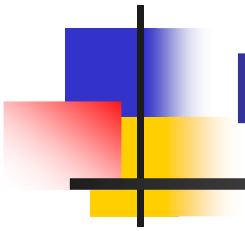
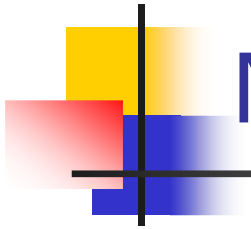


Namespaces and XML Schema

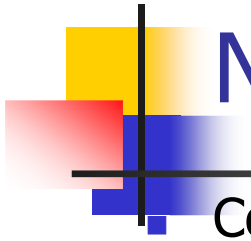
Jaana Holvikivi



Namespaces

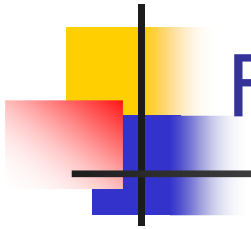
- XML distinguishes between elements and attributes from different markup languages with namespaces
- A namespace is a purely abstract entity: it's nothing more than a group of names that belong with each other conceptually
- Most markup languages have a namespace URI: the internet domain names are unambiguous (URI Uniform Resource Identifier)
- Problem: the URLs can contain invalid characters
  Solution: associate a well-formed prefix with the URI

```
<html:h2 xmlns:html="http://www.w3.org/1999/xhtml">
```



Namespaces: uses

- Combined documents
  - document contains a mixture of elements from more than one mark-up language
  - e.g. XSL and HTML in the same document
- A DTD/ schema owns a namespace where
  - all element names are unique
  - all attribute names for a certain element are unique
  - thus all references to the element and its attributes are unique
- document could contain information which is declared in several namespaces



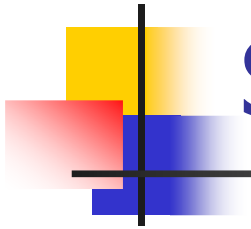
Finding the namespace

- Most standards are on the Web
  - http://www.w3.org/TR/REC-html40
- the namespace recommendation uses URLs as unique identifiers
- the application does not need internet connection, the URL is only a unique character string
- in the document, the prefix is used
  - <X:html xmlns:X="http://www.w3.org/TR/REC-html40">
  - <xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">

Namespaces, defaults

- Defining a default namespace:
- ```
  <person xmlns="http://frogstar.mil/pers"
        xmlns:xhtml="http://www.w3.org/1999/xhtml">
   <name>Mr. President</name>
   <xhtml:p>Here something in XHTML</xhtml:p>
  </person>
  ```
- the namespace is defined without a prefix:

  ```
  <book xmlns="...">
      <para>A normal paragraph</para>
  </book>
  ```
- the default namespace can be changed in any element and any of its descendants
  ```



Schema languages

XML language:

- a set of XML documents with some semantics

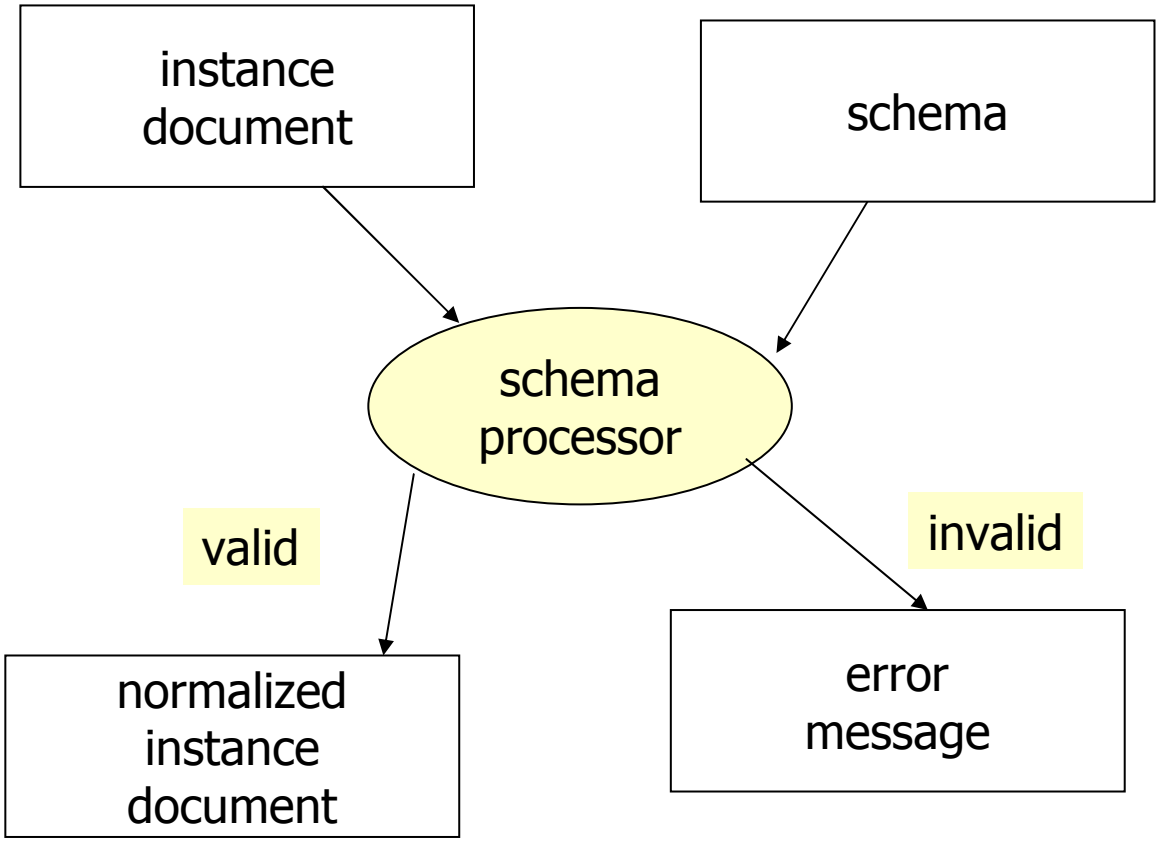
schema:

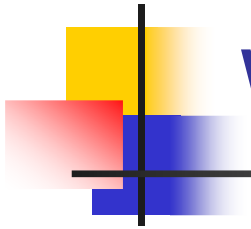
- a formal definition of the syntax of an XML language
- a schema is any type of model document that defines the structure of a database or document

schema language:

- a notation for writing schemas

Validation


instance document
schema
schema processor
valid
invalid
normalized instance document
error message



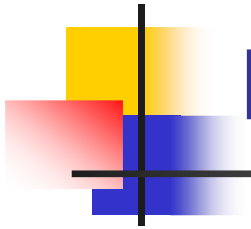
Why use Schemas?

- Formal but human-readable descriptions
- Data validation can be performed with existing schema processors
- General Requirements
  - Expressiveness
  - Efficiency
  - Comprehensibility

XML Schema

- A schema is any type of model document that defines the structure of a database or document

- <?**xml version**="1.0"?>
  <**xsd**:schema
  xmlns:**xsd**="http://www.w3.org/2001/XMLSchema">
  <**xsd**:**element name**="letter">
  </**xsd**:**element**>
  </**xsd**:schema>
- prefix xsd or xs



Requirements for XML Schema

W3C's proposal for replacing DTD

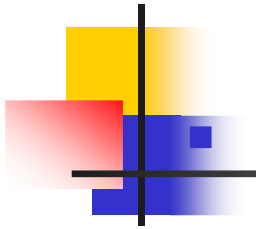
Design principles:

- More expressive than DTD
- Use XML notation
- Self-describing
- Simplicity

Technical requirements:

- Namespace support
- User-defined datatypes
- Inheritance (OO-like)
- Evolution
- Embedded documentation

XML Schema



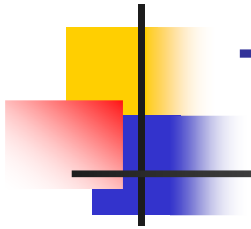
- XML Schema 1.0 recommendation 2001:
  - Part 1: Structures
  - Part 2: Datatypes
- Features in the XML Schema Recommendation include:
  - element types,
  - a simple pattern matching grammar,
  - defined ordering of sub-elements so that document structure can be tightly controlled,
  - selection between different elements so that documents can share a Schema without having identical structure.

Schema namespaces

```
<?xml version="1.0"?>
<schema xmlns = "http://www.w3.org/2001/XMLSchema"
    xmlns:pers ="http://frogstar.mil/pers"
    targetnamespace = "http://frogstar.mil/pers"
    elementFormDefault="qualified">
```

In the document instance:

```
<?xml version="1.0"?>
<person xmlns ="http://frogstar.mil/pers"
    xmlns:xsi = "http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation ="http://frogstar.mil/pers people2.xsd"
    version = "1.0">
```



Types and Declarations

Simple type definition:

- defines a family of Unicode text strings

Complex type definition:

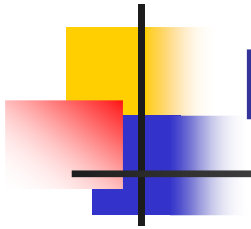
- defines a content and attribute model

Element declaration:

- associates an element name with a simple or complex type

Attribute declaration:

- associates an attribute name with a simple type



Example (1/3) document

Instance document:

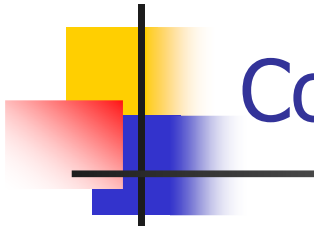
```
<?xml version="1.0"?>
<b:card xmlns:b="http://businesscard.org">
<b:name>John Doe</b:name>
<b:title>CEO, Widget Inc.</b:title>
<b:email>john.doe@widget.com</b:email>
<b:phone>(202) 555-1414</b:phone>
<b:logo b:uri="widget.gif"/>
</b:card>
```

Example (2/3) schema

```
<schema xmlns="http://www.w3.org/2001/XMLSchema"
xmlns:b="http://businesscard.org"
targetNamespace="http://businesscard.org">
<element name="card" type="b:card_type"/>
<element name="name" type="string"/>
<element name="title" type="string"/>
<element name="email" type="string"/>
<element name="phone" type="string"/>
<element name="logo" type="b:logo_type"/>
<attribute name="uri" type="anyURI"/>
```

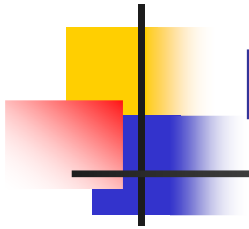
Example (3/3) schema

```
<complexType name="card_type">
  <sequence>
      <element ref="b:name"/>
      <element ref="b:title"/>
      <element ref="b:email"/>
      <element ref="b:phone" minOccurs="0"/>
      <element ref="b:logo" minOccurs="0"/>
  </sequence>
</complexType>
<complexType name="logo_type">
      <attribute ref="b:uri" use="required"/>
</complexType>
</schema>
```



Connecting schemas and instances

```
<?xml version="1.0"?>
<b:card xmlns:b="http://businesscard.org“
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://businesscard.org business_card.xsd">
      <b:name>John Doe</b:name>
      <b:title>CEO, Widget Inc.</b:title>
      <b:email>john.doe@widget.com</b:email>
      <b:phone>(202) 555-1414</b:phone>
      <b:logo b:uri="widget.gif"/>
</b:card>
```



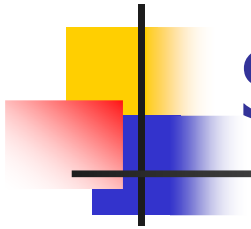
Element and Attribute Declarations

Examples:

- <element name="serialnumber"
  type="nonNegativeInteger"/>
- <attribute name=”alcohol"
  type=”r:percentage"/>

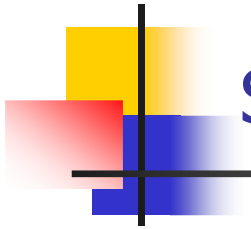
Schema: adding attributes

```
<xsd:complexType name="name">
    <xsd:sequence>
    <xsd:element name="title" type="xsd:string" maxOccurs="1"
    default="Miss"/>
    <xsd:element name="firstname" type="xsd:string" minOccurs
    ="2"/>
    <xsd:element name="surname" type="xsd:string" />
    </xsd:sequence>
    <xsd:attribute name="gender" type="xsd:string"
    default="female"/>
</xsd:complexType>
```



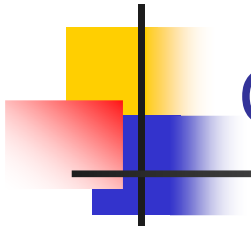
Simple Types (Datatypes) – Primitive

string	*any Unicode string*
boolean	true, false, 1, 0
decimal	3.1415
float	6.02214199E23
double	42E970
dateTime	2007-09-26T16:29:00-05:00
time	16:29:00-05:00
date	2009-09-26
hexBinary	48676c6c6f0a
base64Binary	SGVsbG8K
anyURI	http://www.boxes.se/dir/
QName	pur:order, order



Simple types (datatypes)

Primitive	Built-in
xs:string	normalizedString
xs:decimal	integer
xs:boolean	language
xs:date	negativeInteger
xs:time	nonNegativeInteger
float	positiveInteger
double	long
duration	int
hexBinary	short
anyURI	byte
QName	



Constraining facets for simple types

- minExclusive
- minInclusive
- maxExclusive
- maxInclusivetotalDigits
- fractionDigits
- length
- minLength
- maxLength
- enumeration
- whiteSpace
- pattern

Schema: element types

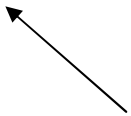
```
<xsd:element name="serialnumber" type="nonNegativeInteger"/>


<simpleType name="score_from_0_to_100">
    <restriction base="integer">
      <MinInclusive value="0" />
      <MaxInclusive value="100" />
    </restriction>
<simpleType>
```

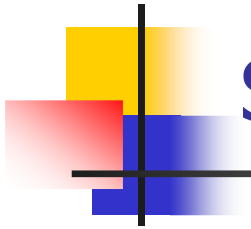
Simple type examples

```
<xsd:simpleType name="userType">
      <xsd:restriction base="xsd:string">
         <xsd:pattern value="U\w{2,6}\d{2}" />
      </xsd:restriction>
</xsd:simpleType>

<simpleType name="percentage">
  <restriction base="string">
  <pattern value="([0-9]|[1-9][0-9]|100)%"/>
  </restriction>
</simpleType>
```



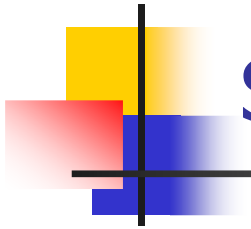
regular expression



Schema: element types

<xsd:complexType>
includes children
<sequence> or <choice> or <all>

```
<complexType name="card_type">
     <sequence>
      <element name="firstname" type="string"/>
      <attribute name="title" type="string"/ >
      <choice>
        <element name="email"/>
        <element name="phone"/>
      </choice>
      </sequence>
</complexType>
```



Schema: element types

Global elements are child elements for the schema element

Local elements are children for another element

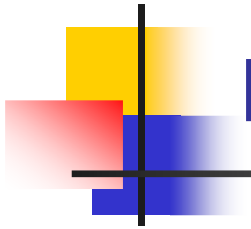
Complex content model:

element	element reference
sequence	concatenation
choice	union
all	unordered sequence
any	any element
group	named subexpression

Element definitions: groups

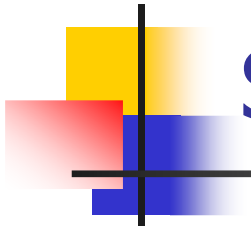
```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<group name="nimiryhmä">
    <sequence>
        <element name="etunimi" type="string"/ >
        <element name="sukunimi" type="string"/ >
    <sequence>
</group>

<complexType name="nimiryhmä">
    <group ref="target:nimiryhmä"/>
    <attribute name="arvo" type="string"/ >
</complexType>
<element name="Nimet" type="target:nimiryhmä"/>
</schema>
```



DTD or Schema or another definition?

- XML Schema
  - Follows the XML definition
  - Supports namespaces
  - Allows complex element types, object inheritance
- DTD
  - Can be included in the XML document instance
  - Allows entities
  - A large base of DTDs
- Relax NG

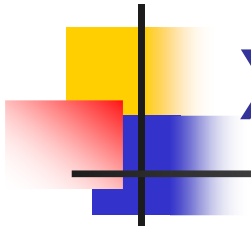


Schema status in March 2009

- "The XML Schema (XSD) specification from W3C is a paradox: it is one of the most heavily criticised specifications to come out of the organisation, but at the same time it has been widely adopted and implemented, and it can be said to have met all its design objectives.
- The responsible working group has been developing a new version, XSD 1.1, which is starting to get close to the finish line. Many of the difficulties with the specification (such as its immense complexity) will still be there, but some of the criticisms, notably those concerned with the limited functionality of the spec, are met head on with some powerful new features:
- Assertions, borrowed from Schematron, supplement the ability to define constraints using grammar and datatypes by a general predicate mechanism based on XPath."

GML schema -definition GML.XSD

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema targetNamespace="http://www.opengis.net/gml"
xmlns:xlink="http://www.w3.org/1999/xlink" xmlns:gml="http://www.opengis.net/gml"
xmlns:sch="http://www.ascc.net/xml/schematron" xmlns="http://www.w3.org/2001/XMLSchema"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"
attributeFormDefault="unqualified" version="3.0.1">
        <xsd:annotation>
        <xsd:appinfo source="urn:opengis:specification:gml:schema-xsd:gml:v3.0.1">
                gml.xsd
        </xsd:appinfo>
        <xsd:documentation>
        Copyright (c) 2002 OGC, All Rights Reserved. Top level GML schema
        </xsd:documentation>
        </xsd:annotation>
<!-- ========================================================================== -->
<xsd:include schemaLocation="dynamicFeature.xsd"/>
<xsd:include schemaLocation="topology.xsd"/>
<xsd:include schemaLocation="coverage.xsd"/>
<xsd:include schemaLocation="coordinateReferenceSystems.xsd"/>
<xsd:include schemaLocation="observation.xsd"/>
<xsd:include schemaLocation="defaultStyle.xsd"/>
<!-- ========================================================================== -->
</xsd:schema>
```



XML Schema: future

- Large base of existing schemas
- But some problems: allows options in definitions
  - Does not require data type definition – important
  - Version control not defined: plan yourself!
  - Extensible: allows extensions but does not define how
- Supported by many tools like . NET Studio