

XML Schema

Alternatywne metody definiowania struktury dokumentów

Patryk Czarnik

Instytut Informatyki UW

XML i nowoczesne technologie zarządzania treścią – 2007/08

Plan

- 1 XML Schema – uzupełnienia
 - Nazwy kwalifikowane i nie
 - Grupy zamienne
 - Symbole wieloznaczne
- 2 Alternatywy dla XML Schema
 - Ograniczenia XML Schema
 - Relax NG
 - Schematron
- 3 Zarządzanie zmianami struktury

Forma nazwy lokalnych elementów i atrybutów

- **Atrybut `form` w deklaracjach lokalnych:**
 - `qualified` – w dokumencie nazwa kwalifikowana,
 - `unqualified` – w dokumencie nazwa niekwalifikowana.
- **Atrybuty `elementDefaultForm` i `attributeDefaultForm` w elemencie głównym `schema`.**
- **Elementy i atrybuty zadeklarowane globalnie – zawsze kwalifikowane.**
- **Najbardziej intuicyjne: elementy kwalifikowane, atrybuty niekwalifikowane.**

Grupy zamienne (*substitution groups*)

Schemat

```
<xs:element name="katalog">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="produkt"
        maxOccurs="unbounded"/>
      <xs:sequence>
        <xs:complexType>
</xs:element>
<xs:element name="produkt" type="ProduktTyp"/>
<xs:element name="komputer" type="KomputerTyp"
  substitutionGroup="produkt"/>
<xs:element name="ubranie" type="UbranieTyp"
  substitutionGroup="produkt"/>
```

Grupy zamienne (*substitution groups*)

Egzemplarz

```
<katalog>
  <produkt>...</produkt>
  <komputer>...</komputer>
  <ubranie>...</ubranie>
  <produkt>...</produkt>
</katalog>
```

Symbole wieloznaczne (1)

- Symbole wieloznaczne dla **elementów** (*element wildcards*).
- Symbole wieloznaczne dla atrybutów (*attribute wildcards*).

Przykład

```
<xs:complexType name="OsobaTyp">
  <xs:sequence>
    <xs:element name="imie" type="xs:string"/>
    <xs:element name="nazwisko" type="xs:string"/>
    <xs:any namespace="##other" minOccurs="0"
      maxOccurs="unbounded"
      processContents="skip"/>
  </xs:sequence>
  <xs:anyAttribute namespace="##other"
    processContents="lax"/>
</xs:complexType>
```

Symbole wieloznaczne (1)

- Symbole wieloznaczne dla elementów (*element wildcards*).
- Symbole wieloznaczne dla **atrybutów** (*attribute wildcards*).

Przykład

```
<xs:complexType name="OsobaTyp">
  <xs:sequence>
    <xs:element name="imie" type="xs:string"/>
    <xs:element name="nazwisko" type="xs:string"/>
    <xs:any namespace="##other" minOccurs="0"
      maxOccurs="unbounded"
      processContents="skip"/>
  </xs:sequence>
  <xs:anyAttribute namespace="##other"
    processContents="lax"/>
</xs:complexType>
```

Symbole wieloznaczne (2)

Atrybut `namespace`

- rozdzielone białymi znakami **identyfikatory przestrzeni nazw** i/lub `##targetNamespace`,
- `##any` – dowolna przestrzeń nazw,
- `##other` – przestrzeń nazw inna niż docelowa.

Atrybut `processContents`

- `skip` – nie sprawdzaj poprawności strukturalnej,
- `strict` – sprawdzaj poprawność; błąd jeśli brak unikalnej deklaracji w schemacie (wraz z importami i includami),
- `lax` – sprawdzaj tylko elementy/atributy, dla których jest dostępna deklaracja.

Symbole wieloznaczne (2)

Atrybut `namespace`

- rozdzielone białymi znakami **identyfikatory przestrzeni nazw** i/lub `##targetNamespace`,
- `##any` – dowolna przestrzeń nazw,
- `##other` – przestrzeń nazw inna niż docelowa.

Atrybut `processContents`

- **skip** – nie sprawdzaj poprawności strukturalnej,
- **strict** – sprawdzaj poprawność; błąd jeśli brak unikalnej deklaracji w schemacie (wraz z importami i includami),
- **lax** – sprawdzaj tylko elementy/atributy, dla których jest dostępna deklaracja.

Typowe zastosowanie

```
<xs:element name="description">
  <xs:complexType mixed="true">
    <xs:sequence>
      <xs:any namespace="http://www.w3.org/1999/xhtml"
        minOccurs="0" maxOccurs="unbounded"
        processContents="skip"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Plan

- 1 XML Schema – uzupełnienia
 - Nazwy kwalifikowane i nie
 - Grupy zamienne
 - Symbole wieloznaczne
- 2 Alternatywy dla XML Schema
 - Ograniczenia XML Schema
 - Relax NG
 - Schematron
- 3 Zarządzanie zmianami struktury

Ograniczenia XML Schema

Brak sprawdzania kontekstowego

- Brak kontekstowego sprawdzania poprawności, np.:
 - zawartość elementu cena-netto jest mniejsza lub równa od zawartości elementu cena-brutto,
 - jeżeli wartością atrybutu sposób-transportu jest powietrze, to element środek-transportu ma zawartość samolot lub balon.
- Metody kontekstowego sprawdzania poprawności:
 - zaprogramowane w kodzie aplikacji, transformacja XSLT,
 - wykorzystanie innego języka schematów, np. Schematron.

Niedeterministyczne modele zawartości

- **Model niejednoznaczny:**

Nawet znając cały dokument nie jesteśmy w stanie wskazać które deklaracje w schemacie odpowiadają elementom w dokumencie.

$$(a, a) * \mid (a, a, a) *$$

- **Model niedeterministyczny:**

Czytając dokument od początku gdy pojawia się pewien znacznik otwierający nie jesteśmy w stanie wskazać która deklaracja w schemacie odpowiada bieżącemu elementowi w dokumencie (bez czytania dalej dokumentu).

$$(a, a?, a?) \quad (a, b) +, a?$$

- **Modele definiowane w XML Schema muszą być deterministyczne:**

Czytając dowolny dokument od początku gdy pojawia się pewien znacznik otwierający jesteśmy w stanie wskazać która deklaracja w schemacie odpowiada bieżącemu elementowi w dokumencie (bez czytania dalej dokumentu).

$$(a, (a, a?) ?)$$

Rozgadana składnia XML Schema

„Rozgadana” składnia, długie nazwy – utrudniona ręczna edycja.

Element z prostą zawartością i atrybutem

```
<xs:element name="masa">
  <xs:complexType>
    <xs:simpleContent>
      <xs:extension base="xs:decimal">
        <xs:attribute name="jm" type="xs:string"/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
</xs:element>
```

Relax NG

- REgular LAnguage description for XML – New Generation:
 - składnia XML-owa, „bliska opisowi struktury w języku naturalnym”,
 - wspiera typy danych (np. XML Schema Datatypes),
 - atrybuty opisywane (prawie) tak samo, jak elementy,
 - prosta technika modularyzacji: define / ref, model przetwarzania oparty na wyrażeniach regularnych.
- RELAX NG a DTD:
 - elementy wymagane, ale bez określonego porządku,
 - model mieszany – więcej możliwości;
- RELAX NG a XML Schema:
 - niedeterministyczne modele zawartości,
 - modele zawartości wrażliwe na kontekst,
 - zwięzła składnia.

Relax NG – przykłady (1)

DTD

```
<!DOCTYPE addressBook [  
  <!ELEMENT addressBook (card*)>  
  <!ELEMENT card (name, email)>  
  <!ELEMENT name (#PCDATA)>  
  <!ELEMENT email (#PCDATA)>  
>
```

Relax NG

```
<element name="addressBook"  
  xmlns="http://relaxng.org/ns/structure/1.0">  
  <zeroOrMore>  
    <element name="card">  
      <element name="name"> <text/> </element>  
      <element name="email"> <text/> </element>  
    </element>  
  </zeroOrMore>  
</element>
```

Relax NG – przykłady (2)

Wybór między typem prostym a złożonym – nie ma w XML Schema

```
<element name="name">
  <choice>
    <text/>
    <group>
      <element name="firstName"><data type="token"/>
    </element>
    <optional>
      <element name="middleName"><data type="token"/>
    </element>
    </optional>
    <element name="lastName"><data type="token"/>
    </element>
  </group>
</choice>
</name>
```

Źródło: RELAX NG Tutorial, <http://www.relaxng.org/tutorial-20011203.html>

Relax NG – przykłady (2)

Niedeterminizm – nie da się zapisać w XML Schema

Model w notacji DTD

$(a, b)^*, a?$

Relax NG

```
<element name="listaAB">
  <zeroOrMore>
    <ref name="a"/>
    <ref name="b"/>
  </zeroOrMore>
  <optional>
    <ref name="a"/>
  </optional>
</element>
```

Schematron

- Język oparty na własnościach (asercjach), a nie na gramatyce:
 - łatwe wyrażanie reguł walidacji kontekstowej,
 - trudne, nieintuicyjne modelowanie struktury dokumentu,
 - użyteczny w połączeniu ze zwykłą DTD lub schematem XML Schema.
- Status: norma ISO (ISO/IEC 19757-3:2006).
- Implementacja referencyjna:
 - przekształcenie (generator) XSLT,
 - dla zadanego schematu Schematronowego, generuje XSLT sprawdzający poprawność dokumentów.
- Dostępnych kilkanaście implementacji.

Schematron

- Własności ewaluowane w kontekście konkretnego węzła dokumentu:
 - `assert` – własność, która musi być spełniona,
 - `report` – własność, której spełnienie oznacza błąd.
- Określanie kontekstu i własności: wyrażenia XPath.

Przykład

```
<rule context="towar">
  <assert test="@wartosc = @cena * @liczba">
    wartość = cena * liczba</assert>
</rule>

<rule context="faktura">
  <report test="@platnosc != 'przelew' and ./przelew">
    Przelew określony, a nie płacimy przelewem
  </report>
</rule>
```

Schematron + XML Schema (1)

```
<xs:schema
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  targetNamespace="http://www.demo.org"
  xmlns="http://www.demo.org"
  xmlns:sch="http://www.ascc.net/xml/schematron"
  elementFormDefault="qualified">
  <xs:annotation>
    <xs:appinfo>
      <sch:title>Schematron validation</sch:title>
      <sch:ns prefix="d" uri="http://www.demo.org"/>
    </xs:appinfo>
  </xs:annotation>
  ...
```

Schematron + XML Schema (1)

```
<xs:element name="demo">
  <xs:annotation>
    <xs:appinfo>
      <sch:pattern name="Check A greater than B">
        <sch:rule context="d:Demo">
          <sch:assert test="d:A > d:B"> A should be
            greater than B. </sch:assert>
        </sch:rule>
      </sch:pattern>
    </xs:appinfo>
  </xs:annotation>
  <xs:complexType>
    <xs:sequence>
      <xs:element name="A" type="xs:integer"/>
      <xs:element name="B" type="xs:integer"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
</xs:schema>
```

Plan

- 1 XML Schema – uzupełnienia
 - Nazwy kwalifikowane i nie
 - Grupy zamienne
 - Symbole wieloznaczne
- 2 Alternatywy dla XML Schema
 - Ograniczenia XML Schema
 - Relax NG
 - Schematron
- 3 Zarządzanie zmianami struktury

Zarządzanie zmianami struktury

Za Szymonem Zióło

- Zmiany niekompatybilne wstecz – przykład:
 - dodanie elementu wymaganego.
- Sposób postępowania w „żywym” systemie:
 - 1 wprowadzamy zmianę modelu kompatybilną wstecz (np. dodajemy element, ale opcjonalny),
 - 2 migrujemy dokumenty:
 - przekształcamy automatycznie i/lub
 - instruujemy użytkowników o konieczności migracji do nowej struktury,
 - 3 po dodaniu brakujących elementów (lub po upływie wyznaczonego czasu) – wprowadzenie zmiany docelowej.
- Większe zmiany modelu:
 - deklarujemy osobny element z nowym modelem i przez pewien czas dopuszczamy stary lub nowy model,
 - stosujemy przez pewien czas równolegle dwie wersje.

Zmiany struktury a aplikacje

- Typowa zależność między treścią programu a strukturą danych:
 - w treści programu zakładamy konkretną postać struktur danych,
 - dane wejściowe/wyjściowe – możliwa zmiana struktury w czasie,
 - zmiana struktury danych – konieczność zmian w kodzie.
- Tworzenie aplikacji niezależnych od zmian struktury danych:
 - ignorowanie elementów i atrybutów nie znaczących dla aplikacji,
 - unikanie nadmiernej zależności od struktury dokumentu,
 - parametryzowanie aplikacji dodatkowymi informacjami umieszczonymi w schemacie, np.:

Przykład

```
<xs:element name="NIP">
  ...
  <xs:attribute name="opis" type="xs:string"
    fixed="Numer Identyfikacji Podatkowej"/>
  ...
```

Przestrzenie nazw a aplikacje niezal. od struktury (1)

- Przykład – XLink:

- linki w elementach o dowolnych nazwach,
- typ linku i jego parametry przekazywane przez specjalne atrybuty.

Przykład użycia

```
<osoba xmlns:xlink="http://www.w3.org/1999/xlink">  
  <nazwisko>Kopernik, Mikołaj</nazwisko>  
  <biogram>Wybitny polski astronom, urodzony w  
    <geogr xlink:type="simple" xlink:href="Torun.xml">  
      Toruniu</geogr>.</biogram>  
</osoba>
```

Przestrzenie nazw a aplikacje niezal. od struktury (2)

- Przykład – aplikacja weryfikująca numery PESEL i daty urodzenia w dokumencie XML:
 - nie powinna zależeć od struktury dokumentu wejściowego,
 - jak „przekazać” aplikacji, co ma zweryfikować?

Realizacja

```
<podanie xmlns:pv="http://PeselValidator.pl">
<nadawca nr-ewid="60101012345" pv:PESEL="nr-ewid">
  <nazwisko>Zenon Niemrawy</nazwisko>
  <urodzony pv:data-ur="#CONTENT">1960-10-10
</urodzony>
</nadawca>
...
</podanie>
```