

# XML – stan obecny i trendy rozwojowe

*Tomasz Traczyk*

Politechnika Warszawska

Od kilku lat XML jest jedną z najbardziej popularnych technologii informatycznych, stosowaną szeroko m.in. dzięki swojej prostocie i dostępności narzędzi, a przede wszystkim dzięki dobrej standaryzacji. Niektóre związane z XML standardy (np. XPath, XSLT) są znane i szeroko stosowane, inne (np. XLink, XSL-FO) jeszcze nie zdobyły oczekiwanej popularności. Wciąż też trwają prace nad kolejnymi standardami z „rodziny” XML, a istniejące standardy są udoskonalane.

Pierwsza część referatu przypomni podstawowe fakty związane z językiem XML, rolę poszczególnych standardów XML-owych oraz podsumuje obecne osiągnięcia w dziedzinie ich implementacji. Krótko omówione zostaną typowe zastosowania XML. W drugiej części referatu zaprezentowane będą najnowsze prace związane z rozwojem „rodziny” XML. Polegają one zarówno na udoskonalaniu i dopracowywaniu istniejących standardów (np. XPath, XPointer, XSLT), jak i na opracowywaniu nowych (np. XML Encryption, XML Signature, XForms).

## Informacja o autorze:

Dr inż. Tomasz Traczyk jest adiunktem w Instytucie Automatyki i Informatyki Stosowanej Politechniki Warszawskiej.

## 1. Czym dziś jest XML?

XML – metajęzyk umożliwiający tworzenie specjalizowanych języków znakowania – stał się w ostatnich latach jedną z najważniejszych technologii informatycznych. Pomyślany początkowo jako język wymiany informacji w sieci WWW, zdobył szersze zastosowania. Stał się też składnikiem wielu ważnych i modnych współczesnych technologii.

### 1.1. Główne cechy XML

XML ma szereg ważnych cech, dzięki którym jest wygodnym i uniwersalnym środkiem zapisu i wymiany informacji. Wśród zalet XML wymienić można:

- Sformalizowany zapis informacji – umożliwiający jej bezstratne odczytanie, dokładną weryfikację poprawności i łatwe dalsze przetwarzanie.
- Uniwersalność – XML pozwala zapisać wszelkie informacje, które mogą być wyrażone tekstowo.
- Dużą siłę wyrazu – za pomocą XML można zapisać nawet bardzo złożone struktury informacji.
- Elastyczność – struktura informacji jest łatwa do rozszerzania i dostosowywania, z możliwością wykorzystania struktur wcześniej zdefiniowanych.
- Możliwość zróżnicowanej prezentacji – dzięki zastosowaniu arkuszy stylistycznych ten sam dokument może być różnorodnie prezentowany w zależności od potrzeb i możliwości.
- Łatwość przetwarzania – dzięki prostej i regularnej składni oraz ustandaryzowanym narzędziom.
- Czytelność – dokumenty XML są zrozumiałe dla człowieka, każdy dokument niesie bowiem metainformację w postaci znaczników. Ułatwia to posługiwanie się dokumentami, uruchamianie oprogramowania itd.
- Dostosowanie do specyfiki przetwarzania w sieci Web – liczne szczegółowe rozwiązania XML są dostosowane do przetwarzania w Sieci. Popularne narzędzia internetowe (przeglądarki, serwery aplikacyjne itd.) wyposażono w rozwiązania (parsery, procesory XSLT itp.) ułatwiające użycie XML.
- Internacjonalizację – XML może używać wielu różnych stron kodowych, a przetwarzanie XML odbywa się z użyciem UTF-8.
- Niewygórowane koszty – do przetwarzania i prezentacji dokumentów XML można użyć standardowego (często darmowego) oprogramowania. Można zatem tworzyć rozwiązania bazujące na XML nie ponosząc wielkich nakładów.
- Względną prostotę – idea języków znakowania jest powszechnie znana, a nawet zaawansowane rozwiązania są stosunkowo łatwe do zrozumienia.

Wśród wad XML najistotniejsze są:

- Ograniczenia wynikające z hierarchicznej struktury danych – w strukturach takich w naturalny sposób przedstawić można jedynie jeden typ powiązań pomiędzy dwoma typami obiektów. Jeśli niezbędne jest zapisanie większej różnorodności powiązań, to trzeba stosować różne sztuczki – w XML z reguły stosuje się identyfikatory elementów i odesłania do nich, co jest nieco podobne do typowych rozwiązań relacyjnych, ale nie ma wiele wspólnego z hierarchiczną strukturą dokumentu.

- Rozwlekłość zapisu – decyduje o niej znaczny narzut na znaczniki niosące metainformację. Nie jest to na ogół wadą istotną, np. przy przesyłaniu informacji stosować można kompresję, która jest zwykle bardzo efektywna.
- Problemy z wydajnością przetwarzania – wynikające zarówno z rozwlekłości plików jak ze stosowania uniwersalnych (nieoptymalnych dla konkretnych zastosowań) narzędzi. Dla małych dokumentów ograniczenia wydajności zwykle nie stanowią problemu, dla dokumentów większych stosowanie odpowiedniego typu parserów (np. SAX) daje na ogół wydajność dostatecznie dobrą.
- Niedostatki implementacji standardów XML-owych, zwłaszcza niepełna implementacja standardów w przeglądarkach WWW – co powoduje, że XML nie może wciąż jeszcze być stosowany jako podstawowy język wymiany informacji w sieci Web.

## 1.2. Przyczyny popularności XML

Nadzwyczajna kariera XML wydaje się mieć wiele przyczyn, wśród których najważniejsze są zapewne:

- szeroka akceptacja dla koncepcji języków znakowania (SGML ma długą tradycję, HTML jest zaś powszechnie znany);
- rozwój koncepcji *e-business* (można tu nawet mówić o rodzaju mody);
- związana z tym silnie odczuwana potrzeba środków sformalizowanej wymiany informacji w Sieci (HTML nie spełnia oczywiście tej potrzeby w stosunku do informacji, która ma być dalej przetwarzana automatycznie).

Sam w sobie XML nie wnosi żadnych rewolucyjnych rozwiązań: idea języków znakowania jest dość stara, koncepcja hierarchicznych struktur danych jeszcze starsza. Jednak, jak się wydaje, XML „podał” te koncepcje w nowym, atrakcyjnym dla użytkowników opakowaniu; pojawił się też w odpowiednim momencie, trafiając na wielkie zapotrzebowanie. To co jest w XML istotnie nowe, to szeroka akceptacja standardów oraz duża dostępność tanich (lub zgoła darmowych) narzędzi.

Do kariery XML istotnie przyczynili się też główni wytwórcy oprogramowania, którzy szybko dostrzegli możliwości tej technologii i włączyli do swoich popularnych produktów elementy wsparcia dla niej, poważnie traktując potrzebę zgodności swych produktów z XML-owymi standardami.

Pozycja XML szybko się ugruntowała i nie wydaje się zagrożona, a obszar jego zastosowań wciąż rośnie.

## 1.3. Typowe zastosowania XML

Jak się wydaje, intencją twórców XML było początkowo stworzenie języka, który mógłby zastąpić HTML jako narzędzie do tworzenia stron WWW w tych zastosowaniach, w których niezbędna jest większa formalizacja przekazu informacji, gdyż informacja ma być dalej przetwarzana. Tymczasem jednak rzeczywiste zastosowania XML ukształtowały się inaczej, przede wszystkim koncentrując się wokół elektronicznej wymiany danych.

Ważniejsze typowe zastosowania XML krótko scharakteryzowano poniżej.

### 1.3.1. Tworzenie stron internetowych

XML ciągle nie jest zbyt popularny w tym zastosowaniu, króluje tu wciąż HTML. Strony w XML spotyka się tylko w zastosowaniach specjalistycznych, głównie związanych z gospodarką elektroniczną. Wyjątek stanowi komunikacja mobilna – w świecie telefonów komórkowych powszechnie używany jest WML, który jest dialektem XML. Nową jakość stworzy dopiero popula-

ryzacja XHTML. Ta nowa wersja HTML, opracowana przez W3C, jest także dialektem XML, co pozwala na użycie wszystkich standardów i narzędzi XML-owych do tworzenia i przetwarzania stron WWW. Ale do upowszechnienia XHTML droga wydaje się jeszcze dość daleka.

**1.3.2. Opis zasobów** Sprawne wyszukiwanie informacji w Sieci jest, jak wiadomo, trudne. Jedną z głównych przyczyn jest brak dających się przetwarzać maszynowo opisów (deskryptorów) zasobów dostępnych w Internecie. XML nadaje się bardzo dobrze do tworzenia takich, czytelnych i łatwych w przetwarzaniu, deskryptorów zasobów sieciowych, oprogramowania itp. Spośród propozycji standardów tego typu opisów najważniejsze są RDF (*Resource Description Framework*) oraz OWL (*Web Ontology Language*).

Specyficzny rodzaj opisu zasobów zapewnia WSDL (*Web Services Description Language*). Jest to język opisu usług sieciowych (*Web services*), pozwalający opisywać owe usługi i sposób ich wywoływania.

### 1.3.3. Reprezentacja informacji semistrukturalnej

XML jest niezastąpionym środkiem reprezentowania informacji semistrukturalnej, tzn. takiej, gdzie informacje mają pewną strukturę, ale jest ona zmienna, słabo ustalona czy też częściowo nieistotna. Typowym przykładem takiej informacji są złożone dokumenty tekstowe, mające wprawdzie strukturę akapitów, podrozdziałów i rozdziałów, ale słabo sformalizowaną. Tego typu informacje dają się bez większych trudności zapisywać w XML, zaś bardzo źle reprezentuje się je w relacyjnych bazach danych. XML może tu zatem stanowić dobre uzupełnienie możliwości typowych baz danych.

### 1.3.4. Multimedia

XML znalazł dość szerokie zastosowania związane z multimediami, w co najmniej dwóch aspektach: bezpośredniego zapisu informacji multimedialnej, np. grafiki wektorowej (*Scalable Vector Graphics* – SVG) oraz sterowania przetwarzaniem informacji multimedialnej (np. SMIL czy Voice-ML).

**1.3.5. Specjalistyczne struktury danych** Jednym z pierwszych zastosowań XML – w pełni zgodnym z pierwotnymi intencjami twórców języka – jest tworzenie specjalistycznych struktur do przekazywania informacji naukowej, ekonomicznej itp. w społeczności specjalistów z danej dziedziny. Powstają zarówno specjalistyczne dialekty, mające charakter środowiskowych standardów (np. MathML do zapisu wzorów matematycznych, CML – *Chemical Markup Language*), jak i struktury tworzone *ad-hoc* na potrzeby konkretnych organizacji czy projektów.

### 1.3.6. Komunikacja w sferze publicznej

Specjalną rolę XML może i powinien odegrać w komunikacji w sferze publicznej, zwłaszcza w wymianie informacji między obywatelem czy przedsiębiorstwem a urzędem. Zastosowanie do takiej wymiany otwartego standardu jakim jest XML likwiduje przewagę, którą uzyskują firmy tworzące oprogramowanie dla urzędów. Istnienie takiej przewagi, a co za tym idzie przymusowe związanie obywateli czy przedsiębiorstw z określonym dostawcą oprogramowania, jawnie kłóci się z zasadami wolności gospodarczej i w praworządym państwie nie powinno mieć miejsca (na gruncie krajowym wspomnieć tu można sprawy formatu wymiany danych z ZUS czy gromadzenia danych przez b. Kasy Chorych). Tymczasem w XML informacje wymieniać może nawet podmiot nie dysponujący specjalistycznym oprogramowaniem; do utworzenia odpowiedniego pliku danych wystarczać bowiem powinien dowolny edytor tekstowy, jeśli tylko format danych jest odpowiednio dobrze udokumentowany i opublikowany. Tego typu próby były (i – miejmy nadzieję – nadal będą) podejmowane, przykładem może być projekt formatu gromadzenia danych medycznych [MZ01], ostatecznie porzucony z przyczyn dalekich od merytorycznych.

**1.3.7. Elektroniczna wymiana danych (EDI)** Mającym obecnie największe znaczenie zastosowaniem XML jest, jak się wydaje, elektroniczna wymiana danych i dokumentów (EDI). XML idealnie nadaje się do wymiany dokumentów za pośrednictwem sieci Internet, np. w zastosowaniach z dziedziny handlu elektronicznego typu B2B (*business to business*) czy przy integracji systemów heterogenicznych [Tra02a]. Ze względu na swą prostotę oraz dostępność narzędzi, XML zapewne wyprze klasyczne technologie i standardy EDI.

W dziedzinie EDI istnieje znaczny dorobek standaryzacyjny co do treści dokumentów-komunikatów dotyczących różnych dziedzin życia. Koncepcja XML/EDI zakłada zachowanie tego dorobku przez zawarcie semantyki wcześniej ustandaryzowanych komunikatów (np. EDI-FACT) w składni XML.

**1.3.8. Konfiguracja oprogramowania** XML stał się bardzo popularny jako format zapisu różnego rodzaju plików konfiguracyjnych oprogramowania. XML jest prosty, łatwy w interpretacji przez użytkownika, a do interpretacji zawartości plików konfiguracyjnych użyć można standardowych parserów, co ułatwia pracę programistom. Przykładem takiego zastosowania mogą być deskryptory wdrożeniowe w technologii EJB.

**1.3.9. Protokoły komunikacyjne** XML znalazł także zastosowanie w różnego rodzaju protokołach wymiany komunikatów, zdalnego wywoływania procedur itp. Jego zastosowanie zapewnia czytelność przesyłanych komunikatów, umożliwia też zastosowanie standardowych parserów XML do interpretacji tych komunikatów. Przykładami tego typu zastosowań są protokoły SOAP, XML-RPC i Web-DAV.

## 2. Stan obecny technologii XML

Sam standard XML 1.0 jest dość prosty, ale nie rozwiązuje wielu ważnych problemów dotyczących definiowania struktur dokumentów, ich przetwarzania, prezentacji itd. Technologia XML-owa obrasta więc w kolejne, uzupełniające ją, standardy, stanowiące sporą, ale spójną rodzinę.

### 2.1. Standaryzacja XML

Standaryzacją XML od początku zajmuje się organizacja *World Wide Web Consortium* (W3C). Poszczególne standardy związane z XML opracowuje kilkanaście działających w jej ramach grup. Wczesne propozycje standardów są publikowane w postaci tzw. *working drafts*; następnie – już blisko zatwierdzenia – ogłaszane są tzw. *proposed recommendation*; zatwierdzone standardy publikowane są jako *recommendation*. Choć W3C nie ma statusu oficjalnej organizacji standaryzacyjnej (jak np. ISO), to jednak jej rekomendacje są powszechnie uważane za obowiązujące. O ile w przypadku HTML i innych standardów internetowych producenci oprogramowania dość „swobodnie” traktują standardy i implementacje niejednokrotnie dość daleko od nich odbiegają, o tyle w przypadku XML przyznać trzeba, iż nawet najsilniejsi producenci traktują standaryzację poważnie i co najwyżej nieco rozszerzają standardy, starają się też nadażyć za biegiem prac standaryzacyjnych prowadzonych przez W3C. Jest to o tyle zrozumiałe, iż jedną z głównych zalet XML ma być niezależność od platformy programowej, co jest szczególnie ważne w zastosowaniach z dziedziny elektronicznej wymiany danych, zaś wszelkie odstępstwa od standardów mogłyby łatwo tę zaletę zniweczyć.

Poszczególne składniki rodziny XML znajdują się na różnych etapach standaryzacji: niektóre z szerzej przyjętych standardów (np. XPath czy XSLT) doczekały się już prac nad kolejnymi, udoskonalonymi wersjami; inne standardy (np. XPointer) wciąż nie mogą osiągnąć dojrzałości. Stan prac standaryzacyjnych przedstawia poniższa tabela.

Tabela 1. Ważniejsze standardy związane z XML

Nazwa	Wersja	Etap standaryzacji	Źródło
Extensible Markup Language (XML)	1.0	Recommendation	<a href="http://www.w3.org/TR/2000/REC-xml-20001006">http://www.w3.org/TR/2000/REC-xml-20001006</a>
XML Base	1.0	Recommendation	<a href="http://www.w3.org/TR/2001/REC-xmlbase-20010627">http://www.w3.org/TR/2001/REC-xmlbase-20010627</a>
XML Namespaces	1.0	Recommendation	<a href="http://www.w3.org/TR/REC-xml-names">http://www.w3.org/TR/REC-xml-names</a>
	1.1	Candidate Recommendation	<a href="http://www.w3.org/TR/xml-names11">http://www.w3.org/TR/xml-names11</a>
XML Linking Language (XLink)	1.0	Recommendation	<a href="http://www.w3.org/TR/2000/REC-xlink-20010627">http://www.w3.org/TR/2000/REC-xlink-20010627</a>
XML Pointer Language (XPointer)	–	Fragmenty na różnych etapach prac	<a href="http://www.w3.org/XML/Linking">http://www.w3.org/XML/Linking</a>
XML Schema (3 części)	–	Recommendation	<a href="http://www.w3.org/TR/2001/REC-xmlschema-n-20010502">http://www.w3.org/TR/2001/REC-xmlschema-n-20010502</a> gdzie $n \in \{1,2,3\}$
XML Path Language (XPath)	1.0	Recommendation	<a href="http://www.w3.org/TR/1999/REC-xpath-19991116">http://www.w3.org/TR/1999/REC-xpath-19991116</a>
	2.0	Working Draft	<a href="http://www.w3.org/TR/2003/WD-xpath20-20030822">http://www.w3.org/TR/2003/WD-xpath20-20030822</a>
XSL Transformations (XSLT)	1.0	Recommendation	<a href="http://www.w3.org/TR/1999/REC-xslt-19991116">http://www.w3.org/TR/1999/REC-xslt-19991116</a>
	2.0	Working Draft	<a href="http://www.w3.org/TR/2003/WD-xslt20-20030502">http://www.w3.org/TR/2003/WD-xslt20-20030502</a>
Extensible Stylesheet Language (XSL)	1.0	Recommendation	<a href="http://www.w3.org/TR/2001/REC-xsl-20011015">http://www.w3.org/TR/2001/REC-xsl-20011015</a>
Document Object Model (DOM) Levels 1 & 2	1.0	Recommendation	<a href="http://www.w3.org/DOM/DOMTR">http://www.w3.org/DOM/DOMTR</a>
Document Object Model (DOM) Level 3	–	Working Draft in Last Call	<a href="http://www.w3.org/DOM/DOMTR#dom3">http://www.w3.org/DOM/DOMTR#dom3</a>
XML-Signature	–	Recommendation	<a href="http://www.w3.org/TR/2002/REC-xmlsig-core-20020212">http://www.w3.org/TR/2002/REC-xmlsig-core-20020212</a>
XML Encryption	–	Recommendation	<a href="http://www.w3.org/TR/2002/REC-xmlenc-core-20021210">http://www.w3.org/TR/2002/REC-xmlenc-core-20021210</a>
XML Key Management Specification (XKMS)	2.0	Working Draft	<a href="http://www.w3.org/TR/2003/WD-xkms2-20030418">http://www.w3.org/TR/2003/WD-xkms2-20030418</a>
XML Query Language (XQuery)	1.0	Working Draft	<a href="http://www.w3.org/TR/2003/WD-xquery-20030822">http://www.w3.org/TR/2003/WD-xquery-20030822</a>
Extensible HyperText Markup Language (XHTML)	1.0	Recommendation	<a href="http://www.w3.org/TR/2000/REC-xhtml1-20000126">http://www.w3.org/TR/2000/REC-xhtml1-20000126</a>
	1.1	Recommendation	<a href="http://www.w3.org/TR/xhtml11">http://www.w3.org/TR/xhtml11</a>
	2.0	Working Draft	<a href="http://www.w3.org/TR/xhtml2">http://www.w3.org/TR/xhtml2</a>
XForms	1.0	Proposed Recommendation	<a href="http://www.w3.org/TR/2003/PR-xforms-20030801">http://www.w3.org/TR/2003/PR-xforms-20030801</a>

## 2.2. Ważniejsze elementy rodziny XML

Standard XML 1.0 definiuje sam język XML oraz określa jeden ze sposobów deklarowania budowy dokumentów XML, tzw. DTD. Jednak by z XML-a można było rzeczywiście korzystać, niezbędne są liczne dodatkowe standardy, opisujące sposób formułowania w XML pewnych typowych informacji, sposoby przetwarzania XML itd.

### 2.2.1. Adresy URI i XML Base

Adresy sieciowe używane w XML mają postać tzw. URI (*Unified Resource Identifiers*). URI może być po prostu adresem URL, ale może też być nazwą (URN – *Unified Resource Name*) nie mającą znaczenia adresu (tzn. jedynie identyfikującą zasób, a nie wskazującą jego lokalizacji).

Uzupełnieniem adresowania jest możliwość ustalenia bazowego URI, w stosunku do którego podawane są adresy względne. Standard XML Base definiuje specjalny atrybut `xml:base`, służący właśnie do tego celu, podobny do elementu `BASE` w HTML.

### 2.2.2. Przestrzenie nazw – XML Namespaces

Ponieważ w XML tworzone są liczne specjalizowane dialekty, istnieje znaczne prawdopodobieństwo, że to samo słowo zostanie w różnych dialektach użyte do nazwania elementów czy atrybutów mających w nich odmienne znaczenie. Gdyby zaistniała potrzeba stworzenia nowego dialektu, łączącego kilka już istniejących, to powstanie ryzyko wystąpienia konfliktu nazw.

Aby takie ryzyko usunąć, stworzono pojęcie przestrzeni nazw. Twórca dialektu XML może umieścić tworzone przez siebie nazwy (terminy owego dialektu) w przestrzeni nazw. Gdy nazwy te będą użyte w dokumencie, przestrzenie z których nazwy owe pochodzą mogą być rozróżniane za pomocą swych identyfikatorów (zapisywanych jako URI) i związanych z nimi prefiksów. Pozwala to tworzyć dokumenty łączące terminy z wielu dialektów, bez ryzyka wystąpienia niejasności co do tego, z którego dialektu pochodzi dany termin.

Przestrzenie nazw stały się szybko jednym z podstawowych elementów technologii XML i są powszechnie używane; korzysta z nich wiele innych standardów XML-owych.

### 2.2.3. Schematy XML – XML Schema

Ważny, choć historycznie dość późny, jest standard XML Schema, który określa sposób deklarowania składni konkretnych dokumentów XML (czyli „dialektów” tworzonych za pomocą XML). Schemat jest zapisany także w XML; określa budowę dokumentu (nazwy, następstwo i zawieranie się elementów oraz ich atrybuty), a także typy danych zawartych w elementach i atrybutach. Schematy zastępują wcześniej używane – a pochodzące jeszcze z SGML – DTD (*Document Type Definitions*), mając nad DTD liczne przewagi. Za pomocą schematów można m.in. precyzyjnie definiować typy danych, używając rozbudowanego słownika typów elementarnych oraz tworząc własne typy. Schematy są tak skonstruowane, by można było w sposób sformalizowany tworzyć nowe schematy (a więc nowe dialekty XML), używając schematów wcześniej istniejących. W ten sposób można np. połączyć informacje z kilku schematów, rozszerzyć istniejący schemat o nowe dane czy też zawęzić schemat przez narzucenie dodatkowych warunków.

Istnienie dobrego języka opisu składni dokumentu XML jest niezwykle ważne. Jeśli bowiem język taki pozwala na precyzyjne określenie budowy poprawnego dokumentu wraz z silnym systemem typów danych, to zdecydowanie upraszcza się przetwarzanie dokumentów w programach. Programista bowiem może ograniczyć się do analizy jedynie dokumentów poprawnych, a kontrolę poprawności wykonywać może typowa biblioteczna procedura walidująca, korzystająca ze schematu. XML Schema nadaje się do tego typu walidacji, dostarczając silnego typowania oraz możliwości zapisu prostych warunków poprawności danych (np. przedziałów dopuszczalnych wartości czy dopasowywania wartości do wyrażenia regularnego). Niestety, bardziej skomplikowanych

warunków poprawności (np. przypominających więzy *check* znane z baz danych) nie daje się w tym języku wyrazić. Powstały konkurencyjne w stosunku do XML Schema rozwiązania (np. Schematron), mające takie możliwości; można się zatem spodziewać, iż kolejne wersje standardu będą rozszerzane w podobnym kierunku.

Standard XML Schema został szybko zaakceptowany i pojawiły się liczne jego implementacje we wszystkich ważniejszych narzędziach służących do przetwarzania XML.

#### 2.2.4. Arkusze stylistyczne i przekształcenia – XSLT

Ponieważ jednym z głównych założeń dotyczących użycia XML jest tworzenie dialektów odzwierciedlających semantykę opisywanego zagadnienia, należy przyjąć, iż prawidłowe znakowanie w XML ma zawsze charakter znaczeniowy, a nie typograficzny (jest to zresztą warunek łatwości automatycznego przetwarzania zawartości). W dodatku, ponieważ XML nie ma żadnych predefiniowanych znaczników, nie może też mieć określonych żadnych predefiniowanych sposobów prezentacji. Do prezentowania dokumentów XML konieczne jest więc dostarczenie dodatkowej informacji na temat pożądanego wyglądu – czyli tzw. arkusza stylistycznego (*stylesheet*). Zbudować takie arkusze można przy używając języka CSS i przypisując określony wygląd poszczególnym znacznikom. Jednak najczęściej także struktura znaczników – odzwierciedlająca logiczną strukturę informacji – wcale nie jest odpowiednia do bezpośredniej prezentacji i konieczna jest jej transformacja przez wyświetlenie.

Język XSL (*Extensible Stylesheet Language*) miał zaspokoić takie potrzeby: umożliwić transformację struktury wejściowego dokumentu XML na postać właściwą do prezentacji oraz opisywać sposób wyświetlania poszczególnych elementów struktury po transformacji. Ponieważ prace nad częścią dotyczącą transformacji przebiegały szybciej, a i dość prędko pojawiły się pierwsze ważne implementacje, tę część języka wydzielono w osobny standard XSLT (*XSL Transformations*).

Jak się wkrótce okazało, XSLT stał się jednym z najważniejszych składników technologii XML. Używa się go powszechnie do przekształcania dokumentów XML na HTML, co umożliwia ich prezentację we współczesnych przeglądarkach WWW. Znalazł on także szerokie zastosowanie do transformacji dokumentów XML wcale nie związanej z ich prezentacją – np. przekształcania informacji z jednego schematu XML na inny. Istnieją liczne implementacje XSLT zarówno w przeglądarkach WWW, jak i w narzędziach używanych po stronie serwerów.

#### 2.2.5. Wyszukiwanie fragmentów dokumentu – XPath

Przetwarzanie w XSLT bazuje na przeglądaniu drzewa reprezentującego wejściowy dokument, dopasowywaniu od jego fragmentów wzorców (*patterns*) zdefiniowanych w arkuszu stylistycznym i stosowaniu w stosunku do owych znalezionych fragmentów treści tzw. szablonów (*templates*). Kluczową rolę w tego rodzaju przetwarzaniu ma możliwość łatwego, a jednocześnie dającego duże możliwości, definiowania wzorców.

Właśnie do tego celu stworzono język XPath. Wzorzec jest zapisywany w postaci ścieżki, nieco podobnej do ścieżek katalogów znanych z systemów operacyjnych, co powoduje, że zapis ten jest intuicyjny i łatwy do opanowania. Ścieżka składa się ze współrzędnych opisujących położenie wyszukiwanego fragmentu w stosunku do bieżącego kontekstu, predykatów określających dodatkowe warunki oraz testu węzła, który mówi jaki fragment ma być wynikiem wyszukiwania.

XPath znalazł zastosowanie nie tylko w XSLT, jest także podstawą standardu XPointer, a język XQuery stanowi rozszerzenie XPath.

#### 2.2.6. Obiekty formatujące – XSL-FO

Pozostała po wydzieleniu XSLT część standardu XSL zajmuje się określaniem wyglądu elementów będących wynikiem transformacji. Pożądany wygląd wyraża się za pomocą tzw. obiek-

tów formatujących (*formatting objects*). Jest to dialekt XML, pozwalający opisywać formatowanie dokumentu z użyciem środków właściwych dla profesjonalnej poligrafii, takich jak złożone stronicowanie, żywa pagina, odnośniki, elementy ruchome (np. przypisy), separacje barw itp. Standard nie ogranicza się do formatowania dla potrzeb druku, przewidziano też środki odpowiednie dla prezentacji interaktywnych, a nawet głosowych!

Niestety, obiekty formatujące nie mają jeszcze implementacji w popularnych przeglądarkach WWW, co powoduje, że nadal częściej stosuje się transformację XML na HTML. Istnieją już jednak niezłe implementacje typu *server-side*.

### 2.2.7. Łączniki (XLink, XPointer)

Ponieważ XML ze swego założenia ma służyć głównie do komunikacji w Internecie, dokumenty w nim zapisywane powinny móc zawierać hiperłącza. XML nie ma jednak żadnych predefiniowanych znaczników, a więc także żadnego predefiniowanego sposobu wyrażania łączników.

Problem ten rozwiązano definiując w standardzie XLink specjalną przestrzeń nazw i zestaw zastrzeżonych atrybutów leżących w tej przestrzeni. Każdy element XML, mający takie atrybuty, nabiera znaczenia łącznika.

Planowane możliwości łączników są w XLink znacznie większe, niż w łącznikach znanych z HTML. Łączniki proste (*simple*), choć mają inną składnię niż w HTML, są funkcjonalnie podobne. Za to łączniki złożone (*extended*) mogą wskazywać wiele punktów docelowych (np. do wyboru). Dokument wskazywany przez łącznik może być wyświetlany w miejsce dotychczasowego, w nowym środowisku (oknie), albo wstawiony do bieżącego dokumentu. Łącznik może być uaktywniany przez użytkownika (np. kliknięciem) albo automatycznie w czasie ładowania dokumentu. Kombinacja tych możliwości pozwala np. na realizację przekierowania lub włączenia (*include*).

Standard XLink długo nie mógł doczekać się implementacji w popularnych narzędziach. Ostatnio ograniczona implementacja (tylko łączniki proste) pojawiła się w przeglądarkach bazujących na projekcie Mozilla.

Łącznik powinien móc wskazywać nie tylko cały dokument, ale też określone w nim miejsce lub fragment, przy czym taka możliwość nie powinna być uwarunkowana istnieniem specjalnego oznakowania wewnątrz wskazywanego dokumentu (twórca łącznika może przecież nie mieć praw do modyfikacji docelowego dokumentu). Wskazanie tego typu może bazować na kontekście, a do jego opisu wykorzystać można notację podobną do XPath. Takie adresowanie fragmentów dokumentu opisuje propozycja standardu XPointer. Niestety, prace standaryzacyjne poświęcone temu zagadnieniu nie posuwają się w oczekiwanym tempie, a implementacji w popularnym oprogramowaniu brak.

### 2.2.8. Przetwarzanie XML – parsery DOM, SAX, JAXP

Jedną z najważniejszych zalet XML jest łatwość przetwarzania. Już sama składnia języka, z restrykcyjnymi regułami domykania znaczników, sprzyja łatwemu przetwarzaniu. Ale programista w ogóle nie musi zajmować się bezpośrednio analizowaniem tekstu – robią to za niego gotowe, biblioteczne analizatory leksykalne (parsery). Parsery te zwykle wyposażone są w możliwość walidacji dokumentu z użyciem DTD i/lub schematu oraz są często zintegrowane z procesorami XSLT.

Istnieje kilka standardów opisujących typowe API do analizatorów leksykalnych XML. API te definiują interfejsy dla różnych języków programowania, ale najpopularniejsze implementacje dotyczą języka Java.

Najbardziej znany jest standard DOM, będący dziełem W3C. Opisuje on sposób przetwarzania dokumentu XML, reprezentowanego jako na drzewo obiektów. API parsera typu DOM umożliwia dokonanie przekształcenia dokumentu na takie drzewo, manipulacji na tym drzewie oraz serializacji, tzn. przekształcenia drzewa z powrotem w tekst XML. DOM pozwala stosunkowo łatwo operować na dokumencie i dokonywać jego modyfikacji, ale wymaga sporych zasobów, ponieważ cały dokument jest przekształcany na strukturę w pamięci operacyjnej. W przypadku konieczności operowania na dużych dokumentach DOM może więc nie zapewniać dostatecznej wydajności.

SAX (*Simple API for XML*) to drugi znany standard (nie będący opracowaniem W3C, ale stanowiący powszechnie uznany standard *de facto*, obecnie rozwijany w ramach SourceForce). W przeciwieństwie do DOM, przetwarzanie za pomocą SAX nie wymaga tworzenia obrazu dokumentu w pamięci, opiera się zaś na programowaniu zdarzeniowym. Parser wczytuje dokument i natrafiając na charakterystyczne jego części, np. początki i końce znaczników, generuje zdarzenia. Programista może obsługiwać te zdarzenia i dzięki temu analizować treść dokumentu. Przetwarzanie za pomocą SAX jest zwykle zdecydowanie wydajniejsze od użycia DOM, zwłaszcza dla dużych dokumentów. Ponieważ jednak nie jest tworzony obraz całego dokumentu, ten sposób przetwarzania nie jest odpowiedni wtedy, gdy na dokumencie chcemy dokonać pewnych manipulacji i tak zmieniony dokument z powrotem zapisać w XML.

Choć powyższe standardy opisują interfejsy API parserów, nie definiują jednak precyzyjnie wszystkich szczegółów implementacji, potrzebnych programiście chcącemu użyć parsera. Dlatego w programach stosujących te API nie można w łatwy sposób wymienić jednej implementacji parsera na inną. Dla języka Java zdefiniowano dodatkowy standard JAXP (*Java API for XML Processing* [JAXP]), stanowiący rodzaj „nakładki” na parsery DOM i SAX oraz procesory XSLT. JAXP zapewnia jednolity interfejs do owych narzędzi, niezależny od dostawcy ich implementacji.

Implementacje parserów XML wchodzą w skład wszystkich pakietów narzędzi XML-owych, są też wbudowane w lepsze przeglądarki WWW, co umożliwi programowe manipulacje na dokumentach XML po stronie klienta (np. w języku JavaScript).

### 2.3. Ważniejsze implementacje

Istniejące obecnie implementacje standardów XML-owych podzielić można na dwie grupy: rozwiązania wbudowane w przeglądarki internetowe i narzędzia dla programistów.

W pierwszej grupie niewątpliwie najbardziej znane rozwiązania wbudowane są w przeglądarkę Microsoft Internet Explorer. Od wersji 5 narzędzie to ma wbudowany parser XML; w wersji 6 wbudowany jest parser zgodny ze standardami DOM i SAX, z możliwością walidacji z użyciem DTD i XML Schema. Do formatowania dokumentów użyć można języków CSS lub XSLT (procesor XSLT jest zintegrowany z parserem). Narzędzia te są wykorzystywane bezpośrednio przez przeglądarkę (np. do wykonywania transformacji XML → HTML po stronie przeglądarki); mogą też być używane w programach, np. w języku JavaScript. Specyficzną cechą tego rozwiązania jest udostępnienie rozszerzeń HTML, umożliwiających umieszczanie w HTML tzw. wysp danych XML oraz korzystanie z tych danych na stronach WWW (możliwe bez konieczności programowania).

Dość znaczne wsparcie dla XML oferują także przeglądarki korzystające z technologii Mozilla. Zawierają one parser XML (bez walidacji), API typu DOM, obsługę prostych łączników XLink i fragmentów specyfikacji XPointer oraz możliwości formatowania XML za pomocą CSS. Netscape w nowszych wersjach zawiera także możliwość formatowania za pomocą XSLT.

Niezależnych narzędzi dla programistów dostarczają zarówno znani producenci oprogramowania, jak i organizacje zajmujące się oprogramowaniem *open-source*. Jako przykłady wymienić należy Oracle XDK (*XML Developer's Kit*), rodzinę parserów Xerces i procesor XSLT Xalan – rozwijane w ramach projektu Apache oraz parser SAX i procesor XSLT Saxon – projekty prowadzone przez SourceForge.

### 3. Nowe standardy z rodziny XML

Technologia XML wciąż dynamicznie się rozwija. Znale już od jakiegoś czasu standardy XML-owe są udoskonalane – wykorzystuje się nagromadzone już spore doświadczenie w użytkowaniu tej technologii. Powstają też nowe składniki rodziny XML.

#### 3.1. Nowe prace nad wcześniejszymi standardami

##### 3.1.1. Przestrzenie nazw

W opracowywanej wersji 1.1 rekomendacji przestrzeni nazw poprawiono błędy zauważone w specyfikacji XML Namespaces 1.0. Wprowadzono też dwie istotne nowości: dodano możliwość odwołania deklaracji prefiksu, oznaczającego przestrzeń nazw, oraz wprowadzono IRI (*Internationalized Resource Identifiers*) jako identyfikatory przestrzeni (w wersji 1.0 stosowano URI). IRI stanowią odmianę URI, w której dopuszczono użycie wszelkich znaków Unicode; zdefiniowano też reguły konwersji IRI na URI.

##### 3.1.2. XPath

W XPath 2.0 wprowadzono pojęcie sekwencji (*sequence*) – uporządkowanego ciągu węzłów oraz wartości prostych (atomowych). Wynikiem przetwarzania za pomocą XPath 2.0 jest zawsze taka sekwencja. Możliwe jest powiązanie węzłów z typami prostymi XML Schema i odpowiednie dla danego typu przetwarzanie. Wprowadzono możliwość wykonywania operacji na elementach sekwencji (wyrażenia `for...return`), wyrażenia warunkowe `if...then...else`, kwantyfikatory `some...satisfies` oraz `every...satisfies` oraz operacje przecięcia i różnicy teorii mnogościowej (suma była już w XPath 1.0).

Wersja 2.0 standardu XPath jest używana w językach XQuery oraz XSLT 2.0.

##### 3.1.3. XSLT

Specyfikacja XSLT 2.0 dostosowuje język przekształceń do współpracy z XPath 2.0 (przetwarzanie sekwencji zamiast zbiorów węzłów) oraz wprowadza kilka innych ważnych zmian. Wprowadzono nowe pojęcie drzewa tymczasowego (*temporary tree*) – zmienne mogą przechowywać takie drzewa. Istnieje możliwość wywoływania walidacji drzew tymczasowych i drzewa wynikowego za pomocą XML Schema oraz jawnego specyfikowania typów zmiennych i parametrów. Możliwe jest grupowanie węzłów (nieco podobne do *group by* w SQL) i iteracyjne przetwarzanie grup (`for-each-group`). W arkuszu można zdefiniować funkcje, które mogą następnie być wykorzystywane w wyrażeniach XPath. Możliwe jest tworzenie wielu dokumentów wynikowych za pomocą jednego arkusza (instrukcja `result-document`). Instrukcja `analyze-string` służy do przeszukiwania tekstu z użyciem wyrażen regularnych, umożliwiając m.in. wygodne przetwarzanie danych wyrażonych w tradycyjnych formatach (np. CSV). Dodano nowy format wyjściowy transformacji: XHTML. Ulepszono operacje sortowania przez wprowadzenie możliwości zapamiętywania wzorców sortowania oraz definiowania pożądanej kolejności znaków. Możliwe jest wczytywanie zawartości plików zewnętrznych jako czystego tekstu. Poprawiono także obsługę przestrzeni nazw i dodano możliwość wykorzystywania adresów względnych (np. w stosunku do adresu bazowego podanego zgodnie z XML Base).

#### 3.2. Zapytania do struktur XML – XQuery

XQuery jest językiem zapytań do danych zapisanych w XML, stanowiącym rozszerzenie XPath. Można nim przetwarzać dane zapisane w różnych postaciach – w plikach, bazach danych itp. Podobnie jak w XPath 2.0, wynikiem zapytania jest sekwencja składająca się z węzłów oraz wartości atomowych. Język nie jest dialektem XML, przypomina raczej SQL (a może bardziej OQL –

*Object Query Language*); istnieje jednak dialekt o nazwie XQueryX, który stanowi reprezentację XQuery w XML.

Typowe zapytanie w XQuery ma postać tzw. wyrażenia FLWOR (*for-let-where-order by-return*). W klauzuli *for...in* definiuje się wyrażenie XPath i zmienną, która „przebiega” sekwencję wyników wyrażenia. W klauzuli *let* dokonać można podstawień (wyliczeń) dodatkowych zmiennych. Klauzula *from* służy – jak można się było domyślać – do określenia warunków, które ma spełniać wynik zapytania. Do określenia kolejności wyników służy oczywiście klauzula *order by*. Wreszcie klauzula *return* pozwala sformułować postać wyniku (używa się tu zmiennych zdefiniowanych w poprzednich klauzulach).

W jednym zapytaniu wystąpić może wiele klauzul *for* i *let*. Można tu używać wyrażen warunkowych (*if-then-else*). Za pomocą warunku *where* można budować konstrukcje podobne do złączeń w zapytaniach relacyjnych. W warunkach możliwe jest użycie kwantyfikatorów *every-in-satisfies* oraz *some-in-satisfies*.

Choć standard XQuery nie jest jeszcze ustabilizowany, istnieją już pierwsze implementacje, m.in. w Oracle9i rel. 2.

### 3.3. Bezpieczeństwo dokumentów XML – XML Signature, XML Encryption

Elektroniczna wymiana danych, zwłaszcza w handlu elektronicznym, stała się jednym z najważniejszych zastosowań XML. W tym kontekście szczególnie jest zaś ważne, by przesyłane dokumenty były wiarygodne (tzn. by w sposób pewny znany był ich autor i by można było zagwarantować, iż nikt niepowołany nie zmienił ich treści), a często także by były one poufne. Oczywiście dokumenty XML można byłoby w całości poddać procesowi szyfrowania, ale wówczas informacje które nie powinny być ukryte trzeba byłoby przysyłać osobno. Dlatego podjęto prace nad standardami, które umożliwiają umieszczanie podpisów elektronicznych wewnątrz dokumentów XML oraz szyfrowanie jedynie części dokumentów, przy zachowaniu poprawnej struktury XML.

Standard XML Signature umożliwia umieszczenie w dokumencie XML podpisu elektronicznego autora oraz zagwarantowanie, że dokument nie został zmodyfikowany od momentu podpisania. W tym celu w dokumencie umieszcza się specjalny element *Signature*, należący do przestrzeni nazw określonej w standardzie, w którym umieszcza się certyfikat osoby podpisującej (zawierający jej klucz publiczny), skrót z podpisywanych danych oraz wartość podpisu elektronicznego (którą stanowi skrót zaszyfrowany kluczem prywatnym autora). Standard przewiduje możliwość umieszczania podpisu w osobnym dokumencie, obsługę podpisów wielokrotnych (z możliwością wprowadzania modyfikacji przed każdym z kolejnych podpisów) oraz wykorzystanie różnych metod szyfrowania i tworzenia skrótów.

Z kolei standard XML Encryption opisuje sposób zapisu w XML zaszyfrowanych fragmentów dokumentu, w celu zagwarantowania ich poufności. Wybrany do szyfrowania fragment zastępowany jest elementem *EncryptedData* w taki sposób, że dokument nadal pozostaje poprawny. Oprócz zaszyfrowanej informacji umieszcza się tam również wartość klucza, na ogół także w postaci zaszyfrowanej: symetryczny klucz sesyjny szyfruje się zwykle kluczem publicznym odbiorcy; można też przesłać klucz sesyjny zaszyfrowany kilkoma kluczami publicznymi, jeśli odbiorców ma być kilku.

### 3.4. Budowanie stron WWW – XHTML i XForms

XHTML jest nową wersją standardu, mającą zastąpić HTML. W stosunku do swego poprzednika wprowadza on przede wszystkim ostrzejsze reguły składni, zgodne z wymaganiami XML. Standard stanowi więc przeformułowanie HTML w XML – język staje się dialektem XML. Oznacza to, że do przetwarzania stron WWW zapisanych w XHTML można używać wszelkich

narzędzi XML-owych. XHTML ma budowę modułową, co umożliwia rozszerzanie go przez za-nurzenie w nim innych dialektów XML.

Jednym z takich dialektów jest XForms – moduł XHTML służący do definiowania formularzy na stronach WWW. W stosunku do formularzy znanych z HTML, XForms ma znacznie większe możliwości: większy jest wybór elementów formularza, możliwe jest oddzielenie logiki formularza od sposobu prezentacji. Przede wszystkim jednak wprowadzono znaczne możliwości kontroli wprowadzania danych po stronie samej przeglądarki, bez konieczności odwoływania się do języków typu JavaScript. Można określić typy pól używając typów XML Schema, kontrolować opcjonalność pól, zdefiniować obowiązkowe zależności między polami. Elementy formularza mogą także pojawiać się warunkowo, o ile zawartość innych elementów spełnia podane kryteria. Reakcja na działania użytkownika może być zdefiniowana za pomocą odpowiadających na zdarzenia wyzwalaczy. XForms nie da się niestety jeszcze wykorzystywać w typowych przeglądarkach, ale istnieją już prototypowe implementacje.

### 3.5. Nowe modele przetwarzania XML

Choć modele przetwarzania DOM i SAX zdobyły znaczną popularność, nie zawsze są one wygodne w użyciu. Powstają więc nowe propozycje sposobów przetwarzania.

Parsery typu SAX zapewniają wydajne przetwarzanie dużych dokumentów, ale odbywa się ono w modelu pchającym (*push parsing*): to parser „napędza” całe przetwarzanie. Taki model nie zawsze jest dogodny; w szczególności nie można przerwać analizy dokumentu przed jego końcem (nawet gdy już uzyskano wszystkie potrzebne informacje), model zdarzeniowy komplikuje też pisanie programów (konieczne może być zapamiętywanie złożonego stanu przetwarzania między obsługą kolejnych zdarzeń, nie można też np. użyć rekurencji). Alternatywę stanowić może model przetwarzania strumieniowego: *pull parsing*. W tym modelu to program przetwarzający „napędza” działanie parsera, inicjując przetwarzanie, a następnie żądając od parsera podawania kolejnych „zdarzeń” i analizując je. Tego typu model przetwarzania jest m.in. wykorzystywany w platformie .NET firmy Microsoft, podobnych narzędzi dostarcza też BEA.

Z kolei przetwarzanie parserami typu DOM, choć pozwala na manipulowanie zawartością dokumentów XML, zmusza jednak programistę do stylu pracy niezgodnego z ideologią programowania obiektowego. Programista działa bowiem na obiektach odwzorowujących abstrakcyjną strukturę drzewa, a nie obiekty konkretnej dziedziny, którą opisuje przetwarzany dokument. Inne podejście prezentuje przetwarzanie typu *XML binding*. W tym podejściu za pomocą odpowiedniego generatora tworzy się klasy odpowiadające dziedzinie problemowej, które zapewniają odwzorowanie obiektów dziedziny problemowej na zapis w XML. Program interpretujący dokument korzysta w sposób naturalny z klas dziedziny, zatem jego zrozumienie i utrzymanie jest łatwiejsze. Istnieje sporo narzędzi służących do generowania tego typu reprezentacji obiektowej dokumentów XML; jedno z nich wchodzi w skład pakietu Oracle XDK. Powstał też standard tego typu mapowania dla języka Java – JAXB (*Java Architecture for XML Binding*) [JAXB].

## 4. Podsumowanie

Technologia XML osiągnęła już dojrzałość i znalazła liczne ważne zastosowania. Jednak, jako rozwiązanie stosunkowo młode, XML stale się rozwija. Standardy z nim związane są udoskonalane i rozbudowywane, powstają też nowe idee i związane z nimi propozycje standaryzacji. Za bardzo szybko rozwijającymi się pracami standaryzacyjnymi nie nadążają twórcy implementacji, ale i tu widać znaczne postępy. Implementacjami standardów XML-owych zajmują się zarówno liczni komercyjni producenci oprogramowania, jak i główne organizacje tworzące oprogramowanie *open-source*. Zapewnia to zdrową równowagę, gdyż zmusza producentów do przestrzegania standardów – żaden z nich nie może narzucić własnych firmowych rozwiązań.

Istotne zalety XML zostały docenione, można więc pokusić się o twierdzenie, że język ten znalazł stałe miejsce wśród najważniejszych technologii współczesnej informatyki.

## Bibliografia

- [FOP] FOP (Formatting Objects Processor). <http://xml.apache.org/fop>
- [JAXB] Java Architecture for XML Binding (JAXB). <http://java.sun.com/xml/jaxb>
- [JAXP] Java API for XML Processing (JAXP). <http://java.sun.com/xml/jaxp>
- [KaGw02] Kazienko P., Gwiazda K.: XML na poważnie. Helion, 2002, ISBN 83-7197-765-4.
- [Kaz03] Kazienko P.: Co tam panie w XML-u? Software 2.0, nr 6 (102), 2003, ISSN 1508-1656, ss. 26-30.
- [MZ01] Rozporządzenie Ministra Zdrowia z dn. 27 grudnia 2000 r. w sprawie trybu i sposobu przekazywania oraz zakresu danych o obrocie refundowanymi lekami i materiałami medycznymi przekazywanych przez apteki. Dziennik Ustaw RP, nr 4, 2001.
- [Ple03] Plechawski M.: Pull Parsing – nie pozwól się popychać. Software 2.0, nr 6 (102), 2003, ISSN 1508-1656, ss. 50-57.
- [Tra00] Traczyk T.: Język XSL, Materiały VI konferencji deweloperów i użytkowników Oracle. Zakopane, październik 2000.
- [Tra01] Traczyk T.: Schematy XML. Materiały VII konferencji użytkowników i deweloperów Oracle, ss. 25-36. Zakopane, październik 2001.
- [Tra02a] Traczyk T.: Zastosowanie XML w rozproszonej heterogenicznej bazie danych wspierającej wielki eksperyment fizyki jądrowej. Materiały XIV Górskiej Szkoły PTI, t. 1, ss. 129-140. Szczyrk, czerwiec 2002.
- [Tra02b] Traczyk T.: Obiekty formatujące w języku XSL. Materiały VIII konferencji użytkowników i deweloperów Oracle, ss. 29-44. Zakopane, październik 2002, ISSN 1641-2117.
- [Tra02c] Traczyk T.: Dlaczego XML? Materiały I Seminarium „.Net w e-Firmie i e-Administracji”, Instytut Maszyn Matematycznych. Warszawa 2002.
- [Tra03] Traczyk T.: Wprowadzenie do XML. Materiały II Szkoły PLOUG, ss. 1-61. Poznań, luty 2003, ISSN 1641-2117.
- [Tysz03] Tyszko S.: XML i XSLT w Internet Explorerze. Software 2.0, nr 6 (102), 2003, ISSN 1508-1656, ss. 38-43.
- [Woj03] Wojciechowski M.: Standard SQL/XML. Wprowadzenie do XQuery. Materiały II Szkoły PLOUG, ss. 223-259. Poznań, luty 2003, ISSN 1641-2117.