Modelli di progettazione di schemi XML

Ombretta Gaggi
Università di Padova

Modelli di progettazione

- Le strutture di XML schema possono essere combinate liberamente per ottenere l'effetto che si vuole, ma esistono alcuni modelli di progettazione che garantiscono risultati più eleganti, flessibili, riutilizzabili.
  - elementi locali vs globali
  - namespace
  - attributo elementFormDefault
  - modularizzazione del documento
  - etc

Ricapitoliamo: elementi globali e locali

- Un elemento è globale se è definito all'interno dell'elemento <schema>, nel qual caso può essere l'elemento radice.
- Un elemento è locale se è definito all'interno di un tipo. In questo caso possono esistere elementi omonimi in luoghi diversi che hanno definizioni diverse.

Elementi globali e locali: un esempio

```
<xsd:schema xmlns… >
 <xsd:element name="libro">
  <xsd:complexType>
   <xsd:sequence>
    <xsd:element name="titolo" type="xsd:string"/>
    <xsd:element name="autore" type="xsd:string"/>
    <xsd:element name="editore" type="xsd:string"/>
   </xsd:sequence>
  </xsd:complexType>
 </xsd:element>
</xsd:schema>
```

globale

locale

Ricapitoliamo: l'attributo elementFormDefault

- Regola la qualificazione degli elementi locali ad un tipo. Se l'attributo è unqualified (default), gli elementi locali non debbono essere qualificati esplicitamente (con prefissi), ma ereditano il namespace dall'elemento che li contiene.
- Se l'attributo è qualified, anche gli elementi locali debbono essere esplicitamente qualificati attraverso l'uso di prefissi.
- Nota Bene: selezionare elementFormDefault a unqualified impedisce l'utilizzo di namespace di default (perché gli elementi locali debbono rimanere non qualificati).

XSD e XML

```
<libro xmlns="http://www.libri.com"
    xmlns:xs="http://www.w3.org/2001/XMLSchema-instance"
    xs:schemaLocation="http://www.libri.com libro.xsd">
        <titolo>Querying XML</titolo>
        <autore>J. Melton</autore>
        <editore>Morgan Kaufmann</editore>
    </libro>
```

```
<xsd:schema xmlns…
        targetNamespace="http://www.libri.com ">
  <xsd:element name="libro">
    <xsd:complexType> <xsd:sequence>
        <xsd:element name="titolo" .../>
        <xsd:element name="autore" …/>
        <xsd:element name="editore" …/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```


XSD e XML

```
<l:libro xmlns:l="http://www.libri.com"
    xmlns:xs="http://www.w3.org/2001/XMLSchema-instance"
    xs:schemaLocation="http://www.libri.com libro.xsd">
        <titolo>Querying XML</titolo>
        <autore>J. Melton</autore>
        <editore>Morgan Kaufmann</editore>
    </l:libro>
```

```
<xsd:schema xmlns…
        targetNamespace="http://www.libri.com ">
  <xsd:element name="libro">
    <xsd:complexType> <xsd:sequence>
        <xsd:element name="titolo" .../>
        <xsd:element name="autore" …/>
        <xsd:element name="editore" …/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```


…oppure…

```
<libro xmlns="http://www.libri.com"
    xmlns:xs="http://www.w3.org/2001/XMLSchema-instance"
    xs:schemaLocation="http://www.libri.com libro.xsd">
        <titolo>Querying XML</titolo>
        <autore>J. Melton</autore>
        <editore>Morgan Kaufmann</editore>
    </libro>
```

```
<xsd:schema xmlns…
        targetNamespace="http://www.libri.com "
        elementFormDefault="qualified">
  <xsd:element name="libro">
    <xsd:complexType> <xsd:sequence>
        <xsd:element name="titolo" .../>
        <xsd:element name="autore" …/>
        <xsd:element name="editore" …/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```


Una prima domanda

- Debbo definire elementi locali o globali?
- Tre variabili in gioco:
  - Modularizzazione (e dunque riuso e flessibilità)
  - Stili di qualificazione degli elementi
  - Tipi anonimi o denominati
- Quattro proposte di modelli di schemi:
  - Il modello **Bambole Russe** (Russian Dolls)
  - Il modello **Fette di Salame** (Salami Slices)
  - Il modello **Giardino dell'Eden** (Garden of Eden)
  - Il modello **Tende alla Veneziana** (Venetian Blinds)

Il modello Bambole Russe

- È caratterizzato dal possedere un unico elemento globale, l'elemento radice, con un tipo anonimo che contiene tutti gli elementi di secondo livello, ciascuno con un proprio tipo anonimo che contiene elementi di terzo livello, e così via a gerarchia sempre più annidata.
  - Tutti i tipi sono anonimi
  - Tutti gli elementi sono locali

Esempio: documento

```
<libro xmlns=…>
   <fronte>
      <titolo>I promessi sposi</titolo>
      <autore>Alessandro Manzoni</autore>
      <editore>Mondadori</editore>
   </fronte>
   <corpo>
      <capitolo> <titolo>Capitolo 1</titolo>
         <para> testo testo testo testo <b>testo</b>
                testo testo <i>testo</i> testo testo
         </para>
         <para> testo testo testo testo <b>testo</b>
                testo testo <i>testo</i> testo testo
         </para>
      </capitolo>
      …
   </corpo></libro>
```


Esempio: schema - 1

```
<xsd:element name="libro">
   <xsd:complexType><xsd:sequence>
         <xsd:element name="fronte">
            <xsd:complexType><xsd:sequence>
               <xsd:element name="titolo" type="xsd:string"/>
               <xsd:element name="autore" type="xsd:string"/>
               <xsd:element name="editore" type="xsd:string"/>
            </xsd:sequence></xsd:complexType>
         </xsd:element>
         <xsd:element name="corpo">
                  …
         </xsd:element>
   </xsd:sequence></xsd:complexType>
</xsd:element>
```


Esempio: schema - 2

```
<xsd:element name="corpo">
   <xsd:complexType><xsd:sequence>
    <xsd:element name="capitolo" maxOccurs="unbounded">
       <xsd:complexType><xsd:sequence>
          <xsd:element name="titolo" type="xsd:string"/>
          <xsd:element name="para" maxOccurs="unbounded">
             <xsd:complexType mixed="true">
                <xsd:choice maxOccurs="unbounded">
                   <xsd:element name="b" type="xsd:string"/>
                   <xsd:element name="i" type="xsd:string"/>
                </xsd:choice>
             </xsd:complexType>
          </xsd:element>
       </xsd:sequence></xsd:complexType>
    </xsd:element>
  </xsd:sequence></xsd:complexType>
</xsd:element>
```


Vantaggi e svantaggi

- Pregi
  - Rispecchia la struttura del documento XML
  - Nasconde all'esterno la complessità interna
  - Ambito locale (ogni definizione di elemento è locale ad un elemento contenitore)
  - Compatto (tutto insieme) e coeso (le definizioni sono strettamente collegate tra loro)
  - Facile qualificabilità (posso attivare la qualificazione degli elementi locali con un solo gesto)
- Difetti
  - Contenuto opaco (gli elementi sono nascosti all'esterno)
  - Non è possibile riutilizzare le strutture precedentemente definite

Il modello Fette di Salame

- È caratterizzato dal mettere come globali tutti gli elementi, e dall'usare sistematicamente i ref per definire gli elementi.
- Tutti i tipi sono anonimi e inseriti all'interno della definizione dell'elemento.
- Ogni elemento e attributo fa storia a sé, come tante fette di salame mangiate indipendentemente le une dalle altre.

Esempio - 1

```
<xsd:schema xmlns... >
<xsd:element name="libro">
   <xsd:complexType><xsd:sequence>
       <xsd:element ref="fronte"/>
       <xsd:element ref="corpo"/>
   </xsd:sequence></xsd:complexType>
</xsd:element>
<xsd:element name="fronte">
  <xsd:complexType><xsd:sequence>
    <xsd:element ref="titolo"/>
    <xsd:element ref="autore"/>
    <xsd:element ref="editore"/>
   </xsd:sequence></xsd:complexType>
</xsd:element>
...
```


Esempio - 2

```
…
<xsd:element name="corpo">
   <xsd:complexType><xsd:sequence>
      <xsd:element ref="capitolo" maxOccurs="unbounded"/>
   </xsd:sequence></xsd:complexType>
</xsd:element>
<xsd:element name="titolo" type="xsd:string"/>
<xsd:element name="autore" type="xsd:string"/>
<xsd:element name="editore" type="xsd:string"/>
<xsd:element name="capitolo">
   <xsd:complexType><xsd:sequence>
      <xsd:element ref="titolo"/>
      <xsd:element ref="para" maxOccurs="unbounded"/>
   </xsd:sequence></xsd:complexType>
</xsd:element>
```


Esempio - 3

```
…
<xsd:element name="para">
   <xsd:complexType mixed="true"><xsd:choice>
      <xsd:element ref="b" maxOccurs="unbounded"/>
      <xsd:element ref="i" maxOccurs="unbounded"/>
   </xsd:choice></xsd:complexType>
</xsd:element>
<xsd:element name="b" type="xsd:string"/>
<xsd:element name="i" type="xsd:string"/>
</xsd:schema>
```


Vantaggi e svantaggi

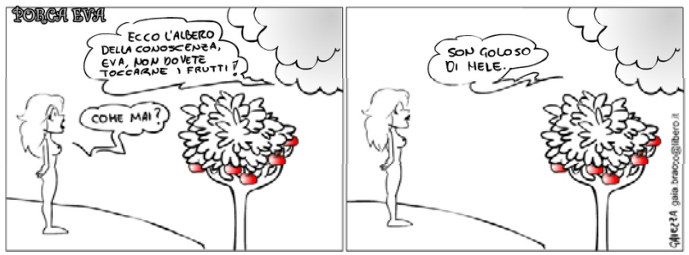
- Pregi
  - Contenuto trasparente (ogni singolo componente del documento è disponibile all'esterno e può essere riusato).
  - Facile conversione in DTD
  - Verboso e coeso
- Difetti
  - Ambito globale (indipendentemente dal modello di qualificazione scelto): tutti gli elementi dovranno essere qualificati perché sono tutti globali.

Il modello Giardino dell'Eden

"Allora il Signore Dio plasmò dal suolo ogni sorta di bestie selvatiche e tutti gli uccelli del cielo e li condusse all'uomo, per vedere come li avrebbe chiamati: in qualunque modo l'uomo avesse chiamato ognuno degli esseri viventi, quello doveva essere il suo nome."

Genesi, 2:19

- È caratterizzato dal mettere come globali tutti gli elementi, e dall'usare sistematicamente tipi denominati.
- Ogni struttura è rigorosamente dotata di nome, sia i tipi che gli elementi e gli attributi.



Esempio - 1

```
<xsd:schema xmlns… >
    <xsd:element name="libro" type="Tlibro"/>
    <xsd:element name="fronte" type="Tfronte"/>
    <xsd:element name="corpo" type="Tcorpo">
    <xsd:element name="titolo" type="xsd:string"/>
    <xsd:element name="autore" type="xsd:string"/>
    <xsd:element name="editore" type="xsd:string"/>
    <xsd:element name="capitolo" type="Tcapitolo"/>
    <xsd:element name="para" type="Tpara">
    <xsd:element name="b" type="xsd:string"/>
    <xsd:element name="i" type="xsd:string"/>
    <xsd:complexType name="Tlibro"><xsd:sequence>
        <xsd:element ref="fronte"/>
        <xsd:element ref="corpo"/>
    </xsd:sequence></xsd:complexType>
    …
```


Esempio - 2

```
… <xsd:complexType name="Tfronte"><xsd:sequence>
        <xsd:element ref="titolo"/>
        <xsd:element ref="autore"/>
        <xsd:element ref="editore"/>
    </xsd:sequence></xsd:complexType>
    <xsd:complexType name="Tcorpo"><xsd:sequence>
        <xsd:element ref="capitolo" maxOccurs="unbounded"/>
    </xsd:sequence></xsd:complexType>
    <xsd:complexType name="Tcapitolo"><xsd:sequence>
            <xsd:element ref="titolo"/>
            <xsd:element ref="para" maxOccurs="unbounded"/>
    </xsd:sequence></xsd:complexType>
    <xsd:complexType name="Tpara" mixed="true">
        <xsd:choice maxOccurs="unbounded">
            <xsd:element ref="b" /><xsd:element ref="i"/>
        </xsd:choice></xsd:complexType>
</xsd:schema>
```


Vantaggi e svantaggi

- Pregi (come per le fette di salame)
  - Contenuto trasparente (ogni singolo componente del documento è disponibile all'esterno e può essere riusato).
  - Facile conversione in DTD
  - Verboso e coeso
- Difetti (a maggior ragione rispetto alle fette di salame)
  - Ambito globale (indipendentemente dal modello di qualificazione scelto) tutti gli elementi dovranno essere qualificati perché sono tutti globali.
  - Poco leggibile

Ricapitolando

- Il modello Bambole Russe
  - Facilita la gestione flessibile dei namespace
  - Non facilita il riuso attraverso la modularità
- Il modello Giardino dell'Eden e Fette di Salame
  - Facilita il riuso e la modularità
  - Impedisce la gestione flessibile dei namespace
- Domanda:
  - Esiste allora un modello che permette sia il riuso che la gestione flessibile dei namespace?

Il modello Tende alla Veneziana

- È caratterizzato dal possedere un unico elemento globale, l'elemento radice, e tanti tipi nominali ciascuno dei quali contiene definizioni di elementi locali associati a tipi nominali ed esterni.
  - Tutti i tipi sono nominali
  - Tutti gli elementi sono locali

Esempio - 1

```
<xsd:element name="libro" type="Tlibro"/>
<xsd:complexType name="Tlibro"><xsd:sequence>
    <xsd:element name="fronte" type="Tfronte"/>
    <xsd:element name="corpo" type="Tcorpo"/>
</xsd:sequence></xsd:complexType>
 <xsd:complexType name="Tfronte"><xsd:sequence>
    <xsd:element name="titolo" type="xsd:string"/>
    <xsd:element name="autore" type="xsd:string"/>
    <xsd:element name="editore" type="xsd:string"/>
</xsd:sequence></xsd:complexType>
<xsd:complexType name="Tcorpo"><xsd:sequence>
    <xsd:element name="capitolo" type="Tcap"
                                  maxOccurs="unbounded"/>
</xsd:sequence></xsd:complexType>
```


Esempio - 2

```
<xsd:complexType name="Tcap"><xsd:sequence>
    <xsd:element name="titolo" type="xsd:string"/>
    <xsd:element name="para" type ="Tpara" maxOccurs="unbounded"/>
</xsd:sequence></xsd:complexType>
<xsd:complexType name="Tpara" mixed="true"><xsd:choice>
    <xsd:element name="b" type="xsd:string" maxOccurs="unbounded"/>
    <xsd:element name="i" type="xsd:string" maxOccurs="unbounded"/>
</xsd:choice></xsd:complexType>
```


Vantaggi e svantaggi

- Pregi
  - Permette il riuso di componenti (i tipi complessi)
  - Permette una specifica flessibile dei namespace
  - Permette un rapido passaggio da un modello di namespacing all'altro semplicemente cambiando il valore di elementFormDefault.
- Il modello Tende alla Veneziana risolve i problemi connessi con gli altri modelli, permettendo sia il riuso dei componenti sia la gestione dell'elementFormDefault.

Obiezioni a XML Schema

- Sebbene XML Schema goda di un grande successo nel mondo W3C e nel mondo del data interchange, non si può dire che sia amato nel mondo dei document designer, e in tutti quelli che trattano con documenti fatti per essere letti da esseri umani.
- XML Schema è ritenuto troppo complesso, troppo verboso, e troppo interessato ad accontentare un po' tutti, piuttosto che molto alcuni.
- Per questo motivo, sotto il cappello dell'ISO, è nato il working group ISO/IEC 19757 - DSDL (Document Schema Definition Languages) che vuole creare uno scheletro di interazione tra diversi linguaggi di schema per permettere vari tipi di validazione diversi.

Altri linguaggi

- All'interno dell'attuale standard sono confluiti due importanti alternative a XML Schema:
  - Relax NG un linguaggio basato sui grammatica generativa (come DTD e XML Schema) ma più semplice e meno verboso
  - Schematron, un linguaggio basato su regole di correttezza locale, non paragonabile ad altri linguaggi di validazione.
  - DSD, *Document Structure Definition*, è un linguaggio pensato per i non esperti di XML, che permette una maggiore flessibilità rispetto a XMLSchema

Relax NG

- Un linguaggio derivato dalla fusione di due linguaggi di schema precedenti, Relax (di Makoto Murata) e Trex (di Jim Clark).
- Basato su XML, ignora molte delle complessità di XML Schema, inclusa una grammatica tutto sommato più compatta.
- In particolare, non gestisce aspetti connessi con il contenuto (come valori di default, fixed, nullable, key e keyref, ecc.) ma solo sulla struttura.

Un esempio di Relax NG

```
<!DOCTYPE addr [
    <!ELEMENT addr (item*)>                                  (item)+
    <!ELEMENT item (name, email)>                            (name, email?)
    <!ELEMENT name (#PCDATA)><!ELEMENT email (#PCDATA)>
]>
sarebbe, in Relax NG:
<grammar xmlns="http://relaxng.org/ns/structure/1.0">
  <start><element name="addr">
          <zeroOrMore>                                       <oneOrMore>
            <element name="item">
               <element name="name"><text/></element>
                                                             <optional>
                <element name="email"><text/></element>
                                                             </optional>
            </element>
          </zeroOrMore>                                      </oneOrMore>
    </element></start>
</grammar>
```


Un esempio di DSD

```
<ElementDef ID="book">
    <AttributeDecl Name="isbn" Optional="yes">
          <StringType IDRef="isbn"/>
    </AttributeDecl>
    <Sequence>
    <If> <Attribute Name="isbn"/>
      <Then>
          <Optional> <Element IDRef="book-title"/> </Optional>
      </Then>
     <Else> <Element IDRef="book-title"/></Else>
    </If>
    <OneOrMore><Element IDRef="author"/></OneOrMore>
    <Element IDRef="publisher"/>
    <Element Name="year">
          <StringType IDRef="digits"/>
    </Element>
</ElementDef>
```