XML tehnologija
i primjena u sustavima procesne informatike

Damir Kirasić
Sveučilište u Zagrebu
Fakultet elektrotehnike i računarstva
Unska 3, 10000 ZAGREB, Hrvatska
E-mail: damir.kirasic@fer.hr

Sažetak – XML je proširivi jezik za označavanje podataka i dokumenata. U ovom pregledu se: definiraju temeljni pojmovi XML-a i njegove sintakse; objašnjavaju pojmovi "dobro formiran" i "valjan"; prikazuju osnove DTD-a i XML Scheme. Rad također daje kratki pregled mogućih načina programske obrade XML dokumenata kao i eventualnih primjena XML tehnologije kao što je npr. područje procesne informatike. Kada jednom znamo osnove XML-a, vrlo lako ga možemo primijeniti na bilo koje područje primjene.

I. ŠTO JE TO XML?

XML je kratica od "Extensible Markup Language". XML je, dakle, proširivi jezik za označavanje. XML-om zapisujemo dokumente i podatke u tekstualnom formatu. XML format je čitljiv za tekst editore kao što su npr. MS Word ili za razne programerske editore. XML dokument se najčešće sprema kao tekstualna datoteka koja se sastoji od *sadržaja* i *oznaka*. Iako je prvenstveno namijenjen za programsku obradu, XML format je čitljiv i ljudima. Evo primjera jedne tekstualne datoteke u XML formatu:

```
<?xml version="1.0" encoding="UTF-8" ?>
<memo>
    <contact type="person">
        <name>Ivo Ivić</name>
        <mobile>098 1234567</mobile>
        <email>ivo.ivic@fer.hr</email>
    </contact>
    <contact type="company">
        <name>Best Corporation</name>
        <mobile>091 123456</mobile>
        <email>best@best.com</email>
    </contact>
    <contact type="person">
        <name>Josip Horvat</name>
        <mobile>098 123456</mobile>
        <email>josip.horvat@fer.hr</email>
    </contact>
</memo>
```

Primjer 1: XML dokument

Gornji XML dokument može poslužiti kao adresar u kojeg spremamo podatke o kontaktima – imena ljudi i tvrtki, telefonske brojeve, email adrese i sl. Sam po sebi dokument je samoobjašnjavajuć radi oznaka koje označavaju dijelove teksta. Vrlo lako bismo ovakav dokument mogli proširiti s dodatnim elementima <meeting>, <todo> , datumima i sl. Tako bismo dobili kompletan podsjetnik kojeg eventualno možemo dijeliti s našim suradnicima uz odgovarajuću programsku podršku.

XML pripada obitelji jezika za označavanje (eng. markup languages). Najpoznatiji iz te obitelji je HTML – osnovni jezik za izradu Web stranica. Osim XML-a i HTML-a, u tu skupinu jezika pripadaju još npr. SGML (Standard Generalized Markup Language) i RTF (Rich Text Format).

XML je podskup od SGML-a (definiran standardom ISO 8879). SGML je složen, moćan i opsežan meta-jezik. Meta-jezici su jezici za opisivanje dozvoljenog pravopisa i gramatike drugih jezika. Jedan od temeljnih koncepata SGML-a je da jezici za opisivanje dokumenata i podataka mogu imati oznake. Oznake su posebne riječi u tekstu kojima određujemo vrstu teksta. Primjeri za oznake u Primjeru 1 su <contact>, <name>, </email>.

XML je zadržao značajku SGML-a da se njime mogu opisivati novi jezici, ali je prilikom definiranja XML-a iz SGML-a izostavljeno sve ono što je kompliciranо za razumjeti i teško za implementirati.

Iako u svom nazivu ima riječ "jezik", XML nije programski jezik kao što su to npr. C++, Java ili C#. Dokument napisan u XML-u se ne može "izvoditi" – on ne radi ništa. Prema jednoj od definicija XML je "tekst format" i kao takav predstavlja nešto što je pasivno.

II. OSNOVNI KONCEPTI: SADRŽAJ I OZNAKE

XML je zamišljen kao jezik za opisivanje dokumenata i podataka. Pod izrazom "dokumenti i podaci" podrazumijevamo tekstualne dokumente ili pak skupove podataka kakvi se obično pohranjuju u baze podataka. Promatrajući XML kao tekstualni format, može se ustvrditi da je on potpuno neovisan o računalnoj platformi na kojoj se nalazi. Također je potpuno neovisan o operacijskom sustavu kojeg koristimo – XML je otvoreni standard čija je specifikacija javna i dostupna svima. Za korištenje XML-a nisu potrebne nikakve licence.
Oznake (eng. tag) koje koristimo za opisivanje podataka nisu unaprijed definirane. XML standard jedino opisuje minimalni skup pravila koja dokument mora zadovoljavati – međutim imena oznaka nisu unaprijed određena. Korisnici XML-a moraju sami definirati dozvoljene oznake za označavanje.

Moguće je da je već netko definirao neki skup oznaka pa možemo koristiti te oznake. U svakom slučaju, ako

razmjenjujemo XML dokumente s drugima, potrebno je da obje strane poštuju zajednički dogovor o oznakama. Osim oznaka potrebno je definirati i dozvoljene kombinacije oznaka. Ta zajednička konvencija, koja definira dozvoljeni format XML dokumenata opisuje se "Document Type Definition" (DTD) ili "XML Schema" dokumentima (više o njima kasnije).

XML dijeli neke zajedničke osobine sa HTML-om, ali je dizajniran za drugačije potrebe i nije izravna zamjena za HTML. Zajedničko im je što se za označavanje dijelova dokumenta koriste oznake. Npr. u HTML-u oznaka <b> se koristi za označavanje dijela teksta koji treba biti podebljan. Uzmimo kao primjer slijedeći HTML tekst, dio neke Web stranice:

```
HTML je jezik za opisivanje <b>prikaza</b>.
```

Nakon prikaza u vašem Internet pretraživaču gornji HTML tekst će izgledati ovako:

HTML je jezik za opisivanje **prikaza.**

Na sličan način, korištenjem drugih HTML oznaka, dobivaju se svi oni različiti prikazi i slike koje ste vidjeli na Internet stranicama. Važno je uočiti da su oznake za HTML unaprijed definirane dok se XML oznake definiraju po vlastitim potrebama. Oznake za HTML (kao što je oznaka <b>) *moraju* biti ranije definirane da bi Internet pretraživači mogli ispravno prikazati ono što je zamislio autor Web stranice.

XML se usredotočuje opis podataka i na to što ti podaci znače, a ne na opis prikaza. To je vrlo očito u Primjeru 1 gdje se oznake koriste za određivanje vrste podataka, a ne njegovog prikaza.

Najčešće se XML dokumenti spremaju u datoteke, ali to nije nužno. XML dokument možemo dobiti preko mreže kao niz znakova ili npr. iz relacione baze podataka.

Posao definiranja XML-a i pratećih tehnologija koordinira World Wide Web konzorcij kojeg često skraćeno nazivaju W3C (http://www.w3c.org). Sam XML standard je približne duljine 32 A4 stranice i nalazi se na adresi: http://www.w3.org /TR/2004/REC-xml11-20040204/. Na matičnoj Web stranici W3C konzorcija naći ćete puno poveznica na Web stranice raznih tehnologija izvedenih ili povezanih sa XML-om.

III. GLAVNI XML DIJELOVI: ELEMENTI I ATRIBUTI

Prema XML standardu, na najnižoj se razini XML dokument sastoji od niza Unicode znakova. Unicode standard definira više načina kodiranja znakova (vidi www. unicode.org) i danas predstavlja najvažniji znakovni standard. Za zapisivanje znakova se uglavnom koristi više okteta, za razliku od ASCII kodiranja u kojem se koristio samo jedan oktet. Unicode podržava sve grafeme najvažnijih svjetskih pisama (uključujući velik broj kineskih grafema) pa tako i hrvatske grafeme č, ć, đ,…
Unicode znakovi tvore različite dijelove XML dokumenta. Najčešće su to elementi, podelementi i atributi kao što je to u gornjem primjeru. Elementi započinju *početnom oznakom* npr. <email>, a završavaju sa *završnom oznakom*, npr. </email>. Između početne i završne oznake je sadržaj elementa, npr. josip. horvat@fer.hr. Element može biti prazan – bez svog sadržaja U tom slučaju pišemo ga sa kosom crtom na kraju imena oznake kao npr. <sendGreetings/>. Ako nije prazan, XML element mora imati završnu oznaku.

Osim elemenata, u XML dokumentu se mogu pojaviti i drugi dijelovi. Jedan smo već vidjeli – to je *XML deklaracija*:

```
<?xml version="1.0" encoding="UTF-8"?>
```

Ako je XML deklaracija prisutna, mora se pojaviti na samom početku dokumena. Obavezno je navesti i inačicu XML-a koja se koristi, a moguće je navesti i tip znakova koji se koriste u zapisu XML dokumenta. (Do sada postoje jedino inačice 1.0 i 1.1 koje se minimalno razlikuju i to ne u sintaksi nego u značenju pojedinih dijelova). Standard dozvoljava da XML dokument bude bez deklaracije.

XML dokument mora imati *samo jedan korijenski element* – element unutar kojeg su svi ostali elementi. U našem primjeru korijenski element je <memo>, unutar kojeg imamo podelemente <contact>. Za element <contact> kažemo da je podelement ili element-dijete od elementa <memo>. Za element <memo> kažemo da je element-roditelj elementa <contact>. <contact> ima svoje podelemente koji pak mogu imati svoje podelemente.

Osim elemenata, u XML dokumentima najčešće srećemo *atribute*. Atributi se pridružuju elementima s idejom da pruže dodatne informacije o svojstvima elemenata.

Svaki <contact> element ima i atribut s imenom "type". U prvom elementu, vrijednost atributa je "person", a u drugom elementu vrijednost atributa je "company". Elementi mogu imati više atributa, ali unutar jednog elementa ne smije biti više atributa s istim imenom. Vrijednost atributa može biti jedino niz znakova, dok npr. vrijednost elementa može biti niz znakova i/ili drugi podelementi. (U XML terminologiji za atribute se kaže da imaju svoju "vrijednost", a za elemente se kaže da imaju svoj "sadržaj").

Da bi neki dokument bio XML dokument moraju biti zadovoljeni minimalni uvjeti koji se skraćeno nazivaju "dobro-formiran" (eng. well formed). Jedan od tih minimalnih uvjeta je, već spomenuti zahtjev, da XML dokument ima samo jedan korijenski element. Osim tih minimalnih uvjeta (sumarno opisanih u poglavlju 5), mogu se postaviti dodatni uvjeti kroz DTD ili XML Schemu. Ako dokument zadovoljava minimalne uvijete, kažemo da je dobro formiran, a ako zadovoljava dodatna ograničenja iz DTD-a ili XML Scheme, kažemo da je "valjan".

Drugi minimalni uvjet koji treba zadovoljavati je da se podelementi moraju kompletno nalaziti unutar elementa roditelja:

```
<a> <b> </b> </a>
```

Nije dozvoljeno da podelement završi nakon završetka elementa-roditelja:

```
<a> <b> </a> </b>
```
KRIVO

Taj se uvjet može sažeto formulirati kao: Elementi i podelementi moraju biti ispravno ugnježđeni.

IV. OSTALI XML DIJELOVI

Osim XML deklaracije, elemenata i atributa, XML može sadržavati slijedeće dijelove:

- Komentari
- Procesorske naredbe
- DTD
- Reference na entitete
- CDATA sekcije

A. Komentari

Svrha XML komentara je mogućnost dodavanja napomena u XML dokument, a koji se neće tretirati kao podaci niti kao dio tekstualnog sadržaja dokumenta. Primjer XML komentara:

```
<!--  Ovo je XML komentar -->
```

Komentar moramo započeti sa <!-- a završiti sa -->. Unutar komentara pišemo proizvoljni tekst, ali unutar komentara ne smijemo imati nove komentare. Komentari se mogu rasprostirati u više redaka.

B. Procesne naredbe

Procesne naredbe se upisuju u XML dokument kada želimo imati dodatne podatke za programe koji će obrađivati XML. Jedna procesna naredba bi mogla izgledati ovako:

```
<? progABC inputParam="123" ?>
```

Sadržaj procesnih naredbi je proizvoljan – nije određen XML standardom. Mora započeti s *imenom ciljnog programa* kome je namijenjen. U gornjem primjeru ime ciljnog programa je "progABC". To znači da je procesna naredba namijenjena programu s tim imenom. Drugi ciljni programi se neće obazirati na tu naredbu. Interpretacija *sadržaja* procesne naredbe (u gornjem primjeru je to inputParam ="123") mora obaviti ciljni program.

U jednom XML dokumentu možemo imati više procesnih naredbi namijenjenih različitim ciljnim programima.

Najčešća procesna naredba koja se susreće u praksi je "stylesheet" procesna naredba koja je namijenjena programu za prikaz XML dokumenta:

```
<?xml-stylesheet type="text/xml"
                        href="beauty.xsl" ?>
```

Ciljni program "xml-stylesheet" će iz sadržaja ove procesne naredbe znati da treba pozvati XSLT stroj koji će obraditi naredbe u datoteci "beauty.xsl" i primijeniti ih na XML dokument. (Više o XSLT-u kasnije).

C. DTD

DTD je kratica od Document Type Description, a odnosi se na dio XML dokumenta koji detaljnije opisuje i ograničava dozvoljeni sadržaj XML dokumenta. DTD dio je opcionalan, što znači da ne mora biti prisutan u XML dokumentu.

```
<!DOCTYPE memo [
   <!ENTITY best "Best Corporation,
             The best is not good enough">
   <!ELEMENT memo (contact)* >
   <!ELEMENT contact (name,mobile, email)>
   <!ELEMENT name (#PCDATA) >
   <!ELEMENT mobile (#PCDATA) >
   <!ELEMENT email (#PCDATA) >
   <!ATTLIST contact type CDATA #REQUIRED >
]>
```

Primjer 2. DTD dokument

Gornji DTD opisuje dozvoljeni sadržaj XML dokumenta – upravo dokumenta prikazanog u Primjeru 1. DTD određuje da XML dokument mora započeti s korijenskim elementom <memo> ispod kojeg može biti 0 ili više (* u definiciji elementa <memo>) elemenata <contact>. Element <contact> mora imati podelemente: <name>, <mobile> i < email>. Sadržaj pak ovih podelemenata je proizvoljni tekst. U DTD terminologiji "proizvoljan tekst" se zapisuje kao (#PCDATA). Poseban dio DTD-a može biti definicija entiteta. U gornjem primjeru to je redak:

```
<!ENTITY best "Best Corporation, The best is
not good enough">
```

Entitete je najbolje razumjeti kao par ime/vrijednost. Ako ste programer entitet možete zamisliti kao makro sa vrijednošću. U gornjem primjeru koristimo ime "best", a vrijednost za "best" je tekst "Best Corporation, The best is not good enough". Nakon ovakve definicije, unutar XML dokumenta možemo umjesto duge vrijednosti koristiti kratko ime &best; Ovakvo pozivanje na vrijednost entiteta se naziva referenca na entitet. Važno je uočiti da u referenci na entitet moramo imati znak & pa ime entiteta pa znak ;.

DTD ne mora biti u istoj datoteci sa XML dokumentom već se može definirati u zasebnoj datoteci. Na taj način se više XML dokumenata može referencirati na jednu DTD datoteku.

DTD pruža još puno drugih mogućnosti: npr. definiciju dozvoljenih atributa, da li su atributi obavezni ili nisu, binarni sadržaj, itd. Detalji ovdje neće prikazivati.

Novija tehnika za opis dozvoljenog formata XML dokumenta je XML Schema. Kratki opis XML Scheme se nalazi u poglavlju 7.

D. Reference na entitete

Reference na entitete su opisane u prethodnom odsječku koji opisuje najvažnije pojmove DTD-a.

E. CDATA sekcije

CDATA sekcija je dio XML dokumenta. Npr. jedna CDATA sekcija bi mogla izgledati ovako:

```
<![CDATA[
Ovdje može biti proizvoljni sadržaj koji
uključuje i znakove <, >, &, ", '
Proizvoljni tekst unutar CDATA se neće
provjeravati niti će se na njemu obavljati
bilo kakve supstitucije teksta.
]]>
```

CDATA sekcije koristimo kada želimo imati dijelove teksta koji se neće ni na koji način obrađivati. U praksi se često koriste za pisanje programskih odsječaka unutar XML-a. Neki programski odsječak bi npr. mogao sadržavati naredbe:

```
if ( placa < 8000 )
      Document.writeln("Hmmmmm?")
```

Znak "<" se u XML-u koristi kao početak oznake, pa bi ga XML program krivo interpretirao i javio grešku. Ako bismo ovaj programski odsječak ubacili unutar CDATA sekcije, program se ne bi bunio.

V. SAŽETAK PRAVILA ZA "DOBRO FORMIRAN" XML

- Dokument bi trebao početi s XML deklaracijom.

- Postoji samo jedan korijenski element.

- Imena elemenata ili atributa ne smiju sadržavati praznine.

- Neprazni elementi moraju imati početnu i završnu oznaku.

- Prazni element ima na kraju oznake znak /.

- Elementi moraju biti ispravno ugnježđeni.

- Atributi unutar istog elementa moraju imati jedinstvena imena.

- Vrijednost atributa se mora definirati unutar navodnika.

- U sadržaju elemenata ili vrijednostima atributa ne smijemo izravno koristiti znakove: < > & Moramo koristiti reference na predefinirane entitete: < > &

VI. NAMESPACE – IMENSKI PROSTOR

XML podržava imenske prostore (eng. namespace). Pojam "imenski prostor" možemo definirati kao "imenik" ili možemo reći da je to "skup imena".
Potreba za imenskim prostorima se pojavila kada su korisnici počeli spremati u jedan XML dokument dijelove koji su dolazili iz različitih izvora. Imenski prostori se koriste da ne bi došlo do sudaranja oznaka za elemente i atribute. Oni omogućuju da miješamo dijelove iz jednog XML dokumenta s dijelovima iz drugog XML dokumenta.

Svaki imenski prostor ima jedinstveni identifikator – ime, tako da se može razlikovati od drugih imenskih prostora. Tako npr. možemo zamisliti skup imena koja pripadaju imenskom prostoru "abc". Drugi imenski prostor bismo npr. mogli nazvati "fer". Ova dva imenska prostora bi mogli sadržavati isto ime npr. "var" koje bismo punim imenom pisali "abc:var" ili "fer:var".

U XML-u se imenski prostori koriste za imena elemenata i atributa. Npr. dio nekog XML dokumenta u kojem koristimo imenski prostor bi mogao izgledati ovako:

```
<fer:contact fer:type="company">
   <fer:name>Best Corporation </fer:name>
   <nasdaqNM:name>BEST
                  </nasdaqNM:name>
   <fer:mobile>091123456 </fer:mobile>
   <fer:email>best@best.com
                   </fer:email>
</fer:contact>
```

Početna oznaka elementa <contact> sada je postala <fer:contact> što znači da ime "contact" pripada imenskom prostoru "fer". Ime atributa "type" sada je postalo "fer:type". Uočite da sada imamo dva imena za tvrtku i dva elementa <name>, ali oni sada pripadaju drugim imenskim prostorima ("fer" i "nasdaqNM") pa nema sudaranja. Iako su oznake iste, dodavanjem imenskog prostora one dobivaju drugo *značenje*.

Imena imenskog prostora se formiraju prema konvenciji o web adresama. Tako će npr. Fakultet elektrotehnike i računarstva koji ima dodijeljenu web domenu "fer.hr" moći napraviti imenski prostor s jedinstvenim identifikatorima

"http://www.fer.hr/project1/imenik1" ili

"http://www.fer.hr/project2/ns"

Ovo su puna imena, jedinstveni identifikatori koji definiraju ime imenskog prostora. Ta imena ne moraju pokazivati na neku stvarnu web stranicu.

Obzirom da su ovakvi identifikatori dugački i nezgodni za pisanje ispred svakog imena elementa ili atributa, koriste se kratice ili "prefiksi". Definiranje imenskog prostora i pridruženog mu prefiksa bi moglo izgledati ovako:

```
<fer:memo xmlns:fer=
       "http://www.fer.hr/project1/imenik1>
       . . .
</fer:memo>
```

Ovim smo definirali da element "memo" pripada imenskom prostoru "www.fer .hr/project1/imenik1". Tom imenskom prostoru je pridružen prefiks "fer" kojeg možemo koristiti umjesto dugog imena. Puna imena imenskog prostora moraju biti jedinstvena, a prefiksi mogu imati proizvoljna imena.

Ako u definiciji imenskog prostora ispustimo prefiks, onda se podrazumijeva da sva imena pripadaju tom imenskom prostoru. U tom slučaju nije potrebno ispred svakog elementa ili atributa pisati prefiks. To je onda podrazumijevani (eng. default) imenski prostor. U slijedećem primjeru definiramo imenski prosto bez prefiksa:

```
<memo xmlns="http://www. fer.hr/
                      project1/imenik1" >
   <contact type="…" > … </contact>
. . .
<memo>
```

Ovdje je definiran podrazumijevani imenski prostor. Svi podelementi od <memo>, kao i sam element <memo>, pripadaju tom imenskom prostoru.

I zaključno o imenskim prostorima: Imenski prostori rješavaju problem preklapanja imena i omogućuju da u istom XML dokumentu imamo ista imena s različitim značenjem.

VII. XML SCHEMA

XML Schema je tekstualni opis koji definira dozvoljenu strukturu XML dokumenata. XML Schema ima istu funkciju kao i DTD, ali je novija i opsežnija od DTD-a. XML Schema:

- Postavlja uvijete i ograničenja na sadržaj i strukturu XML dokumenata.
- Definira rječnik oznaka i gramatiku (dozvoljene slijedove XML dijelova) koji se mogu pojaviti u dokumentu.
- Određuje pravila koja se moraju slijediti da bi neki XML dokument bio valjan.

Slijedeći primjer prikazuje XML Schemu koja opisuje dokument iz Primjera 1. Za dokument iz Primjera 1 kažemo da je valjan u odnosu na prikazanu Schemu.

1. Ljudi su se žalili da je DTD mehanizam prekompliciran.
2. W3C je ustanovio radnu grupu da riješi taj problem.
3. Nastala je XML Schema koja je puno kompliciranija od DTD-a.

Iako je Schema kompliciranija, ona pruža i puno više mogućnosti. Npr. u DTD-u ne možemo odrediti da sadržaj nekog elementa mora biti broj, dok se to u Schemi može. Npr. ako bismo željeli sadržaj elementa <mobile> ograničiti tako da je njegov sadržaj jedino dozvoljen kao broj, to bismo napravili sa slijedećim odsječkom:

```
<xsd:element name="mobile" type=
                    "xsd:positiveInteger"/>
```

Za razliku od DTD-a, XML Schema je pisana kao dobro formiran XML dokument. Iako prvi dojam o Schemi može biti "To je komplicirano", Schema je definirana vrlo logično i uz vrlo malo truda se može dobro razumijeti.

XML Schema je definirana W3C standardima koje možemo naći na adresi: http://www.w3.org/XML/Schema#dev. Schema obuhvaća detaljne opise svih aspekata XML-a, a u ovom kratkom pregledu prikazan je samo osnovni koncept.

```
<?xml version="1.0" encoding="UTF-8" ?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <xsd:element name="memo" type="memoType" />
    <xsd:complexType name="memoType" >
        <xsd:sequence>
            <xsd:element name="contact" type="contactType"
                minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType name="contactType" >
        <xsd:sequence>
            <xsd:element name="name"/>
            <xsd:element name="mobile"/>
            <xsd:element name="email"/>
        </xsd:sequence>
        <xsd:attribute name="type" type="xsd:string" use="required" />
    </xsd:complexType>
</xsd:schema>
```

Primjer 3: XML Schema

Odmah se može uočiti da je Schema XML dokument: ima XML deklaraciju, ima jedan korijenski element <schema> i definira svoj imenski prostor s prefiksom "xsd".

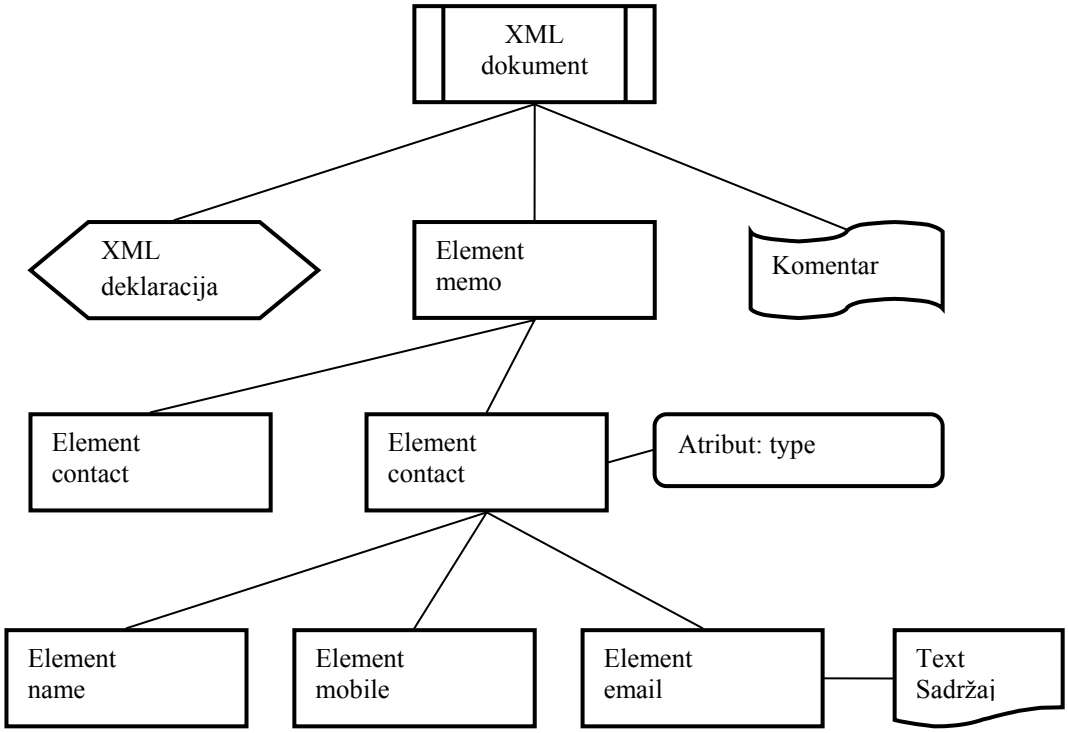
Gornja Schema određuje da dokument mora započeti s elementom <memo> koji može imati proizvoljan broj podelemenata <contact>. Svaki element <contact> mora atribut type="…" i mora imati tri podelementa: <name>, <mobile> i <email>.

Kako je nastala XML Schema? Nastajanje XML Scheme se može prikazati u nekoliko koraka:

Upravo XML Schema i DTD su mehanizmi pomoću kojih korisnici mogu striktno odrediti vlastite formate zapisa podataka i dokumenata. Ovi striktni opisi dozvoljenih formata su ključni za strojnu obradu razmijenjenih podataka. Programi mogu provjeriti da li je primljena informacija u određenom formatu i tako izbjeći potencijalne greške.

VIII. APSTRAKTNI PRIKAZ XML DOKUMENTA

XML možemo promotriti i na drugačiji način nego što smo ga opisivali do sada. XML možemo promatrati kao konkretnu implementaciju apstraktnog dokumentacijskog modela. Struktura XML dokumenta ocrtava hijerarhijsku strukturu kao što je prikazano na slici 1.



Slika 1: XML Document object model

Hijerarhijska struktura u formi stabla se sastoji od povezanih čvorova (objekata). Čvorovi su različitog tipa: "element", "atribut", "komentar",… Na početku stabla je čvor tipa "dokument" koji sadrži sve ostale dijelove. Neki čvorovi, kao što je npr. čvor tipa element, može imati podelemente ili pridružene atribute. Na vrhu hijerarhije elemenata je samo jedan element – korijenski element. Sadržaj elementa je spremljen u posebnom čvoru tipa "text".

Ovakav prikaz je apstraktniji od XML-a. On ne određuje kako se zapisuje ovakva struktura, a XML je konkretan format zapisa ovakvog apstraktnog modela.

Apstraktni prikaz dokumenta se koristi u programskoj obradi XML dokumenata pod engleskim nazivom "Document Object Model" ili skraćeno DOM. Prilikom obrade XML dokument se učitava u memoriju i sprema kao niz programskih objekata povezanih u stablastu strukturu. Programi "šeću" po stablu i mijenjaju, brišu ili dodaju pojedine čvorove – objekte.

IX. OBRADA XML DOKUMENATA

Obrade XML dokumenata uključuju prikaz XML-a, pronalaženje podataka unutar dokumenta, dodavanje novi elemenata i atributa i sl. Obradu obavljaju korisnički programi ili aplikacijski programi. Sve te raznolike poslove jednom riječju nazivamo "obrada". U posebnu skupinu programa za obradu možemo izdvojiti programe za provjeru valjanosti kao i programe za razvrstavanje (parsiranje) XML dokumenta, koji se u literaturi nazivaju validatorima odnosno XML procesorima.

Korisnički programi najčešće koriste usluge XML procesora i validatora. Obrada XML dokumenata korisničkim programima je vrlo jednostavna. Razlog za tu jednostavnost leži u gotovim i vrlo moćnim bibliotekama programa koji za nas obavljaju najveći dio posla. XML procesori i validatori su najčešće sastavni dijelovi biblioteke programa.

Jedan od glavnih razloga za eksplozivno širenje XML tehnologija je upravo to što je pisanje programa za obradu XML dokumenata jednostavno i što postoje gotove, jake biblioteke programa. Sama ideja XML-a, bez te jednostavnosti obrade, ne bi bila dovoljna za širenje XML-a. Najbolja potvrda ove tvrdnje je SGML meta-jezik. Taj je jezik odavno definiran kao vrlo moćno oruđe za opis podataka, ali nikada nije jače zaživio radi svoje prevelike kompliciranosti. Obrada SGML-a je bila prekomplicirana.

Obrada XML dokumenta se može promatrati unutar više okruženja:

- "Native" XML baze podataka (vidi npr. http://exist.sf.net ili http://www.sleepycat. com/products/xml.shtml). Složeni programski sustavi koji izravno spremaju XML dokumente ili imaju *XML sučelje prema relacionim bazama podataka*.

- Specijalizirani programski jezici (npr. XPath, XSLT,

XQuery). Posebno dizajnirani jezici i oruđa koja olakšavaju obrade XML dokumenata.

X. PRIMJENE XML-A

XML je tekstualni format zapisivanja dokumenata i

```
using System;
using System.IO;
using System.Xml;
using System.Xml.Schema;

/// <summary>Najjednostavniji XML validator.</summary>

public class Xvalidate {
    private static XmlValidatingReader vreader = null;
    private static Boolean success = true;

    public static void Main(string[] args) {
        try {
            XmlSchemaCollection xsc = new XmlSchemaCollection();
            xsc.Add("", args[1]);
            Console.WriteLine("Validating XML file: {0}...", args[0]);
            XmlTextReader reader = new XmlTextReader(args[0]);
            vreader = new XmlValidatingReader(reader);
            vreader.ValidationEventHandler +=
                    new ValidationEventHandler(ValidationCallBack);
            while (vreader.Read()) { }
            Console.WriteLine("Validation {0}",
                    (success == true ? "successful." : "failed."));
            Console.WriteLine();
        } catch (XmlException xe) {
            Console.WriteLine("Sintax error: " + xe.Message);
        }
        catch (Exception e) {
            Console.WriteLine("General exception: " + e.Message);
        }
        finally {
            if (vreader != null)
                vreader.Close();
        }
    }
    static void ValidationCallBack(object s, ValidationEventArgs a) {
        success = false;
        Console.Write("\r\nValidation error: " + a.Message + "\r\n");
    }
}
```

Primjer 4: C# program za validaciju

- Opći programski jezici (npr. Java, C#) s bibliotekama programa. Primjer 4 je kompletan program za validaciju XML dokumenta prema XML Schemi. Napisan je u programskom jeziku C# i koristi standardnu biblioteku programa koja dolazi uz C#. Ako bismo zanemarili 10-tak redaka programa, koji moraju biti prisutni u svakom programu, dolazimo do veličine programa od 30-tak redaka za vrlo složeni posao validacije. Kod pokretanja programa 1. parametar (args[0]) bi bila XML datoteka, a 2. parametar (args[1]) bi bila datoteka sa XML Schemom. (Za kompiliranje i pokretanje programa potrebno je .Net razvojno oruđe).

podataka, i kao takav ni na koji način ne ograničava područja svoje primjene. XML je neovisan o tipu računala, mreže ili operacijskog sustava koji koristimo. Ono što ćemo zapisivati u XML dokumente – sadržaj dokumenata - kao i format zapisa - imena oznaka - odrediti će korisnici.
Početna namjena tvoraca XML-a je bila primjena u webu. Sadržaj web stranica (pisan u HTML-u) je nestrukturiran. U HTML-u imamo samo oznake za prikaz, ali ne i oznake koje govore o *sadržaju* web stranice. Taj nedostatak je trebalo riješiti XML-om.

Pokazalo se, međutim, da je XML pogodan i za druge primjene. Štoviše primjena XML-a je doživjela pravu eksploziju i gotovo je nemoguće dati kompletan pregled primjene.

Navedimo samo neke općenite primjene:

- XHTML je inačica HTML-a napravljena prema XML preporukama
- Jezik za definiranje matematičkih formula za prikaz na webu (MathML)
- Jezik za opis općenitih tvrdnji i odnosa između tvrdnji (RDF XML)
- Web usluge s pratećim protokolima (UDDI, WSDL, SOAP) su utemeljene na TCP/IP protokolima i XML formatu informacija.
- Razmjena poslovnih dokumenata: ebXML (poslovne poruke, trgovina, poslovni procesi,…)
- Jezik za integraciju električkih sustava (CIM)
- Konfiguracijske datoteke za složene uređaje ili programske parametre u procesnom računarstvu
- Primjena koju će te Vi sami definirati.

Početna točka za pretragu mogućih primjena mogla bi biti poveznice:

http://www.oasis-open.org ili

http://xml.coverpages.org/xmlApplications.html

Primjena XML-a je često prekrivena razinom koja se bavi vrlo uskim područjem. Tako je npr. u Common Information Model-u (CIM/XML) XML iskorišten kao "jezik za opisivanje modela". Uglavnom se pod "model" podrazumijevaju modeli energetskih sustava. CIM specificira "sredstva, njihove odnose i svojstva u energetskim sustavima". No CIM je tek jedna razina iznad razine RDF. RDF je kratica od Resource Description Framework – vrlo općenitog okvira za opisivanje bilo kakvih entiteta. Taj okvir kao temelj koristi XML. Imamo dakle situaciju:

Program – korisnička aplikacija
CIM
RDF
XML

Korisnička aplikacija "zna" jedino za CIM. Aplikacija će npr. dobivati preko mreže podatke o modelima sustava iz "susjedstva" radi izrade planova razmjene i sl.. Ili će aplikacija biti implementacija IEC 61970 standarda koji definira razmjenu podataka između kontrolnih centara pa će na već prikazane dodati još jednu razinu. No goli XML je duboko ispod. Ovo je tipičan obrazac primjene XML-a koji se koristi kao temeljna tehnologija za definiranje i razmjenu polustrukturiranih podataka.

XI. Zaključak

XML tehnologija omogućuje jednostavno definiranje formata dokumenata i podataka. Prednost XML-a nad ostalim tehnologijama je njegova neovisnost, otvorenost i prenosivost. Iako je čitljiv običnim editorima i ljudima, XML je prvenstveno namijenjen strojnoj, programskoj obradi.
Programska obrada XML-a je jednostavna jer postoji velik broj oruđa i biblioteka programa koji ubrzavaju razvoj aplikacijskih programa.

XML je našao primjenu u izuzetno puno područja, a upravo njegovo svojstvo da je jednostavan i prilagodljiv korisničkim potrebama daje jamstvo za njegovu budućnost.

LITERATURA

[1] SGML - ISO 8879 http://www.iso.org

[2] Unicode http://www.unicode.org

[3] Početna stranica W3C organizacije: http://www.w3c.org

[4] XML standard: http://www.w3.org/TR/2004/REC-xml11-20040204

[5] XML Schema: http://www.w3.org/TR/xmlschema-0

[6] XML namespaces: http://www.w3.org/TR/REC-xml-names

[7] XSLT 2.0: http://www.w3.org/TR/2005/WD-xslt20-20050404

[8] On-line XML tutorial: http://www.w3schools.com/xml

[9] Steve Holzner, Inside XML, Pearson Education, 2000

[10] Sal Mangano, XSLT Cookbook, O'Reilly , 2002

[11] Brian Benz, John Durant, XML Programming Bible, John Wiley & Sons, 2003