VEEBIRAKENDUSTE LOOMINE

MTAT.03.230 (6EAP)

10. Loeng

Helle Hein

Teema: XML (*EXtensible Markup Language*)

On kasutatud Marlon Dumas slaide 2008

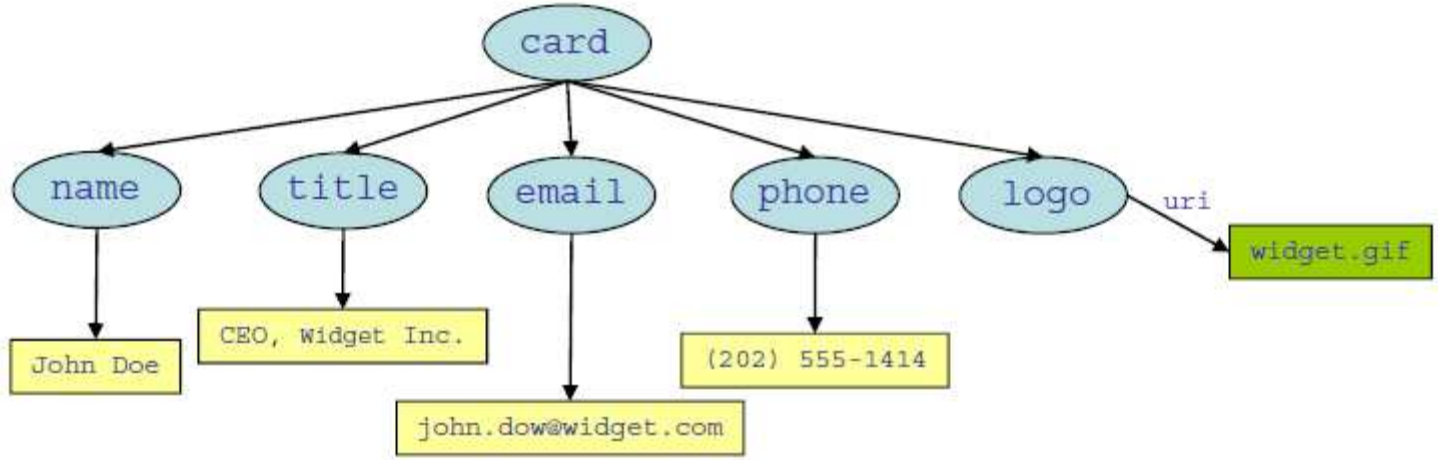
Täna loengus:

- XML
- XML Schema

Mis on XML?

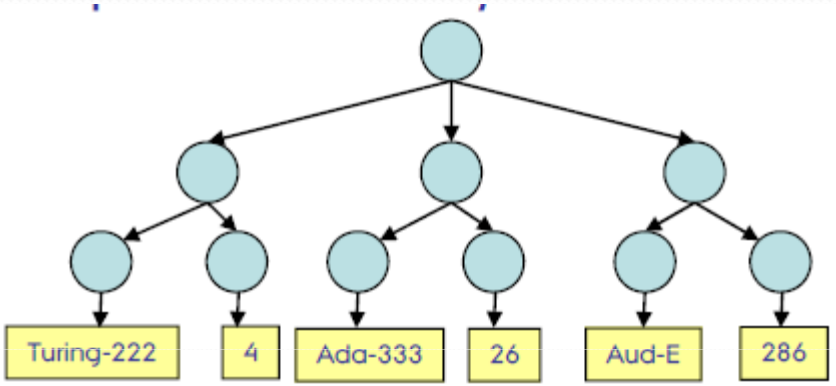
- Andmete kirjeldamiseks mõeldud keel kasutades hierarhilisi struktuure (puu kujulisi)
- Kasutatakse andmete ja dokumentide hoidmiseks ja edastamiseks
- Sõltumatu rakendusest ja platvormist
- Laiendatav: jätab ruumi variatsioonidele (semi-structured)

```
<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>
```



Andmebaasi tabelitele vastavad lihtsad puud:

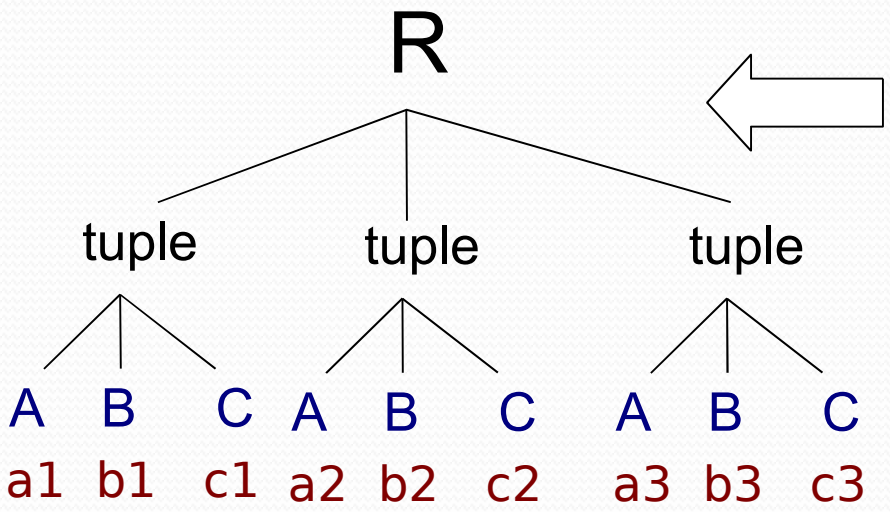
room	capacity
Turing-222	4
Ada-333	26
Aud-E	286



Kuid XML-le vastavad kõik puud

Tabelandmed XML-s

A	B	C
a1	b1	c1
a2	b2	c2
a3	b3	c3




```
<R>
   <tuple>
      <A>a1</A>
      <B>b1</B>
      <C>c1</C>
   </tuple>
   <tuple>
      <A>a2</A>
      <B>b2</B>
      <C>c2</C>
   </tuple>
     …
</R>
```

- Näide XML dokumendist:

```
<?xml version="1.0"?>
 <course>
   <code> MTAT.03.230 </code>
   <name> Web Application Development </name>
   <description> an excellent course </description>
 </course>
```

- XML dokumendil on üks juurelement (*root*) (kuid XML fragmendil võib olla palju juurelemente)
- Tõstutundlik!

XML dokumendi süntaksireeglid

- Esimene rida algab deklaratsiooniga, nt

```
<?xml version="1.0" encoding="ISO-8859-1"?>
```

- Teisel real tuleb määrata XML dokumendi DTD või XML Schema, kuid see ei ole kohustuslik

```
<!DOCTYPE course SYSTEM "course.dtd">
```

- Kõik märgised tuleb lõpetada
- Elemendid peavad olema üksteisesse tervenisti sisestatud
- Märgised on tõstutundlikud
- Tühikuid ei eemaldata

Millest koosneb XML dokument?

- Elemendid
  - Liitelemendid (compound), võivad sisaldada teisi elemente
  - Sisaldada väärtusi.
- Atribuudid
  - atomaarsed,
  - kasutatakse elementide kirjeldamiseks.
- Nimeruumid:
  - Sarnased nimeruumidega programmeerimiskeeltes (nt. Java, C#),
  - esitatakse URI-na, kuid nad on lihtsalt identifikaatorid – mitte reaalsed URI-d.

XML: elemendid

- Elemendid esitavad informatsiooni, objekte

  nt `<course> … </course>`

  Elemendid võivad sisaldada teksti

  `<description> A well recognised degree </description>`

- Elemendid võivad sisaldada teisi elemente, nt.

```
<person>
    <name> Maido </name>
    <age> 21 </age>
</person>
```

- Elemendid võivad olla tühjad, sellisel juhul kirjutatakse:

  `<age></age>` või `<age/>`

```
<!-- comments are the same as in HTML -->
```

XML: atribuudid

- Elemendid võivad lisaks tekstile ja teistele elementidele sisaldada ka atribuute
- Atribuudid on väärtused, mis on seotud elementidega; nad on atomaarsed (ei sisalda elemente):

```
<person name=“Maido" age="21"> </person>
```

`name` ja `age` on atribuudid; alati “” vahel

- Kui element sisaldab vaid atribuute, siis võib lühendada esitust:

```
<person name="Maido" age="21"/>
```

XML nimeruumid (*namespaces*)

- Nimeruumid võimaldavad ilmutatud sisu XML dokumentidele nt. ut kursuste nimed:

```
<?xml version="1.0"?>
 <course xmlns:ut="http://www.ut.ee">
   <ut:code> MTAT.03.230 <ut:code>
   <ut:name> Web Application Development </ut:name>
   <ut:description> An useful course </ut:description>
 </course>
```

- Nimeruum ei pea järgima mingit semantikat, see ei pea olema isegi kehtiv – see on lihtsalt eristav sõne

Vaikimisi nimeruumid

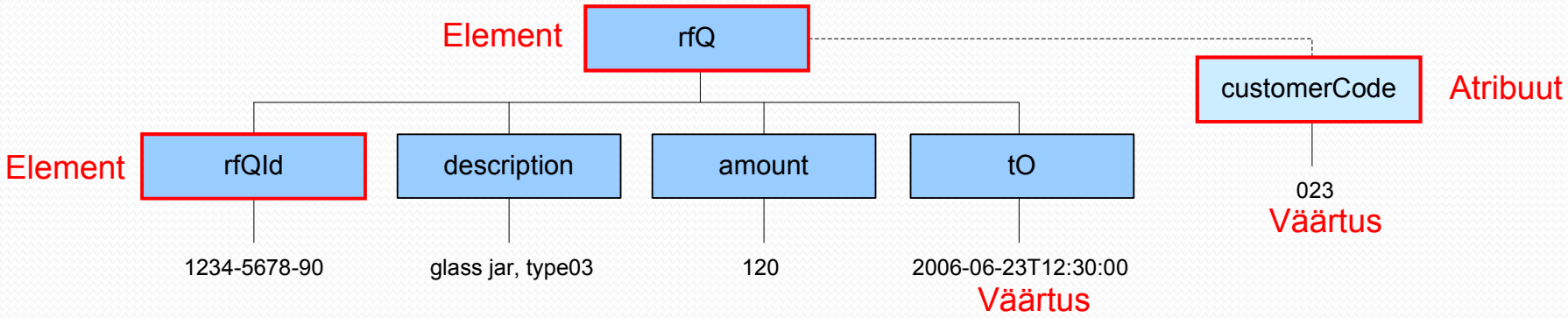
- Vaikimisi nimeruum rakendub kõikidele elementidele skoobis

```
<?xml version="1.0"?>
<course xmlns=http://www.ut.ee>
    <code> MTAT.03.230 </name>
    <name> Web Application Development </name>
    <description> An useful course </description>
</course>
```

Mitu nimeruumi

```
<?xml version="1.0"?>
<course xmlns:ut="http://www.ut.ee"
        xmlns:erasmus="http://ec.europa.eu">
  <ut:code> MTAT.03.230 </ut:code>
  <ut:name> Web Application Development </ut:name>
  <ut:description> An useful course </ut:description>
  <ut:credits> 6 </ut:credits>
  <eramus:credits> 6 </erasmus:credits>
</course>
```

XML dokumendid on esitatavad puu kujul



...ja selle esitus:

```
<?xml version="1.0" encoding="UTF-8"?>
<tns:rfQ xmlns:tns="http://www.company.com/BPM/salesX"
    tns:customerCode="023">
  <tns:rfQId>1234-5678-90</tns:rfQId>
  <tns:description>glass jar, type03</tns:description>
  <tns:amount>120</tns:amount>
  <tns:tO>2006-06-23T12:30:00</tns:tO>
</tns:rfQ>
```

Nimeruum Töötlemise eeskiri

Atribuut & väärtus

Element & väärtus

HTML vs XML

- HTML on keel, mille abil kirjeldatakse teksti esitust veebilehitsejas.
- Ta on segu sisust ja esitusest, nt:

```
<B>Marlon Dumas</B>
```

  Väidab, et Marlon Dumas tuleb esitada `bold` stiilis; ta ei väida, et see on nimi.
- XML on keel andmete kirjeldamiseks – mitte tingimata esitamiseks
- XML on laiendatav, st selles ei ole fikseeritud märgiseid nagu keeles HTML

Miks on XML oluline?

- Laialt tunnustatud standard, mida toetavad paljud vahendid
- XML saab kasutada andmete esitamiseks ja andmevahetuseks:
  - Veebirakendustes: andmete sisu ja esituse eraldamiseks
  - Organisatsioonide vahel (nt. RosettaNet, HL7)
  - Konfiguratsiooniandmete, kasutajaprofiilide jm salvestamiseks
- Palju taustakomponente kasutavad XML (või SOAP) liideseid:
  - otsingumootorid (nt. Solr)
  - ettevõtte süsteemid (nt. Salesforce)
- Paljud infosüsteemid kasutavad XML API-sid, nt.
  - aadressi tuvastus, ilmaennustus jne
  - Eesti X-Tee infrastruktuur (nt. äriregister)

XML valideerimine (*validation*)

- Korrektselt struktureeritud XML dokumendi kohta öeldakse, et see on reeglipärane, vormikohane (*well formed)*. Kuid on soovitatav teha piiranguid XML dokumentidele konkreetsetes rakendustes

- Võib nõuda, et kindlad elemendid ja atribuudid on olemas ja teised mitte

- Tahame defineerida kehtiva dokumendi struktuuri
- ➔ XML Schema

Reeglipärane *(well-formed)* XML

XML dokument on reeglipärane, kui kehtivad järgmised süntaksireeglid:

- algab XML deklaratsiooniga;
- tal on üks unikaalne juurelement (*root*);
- algusmärgised peavad olema kooskõlas lõpumärgistega;
- elemendid on tõstutundlikud;
- kõik märgised on lõpetatud;
- elemendid peavad olema üksteisesse tervenisti sisestatud;
- kõikide atribuutide väärtused peavad olema “” sees;
- erimärkide jaoks tuleb kasutada olemeid;

Valiidne (*valid*) XML

- Reeglipärane XML dokument võib lisaks olla ka valiidne, kui ta täidab teatud lisanõudeid
  - talle on vastavusse seatud dokumendi tüübi deklaratsioon (DTD) või skeem (*schema*) ja dokument täidab selles olevaid nõudeid

XML skeem (*schema*)

- XML skeem on XML-l baseeruv alternatiiv DTD-le.
- XML skeem kirjeldab XML dokumendi struktuuri.
- XML skeem:
  - defineerib dokumendi elemendid ja atribuudid;
  - defineerib alamelemendid, nende arvu ja järjekorra;
  - defineerib, kas element on tühi või võib sisaldada teksti;
  - defineerib elementide ja atribuutide andmetüübid;
  - defineerib elementide ja atribuutide fikseeritud ja vaikeväärtused;
  - ...

XML skeem

- XML skeem on samuti XML dokument
- Element "`element`" võimaldab defineerida elemente:
  - `name`
  - `type` – elemendi tüüp nt string, integer, decimal, date, liittüüp, jne.
- `minOccurs, maxOccurs` – esinemise kitsendused
- Ühesuse (unikaalsuse) kitsendused sarnaselt relatsiooniliste andmebaasidega (`unique`)
- Element "`attribute`" võimaldab defineerida atribuute:
  - `name`
  - `type` – atribuudi tüüp, peab olema lihtne XML tüüp
- `use` – kas element on kohustuslik (`use = "required"`)

Lihtelemendid

- Lihtelement on XML element, mis sisaldab ainult teksti ja ei tohi sisaldada teisi elemente ega ka atribuute

```
<lastname>Tamm</lastname>
<age>36</age>
<dateborn>1970-03-27</dateborn>
```

Skeemi süntaks lihtelemendi jaoks:

```
<element name="xxx" type="yyy"/>
```

Tüübid:

- string
- decimal
- integer
- boolean
- date
- time
- …

```
<element name="lastname" type="string"/>
<element name="age"      type="integer"/>
<element name="dateborn" type="date"/>
```

Liitelemendid

Liitelement on XML element, mis sisaldab teisi elemente ja/või atribuute.

Liitelemente on nelja tüüpi:

- Tühi element

```
<product pid="1345"/>
```

- Element, mis sisaldab ainult teisi elemente

```
<employee>
  <firstname>John</firstname>
  <lastname>Smith</lastname>
</employee>
```

Märkus: Iga selline element võib sisaldada ka atribuute!

Liitelemendid

- Element, mis sisaldab ainult teksti

```
<food type="dessert">Ice cream</food>
```

- Element, mis sisaldab nii teisi elemente kui ka teksti

```
<description> It happened on
   <date lang="norwegian">03.03.99</date>
....
</description>
```

Märkus: Iga selline element võib sisaldada ka atribuute!

XML skeem – liittüübid (*complex types*)

- Liittüüp defineerib XML elemendi struktuuri; ta kirjeldab, kuidas element peab välja nägema, kuid see ei ole XML element ise!
- Liittüüp kirjeldab:
  - Milliseid atribuute võivad sisaldada tema isendid
  - Milliseid elemente (st. alamelemente) võivad sisaldada tema isendid
  - Kas elemendid peavad olema järjestatud (*sequence*)

XML skeem – lihtne näide

note.xml

```
<?xml version="1.0"?>
<note>
     <to>Tove</to>
     <from>Jani</from>
     <heading>Reminder</heading>
     <body>Don't forget me this weekend!</body>
</note>
```

XML skeem – lihtne näide

```
<?xml version="1.0"?>
<schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.w3schools.com"
xmlns="http://www.w3schools.com"
elementFormDefault="qualified">

<element name="note">
  <complexType>
    <sequence>
      <element name="to" type="string"/>
      <element name="from" type="string"/>
      <element name="heading" type="string"/>
      <element name="body" type="string"/>
    </sequence>
  </complexType>
</element>

</schema>
```

XML skeem - näide

```
<schema targetNamespace="http://www.company.com/BPM/salesX"
  xmlns:tns="http://www.company.com/BPM/salesX"
  xmlns="http://www.w3.org/2001/XMLSchema">
  <element name="rfQ" type="tns:rfQMsgType"/>
  <element name="quote" type="tns:quoteMsgType"/>
  <element name="order" type="tns:quoteMsgType"/>
  <element name="cancelOrder" type="tns:quoteMsgType"/>
  <element name="rejectRfQ" type="tns:rejectRfQMsgType"/>
  <complexType name="rfQMsgType">
    <sequence>
      <element name="rfQId" type="string"/>
      <element name="description" type="string"/>
      <element name="amount" type="integer"/>
      <element name="tO" type="dateTime"/>
    </sequence>
    <attribute name="customerCode" type="string"/>
  </complexType>
```

Elemendid ja nende tüübid

Liittüüp elemendi rfQ jaoks

Atribuut ja selle tüüp

XML skeem – näide jätkub

```
<complexType name="quoteMsgType">
    <sequence>
      <element name="rfQId" type="string"/>
      <element name="cost" type="double"/>
    </sequence>
  </complexType>
  <complexType name="rejectRfQMsgType">
    <sequence>
      <element name="rfQId" type="string"/>
      <element name="reason" type="string"/>
    </sequence>
  </complexType>
</schema>
```

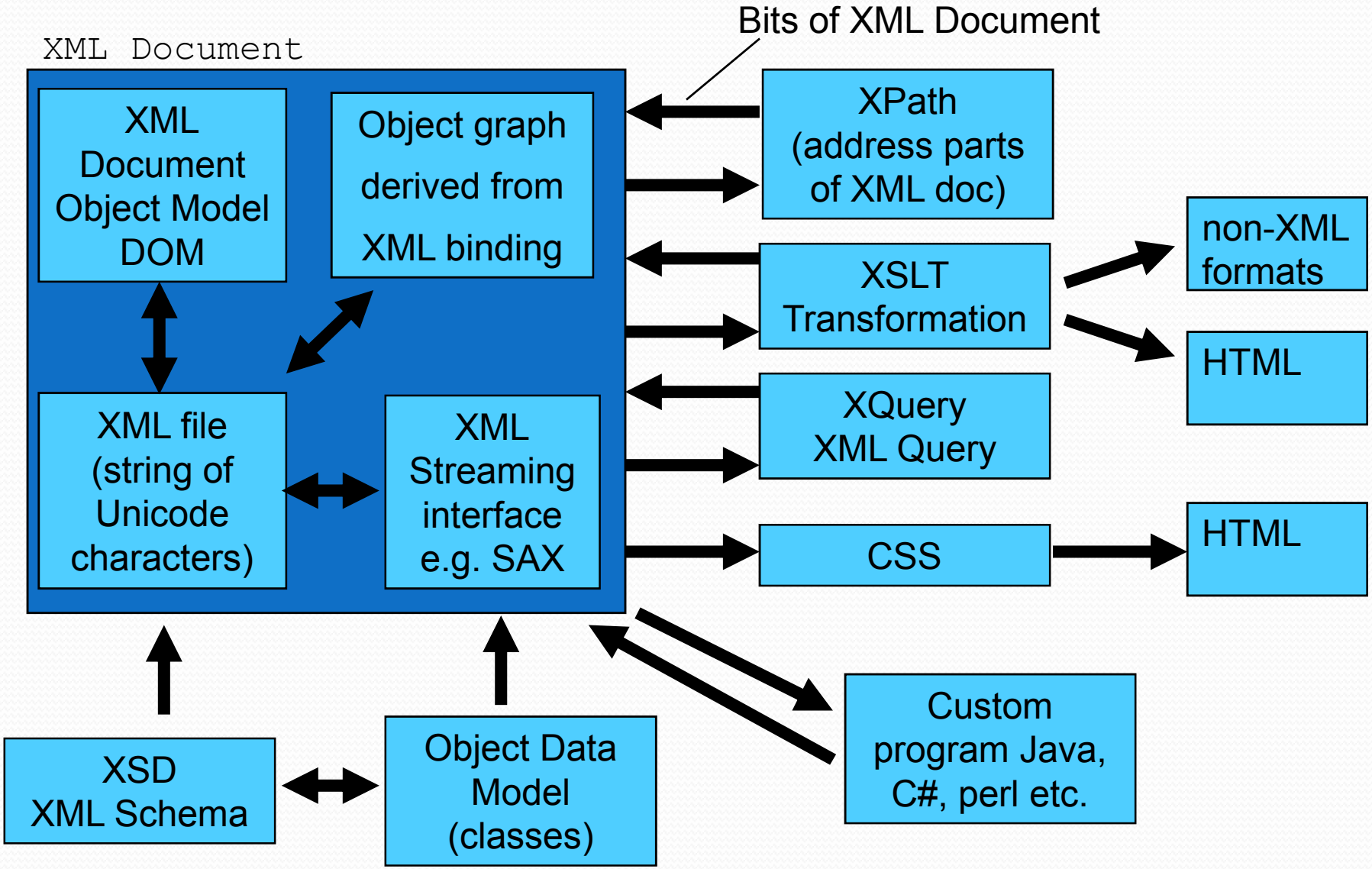
XML skeem – lisatunnused

- Tüübi piirang: uue tüübi loomisel piiratakse olemasolevate väärtuste hulka
- Tüübi laiendamine: uue tüübi loomine laiendades varem defineeritud tüübi elemente ja atribuute
- Välisvõtmed: inspireeritud relatsioonilistest andmebaasidest
- Segaelemendid, mis võivad sisaldada vaba teksti ja alamelemente
- Ja palju muud… XML skeem on keeruline standard!
- XML valideerimise analüüsijad lubavad kontrollida, kas XML dokument on antud XML skeemi isend
  - Nt Apache Xerces

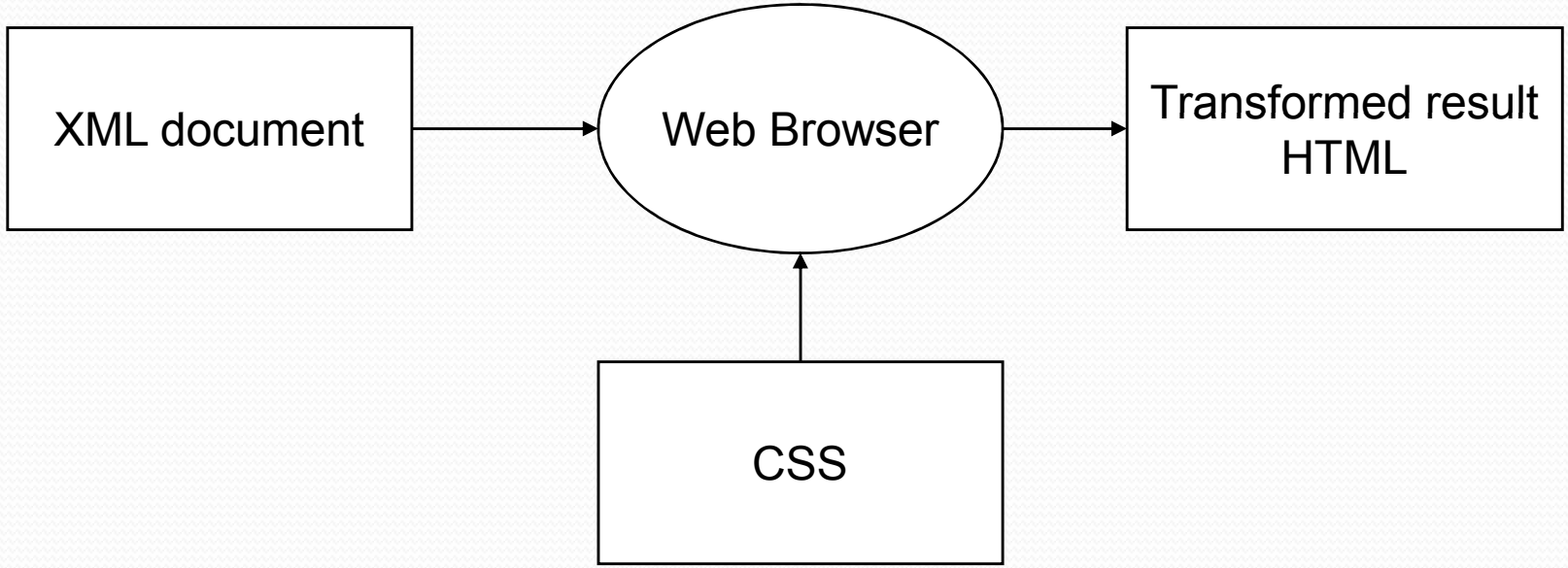
XML töötlemine

- Visualiseerimiseks:
  - *Cascading Style Sheets* (CSS)
  - *Extensible Style Language* (XSL)
- Teisendamiseks, *extraction*, *aggregation*:
  - DOM (*Document Object Model*)
  - SAX (Simple API for XML)
  - XPath, XSL ja XSL *Transformations* (XSLT)
  - XQuery

XML Processing



Cascading Style Sheets: CSS



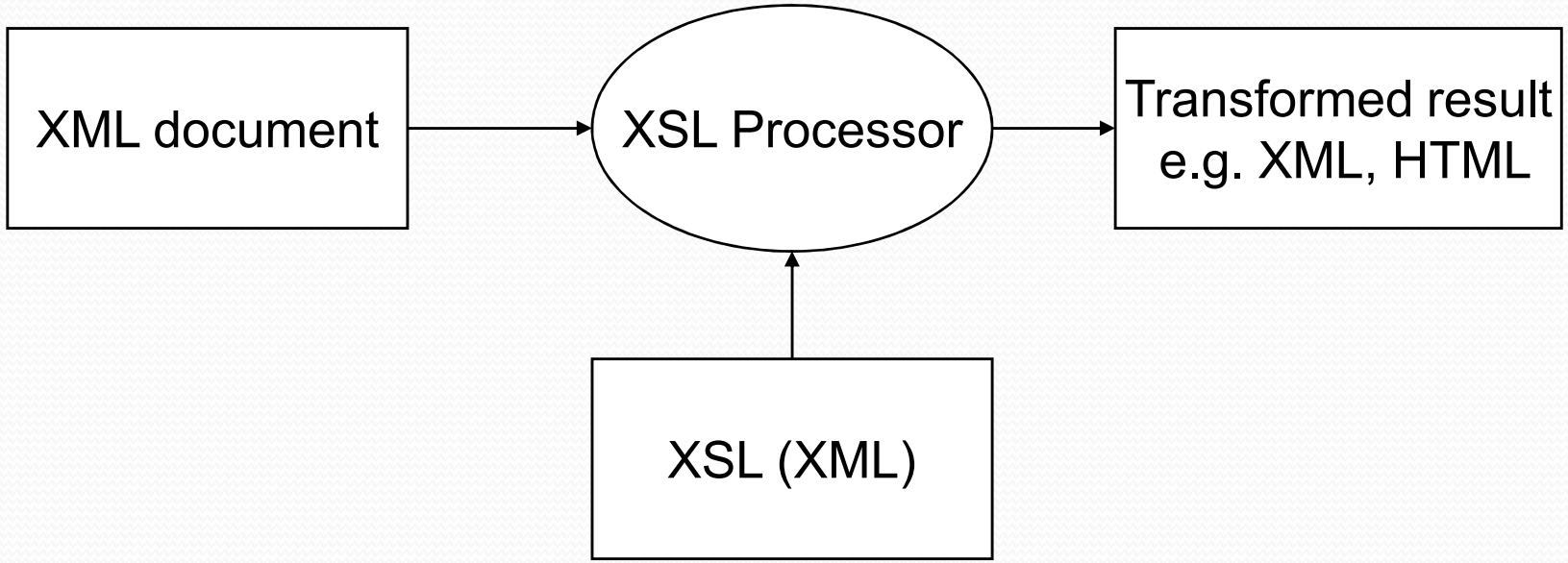
XML näitamine kasutades CSS

```
<?xml version="1.0" ?>
<?xml-stylesheet href="my-style.css" type="text/css"?>
<course>
  <name> Web Information Systems </name>
  <description> an excellent course</description>
</course>
```

my-style.css:

```
course {
  display: block;
  background-color: white;
  text-align:left;
  border: 4px double black;}
name {
  …}
description {
  display: block;
  …}
```

XSL & XSLT



`XSL & XLST` teema on järgmisel nädalal

CSS vs XSL

	CSS	XSL
Kas saab kasutada koos HTML-ga?	Jah	Ei
Kas saab kasutada koos XML-ga?	Jah	Jah
Transformatsiooni (teisenduse) keel?	Ei	Jah
Süntaks	CSS	XML

References

- CSS and XSL
- http://www.w3.org/Style/

DOM, SAX and XML-Object bindings

- SAX (Simple API to XML) - low overhead way to parse XML, event based so that don't have to load whole document before start parsing.

`http://msdn.microsoft.com/xml/articles/joyofsax.asp`

- DOM Document Object Model is an object model for XML, typically available though an API, data structure representing XML in memory - result of parsing XML
- XML Bindings (e.g. JAXB):
  - Generate classes from an XML schema
  - Loads XML documents into object graph
  - Exports object graphs into XML.

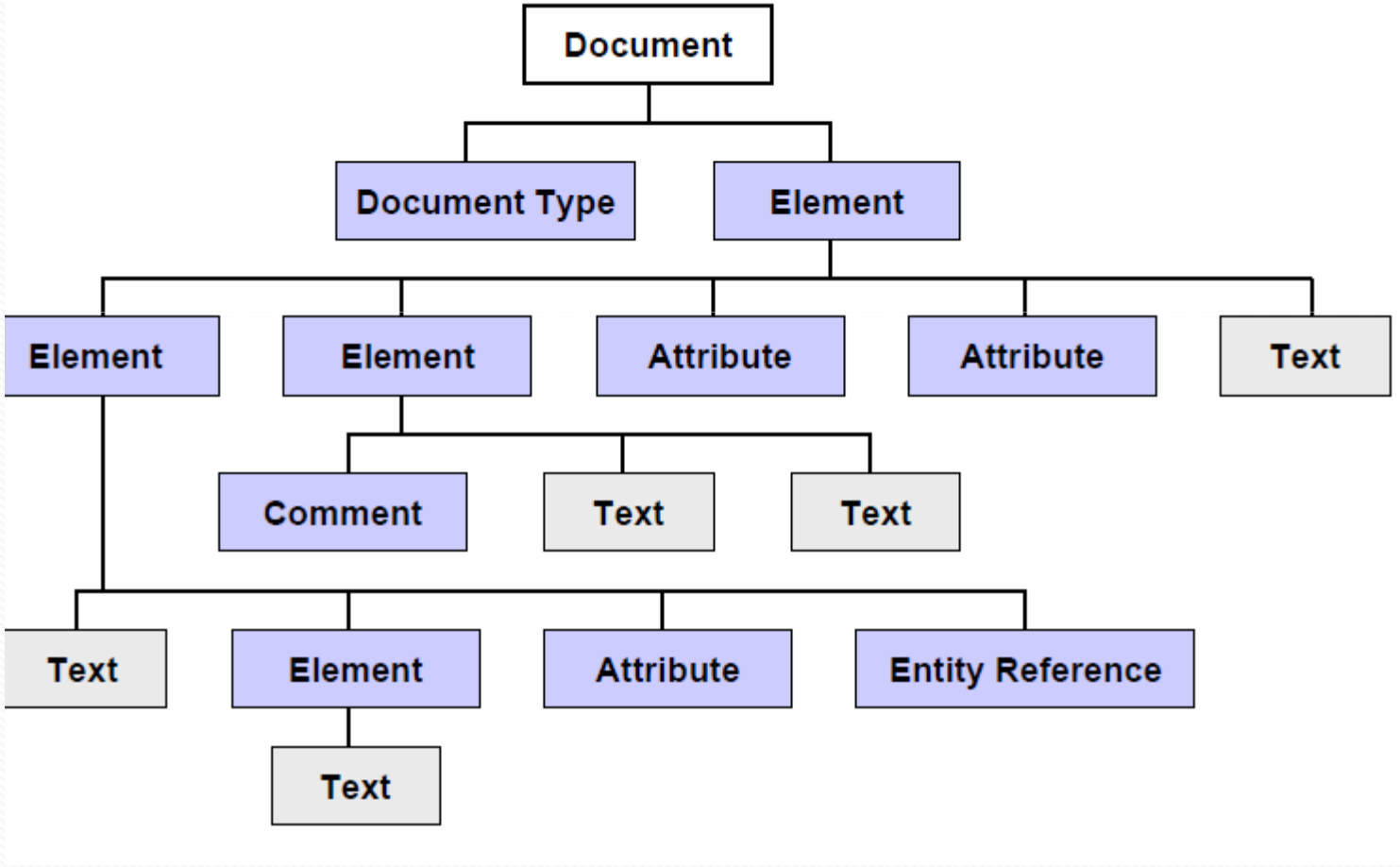
XML analüüs: DOM ja SAX

- **DOM (Document Object Model)**
  - analüüsib tervet dokumenti
  - esitab tulemuse puuna
  - võimaldab puust otsida
  - võimaldab puud muuta
  - sobib konfiguratsioonifailide lugemiseks/muutmiseks
- **SAX**
  - analüüsib niikaua, kuni kasutaja teda peatab
  - tekitab sündmusekäsitlejaid iga:
    - algusmärgise
    - märgise keha
    - märgise lõpu korral
  - madalatasemeline
  - Sobib suurte dokumentide jaoks, eriti kui on vaja sealt väikesi portse

Document Object Model (DOM)

- **DOM supports navigating and modifying XML documents**
  - Hierarchical tree representation of document
- Tree follows standard API
- Creating tree is vendor specific
- **DOM is a language-neutral specification**
  - Bindings exists for Java, C++, CORBA, JavaScript, C#
- Bad news: methods do not precisely follow usual Java naming convention
- Good news: you can switch to other languages (e.g., JavaScript and AJAX) with minimal learning curve
- **Official Website for DOM**
  - http://www.w3c.org/DOM/

DOM puu


Document
Document Type
Element
Element
Element
Attribute
Attribute
Text
Comment
Text
Text
Text
Element
Attribute
Entity Reference
Text

Steps to Using DOM:

Creating a Parsed Document

1. Import XML-related packages

```
import org.w3c.dom.*;
import org.xml.sax.*;
import javax.xml.parsers.*;
import java.io.*;
```

2. Create a DocumentBuilder

```
DocumentBuilderFactory factory =
DocumentBuilderFactory.newInstance();
DocumentBuilder builder = factory.newDocumentBuilder();
```

3. Create a Document from a file or stream

```
Document document = builder.parse(new File(file));
```

4. Extract the root element

```
Element root =
document.getDocumentElement();
```

5. Examine attributes

`getAttribute("attributeName")` returns specific attribute

`getAttributes()` returns a `Map` (table) of names/values

6. Examine sub-elements

`getElementsByTagName("subelementName")` returns a list of subelements of specified name

`getChildNodes()` returns a list of all child nodes

– Both methods return data structure containing `Node` entries, not `Element`.

`Node` is parent interface of `Element`

Results of `getElementsByTagName` can be typecast to `Element`

Results of `getChildNodes` are `Node` entries of various types

Example: Extracting Simple Top- Level Information

- Look at root element only
- Assume attribute names known in advance

```
<?xml version="1.0" encoding="utf-8"?>
  <company name="... Applied Physics Laboratory"
       shortName="JHU/APL" mission="Enhancing national
       security ...">
    <head name="Richard Roca" phone="410-778-1234"/>
  <department name="Air and Missile Defense"
       mission="Enhance the operational ...">
    <group name="A1E" numStaff="20"/>
    <group name="A1F" numStaff="15"/>
    <group name="A1G" numStaff="25"/>
  </department>
  <department name="Research and Technology..."
       mission="Assure that ...">
    <group name="RSI" numStaff="15"/>
    <group name="RSS" numStaff="10"/>
  </department>
</company>
```

Example: Source Code

```
import org.w3c.dom.*;
import org.xml.sax.*;
import javax.xml.parsers.*;
import java.io.*;
public class DomTest1 {
 public static void main(String[] args) {
  String file = "test1.xml";
   if (args.length > 0) {
    file = args[0];
 }
```

```
try {
    DocumentBuilderFactory factory =
      DocumentBuilderFactory.newInstance();
    DocumentBuilder builder = factory.newDocumentBuilder();
    Document document = builder.parse(new File(file));
    Element root = document.getDocumentElement();
    System.out.println(root.getTagName());
    System.out.printf(" name: %s%n", root.getAttribute("name"));
    System.out.printf("short name: %s%n",root.getAttribute("shortName"));
    System.out.printf(" mission: %s%n",root.getAttribute("mission"));
} catch(Exception e) {
    e.printStackTrace();
}
}
}
```

DOM vs. XML-Object bindings

```
<?xml version="1.0" encoding="UTF-8"?>
<tns:rfQ xmlns:tns="http://www.company.com/BPM/salesX"
tns:customerCode="023">
  <tns:rfQId>1234-5678-90</tns:rfQId>
  <tns:description>glass jar, type03</tns:description>
  <tns:amount>120</tns:amount>
  <tns:tO>2006-06-23T12:30:00</tns:tO>
</tns:rfQ>
```

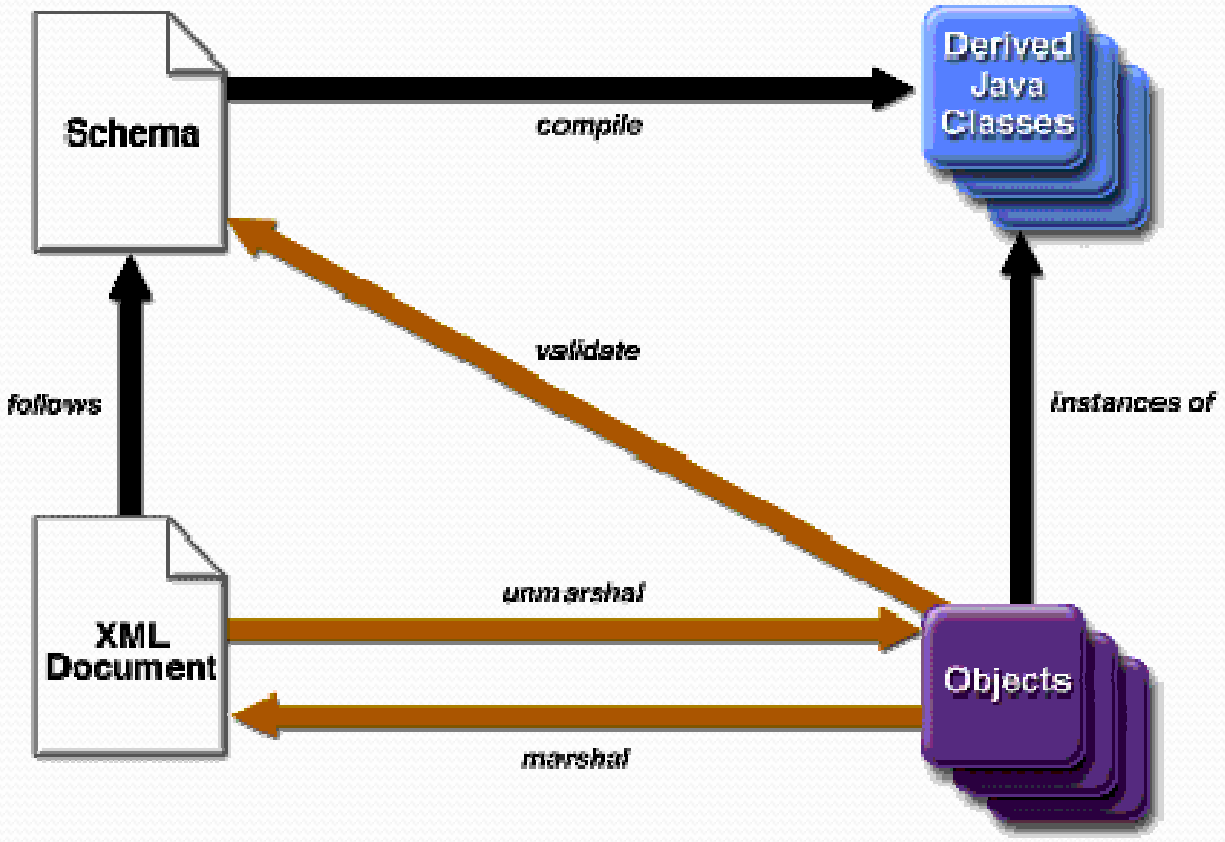
Language Binding (XML types - objects)

```
public class rfQ{
    public String customerCode;
    public String rfQId;
    public String description;
    public int amount;
    public String tO;}
```

DOM (explicit XML representation)

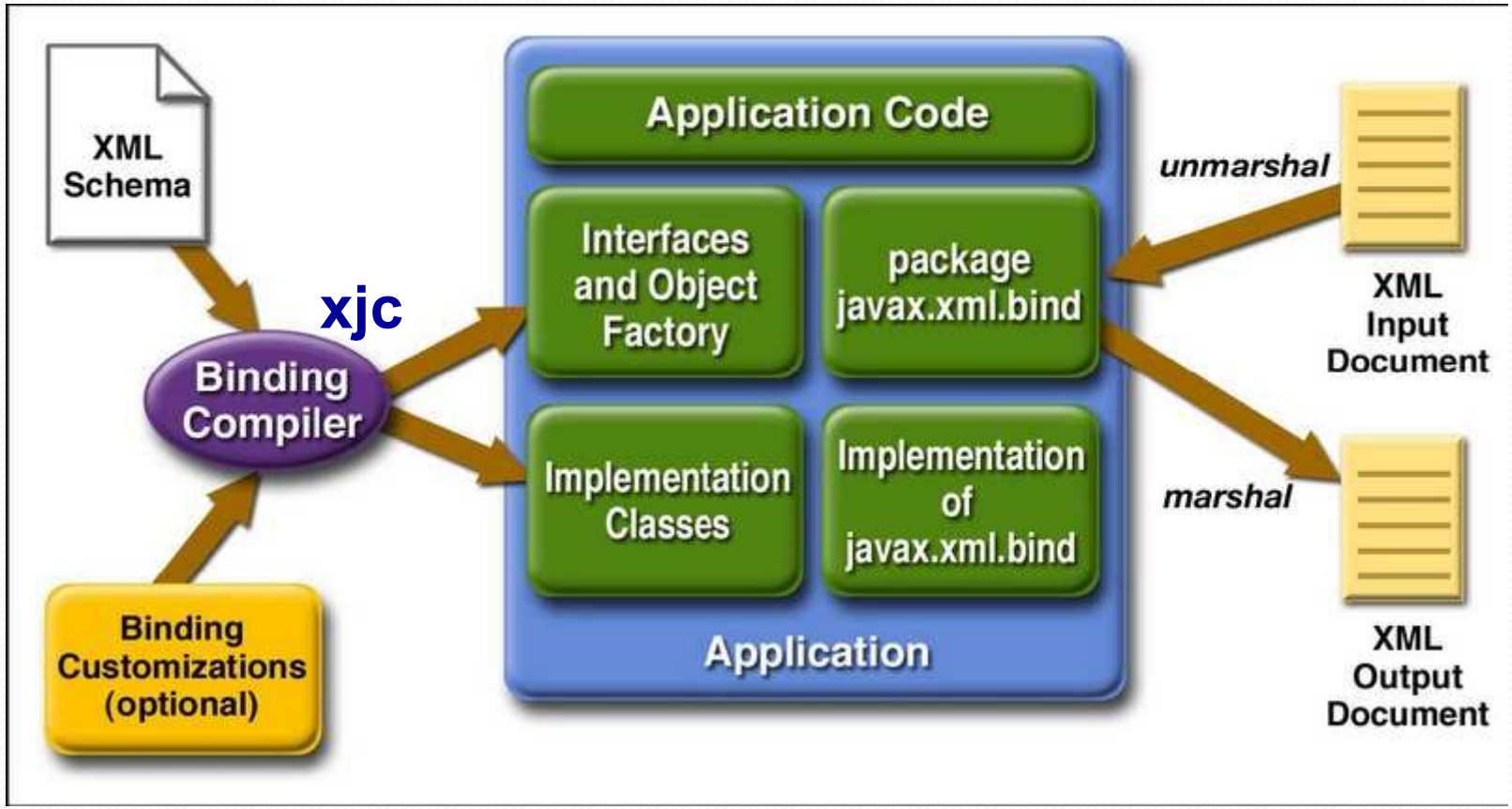
```
public class Element{
    public String Name;
    public String Value;
    public ElementList elems;
    public AttributeList atts;}

public class Attribute{
    public String Name;
    public String Value;}
```

Java Architecture for XML Binding



© RWTH, Aachen

JAXB – Schema-first approach



© SUN

Reverse approach: Generate schema from existing classes

JAXB: Näide (library.xml)

```
<library>
  <book>
      <title>title 1</title>
      <author>author 1</author>
      <editor>editor 1</editor>
  </book>
  <book>
      <title>title 2</title>
      <author>author 2</author>
      <editor>editor 2</editor>
  </book>
</library>
```

JAXB: Näide (library.xsd)

```
<xs:schema
  xmlns:xs="http://www.w3.org/2001/XMLSchema">
<xs:element name="library">
  <xs:complexType>
      <xs:sequence>
      <xs:element ref="book"
  maxOccurs="unbounded"/>     </xs:sequence>
  </xs:complexType>
</xs:element>
```

JAXB: Näide (library.xsd)

```
<xs:element name="book">
  <xs:complexType>
  <xs:sequence>
      <xs:element name="title" type="xs:string"/>
      <xs:element name="author"
  type="xs:string"/>
      <xs:element name="editor"
  type="xs:string"/>
  </xs:sequence>
  </xs:complexType>
</xs:element>
</xs:schema>
```

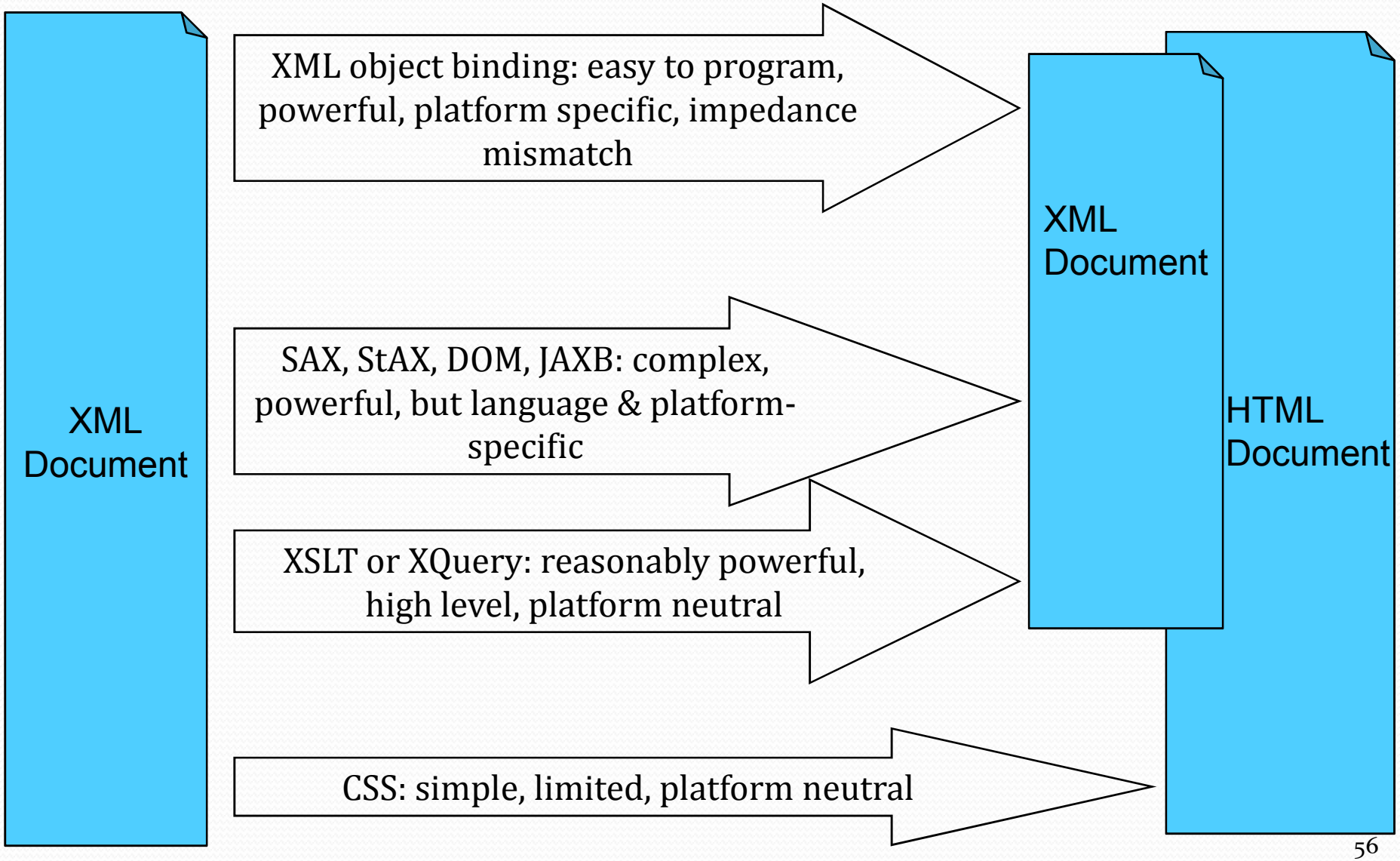
JAXB: `xjc` kompileerija

- Käsk "`xjc library.xsd`" genereerib paketi "`generated`" koos:
  - `Library.java`
  - `Book.java`
  - `ObjectFactory.java`
- Neid klasse ja JAXB API klasse võib hiljem kasutada (*marshall*), *unmarshall*

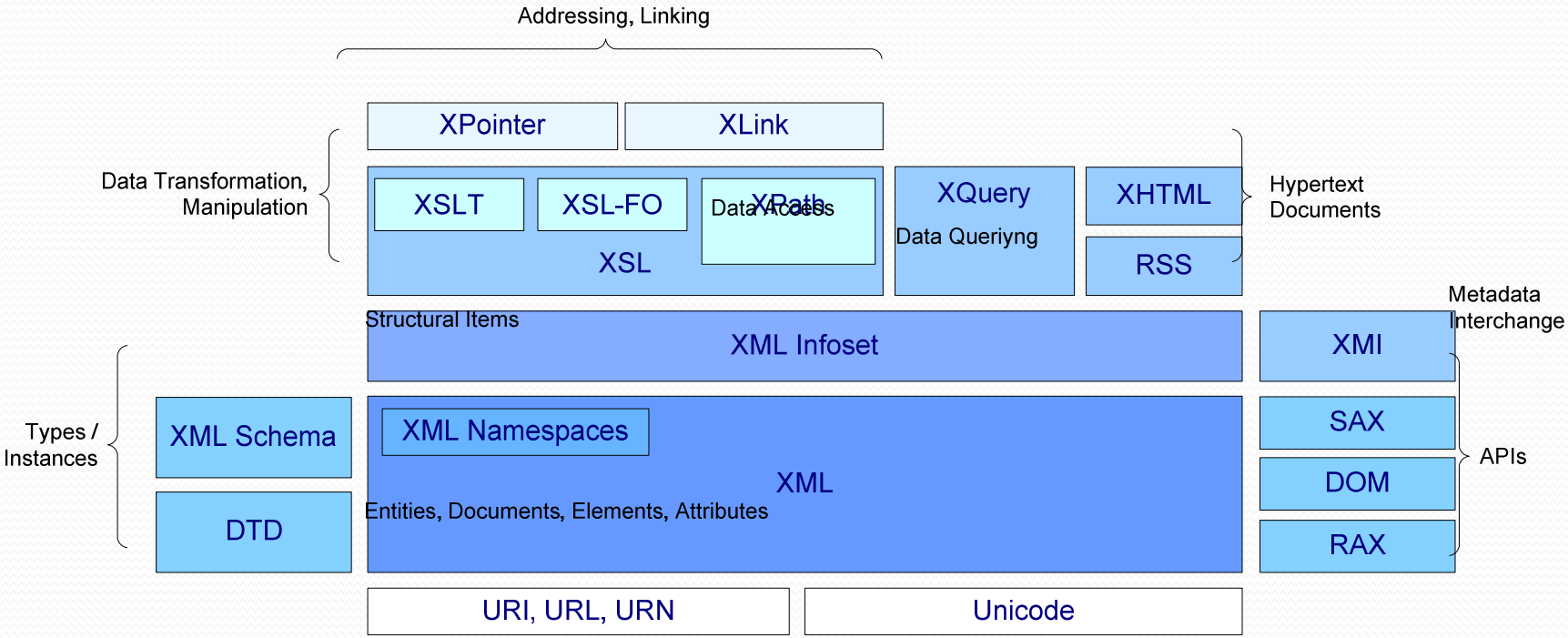
JAXB: näidis Java kood

```
import generated.*;
import javax.xml.bind.*;
import java.io.*; import java.util.*;
public class Main {
    public static void main(String[] args) {
  try {
      JAXBContext jc = JAXBContext.newInstance("generated");
      Unmarshaller unmarshaller = jc.createUnmarshaller();
      Library library = (Library) unmarshaller.unmarshal(new
       File("test.xml"));
      List<Book> books = library.getBook();
      for (int i = 0; i < books.size(); i++) {
        Book book = (Book) books.get(i);
        System.out.println("Title   : " + book.getTitle());
        System.out.println("Author  : " + book.getAuthor());
        System.out.println("Editor : " + book.getEditor());
  } catch (Exception e) { e.printStackTrace();  }
 }
}
```

Kokkuvõte: Võimalused XML töötluseks



Overview of XML Technology



Viited

- XML portals:
  - http://www.xml.com, http://www.xml.org
  - http://www.w3.org/xml
- Tutorials:
  - http://www.w3schools.com/xml
  - http://cm.bell-labs.com/cm/cs/who/wadler/xml/
- XML-related tools:
  - Apache Xerces: http://xerces.apache.org
  - XML Spy http://www.xmlspy.com
  - MS XML parsing and processing tools: http://msdn.microsoft.com/XML/XMLDownloads