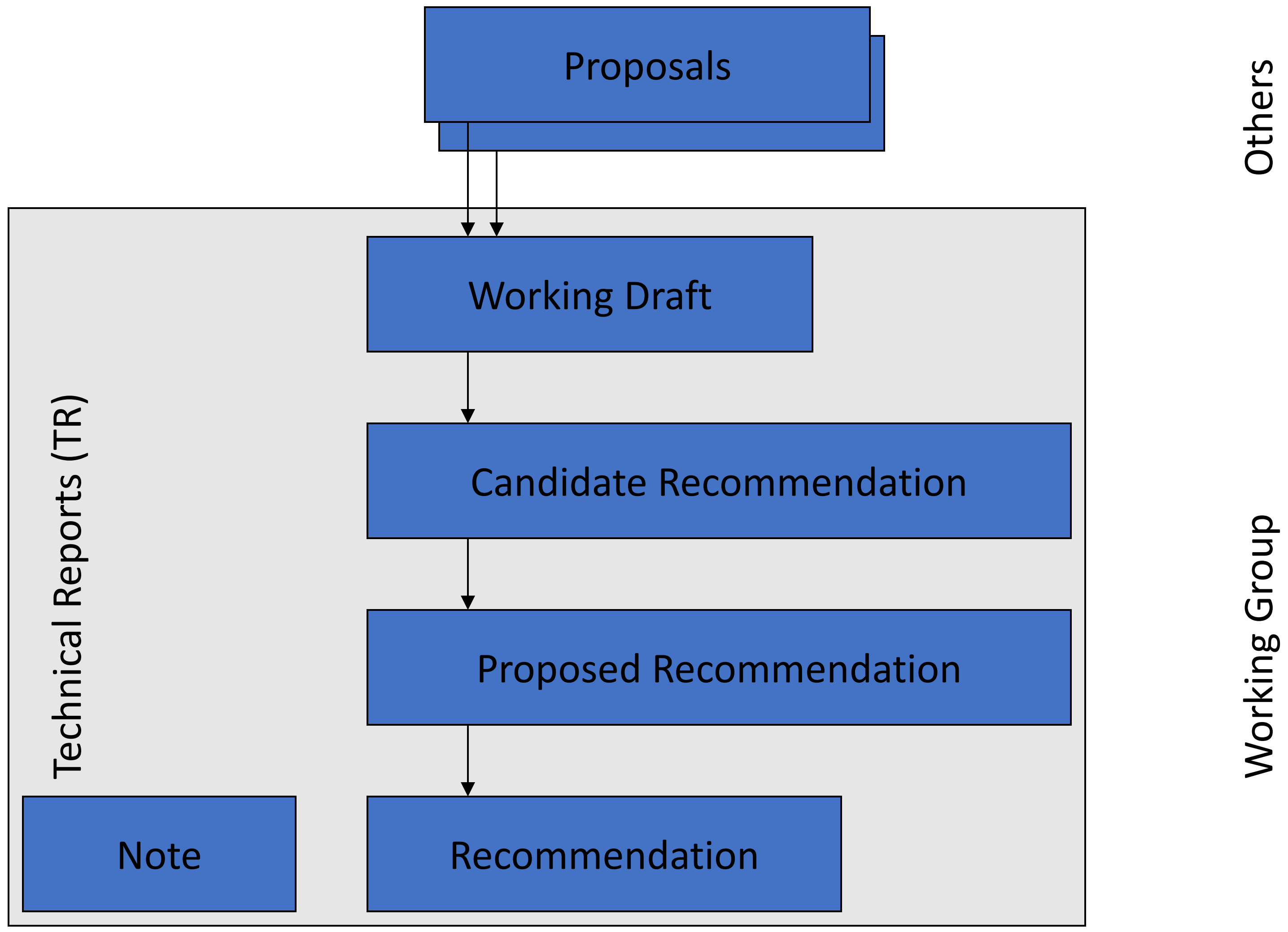
XML Schema

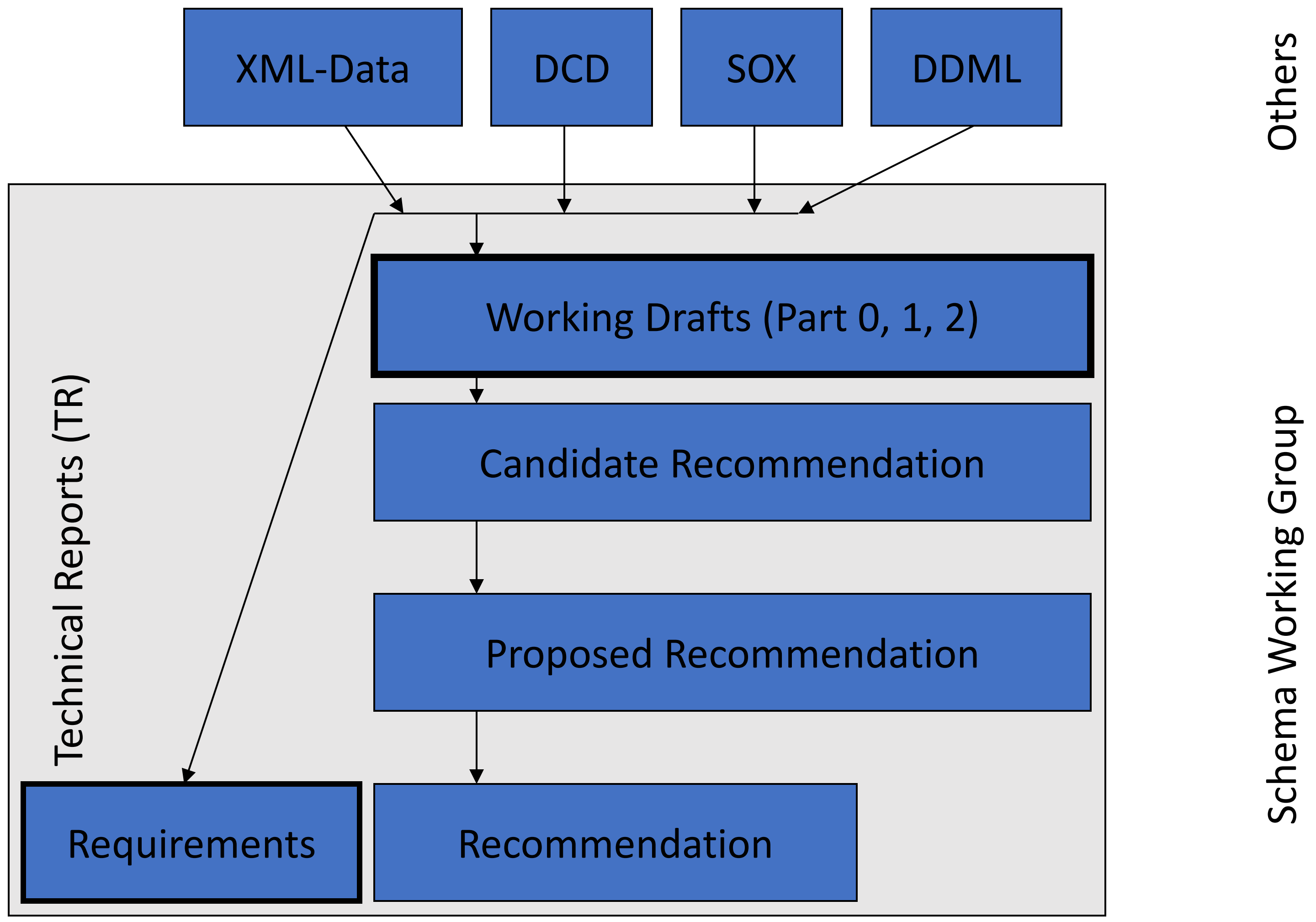
Agenda

- W3C Process
- XML Schema Requirements
- The Specifications
- Schema Tools

The W3C Process



XML Schema: Status



XML Schema Requirements

- Structural
  - namespaces
  - primitive types & structural schema integration
  - inheritance
- Data type
  - integers, dates, … (like in languages)
  - user-defined (constrain some properties)
- Conformance
  - processors, validity

Design Principles

- More expressive than DTDs
- Expressed in XML
- Self-describing
- Usable by various XML applications
- Simple enough

The Specifications

- Part 0: Primer
    - non-normative introduction
- Part 1: Structures
    - define structure
    - constraining contents
- Part 2: Datatypes
    - specify datatypes on elements and attributes

An Example Document (1/2)

```
<?xml version="1.0"?>
<purchaseOrder orderDate="1999-10-20">
 <shipTo country="US">
  <name>Matthias Hauswirth</name>
  <street>4500 Brookfield Dr.</street>
  <city>Boulder</city>
  <state>CO</state>
  <zip>80303</zip>
 </shipTo>
 <billTo country="US">
  <name>Brian Temple</name>
  <street>1234 Strasse</street>
  <city>Boulder</city>
  <state>CO</state>
  <zip>80302</zip>
 </billTo>
 ...
```

An Example Document (2/2)

```
  <comment>Brian pays</comment>
  <items>
   <item partNum="123-AB">
    <productName>Porsche</productName>
    <quantity>1</quantity>
    <price>129400.00</price>
    <comment>Need a new one</comment>
   </item>
   <item>
    <productName>Ferrari</productName>
    <quantity>2</quantity>
    <price>189000.25</price>
    <shipDate>1999-05-21</shipDate>
   </item>
  </items>
</purchaseOrder>
```

An Example Schema (1/3)

```
<xsd:schema xmlns:xsd=“http://www.w3.org/1999/XMLSchema”>

 <xsd:element name=“purchaseOrder” type=“purchaseOrderType”/>

 <xsd:element name=“comment” type=“xsd:string”/>

 <xsd:complexType name=“PurchaseOrderType”>
      <xsd:element name=“shipTo” type=“AddressType”/>
      <xsd:element name=“billTo” type=“AddressType”/>
      <xsd:element ref=“comment” minOccurs=“0”/>
      <xsd:element name=“items” type=“ItemsType”/>
      <xsd:attribute name=“orderDate” type=“xsd:date”/>
 </xsd:complexType>

 ...
```

An Example Schema (2/3)

```
<xsd:complexType name=“AddressType”>
     <xsd:element name=“name” type=“xsd:string/>
     <xsd:element name=“street” type=“xsd:string”/>
     <xsd:element name=“city” type=“xsd:string”/>
     <xsd:element name=“state” type=“xsd:string”/>
     <xsd:element name=“zip” type=“xsd:decimal”/>
     <xsd:attribute name=“country” type=“xsd:NMTOKEN”
           use=“fixed” value=“US”/>
</xsd:complexType>

<xsd:simpleType name=“SkuType” base=“xsd:string”>
     <xsd:pattern value=“\d{3}-[A-Z]{2}”/>
</xsd:simpleType>

...
```

An Example Schema (3/3)

```
  <xsd:complexType name=“ItemsType”>
   <xsd:element name=“item” minOccurs=“0” maxOccurs=“unbounded”>
    <xsd:complexType>
     <xsd:element name=“productName” type=“xsd:string/>
     <xsd:element name=“quantity”>
      <xsd:simpleType base=“xsd:positiveInteger”>
       <xsd:maxExclusive Value=“100”/>
      </xsd:simpleType>
     </xsd:element>
     <xsd:element name=“price” type=“xsd:decimal”/>
     <xsd:element ref=“comment” minOccurs=“0”/>
     <xsd:element name=“shipDate” type=“xsd:date” minOccurs=“0”/>
     <xsd:attribute name=“partNum” type=“SkuType”/>
    </xsd:complexType>
   </xsd:element>
  </xsd:complexType>
 </xsd:schema>
```

Part 1: Structures

- Type Definitions ***`<simpleType>`*** **`<complexType>`**
  `<element> <group> <all> <choice> <sequence>`
  `<attribute> <attributeGroup>`
- Attribute Declarations **`<attribute>`**
  `<simpleType>`
- Element Declarations **`<element>`**
  `<simpleType>` `<complexType>`
- Attribute Group Definitions **`<attributeGroup>`**
  `<attribute> <attributeGroup>`
- Model Group Definitions **`<group>`**
  `<element> <group> <all> <choice> <sequence>`
- Notation Declarations **`<notation>`**
- Annotations **`<annotation>`**
  `<appinfo> <documentation>`

DTD vs. Schema Structure

- DTD

```
<!ELEMENT e1
  ((e2,e3?)+|e4)>
```

- Schema

```
<element name="e1">
 <complexType>
  <choice>
   <sequence maxOccurs="unbounded">
    <element ref="e2"/>
    <element ref="e3" minOccurs="0"/>
   </sequence>
   <element ref="e4">
  </choice>
 </complexType>
</element>
```

Referential/Uniqueness Integrity

- *Define Constraints using XPath expressions*

- `<unique>`
- `<key>`
- `<keyref>`
- `<selector>`
- `<field>`

Part 2: Datatypes `<simpleType>`

- Value Space
  - defined axiomatically (primitive types)
  - enumerated outright
  - defined by restricting value space of other type
  - combination of values of other type (list)
  - *has certain properties (e.g. cardinality, equality, ordered)*
- Lexical Space
  - set of literals for a type (e.g. 100 and 1.0E2 denote same value)
- Facets
  - fundamental facets (define the type)
  - constraining facets (allow to constrain the value space)

Fundamental Facets

- *Fundamental facets can’t be changed*
- Equal
  - all types provide an equality relation
- Order
  - some types provide an ordering relation
- Bounds
  - upper bound and lower bound
- Cardinality
  - finite, infinite
- Numeric
  - yes or no

Constraining Facets

- length
- minLength
- maxLength
- pattern
- enumeration
- maxInclusive / maxExclusive
- minInclusive / minExclusive
- precision
- scale
- encoding
- duration
- period

Primitive vs. Derived Types

- Primitive Types
  - string
  - boolean
  - float
  - double
  - decimal
  - timeDuration
  - recurringDuration
  - binary
  - uriReference
  - ID
  - IDREF
  - ENTITY
  - NOTATION
  - QName

‣ *exist ab initio*

- Derived Type
  - by restriction
    - use constraining facets

```
<simpleType name="sku"
 base="xsd:string>
 <pattern
   value="\d{3}-[A-D]{4}"/>
</simpleType>
```

  - by list
    - next slide

Built-in vs. User-Derived Types

- Built-in types
  - primitive
  - derived
    - language
    - IDREFS
    - long
    - int
    - short
    - positiveInteger
    - time
    - month
    - recurringDay
    - …

- User-derived types
  - derived-only

Atomic vs. List Types

- Atomic
  - values indivisible

```
<simpleType name=“ShoeSize”
  base=“xsd:decimal”/>
```

```
<element name=“shoe”
  type=“ShoeSize”/>
```

```
<shoe>10.5</shoe>
```

- List
  - sequence of values of atomic type

```
<simpleType name=“ShoeSizes”
  base=“shoeSize”
  derivedBy=“list”/>
```

```
<element name=“shoes”
  type=“ShoeSizes”/>
```

```
<shoes>8 10 10.5</shoes>
```

Tools

- XML Schema-aware Parser
    - Xerces-J
    - Oracle XML Schema Processor
- XML Schema Validator (XSV, online)
- DTD to Schema Conversion Tools
- XML Schema Editor
    - Extensibility’s XML Authority
- XML Schema-aware Instance Editor
    - Extensibility’s XMLInstance
    - ChannelPoint’s Merlot (maybe in future)