XML
i nowoczesne metody zarządzania treścią

Wykład 5: Dokumentacja schematu
oraz inne formalizmy modelowania dokumentów

Maciej Ogrodniczuk

MIMUW, 7 listopada 2011

Trzy sposoby dokumentacji schematu

Za pomocą:

1. specjalnego elementu `<xsd:annotation>`,
2. komentarzy XML-owych,
3. atrybutów dołączanych (ang. *foreign attributes*).

Oraz oczywiście wykorzystując sposoby nieschematocentryczne:

- opisując schemat poza nim samym,
- przechowując łącznie opisy danego komponentu schematu, dostępne przekształcenia i style,
- ...

<xsd:annotation> — dokumentacja schematu

Element <xsd:annotation> może wystąpić w dowolnym miejscu na poziomie globalnym oraz na początku wszystkich konstrukcji XML Schema.

Jego zawartość to mieszanka elementów <xsd:documentation> i <xsd:appinfo> zawierających dodatkowe informacje (tekst i znaczniki) odpowiednio dla ludzi i maszyn.

```
<xsd:element name="nazwisko" type="xsd:string">
  <xsd:annotation>
    <xsd:documentation xml:lang="pl">
      Nazwisko adresata.
    </xsd:documentation>
    <xsd:appinfo>
      <form:label>Podaj nazwisko adresata:</form:label>
    </xsd:appinfo>
  </xsd:annotation>
</xsd:element>
```

Definicja <xsd:appinfo>

```
<xsd:element name="appinfo">
  <xsd:annotation>
    <xsd:documentation source="http://www.w3.org/TR/
                         xmlschema-1/#element-appinfo"/>
  </xsd:annotation>
  <xsd:complexType mixed="true">
    <xsd:sequence minOccurs="0" maxOccurs="unbounded">
      <xsd:any processContents="lax"/>
    </xsd:sequence>
    <xsd:attribute name="source" type="xsd:anyURI"/>
  </xsd:complexType>
</xsd:element>
```

Oba elementy mogą zostać opatrzone opcjonalnym atrybutem `source` wskazującym źródło informacji.

Co umieszczać w <xsd:appinfo>?

Na przykład:

- dodatkowe reguły walidacji (np. schematronowe),
- odwzorowanie na inne technologie (np. schemat relacyjny),
- odwzorowanie na formularze (np. XHTML-owe znaczniki <form>).

Własne atrybuty dołączane

Każdy komponent schematu może zawierać atrybuty pochodzące z dowolnej przestrzeni nazw. Atrybuty te nie są walidowane i nie trzeba ich deklarować.

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            xmlns:doc="http://www.mimuw.edu.pl/xml">
  <xsd:element name="wykład" type="typPrezentacja"
               doc:opis="Element główny prezentacji
                         na potrzeby wykładu."/>
</xsd:schema>
```

Zarządzanie złożonymi schematami

Warto standaryzować:

- nazewnictwo,
- sposób budowy schematu,
- sposoby rozszerzania schematu,
- zakres wykorzystania konstrukcji XML Schema.

Często tego rodzaju zalecenia zapisuje się formalnie w dokumencie NDR (ang. *naming and design rules*).

Warto też używać oprogramowania do kontroli nad:

- edycją: kto co zmienia,
- wersjami składników schematu,
- fragmentami schematu stosowanymi w różnych zastosowaniach.

Jeszcze o samym procesie modelowania

Jak modelować?

- analizując zależności między modelowanymi obiektami i ich częściami,
- wyodrębniając podstruktury obiektów,
- analizując dostępne dokumenty przykładowe,
- analizując potencjalne zastosowania dokumentów oraz przypadki użycia,
- budując abstrakcyjny projekt struktury,
- zapisując model,
- testując model w „rzeczywistym świecie”,
- utrzymując (pielęgnując) strukturę podczas jej wykorzystania,
- pamiętając o zarządzaniu zmianami.

Zarządzanie zmianami struktury

Problem:

- Konieczność zmiany struktury.

Warianty rozwiązania:

1. wprowadzamy zmianę modelu kompatybilną wstecz (np. dodajemy elementy, ale opcjonalne),
2. używamy dwóch wersji schematu,
3. migrujemy dokumenty: przekształcamy automatycznie i/lub instruujemy użytkowników o konieczności migracji do nowej struktury.

Najlepiej: tworzymy aplikację odporną na zmianę struktury dokumentów.

Metody zapewniania odporności na zmianę struktury

Kilka metod:

- przeniesienie sprawdzania poprawności dokumentu na poziom schematu,
- pomijanie nieistotnych elementów i atrybutów,
- unikanie zależności od struktury dokumentu: `//produkt/numer` zamiast `/katalog/produkt/numer`,
- parametryzacja schematem — użycie atrybutów stałych np. do przechowania etykiet pól formularza, odwzorowania elementów na tabele i pola w bazie danych itp.,
- użycie przestrzeni nazw (jak w standardzie XLink).

Nadrzędna zasada: zdrowy rozsądek.

O czym nie powiedzieliśmy (i nie powiemy)

Zachęcam do samodzielnych studiów nad:

- zależnościami aspektów od typów danych,
- regułami walidacji i ograniczeniami budowy schematu,
- użyciem encji, notacji, odpowiedników konstrukcji z DTD,
- grupami zamiennymi (ang. *substitution groups*),
- modelami struktury schematów (*Russian Doll*, *Salami Slice*, *Venetian Blind*, *Garden of Eden*),
- specyfikacją XML Schema w oryginale.

Polecam książkę Priscilli Walmsley *Wszystko o XML Schema* (WNT 2008).
Przykłady z książki: `http://www.datypic.com/books/defxmlschema/examples.html`.

Czego nie da się zrobić w XML Schema?

Problem:

- kontekstowe sprawdzanie poprawności,
  np. *zawartość elementu* `<cena-netto>` *jest mniejsza lub równa zawartości elementu* `<cena-brutto>`,
  albo: *lista liczb wylosowanych w Lotto jest posortowana.*

Rozwiązania:

- zaprogramować w kodzie aplikacji,
- wykorzystać XSLT,
- użyć innego języka schematów, np. Schematrona.

Czego jeszcze?

Problemy:

- niejednoznaczność (ang. *ambiguity*), czyli poprawność fragmentu względem kilku wzorców,
- niedeterminizm (ang. *non-determinism*), czyli sytuacja, w której procesor ma do wyboru wiele pasujących wzorców (produkcji gramatyki), a równoważny model deterministyczny nie istnieje.

Rozwiązanie:

- Relax NG, czyli
  REgular LAnguage description for XML – New Generation.

Schematron

Autorstwa Ricka Jelliffe'a (1999). Standard ISO od 2006 r.
(jako część 3 standardu DSDL — Document Schema Definition Languages).

Idea: wzorce (`<pattern>`) zawierające kontekstowe reguły walidacji (`<rule>`) złożone z posługujących się wyrażeniami XPath własności `<assert>` i `<report>` odpowiednio wymaganych do spełnienia i oznaczających błąd.

Schemat schematronowy dla schematrona (`www.schematron.com`):

```
<schema xmlns="http://www.mimuw.edu.pl/dsdl/schematron">
  <ns prefix="sch" uri="http://www.mimuw.edu.pl/dsdl/schematron">
  <pattern>
    <rule context="sch:schema">
      <assert test="sch:pattern">Schemat składa się z wzorców.</assert>
      <assert test="sch:pattern/sch:rule[@context]">Wzorzec składa
          się z reguł. Każda powinna mieć atrybut 'context'.</assert>
      <assert test="sch:pattern/sch:rule/sch:assert[@test]
           or sch:pattern/sch:rule/sch:report[@test]">Reguła
           składa się z instrukcji 'assert' i 'report',
           które muszą posiadać atrybut 'test'.</assert>
    </rule>
  </pattern>
</schema>
```

Asercje wbudowane w schemat: przykład

```
<xsd:element name="towar">
  <xsd:annotation>
    <xsd:appinfo>
      <sch:pattern name="Cena brutto większa od netto.">
        <sch:rule context="sklep:towar">
          <sch:assert test="sklep:cenaBrutto > sklep:cenaNetto">
             Zawartość elementu 'cenaBrutto' powinna być większa
             niż zawartość elementu 'cenaNetto'.</sch:assert>
        </sch:rule>
      </sch:pattern>
    </xsd:appinfo>
  </xsd:annotation>
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="cenaNetto" type="xsd:integer"/>
      <xsd:element name="cenaBrutto" type="xsd:integer"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

Relax NG w pigułce

Autorstwa OASIS, bazuje na wcześniejszych propozycjach RELAX Makoto Muraty i TREX Jamesa Clarka.

Od 2006 r. standard ISO — jako część 2 standardu DSDL.

Regular Language description for XML – New Generation:

- dwa warianty składni: XML-owa i „kompaktowa”,
- wsparcie dla przestrzeni nazw,
- jednolite traktowanie elementów i atrybutów,
- wsparcie dla zawartości nieuporządkowanej i mieszanej,
- może funkcjonować wraz z osobnym językiem typów (np. XML Schema).

W zasadzie same zalety:

+ prostszy w opisie schematu,
+ bardziej zaawansowany technicznie,
+ oferujący więcej możliwości,
− cieszący się zdecydowanie mniejszą popularnością (czyt.: wsparciem producentów oprogramowania).

Relax NG a DTD

RELAX NG rozszerza funkcjonalność DTD — w szczególności:

- wprowadza typy danych,
- integruje atrybuty z modelem zawartości,
- obsługuje przestrzenie nazw,
- zapewnia wsparcie dla dowolnej kolejności wystąpień elementów,
- obsługuje modele kontekstowe.

Jednocześnie RELAX NG:

- nie zapewnia walidacji `ID/IDREF` (jest dodatek, który to umożliwia),
- nie obsługuje atrybutów domyślnych,
- nie obsługuje encji znakowych, notacji,
- nie pozwala na określenie, czy białe znaki są znaczące,
- nie określa sposobu powiązania schematu Relax NG z dokumentem.

Dwie składnie RELAX NG

DTD:

```
<!ELEMENT wizytownik<!ELEMENT
wizytownik
          (wizytówka*)>(wizytówk

<!ELEMENT wizytówka<!ELEMENT
wizytówka
          (osoba,
email)>(osoba, email)>
<!ELEMENT osoba
(#PCDATA)><!ELEMENT osoba
(#PCDATA)>
<!ELEMENT email
(#PCDATA)><!ELEMENT email
(#PCDATA)>
```

Składnia skrócona:

```
element wizytownik {element
```

Składnia XML-owa:

```
<element
name="wizytownik"<element
name="wizytownik"
  xmlns="http://relaxng.org/xmlns

          ns/structure/1.0">ns/str

  <zeroOrMore><zeroOrMore>
    <element<element
      name="wizytówka">name="wizy

      <element
name="osoba"><element
name="osoba">
        <text/><text/>
      </element></element>
      <element
name="e-mail"><element
```

Kurs RELAX NG w przykładach: element wymagany

RELAX NG:

```
<element name="wizytownik"
         xmlns="http://relaxng.org/ns/structure/1.0">
  <oneOrMore>
    <element name="wizytówka">
    ...
    </element>
  </oneOrMore>
</element>
```

DTD:

```
<!ELEMENT wizytownik (wizytówka+)>
```

Kurs RELAX NG w przykładach: element opcjonalny

RELAX NG:

```
<element name="wizytówka">
  <element name="osoba">
    <text/>
  </element>
  <element name="e-mail">
    <text/>
  </element>
  <optional>
    <element name="telefon">
      <text/>
    </element>
  </optional>
</element>
```

DTD:

```
<!ELEMENT wizytówka (osoba, e-mail, telefon?)>
```

Kurs RELAX NG w przykładach: grupa i alternatywa

RELAX NG:

```
<element name="wizytówka">
  <choice>
    <element name="osoba">
      <text/>
    </element>
    <group>
      <element name="imię">
        <text/>
      </element>
      <element name="nazwisko">
        <text/>
      </element>
    </group>
  </choice>
</element>
```

DTD:

```
<!ELEMENT wizytówka (osoba | (imię, nazwisko))>
```

Kurs RELAX NG w przykładach: element pusty i atrybuty

RELAX NG:

```
<element name="osoba">
  <attribute name="telefon"/>
</element>
```

Uwagi:

- `<text/>` jest domyślną zawartością atrybutów,
- atrybuty są domyślnie wymagane, `IMPLIED` wymaga użycia `<optional>`,
- gdy nie ma atrybutów, element pusty oznaczamy jako `<empty/>`.

DTD:

```
<!ELEMENT osoba EMPTY>
<!ATTLIST osoba telefon CDATA REQUIRED>
```

Kurs RELAX NG w przykładach: fragmenty i *doctype*

RELAX NG:

```
<grammar>
  <start>
    <element name="wizytownik">
      <zeroOrMore>
        <element name="wizytówka">
          <ref name="treśćWizytówki"/>
        </element>
      </zeroOrMore>
    </element>
  </start>

  <define name="treśćWizytówki">
    <element name="osoba">
      <text/>
    </element>
    <element name="e-mail">
      <text/>
    </element>
  </define>
</grammar>
```

DTD:

```
<!DOCTYPE
  wizytownik

  [<!ELEMENT
     wizytownik
     (wizytówka*)>

   <!ENTITY %
     treśćWizytówki
     "osoba, e-mail">

   <!ELEMENT
     wizytówka
     (%treśćWizytówki;)>

   <!ELEMENT
     osoba
     (#PCDATA)>

   <!ELEMENT
     e-mail
     (#PCDATA)>]
```

Kurs RELAX NG w przykładach: typy danych

RELAX NG:

```
<element name="e-mail" datatypeLibrary=
         "http://www.w3.org/2001/XMLSchema-datatypes">
  <data type="string">
    <param name="maxLength">127</param>
  </data>
</element>
```

XML Schema:

```
<xsd:element name="e-mail">
  <xsd:simpleType>
    <xsd:restriction base="xsd:string">
      <xsd:maxLength value="127"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>
```

Kurs RELAX NG w przykładach: wyliczenia

RELAX NG:

```
<element name="plik">
  <attribute name="format">
    <choice>
      <value>HTML</value>
      <value>PDF</value>
    </choice>
  </attribute>
</element>
```

W podobny sposób można wyliczać zawartość elementów (jak w XML Schema).

DTD:

```
<!ATTLIST plik format (HTML | PDF) #REQUIRED>
```

Kurs RELAX NG w przykładach: listy

RELAX NG:

```
<element name="listaParzysta">
  <list>
    <oneOrMore>
      <data type="integer"/>
      <data type="integer"/>
    </oneOrMore>
  </list>
</element>
```

Kurs RELAX NG: dowolny porządek, zawartość mieszana

RELAX NG:

```
<element name="head">
  <interleave>
    <ref name="title"/>
    <zeroOrMore>
      <ref name="meta"/>
    </zeroOrMore>
  </interleave>
</element>

<element name="p">
  <mixed>
    <ref name="b"/>
    <ref name="i"/>
    <ref name="u"/>
  </mixed>
</element>
```

DTD dla modelu mieszanego:

```
<!ELEMENT p
  (#PCDATA | b | i | u)*>
```

RELAX NG: jeszcze raz (prawie) wszystko razem

- `<element name="`*nazwa*`">`, `<attribute name="`*nazwa*`">`,
- `<optional>` — atrybut lub element opcjonalny (domyślnie atrybuty wymagane),
- `<text/>` — to samo, co `#PCDATA`
- `<empty/>` — element pusty,
- `<zeroOrMore>`, `<oneOrMore>`,
- `<choice>`, `<group>`, `<interleave>` (dowolny porządek),
- `<mixed>`,
- `<grammar>` — element główny gramatyki,
- `<start>` — element główny dokumentu,
- ...

RELAX NG: zawartość prosta lub elementowa

```
<element name="osoba">
  <choice>
    <text/>
    <group>
      <element name="imię">
        <data type="token"/>
      </element>
      <optional>
        <element name="drugieImię">
          <data type="token"/>
        </element>
      </optional>
      <element name="nazwisko">
        <data type="token"/>
      </element>
    </group>
  </choice>
</name>
```

Examplotron

Autorstwa Erica van der Vlista, prace rozpoczęte w 2001 r., zamarły w 2003 r.

Idea: „definicja przez przykład” – egzemplarze dokumentów definiują schemat.

Jak to działa? Przekształcenie XSLT zamienia egzemplarze na „prawdziwy” schemat RELAX NG.

Obsługuje w prosty sposób 80% przypadków:

- definicja elementów i atrybutów,
- kontrola nad liczbą wystąpień elementów,
- zawartość mieszana,
- predefiniowane typy proste XML Schema.

`http://examplotron.org`

Examplotron: przykład

Examplotron:

```
<osoba>
  <imię>Petronela</imię>
  <imię>Zenobia</imię>
</osoba>
```

Relax NG:

```
<grammar xmlns:ega="http://examplotron.org/annotations/"
   xmlns="http://relaxng.org/ns/structure/1.0"
   xmlns:sch="http://www.ascc.net/xml/schematron"
   datatypeLibrary="http://www.w3.org/2001/XMLSchema-datatypes">
  <start>
    <element name="osoba">
      <oneOrMore>
        <element name="imię">
          <text><ega:example>Petronela</ega:example>/text>
        </element>
      </oneOrMore>
    </element>
  </start>
</grammar>
```

ODD: One Document Does It All

Format pochodzący z Text Encoding Initiative — standardu zapisu danych humanistycznych i lingwistycznych:

- prace od 1987 r.,
- początkowo wersje SGML-owe, obecnie XML-owa,
- obecna wersja: P5 z 2007 r.,
- schemat zawiera moduły (zestawy znaczników) włączane w zależności od potrzeb.

Format ODD umożliwia zawarcie w jednym dokumencie:

- fragmentów schematu,
- dokumentacji w formacie zgodnym z typem dokumentów TEI

w sposób uniezależniający model od rodzaju schematu (DTD, XML Schema, Relax NG),

ODD TEI — przykład

```
<TEI xmlns="http://www.tei-c.org/ns/1.0">
  <teiHeader> ... </teiHeader>
  <text>
    <body>
      <schemaSpec ident="odd1" start="TEI">
        <moduleRef key="header"/>
        <moduleRef key="core"/>
        <moduleRef key="tei"/>
        <moduleRef key="textstructure"/>
      </schemaSpec>
    </body>
  ...
  </text>
</TEI>
```

ODD: operacje na modelu

```
<schemaSpec xmlns:r="http://relaxng.org/ns/structure/1.0"
            ident="oddex1.5" start="TEI"
            xml:base="examples/odd1.5.xml">
  <moduleRef key="header"/>
  <moduleRef key="core"/>
  <moduleRef key="tei"/>
  <moduleRef key="textstructure"/>
  <elementSpec ident="soundClip" mode="add">
    <classes>
      <memberOf key="model.pPart.data"/>
    </classes>
    <content>
      <r:text/>
    </content>
  </elementSpec>
</schemaSpec>
```

Case study z użycia TEI: NKJP

Narodowy Korpus Języka Polskiego (`http://nkjp.pl`):

- to największy korpus językowy dla polszczyzny,
- powstały w latach 2007-2011 w ramach projektu badawczo-rozwojowego MNiSW,
- stworzony przez konsorcjum w składzie:
  - Instytut Podstaw Informatyki PAN (koordynator),
  - Instytut Języka Polskiego PAN,
  - Wydawnictwo Naukowe PWN,
  - Zakład Językoznawstwa Komputerowego i Korpusowego Uniwersytetu Łódzkiego,
- zawiera 1B segmentów, z czego 250M to korpus zrównoważony, a 1M – anotowany ręcznie.

XML-owe warstwy anotacji

Idea: anotacja zewnętrzna (ang. *stand-off*).

Elementy opisu:

- nagłówek korpusu,
- nagłówek próbki,
- struktura tekstu,
- segmentacja (zdania, tokeny),
- morfoskładnia (zestaw analiz morfologicznych z lematami, ujednoznacznienie),
- słowa składniowe („po prostu"),
- grupy składniowe („upadek komunizmu"),
- nazwy własne,
- sensy słów.

Zachęcam do zajrzenia na stronę
`http://nlp.ipipan.waw.pl/TEI4NKJP/`.

Nagłówki korpusu (skrót)

```
<teiHeader>
  <fileDesc>
    <titleStmt>
      <title xml:lang="pl">Narodowy Korpus Języka
                           Polskiego</title>
      <title xml:lang="en">National Corpus
                           of Polish</title>
      <funder>Ministry of Science
              and Higher Education</funder>
      <respStmt>
        <persName>Adam Przepiórkowski</persName>
        <resp>project coordinator</resp>
      </respStmt>
    </titleStmt>
    ...
  </fileDesc>
</teiHeader>
```

Nagłówki korpusu (skrót)

```
<teiHeader>
  <fileDesc>
    <titleStmt> ... </titleStmt>
    <publicationStmt>
      <publisher>Institute of Computer Science,
                 Polish Academy of Sciences</publisher>
    </publicationStmt>
    <sourceDesc>
      <p>The origin of texts in NKJP may be
        <list type="bulleted">
          <item>the IPI PAN Corpus</item>
                ...
        </list>
      </p>
    </sourceDesc>
  </fileDesc>
</teiHeader>
```

Struktura tekstu i segmentacja

```
<text>
  <body>
    <p xml:id="txt_1-p">Ważny list? Tu go miałem.</p>
  </body>
</text>

<text xml:id="p-text1">
  <body>
    <p xml:id="segm_1-p" corresp="text.xml#txt_1-p">
      <s xml:id="segm_1.1-s">
        <!-- ... -->
        <!-- list -->
        <seg xml:id="segm_1.1.3-seg"
             corresp="text.xml#string-range(txt_1-p,8,4)"/>
        <!-- ... -->
  </body>
</text>
```

Morfoskładnia

```
<p xml:id="morph_1-p" corresp="ann_segmentation.xml#segm_1-p">
  <s xml:id="morph_1.1-s"
     corresp="ann_segmentation.xml#segm_1.1-s">
    <!-- ... -->
    <seg xml:id="morph_1.1.3-seg"
         corresp="ann_segmentation#segm_1.1.3-seg">
      <fs type="morph">
        <f name="orth">
          <string>list</string>
        </f>
        <f name="interps">
           <!-- ... -->
        </f>
        <f name="disamb">
           <!-- ... -->
        </f>
      </fs>
    </seg>
```

Interpretacje morfoskładniowe

```
<f name="interps">
  <fs type="lex" xml:id="morph_1.1.3.1-lex">
    <f name="base">
      <string>list</string>
    </f>
    <f name="ctag">
      <symbol value="subst"/>
    </f>
    <f name="msd">
      <vAlt>
        <symbol xml:id="morph_1.1.3.1.1-msd"
                value="sg:nom:m3"/>
        <symbol xml:id="morph_1.1.3.1.2-msd"
                value="sg:acc:m3"/>
      </vAlt>
    </f>
  </fs>
  <!-- ... -->
```

Interpretacje morfoskładniowe

```
<f name="interps">
  <fs type="lex" xml:id="morph_1.1.3.1-lex">
    <!-- ... -->
  </fs>
  <fs type="lex" xml:id="morph_1.1.3.2-lex">
    <f name="base">
      <string>lista</string>
    </f>
    <f name="ctag">
      <symbol value="subst"/>
    </f>
    <f name="msd">
      <symbol xml:id="morph_1.1.3.2.1-msd"
              value="pl:gen:f"/>
    </f>
  </fs>
</f>
```

Ujednoznacznienie morfoskładni

```
<f name="interps">
  <fs type="lex" xml:id="morph_1.1.3.1-lex">
    <f name="msd">
      <vAlt>
        <symbol xml:id="morph_1.1.3.1.1-msd"
                value="sg:nom:m3"/>
        <symbol xml:id="morph_1.1.3.1.2-msd"
                value="sg:acc:m3"/>
      </vAlt>
    </f>
  </fs>
  <!-- ... -->
</f>
<f name="disamb">
  <fs type="tool_report">
    <f fVal="#morph_1.1.3.1.2-msd" name="choice"/>
    <!-- ... -->
  </fs>
</f>
```

Paczkowy wariant formatu

Zaimplementowany na potrzeby Multiserwisu (`http://chopin.ipipan.waw.pl/multiservice/`):

- poziomy anotacji jako elementy `<TEI>` wewnątrz elementów `<teiCorpus>`,
- minimalne nagłówki (`<teiHeader>`) zawierające:
  - nazwę warstwy (`<titleStmt>/<title>`),
  - informację o narzędziu wykorzystanym do jej stworzenia (`<publicationStmt>/<distributor>`),
  - datę/czas przetwarzania (`<publicationStmt>/<date>`).

Co dalej?

Na następnych wykładach:

- 14 XI: Patryk Czarnik o tym, jak prezentować XML-a,
- 21 XI: Patryk Czarnik o standardach XPath/XQuery,
- 28 XI: Patryk Czarnik o XSLT,
- 5 XII: ja o XPointerze, XInclude i zastosowaniach XML-a (XML DocBook, SVG, MathML, OOXML vs. OpenDocument).