The XML to TTCN-3 Mapping

George Din

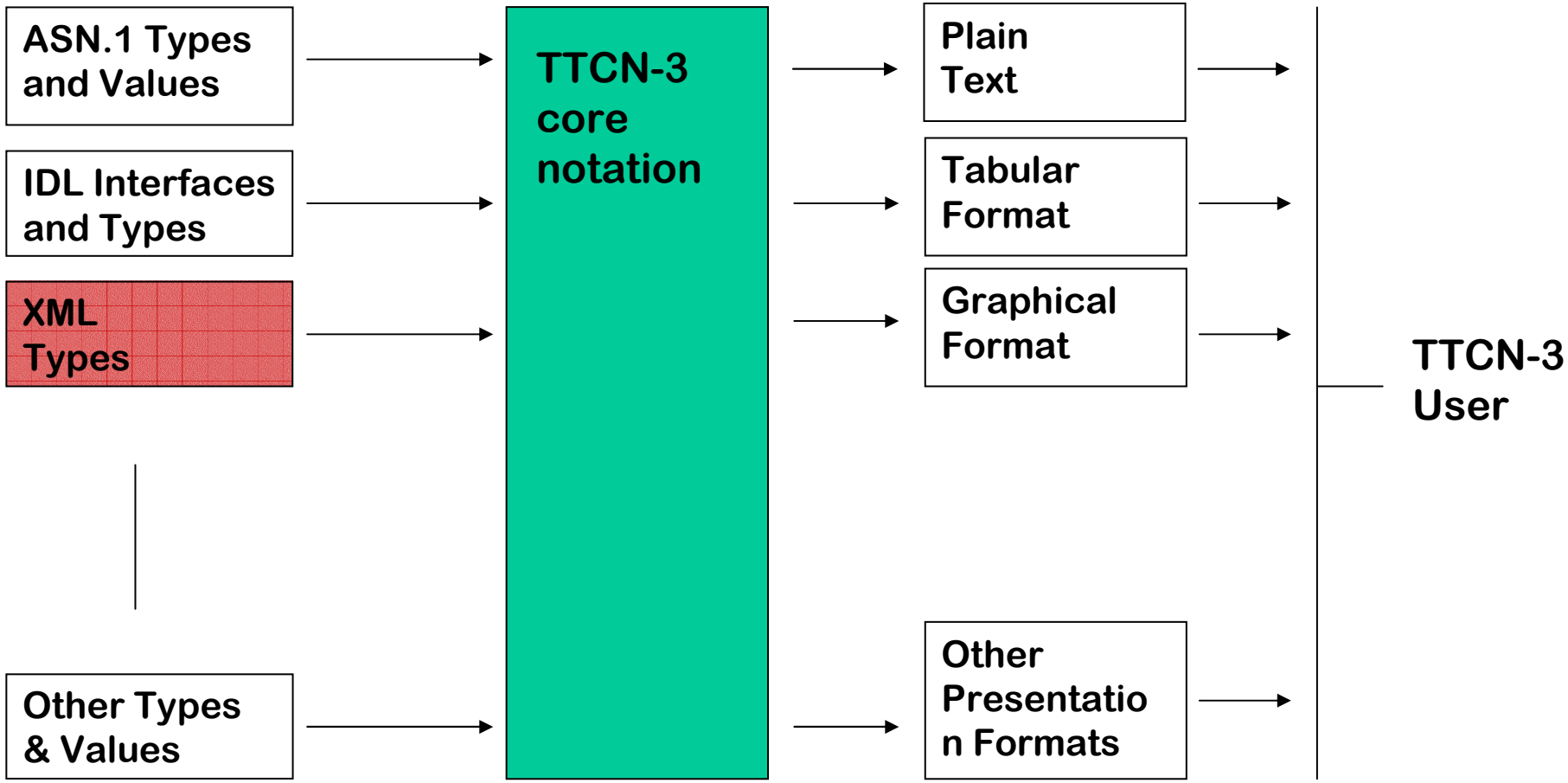
Fraunhofer Fokus

TTCN-3 User Conference, May 2004

Contents

- Motivation
- The XML revolution
- Testing XML based applications
- Mapping XML data to TTCN-3
- Conclusions

Motivation


ASN.1 Types and Values
IDL Interfaces and Types
XML Types
Other Types & Values
TTCN-3 core notation
Plain Text
Tabular Format
Graphical Format
Other Presentation Formats
TTCN-3 User

What is XML

- XML stands for EXtensible Markup Language
- Evolution:
  - GML ’69
  - SGML ’74
  - HTML ’89
  - XML ‘98
- Tag based data description language
- Tags are not predefined. Tags must be defined.
- Uses a Document Type Definition (DTD) or XML Schema (XSD) to describe data

XML Revolution

- “XML … is destined to become the "lingua franca" of the Internet”

  Steve Ballmer, Microsoft Corp ‘02
- We use XML everyday and everywhere
  - Complex activities description (Ex. Ant)
  - Text editors (e.g. OpenOffice, Docbook)
  - Web services (e.g. Amazon)
  - E-commerce (e.g. content management, data exchange)
  - Graphical design (e.g. SVG)
  - Communication protocols (e.g. SOAP)

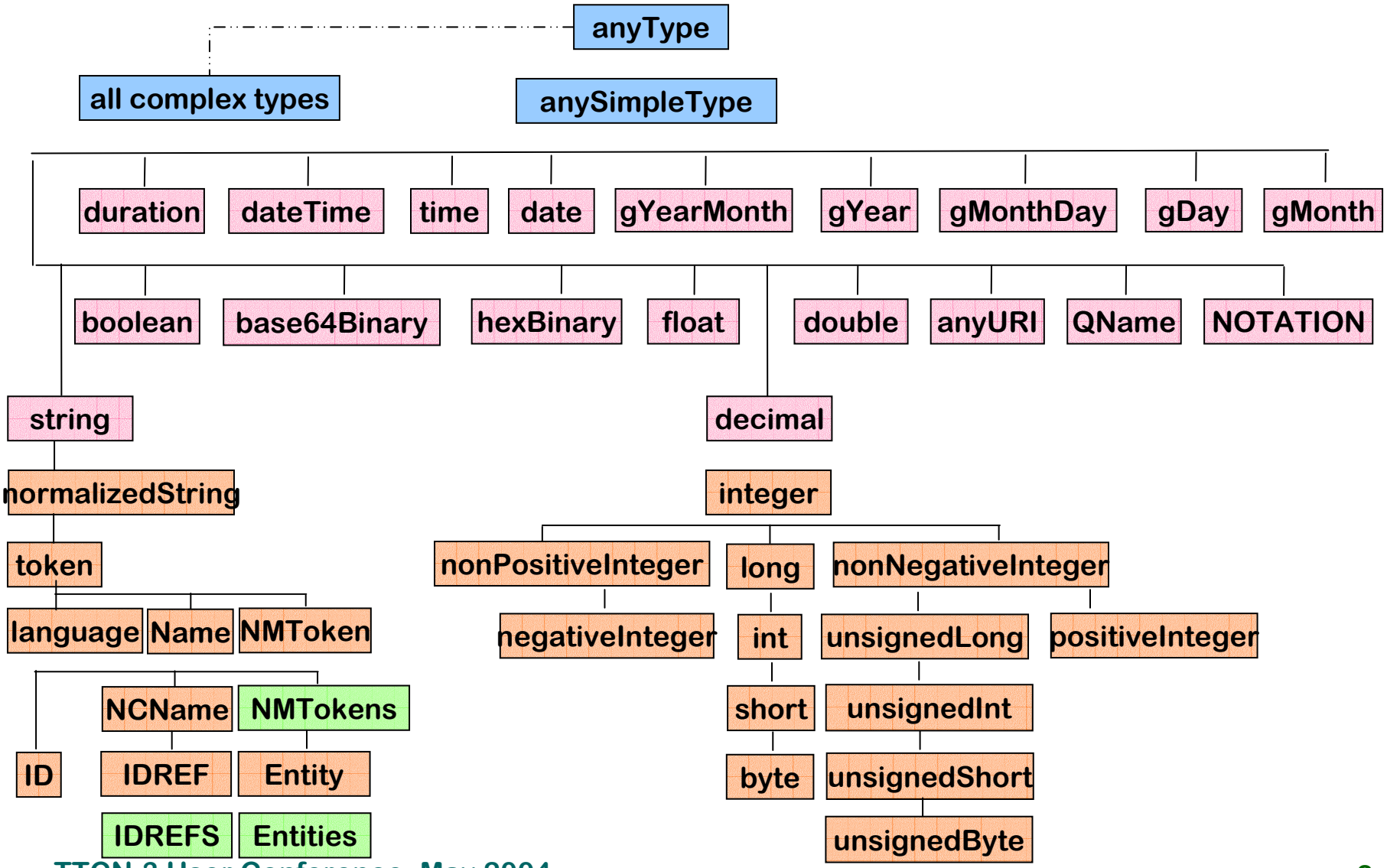
Testing XML Based Applications with TTCN-3

- Testing ?
- XML based communication applications are candidates for testing with TTCN-3
  - Use „import“ mechanism to import XML data types
  - Readable representation of XML data
  - Poweful pattern matching mechanism
- Generic adaptation possible (e.g. SOAP adapter)
- Generic Codec
  - XML to TTCN-3
  - TTCN-3 to XML

Mapping XML to TTCN-3

- Map element tags and attributes to TTCN-3 fields and types
- Datatype dichotomies
  - Atomic vs. list vs. union datatypes
  - Primitive vs. derived datatypes
  - Build-in vs. user-derived datatypes
- Different mappings
  - Embedded approach
  - Flat-Catalog approach
  - Named Type approach

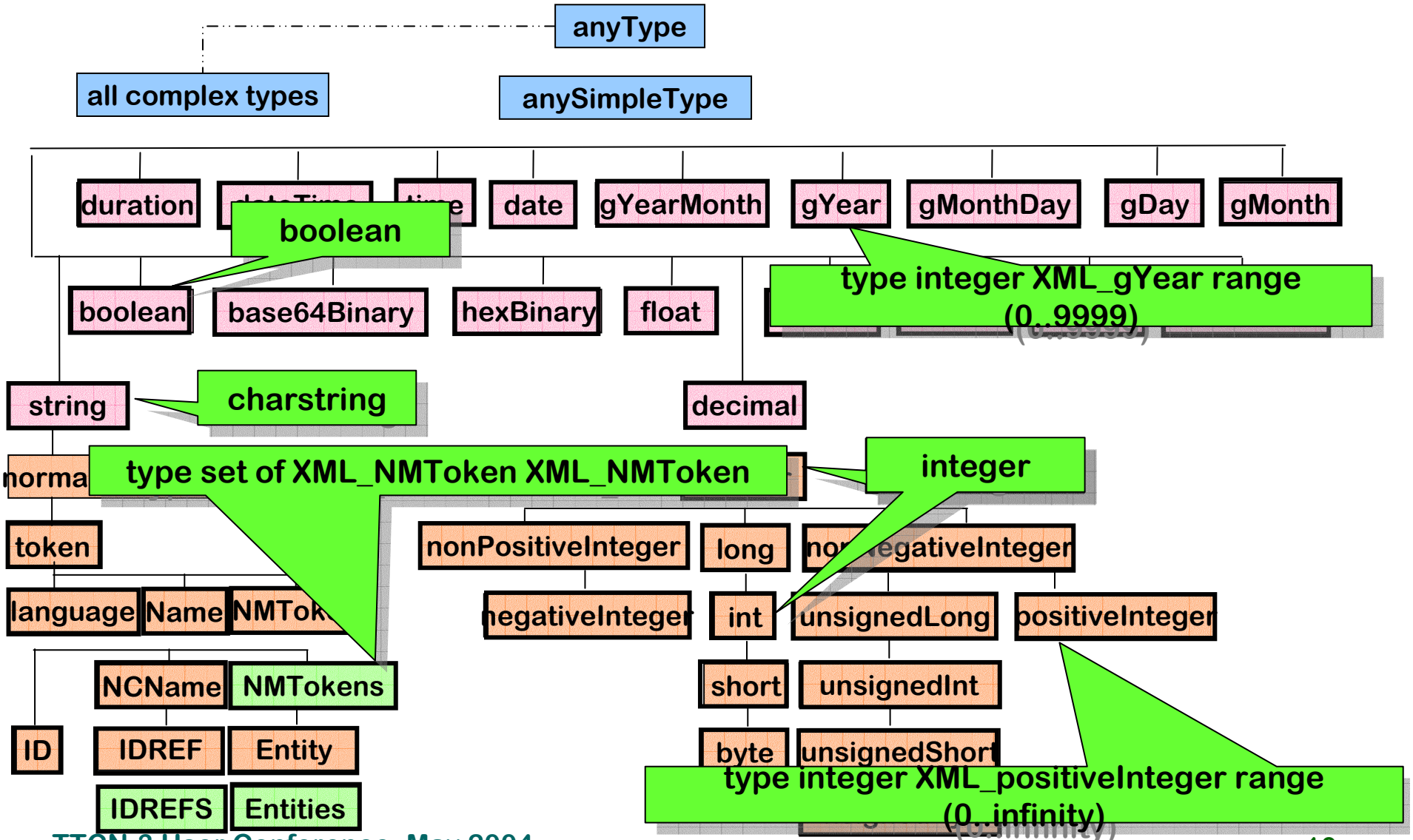
XML Schema Build-in Types



Principle Mapping

XML	TTCN-3
Primitive types	Basic types with additional attributes / useful TTCN-3 types
Derived types - elements, attributes, sequence, choice	TTCN-3 structured types - records, fields, set, union and separate types
Named type schema - define data types	One-to-one mapping to records and fields
Embedded schema - local types	Explicit types for local ones
Flat-catalog schema - type substitution	According to the above rules

XML Schema Build-in Types



Constraining facets

Facets	TTCN-3 constraints
• pattern	range, length, subtype
• enumeration	list
• whiteSpace	type attribute
• maxInclusive	range
• maxExclusive	range
• minInclusive	range
• minExclusive	range
• length	length
• minLength	length
• maxLength	length
• totalDigits	range
• fractionDigits	range

User Derived Types

- simpleType — extension of other types
- choise — union
- union — union
- list — set of type definition +
  length, range
- complexType — record

Example

```
<conference c_title="TTCN-3 User Conference">
  <presentation ID="21">
    <title>XML to TTCN-3 Mapping</title>
    <speaker>George Din</speaker>
    <day>May 5, 2004</day>
    <keywords>XML, TTCN-3, Mapping</keywords>
    <file>/net/TTconf/XML2TTCN.ppt</file>
  </presentation>
</conference>
```

Mapping Schemas in Embedded Approach

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.etsi.org/XMLSchema">
<xs:element name="conference">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="presentation">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="title" type="
              <xs:element name="speaker" type
               <xs:element name="day" type="gD
               <xs:element name="keywords" typ
            </xs:sequence>
          </xs:complexType>
          <xs:attribute name="ID" type="xs:in
        </xs:element>
      </xs:sequence>
    </xs:complexType>
    <xs:attribute ref="c_title"/>
  </xs:element>
</xs:schema>
```

```
type charstring c_titleType_00 with {extension
    „attribute", extension „generated"};

type integer IDType_00 with {extension
    „attribute", extension=„generated"};

type record presentationType
{
    charstring      title,
    charstring      speaker,
    XMLTypes.gDay   day,
    charstring      keywords,
    integer         ID
}

type record of presentationType
    presentationType_00;

type record conferenceType
{
    c_titleType                 c_title,
    presentationType_00         presentation
}
```

Mapping Schemas in Flat Catalog Approach

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.etsi.org/XMLSchema">
  <xs:element name="c_title" type="xs:string"/>
  <xs:element name="title" type="xs:string"/>
  <xs:element name="speaker" type="xs:stri
  <xs:element name="day" type="xs:gDay"/>
  <xs:element name="keywords" type="xs:str
  <xs:attribute name="ID" type="xs:integer
  <xs:element name="presentation">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="title"/>
        <xs:element ref="speaker"/>
        <xs:element ref="day"/>
        <xs:element ref="keywords"/>
      </xs:sequence>
    </xs:complexType>
    <xs:attribute ref="ID"/>
  </xs:element>
  <xs:element name="conference">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="presentation"/>
      </xs:sequence>
    </xs:complexType>
    <xs:attribute ref="c_title"/>
  </xs:element>
</xs:schema>
```

```
type charstring c_titleType with {extension
    „attribute"};

type charstring titleType;
type charstring speakerType;
type XMLType.gDay   dayType;
type charstring keywordsType;
type integer IDType with {extension „attribute"};

type record presentationType
{
    titleType       title,
    speakerType     speaker,
    dayType         day,
    keywordsType    keywords,
    IDType          ID
}

type record of presentationType
    presentationType_00 with {extension
    „generated"};

type record conferenceType
{
    c_titleType                 c_title,
    presentationType_00         presentation
}
```

Mapping Schemas in Named Type Approach

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.etsi.org/XMLSchema">
  <xs:complexType>
      <xs:sequence>
        <xs:element name="title“ type=“xs:string”/>
        <xs:element name=“speaker“ type=“xs:string”/>
        <xs:element name=“day“ type=“xs:gDay”/>
        <xs:element name=“keywords“ type=“xs:string”/>
      </xs:sequence>
  </xs:complexType>
  <xs:element name=“conference">
    <xs:complexType>
      <xs:sequence>
        <xs:element name=“presentation” ty
          <xs:attribute name=“ID" type="xs
        </xs:element>
      </xs:sequence>
    </xs:complexType>
    <xs:attribute name=“c_title“ type=“xs:
  </xs:element>
</xs:schema>
```

```
type charstring c_titleType_00 with {extension
    „attribute“, extension „generated“};

type record presentationType
{
    charstring      title,
    charstring      speaker,
    XMLTypes.gDay   day,
    charstring      keywords,
    integer         ID
}

type record of presentationType
    presentationType_00 with {extension
    „generated“};

type record conferenceType
{
    c_titleType                 c_title,
    presentationType_00         presentation
}
```

Conclusions

- A standard mapping is needed [ongoing work at ETSI]
- Similar to ASN.1 or IDL, the XML data types can be imported in TTCN-3
- Automated testing with TTCN-3 of XML based applications
- The mapping can be supplemented by the generation of the Codecs