SGML \ XML

Fabio Vitali

Introduzione

Qui esaminiamo in breve tutti gli aspetti di SGML che non sono transitati in XML:

- Dichiarazioni multiple
- Inclusioni ed esclusioni
- Meccanismi di minimizzazione
- Il problema del white space
- La dichiarazione SGML

Caratteristiche solo di SGML

Parliamo qui delle caratteristiche dei DTD di SGML che non è possibile utilizzare in XML.

DTD SGML che usano queste caratteristiche non sono facilmente trasportabili in XML.

Quando parleremo di XML, citeremo anche le caratteristiche specifiche di XML non presenti (naturalmente) in SGML

Dichiarazione multipla di elementi

Elementi con lo stesso content model possono essere definiti in una sola dichiarazione.

La dichiarazione

```
<!ELEMENT (sezione | blocco | capitolo) - - (para)+>
```

è equivalente a:

```
<!ELEMENT sezione - - (para)+ >
<!ELEMENT blocco - - (para)+ >
<!ELEMENT capitolo - - (para)+ >
```

```
<!ENTITY % para_type “(para)+” >
<!ELEMENT sezione - - %para_type; >
<!ELEMENT blocco - - %para_type; >
<!ELEMENT capitolo - - %para_type; >
```


Dichiarazione multipla di attributi

Attributi comuni a più elementi possono essere specificati in una sola dichiarazione.

La dichiarazione

```
<!ATTLIST (sezione | blocco | capitolo)
                    id  IDREF   #REQUIRED>
```

è equivalente a:

```
<!ATTLIST sezione   id  IDREF   #REQUIRED>
<!ATTLIST blocco    id  IDREF   #REQUIRED>
<!ATTLIST capitolo  id  IDREF   #REQUIRED>
```


Inclusioni

In SGML, ma non in XML, è possibile specificare anche delle eccezioni al content model utilizzando le dichiarazioni di inclusione ed esclusione.

La dichiarazione

```
<!ELEMENT sezione - - (titolo, para+) +(nota)>
```

significa che l'elemento nota può apparire sia all'interno di sezione, che all'interno di titolo e para.

Uso tipico:

```
<!ENTITY % content "(#PCDATA | bold | it)*">

<!ELEMENT para_normale  %content; >
<!ELEMENT para_speciale %content; +(link)>
```

Le inclusioni possono essere espresse in maniera equivalente espandendo il content model e le eventuali entità parametriche usate

Esclusioni

La dichiarazione

```
<!ELEMENT section - - (para+) -(index)>
```

indica che dentro a section sono ammessi dei paragrafi, ma mai degli elementi indice, anche se eventualmente l'elemento para potesse contenerli.

Uso tipico:

```
<!ENTITY % content "(#PCDATA | bold | it)*">

  <!ELEMENT bold  - - %content; -(bold)>
  <!ELEMENT it    - - %content; -(it)>
```

Le esclusioni non possono in generale essere espresse in maniera equivalente espandendo il content model, perché in realtà possono influenzare anche il content model degli elementi contenuti.

Declared content

Il content model degli elementi di un DTD contiene normalmente o altri elementi o #PCDATA (parsed character data), cioè tutte le entità ed i caratteri.

Oltre a questi, SGML ammette anche:

- **CDATA** (character data): tutti i caratteri ma non le entità.
- **RCDATA** (Replaceable character data): caratteri ed entità ma non altri sotto-elementi, anche se ammessi dalla definizione di inclusioni di livello superiore.

Mixed content

L'uso di #PCDATA insieme ad altri elementi come parte della definizione di un elemento può essere pericoloso e non è raccomandato in altra forma che la seguente:

```
<!ELEMENT a - - (#PCDATA | b | c)*>
```

cioè in prima posizione all'interno di un gruppo opzionale e ripetibile.

SGML invece ammette (causando infiniti problemi) anche forme del tipo:

```
<!ELEMENT a - - (b?, #PCDATA)>
```

È tuttavia sconsigliabile usarle anche in SGML.

Contenuto non ordinato

Oltre agli operatori ‘,’ (sequenza) e ‘|’ (alternativa), SGML permette anche l’operatore ‘&’ che indica obbligo di presenza in qualunque ordine.

La dichiarazione

```
<!ELEMENT a - - (b & c & d)>
```

significa che l’elemento a deve contenere esattamente un elemento b, un elemento c, ed un elemento d, ma che questi possono apparire in qualunque ordine.

Sono quindi leciti:

- **<a> … <c> … </c> <d> … </d> **
- **<a> <c> … </c> <d> … </d> … **
- **<a> <d> … </d> <c> … </c> … **

Valori di default degli attributi

Oltre ai valori espliciti e ai valori `#FIXED`, `#IMPLIED` e `#REQUIRED`, SGML permette due ulteriori tipi di valori:

- **`#CURRENT`**: indica il valore specificato nell'ultima istanza dell'attributo. È necessario specificare un valore nella prima istanza dell'attributo, e poi salvo cambiamenti tutti gli elementi prenderanno quel valore.
- **`#CONREF`**: Il valore dell'attributo va a sostituire il contenuto dell'elemento: l'elemento può o avere un contenuto, o quest'attributo, ma non entrambi.

La minimizzazione (1)

SGML nasce nei primi anni 80 specialmente per la codifica di documentazione tecnica e commerciale: manuali, contratti, ecc.

SGML è innanzitutto pensato per essere inserito a mano con editor generici a carattere. Questo è dovuto a vari motivi:

- La diffusione di editor a carattere
- La abitudine ad inserire a mano codici di formattazione e di stampa (vedi LaTeX, nroff/troff, ma anche WordStar,
- La mancanza di applicazioni specifiche per SGML

I progettisti di SGML, dunque, idearono alcuni tipi di minimizzazione dei caratteri di markup da inserire, al fine di diminuire il numero di keystroke degli addetti all'inserimento.

La minimizzazione (2)

Tutti i tipi di minimizzazione sono facoltativi, ma possono avere effetti sulla sintassi.

Ci sono tre tipi di minimizzazione:

- Omitted tag minimization
- Short reference minimization
- Short tag minimization
  - Minimizzazione delle virgolette
  - Minimizzazione dei nomi degli attributi
  - End-tag vuoto
  - Minimizzazione dei caratteri di tag
  - Start-tag vuoto
  - Accorpabilità di tag adiacenti

Omitted tag minimization

E’ possibile omettere i tag di chiusura (e più raramente, di apertura), specificandone la facoltatitività nel DTD, purché la sintassi rimanga non ambigua.

```
<!ELEMENT  sezione - O (titolo, para*, sezione*)>
<!ELEMENT  titolo  O O (#PCDATA) >
<!ELEMENT  para    - O (#PCDATA)>


<sezione>Questo e’ il titolo della sezione
   <para>Primo paragrafo di questa sezione
   <sezione>Questo è il titolo della sottosezione
          <para>Primo paragrafo della sottosezione
   </sezione>
</sezione>
```


Short reference minimization

E’ possibile utilizzare caratteri del contenuto come se fossero elementi di markup. Per esempio, è possibile definire le virgolette come se fossero tag di definizione di elementi, ad es. <citazione>.

```
<para>Alice nel Paese delle Meraviglie pensò: “A cosa servono i libri se non hanno le figure?”</para>
```

può essere reso equivalente a

```
<para>Alice nel Paese delle Meraviglie pensò:
<citazione>A cosa servono i libri se non hanno le figure?</citazione></para>
```


Short tag minimization (1)

Minimizzazione delle virgolette

Se il valore dell'attributo è riconducibile ad un NAME (cioè è una parola sola), è possibile evitare le virgolette intorno al valore.

```
<H1 align="center">Intestazione </H1>
```

è equivalente a

```
<H1 align=center>Intestazione </H1>
```


Short tag minimization (2)

Minimizzazione dei nomi degli attributi

Se i valori leciti di un attributo sono definiti da una lista, allora è possibile omettere sia il nome dell'attributo che "="

```
<!ATTLIST doc stato (bozza|impaginato|finale) "bozza">
<doc stato="finale"> … </doc>
```

è equivalente a

```
<!ATTLIST doc stato (bozza|impaginato|finale) "bozza">
<doc finale> … </doc>
```

N.B.: Questo implica che le liste di valori debbono essere tutte diverse nello stesso DTD!

Short tag minimization (3 - 4)

Minimizzazione degli end-tag

E’ possibile usare un end-tag vuoto </> per rappresentare la chiusura dell’ultimo tag aperto.

```
<p>Il comando <command>ls</command> serve per… </p>
```

è equivalente a

```
<p>Il comando <command>ls</> serve per… </p>
```

Minimizzazione dei caratteri di tag

Nella stessa situazione descritta in precedenza, è possibile addirittura accorpare il contenuto dell’elemento al tag di apertura:

```
<p>Il comando <command/ls/ serve per… </p>
```


Short tag minimization (5 - 6)

Start-tag vuoto

E’ possibile usare uno start-tag vuoto <> per rappresentare o un elemento uguale al precedente, o l’elemento radice del documento.

```
<lista> <item> numero 1 <> numero 2 <> numero 3 </lista>
```

Accorpabilità di tag adiacenti

Quando due tag sono adiacenti, è possibile evitare il carattere di chiusura del tag:

```
<sezione<titolo> …

… </para</sezione>
```


Il white space

Il white space sono tutti quei caratteri senza aspetto visibile, che influenzano la formattazione del documento.

Essi includono sicuramente lo spazio, il tab, il carriage return ed il line feed, più eventuali altri caratteri.

SGML è nato in ambiente IBM, e le righe erano separate da caratteri chiamati RS e RE (record start e record end). Essi sono poi confluiti in ASCII 13 (CR) e ASCII 10 (LF).

Il white space nel markup

All’interno dei tag, il white space è equivalente ad un singolo spazio, e può essere usato per separare attributi ed altro.

```
<book issue=“3” date=“15/3/97”>
```

è equivalente a

```
<book
  issue   =“3”
  date    =“15/3/97”          >
```


Il white space nel contenuto (1)

Per leggibilità del documento SGML, gli autori possono inserire white space tra elementi e tra contenuto ed elementi.

```
<!ELEMENT autore (nome, cognome, e-mail)>
<!ELEMENT nome    #PCDATA>
<!ELEMENT cognome #PCDATA>
<!ELEMENT e-mail  #PCDATA>

<autore><nome>Fabio</nome><cognome>Vitali</cogno
me><e-mail>fabio@cs.unibo.it</e-mail></autore>
```

è equivalente a

```
<autore>
    <nome>Fabio</nome>
    <cognome>Vitali</cognome>
    <e-mail>fabio@cs.unibo.it</e-mail>
</autore>
```


Il white space nel contenuto (2)

Nell'esempio precedente, il parser, avendo visto dal DTD che il content model di autore non contiene #PCDATA, può dedurre che il white space non è parte del contenuto.

Tuttavia possono esservi problemi di ambiguità:

```
<para>
    Questo e' un paragrafo con degli
    <emph> spazi </emph>
    che non so come trattare.
</para>
```

Poiché para ha ovviamente un content model misto, non so più distinguere tra white space significativo e white space di presentazione.

Il white space nel contenuto (3)

L’operazione di rimuovere il white space presentazionale lasciando intatto il contenuto vero e proprio è detto ***normalizzazione*** del documento.

Il white space può essere:

- **Preservato**: tutti i caratteri di white space vengono passati intatti all’applicazione
- **Collassato**: tutti i caratteri di white space vengono ridotti ad uno e passati come **un singolo spazio**.
- **Rimosso**: tutti i caratteri di white space vengono cancellati non passati all’applicazione.

SGML e white space

SGML rimuove il white space nel markup (ovviamente) e negli elementi con content model di elemento (cioè senza #PCDATA)

SGML preserva il white space dentro agli elementi con content model misto (cioè con #PCDATA), tranne:

- Il carattere Record Start viene rimosso ovunque.
- Il carattere Record End viene rimosso ovunque tranne che in mezzo ad altro testo
- Il white space viene rimosso in qualunque riga contenga solo markup, commenti o PI
- In particolare, RE viene rimosso se segue immediatamente uno start tag o se precede immediatamente un end tag.

SGML declaration: a volo d'angelo (1)

- Nella dichiarazione SGML vengono specificati parametri generali sulla sintassi e il set dei caratteri utilizzabili nel documento SGML.
- La dichiarazione va necessariamente posta all'inizio del documento SGML, e non può essere citata tramite un'entità (perché ancora non sono definite). Alcuni programmi permettono di metterla in un file separato.
- In mancanza di una dichiarazione SGML esplicita, le applicazioni sono tenute a utilizzare una dichiarazione di default, chiamata ***reference concrete syntax*** (RCS).
- La dichiarazione SGML è composta di 6 parti: set di caratteri, capacità, ambito della sintassi concreta, sintassi concreta, caratteristiche, informazioni specifiche dell'applicazione.

SGML declaration: a volo d'angelo (2)

```
<!SGML "ISO 8879:1986"
CHARSET
   BASESET "ISO Registration
Number 100//CHARSET ECMA-94 Right
Part of Latin Alphabet Nr. 1//ESC
2/13 4/1"
   DESCSET
      0 9 UNUSED
      9 2 9
      11 2 UNUSED
      13 1 13
      14 18 UNUSED
      32 95 32
      127 1 UNUSED
      128 127 128
      255 1 UNUSED
```

CHARSET: Specifica il set di caratteri da usare.

- **BASESET** specifica il nome pubblico del set di caratteri
- **DESCSET** specifica come associare i codici del documento ai codici del set di caratteri.
  - **`Desc-char-num`**: il codice trovato nel file sorgente
  - **`Count`**: il numero di caratteri da considerare
  - **`Base-char-num`**: il codice corrispondente da associare (o il primo di questi codici)

SGML declaration: a volo d'angelo (3)

```
CAPACITY
        SGMLREF
        TOTALCAP        500000
        ATTCHCAP         70000
        ELEMCAP          35000

SCOPE  DOCUMENT
```

- CAPACITY: Specifico il numero massimo di occorrenze (in caratteri) delle varie caratteristiche di un documento SGML (numero di elementi, di attributi, ecc.). La dicitura SGMLREF fa riferimento a ciò che è specificato nella RCS.
- SCOPE: Permette di specificare se l'ambito della sintassi concreta posta di seguito sia DOCUMENT, cioè tutto il documento, DTD compreso, o se invece si riferisca soltanto all'istanza del documento (cioè solo il testo e il markup). In quest'ultimo caso si usa la RCS per il DTD.

SGML declaration: a volo d’angelo (4)

SYNTAX: Introduce i dettagli della sintassi concreta per il documento, ovvero quali delimitatori, caratteri speciali, e lunghezze ammettere. È divisa in 7 parti:

- SHUNCHAR
- BASESET
- FUNCTION
- DELIM
- NAMES
- QUANTITY

SGML declaration: a volo d'angelo (5)

```
SYNTAX
   SHUNCHAR  0 1 2 3 4 5 6 7 8 9
         10 11 12 13 14 15 16 17
         18 19 20 21 22 23 24 25
         26 27 28 29 30 31 127 255
   BASESET "ISO Registration
         Number 100//CHARSET ECMA-
         94 Right Part of Latin
         Alphabet Nr. 1//ESC 2/13
         4/1"
          DESCSET 0 256 0
   FUNCTION RE 13
            RS 10
            SPACE 32
            TAB SEPCHAR 9
```

- **SHUNCHAR**: quali caratteri sono proibiti nel markup (non nei dati!)
- **BASESET**: qual è il set dei caratteri usabili nel markup (non nei dati!). **DESCSET** permette di specificare i caratteri con codici speciali, come visto prima.
- **FUNCTION**: quali sono i codici dei caratteri con funzioni speciali, in particolare RS (record start), RE (record end) e SPACE (spazio).

SGML declaration: a volo d'angelo (6)

```
NAMING   LCNMSTRT ""
         UCNMSTRT ""
         LCNMCHAR ".-_:"
         UCNMCHAR ".-_:"
       NAMECASE
         GENERAL YES
         ENTITY NO

DELIM    GENERAL SGMLREF
         SHORTREF SGMLREF

NAMES SGMLREF
```

- **NAMING** permette di specificare i caratteri usabili nei nomi (all'inizio e nel corpo di un nome) e la case-sensitivity dei nomi.
- **DELIM**: **GENERAL** stabilisce quali sono i caratteri associati ai vari delimitatori. Per esempio STAGO (Start of Tag Open, '<'), ETAGO (End of Tag Open, '</'), o TAGC (Tag Close, '>'). **SHORTREF** definisce quali sono i caratteri disponibili per lo short reference.
- **NAMES**: quali sono i nomi riservati utilizzati nei DTD, come ELEMENT, DOCTYPE, ENTITY, ecc.

SGML declaration: a volo d'angelo (7)

```
QUANTITY  SGMLREF
          ATTCNT    60
          ATTSPLEN  65536
          LITLEN    65536
          NAMELEN   65536
          PILEN     65536
          TAGLVL    100
          TAGLEN    65536
          GRPGTCNT  150
          GRPCNT    64
```

- **QUANTITY**: qual è la lunghezza massima delle caratteristiche del markup (per esempio, i nomi di elementi, attributi, entità).
  - **ATTCNT**: il numero massimo di valori diversi in una lista di attributi
  - **NAMELEN**: il numero massimo di caratteri in un nome o in un token (CRS. 8)
  - **TAGLVL**: la profondità massima di annidamento degli elementi (CRS: 24)
  - **TAGLEN**: il numero massimo di caratteri presenti in uno start–tag, inclusi nomi, attributi e valori (CRS: 960)

SGML declaration: a volo d'angelo (8)

```
FEATURES
   MINIMIZE DATATAG NO
            OMITTAG YES
            RANK NO
            SHORTTAG YES
   LINK      SIMPLE NO
          IMPLICIT NO
          EXPLICIT NO
   OTHER CONCUR NO
          SUBDOC NO
          FORMAL YES

APPINFO NONE
```

FEATURES: permette di specificare l'attivazione o meno di alcune caratteristiche facoltative di SGML.

- **MINIMIZE**: è possibile utilizzare formule di minimizzazione? Per esempio, usare caratteri del testo come delimitatori di tag (DATATAG), omettere interamente certi tag (OMITTAG), o usare forme accorciate per i tag (SHORTTAG).
- **LINK**: è possibile creare associazioni esplicite tra il documento e altri documenti connessi, come fogli di stile o cose simili?
- **OTHER**: altre caratteristiche attivabili nei programmi SGML.

APPINFO: Specificazioni di informazioni specifiche di certe applicazioni. Per esempio, se si usa HyTime, va specificato qui con la keyword '**HYTIME**'.

Conclusioni

Qui abbiamo parlato delle caratteristiche di SGML diverse da XML:

- Dichiarazioni multiple
- Eccezioni
- Meccanismi di minimizzazione
- Il problema del white space
- La dichiarazione SGML

Riferimenti

Wilde's WWW, capitolo 4

Altri testi:

- E. Maler, J. Al Andaloussi, ***Developing SGML DTDs, from text to model to markup***, Prentice Hall, 1997
- N. Bradley, ***The XML companion***, Addison-Wesley, 1998
- C.F. Goldfarb, ***The SGML handbook***, Clarendon Press, 1990