

Le idee sono la cosa più importante,  
il resto è una conseguenza.

*Richard Feynman*

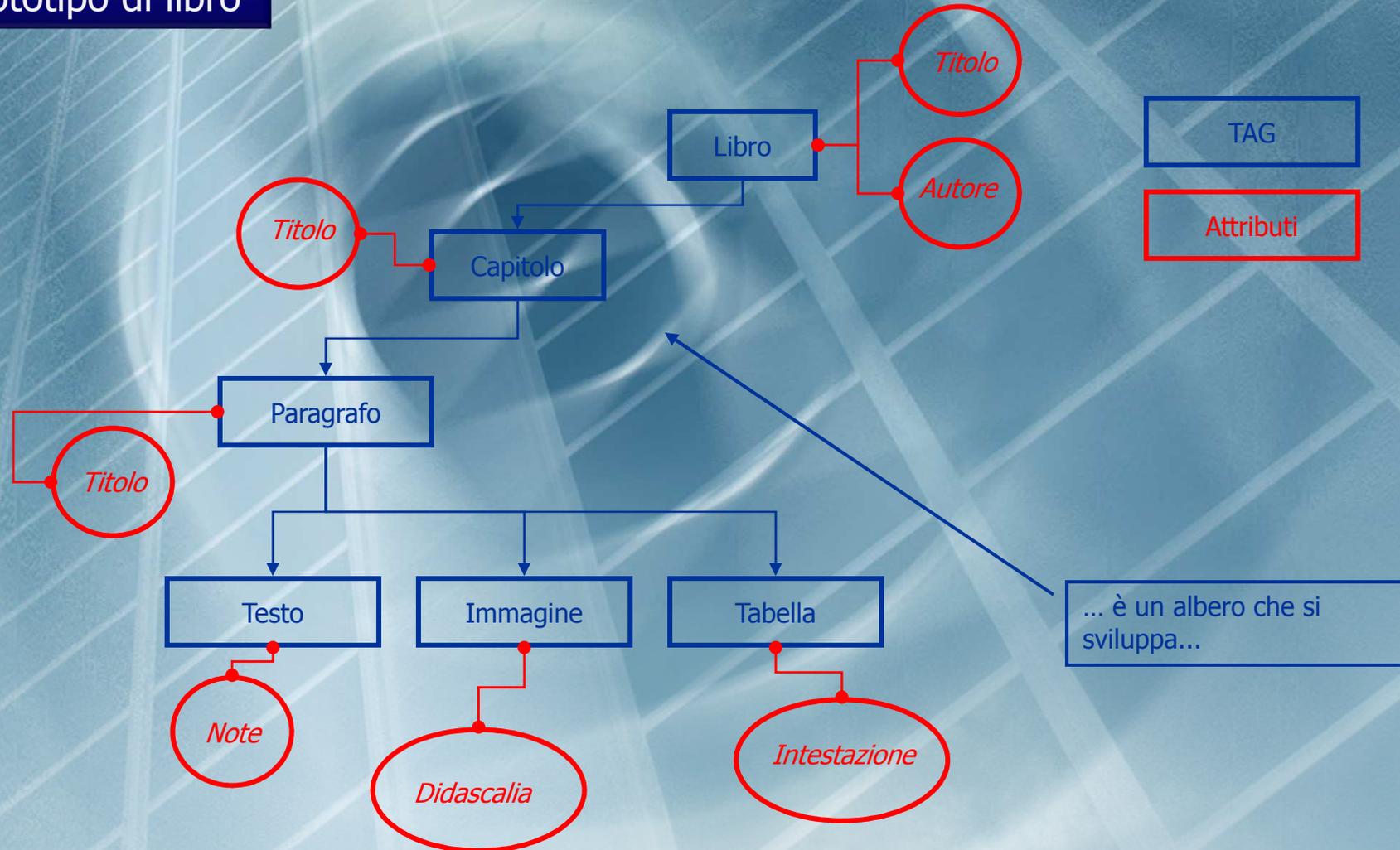
# Tecniche Multimediali

Corso di Laurea in «Informatica» - aa 2010-2011

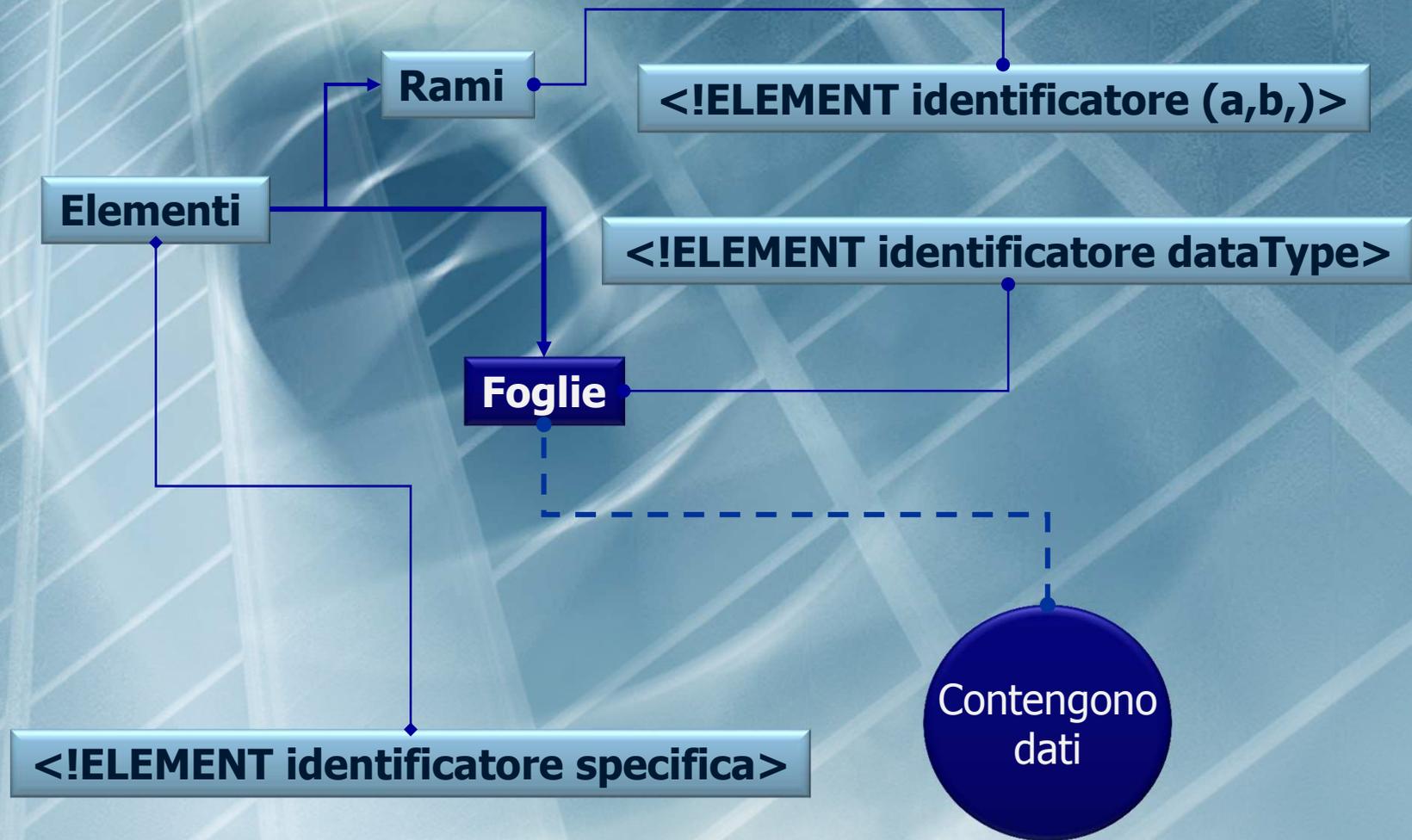
*Prof. Giorgio Poletti – [giorgio.poletti@unife.it](mailto:giorgio.poletti@unife.it)*

# XML struttura di un documento

## Prototipo di libro



# DTD: gli elementi



# DTD: gli elementi e la sintassi

**<!ELEMENT identificatore (a,b,)>**

**( )** Racchiudono una sequenza di elementi

**(elemento1, elemento2, elemento3)**

← virgola

**(elemento1 | elemento2 | elemento3)**

← pipe

**Elemento1** → l'elemento1 è obbligatorio e singolo

**Elemento1+** → l'elemento1 è obbligatorio e almeno una volta

**Elemento1?** → l'elemento1 è opzionale e al massimo una volta

**Elemento1\*** → l'elemento1 è opzionale e senza limiti di numero

# DTD: elementi e molteplicità

Molteplicità massimo 1

Molteplicità minimo 1

Dati Obbligatori

Numero di matricola

Destinatari di una mail

Numero di protocollo di un documento

Esempio Destinatari per conoscenza di una mail

Dati Opzionali

# DTD: gli elementi (tabella)

## SUFFISSI

Null	Obbligatorio
?	Opzionale (0 o una volta)
+	Uno o più volte
*	Zero o più volte

---

## SEPARATORI

,	(virgola)	tutti nell'ordine dell'elenco
	(pipe)	scelta (operatore OR esclusivo → XOR)

---

## RAGGRUPPAMENTO

( )	elenco di elementi
-----	--------------------

# DTD: attributi

**TAG** che identifica la dichiarazione della lista di attributi di un elemento



**Elemento** di cui si stanno definendo gli attributi

**<!ATTLIST NomeElemento NomeAttributo Tipo Default>**

**Nome** dell'attributo che si sta definendo

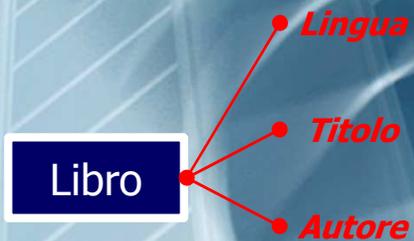


**Tipo** dell'attributo che si sta definendo



**Valore** predefinito dell'attributo

# DTD: attributi, un esempio



```
<!ELEMENT Libro (.....) >  
  <!ATTLIST Libro Titolo CDATA #REQUIRED  
    Autore CDATA #REQUIRED  
    Lingua (Italiano | Inglese) 'Italiano' >
```

# DTD: elementi e tipi di dati

`<!ELEMENT Paragrafo (DataType) >`

EMPTY

Un elemento vuoto  
(ad esempio il tag `<BR />`)

`<!ELEMENT br EMPTY>`

#PCDATA

Parsed Character DATA

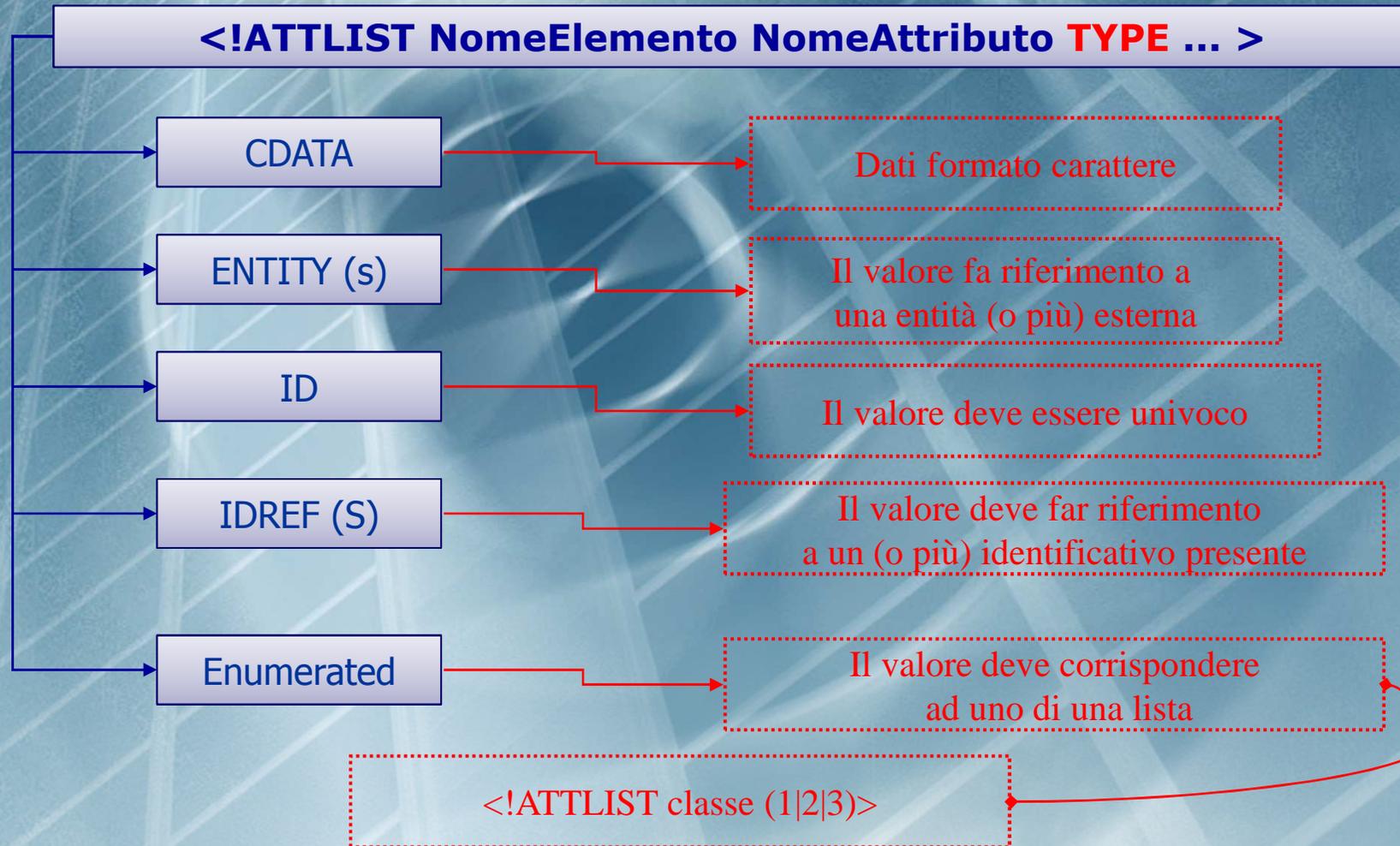
*Un testo generico*

ANY

*Elemento sicuramente non vuoto  
e di cui si ignora il tipo di contenuto*

`<!ELEMENT nota ANY>`

# DTD: attributi e tipi di dati



# DTD: attributi e tipi di dati

**<!ATTLIST NomeElemento NomeAttributo Type ...IMPOSTAZIONE >**

#REQUIRED

Attributo per cui è obbligatorio specificare un valore

#IMPLIED

Attributo opzionale, può essere Ignorato se manca il valore

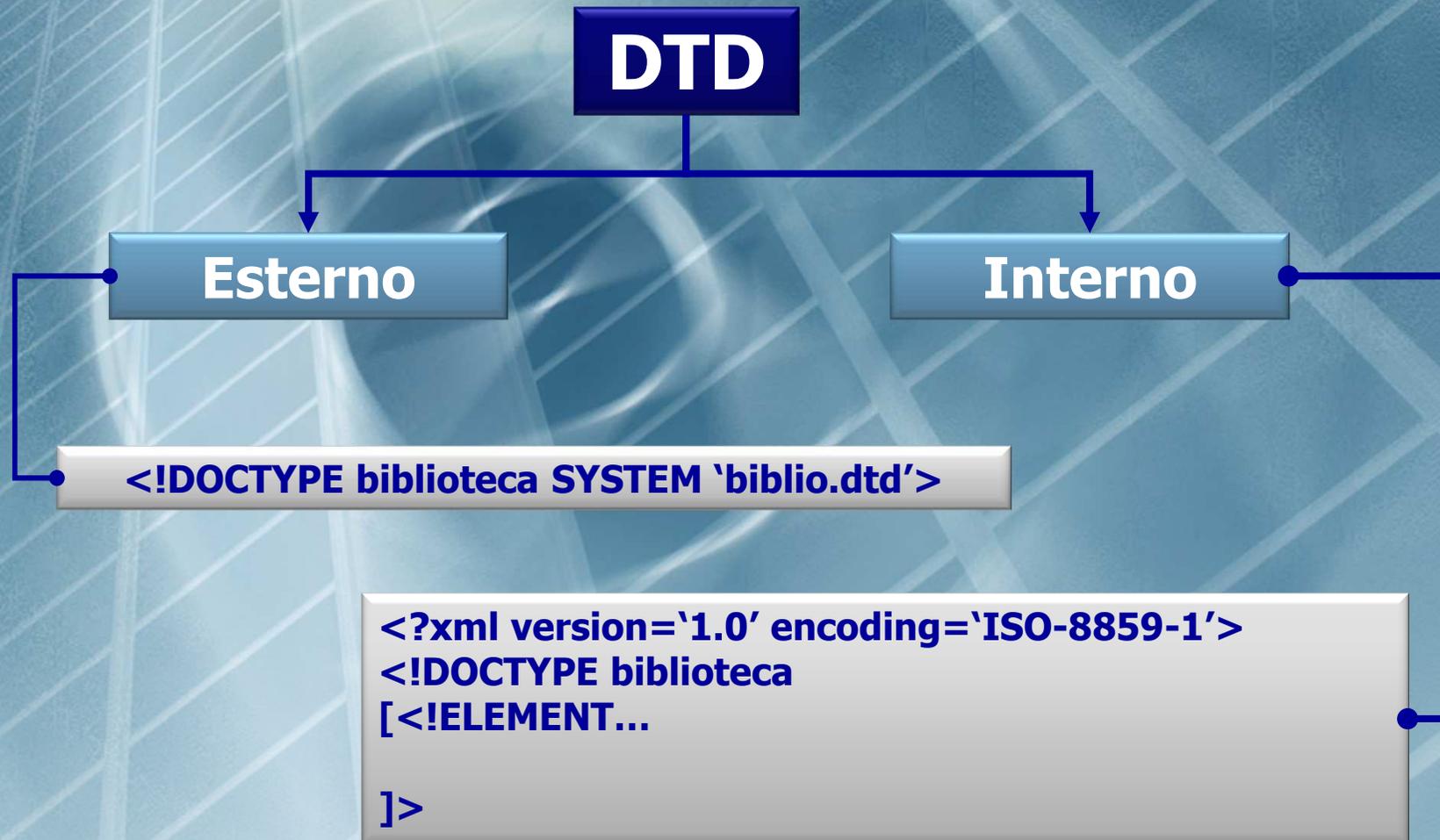
#FIXED ValoreFissato

Attributo con valore *ValoreFissato* che viene attribuito se l'attributo non è inserito nell'elemento

*Default*

Identifica un valore di DEFAULT per l'attributo

# DTD e file XML

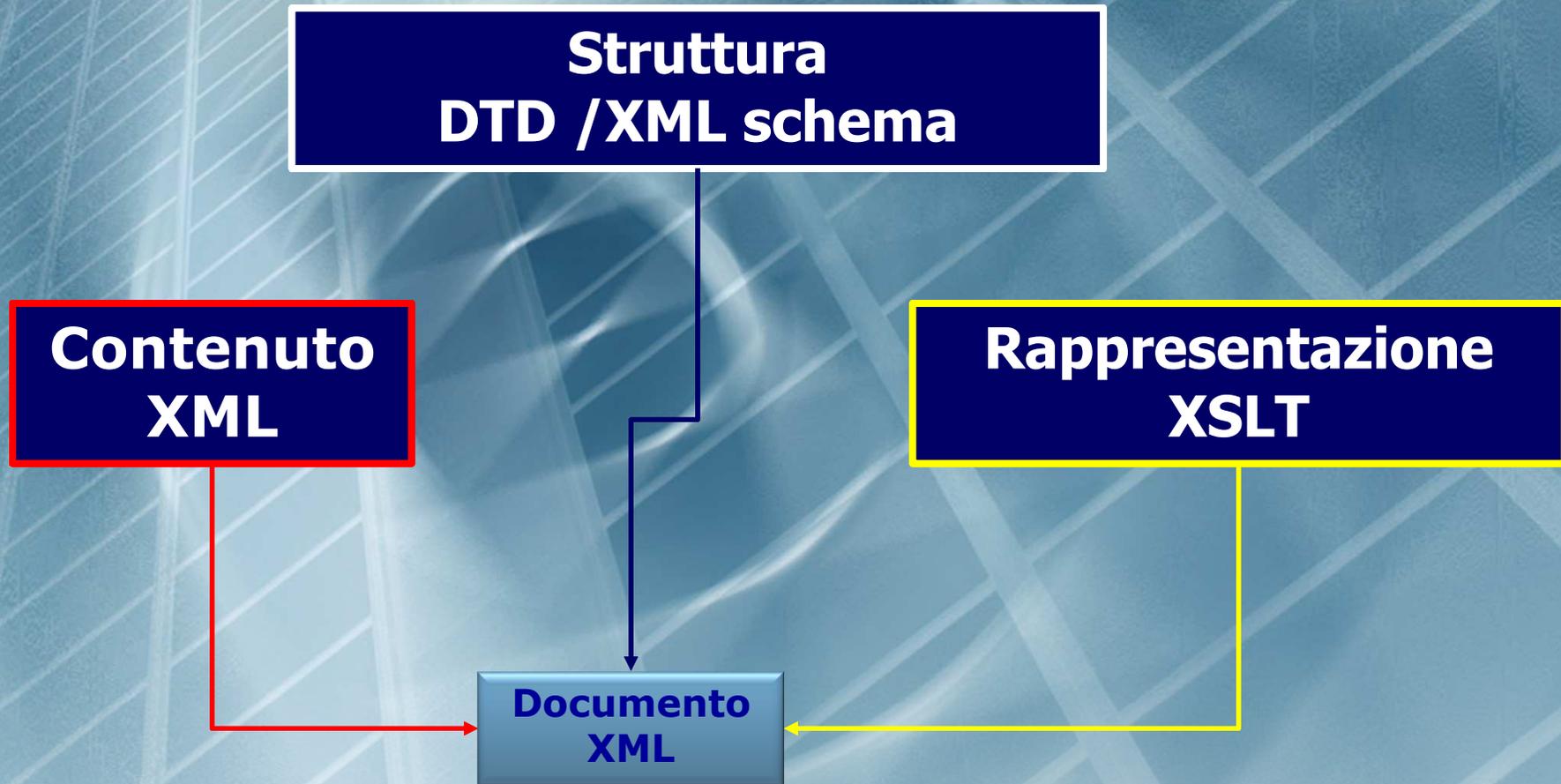


```
<!DOCTYPE biblioteca SYSTEM 'biblio.dtd'>
```

```
<?xml version='1.0' encoding='ISO-8859-1'>  
<!DOCTYPE biblioteca  
[<!ELEMENT...  
>]
```

# Documento XML :

Dati+Struttura+Rappresentazione



# XSLT e XML: la rappresentazione

Documento  
XML

Rappresentazione  
XSLT

Dal FORMATO



alla RAPPRESENTAZIONE

What You See Is What You Get  
(WYSIWYG)



What You See Is **NOT** What You Get  
(WYSINWYG)

# XSLT e XML: la rappresentazione

**Documento  
XML**

**Rappresentazione  
XSLT**

**eXtensible Stylesheet Language (XSL)**  
*Famiglia di linguaggi*

**XSL Transformations (XSLT)**  
*linguaggio per il transforming di XML*

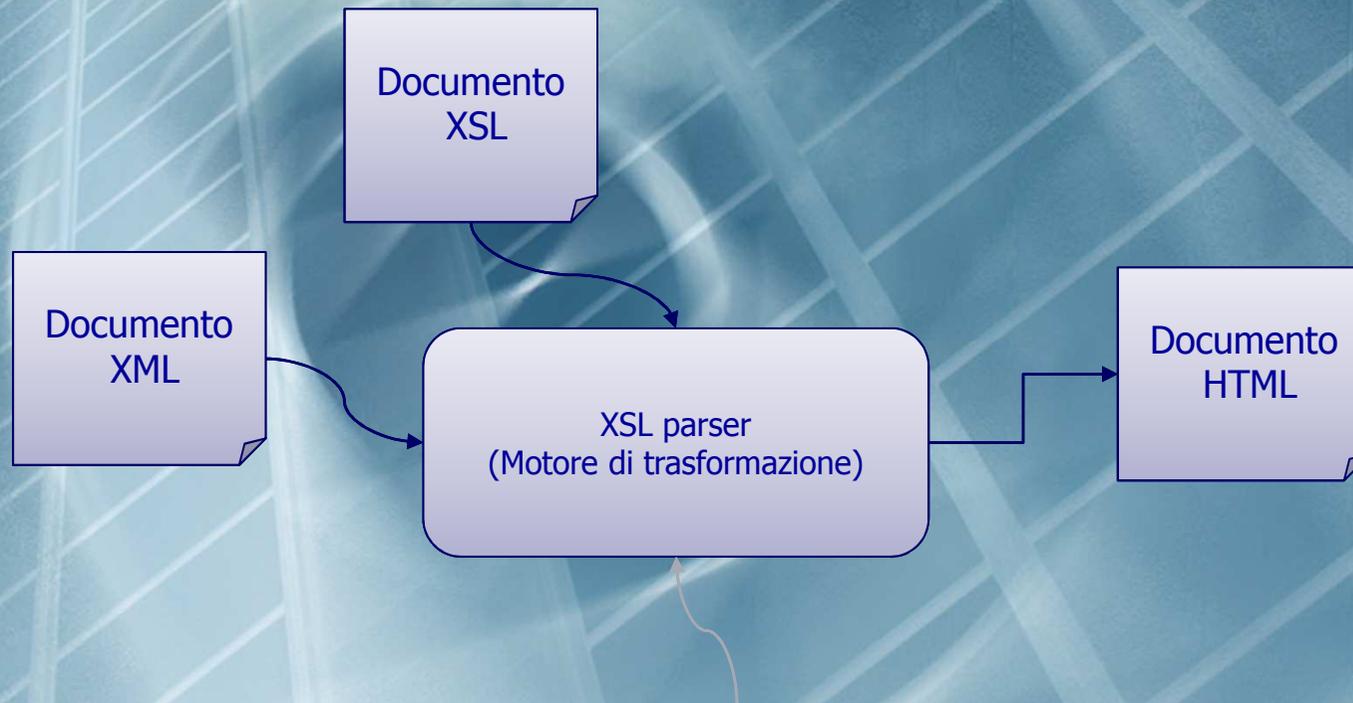
# XSLT e XML: la rappresentazione

*XSL Transformations (XSLT) è un linguaggio per il trasformazione*

*Trasforma un documento scritto in XML in un documento rappresentato in un altro linguaggio (ad esempio HTML)*

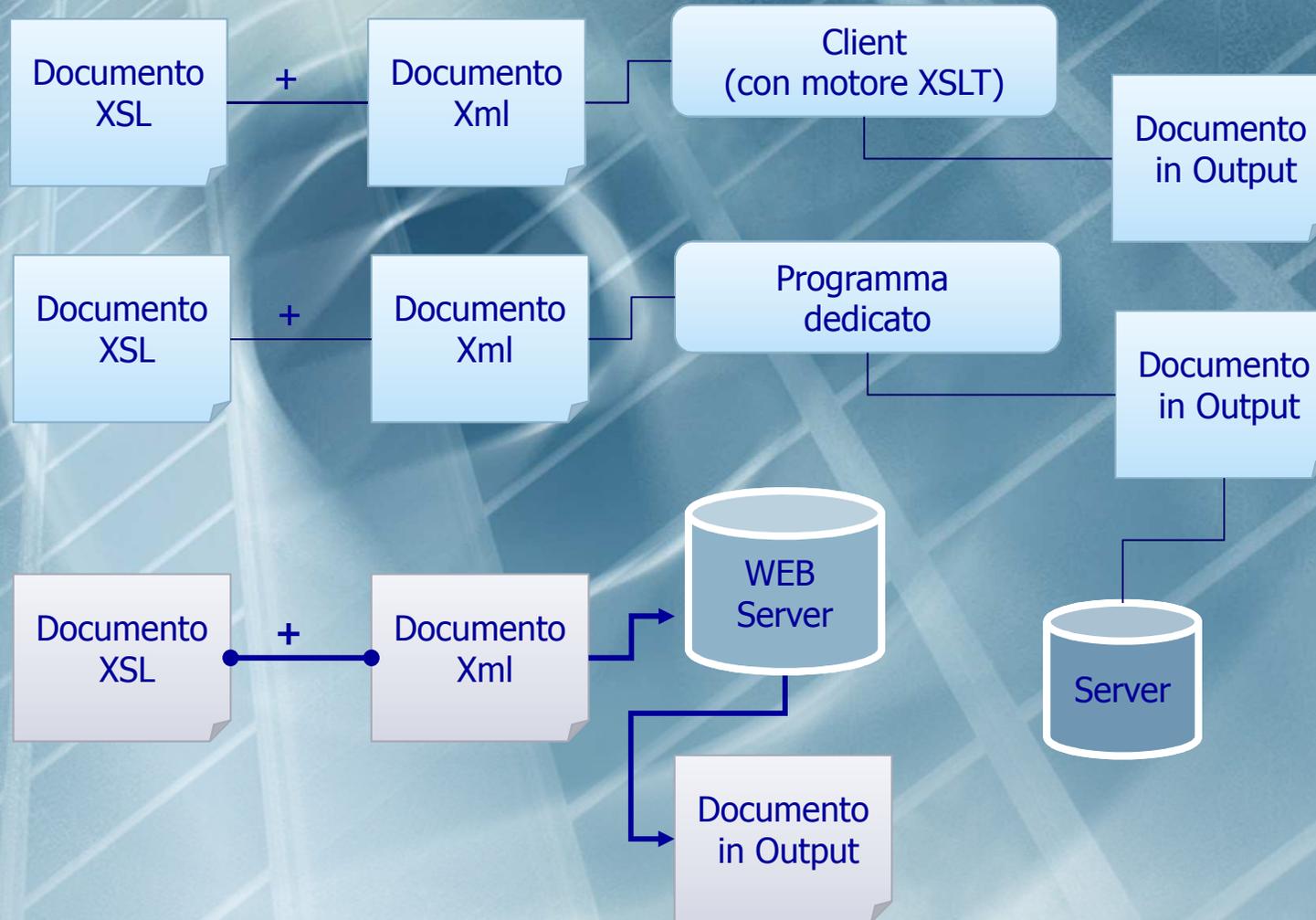


# XSLT e XML: il parser

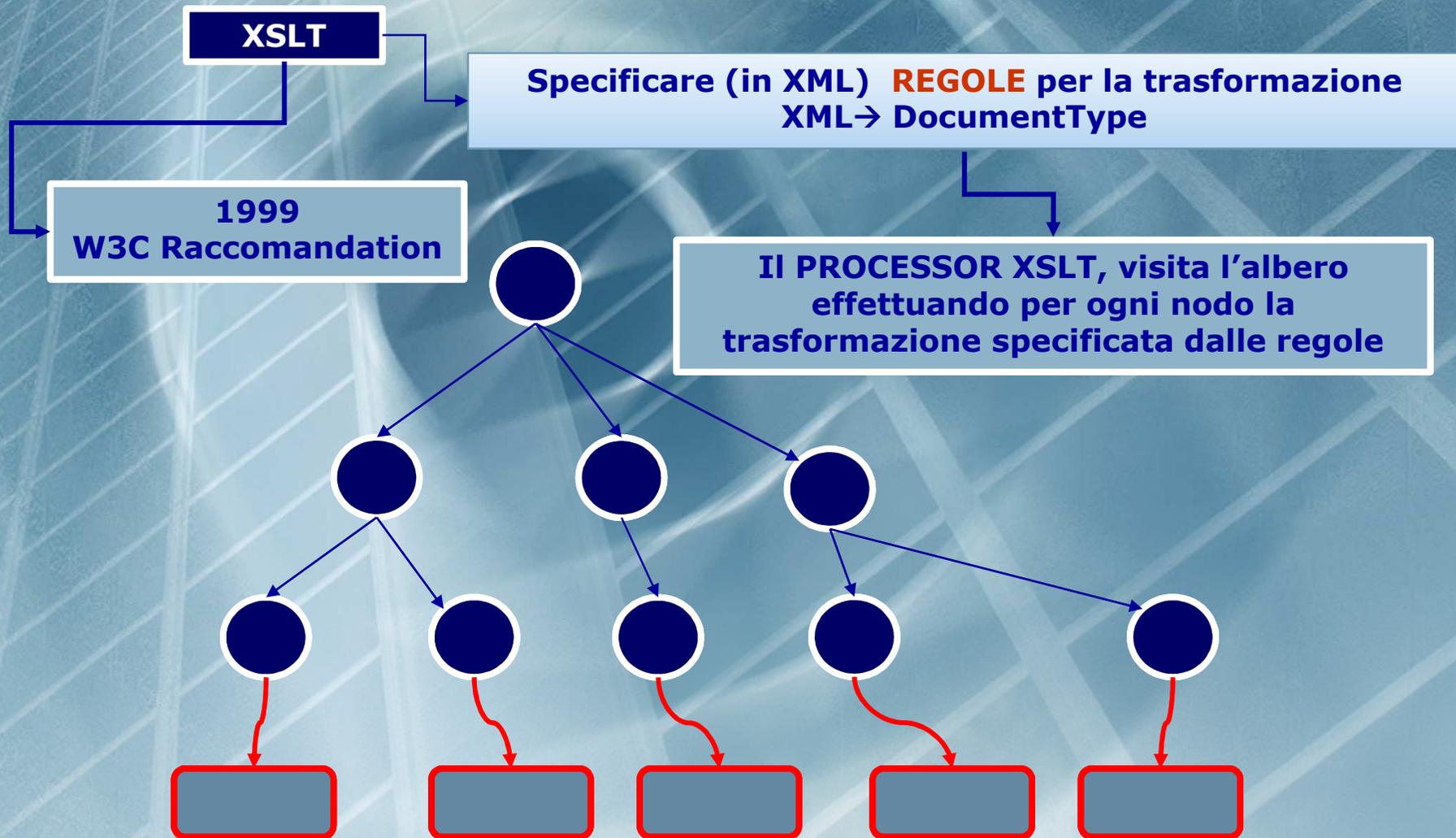


Il **parsing** o **analisi sintattica** è il processo atto ad analizzare uno stream (flusso) continuo in input (letto per esempio da un file o una tastiera) in modo da determinare la sua struttura grammaticale grazie ad una data grammatica formale. Un **parser** è un programma che esegue questo compito. *(definizione da WIKIPEDIA)*

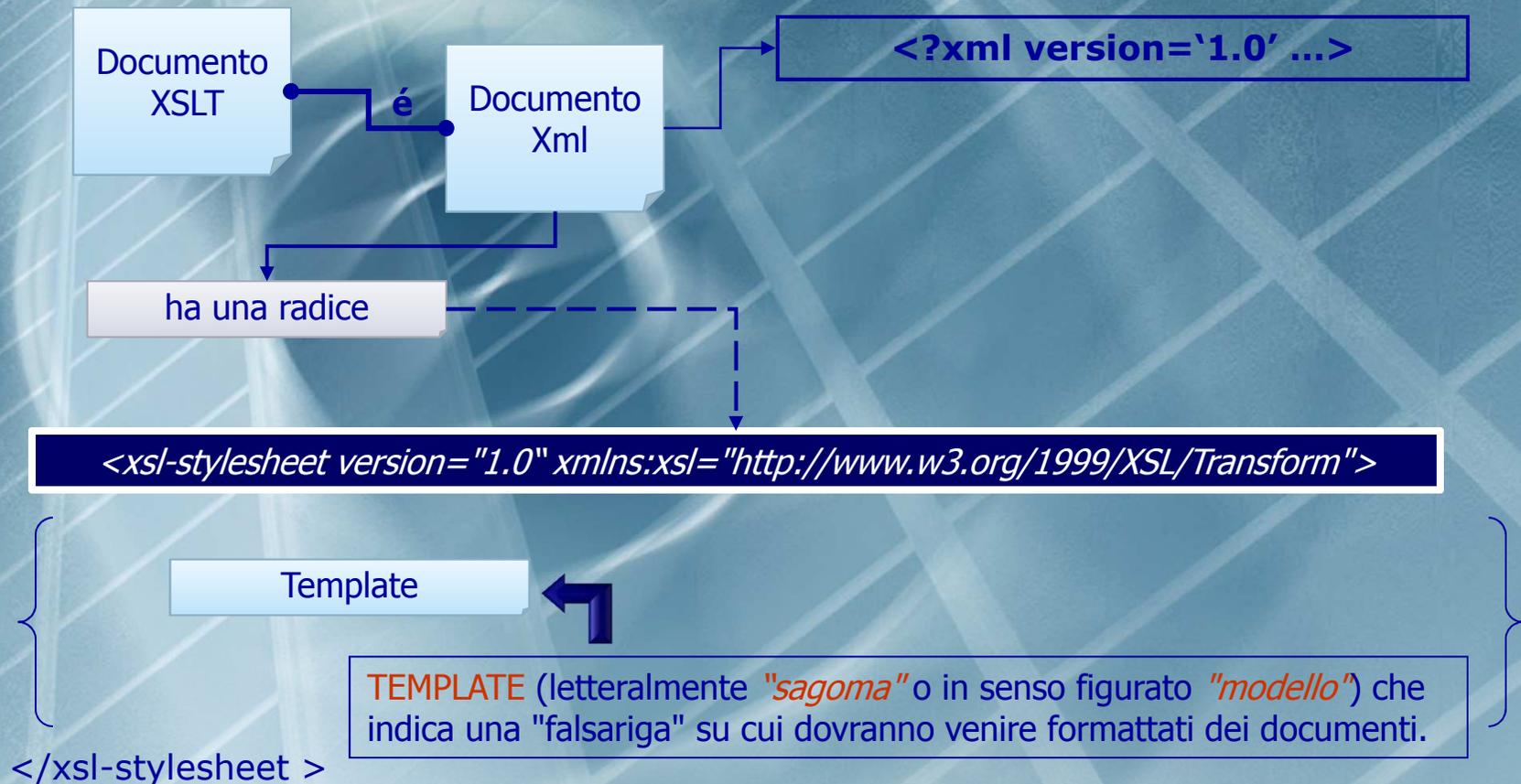
# XST: il parsing



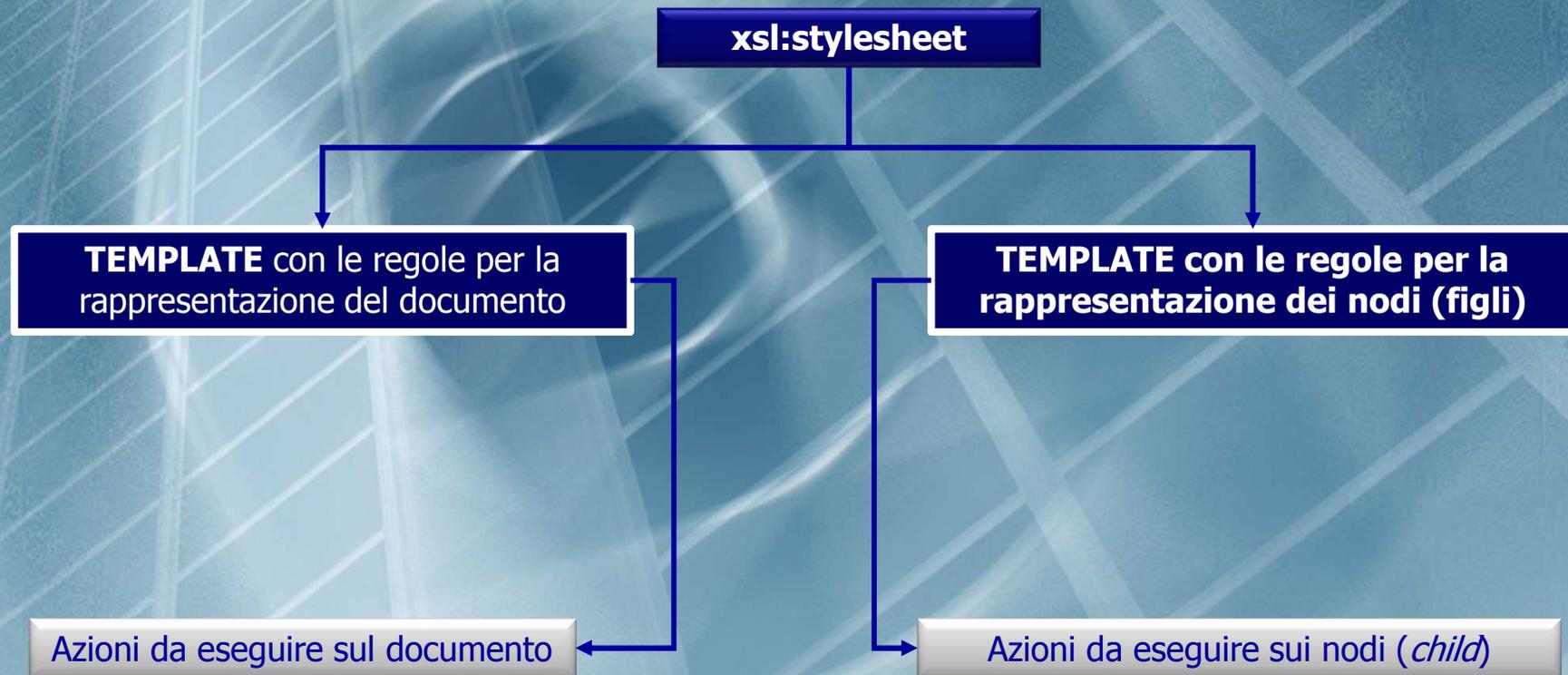
# XSLT



# XSLT struttura di un documento



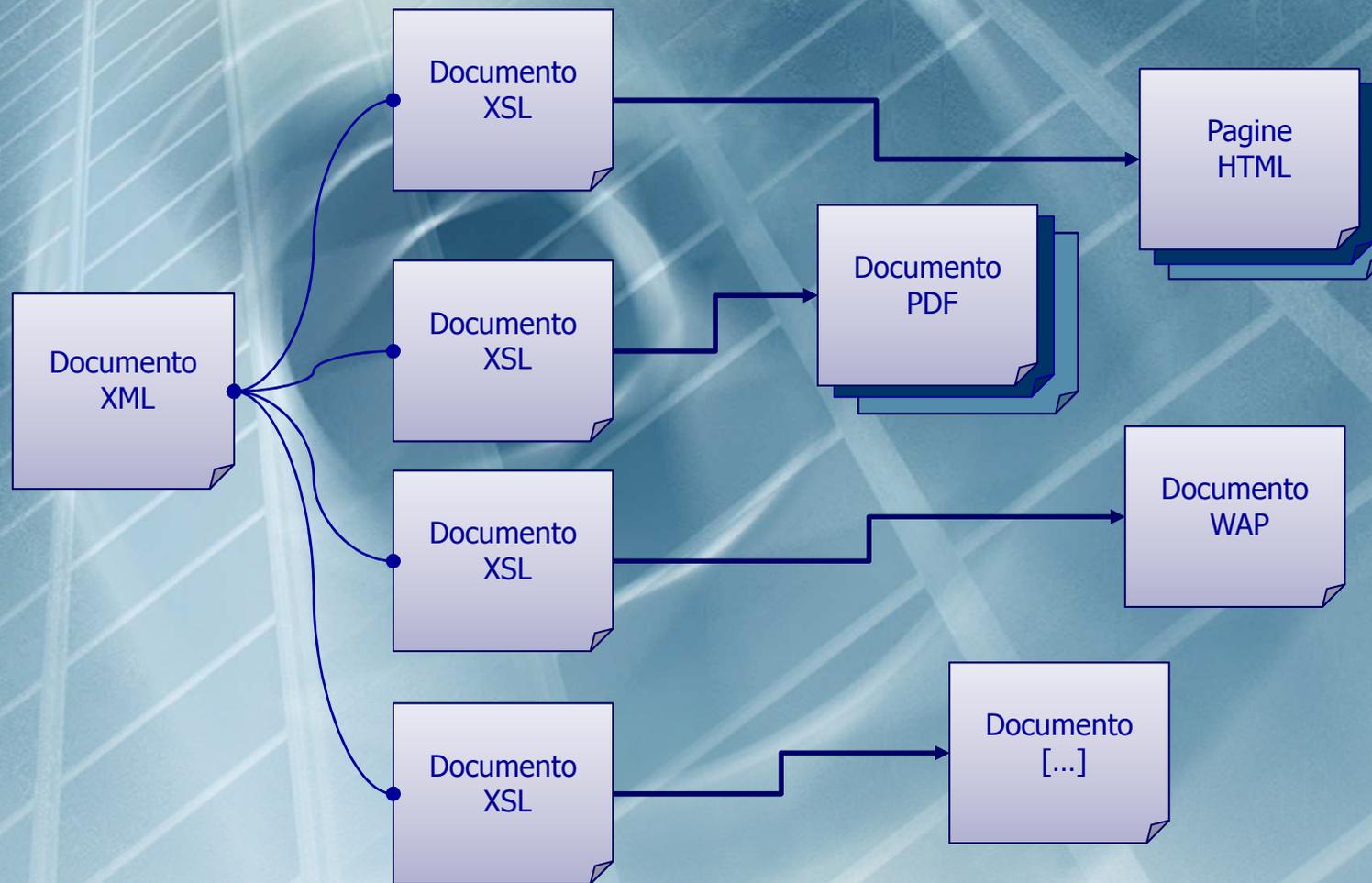
# XSLT struttura di un documento



# XSLT: un primo macro esempio

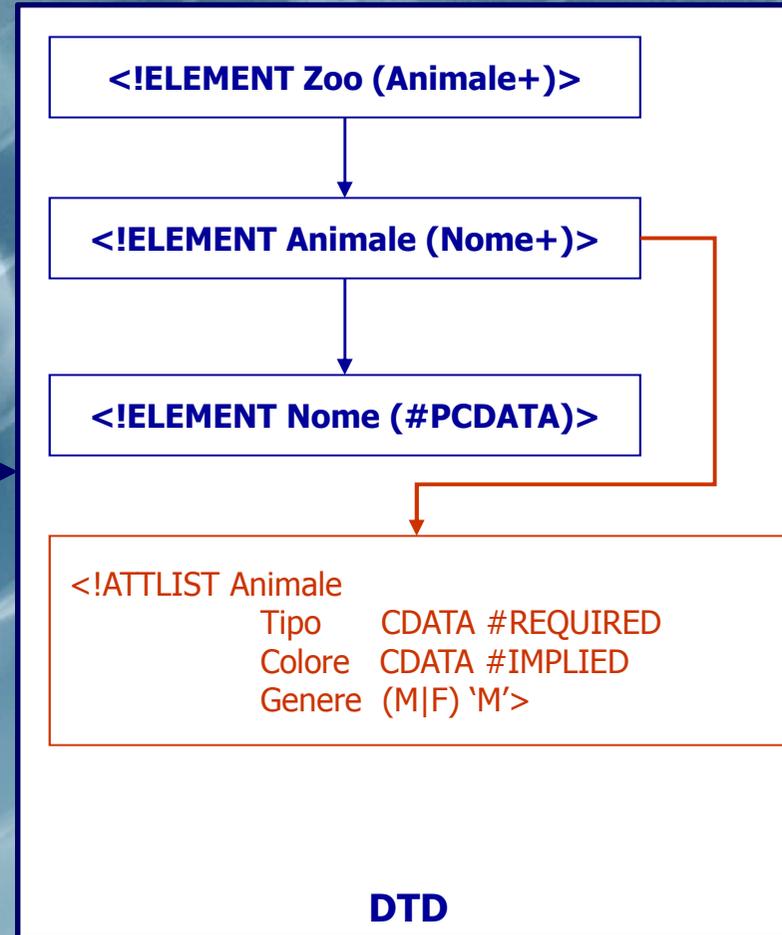
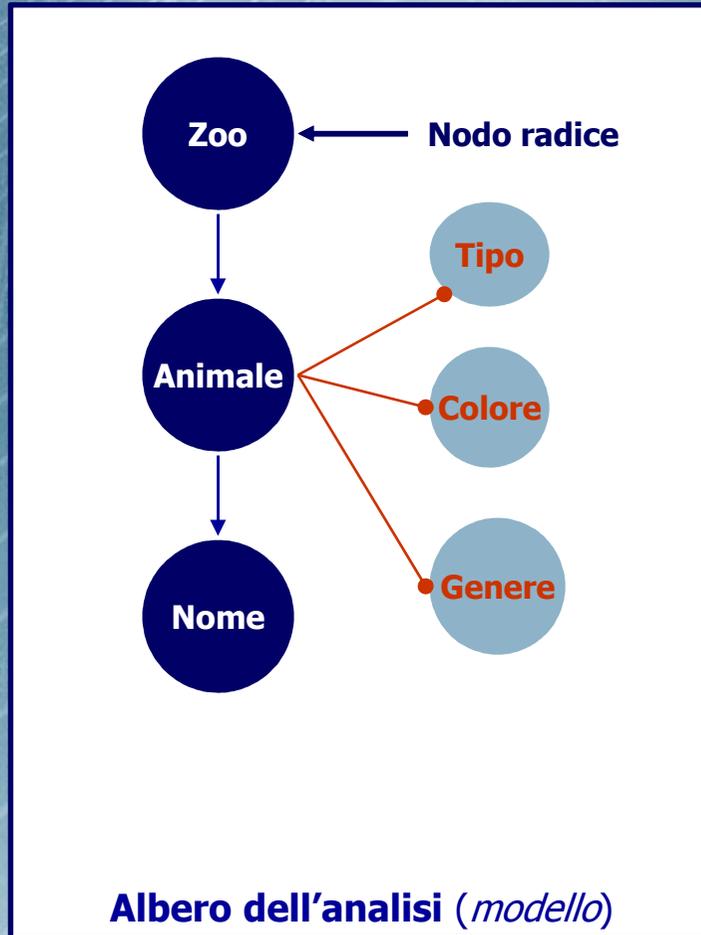
```
<?xml version="1.0"?>
<xsl:stylesheet>
  <xsl:template match="/">
    [action]
  </xsl:template>
  <xsl:template match="BookCatalogue">
    [action]
  </xsl:template>
  <xsl:template match="Book">
    [action]
  </xsl:template>
  ...
</xsl:stylesheet>
```

# XSLT: schema riassuntivo



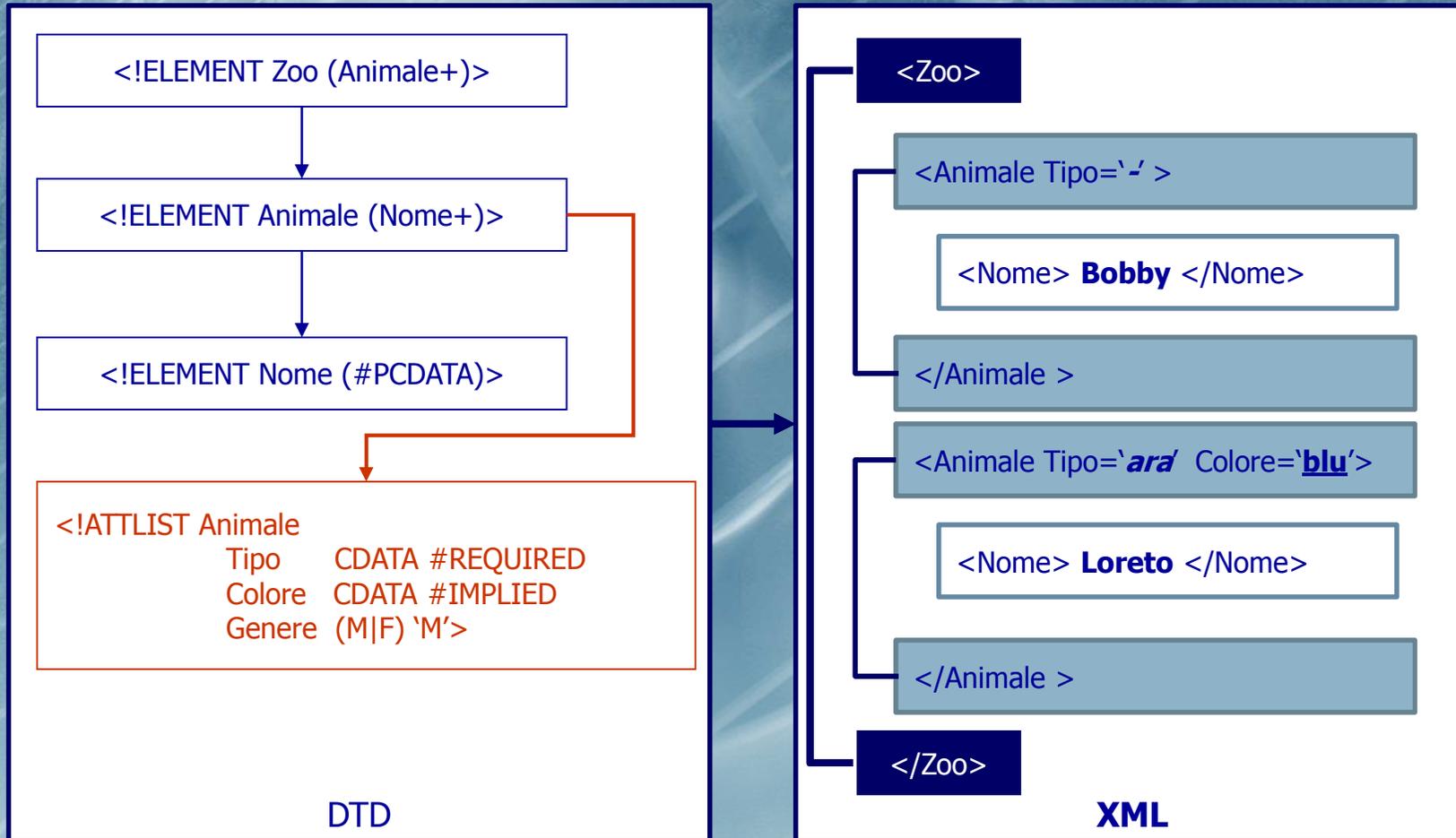
# XSLT: esempio di trasformazione

modello  $\Rightarrow$  DTD  $\Rightarrow$  XML  $\Rightarrow$  XSLT  $\Rightarrow$  Testo



# XSLT: esempio di trasformazione

modello ⇒ DTD ⇒ XML ⇒ XSLT ⇒ Testo



# XSLT: esempio di trasformazione

modello  $\Rightarrow$  DTD  $\Rightarrow$  XML  $\Rightarrow$  XSLT  $\Rightarrow$  Testo





# XSLT: esempio di trasformazione

modello ⇒ DTD ⇒ XML ⇒ XSLT ⇒ Testo

```
<?xml version='1.0' encoding='ISO-8859-1'?>
<xsl:stylesheet version='1.0' xmlns:xsl='http://www.w3.org/1999/XSL/Transform'>
<xsl:output method='text' encoding='ISO-8859-1' omit-xml-declaration='yes' />

  <xsl:template match='/'>
    <xsl:apply-templates />
  </xsl:template>

  <xsl:template match='Animale'>
    <xsl:apply-templates />
  </xsl:template >

  <xsl:template match='Animale'>
    Allo zoo ho visto <xsl:value-of select='.' /> che è un <xsl:value-of select='@Tipo' />
    <xsl:apply-templates select='@Colore' />
  </xsl:template>

  <xsl:template match='@Colore'>
    che è di colore <xsl:value-of select='.' />
  </xsl:template>

</xsl:stylesheet>
```

# XSLT: esempio di trasformazione

modello ⇒ DTD ⇒ XML ⇒ XSLT ⇒ Testo

```
<?xml version='1.0' encoding='ISO-8859-1'?>
<xsl:stylesheet version='1.0' xmlns:xsl='http://www.w3.org/1999/XSL/Transform'>
<xsl:output method='html' encoding='ISO-8859-1' omit-xml-declaration='yes' />
  <xsl:template match='/'>
    <html>
      <head>
        <title>From XML To HTML</title>
      </head>
      <body>
        <xsl:apply-templates />
      </body>
    </html>
  </xsl:template>
  <xsl:template match='Animale'>
    <xsl:apply-templates />
  </xsl:template >
  <xsl:template match='Animale'>
    <P>Allo zoo ho visto <xsl:value-of select='.' /> che è un <xsl:value-of select='@Tipo' /></P>
  <xsl:apply-templates select='@Colore' />
  </xsl:template>
  <xsl:template match='@Colore'>
    che è di colore <B> <xsl:value-of select='.' /> </B>
  </xsl:template>
</xsl:stylesheet>
```

«Il test di un programma può essere usato per mostrare la presenza di bug, ma mai per mostrare la loro assenza.»

*Edsger Wybe Dijkstra*