Dieci anni di XML

FEDERICO MESCHINI

And one man in his time plays many parts,
His acts being seven ages […]
At first the infant,
Mewling and puking in the nurse's arms.
Then, the whining schoolboy with his satchel
Unwillingly to school […]
And so he plays his part.
(William Shakespeare, *The seven ages of men.*)

Nobody has added to the 10-year, 20-year vision of what the Web should fundamentally be and whether it should be changing.
(Tim Berners-Lee)

Con alle spalle dieci anni di età, XML è ormai uno standard affermato e largamente utilizzato, ma al tempo stesso in continua evoluzione. Questo articolo ripercorre le tappe principali della sua storia, partendo dai motivi che ne hanno determinato la creazione, in primis *la diffusione del* World Wide Web*, e il rapporto con SGML, il suo progenitore. Vengono anche presentate le altre tecnologie collegate che completano il disegno d'insieme, come i fogli di stile o i linguaggi d'interrogazione, e il rapporto di XML con gli altri paradigmi dell'informazione digitale, tra cui i database relazionali e i formati delle* suite *di Office Automation.*

Parole chiave: XML - SGML - World Wide Web - Xquery - Biblioteca digitale - XML database

Dieci anni sono un periodo di tempo considerevole in qualsiasi settore. Nell'informatica però, disciplina caratterizzata da un proprio ritmo particolare[1], questo stesso intervallo di tempo assume una diversa dimensione e importanza, soprattutto considerando che la sua intera storia è composta da cinque, massimo sei decenni e, se si provasse a fotografare per ognuno di essi un determinato momento, uno stato

[1] Vedi l'ormai famosa *Moore's Law* <http://en.wikipedia.org/wiki/Moore_Law>.

cristallizzato, il *collage* complessivo risultante sarebbe estremamente variegato, con differenze profonde, se non addirittura impensabili, nel passaggio da uno spaccato all'altro, sia per il tipo di tecnologie disponibili, sia per l'utilizzo che ne viene fatto.

Questa apparente disomogeneità è in realtà percorsa da tutta una serie di linee evolutive parallele, fatte di ricerca, sviluppo e sperimentazioni, i cui effetti tangibili spesso si manifestano a distanza di diverso tempo e con risultati inaspettati. È il caso, ad esempio, dei database relazionali, che hanno iniziato a diffondersi su larga scala verso la fine degli anni '80, ma la cui fondazione teorica risale al 1970[2]. Per il *World Wide Web*, inventato tra il 1989 ed il 1990 ed esploso qualche anno dopo, è necessario arrivare fino agli anni '60, con i primi esperimenti sulle reti telematiche e quando, successivamente, prese forma l'idea di quella che poi sarebbe diventata Internet.

XML[3], uno degli standard più di successo dell'ultimo periodo, non fa eccezione a questa regola. Infatti, se prendiamo come punto di partenza la presentazione ufficiale della prima *Working Draft*[4] effettuata da John Bosak alla *SGML '96*

[2] Edgard Codd, *A relational model of data for large shared data banks*. "Communications of the ACM", 13 (1970), n. 6, p. 377-387 <http://www.acm.org/classics/nov95/toc.html>.

[3] *eXtensible MarkUp Language* <http://www.w3.org/XML/>. Per un'introduzione vedi Lou Burnard, *A gentle introduction to XML*, in C. M. Sperberg-McQueen - Lou Burnard (eds.), *TEI P5: guidelines for electronic text encoding and interchange*. TEI Consortium, 2007 <http://www.tei-c.org/release/doc/tei-p5-doc/html/SG.html>, ed. it. Lou Burnard, *Una semplice introduzione a XML*, in Fabio Ciotti (a cura di), *Il manuale TEI LITE. Introduzione alla codifica elettronica dei testi letterari*. Sylvestre Bonnard, 2006. Tra la vasta manualistica disponibile vedi Elliotte Rusty Harold - W. Scott Means, *XML in a nutshell*. O'Reilly, 2004; David Hunter - Jeff Rafter - Joe Fawcett - Eric van der Vlist - Danny Ayers - Jon Duckett - Andrew Watt - Linda McKinnon, *Beginning XML*. Wrox, 2007. Per una panoramica di casi concreti e consigli d'uso vedi Elliotte Rusty Harold, *Effective XML: 50 specific ways to improve your XML*. Addison-Wesley Professional, 2003. In lingua italiana Massimo Canducci, *XML pocket*. Apogeo, 2005; Paolo Pialorsi, *XML*. Mondadori Informatica, 2002; Anders Møller - Micheal I. Schwartzbach, *Introduzione a XML*. Pearson Education, 2007.

[4] *Extensible Markup Language (XML) W3C Working Draft 14-Nov-96* <http://www.w3.org/TR/WD-xml-961114.html>. Vedi anche *XML development history* <http://www.w3.org/XML/hist2002>. All'interno delle specifiche del W3C, una *Working Draft* (WD) rappresenta il primo livello di maturità, una sorta di presentazione semi ufficiale, aperta a suggerimenti e modifiche da parte della comunità di utenti e sviluppatori. Alla WD segue la *Candidate Recommendation* (CR) ed infine la *W3C Recommendation* (REC), che definisce lo standard in maniera ufficiale. Tra la pubblicazione di una WD e di una REC può passare diverso tempo, vedi il caso di *XQuery*, le cui prime specifiche risalgono al 2003, mentre la *Recommendation* è stata pubblicata nel gennaio 2007 <http://www.w3.org/TR/2007/REC-xquery-20070123/>. Di conseguenza notevoli modifiche possono essere presenti nel passaggio da una fase all'altra. Nel caso di *XML* la CR è stata pubblicata nel dicembre 2007 <http://www.w3.org/TR/PR-xml-971208> e la prima REC nel febbraio 1998 <http://www.w3.org/TR/1998/REC-xml-19980210>. La REC più recente è quella relativa alla quarta edizione, dell'agosto 2006 <http://www.w3.org/TR/2006/REC-xml-20060816/>.

Conference[5], stiamo già a più di dieci anni di anzianità sul campo. Per avere un quadro completo, però, è necessario fare qualche passo indietro, partendo proprio dal già citato *SGML*[6], che, a sua volta, a quella stessa conferenza festeggiava il proprio decimo compleanno come standard ufficiale ISO[7], e le cui radici risalgono alla metà degli anni '60[8]. Come molti già ben sanno, *XML* è un *subset*, un sottoinsieme di *SGML*, esplicitamente ideato per semplificare il suo potente, mastodontico e complicato genitore, adattandolo all'allora emergente Web[9].

SGML nasce come un metalinguaggio dichiarativo di marcatura[10] con lo scopo di creare, manipolare e gestire documenti elettronici *machine-readable*, che non siano legati ad una determinata architettura *hardware* o *software*, e che di conseguenza non presentino problemi di condivisione e preservazione, o che perlomeno li riducano di molto. Charles Goldfarb, padre di *SGML*, prima di andare a lavorare all'IBM, esercitava la professione di avvocato e di conseguenza questa tecnologia nasce con già nel DNA la propensione verso fenomeni testuali di un certo tipo, come documentazione legale, tecnica e amministrativa. Questo approccio non esaurisce né soddisfa totalmente tutta la vasta gamma di possibilità previste dall'informazione cosiddetta documento-centrica[11], termine ombrello coniato negli ultimi anni e contrapposto all'altra tipologia informativa, denominata data-centrica. Sebbene questa terminologia si sia diffusa solo recentemente, la differenza cui fa riferimento ha radici profonde.

Negli anni '70, durante la fase di sviluppo e adozione di *SGML*, il paradigma dominante nei linguaggi di programmazione era quello della cosiddetta programmazione strutturata, riassunto molto bene dall'espressione "algoritmi più strutture dati". Ma i documenti, e il tipo d'informazione che rappresentano, caratterizzata da uno stato fortemente fluido e polimorfico, sono sempre stati un qualcosa di ostico da riportare basandosi solamente sulle strutture dati tipiche dell'informatica[12]. Di

5 Che dall'anno successivo divenne *SGML/XML '97 Conference*, per eliminare totalmente dal titolo, a partire dal 1998, la parte relativa ad *SGML*, diventando così ufficialmente *XML Conference* <http://www.xmlconference.org>.

6 *Standard Generalized Markup Language* <http://www.w3.org/MarkUp/SGML/>. Charles F. Goldfarb, *The SGML handbook*. Oxford University Press, 1991.

7 ISO 8879:1986.

8 Charles F. Goldfarb, *The roots of SGML -- A personal recollection.* <http://www.sgmlsource.com/history/roots.htm>.

9 Vedi *Design principles for XML* <http://www.textuality.com/sgml-erb/dd-1996-0001.html>.

10 Per un'introduzione al concetto di linguaggio di marcatura vedi Francesca Tomasi, *La rappresentazione dell'informazione testuale e i linguaggi di codifica*, in T. Numerico - A. Vespignani (a cura di), *Informatica per le scienze umanistiche*. Bologna : Il Mulino, 2003.

11 Per un approfondimento di queste tematiche vedi Domenico Fiormonte, *Scrittura e filologia nell'era digitale*. Torino : Bollati Boringhieri, 2003, in particolare i capitoli 4 e 5.

12 Come ad esempio alberi, tabelle, pile, vettori, matrici, record. Vedi Thomas H. Cormen - Charles E. Leiserson - Ronald L. Rivest - Clifford Stein, *Introduzione agli algoritmi e strutture dati*. McGraw-Hill, 2005.

ostico, ma allo stesso tempo anche estremamente importante, così *SGML* a partire dalla fine anni '70, inizio '80 è stato implementato proficuamente in diversi settori, principalmente quello editoriale, da chi all'epoca, ma ancora oggi, poteva permettersene, sia a livello di investimento economico sia di competenze necessarie, l'utilizzo. Di conseguenza l'elenco è ristretto a non molti nomi, seppure prestigiosi[13], e ad applicazioni specialistiche[14]. Questo principalmente perché nel progettarne le specifiche, la potenza è stata preferita a scapito della semplicità, limitandone così la diffusione. Ciò nonostante *SGML* è stato adottato in tutti quei casi in cui l'importanza della componente testuale sia a livello contenutistico che strutturale era fondamentale, con una scelta che si è rivelata al tempo stesso coraggiosa e lungimirante. Qui non è possibile non citare la TEI[15], un'associazione internazionale che ha come scopo principale la creazione di uno standard per la codifica elettronica dei testi letterari[16], che nel 2007 festeggia il suo ventesimo anno di attività[17], e che, sin dalla prima versione delle relative *Guidelines*[18], decise di utilizzare *SGML*, nonostante la già citata propensione di questo linguaggio per un certo tipo di informazione testuale, in cui l'ambiguità va per forza di cose eliminata totalmente e la struttura viene uniformata a quella che è stata definita una *Ordered Hierarchy of Content Objects (OHCO)*[19].
Questo approccio è estremamente funzionale e adeguato nella maggior parte dei casi, ma presenta però i suoi limiti, soprattutto nel gestire le cosiddette gerarchie sovrapposte (*overlapping hierarchies* [20], situazione che per i testi letterari si è rivelata

[13] Tra cui Elsevier, Springer-Verlag, Kluwer, Oxford University Press, CERN [Organisation Européenne pour la Recherche Nucléaire], IOP [Institute of Physics], European Patent Office, USDD [United States Department of Defense], American Chemical Society e Air Transport Association of America.

[14] Va citato in particolare l'*Oxford English Dictionary*, che a partire dalla seconda edizione è stato realizzato utilizzando *SGML*. L. Elliot, *How the Oxford English Dictionary went online.* <http://www.ariadne.ac.uk/issue24/oed-tech/>.

[15] Text Encoding Initiative <http://www.tei-c.org/>.

[16] A. Renear, *Text encoding*, in Susan Schreibman - Ray Siemens - John Unsworth (eds.), *A companion to digital humanities*. Oxford : Blackwell, 2004 <http://www.digitalhumanities.org/companion/>.

[17] *TEI@20: 20 years of supporting the digital humanities* <http://www.lib.umd.edu/dcr/events/teiconference/>.

[18] <http://www.tei-c.org/Guidelines2/>.

[19] S.-J. DeRose - D.-G. Durand - E. Mylonas - A.-H. Renear, *What is text, really*? "Journal of Computing in Higher Education", 1 (1990), p. 3-26.

[20] Possibilità prevista dalle specifiche di *SGML* tramite la funzionalità CONCUR, ma di difficile implementazione. Vedi anche A.-H. Renear - E. Mylonas - D.-G. Durand, *Refining our notion of what text really is: the problem of overlapping hierarchies*, in Susan Hockey - Nancy Ide (eds.), *Research in humanities computing 4: selected papers from the ALLC/ACH Conference*. Christ Church, Oxford, April 1992, p. 263-80 <http://www.digitalhumanities.org/view/Essays/AllenRenearOverlappingHierarchies>.

essere di una certa importanza[21]. Altri standard degni di nota, anche se non certo gli unici, sono *HyTime*, per la creazione di ipertesti e di presentazioni multimediali[22], e *DocBook*, per la documentazione tecnica[23].

Prima si era parlato delle difficoltà relative all'implementazione di un'applicazione basata su *SGML*. Un esempio potrebbe essere la realizzazione di un archivio elettronico di testi codificati in *TEI* o in *DocBook*, che richiederebbe la presenza di un *framework* di pubblicazione come *DynaText*/*DynaWeb*[24] e l'impiego del linguaggio di fogli di stile *DSSSL*[25], entrambi abbastanza onerosi, come scritto sopra sia dal punto di vista economico sia da quello pratico[26].

All'inizio degli anni '90 ben poche persone avrebbero scommesso su una diffusione a larga scala di *SGML*, così come sull'enorme successo che avrebbe avuto di lì a poco il *World Wide Web*, l'invenzione di Tim Berners-Lee, basato principalmente su due tecnologie complementari e dalle sigle simili, ossia *HTTP*[27], un protocollo client-server il cui scopo principale è la trasmissione di documenti ipertestuali, e *HTML*[28], il linguaggio utilizzato per la creazione di questi documenti, derivato proprio da *SGML*. La diffusione di *HTML* è enorme, grazie anche alla sua semplicità e alla rapida curva d'apprendimento. Semplicità che però ben presto diviene un limite. Si vorrebbe di più dalle pagine Web, si vorrebbe che fossero più ricche semanticamente, dotate di una struttura al tempo stesso più rigida ed efficiente, e più semplici

21 Jerome McGann, *Radiant textuality: literary studies after the world wide web*. Palgrave/St. Martins, 2001; Peter Shillingsburg, *From Gutenberg to Google*. Cambridge University Press, 2006. Inoltre uno Special Interest Group della TEI è espressamente dedicato alla questione dell'*overlapping*, <http://www.tei-c.org/wiki/index.php/SIG:Overlap>.

22 <http://www.hytime.org/>. Diversi concetti di *HyTime* sono stati ripresi poi sia dall'*HTML* sia dalle *TopicMaps*, uno standard ISO per la gestione e rappresentazione della conoscenza <http://www.topicmaps.org/>.

23 <http://www.oasis-open.org/docbook/intro.shtml>.

24 <http://www.inso.com/dynatext/index.htm>. *DynaText* prevede la pubblicazione su CD-Rom, LAN e Intranet, tramite l'utilizzo di un *browser* proprietario, mentre *DynaWeb* permette l'implementazione delle stesse funzioni sul Web, trasformando dinamicamente i documenti *SGML* in *HTML*, fruibili quindi da un qualsiasi *browser*.

25 *Document Style Semantics and Specification Language* <http://dsssl.netfolder.com/>.

26 Esempi di biblioteche digitali basate su questa architettura sono il *Women Writers Project* <http://www.wwp.brown.edu/>, la *IEEE Computer Society Digital Library* <http://www.computer.org/epub/>e l'italiano *TIL -Testi in Linea*, confluito poi in *Biblioteca Italiana* <http://www.bibliotecaitaliana.it>. In tutti questi progetti *XML* è ora utilizzato al posto di *SGML*, e il posto di *DynaText* è stato preso da uno dei tanti *framework* disponibili. In particolare il *Women Writers Project* utilizza *Philologic* <http://philologic.uchicago.edu/>, mentre la *IEEE CSDL* un'applicazione *J2EE* appositamente sviluppata.

27 *Hypertext Transfer Protocol* <http://www.w3.org/Protocols/>.

28 *Hypertext Markup Language* <http://www.w3.org/MarkUp/>.

da manipolare tramite i linguaggi di programmazione. Per tutti questi requisiti si guarda indietro, al progenitore di *HTML*, a quel misterioso ed esoterico *Standard Generalized Markup Language*. Già nel 1995 l'idea è quella di portare *SGML* sul Web, mantenendo le sue potenzialità ma rendendolo al tempo stesso più semplice e adatto al nuovo medium. Così l'anno successivo, all'interno del W3C, il *World Wide Web Consortium*, prende forma un *Working Group* con lo scopo di definire le specifiche di questo nuovo linguaggio. Il Gruppo è composto da undici persone e coordinato da James Clark, che propone il nome poi scelto rispetto alle altre possibili alternative: *SLIM - Structured Language for Internet Markup*, *MAGMA - Minimal Architecture for Generalized Markup Applications* e *MGML - Minimal Generalized Markup Language*. Va riconosciuto che, rispetto a queste altre possibilità, *XML* è un acronimo che funziona, in quanto efficace e semplice da ricordare. Dopo un intenso lavoro di circa venti settimane, tra luglio e novembre del 1996, viene pubblicata la prima *Working Draft*, con l'ormai famosa presentazione di John Bosak alla *SGML '96*. Le differenze con il suo pesante genitore si fanno subito vedere: le specifiche di *XML* sono di circa 25 pagine, contro le 150 di *SGML* e uno studente universitario riesce a scrivere un *parser*, un programma che ne verifichi la correttezza sintattica, in poco meno di due settimane [29].

Il nucleo centrale del nuovo standard è pronto. Ora il vero punto cruciale sta nel riuscire a renderlo appetibile, per invogliarne l'utilizzo da parte degli sviluppatori. Così il lavoro su *XML*, inteso come famiglia di tecnologie, prosegue ininterrottamente, aggiungendo di volta in volta nuovi tasselli al disegno globale, che prende forma tra la fine degli anni '90 e l'inizio del 2000. Accanto alla sintassi di base un ruolo importante è certamente ricoperto dalle *Document Type Definitions* (*DTD*), eredità di *SGML*, per la definizione delle regole strutturali che i file *XML* dovranno rispettare per aderire ad una determinata tipologia di documenti. Le *DTD* presentano però diverse limitazioni e verranno ben presto sostituite dagli *Schema* [30] che, oltre ad offrire maggiori funzionalità, si basano sulla stessa sintassi *XML*. Gli *Schema* del W3C, sebbene largamente diffusi, non sono esenti da critiche, in particolare riguardo a una certa complessità, e, per questo motivo, sono state sviluppate delle soluzioni alternative [31]. La parte di specifiche riguardante i fogli di stile, gli *eXtensible Stylesheet Language* [32], per la visualizzazione dei documenti, ha da subito un ruolo di primo

[29] Vedi anche James Clark, *Comparison of SGML and XML*. <http://www.w3.org/TR/NOTE-sgml-xml.html>.

[30] <http://www.w3.org/XML/Schema>.

[31] Tra cui la principale è sicuramente *RelaxNG* <http://relaxng.org/>.

[32] <http://www.w3.org/Style/XSL/>. A livello funzionale gli *XSL* occupano lo stesso ruolo che ha *DSSSL* rispetto ad *SGML*.

piano, ed è composta da tre parti principali: *XSLT*, i fogli di stile di trasformazione [33]; *XSL-FO*, per la rappresentazione visiva dei contenuti; *XPath*, un linguaggio per selezionare [34] le parti di un documento *XML*, utilizzandone la struttura ad albero. Nelle intenzioni originarie *XSL-FO* avrebbe dovuto avere un ruolo di punta in questo gruppo, in realtà ricoperto poi da *XSTL*. Questo perché, originariamente, uno dei compiti principali di *XML* era quello di divenire il linguaggio interpretato dai *browser* per la visualizzazione delle pagine Web, che non avrebbero più utilizzato *HTML*, bensì il suo fratello maggiore, coadiuvato da *XSL-FO* per la formattazione grafica e da *XLink* [35] per la creazione di *link* ipertestuali dotati di funzionalità avanzate rispetto ai semplici collegamenti unidirezionali. Così non è successo per diversi motivi, tra cui il mancato supporto dei *browser* dell'epoca, e *HTML* è rimasto tranquillamente al suo posto.

In questo scenario le caratteristiche di *XSLT*, la sua capacità di manipolare granularmente un documento *XML*, grazie anche all'uso congiunto con *XPath*, trasformandolo in un altro formato "text-based" [36], hanno fatto sì che assumesse un ruolo di primo piano, catalizzando anche l'attenzione degli sviluppatori, che hanno iniziato a rendere disponibili per questo linguaggio numerose estensioni, al fine di aumentarne le capacità [37]; estensioni poi confluite in *XSLT 2.0*, la cui prima *Working Draft* è del dicembre 2001 [38], a solo due anni di distanza dalla *Recommendation* della prima *release* [39]. Parallelamente ai fogli di stile di trasformazione, anche *XPath*, a partire dal 2003, ha iniziato ad evolversi in 2.0 [40], diventando così la base di *XQuery* [41], un linguaggio di interrogazione dotato di funzionalità simili a quelle di *SQL* [42] per i database relazionali e quindi una delle ultime e più interessanti novità. Lo sviluppo di

[33] *XSLT* è abbastanza diverso dalle altre tipologie di linguaggi funzionali, la famiglia a cui appartiene, in quanto basato principalmente sul concetto di modelli iterativi, ereditato direttamente da *DSSSL*.

[34] E che al contrario di *XSLT* e *XSL-FO* non utilizza una sintassi *XML*.

[35] <http://www.w3.org/TR/xlink/>.

[36] Come ad esempio un altro tipo di *XML*, *HTML*, *RTF* o *TXT*.

[37] <http://www.exslt.org/>.

[38] Mentre la *W3C Recommendation* è molto più recente, essendo stata pubblicata nel gennaio 2007 <http://www.w3.org/TR/2007/REC-xslt20-20070123/>. Vedi Michael Kay, *XSLT 2.0 programmer's reference*. Wrox, 2004; oppure Jeni Tennison, *Beginning XSLT 2.0: from novice to professional*. Apress, 2005.

[39] Del 16 novembre 1999 <http://www.w3.org/TR/1999/REC-xslt-19991116>.

[40] <http://www.w3.org/TR/xpath20/>. Vedi anche Michael Kay, *XPath 2.0 programmer's reference*. Wrox, 2004.

[41] <http://www.w3.org/XML/Query/>. Per un'introduzione vedi Michael Kay, *Learn XQuery in 10 minutes*. Per maggiori approfondimenti P. Walmsley, *XQuery*. O'Reilly, 2007; M. Brundage, *XQuery: the XML query language*. Addison-Wesley, 2004.

[42] *Structured Query Language* <http://en.wikipedia.org/wiki/SQL>.

XSLT 2.0, *XPath 2.0* e *XQuery 1.0* è avvenuto in maniera parallela e coordinata tra i vari gruppi, in modo da avere tra di loro una base il più comune possibile, relativamente alla libreria di funzioni ed il modello di dati disponibile. Infatti, anche se esiste una certa sovrapposizione tra quello che è possibile fare con *XSLT 2.0* e *XQuery*, il primo mantiene sempre un approccio più documento-centrico, mentre il secondo è dichiaratamente orientato verso una visione data-centrica.

Di estrema importanza sono anche *DOM*[43] e *SAX*[44], due interfacce di programmazione. Sebbene ormai tutti i principali linguaggi includano al loro interno librerie e funzioni apposite per lavorare direttamente con *XML*, Java è sempre stato particolarmente adatto ad essere utilizzato in coppia con questo formato, sin dalle prime *Recommendation*[45], e proprio in questo linguaggio sono stati realizzati i primi *parser*, *processor*[46] e *framework*[47] di un certo rilievo, di cui la maggior parte *open source*.

L'elenco complessivo delle tecnologie e delle specifiche relative ad *XML* ne include molte di più di quelle citate fino ad ora, anche limitandosi solo a quelle "ufficiali" del W3C, ognuna con il proprio ruolo all'interno di un disegno che via via si è fatto sempre più complesso e stratificato. Si va da casi il cui uso è ormai estremamente dif-

[43] *Document Object Module* <http://www.w3.org/DOM/>. *DOM* è uno standard ufficiale del W3C e la sua caratteristica principale è quella di riprodurre in memoria la struttura ad albero del documento *XML* analizzato, necessitando così, all'aumentare delle dimensioni del documento, di un notevole quantitativo di risorse.

[44] *Simple API for XML* <http://sax.sourceforge.net/>. *SAX* è stato sviluppato da un gruppo di programmatori, e si basa su un procedimento diverso da *DOM*. Il documento *XML* viene letto sequenzialmente e durante la lettura viene segnalato il verificarsi di determinati eventi, come per esempio l'inizio o la fine di un elemento. A causa di questo differente approccio è evidente come *SAX* richieda meno risorse di *DOM*.

[45] Una prima spiegazione è data dalla loro complementarità: il codice portabile di Java e i dati portabili di *XML*. Vedi Matthew Fuchs, *Why XML is meant for Java?* <http://www.xml.com/pub/a/1999/06/fuchs/fuchs.html>. Manuali interamente dedicati al rapporto tra Java e *XML* sono Brett McLaughlin, *Java & XML*. 2nd ed. O'Reilly Media, 2001 ed Elliotte Rusty Harold, *Processing XML with Java: a guide to SAX, DOM, JDOM, JAXP, and TrAX*. Addison-Wesley Professional, 2002 <http://www.cafeconleche.org/books/xmljava/>.

[46] Un *parser*, come scritto sopra, è un programma con il compito di controllare la validità di un documento *XML*, mentre un *processor* viene utilizzato per manipolarlo, come ad esempio l'applicazione di un foglio di stile. *Xerces* <http://xerces.apache.org/xerces-j/> e *Xalan* <http://xml.apache.org/xalan-j/> sono rispettivamente il *parser* e il *processor* dell'*Apache Foundation*. Degno di nota è anche *Saxon* <http://www.saxonica.com/>, sviluppato da Michael Kay, che, nelle versioni più recenti, supporta anche *XQuery*.

[47] Un *framework publishing* estremamente diffuso è *Cocoon* <http://cocoon.apache.org/>, sempre della *Apache Foundation*, integrabile sia con il motore di ricerca *Lucene* <http://lucene.apache.org/> sia con il database *XML eXist* <http://exist.sourceforge.net/>.

fuso, come i *Namespace* [48], ad altri molto più esoterici, l'*Infoset* [49] e l'*Encryption* [50] ad esempio, oppure estremamente interessanti, come le *XForms*[51] o *XPointer*[52], ma ancora abbastanza carenti dal punto di vista di un'implementazione concreta.

Alla fine degli anni '90, anche dopo la pubblicazione della prima *Recommendation*, *XML* appariva ancora abbastanza acerbo, nonostante le sue potenzialità fossero sotto gli occhi di tutti, principalmente quella di essere una specie di coltellino svizzero dell'informazione digitale, impiegabile laddove ci fosse bisogno di dati portabili, di formati non proprietari e possibilmente condivisi e per il dialogo di applicazioni diverse, tramite una base sintattica comune che svolgesse il ruolo di apripista ad una possibile interoperabilità semantica. Quello che mancava era una *killer application* che mostrasse al mondo le reali possibilità di questo standard. Con il senno di poi probabilmente all'epoca da *XML* ci si aspettava troppo, che risolvesse in poco tempo tutti i problemi del Web o addirittura di tutto il mondo digitale. Il progetto originario era quello di fargli prendere direttamente il posto di *HTML* nei *browser*, utilizzando congiuntamente *XML*, *XSL-FO* ed *XLink*, aggiungendo così nuove e più potenti funzionalità alle pagine Web. Questo piano però non ha ottenuto i risultati sperati per tutta una serie di motivi, a partire dallo scarso supporto da parte dei *browser* stessi, e si è rivelato troppo in avanti sui tempi, rendendo così necessarie altre modalità di impiego. Il Web, caratterizzato da un'architettura di *network*, sia tecnologica sia sociale, spesso è poco ricettivo alle imposizioni dall'alto, risultando sensibile in maniera molto maggiore alle iniziative che nascono parallelamente dal basso.

In ogni caso, grazie alla combinazione delle caratteristiche insite, del supporto da parte del W3C e del relativo lavoro sulle specifiche disponibili, insieme alla disponibilità di numerosi programmi, di cui molti *open source*, lo scenario delineato era estremamente diverso rispetto a quello in cui si trovava *SGML*, e di conseguenza il successo di *XML* era solamente rimandato, anche se ancora con la necessità di trovare il modo. Passato il 2000 insieme al *millennium bug*, ed essendo sopravvissuto al proprio *hype*, questo standard si è trovato nella situazione di dover rientrare nel Web, passando stavolta non dalla porta principale bensì dalla finestra, ossia come veicolo non dei contenuti principali, bensì delle (meta)informazioni aggiuntive. Proprio a partire dal 2000, una delle prime applicazioni ad avere una diffusione sempre mag-

[48] <http://www.w3.org/TR/REC-xml-names/>.
[49] <http://www.w3.org/TR/xml-infoset/>.
[50] <http://www.w3.org/Encryption/2001/>.
[51] <http://www.w3.org/MarkUp/Forms/>.
[52] <http://www.w3.org/TR/xptr/>.

giore, e tuttora dilagante, è stata quella relativa ai *feed RSS*[53] per l'aggiornamento delle notizie pubblicate sui *weblog* e sulle fonti informative *on line*. *XML* si è rivelato lo strumento più adatto per veicolare contenuti ridotti alla loro struttura essenziale, in cui l'aspetto presentazionale è totalmente assente, risolvendo così la limitazione principale di *HTML*. Il caso di *RSS* è interessante per diversi motivi, in quanto sotto questa etichetta sono presenti diversi formati, divisi in due famiglie principali (*RSS 0.90*, *RSS 1.0* e *RSS 1.1* da un lato e *RSS 0.91*, *RSS 0.9x* e *RSS 2.0* dall'altro) incompatibili tra di loro. L'utilizzo di una sintassi comune non garantisce automaticamente la compatibilità tra un formato e l'altro, o perlomeno un certo tipo di compatibilità, quella che sta alla base del cosiddetto *semantic web*[54]. Proprio su quest'ultimo si basa fortemente *XML*, in quanto i formati su cui si basa, *RDF*, *RDFSchema* e *OWL*[55], lo impiegano come principali sintassi di serializzazione[56]. Oltretutto proprio la famiglia di cui fa parte *RSS 1.x* è basata su *RDF*. L'utilizzo su larga scala di *RSS*, nei suoi vari dialetti, può essere considerato come il simbolo della diffusione a macchia d'olio di *XML*, inquadrabile intorno al 2002-2003, a cinque anni di distanza dalla pubblicazione della prima *Recommendation*, nel 1998.

Se nei primi anni *XML* aveva stentato un po' cercando la strada giusta per farsi notare, a partire dal 2003, con ormai un lustro di esperienza alle spalle, è ormai inarrestabile e inizia a dilagare un po' dappertutto nel mondo dell'informatica, anche in quei settori in cui la sua presenza non appare così naturale e intuitiva[57]; ma ormai è un fiume in piena, una tecnologia *trendy* e *cutting-edge*, un'etichetta da applicare ad

[53] Acronimo che si può sciogliere in diversi modi: *Really Simple Syndication*, *RDF Site Summary* oppure *Rich Site Summary* <http://en.wikipedia.org/wiki/RSS>.

[54] Tim Berners-Lee, *Weaving the Web: the original design and ultimate destiny of the World Wide Web*. Collins, 2000; T. Berners-Lee - J. Hendler - O. Lassila, *The semantic web*. "Scientific American", May 2001, p. 34-43 <http://www.sciam.com/article.cfm?articleID=00048144-10D2-1C70-84A9809EC588EF21&pageNumber=1&catID=2>; Grigoris Antoniou - Frank van Harmelen, *A semantic web primer*. MIT Press, 2004.

[55] Rispettivamente *Resource Description Framework* <http://www.w3.org/RDF/>, *Resource Description Framework Schema* <http://www.w3.org/TR/rdf-schema/> e *Web Ontology Language* <http://www.w3.org/2004/OWL/>. Vedi Shelley Powers, *Practical RDF*. O'Reilly, 2003 e Lee W. Lacy, *Owl: representing information using the web ontology language*. Trafford Publishing, 2005.

[56] Sebbene non l'unica disponibile. Sintassi non *XML* per *RDF* sono *Notation3* <http://www.w3.org/DesignIssues/Notation3.html> e *Turtle* <http://www.dajobe.org/2004/01/turtle/>.

[57] Interessante sotto questo punto di vista è il sito web *Monastic XML: an ascetic view of XML best practices* <http://monasticxml.org/> realizzato da Simon St. Laurent, autore di numerose pubblicazioni sull'argomento, tra cui *XML: a primer* e *XML pocket reference*. *Monastic XML* si propone come una risorsa informativa sugli utilizzi più appropriati di *XML*, mettendo in guardia contro impieghi che potrebbero rivelarsi controproducenti nel medio e lungo termine.

un'applicazione o a un *software* per renderli più accattivanti. Questo grazie anche all'adozione da parte di società come *Microsoft*, *Sun* e *Oracle*, e il ruolo chiave giocato in architetture in cui la componente d'interoperabilità è fondamentale, come ad esempio i *web service*[58].

Nei settori specialistici, come quello editoriale e documentale, la diffusione di *XML* è sin da subito più graduale e ragionata, con una sostituzione progressiva di *SGML*, tranne in quei pochi casi in cui quest'ultimo rimane ancora insostituibile a causa delle sue peculiarità. All'inizio la presenza *XML* era limitata a programmi fortemente orientati verso la documentazione tecnica e la manualistica[59], ma si è successivamente esteso, seguendo la diffusa tendenza generale, arrivando a coprire anche i *software* più generalisti, come *XPress* e *InDesign*. Secondo Darin McBeath, *chief architect* di Elsevier, i maggiori cambiamenti dell'industria editoriale degli ultimi dieci anni sono dovuti, con importanza decrescente, a *XML*, *XSLT*, *PDF*, *Java* e *Internet*[60]. In particolare, sempre per McBeath, *XML* è veramente prezioso per quel che riguarda la strutturazione dei contenuti, la messaggistica tra sistemi, la configurazione delle applicazioni e l'aderenza agli standard; inoltre numerose potenzialità rimangono ancora da esplorare, tra cui quelle legate ad *XQuery*, che permette di manipolare i contenuti documento-centrici in maniera simile a quella resa possibile dall'impiego di un database.

Standard come *TEI*, *DocBook* e *EAD*[61] sono rapidamente passati da *SGML* ad *XML* approfittando così di un maggior supporto e disponibilità di programmi[62]. In ambito più propriamente biblioteconomico, o *library and information science* se si preferisce la definizione anglosassone, l'impiego di *XML* è stato definito un

[58] Per un'introduzione ai *web service* vedi la definizione di *Wikipedia* <http://en.wikipedia.org/wiki/Web_services> e Venu Vasudevan, *A web services primer*. <http://webservices.xml.com/pub/a/ws/2001/04/04/webservices/index.html>.
Senza scendere troppo nei dettagli e non considerando le differenze tra i diversi approcci, come *REST* o *XML-RPC*, *XML* nei *web service* viene usato sia per la messaggistica, tramite *SOAP*, sia per la descrizione dei metodi disponibili, con *WSDL*, sia per la creazione di registri pubblici, con *UDDI*.

[59] Come *FrameMaker*, attualmente della *Adobe* <http://www.adobe.com/it/products/framemaker/>. *FrameMaker* rimane ancora il prodotto di punta per le pubblicazioni tecniche, grazie anche al supporto del già citato *DocBook* o del più recente *DITA*, *Darwin Information Typing Architecture*, vedi *Introduction to the Darwin Information Typing Architecture* <http://www.ibm.com/developerworks/xml/library/x-dita1/>.

[60] <http://www.oreillynet.com/xml/blog/2006/12/xml_conf_2006_second_day.html>.

[61] *Encoded Archival Description* <http://www.loc.gov/ead/>, usato per le registrazioni archivistiche.

[62] Un *framework* molto interessante è *XTF*, sviluppato dalla *California Digital Library* <http://www.cdlib.org/inside/projects/xtf/>. Basato su *Java* e totalmente *open source*, *XTF* sembra essere un ottimo (e più economico) sostituto per il già citato *DynaText/DynaWeb*.

«marriage made in heaven»[63]. Questo è evidente per ciò che riguarda il paradigma della *digital library*[64], dove diverse specifiche sono state create *ex novo* utilizzando sin da subito questa sintassi, vedi ad esempio *METS*[65], o il protocollo *OAI-PMH*[66], in quanto le potenzialità offerte si combinano perfettamente alle necessità insite delle biblioteche digitali, tra cui l'ormai ben noto e mai troppo poco professato "mantra dell'interoperabilità"[67]. Diverso è invece il caso degli OPAC, in quanto i cataloghi elettronici, essendo maggiormente e storicamente basati sui database relazionali, sembrano a prima vista dotati di maggiori anticorpi nei confronti di una penetrazione di *XML*. Anche qui però molte cose si stanno muovendo, con la ridefinizione degli standard catalografici[68] e la creazione di nuove iniziative, come il *DLF Aquifer*[69].

Cercare di effettuare una panoramica globale degli usi di *XML* è estremamente arduo se non addirittura impossibile, ed è molto probabile che il risultato sia confusionario e controproducente, dato l'elevato numero e l'estrema eterogeneità dei casi da considerare. Più fattibile e interessante è invece cercare di individuare i settori più peculiari e/o strategici, anche nell'ottica di una prospettiva di medio e lungo termine.

Ai primi posti di questa lista c'è indubbiamente il rapporto tra *XML* e database, già accennato di sfuggita in relazione ai cataloghi *on line*. I due modelli implementati

[63] La *mailing list Xml4Lib Electronic Discussion* <http://lists.webjunction.org/xml4lib/> è espressamente dedicata a questo argomento. Vedi inoltre Roy Tennant (ed.), *Xml in libraries*. Neal-Schuman Publishers, 2002 oppure Dick R. Miller - Kevin S. Clarke, *Putting XML to work in the library: tools for improving access and management*. American Library Association, 2003. In lingua italiana Daniela Canali, *Standard per la biblioteca digitale. Nuovi linguaggi di codifica per l'informazione bibliografica*. Editrice Bibliografica, 2006.

[64] Vedi *Standards at the Library of Congress* <http://www.loc.gov/standards/> e le presentazioni disponibili *on line* del recente convegno *Using metadata standards in digital libraries: implementing METS, MODS, PREMIS and MIX* dell'American Library Association <http://www.loc.gov/standards/mods//presentations/litaprogram-an2007.html>.

[65] *Metadata Encoding & Transmission Standard* <http://www.loc.gov/standards/mets/>. Recentemente è stata resa disponibile la prima bozza di un manuale introduttivo, *METS primer*, realizzato dalla Digital Library Federation <http://www.loc.gov/standards/mets/METS%20Documentation%20draft%20070310p.pdf>.

[66] *Open Archives Initiative Protocol for Metadata Harvesting* <http://www.openarchives.org/OAI/openarchivesprotocol.html>.

[67] Si ringrazia Michelle Dalmau, dell'*Indiana University Digital Library Program*, per aver suggerito questa definizione.

[68] In particolare *MODS* <http://www.loc.gov/standards/mods/> e *MARCXML* <http://www.loc.gov/standards/marcxml/>.

[69] <http://www.diglib.org/aquifer/>. Vedi anche le relative *Guidelines* <http://www.diglib.org/aquifer/dlfmodsimplementationguidelines_finalnov2006.pdf> e la presentazione *Creating rich shareable metadata: the DLF Aquifer MODS implementation guidelines* <http://www.loc.gov/standards/mods/presentations/shreeves-ala07/>.

sono estremamenti diversi tra di loro, gerarchico l'uno e relazionale l'altro [70]. I database sono ancora lo strumento principale per gestire l'informazione strutturata, e quasi sicuramente manterranno questo ruolo ancora per molto, data la maggiore anzianità sul campo, la disponibilità di prodotti, la diffusione e familiarità tra gli sviluppatori, ma soprattutto per le questioni legate alla *performance*, sia rispetto alla quantità di dati che è possibile memorizzare, sia alla velocità delle operazioni. Da un lato però la maggior parte dei database relazionali è ormai *XML-enabled*, in grado di gestire questo formato mappandolo con il proprio modello e accettandolo come formato di *input* e *output*, dall'altro negli ultimi anni si è assistito ad una diffusione sempre maggiore dei cosiddetti *native XML database*, basi di dati il cui modello interno è proprio quello gerarchico [71], in quanto questa tipologia si adatta maggiormente ai contenuti presenti sul web rispetto ai "cugini" relazionali. Tra i database *XML* nativi *open source* molto interessante è il caso di *eXist* [72], caratterizzato da una comunità di sviluppatori ed utenti estremamente dinamica. *eXist* supporta la manipolazione dei documenti memorizzati tramite *XQuery*, che, combinato in cascata con *XSLT*, permette così di realizzare numerose applicazioni senza nessun altro ausilio, o naturalmente di essere integrato in architetture *software* più ampie laddove ce ne sia bisogno. Come scritto precedentemente, *XQuery* è una delle tecnologie più interessanti ed è dotata di numerose potenzialità, soprattutto per quel che riguarda la possibilità di essere utilizzato con informazione sia data-centrica sia documento-centrica, l'integrazione di fonti di dati eterogenee tra di loro e la programmazione lato server, al posto di linguaggi molto più complicati [73]. Alcune delle applicazioni "real world", in cui *XQuery* gioca un ruolo chiave e caratterizzate da un approccio innovativo per quel che riguarda la gestione dei contenuti, sono *Scopus* [74], *SafariU* [75] e *O'Reilly Labs* [76].

[70] A cui andrebbe aggiunto il paradigma *object-oriented* dei linguaggi di programmazione, vedi David Mertz, *XML matters: putting XML in context with hierarchical, relational, and object-oriented models*. <http://www.ibm.com/developerworks/library/x-matters8/index.html>.

[71] Sia per gli *XML-enabled* sia per i *native XML database* le caratteristiche variano molto da prodotto a prodotto. Vedi Ronald Bourret, *XML database products* <http://www.rpbourret.com/xml/XMLDatabaseProds.htm> o lo speciale di "Computer Programming", n. 140 (maggio 2005), Infomedia.

[72] <http://exist.sourceforge.net/>.

[73] Vedi Kurt Cagle, *XQuery, the server language* <http://www.xml.com/pub/a/2007/06/01/xquery-the-server-language.html> e *XQuery and data abstraction* <http://www.xml.com/pub/a/2007/07/12/xquery-and-data-abstraction.html>.

[74] Una base di dati di citazioni e *abstract* di letteratura scientifica che permette l'accesso a ben 50 milioni di record <http://www.scopus.com>.

[75] Tramite *SafariU* è possibile creare un libro di testo personalizzato aggregando liberamente quelli già presenti <http://www.safariu.com/>.

[76] <http://labs.oreilly.com/>.

Sempre nel mondo del Web, ma stavolta dalla parte dei client, non è possibile non citare la novità del momento: *AJAX* [77], una tecnica di programmazione in cui l'impiego combinato di *Javascript* e *XML* permette la creazione delle cosiddette *rich Internet application*, applicazioni web che per interfaccia e funzionalità si avvicinano ai tradizionali programmi *desktop*, grazie ad una serie di chiamate asincrone tra la pagina web ed il server, in grado di aggiornare i contenuti della pagina senza dover ricaricare totalmente la pagina. Il ruolo di *XML*, sebbene siano disponibili delle alternative [78], è naturalmente fondamentale e, nel caso in cui il *front-end* dell'applicazione non sia in *HTML*, dinamico o meno, ma bensì in *Flash*, è probabile che sia stato utilizzato anche per realizzare l'interfaccia, tramite uno dei vari formati disponibili, come *MXML* [79] per *Adobe Flex* o *LZX* per *OpenLaszlo* [80]. La separazione dell'interfaccia utente dalla logica operativa è una regola presente da tempo nella programmazione, ma la possibilità di definirla tramite un linguaggio di marcatura aggiunge un livello di astrazione in più, rivelatosi molto utile [81]. Gli esempi riportati fino ad ora, e riguardanti principalmente *Java*, *Javascript* e *Flash*, denotano come tre tecnologie estremamente diverse tra di loro abbiano ottenuto numerosi beneficî grazie ad un utilizzo congiunto con *XML*, che in molti casi, vedi *Flash*, li ha rivitalizzati aprendo la porta a nuove possibilità.

In un ambito diverso, relativo all'*office automation*, il passaggio dai formati binari proprietari a degli standard come *OpenDocument*, da parte della *suite OpenOffice*, e *OpenOfficeXML*, da parte di *Microsoft Office* [82], segna una svolta importante, in quanto rende maggiormente accessibile quell'enorme numero di documenti di testo, fogli di calcolo e presentazioni che vengono prodotti quotidianamente dai computer di tutto il mondo.

XML è sopravvissuto al suo stesso *hype*, non è più una novità affascinante e *trendy* ma è ormai stabile, affermato, estremamente diffuso, dotato di una pervasività sempre maggiore, sia a livello orizzontale sia verticale; ha dimostrato su larga scala l'utilità dei linguaggi di marcatura e al momento è difficile farne a meno. Sul suo utilizzo si è passati in poco tempo dal "perché" al "perché non". Come saranno i prossimi dieci anni? Verrà sostituito da un altro standard, così come è successo ad *SGML*, che

[77] Acronimo di *Asynchronous JavaScript and XML* <http://en.wikipedia.org/wiki/Ajax_%28programming%29>.

[78] Come ad esempio *JSON*, basato su una notazione *Javascript* <http://en.wikipedia.org/wiki/JSON>.

[79] <http://www.adobe.com/it/products/flex/>.

[80] <http://www.openlaszlo.org/>.

[81] Tra i principali *User MarkUp Interface Language* ci sono *XUL* della *Mozilla Foundation* <http://www.mozilla.org/projects/xul/> e *XAML* per il *Framework .NET* <http://www.xaml.net/>.

[82] *OpenOffice* è stato molto più rapido ad adottare *OpenDocument*, che oltretutto è uno standard *ISO*, mentre *OpenOfficeXML* è uno standard *ECMA* <http://www.ecma-international.org/>.

ne eliminerà i difetti, come l'eccessiva ridondanza o il problema delle gerarchie sovrapposte [83]? E, se sì, cosa succederà a tutti quei file che verranno prodotti fino a quel momento, saranno facilmente convertibili? Oppure la sua sintassi di base resterà sempre quella, mentre l'universo di tecnologie collegate continuerà ad espandersi e ad aggiornarsi? Vedremo mai un *XML 2.0* o una sua serializzazione binaria? I cambiamenti portati da *XML* sono sotto gli occhi di tutti, ed è probabile che quello a cui abbiamo assistito sino ad ora sia solamente la punta dell'iceberg. In ogni caso quello che non dovrà mai mancare è una riflessione continua su quelli che sono i risultati da raggiungere e i principî da seguire, tenuti ben distinti dal mezzo impiegato in un determinato intervallo temporale. L'interoperabilità, la condivisione e l'interscambio dei dati, lo sviluppo di standard aperti e l'indipendenza da una particolare piattaforma hanno indubbiamente ricevuto un impulso enorme, e dovranno continuare ad essere le stesse linee guida che dieci anni fa spinsero verso la nascita del linguaggio di marcatura estensibile ormai noto in tutto il mondo [84].

[83] Vedi il sito web *XML Sucks* <http://xmlsucks.org/>.

[84] Vedi Uche Ogbuji, *Thinking XML: the XML decade* <http://www-128.ibm.com/developerworks/library/x-think38.html>; *Celebrating 10 years if XML*. "IBM Systems Journal", 45 (2006), n. 2 <http://www.research.ibm.com/journal/sj45-2.html>.