

Benennung der XML Elemente anhand einer gegebenen DTD

Module

ASN1Syntax2DTD:

<> Parsen einer ASN.1 Definition

<> Erstellen einer DTD nach Regeln:

```
A::=INTEGER
```

die rechte Seite entspricht der Regel:

```
IntegerType -> "INTEGER"
```

Ergebnis:

```
<!ELEMENT root..A (#PCDATA)>
<!ATTLIST root.A typeRef CDATA #FIXED "A"
                 type CDATA #FIXED "INTEGER">
```

Module

ASN12SAX :

<> Decodiert die ASN.1 Daten und generiert einen SAX-Stream

<> Daten werden sequentiell bearbeitet

```
StartElement: _SEQUENCE
StartElement: _SEQUENCE
StartElement: _INTEGER
Characters:   9680453654
EndElement:   _INTEGER
StartElement: _SEQUENCE
...
```

Module

LabelSAX:

<> Namen in den SAX-Streams werden durch aussagekräfigere ersetzt.

<> Erzeugtes XML-Dokument ist gültig nach der DTD

```
StartElement: root..Certificate            type="SEQUENCE"
StartElement: Certificate.tbsCertificate   type="SEQUENCE"
StartElement: TBSCertificate.serialNumber  type="INTEGER"
Characters:   9680453654
EndElement:   TBSCertificate.serialNumber
StartElement: TBSCertificate.signature     type="SEQUENCE"
...
```

Weitere Beispiele

LDAP:

<> Lightweight Directory Access Protocol

<> Protokoll zur Verwaltung von Verzeichnissen

PKCS#7:

<> Public Key Cryptography Syntax

<> Syntaxstandard von verschlüsselten Daten

<> Da die Struktur des Inhaltes einem contentType zugeordnet sind, werden zwei LabelSAX-Streams angewandt

Fazit

<> ASN.1 ist in Kommunikations-, Daten- und Sicherheitsprotokollen weit verbreitet

<> XML ist ein populäres und universelles Datenaustauschformat

<> Zur Darstellung und Bearbeitung von XML gibt es eigene Formate sowie viele Tools

<> Trotz ungleicher Ausdrucksmächtigkeit von ASN.1 und XML erfolgt die Umwandlung verlustfrei, d.h. reversibel

<> Transformation zwischen lesbarer und hardwarenaher/effizienter Darstellung

<> Weiterentwicklungsmöglichkeiten durch Benutzung von XML-Schemes

Literatur

<1> Mapping between ASN.1 and XML, T. Imamutra, H. Maruyama
http://www.trl.ibm.com/projects/xml/xss4j/docs/RT0362.pdf

<2> http://www.trl.ibm.com/projects/xml/xss4j/docs/axt-readme.html

<3> http://www.trl.ibm.com/projects/xml/xss4j/docs/axt-install.html

<4> Network Mannagement, W.-D. Haaß

<5> http://www.itu.int/rec/recommendation.asp?type=products&lang=e&parent=T-REC-X

<6> http://www.itu.int/ITU-T/studygroups/com10/languages/