UNIVERZITET U NOVOM SADU
PRIRODNO – MATEMATIČKI FAKULTET
DEPARTMAN ZA MATEMATIKU I INFORMATIKU

Goran Panić

RAZVOJ NAMENSKOG SISTEMA FAZI LOGIKE ZA PRIMENU U SISTEMIMA ZA UPRAVLJANJE XML DOKUMENTIMA

- DOKTORSKA DISERTACIJA -

Novi Sad, 2013.

PREDGOVOR

Doktorska disertacija pripada oblasti informacionih tehnologija i sistema, a bavi se definisanjem neodređenih informacija u XML-u. Osnovni cilj disertacije je proširenje postojećih sintaksi koje će omogućiti upotrebu neodređenosti u XML dokumentima i XQuery upitima. Od istraživanja se očekuje da kao rezultat ponudi softverski paket koji na jednom mestu objedinjuje i implementira sve mogućnosti definisanih sintaksi.

Višegodišnja istraživanja sprovedena na Univerzitetu u Novom Sadu od strane nekoliko autora imaju za cilj implementaciju sistema sposobnog da koristi fazi logiku sa prioritetima u relacionim bazama podataka. Autori su razvili kompletno aplikativno rešenje nad fazi relacionom bazom podataka. Korišćenjem GPFCSP koncepta opisanog u radovima objavljenim u ovim istraživanjima, pronađen je način za predstavljanje upita sa prioritetima u fazi relacionim bazama podataka, što je rezultiralo PFSQL upitnim jezikom. PFSQL omogućava postavljanje uslova u WHERE klauzulama upita, koji imaju različite stepene prioriteta. Tako GPFCSP daje teorijsku osnovu za realizaciju upita sa prioritetima. Autori su takođe uspeli da formalizuju PFSQL upite dobijanjem interpretacije u postojećoj fazi logici. Takođe je utvrđeno je da ŁΠ½ logika pruža dovoljno elemenata.

Doktorska disertacija predstavlja prirodan nastavak navedenih istraživanja i bavi se mogućnostima upotrebe fazi logike u definisanju neodređenosti u XML dokumentima i XQuery upitima.

Rad se sastoji od deset poglavlja:

1. Uvod
2. Teorijske osnove
3. Prethodni radovi i pravci istraživanja
4. Fazi XML sintaksa
5. Fazi XQuery upiti sa prioritetima
6. Razvoj aplikacije
7. Opis funkcionalnosti Fazi XML editora
8. Analiza aplikacije
9. Mogućnosti praktične primene
10. Zaključak

Prvo poglavlje navodi razloge koji su doveli do potrebe za realizacijom doktorske disertacije i daje kratak opis sadržaja disertacije.

Drugo poglavlje navodi osnovne teorijske definicije iz oblasti XML sintakse, XML tehnologija, fazi skupova, fazi logike i generalizovanih problema zadovoljenja fazi ograničenja sa prioritetima ili GPFCSP-a. Tačnije, opisane su tehnologije i date definicije potrebne za razumevanje upotrebe fazi logike u definisanju neodređenosti XML-a i upotrebu prioriteta u XQuery upitima.

Treće poglavlje daje pregled dosadašnjih istraživanja sprovedenih na fakultetu i analizu naučnih radova iz oblasti upotrebe fazi logike u definisanju neodređenosti u XML dokumentima.

U četvrtom poglavlju opisana je fazi XML sintaksa, koja predstavlja proširenje standardnog XML-a i omogućava definisanje neodređenosti u vrednostima i strukturi XML dokumenta. Sintaksa je opisana pomoću XSD i DTD dokumenata. Na početku je dat fazi XML samo sa aspekta upotrebe neodređenosti u vrednostima XML elemenata, uključujući obradu Null vrednosti, dok se u nastavku obrađuje definisanje neodređenosti u strukturi XML dokumenata.

Razlozi za proširenje standardne XQuery upitne sintakse fazi elementima opisani su u petom poglavlju. Poglavlje daje definiciju fazi XQuery upitne sintakse, koja predstavlja proširenje standardne XQuery sintakse. Nakon toga se fazi XQuery proširuje prioritetima i pragovima zadovoljenja, čija je upotreba moguća zahvaljujući GPFCSP-u. Precizna definicija fazi XQuery upitne sintakse pomoću EBNF gramatike i opis izvršenih proširenja, takođe je data u ovom poglavlju. Na kraju su navedene preporuke za obradu upita u slučaju pojave Null vrednosti.

Šesto poglavlje na osnovu sintaksi definisanih u prethodnim poglavljima opisuje modelovanje aplikacije *Fazi XML Editor* upotrebom UML dijagrama. Opisane su tehnologije korišćene u razvoju aplikacije. Na kraju je objašnjen proces same implementacije korišćenjem WPF tehnologije i .NET-a, kao i delovi XAML-a i C# koda bitni za razumevanje procesa implementacije.

Sedmo poglavlje daje pregled funkcionalnosti kreirane aplikacije. Opisan je proces instalacije aplikacije i njeno inicijalno pokretanje, skladištenje dokumenata, rad sa standardnim i fazi XML, DTD i XSD dokumentima. Upotreba implementiranog *Fazi XQuery editora* u izvršenju fazi XQuery upita sa prioritetima, navedena je na kraju poglavlja. Internet lokacija sa koje se može preuzeti izvršna verzija aplikacije, video datoteke i dodatne datoteke za testiranje, takođe je navedena u ovom poglavlju.

Osmo poglavlje vrši detaljnu analizu kreirane aplikacije i to kroz analizu korisničkog interfejsa sa aspekta funkcionalnosti, testiranja performansi sistema i provere stabilnosti aplikacije.

Deveto poglavlje navodi primere koji na razumljiv način opisuju mogućnosti primene sintaksi i razvijenog softverskog paketa u praksi.

Poslednje poglavlje daje zaključak o značaju i doprinosu teze, kao i predloge za dalje pravce istraživanja u ovoj oblasti.

Zahvaljujem se komisiji koja je detaljno pregledala rad i svojim korisnim predlozima uticala na konačnu verziju disertacije. Posebnu zahvalnost dugujem mentoru prof dr Milošu Rackoviću i prof dr Srđanu Škrbiću na nesebičnoj podršci u toku izrade disertacije.

Zahvaljujem se porodici na razumevanju i podršci.

Novi Sad, 10.29.2013 Goran Panić

SADRŽAJ

Glava I

UVOD

Doktorska disertacija predstavlja rezultat višegodišnjeg rada autora na pronalaženju načina za definisanje neodređenosti u XML dokumentima i XQuery upitima, kao i implementaciji istih. Na ovu temu autor je napisao veći broj seminarskih radova i objavio dva naučna rada koji ukratko definišu dostignuća naučnog istraživanja, a koja će biti detaljno opisana u samoj disertaciji.

XML (eXstensible Markup Language) jezik je u protekloj deceniji napravio pravu revoluciju na području skladištenja i prenosa podataka, sa tendencijom daljeg širenja. Jezik je u isto vreme čitljiv i ljudima i mašinama, sa akcentom na proširivosti i jednostavnosti. Na njegovim osnovama je razvijen veliki broj formalno definisanih jezika. Većina informacija u realnom svetu dolazi u obliku nepreciznih ili nepotpunih vrednosti (na primer brza kola ili visoka zgrada). Standardni XML nije predviđen da sadrži takve informacije. Vrednosti elemenata i atributa su po definiciji apsolutno tačne, a struktura XML dokumenta precizno definisana. Fazi skupovi i fazi logika predstavljaju teorije koje su svoju primenu našle u mnogim sličnim oblastima koje rade sa neodređenostima. Sam koncept fazi logike je mnogo prirodniji od koncepta klasične logike. Njenom upotrebom probijaju se ograničenja klasične logike i omogućava definisanje nepreciznih i nepotpunih informacija.

Proširenje XML-a elementima fazi logike daje mogućnost definisanja nepreciznih ili nedovoljno određenih informacija u XML dokumentima. Kako je XML po definiciji proširiv jezik, nije ga potrebno dodatno proširivati, već je moguće iskoristiti postojeću sintaksu. Najjednostavniji način za uvođenje fazi logike u XML je upotrebom XML šeme. Takve šeme sadržale bi definicije za opisivanje fazi skupova i njihovih funkcija pripadnosti, koji bi služili za opis vrednosti neodređenih elemenata i definisanje neodređenosti u strukturi XML dokumenta.

Drugi korak u definisanju fazi XML-a je definisanje fazi upitne sintakse. XML upitne sintakse omogućavaju jednostavno rukovanje XML dokumentima i jednako su zaslužne za uspeh XML standarda koliko i sam jezik. Jednostavan način za definisanje fazi XML upitne sintakse je proširenje neke standardne XML upitne sintakse fazi elementima. Najznačajniji primer XML upitne sintakse je XQuery, a njegovo proširenje se nameće kao prirodno rešenje. Proširenje XQuery upitne sintakse fazi elementima, omogućilo bi upotrebu fazi vrednosti u definisanju upita. Ovako definisan upit u stanju je da generiše preciznija rešenja.

Ne moraju svi parametri nekog objekta imati isti uticaj na donošenje krajnjeg zaključka o njemu. Da bi se ova osobina mogla korektno obraditi potrebno je precizno definisati stepen uticaja posmatranog parametra na krajnji rezultat. Na taj način se za svaki parametar može precizno odrediti njegov prioritet u donošenju krajnjeg zaključka. Neophodno je primetiti da mogu postojati ograničenja koja se ne smeju prelaziti, jer u slučaju njihovog probijanja uticaj parametra na krajnji rezultat naglo raste i postaje dominantan. Takva ograničenja se nazivaju pragovima zadovoljenja. Jedan od načina uvođenja prioriteta i pragova zadovoljenja u fazi XQuery upitnu sintaksu moguć je upotrebom GPFCSP-a (Generalized Priority Fuzzy Constraint Satisfaction Problem). GPFCSP je definisan u

prethodnim istraživanjima sprovedenim na Univerzitetu u Novom Sadu, a kratak opis sa potrebnim definicijama biće kasnije naveden u radu.

Nedostatak potrebnih informacija česta je pojava kod realnih sistema. Potrebno je definisati vrednosti pomoću kojih se mogu opisati nedostajući podaci u XML-u. Tradicionalni sistemi baza podataka rešenje ovog problema baziraju na uvođenju Null vrednosti. Međutim, postoji više razloga zbog kojih informacije mogu nedostajati, na primer vrednosti postoje ali su nedostupne, vrednosti nije moguće meriti… Pošto različiti tipovi nedostatka vrednosti podrazumevaju i različite načine obrade, potrebno je definisati nekoliko tipova Null vrednosti i njihove obrade. Definisane Null vrednosti trebale bi da pokriju sve uzroke nastanka nedostatka informacija.

Primena fazi logike u definisanju neodređenosti u XML-u česta je tema naučnih radova. Međutim, objavljeni radovi akcenat stavljaju na postavljanje teorija i definisanje sintaksi, a samo mali broj se upušta u proces implementacije istih. Ukoliko bi i postojale implementacije, one se svode na veoma ograničen skup instrukcija, koje najčešće ne ispunjavaju ni osnovne funkcionalnosti za bilo kakvu praktičnu upotrebu. Zbog toga je posebnu pažnju potrebno posvetiti implementaciji, analizi i primeni definisanih sintaksnih rešenja. Nivo implementacije poželjno je dovesti do nivoa praktično upotrebljivog alata, kako bi se kroz testiranje u realnom okruženju došlo do najboljeg rešenja.

Rešavanje gore navedenih problema predstavlja razlog za sprovođenje ovog istraživanja. Cilj disertacije je proširenje postojećih sintaksi tako da se omogući upotreba neodređenosti u XML dokumentima i XQuery upitima. Takođe, potrebno je omogućiti obradu prioriteta i pragova zadovoljenja kao i pravilan rad u slučaju pojave Null vrednosti. Proizvod istraživanja bi trebalo da bude softverski paket, koji na jednom mestu objedinjuje i implementira sve mogućnosti definisanih sintaksi.

Istraživanje se može smatrati uspešnim ukoliko se kao krajnji rezultat dobije:

- sintaksa za opisivanje neodređenosti u vrednostima i strukturi XML dokumenata, koja treba da objedini i proširi najbolja postojeća rešenja u ovoj oblasti;
- sintaksa za postavljanje neodređenih uslova u XQuery upitima. Kreirani fazi XQuery upiti treba da budu zasnovani na GPFCSP-u i da podržavaju rad sa prioritetima i pragovima zadovoljenja, kao i pravilnu obradu Null vrednosti;
- softverski paket koji implementira definisane sintakse iz prethodne dve stavke. Aplikacija mora biti testirana i analizirana;
- praktična potvrda mogućnosti sintaksi i kreiranog alata, nad realnim primerima.

Glavni pozitivan efekat ovog istraživanja je potvrda mogućnosti razvoja sveobuhvatnog fazi XML sintaksnog i aplikativnog rešenja primenljivog u praksi. Dobijeno sintaksno rešenje i razvijena aplikacija, kao krajnji rezultati ovog istraživanja, mogli bi se primeniti u svim oblastima u kojima postoji potreba za skladištenjem i obradom neodređenih informacija zapisanih u XML dokumentima. Na taj način potencijalnim korisnicima će biti ponuđeno praktično primenljiv alat zadovoljavajućih performansi, koji omogućava upotrebu neodređenih vrednosti u XML dokumentima.

Disertacija će biti podeljena na nekoliko međusobno zavisnih logičkih celina, koje će se baviti:

- opisom XML tehnologija i matematičkih formalizama (fazi logika) neophodnih za realizaciju disertacije;

- analizom postojećih naučnih dostignuća u ovoj oblasti;
- definisanjem fazi XML sintakse i fazi XQuery upitne sintakse sa prioritetima, kao i problemom obrade Null vrednosti;
- razvojem softverskog paketa, sa opisom funkcionalnosti i testiranjem;
- pronalaženjem i opisivanjem praktičnih primera koji će pokazati mogućnosti primene istraživanja i razvijenog softverskog paketa nad realnim sistemima;
- analizom i donošenjem zaključka o rezultatima istraživanja.

Osnovne teorijske definicije iz oblasti XML sintakse, XML tehnologija, fazi skupova, fazi logike i generalizovanih problema zadovoljenja fazi ograničenja sa prioritetima ili GPFCSP-a, biće opisane na početku rada.

Druga logička celina će dati pregled dosadašnjih istraživanja i analizu naučnih radova iz oblasti upotrebe fazi logike u definisanju neodređenosti u XML dokumentima.

Naredna celina će na osnovu prethodno navedenih problema u vezi definisanja neodređenosti u XML dokumentima i načinima njihovog rešavanja definisati fazi XML sintaksu upotrebom DTD i XSD šema. Biće razvijena fazi XQuery upitna sintaksa, kao proširenje standardne XQuery sintakse fazi elementima. Nova sintaksa će podržati upotrebu prioriteta i pragova zadovoljenja u uslovima upita. Problem nedostajućih informacija biće rešen uvođenjem dva tipa Null vrednosti, nepoznata i neprimenljiva. Biće objašnjeni načini definisanja oba tipa i date preporuke za njihovu obradu.

U celini koja opisuje razvoj softverskog paketa pre početka implementacije biće izvršeno modeliranje sistema upotrebom objektno orijentisanih metodologija razvoja aplikacija i UML dijagrama. Sam proces implementacije biće sproveden pomoću WPF tehnologije i .NET-a. Aplikacija će biti razvijana u C# programskom jeziku, a njen interfejs upotrebom XAML jezika. Kako bi se povećale mogućnosti i dobile što bolje performanse sistema, aplikacija će se oslanjati na MSSQL bazu podataka kao osnove za razvoj fazi XQuery upitne sintakse i aplikaciju MATLAB za obradu kompleksnih matematičkih operacija. Sastavni deo doktorske disertacije činiće izvršna verzija aplikacije, video datoteke i ostale test datoteke, koje će biti dostupne putem Interneta. Osnovne funkcionalnosti kreirane aplikacije kao što su proces instalacije, inicijalno pokretanje, skladištenje dokumenata, rad sa standardnim i fazi XML, DTD i XSD dokumentima, kreiranje i izvršenje fazi XQuery upita sa prioritetima biće detaljnije opisane u disertaciji. Krajnja verzija aplikacije biće podvrgnuta detaljnoj analizi i testiranju, što će obuhvatati analizu izgleda korisničkog interfejsa sa aspekta funkcionalnosti, testiranje performansi sistema i proveru stabilnosti aplikacije. Cilj testiranja je praktična provera definisanih sintaksi i donošenje zaključka o kvalitetu urađene implementacije.

Pretposlednja celina će za definisana sintaksna rešenja i implementirani softverski paket dati primere upotrebe u realnim sistemima, kako bi se pokazale mogućnosti njihove praktične primene.

Na kraju disertacije biće urađena analiza i donet zaključak o značaju i doprinosu teze, kao i dati predlozi za dalje pravce istraživanja u ovoj oblasti.

Glava II

TEORIJSKE OSNOVE

Poglavlje daje osnovne teorijske definicije iz oblasti XML sintakse, XML tehnologija, fazi skupova, fazi logike i generalizovanih problema zadovoljenja fazi ograničenja sa prioritetima. To su teorije i tehnologije neophodne za razumevanje upotrebe fazi logike u definisanju neodređenosti XML-a i upotrebu prioriteta u XQuery upitima.

2.1 XML

Knjiga (Harold & Means, 2004) i zvaničan sajt W3C organizacije predstavljaju izvor informacija korišćenih za opis sintakse XML jezika i njegovih tehnologija.

XML je meta jezik za označavanje tekstualnih dokumenata. Jezik je standardizovan od strane međunarodne organizacije W3C (World Wide Web Consortium) za izradu Internet standarda. Veliki broj formalno definisanih jezika za označavanje baziran je na XML sintaksi.

Stvaranje jezika koji će biti jednostavan i čitljiv za ljude i računarske programe istovremeno, predstavlja osnovnu ideju kreatora jezika. Zbog toga su XML datoteke u standardnom tekstualnom formatu, a sintaksa se sastoji od čoveku čitljivih tagova unutar kojih se definišu vrednosti podataka. Osnovna svrha XML je da olakša razmenu podataka između različitih informacionih sistema, sa posebnom naznakom na sisteme koji su povezani sa Internetom. Podaci definisani u XML formatu su platformski nezavisni, a za njihovo čitanje se može koristiti svaka programska alatka koja podržava rad sa tekstualnim fajlovima.

Usled gomilanja različite tehničke dokumentacije u kompaniji IBM (International Business Machines Corporation) šezdesetih godina prošlog veka, započinje se sa izradom prvog šire korišćenog jezika za označavanje podataka. Na njegovoj izradi su angažovani Charles Goldfarb, Ed Mosher i Ray Lorie, a dobija ime **GML** (Generalized Markup Language). Ovaj projekat se pokazao kao uspešan i nastavilo se sa daljim razvojem. Rezultat toga je **SGML** (Standard Generalize Markup Language) koji je ISO 1986. godine prihvatio kao standard 8879. SGML je bio dovoljno formalizovan da može da garantuje vernost dokumenata, dovoljno strukturiran da može da operiše kompleksnim dokumentima i dovoljno otvoren da može podržati rukovanje velikim količinama podataka. **HTML** (HyperText Markup Language) je jedan od šire poznatih jezika koji je nastao na osnovama SGML. Tim Berneres Lee je izdvojio skup etiketa iz SGML-a i primenio ih na formatiranje dokumenata sa posebnim akcentom na Web stranice.

Problem SGML-a je to što je bio jako opširan, složen za korišćenje i skup za upotrebu, te se iz tog razloga nije raširio. Jon Bosak, Tim Bray, C. M. Sperberg-McQueen i James Clark i drugi, su 1996. godine započeli rad na uprošćavanju SGML jezika. Cilj je bio izbacivanje suvišnih, zbunjujućih i nekorisnih elementa, uz zadržavanje isključivo korisnih funkcionalnosti. Pravci razvoja novog jezika, koji je dobio naziv XML, definisani su pomoću deset tačaka:

- XML mora biti direktno primenljiv preko Interneta
- XML mora podržavati širok spektar primena
- XML mora biti kompatibilan sa SGML-om
- XML mora omogućiti lako pisanje programa za procesiranje XML dokumenata
- broj opcionih mogućnosti u XML-u mora biti apsolutno minimalan, a u idealnom slučaju jednak nuli
- XML dokumenti moraju biti čitljivi ljudima i u razumnoj meri jednostavni
- standard mora biti specifikovan što pre
- dizajn XML-a mora biti formalan i precizan
- kreiranje XML dokumenta mora biti jednostavno
- sažetost kod označavanja dokumenta XML-om je od minimalnog značaja

Organizacija W3C objavljuje 1998. godine ovaj standard, pod nazivom XML 1.0. Nova verzija XML 1.1 pojavila se 2004. godine i u odnosu na prethodnu, proširena je novim karakterima kojima je u međuvremenu dopunjen Unikod standard.

Dobar deo onoga što je zacrtano u razvoju jezika je i ostvareno, a XML je postao najrasprostranjeniji meta jezik. Svoju popularnost duguje dobrim osobinama, a neke od njih su:

- XML je čitljiv i čoveku i računaru
- kako je XML dokument obično tekstualna datoteka, to praktično znači da je čitljiv na svakoj platformi koja može čitati tekstualne podatke
- tekstualna struktura omogućava da na XML ne utiču tehnološke promene
- XML podržava Unikod standard što omogućava prikaz teksta na svim poznatim jezicima koji su obuhvaćeni tim standardom
- tagovi opisuju sadržaj koji se nalazi unutar njih, tako da format sam sebe dokumentuje
- zbog stroge sintakse jednostavno je kontrolisati ispravnosti XML dokumenta
- XML je međunarodno prihvaćen standard koji koriste mnogi proizvođači programa
- kompatibilan je sa SGML-om
- predstavlja jezik za meta-označavanje, što podrazumeva da nema fiksan skup oznaka i elemenata
- jezik je proširiv

Međutim nije sve savršeno u XML-u. Neke od mana ovog meta jezika su prikazane u nastavku:

- krajnje redundantna i opširna sintaksa dovodi do usporavanja programa koji vrše obradu dokumenata, a mogu zbuniti i čoveka koji čita dokument
- XML nije programski jezik, te je sa njim jedino moguće dokumentovati podatke, dok se za njihovu obradu mora koristiti neka druga tehnologija
- XML nije u stanju da komunicira preko mreže, a samim tim nije ni protokol za mrežni prenos. Za prenos XML-a preko mreže mora se upotrebljavati neki od mrežnih protokola poput HTTP, FTP, NFS…
- XML nije baza podataka. Iako baza podataka može sadržati XML podatke, ona nije XML dokument

2.1.1 XML dokumenti i deklaracije XML-a

XML dokument predstavlja tekstualni dokument koji može sadržati isključivo tekstualni sadržaj, dok se podaci binarnog tipa u njemu ne mogu pojaviti. Najosnovniji primer XML dokumenta dat je u listingu 2.1.

```
<vrednost>50,00 DIN</vrednost>
```

Listing 2.1. Jednostavan XML koji definiše vrednost.

Ovako definisan tag XML-a može predstavljati sadržaj **XML datoteke**, koja može imati proizvoljno ime, sa ekstenzijom .xml. Kao na primer cena_karte.xml. Na samom početku XML dokumenta može se nalaziti deklaracija XML-a, koja može sadržati pseudo-atribute: *version*, *encoding* i *standalone*. Atribut **version** označava verziju XML-a, a može sadržati vrednosti 1.0 i 1.1. Skup znakova korišćenih u dokumentu može se definisati u atributu **encoding**, koji je opcioni i ako se ne navede podrazumeva se vrednost Unicode. Poslednji atribut je **standalone**, koji može imati vrednost "yes" ako je samostalan, ili "no" ako nije. Ukoliko element nije samostalan, tada aplikacija može učitavati spoljni DTD, radi utvrđivanja pravih vrednosti u nekim delovima dokumenta. Atribut je opcioni, a podrazumevana vrednost je "no". Primer potpune deklaracije XML dokumenta prikazan je u listingu 2.2.

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

Listing 2.2. XML deklaracija.

2.1.2 Elementi i tagovi

Svaki element mora sadržati početni i završni tag, a sve što se nalazi između, uključujući i prazna mesta, predstavlja sadržaj elementa. Tako na primer u prethodnom primeru postoji jedan element *<vrednost>50,00 DIN</vrednost>*, koji nosi naziv *vrednost*. Taj element se sastoji iz početne oznake *<vrednost>*, sadržaja elementa 50,00 DIN i završava oznakom *</vrednost>*. Početna oznaka počinje znakom < , a završava se znakom >, dok završna oznaka počinje znakom </ , a završava se znakom >. Između oznaka se nalazi ime elementa (u prethodnom primeru ime je "*vrednost*"), koje može biti proizvoljno i po pravilu opisuje sadržaj unutar elementa. U XML-u može postojati element bez vrednosti ili prazan element. Za prikaz ovakvog elementa, može se koristiti postojeća pravila prikazivanja elementa. U tom slučaju bi sintaksa izgledala kao u listingu 2.3.

```
<vrednost></vrednost>
```

Listing 2.3. Prazan XML elemenat.

Postoji kraći način za pisanje praznog elementa. Primer je dat u listingu 2.4. Korisnik može da odabere koji će način definisanja praznog elementa koristiti.

```
<vrednost/>
```

Listing 2.4. Skraćeno definisanje praznog elemenat.

Pri definisanju imena tagova, dozvoljena je upotreba i velikih i malih slova. Međutim, bitno je napomenuti da XML standard razlikuje velika i mala slova. Tako na primer tag *<vrednost>* nije isto što i *<VreDnosT>*, a početni tag *<vrednost>* se ne može zatvoriti završnim tagom *</VREDNOST>*.

2.1.3 Struktura XML-a

XML dokument ima tačno jedan **korenski element**, koji u sebi direktno ili indirektno sadrži sve druge elemente dokumenta. Taj elemenat se uvek nalazi na vrhu dokumenta, neposredno ispod deklaracije. Listing 2.1 predstavlja najprostiju strukturu XML dokumenta, koja sadrži samo jedan elemenat, a koji naravno predstavlja korenski elemenat. Da XML dokumenat može biti složeniji može se videti u listingu 2.5.

```
<porudzbina>
    <brojPorudzbine>1000</brojPorudzbine>
    <kupac>
        <ime>Goran</ime>
        <prezime>Panić</prezime>
        <adresa>
            <drzava>Srbija</drzava>
            <grad>Novi Sad</grad>
            <postanskiBroj>21000</postanskiBroj>
            <ulica>Matice Srpske</ulica>
            <broj>11</broj>
        </adresa>
    </kupac>
    <artikal>
        <oznakaArtikla>874</oznakaArtikla>
        <naziv>Knjiga</naziv>
        <cena>2000,00</cena>
        <komada>1</komada>
    </artikal>
    <artikal>
        <oznakaArtikla>222</oznakaArtikla>
        <naziv>Olovka</naziv>
        <cena>20,00</cena>
        <komada>3</komada>
```

```
    </artikal>
</porudzbina>
```

Listing 2.5. Kompleksan XML dokument – Porudžbina.

Svaki element može imati nula ili više elemenata potomaka, pravilno ugnježdenih u njega samog, a koji takođe mogu imati svoje potomke. Element ima samo jednog roditelja kome pripada, izuzev korenog elementa koji nema roditelja. Bratski elementi su svi elementi koji imaju zajedničkog roditelja. Na primer, u prethodnom primeru imamo korenski element <*porudzbina*>, koji sadrži četiri elementa potomka: <*brojPorudzbine*>, <*kupac*> i dva elementa sa imenom <*artikal*>. Za element <*brojPorudzbine*> kažemo da ne sadrži elemente potomke, ali zato ima roditelja <*porudzbina*> i još tri bratska elementa. Elemenat <*artikal*> takođe ima roditelja <*porudzbina*> i tri bratska elementa, od kojih je element <*artikal*> istog tipa kao on i nose isto ime.

Elementi definisani u listingu 2.5 imaju dve vrste sadržaja: sadržaj definisan znakovnim podacima i sadržaj koji sadrži elemente potomke. Takva vrsta XML sadržaja se u praksi najčešće sreće. Međutim sadržaj elementa XML-a se može definisati i kombinacijom znakovnih sadržaja i elemenata potomaka u istom XML elementu. Dokumenti sa ovako definisanim sadržajem nazivaju se XML dokumentima mešovitog sadržaja. Tako se XML dokument iz listinga 2.5 može definisati upotrebom mešovitog sadržaja kao u listingu 2.6.

```
<porudzbina>
    1000
    <kupac>
        Goran Panić
        <adresa>Srbija Novi Sad 21000 Matice Srpske 11</adresa>
    </kupac>
    <artikal>
        874 Knjiga 2000,00
        <komada>1</komada>
    </artikal>
    <artikal>
        222 Olovka 20,00
        <komada>3</komada>
    </artikal>
</porudzbina>
```

Listing 2.6. Definisanje porudžbine upotrebom mešovitog sadržaja XML-a.

Ovakav vid kreiranja XML dokumenata pogodan je za upotrebu u dokumentima čija je sadržina knjiga, članak, priča, Web stranica ili bilo koji oblik teksta. Rad sa ovakvim dokumentima je mnogo komplikovaniji od rada sa dokumentima nemešovitog sadržaja.

2.1.4 Atributi

XML elementi mogu sadržati **atribute** unutar svojih početnih tagova. Atribut je kombinacija imena i vrednosti koji se razdvajaju znakom jednakosti i eventualno razmakom. Vrednosti atributa se moraju nalaziti unutar navodnika ili polunavodnika. Polunavodnici su izuzetno korisni ako vrednost atributa sadrži navodnike. Dva identična primera XML elementa koji sadrže atribute navedena su u listingu 2.7.

```
<kupac ime="Goran" prezime="Panić">
    <drzava>Srbija</drzava>
    <grad>Novi Sad</grad>
    <ulica>Matice Srpske 11</ulica>
</kupac>
<kupac ime = 'Goran' prezime = 'Panić'>
    <drzava>Srbija</drzava>
    <grad>Novi Sad</grad>
    <ulica>Matice Srpske 11</ulica>
</kupac>
```

Listing 2.7. Dve identične definicije atributa.

O tome da li je dobro koristiti atribute ili ne nisu sasvim usaglašena mišljenja. Sigurno je da se atributi mogu koristiti ako ne utiču na sam sadržaj elementa, kao na primer promena boje slova. Definisanje dela sadržaja pomoću atributa dovodi do smanjenja broja elemenata potomaka, ali se gubi na preglednosti dokumenta.

2.1.5 Imenovanje, reference i CDATA sekcije

Pravila imenovanja su ista za sve stavke u XML-u. To znači da se pravila koja definišu imena XML elemenata, mogu upotrebljavati za definisanje imena njihovih atributa ili drugih komponenti. Skup pravila za definisanje imena u XML-u nazivaju se **pravilima XML imenovanja**.

Ime može sadržati velika i mala engleska slova, brojeve, neengleska slova, kao i znakove interpunkcije: tačka (.), donja crta (_) i crtica (-). Verzija XML 1.0 može sadržati Unicode 2.0 znakove, dok verzija XML 1.1 može sadržati Unicode 3.0 znakove. Upotreba drugih znakova interpunkcije: navodnici (“), polunavodnici (‘), znak za dolar ($), kapica (^), procenat (%), tačka-zarez (;), kosa crta (/)… , nije dozvoljena. Upotreba „belih“ znakova takođe nije dozvoljena. Poslednje ograničenje vezano je za davanje imena koja počinju nizom znakova “XML” (velika i mala slova), koja su rezervisana za potrebe standardizacije XML-a od strane W3C organizacije.

Imena u XML-u smeju da počinju slovom, ideogramom ili znakom donja crta, dok dužina samog imena nije ograničena standardom. Postoje neki znakovi čija je upotreba dozvoljena u XML dokumentima, ali samo pod određenim uslovima. Tako, znakovni podaci unutar elementa ne smeju sadržati znak < , ako ne postoji njegova izlazna sekvenca </ . Ako bi korisniku ipak zatrebao ovaj znak, on se može dobiti upotrebom **reference** i to na tri načina:

- *<* - pomoću reference entiteta
- *<* - pomoću numeričke reference znaka, &# + pozicija elementa u Unicode sistemu.
- *<* - pomoću heksadecimalne reference znaka, &# + x + heksadecimalno konvertovan Unicode broj elementa.

Kada aplikacija analizira XML dokumenta, markiranja (referenciranje spada u markiranje) se zamenjuju onim znakom na koji upućuje referenca. U tabeli 2.1 prikazane su unapred definisane reference entiteta.

ZNAK	NAZIV	OPCIONI
&amp;	ampersend (&)	Ne
&lt;	manje od (<)	Ne
&gt;	veće od (>)	Da
&quot;	navodnik (")	Da
&apos;	polunavodnik (’)	Da

Tabela 2.1. Predefinisane reference entiteta.

Reference entiteta i znakova, mogu se koristiti samo u sadržaju elemenata i vrednostima atributa. Upotreba u imenima elemenata, imenima atributa, kao i u drugim vrstama markiranja nije dozvoljena.

Ukoliko dokument sadrži velik broj znakova < i &, tada kodiranje može da bude zamorno. Lakši način da se reši ovaj problem je upotreba **CDATA sekcije**. CDATA sekcija započinje skupom znakova *<![CDATA[* , a završava se znakovima *]]>* . Sve što se nalazi između tretira se kao običan znakovni kod. Primer je dat u listingu 2.8.

```
<![CDATA[  <testKod> <  &  >
      “ ‘ </testKod>
      Neki tekst
]]>
```

Listing 2.8. CDATA sekcije.

Postoji samo jedno ograničenje po pitanju pojave znakova unutar CDATA sekcije, a to je upotreba završnog skupa znakova *]]>* . Jasno je da će na pojavu takvog niza program reagovati kao na kraj CDATA sekcije.

2.1.6 Komentari i instrukcije za obradu

Koliko god se kreator XML dokumenta trudio da zadavanjem dobro definisanih imena elementima kod bude čitljiv budućem korisniku ili njemu samom, nekada to nije tako jednostavno izvesti. Neretko je potrebno dodatno dokumentovati funkciju određenih elemenata u kodu. Ovaj problem se u programskim jezicima rešava uvođenjem komentara, a isti način je i ovde rešen. **XML komentari** počinju zadavanjem niza znakova *<!--* , a završava se sa --> . Između ova dva znaka se nalazi tekst komentara. Primer XML komentara dat je u listingu 2.9.

```
<!-- Ovde se nalazi neki komentar. -->
```

Listing 2.9. Komentar.

Komentari se mogu nalaziti bilo gde u dokumentu. Kako komentari nisu XML elementi ne postoji nikakva prepreka da budu navedeni pre ili posle korenskog elementa. Međutim oni se ne mogu nalaziti unutar drugih komentara, niti unutar tagova.

Upotreba komentara je isključivo ograničena za pomoć čoveku, iako ih neka nestandardna proširenja koriste da u njih ubace određeni skript kod, koji je čitljiv programu. Pravilnije je u tu svrhu koristiti **instrukcije za obradu**. One počinju skupom znakova *<?* , a završavaju se skupom znakova *?>*. Neposredno posle *<?* dolazi XML ime cilja, što može biti ime aplikacije kojoj je ta instrukcija namenjena ili identifikator instrukcije. Sadržaj predstavlja tekst u formatu koji odgovara aplikaciji koja će čitati takav XML dokument. Primer instrukcije za obradu dat je u listingu 2.10.

```
<?startAPP id="0" iteration="50" input="no"?>
```

Listing 2.10. Instrukcije za obradu.

Najčešće upotrebljavana instrukcija za obradu je *xml-style-sheet* koja dokumentu pridružuje opis stilova. Poput komentara i instrukcije se mogu pisati bilo gde u dokumentu, ali ne i u oznakama. Takođe, ciljno ime ne može nositi naziv *xml*, iz razloga preklapanja sa deklaracijom.

2.1.7 DTD

Ponekad je nad nekim XML dokumentom ili grupom istih, potrebno definisati dozvoljeni skup tagova, njihov raspored u dokumentu, tip sadržaja elementa ili njihovih atributa. U tu svrhu se upotrebljava **XML šema**. XML šema predstavlja dokument koji opisuje šta sve validan XML dokument sme sadržati. Upotreba šema u XML-u je opciona, a ne obavezujuća. Načini povezivanja XML šeme sa XML dokumentima, zavisiće od jezika na kome je šema napisana. Povezivanje se može definisati u samom XML dokumentu ili izvan njega.

DTD (Document Type Definition) je jedini šema jezik definisan u samoj specifikaciji XML-a. Zbog tih razloga spada u najrasprostranjeniji XML šema jezik. Loše osobine su ono što prati DTD, zbog čega je pokrenuto pitanje njegovog izbacivanja iz buduće sintakse XML-a, počev od verzije 2.0. Neke od loših osobina DTD-a:

- sintaksa nije XML tipa
- nema tipizacije podataka, naročito za sadržaj elemenata
- ograničena proširivost
- minimalna kompatibilnost sa namespaces
- ne može se koristiti mešani sadržaj i istovremeno nametnuti redosled i broj pod-elemenata
- ne može se nametnuti broj pod-elemenata bez nametanja redosleda

Pretpostavimo da imamo XML dokument koji izgleda kako u listingu 2.11.

```
<?xml version="1.0" ?>
<kupac>
      <ime>Goran</ime>
      <prezime>Panić</prezime >
      <grad>Novi Sad</grad>
      <ulica>Matice Srpske 11</ulica>
</kupac>
```

Listing 2.11. Testni XML dokumenat.

Za XML dokumenat iz prethodnog listinga odgovarala bi DTD šema prikazana u listingu 2.12. Dokument koji bi sadržao ovakvu šemu nosio bi ekstenziju ***dtd***.

```
<!ELEMENT kupac (ime, prezime, grad, ulica)>
<!ELEMENT ime (#PCDATA)>
<!ELEMENT prezime (#PCDATA)>
<!ELEMENT grad (#PCDATA)>
<!ELEMENT ulica (#PCDATA)>
```

Listing 2.12. DTD šema.

Referenciranje na postojeću šemu iz XML dokumenta može se postići dodavanjem koda kao u listingu 2.13.

```
<?xml version="1.0" ?>
<!DOCTYPE note SYSTEM http://www.test.com/kupac.dtd">
<kupac>
      ...
</kupac>
```

Listing 2.13. Povezivanje DTD šeme i XML dokumenta.

2.1.8 XSD

DTD jezik potiče iz same XML specifikacije, i predstavlja šema jezik relativno ograničenih mogućnosti. Iz tog razloga se razvijaju i drugi jezici za kreiranje XML šema. XML šema jezik ili **XSD** (XML Schema Language) predstavlja XML baziranu alternativu DTD šemi. Preporučena je 2001. godine od strane W3C organizacije. Za XML dokumenat iz listinga 2.11 odgovarala bi XML šema prikazana u listingu 2.14. XSD datoteka nosi ekstenziju ***xsd***.

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.test.com"
xmlns="http://www.test.com"
elementFormDefault="qualified">
      <xs:element name="kupac">
            <xs:complexType>
                  <xs:sequence>
                        <xs:element name="ime" type="xs:string"/>
                        <xs:element name="prezime" type="xs:string"/>
                        <xs:element name="grad" type="xs:string"/>
                        <xs:element name="ulica" type="xs:string"/>
                  </xs:sequence>
            </xs:complexType>
      </xs:element>
</xs:schema>
```

Listing 2.14. XML Schema Language.

Svaka XML šema se sastoji iz sledećih elemenata:

- šema element
- deklaracije elemenata
- deklaracije atributa
- definicije prostih i složenih tipova

Šema element ukazuje na definiciju XML šeme u kojoj se nalaze svi potrebni elementi za kreiranje XML šeme. Zbog toga predstavlja koreni element svake XML šeme. Primer šema elementa *<xs:schema>* dat je u listingu 2.15.

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema>
...
<xs:schema>
```

Listing 2.15. Šema element.

Šema element takođe može sadržati atribute, što se može videti u primeru datom u listingu 2.16.

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema id="AddMobilePhoneType"
      elementFormDefault="qualified"
      xmlns:xs="http://www.w3.org/2001/XMLSchema"
      targetNamespace="http://www.w3schools.com"
      xmlns="http://www.w3schools.com">

...
<xs:schema>
```

Listing 2.16. Šema element sa atributima.

Deklaracije elemenata služe za definisanje prostih i kompleksnih elemenata u XML dokumentima. Prosti elementi mogu sadržati samo tekst, dok kompleksni sadrže druge elemente ili atribute. Primeri deklaracija elemenata su prikazani u listingu 2.17.

```
<xs:element name="Ime" minOccurs ="1" type="xs:string"/>
<xs:element name="Broj" type="xs:integer"/>
<xs:element name="DarumRođenja" type="xs:date"/>
```

Listing 2.17. Deklarisanje elemenata.

Ako je potrebno definisanje atributa u nekom elementu, tada se koriste **deklaracije atributa**. Pošto jednostavni elementi ne mogu imati atribute, element koji sadrži deklaraciju atributa smatraće se kompleksnim. Međutim, sam atribut uvek predstavlja prost tip. Primeri deklaracija atributa dati su u listingu 2.18.

```
<xs:attribute name="NekiAtribut" type="xs:string"/>
```

Listing 2.18. Deklaracija atributa.

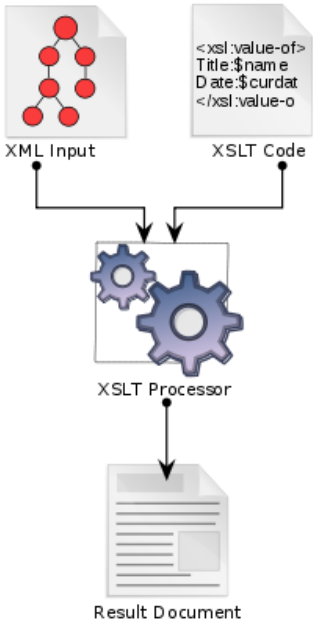
Tipovi elementa se takođe mogu podeliti na proste i složene. Prosti tipovi ne mogu imati podelemente ili atribute, dok složeni tipovi mogu imati podelemente i atribute. Najčešći prosti tipovi koji se definišu u XML šemi su: *xs:string, xs:decimal, xs:integer, xs:boolean, xs:date, xs:time*... Pozivanje tako kreirane šeme iz XML dokumenta može se ostvariti dodavanjem atributa prikazanih u listingu 2.19.

```
<?xml version="1.0" ?>
<kupac
      xmlns="http://www.goranpanic.com"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://www.test.com kupac.xsd">
...
</kupac>
```

Listing 2.19. Povezivanje XSD šeme i XML dokumenta.

2.1.9 XSL transformacije

XSLT (eXtensible Stylesheet Language Transformation) je deklarativni XML baziran jezik koji se koristi za opis pravila transformacija XML dokumenta u drugi XML dokumenat, HTML dokumenat ili tekst. Godine 1999. objavljena je specifikacija XSLT 1.0 razvijena od strane W3C-a. Trenutna verzija je XSLT 2.0, a specifikacija se može pronaći u dokumentu (Kay, 2007). Pri transformaciji originalni dokument se ne menja, već se kreira novi na osnovu sadržaja postojećeg. Nastali dokument može biti serijalizovan od strane procesa u standardnom XML formatu ili nekom drugom formatu. Slika 2.1 prikazuje osnovne elemente XSLT transformacionih procesa i veze koje vladaju među njima.



Slika 2.1. XSLT transformacioni proces.

XSLT stylesheet dokument je XML dokument čiji XML elementi predstavljaju alat za izražavanje instrukcija. Elementi koji se koriste za konstruisanje stylesheet dokumenta definisani su preko XSLT namespace-a, a lokacije se specificiraju preko URL-a: *http://www.w3.org/1999/XSL/Transform*. Stylesheet element *</xsl:stylesheet>* predstavlja koreni element XSLT dokumenta. Stylesheet element sadrži skup pravila koja se deklarišu templejt elementima *</xsl:template>*. Pravila opisuju kako se pojedini elementi u ulaznom XML dokumentu transformišu u rezultujuće elemente u izlaznom dokumentu. Telo templejt pravila se sastoji iz XSLT instrukcija i elemenata koji specificiraju izlazni tekst. Primer karakteristične XSLT transformacije (listing 2.21), sa izgledom ulaznog (listing 2.20) i izlaznog XML dokumenata (listing 2.22) dat je u nastavku.

```
<?xml version="1.0" ?>
<clanovi>
	<korisnik ID="001">
		<ime>Goran</ime>
		<prezime>Panić</prezime>
	</korisnik>
	<korisnik ID="002">
		<ime>Dragan</ime>
		<prezime>Savić</prezime>
	</korisnik>
</clanovi>
```

Listing 2.20. XSLT transformacija - Ulazni XML dokumenat.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet
xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">

	<xsl:output method="xml" indent="yes"/>
	<xsl:template match="/clanovi">
		<users>
```

```
                  <xsl:apply-templates select="korisnik"/>
            </users>
      </xsl:template>
      <xsl:template match="korisnik">
            <user userID="{@ID}">
                  <xsl:value-of select="ime" />
            </user>
      </xsl:template>

</xsl:stylesheet>
```

Listing 2.21. XSLT transformacija - Transformacija.

```
<?xml version="1.0" encoding="UTF-8"?>
<users>
      <user userID="001">Goran</name>
      <user userID="002">Dragan</name>
</users>
```

Listing 2.22. XSLT transformacija - Izlazni dokument.

Templejt pravila se sastoje iz paterna i akcije. **Patern** identifikuje i izdvaja elemente ulaznog XML dokumenta na koje će biti primenjena transformacija. **Akcija** predstavlja kod koji se nalazi u *template* elementu i opisuje transformaciju koja se primenjuje. Atribut *match* upotrebljava se za povezivanje *template* elementa sa nekim delom ulaznog XML dokumenta. Vrednost *match* atributa predstavlja XPath patern. U XSLT dokumentu XPath se koristi za izdvajanje delova XML dokumenta na koje će biti primenjena transformacija.

2.1.10 XPath

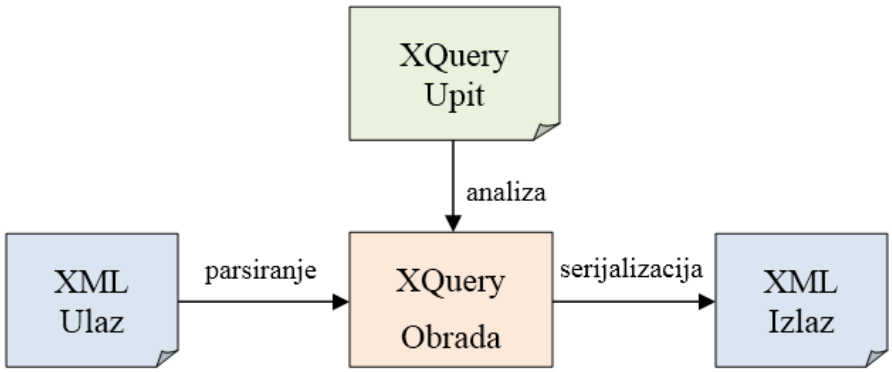
XPath (XML Path Language) je upitni jezik koji omogućava navigaciju do elementa, atributa ili njihovih vrednosti unutar XML dokumenta. Pored toga, XPath se može koristiti za izračunavanje vrednosti niza brojeva ili logičkih vrednosti iz sadržaja XML dokumenta. XPath je definisan od strane W3C organizacije, a trenutno postoje dve verzije XPath v1.0 i XPath v2.0. XML dokument se može posmatrati kao stablo čvorova sa definisanim čvorom koji se naziva koren dokumenta. Koren dokumenta je bezimeni čvor čiji je pod-element koreni element XML dokumenta. Čvor se adresira preko izraza putanje koja predstavlja niz od jednog ili više koraka razdvojenih znakom / . Tabela 2.2 daje pregled nekih XPath operacija i njihovu sintaksu.

IZRAZ	OPIS
nodename	Selektovanje svih pod-elemenata imenovanog elementa
/	Selekcija korenskog elementa
//	Selekcija elementa u odnosu na tekući element
.	Selekcija trenutnog elementa
..	Selekcija roditelja trenutnog elementa
@	Selekcija atributa
*	Poklapanje bilo kog elementa u čvoru
@*	Poklapanje bilo kog atributa u čvoru
node()	Poklapanje čvora bilo koje vrste

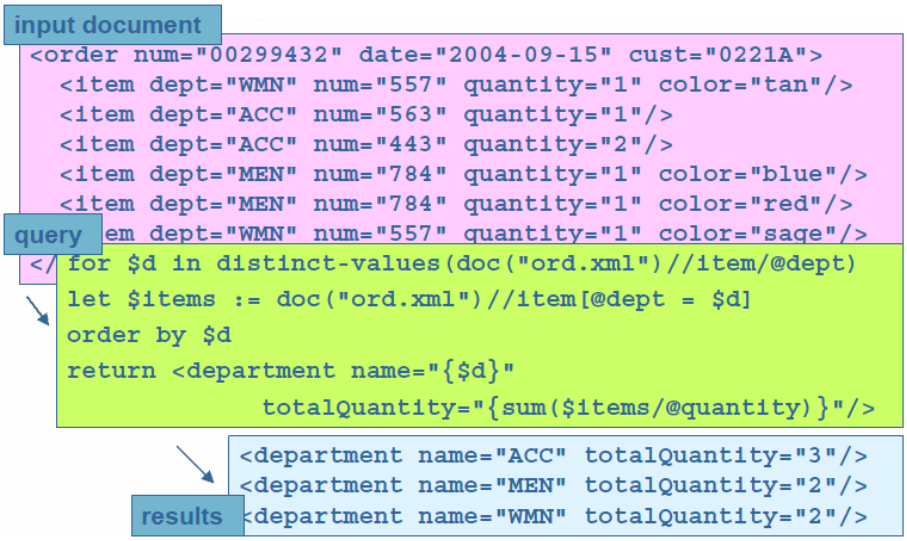
Tabela 2.2. Pregled XPath operacija i njihova sintaksa.

2.1.11 XQuery

XQuery spada u grupu XML tehnologija i predstavlja upitni jezik za pretraživanje XML dokumenta. Razvijen je od strane XQuery radne grupe W3C organizacije, a njegova detaljna specifikacija data je dokumentu (Boag, et al., 2011). Ideja je bila da se kreira upitni jezik koji ima istu širinu funkcionalnosti kao SQL nad relacionim bazama podataka. Projektovan je sa namerom da bude mali, lak za implementaciju i razumljiv upitni jezik. Koristi se za pristupanje i rad sa XML podacima. Pojednostavljen model XQuery procesiranja i primer izvršenja upita prikazani su na slikama 2.2 i 2.3.



Slika 2.2. Pojednostavljen model XQuery procesiranja.



Slika 2.3. Način izvršenja XQuery upita.

XQuery jezik dozvoljava:

- selekciju elemenata i atributa iz ulaznog dokumenta
- spajanje podataka iz više ulaznih dokumenta
- kreiranje i izmenu podataka
- izračunavanje novih podataka
- dodavanje elemenata i atributa u rezultat
- sortiranje rezultata

XQuery je funkcionalni jezik u kome je svaki upit iskaz. Svi iskazi se dele u nekoliko grupa:

Iskazi putanje predstavljaju nad-skup putanja u XPath-u. Primer je dat u listingu 2.23 i odgovara pitanju "U XML dokumentu *studenti.xml* izlistati indekse studenata prve i druge godine".

```
document("studenti.xml")//godina[1 TO 2]//indeks
```

Listing 2.23. Iskaz putanje.

Konstruktori elemenata služe za generisanje elemenata direktno u upitu. Primer je dat u listingu 2.24 i odgovara pitanju "Generisati elemente <*student*> koji imaju atribute sa imenom *id*. Vrednost atributa i sadržaj elementa su specifikovani promenljivom *$id* ."

```
<student id= {$id}>
      {$ime}
      {$prezime}
      {$smer}
      {$godina}
      {$indeks}
</student>
```

Listing 2.24. Konstruktor elementa.

FLWR iskazi predstavljaju iskaze koji se sastoje od **F**OR, **L**ET, **W**HERE i **R**ETURN izraza. Primer je dat u listingu 2.25 i odgovara pitanju "Izlistati sve smerove koji imaju više od 50 studenata."

```
<studenti>
{
      FOR $p IN distinct(document("studenti.xml")//smer)
      LET $b := document("studenti.xml")//student[smer= $p]
      WHERE count($b) > 50
      RETURN $p
}
</studenti>
```

Listing 2.25. FLWR iskaz.

Uslovni iskazi proveravaju iskaze i vraćaju jedan od dva rezultujuća iskaza. Ukoliko je vrednost iskaza tačno, vraća se vrednost prvog rezultujućeg iskaza, u suprotnom, vraća se vrednost drugog. Primer je dat u listingu 2.26 i odgovara pitanju "Izlistati studente sortirane po indeksu, sa tim da se za studente informatike prikaže ime, a za ostale prezime".

```
FOR $k IN //student
RETURN
      <studenti>
            {$k/indeks, IF ($k[@smer= "informatika"])THEN
             $k/ime ELSE $k/prezime}
      </studenti>
      SORTBY (indeks)
```

Listing 2.26. Uslovni iskaz.

Kvantifikovani iskazi su oni izrazi koji sadrže klauzule *SOME* i *EVERY*. Primer upotrebe klauzule *SOME* dat je u listingu 2.27 i odgovara pitanju "Pronaći indekse studenata koji u nazivu smera sadrži reč *biologija* ". Na identičan način se koristi i *EVERY* klauzula.

```
FOR $k IN //student
      WHERE SOME $p IN $b//smer SATISFIES
      contains($p, "biologija)
      RETURN $k/indeks
```

Listing 2.27. Kvalifikovani iskaz tipa SOME.

2.1.12 XML Namespace

XML Namespace omogućava da neki dokumenti koji sadrže XML elemente i atribute budu uzeti iz različitih rečnika bez imenovanja, time se eliminiše konfuzija prilikom kombinovanja više šema u jednom XML dokumentu. Kada se u XML-u koriste *namespace* prefiksi, svaki od njih mora biti definisan. Definisanje se vrši u XML atributima u početnom elementu, a sintaksa ima sledeći oblik: *xmlns:prefix="URI"*. Primer upotrebe *namespace* elementa u XML dokumentu, dat je u listingu 2.28, a više o njemu može se pronaći u dokumentu (Bray, et al., 2009).

```
<?xml version="1.0" encoding="unicode" ?>
<root
    xmlns:a="http://www.w3.org/TR/html4/"
    xmlns:b="http://www.w3schools.com/furniture">
    <a:table>
        <a:tr>
            <a:td>Ime</a:td>
            <a:td>Prezime</a:td>
        </a:tr>
    </a:table>
    <b:table>
        <b:width>50</b:width>
        <b:length>100</b:length>
    </b:table>
</root>
```

Listing 2.28. Definisanje namespace elemenata u root XML elementu.

2.2 Fazi skupovi i logika

Kao izvor informacija iz oblasti fazi skupova i fazi logike korišćene su knjige (Lin & Lee, 1996), (Tanaka & Nimura, 1996) i (Kril & Yuan, 1995). U našem jeziku reč fuzzy se ponekad prevodi kao rasplinut, ali je termin fazi češće u upotrebi i biće korišćen u ovom radu.

Svet oko nas često sadrži informacije o sistemu koje se ne mogu apsolutno precizno reprezentovati kao tačno ili pogrešno. Tačnije, mnogo je više situacija u kojima koristimo neprecizne konstatacije, kao na primer: skup sat, lepa devojka ili dobra ocena. Upotreba klasične (binarne) logike, podrazumeva rad sa informacijama koje su ili potpuno tačne ili potpuno pogrešne. Nepreciznim ili nekompletnim informacijama nije moguće upravljati pomoću klasične logike. Da bismo bili u mogućnosti reprezentovati informacije iz ovakvih sistema morali bismo proširiti binarnu logiku u fazi logiku.

Fazi (rasplinuta) logika predstavlja prirodniji koncept u odnosu na klasičnu logiku, te je njena upotreba u realnim sistemima ponekad jednostavnija nego upotreba klasičnih metoda. To što se nepreciznim izrazima predstavljaju informacije, ne znači da će i zaključci biti neprecizniji. Čest je slučaj da se upotrebom fazi reprezentacije informacija dođe do preciznijeg zaključka.

Pojam **fazi (rasplinutih) skupova** prvi put je definisan još 1965. godine u radu (Zadeh, 1965). Fazi logika je 1974. godine eksperimentalno primenjena u vođenju i upravljanju parnom mašinom (Ebrahim Mamdani), a 1980. javljaju se prvi slučajevi komercijalne primene u vođenju cementne peći (F.H.Smidth) i vođenju procesa prečišćavanja vode (Fuji Electric). Od tada su fazi sistemi u velikoj meri postali zamena konvencionalnim tehnologijama u velikom broju naučnih aplikacija i inženjerskih sistema, naročito u oblasti upravljanja sistemima i prepoznavanju oblika. Uređaji kao što su klime, usisivači, navigacije, mašine za pranje veša i aparati za merenje pritiska dovoljan su dokaz velike rasprostranjenosti i upotrebljivosti ove tehnike. U informacionim tehnologijama i ekspertskim sistemima, fazi logika se koristi kao podrška odlučivanju.

Teorija fazi logike neodoljivo podseća na teoriju verovatnoće, međutim iako su ove dve teorije veoma slične, među njima postoje suštinske razlike. Osnovna razlika je što fazi logika operiše sa determinističkim nedorečenostima i neodređenostima, dok se verovatnoća bavi verodostojnošću stohastičkih događaja i iza nje suštinski stoji eksperiment. Fazi logika pokriva subjektivnost ljudskog mišljenja, osećanja, jezika, dok verovatnoća pokriva objektivnu statistiku u prirodnim naukama. Takođe njihovi modeli nose drugačiji vid informacija. Fazi funkcija pripadnosti predstavlja sličnost objekata u kontekstu neprecizne definicije osobina, dok verovatnoća daje informaciju o frekvenciji ponavljanja.

2.2.1 Fazi skupovi

Teorija fazi skupova predstavlja pogodni matematički aparat za izradu modela različitih procesa u kojima dominira neizvesnost, višeznačnost, subjektivnost, neodređenost itd.

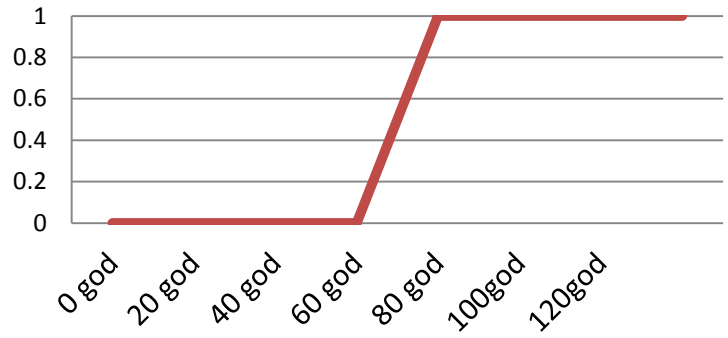
Pojam **fazi skupa** definisao je Zadeh. Tako, fazi skupovi predstavljaju osnovne elemente kojima se može opisati nepreciznost.

Element u nekom diskretnom skupu može da ima dva stanja, pripada ili ne pripada određenom skupu. Matematički definisano stepen pripadnosti skupu je 1 ako pripada i 0 ako ne pripada. Kod fazi skupa elementi pored dva navedena stanja poseduju i treće, tačnije mogu i delimično da pripadaju. Matematički definisano stepen pripadnosti elementa nekom fazi skupu kreće se od nepripadanja ili 0, preko delimičnog pripadanja ili nekog broja iz intervala (0,1), sve do apsolutnog pripadanja ili vrednosti 1.

2.2.2 Funkcija pripadnosti

Za opisivanje pripadnosti nekog elementa fazi skupu A, koristi se **fazi funkcija pripadnosti** μ_A, čiji je kodomen realni interval [0,1] umesto klasične karakteristične funkcije koja odgovara klasičnom skupu i čiji je kodomen skup $\{0,1\}$. Budući da se stepen pripadnosti elementa fazi skupu može okarakterisati brojem iz intervala, fazi skup predstavlja generalizaciju klasičnog skupa.

Imamo da, funkcija pripadnosti fazi skupa preslikava svaki element univerzalnog skupa u pomenuti interval realnih brojeva. Na slici 2.4 dat je primer funkcije pripadnosti skupu starih ljudi, koja opisuje stepen pripadnosti svakog elementa, u ovom slučaju čoveka, tom fazi skupu.



Slika 2.4. Funkcija pripadnosti skupu starih ljudi.

2.2.3 Osnovne definicije

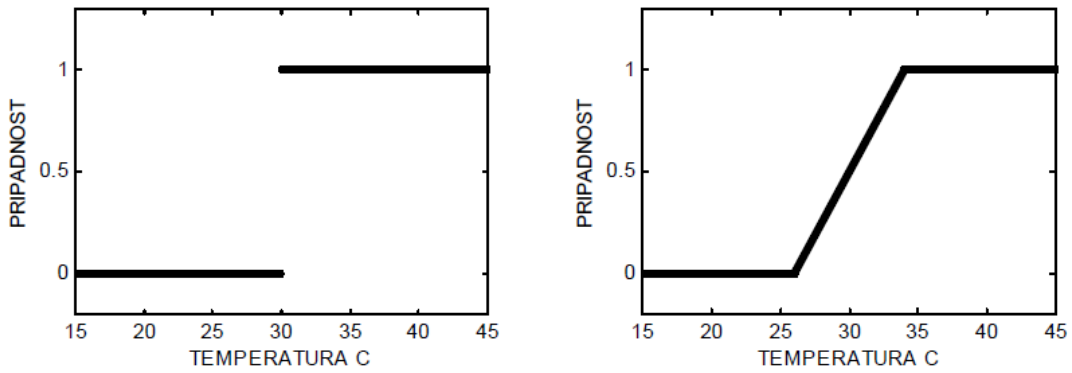
Neke od osnovnih definicija koje se javljaju u teoriji fazi skupova, biće navedene u nastavku.

Definicija 2.1 Fazi skup A iz skupa U može se definisati kao skup uređenih parova $(x, \mu_A(x))$,

$$A = \{(x, \mu_A(x)) \mid x \in U\}, \qquad \mu_A(x) \in [0,1]\ ,$$

gde je U skup pozitivnih realnih brojeva, a μ_A funkcija pripadnosti (ili karakteristična funkcija) skupa A i pri tom $\mu_A(x)$ predstavlja stepen pripadanja elementa *x* fazi skupu A.

Primeri funkcija pripadnosti klasičnog i fazi skupa dati su na slici 2.5.



Slika 2.5. Funkcija pripadnosti a) klasičan skup i b)fazi skup.

Definicija 2.2 Dva fazi skupa A i B su jednaka, u oznaci A = B ako i samo ako važi:

$$\forall x \in U, \mu_A(x) = \mu_B(x)$$

Definicija 2.3 Za fazi skup A kažemo da je poskup fazi skupa B, u oznaci A⊆B ako i samo ako važi:

$$\forall x \in U, \mu_A(x) \leq \mu_B(x)$$

Definicija 2.4 Fazi singlton je fazi skup sa samo jednim pripadajućim elementom čiji je stepen pripadnosti.

$$\mu_A(x) = 1$$

Definicija 2.5 Tačka prolaska je uređen par

$$\{(x, \mu_A(x)) \mid x \in U\}, \text{ i važi } \mu_A(x) = 0.5 ,$$

a može postojati više tačaka prolaska u fazi skupu.

Definicija 2.6 Nosač ili podrška fazi skupa A je skup:

$$Supp(A) = \{x \in U \mid \mu_A(x) > 0\}$$

Definicija 2.7 Jezgro fazi skupa A je skup:

$$ker(A) = \{x \in U \mid \mu_A(x) = 1\}$$

Definicija 2.8 Visina fazi skupa A je broj:

$$height(A) = \sup_{x \in U} \mu_A(x)$$

Definicija 2.9 Za fazi skup A kažemo da je normalizovan ako i samo ako važi

$$\exists x \in U, \mu_A(x) = height(A) = 1 ,$$

u suprotnom je skup sub-normalizovan.

Definicija 2.10 α-preseci (α-odsečci ili α-nivoi) fazi skupa A, u oznaci $[A]_\alpha$ za svako $\alpha \in [0,1]$, definišu se na sledeći način:

$$[A]_\alpha = \{x \in U \mid \mu_A(x) \geq \alpha\},$$

dok se **strogi** α – preseci mogu definisati jednačinom:

$$[A]_\alpha = \{x \in U \mid \mu_A(x) > \alpha\}$$

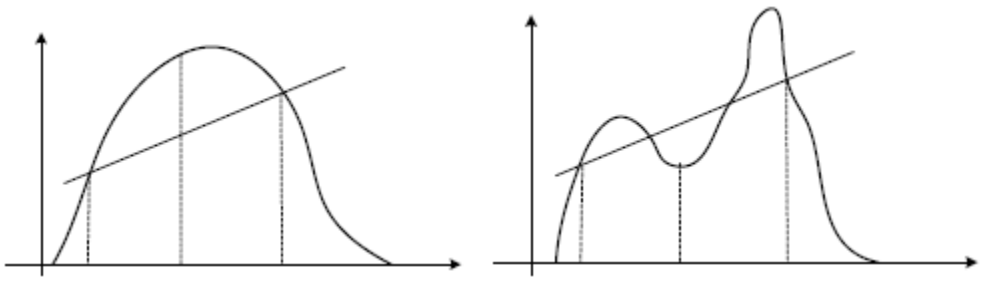
α - preseci predstavljaju obične podskupove skupa realnih brojeva U.

Definicija 2.11 Fazi skup A se naziva konveksnim ako i samo ako su za svako $\alpha \in [0,1]$, α-odsečci konveksni podskupovi od U. Ekvivalentno, ako je U skup realnih brojeva, fazi skup A je konveksan ako važi:

$$\mu_A(\lambda x_1 + (1-\lambda)x_2) \geq min(\mu_A(x_1), \mu_A(x_2)),$$

gde je λ proizvoljan parametar između 0 i 1.

Interpretacija definicije 2.11: Uzmemo dva proizvoljna elementa x_1 i x_2 iz skupa realnih brojeva U i posmatramo duž od jedne do druge tačke. Fazi skup je konveksan ako funkcija pripadnosti za svaku tačku te duži mora biti veća ili jednaka od minimuma vrednosti $\mu_A(x_1)$ i $\mu_A(x_2)$, kao na slici 2.6.



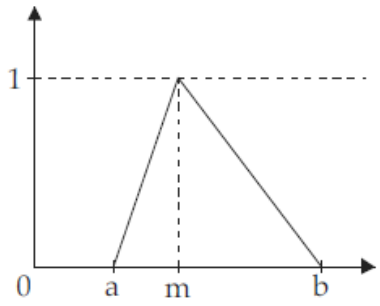
Slika 2.6. Interpretacija konveksnosti fazi skupa: a) konveksan i b) nekonveksan.

Definicija 2.12 Fazi broj je fazi skup A koji je normalizovan, konveksan i ima ograničeno jezgro. Za skup kažemo da ima ograničeno jezgro kada je jezgro kao podskup skupa realnih brojeva U ograničen skup.

2.2.4 Najčešći tipovi funkcija pripadnosti

Prema Zadeh-u postoje dve grupe ovakvih funkcija: linearne i nelinearne. Najčešće korišćeni tipovi funkcija pripadnosti prikazani su u nastavku.

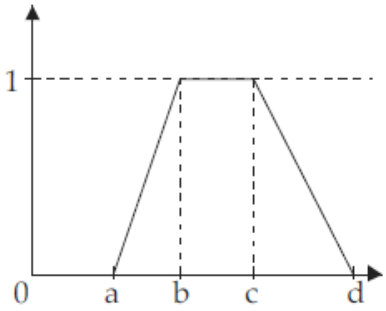
1. **Trougaona** funkcija pripadnosti je prikazana na slici 2.7.



$$\mu_A(x) = \begin{cases} 0, x \leq a, x \geq b \\ \dfrac{(x-a)}{(m-a)}, x\epsilon(a,m) \\ \dfrac{(b-x)}{(b-m)}, x\epsilon[m,b) \end{cases}$$

Slika 2.7. Trougaona funkcija pripadnosti.

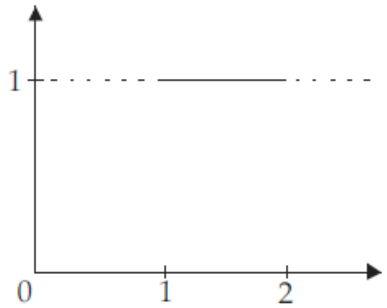
2. **Trapezoidna** funkcija pripadnosti je prikazana na slici 2.8.



$$\mu_A(x) = \begin{cases} 0, x \leq a, x \geq d \\ \dfrac{(x-a)}{(b-a)}, x\epsilon(a,b) \\ 1, x\epsilon[b,c] \\ \dfrac{(d-x)}{(d-c)}, x\epsilon(c,d) \end{cases}$$

Slika 2.8. Trapezoidna funkcija pripadnosti.

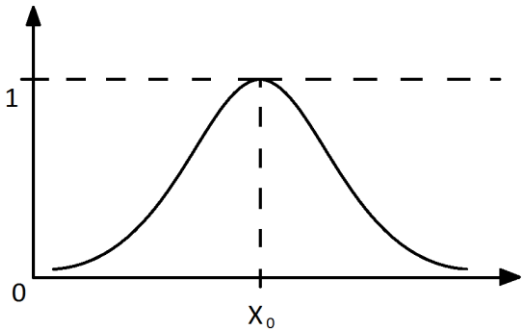
3. **Interval** se takođe može predstaviti kao funkcija pripadnosti. Na slici 2.9 je prikazan interval [1,2] kao funkcija pripadnosti.



$$\mu_A(x) = \begin{cases} 0, x \leq a, x \geq d \\ 0, x\epsilon(a,b) \\ 1, x\epsilon[b,c] \\ 0, x\epsilon(c,d) \end{cases}$$

Slika 2.9. Interval kao funkcija pripadnosti.

4. **Gausova** funkcija pripadnosti je prikazana na slici 2.10.



$$\mu_A(x) = e^{\frac{-(x-x_0)^2}{d}}$$

Slika 2.10. Gausova funkcija pripadnosti.

2.2.5 Fazi logika

Prva proučavanja logike koja ima više od dve nepoznate započeo je 1920. godine Lukašijevič, uvođenjem istinitosne vrednosti "delimično tačno" u klasičnu logiku. Sam koncept fazi logike definiše se u sklopu teorije fazi skupova. Na početku svog razvoja smatralo se da fazi logika ima potencijala za primenu u društvenim naukama. Međutim, punu afirmaciju fazi logika doživljava u područjima tehničke primene, poput automatske regulacije, analize podataka, upravljanja sistemima itd.

Nemački matematičar G. Cantor (1845.-1918.) smatra se utemeljivačem klasične teorije skupova. Po toj teoriji pripadnost objekta klasičnom skupu određuje se sa dva eksplicitna stanja: objekat pripada skupu (1) i objekat ne pripada skupu (0). Tako da skup istinitosnih vrednosti klasične logike sadrži elemente $\{0,1\}$. Primenom fazifikacije nad klasičnim skupom, dobijamo skup čije se istinitosne vrednosti proširuju na interval $[0,1]$. Ako se konjunkcija zameni t-normom kao u definiciji 2.14, disjunkcija t-konormom kao u definiciji 2.15, a negacija definiše po definiciji 2.13, dobijaju se osnovne logičke operacije za rad sa fazi logikom.

2.2.6 Osnovne operacije nad fazi skupovima

Definisanje operacija negacije, trougaonih normi i konormi, pružiće mogućnost definisanja osnovnih operacija nad fazi skupovima.

Definicija 2.13 Uopštenje klasične negacije.

- Nerastuća funkcija je negacija ako važi: $N(0) = 1$ i $N(1) = 0$.
- Negacija $N: [0,1] \to [0,1]$ je stroga negacija ako je neprekidna i strogo opadajuća.
- Stroga negacija $N: [0,1] \to [0,1]$ je jaka negacija ako je involucija, odnosno ako važi $N \circ N = id_{[0,1]}$.

Najčešće korišćena negacija je standardna negacija $N(x) = 1 - x$.

Definicija 2.14: Funkcija $T: [0,1] \times [0,1] \to [0,1]$ je trougaona norma ili t-norma, ukoliko zadovoljava sledeće osobine:

- Komutativnost: $T(x, y) = T(y, x)$

- Asocijativnost: $T\big(x,T(y,z)\big)=T(T(x,y),z)$
- Monotonost: $(x\le z)\ \Rightarrow\ T(x,y)\le T(z,y)$
- Granični uslovi: $T(x,1)=x$ i $T(x,0)=0$

Četiri osnovne t-norme su:

- $T_M(x,y)=min(x,y)$
- $T_P(x,y)=x*y$ - predstavlja obično množenje
- $T_L(x,y)=max(x+y-1,0)$
- $T_D(x,y)=\begin{cases}0,(x,y)\in[0,1)^2\\ \min(x,y),inače\end{cases}$

Upotrebom osobine asocijativnosti osnovne t-norme možemo proširiti na n-arne operatore koristeći indukciju, čime se dobijaju sledeće t-norme:

- $T_M(x_1,\dots,x_n)=min(x_1,\dots,x_n)$
- $T_P(x_1,\dots,x_n)=x_1*\dots*x_n$
- $T_L(x_1,\dots,x_n)=max(\sum_{i=1}^{n}x_i-(n-1),0)$
- $T_D(x_1,\dots,x_n)=\begin{cases}x_i,ako\ \forall j\neq i,x_j=1\\ 0,inače\end{cases}$

Može se pokazati da važi $T_D\le T_L\le T_P\le T_M$ i $T_D\le T\le T_M$ za svaku t-normu T.

Definicija 2.15: Funkcija $S\colon[0,1]\times[0,1]\to[0,1]$ je t-konorma (trougaona konorma) ako zadovoljava sledeće osobine:

- Komutativnost: $S(x,y)=S(y,x)$
- Asocijativnost: $S\big(x,S(y,z)\big)=S(S(x,y),z)$
- Monotonost: $(x\le z)\ \Rightarrow\ S(x,y)\le S(z,y)$
- Granični uslovi: $S(x,1)=1$ i $S(x,0)=x$

Negacija se koristi za definisanje komplementa fazi skupa kao i definisanje dualne t-konorme ako je data t-norma. Upotrebnom stroge negacije N za svaku t-normu T, može se dobiti njena dualna t-konorma S, što je dato sledećom jednačinom:

$$S(x,y)=N\Big(T\big(N(x),N(y)\big)\Big).$$

Ako se uzme da je N standardna negacija $N(x)=1-x$, tada jednačina glasi:

$$S(x,y)=1-\ T(1-x,1-y).$$

Isto važi i obratno, za svaku t-konormu S upotrebom stroge negacije N dobija se t-norma T, što je dato sledećom jednačinom:

$$T(x,y)=N\Big(S\big(N(x),N(y)\big)\Big),$$

ako je N standardna negacija, sledi da je:

$$T(x,y)=1-\ S(1-x,1-y).$$

Korišćenjem ovih transformacija mogu se dobiti osnovne t-konorme:

- $s_M(x,y) = max(x,y)$
- $s_P(x,y) = x + y - xy$
- $s_L(x,y) = min(x+y, 1)$
- $s_D(x,y) = \begin{cases} 1, (x,y) \in (0,1]^2 \\ \max(x,y), inače \end{cases}$

Slično t-normama, osnovne t-konorme se mogu proširiti na n-arne operatore na sledeći način:

- $s_M(x_1, \dots, x_n) = max(x_1, \dots, x_n)$
- $s_P(x_1, \dots, x_n) = 1 - \prod_{i=1}^{n}(1 - x_i)$
- $s_L(x_1, \dots, x_n) = min(\sum_{i=1}^{n} x_i, 1)$
- $s_D(x_1, \dots, x_n) = \begin{cases} x_i, ako\ \forall j \neq i, x_j = 0 \\ 1, inače \end{cases}$

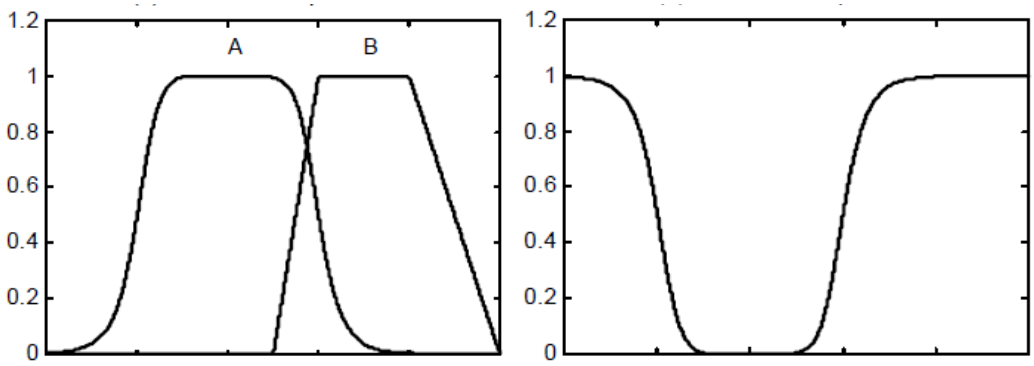
Prethodno navedene definicije negacije (definicija 2.13), t-normi (definicija 2.14) i t-konormi (definicija 2.15) omogućavaju uvođenje elementarnih operacija nad fazi skupovima kao što su komplement, unija i presek, što će biti opisano u nastavku.

Neka su date t-norma T, t-konorma S, negacija N, fazi skupovi A i B i njihove karakteristične funkcije $\mu_A(x)$ i $\mu_B(x)$. Skupovi A∪B , A∩B i C_N(A) definišu se preko njihovih karakterističnih funkcija, što je dato sledećim definicijama.

Definicija 2.16 Funkcija pripadnosti za komplement fazi skupa A glasi:

$$\mu_{C_N(A)}(x) = N(\mu_A(x))$$

Primer fazi funkcije pripadnosti $\mu_{C_N(A)}(x)$ komplementa fazi skupa A, koji se najčešće definiše kao standardna negacija $N(x) = 1 - x$, dat je na slici 2.11.

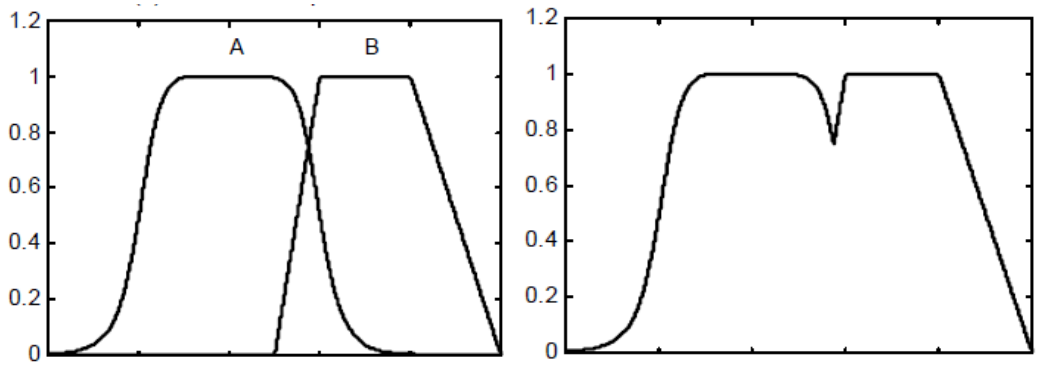


Slika 2.11. Komplement a) početni skupovi i b) komplement fazi skupa A.

Definicija 2.17 Funkcija pripadnosti za uniju dva fazi skupa A i B je data sa:

$$\mu_{A\cup B}(x) = S\big(\mu_{\mathrm{A}}(x), \mu_{\mathrm{B}}(x)\big)\,,$$

gde je S proizvoljna trougaona konorma. Primer funkcije pripadnosti za uniju dva fazi skupa najčešće korišćene T-norme maksimum dat je na slici 2.12.

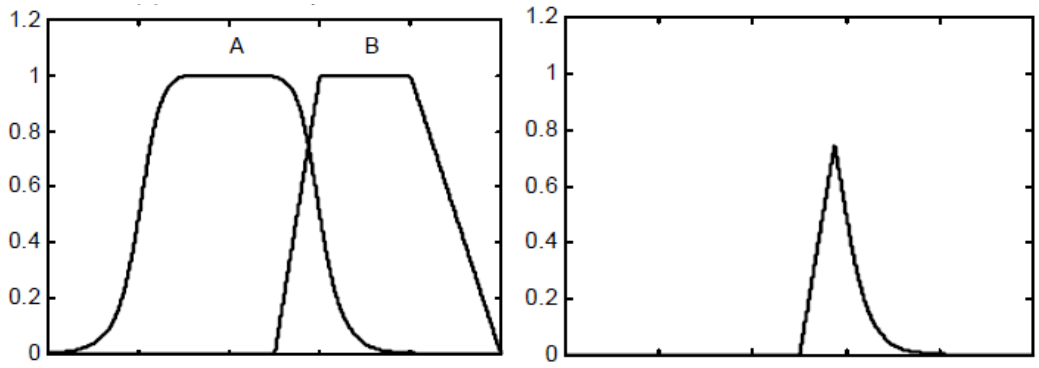


Slika 2.12. Unija a) početni fazi skupovi i b) unija dva fazi skupa.

Definicija 2.18 Funkcija pripadnosti za presek dva fazi skupa A i B je data sa:

$$\mu_{A\cap B}(x) = T\big(\mu_{\mathrm{A}}(x), \mu_{\mathrm{B}}(x)\big)\,,$$

gde je t proizvoljna trougaona norma. Primer funkcije pripadnosti za presek dva fazi skupa najčešće korišćene T-norme minimum dat je na slici 2.13.



Slika 2.13. Presek a) početni fazi skupovi i b) presek dva fazi skupa.

2.3 Generalizovani problemi zadovoljenja fazi ograničenja sa prioritetima

Pronalaženje rešenja koje zadovoljava sva ograničenja za optimalno vreme (Dubois & Fortemps, 1999) rešava se primenom problema zadovoljenja ograničenja ili CSP-a (Constraint Satisfaction Problem). Ako zadovoljenje ograničenja ima vrednost iz intervala [0,1], očigledno je da se u CSP može uvesti fazi logika. Ograničenja se mogu modelirati kao fazi skupovi, te se u tom slučaju dobijaju fazi problemi zadovoljenja ograničenja ili FCSP (Fuzzy Constraint Satisfaction Problem). Mera zadovoljenja svih ograničenja se dobija agregacijom vrednosti zadovoljenja pojedinačnih ograničenja, koja se vrši pomoću operatora

fazi logike. Radovi (Dubois, et al., 1996), (Schiex, 1992), (Ruttkay, 1994), (Meseguer & Larrosa, 1997) i (Bistarelli, et al., 1999) mogu poslužiti za bolje razumevanje problema CSP-a i FCSP-a.

2.3.1 CSP i FCSP

Neka posmatramo CSP kao skup domena i ograničenja nad tim domenima koja su n-arne relacije nad Dekartovim proizvodom tih domena. Definicija CSP-a izgledala bi ovako:

Definicija 2.19: Problem zadovoljenja ograničenja je uređena trojka (X, D, C), gde je X konačan skup promenljivih, D konačan skup domena, a C skup ograničenja, i važi:

1. Skup $X = \{x_i \mid i = 1, 2, \dots, n\}$ je konačan skup promenljivih.

2. Skup $D = \{D_i \mid i = 1, 2, \dots, n\}$ je konačan skup domena. Svaki domen D_i je skup elemenata koje može uzimati promenljiva x_i iz skupa X.

3. Skup ograničenja $C = \{R_i \mid i = 1, 2, \dots, m\}$ je relacija R_i, gde je svako R_i relacija nad direktnim proizvodom domena.

$$R_i : \left(\prod_{x_j \in var(R_i)} D_j \right) \rightarrow \{0,1\},$$

gde $var(R_i)$ predstavlja skup promenljivih u navedenim ograničenjima R_i.

Valuacijom promenljive x se naziva dodela vrednosti promenljivoj iz odgovarajućeg domena i označava se sa v_x. Ukoliko skup X sadrži n promenljivih $(x_1, x_2, \dots, x_n)$ tada se valuacija svih promenljivih u skupu naziva kombinovanom valuacijom i označavamo je sa $v_x = (v_{x_1}, v_{x_2}, \dots, v_{x_n})$.

Vrednost karakteristične funkcije R_i se može izračunati za svako ograničenje iz kombinovane valuacije, koje može biti potpuno zadovoljeno ili potpuno nezadovoljeno u slučaju CSP-a.

Proširenje CSP-a fazi ograničenjima pruža nove mogućnosti njihove primene (Takači, 2005). Na taj način dobijena je definicija FCSP problema zadovoljenja ograničenja proširenog upotrebom fazi skupova i fazi logike. Sam pojam fazi CSP prvi put je definisao Rosenfeld u radu (Rosenfeld, et al., 1976).

Definicija 2.20: Problem zadovoljenja fazi ograničenja je uređena trojka (X, D, C^f), takva da je:

1. Skup $X = \{x_i \mid i = 1, 2, \dots, n\}$ je konačan skup promenljivih.

2. Skup $D = \{D_i \mid i = 1, 2, \dots, n\}$ je konačan skup domena. Svaki domen D_i je skup elemenata koje može uzimati promenljiva x_i iz skupa X.

3. Fazi skup ograničenja C^f predstavljen je fazi relacijama čije se funkcije pripadnosti definišu na sledeći način:

$$\mu_{R_i^f} : \left(\prod_{x_j \in var\left(R_i^f\right)} D_j \right) \rightarrow [0,1] ,$$

gde $\mathrm{var}(R_i^f)$ predstavlja skup promenljivih u navedenim ograničenjima R_i^f.

Kod klasičnog CSP-a ograničenja su funkcije koje preslikavaju Dekartov proizvod jednog ili više domena iz D u skup {0,1}. Za razliku od CSP-a u FCSP-u je skup {0,1} zamenjen zatvorenim intervalom [0,1]. Na taj način je binarna logika zamenjena fazi logikom. Dobijene su funkcije koje preslikavaju Dekartov proizvod jednog ili više domena iz D u jedinični interval, a takve funkcije se mogu posmatrati kao karakteristične funkcije fazi skupova. To znači da su ograničenja u FCSP-u fazi skupovi, a mera zadovoljenja ograničenja je vrednost karakteristične funkcije fazi skupa koji opisuje to ograničenje.

Fazi logiku koristimo kada želimo da dobijemo globalnu meru zadovoljenja ograničenja. Ukoliko je potrebno da sva ograničenja budu zadovoljena, tada se za operator agregacije (u oznaci $\oplus$) koristi t-norma. U slučaju da je potrebno da najmanje jedno ograničenje bude zadovoljeno, koristi se t-konorma.

2.3.2 PFCSP

Ukoliko su neka ograničenja važnija od drugih, kažemo da nemaju isti prioritet. Ako FCSP proširimo prioritetima dobija se PFCSP (Prioritised Fuzzy Constraint Satisfaction Problem). Tako se pored stepena zadovoljenja ograničenja može dodati i prioritet, za svako ograničenje. Od važnosti ograničenja zavisiće njegov uticaj na krajnji rezultat. U radu (Luo, et al., 2003) data je definicija PFCSP-a, koja se u celosti navodi u nastavku teksta.

Definicija 2.21: PFCSP je uređena četvorka (X, D, C^f, ρ), gde je (X, D, C^f) FCSP, a ρ je funkcija prioriteta $\rho: C^f \rightarrow (0, \infty]$.

Definicija 2.22: Neka je dat PFCSP (X, D, C^f, ρ), i kombinovana valuacija svih promenljivih iz X, ϑ_x. Tada je α_ρ, dato sa

$$\alpha_\rho(\vartheta_x) = \oplus_\rho \left\{ g\left(\rho(R^f), \mu_{R^f}\left(\vartheta_{var(R^f)}\right)\right) \mid R^f \in C^f \right\},$$

gde je $\oplus_\rho: [0,1]^n \rightarrow [0,1]$, a $g: [0,\infty) \times [0,1] \rightarrow [0,1]$, naziva globalnim stepenom zadovoljenja ako se zadovoljene sledeće osobine - aksiome:

1. Ako za fazi ograničenje R_{max}^f važi

$$\rho_{max} = \rho_{max}\left(R_{max}^f\right) = \max\{\rho(R^f \mid R^f \in C^f)\},$$

tada važi i

$$\mu_{R_{max}^f}\left(\vartheta_{var\left(R_{max}^f\right)}\right) = 0 \;\; \text{sledi} \;\; \alpha_\rho(\vartheta_X) = 0 .$$

2. Ako je $\exists \rho_0 \in [0,1]$ takvo da $\forall R^f \in C^f$ važi da je $\rho(R^f) = \rho_0$, onda je globalni stepen zadovoljenja ograničenja dat sa:

$$\alpha_\rho(\vartheta_X) = \bigoplus_\rho \left\{ \mu_{R^f}\left(\vartheta_{var(R^f)}\right) \mid R^f \in C^f \right\},$$

gde je $\bigoplus_\rho$ t-norma.

3. Pretpostavimo da za $R_i^f,\ R_j^f \in C^f$ važi $\rho\left(R_i^f\right) \geq \rho\left(R_j^f\right)$, neka je dato $\delta > 0$ i neka su date dve kombinovane valuacije ϑ_X i $\vartheta_X{}'$, takve da za $\forall R^f \in C^f$ važi:

a) ako $R^f \neq R_i^f$ i $R^f \neq R_j^f$, onda $\mu_{R^f}\left(\vartheta_{var(R^f)}\right) = \mu_{R^f}\left(\vartheta'_{var(R^f)}\right)$

b) ako $R^f = R_i^f$, onda $\mu_{R^f}\left(\vartheta_{var(R^f)}\right) = \mu_{R^f}\left(\vartheta'_{var(R^f)}\right) + \delta$

c) ako $R^f = R_j^f$, onda $\mu_{R^f}\left(\vartheta_{var(R^f)}\right) = \mu_{R^f}\left(\vartheta'_{var(R^f)}\right) + \delta$

Tada, ako važi:

$$g\left(\rho\left(R_i^f\right),\ \mu_{R_i^f}\left(\vartheta_{var\left(R_i^f\right)}\right)\right) \leq g\left(\rho\left(R_j^f\right),\ \mu_{R_j^f}\left(\vartheta_{var\left(R_j^f\right)}\right)\right),$$

onda sledi: $\alpha_\rho(\vartheta_X) \geq \alpha_\rho(\vartheta'_X)$.

4. Neka su date dve različite kombinacije valuacije ϑ_X i ϑ'_X sa osobinom da $\forall R^f \in C^f$ važi:

$$\mu_{R^f}\left(\vartheta_{var(R^f)}\right) \geq \mu_{R^f}\left(\vartheta'_{var(R^f)}\right)$$

Tada je $\alpha_\rho(\vartheta_X) \geq \alpha_\rho(\vartheta'_X)$.

5. Ako postoji kombinovana valuacija ϑ_X takva da $\forall R^f \in C^f$ važi $\mu_{R^f}\left(\vartheta_{var(R^f)}\right) = 1$, onda sledi $\alpha_\rho(\vartheta_X) = 1$.

U prethodnoj definiciji se može videti funkcija prioriteta ρ, kod koje u slučaju nekog ograničenja R, veća vrednost $\rho(R)$ predstavlja veći prioritet za to ograničenje. Za agregiranje prioriteta svakog ograničenja sa vrednošću tog ograničenja služi funkcija g, pri čemu se tako agregirane vrednosti uz pomoć operatora agregacije $\bigoplus_\rho$ agregiraju u globalni stepen zadovoljenja ograničenja.

Date aksiome precizno definišu značenje prioriteta ograničenja i njegovo ponašanje. Ukoliko je stepen zadovoljenja 0, kod ograničenja sa maksimalnim prioritetom, tada će istu vrednost imati i globalni stepen zadovoljenja. Kada su svi prioriteti jednaki tada PFCSP sistem prelazi u FCSP. Ako jedno ograničenje ima veći prioritet, onda će povećanjem stepena zadovoljenja tog ograničenja, povećanje globalnog stepena zadovoljenja biti veće nego u slučaju kada bi se za istu vrednost povećalo neko drugo ograničenje sa manjim prioritetom.

Takači u svojoj doktorskoj disertaciji (Takači, 2006) detaljno predstavlja strožiji koncept prioriteta koji je usvojen i upotrebljen i u ovoj disertaciji. GPFCSP sistemi čija definicija sledi u sledećoj sekciji, se zasnivaju na razvijenom strožijem konceptu, koji se dobija eliminacijom uslova iz treće aksiome u definiciji PFCSP-a.

$$g\left(\rho\left(R_i^f\right),\ \mu_{R_i^f}\left(\vartheta_{var\left(R_i^f\right)}\right)\right) \leq g\left(\rho\left(R_j^f\right),\ \mu_{R_j^f}\left(\vartheta_{var\left(R_j^f\right)}\right)\right)$$

2.3.3 GPFCSP

PFCSP sistemi dozvoljavaju jedino upotrebu konjunkcije, odnosno t-norme nad ograničenjima, što znači da bi uslovi ovog tipa dozvoljavali samo upotrebu logičkog operatora AND. Da bi se omogućila upotreba disjunkcije i negacije potrebno je generalizovati PFCSP sisteme, čime dobijamo generalizovani problem zadovoljenja fazi ograničenja sa prioritetima ili GPFCSP. Za disjunkciju uzimamo t-konormu dualnu t-normi izabranoj za konjunkciju, dok za negaciju koristimo standardnu negaciju: $N(x) = 1-x$. To znači da je potrebno proširiti treću i četvrtu aksiomu PFCSP-a, pošto su ostale aksiome odnose samo na konjunkciju ograničenja, te u ovom slučaju ostaju nepromenjene. Izmene kod treće aksiome se odnose na dodavanje disjunkcije, dok je četvrtu aksiomu potrebno generalizovati tako da monotonost važi samo u slučaju kada se ne upotrebljava negacija. Proširena definicija i njene aksiome dati su u nastavku.

Definicija 2.23: Neka su X, D, C^f, ρ , g definisani kao u definiciji 2.22, a GPFCSP je devetorka $(X, D, C\ f, \rho\ , g, \wedge, \vee, \neg)$.

Elementarna formula je uređen par $\left(x, \rho\left(R_i^f\right)\right)$ gde je $R_i^f \in C^f$, a $x \in Dom\left(R_i^f\right)$ je stepen zadovoljenja u navedenim ograničenja R_i^f , dok je $\rho\left(R_i^f\right)$ njegov prioritet.

Formula GPFCSP-a je definisana na sledeći način:

1. Elementarna formula je formula.
2. Ako su f_1 i f_2 formule, onda su i $\wedge(f_1, f_2)$, $\vee(f_1, f_2)$ i $\neg(f_1)$ takođe formule.

Izračunavanje stepena zadovoljenja ograničenja $\alpha_F(\vartheta_X)$ za neku valuaciju ϑ_X se izračunava u odnosu na interpretaciju veznika.

Sistem je GPFCSP ako važi:

1. Ako je $F = \wedge_{i\in\{1,\dots,n\}} f_i$ GPFCSP formula, gde su $f_i, i \in \{1, \dots, n\}$ elementarne formule i neka je C^F skup ograničenja koji se pojavljuju u formuli i ako za fazi ograničenje R_{max}^F važi
   $$\rho_{max} = \rho_{max}(R_{max}^F) = max\{\rho(R^F | R^F \in C^F)\}$$
   tada za svaku formulu F važi i
   $$\mu_{R_{max}^f}\left(\vartheta_{var\left(R_{max}^f\right)}\right) = 0 \text{ sledi } \alpha_F(\vartheta_X) = 0\ .$$
2. Ako je $\exists\rho_0 \in [0,1]$ takvo da $\forall R^f \in C^f$ važi da je $\rho(R^f) = \rho_0$, onda je globalni stepen zadovoljenja ograničenja dat sa:
   $$\alpha_F(\vartheta_X) = F_\wedge(\vartheta_X)\ ,$$
   gde je $F_\wedge$ interpretacija logičke formule F u fazi logici.
3. Pretpostavimo da za R_i^f , $R_j^f \in C^f$ važi $\rho\left(R_i^f\right) \geq \rho\left(R_j^f\right)$, neka je dato $\delta > 0$ i neka su date dve kombinovane valuacije ϑ_X i $\vartheta_X{}'$, takve da za $\forall R^f \in C^f$ važi:
   a) ako $R^f \neq R_i^f$ i $R^f \neq R_j^f$, onda $\mu_{R^f}\left(\vartheta_{var(R^f)}\right) = \mu_{R^f}\left(\vartheta'_{var(R^f)}\right)$
   b) ako $R^f = R_i^f$, onda $\mu_{R^f}\left(\vartheta_{var(R^f)}\right) = \mu_{R^f}\left(\vartheta'_{var(R^f)}\right) + \delta$
   c) ako $R^f = R_j^f$, onda $\mu_{R^f}\left(\vartheta_{var(R^f)}\right) = \mu_{R^f}\left(\vartheta'_{var(R^f)}\right) + \delta$

   Tada, ako važi:
   $$F = \wedge_{k=1,\dots,n}\left(x_k, \rho\left(R_k^f\right)\right), x_k \in Dom(R_k)$$

$$F = \bigvee_{k=1,\dots,n} \left(x_k, \rho\left(R_k^f\right)\right), x_k \in Dom(R_k)$$

sledi da je:

$$\alpha_F(\vartheta_X) \geq \alpha_F(\vartheta'_X) .$$

4. Neka su date dve različite kombinacije valuacije ϑ_X i ϑ'_X sa osobinom da $\forall R^f \in C^f$ važi:

$$\mu_{R^f}\left(\vartheta_{var(R^f)}\right) \geq \mu_{R^f}\left(\vartheta'_{var(R^f)}\right)$$

Tada, ako formula *F* ne sadrži negaciju važi $\alpha_F(\vartheta_X) \geq \alpha_F(\vartheta'_X)$.

5. Ako postoji kombinovana valuacija ϑ_X takva da $\forall R^F \in C^F$ važi $\mu_{R^F}\left(\vartheta_{var(R^F)}\right) = 1$ i ako je *F* formula oblika $F = \bigwedge_{i\in\{1,\dots,n\}} f_i$ ili $F = \bigvee_{i\in\{1,\dots,n\}} f_i$, gde su f_i , $i \in \{1, \dots, n\}$ elementarne formule, tada sledi da je $\alpha_F(\vartheta_X) = 1$.

Više o GPFCSP sistemima i njihovoj primeni može se pročitati u radovima koji se bave ovom tematikom, a koji su navedeni i opisani u sledećem poglavlju.

U nastavku sledi teorema koju opisuje i dokazuje Takači u radu (Takači, 2006), a koja daje jedan specijalan GPFCSP sistem.

Teorema 2.1: Sledeći sistem $(X, D, C^f, \rho, g, \wedge, \vee, \neg, \lozenge)$ gde je $\wedge = T_L$, $\vee = S_L$, $\neg = N_S$ i, konačno, $\lozenge(x_i, p_i) = S_P(x_i, 1 - p_i)$ je GPFCSP. Globalni stepen zadovoljenja ograničenja valuacije ϑ_X za formula F se računa na sledeći način:

$$\alpha_F(\vartheta_X) = \Phi\left\{\lozenge\left(\vartheta_{X_i}\right)^{\frac{\rho\left(R_i^f\right)}{\rho_{max}}} \middle| R^f \in C^f\right\}$$

gde je C^f skup ograničenja formule F, $\rho_{max} = \max\{\rho(R_i^f), R^f \in C^f\}$, a Φ je interpretacija formule F u GPFCSP sistemu.

Primer koji sledi prikazaće izračunavanje globalnog stepena zadovoljenja ograničenja za datu valuaciju i služi da se uoči veza između GFPCSP sistema i selektivnog upita.

Pretpostavimo da želimo da kupimo automobil. Iz grupe ponuđenih automobila, potrebno je izabrati najbolju ponudu. Kriterijumi za odabir su mala potrošnja, što duži garantni rok i niska cena.

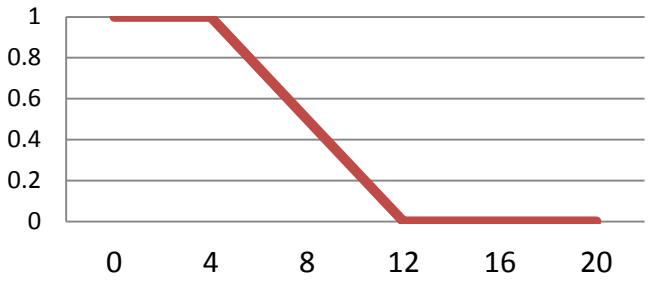
Neka postoji sledeći skup promenljivih i njihovih domena:

- X_1 = potrošnja, d_1 = [0, 20],
- X_2 = garancija, d_2 = [0, 10],
- X_3 = cena, d_3 = [0, 30.000].

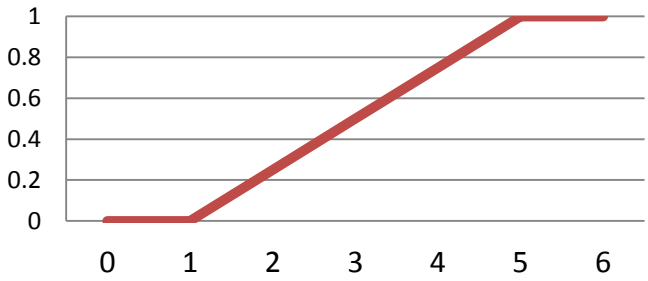
Skup ograničenja bi mogao izgledati ovao:

- R_1^f = mala potrošnja,
- R_2^f = duža garancija,
- R_3^f = niska cena,

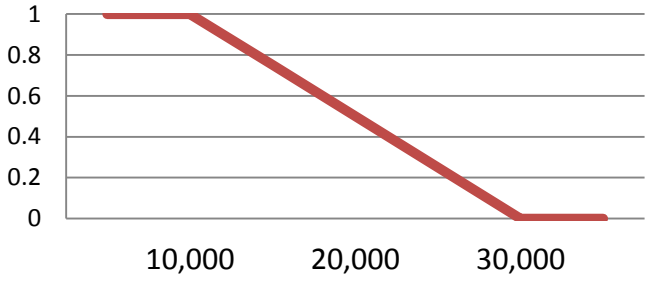
Ograničenja predstavljaju fazi podskupove odgovarajućih domena. Funkcije pripadnosti navedenih ograničenja mogle bi da izgledaju kao na slikama 2.14, 2.15 i 2.16.



Slika 2.14. Funkcija pripadnosti R_1^f – Mala potrošnja.



Slika 2.15. Funkcija pripadnosti R_2^f– Duža garancija.



Slika 2.16. Funkcija pripadnosti R_3^f– Niska cena.

Pretpostavimo dalje da nad grupom uslova postoje određeni prioriteti: $\rho(R_1^f) = 0.8$, $\rho(R_2^f) = 0.7$ i $\rho(R_3^f) = 1$. To znači da je u odabiru automobila najbitnija cena (prioritet 1), zatim mala potrošnja (prioritet 0.8) i na kraju garantni rok (prioritet 0.7).

Neka kao ponuda postoji grupa automobile data u tabeli 2.3.

Naziv	Potrošnja (l/100km)	Garancija (godina)	Cena (euro)
Renault Clio	6.0	2	14000
Audi A3	5.5	3	23000
Mercedes C	7.0	4	25000
Fiat 500L	6.0	2	13000
VW Golf 7	5.0	3	20000

Tabela 2.3. Testni podaci.

Dalje se može izračunati stepen zadovoljenja ograničenja, kao presek zadatih vrednosti i navedenih funkcija pripadnosti datih ograničenja. Rezultati su dati u tabeli 2.4.

Naziv	Potrošnja (l/100km)	Garancija (godina)	Cena (euro)
Renault Clio	0.75	0.25	0.8
Audi A3	0.8125	0.5	0.35
Mercedes C	0.625	0.75	0.25
Fiat 500L	0.75	0.25	0.85
VW Golf 7	0.875	0.5	0.5

Tabela 2.4. Stepeni zadovoljenja ograničenja.

Formula za izračunavanje globalnog stepena zadovoljenja ograničenja za konkretni primer izgledala bi ovako:

$$\alpha = T_L\left(T_L\left(S_P\left(\mu_{R_1^f}(\vartheta), 1-\rho\left(R_1^f\right)\right), S_p\left(\mu_{R_2^f}(\vartheta), 1-\rho\left(R_2^f\right)\right)\right), S_p\left(\mu_{R_3^f}(\vartheta), 1-\rho\left(R_3^f\right)\right)\right)$$

Gde je $S_P(x,y)$ t-konorma, $T_L(x,y)$ t-norma, i važe sledeće jednačine:

$$S_P(x,y) = x + y - xy$$
$$T_L(x,y) = max(x+y-1, 0)$$

Sada se izračunavaju vrednosti globalnog zadovoljenja ograničenja za svaki automobil iz liste.

$$\alpha_{Renault} = T_L\left(T_L\left(S_P(0.75, 0.2), S_p(0.25, 0.3)\right), S_p(0.8, 0)\right)$$
$$= T_L(T_L(0.8, 0.475), 0.8) = T_L(0.275, 0.8) = 0.075$$
$$\alpha_{Audi} = T_L\left(T_L\left(S_P(0.8125, 0.2), S_p(0.5, 0.3)\right), S_p(0.35, 0)\right)$$
$$= T_L(T_L(0.85, 0.65), 0.35) = T_L(0.5, 0.35) = 0$$
$$\alpha_{Mercedes} = T_L\left(T_L\left(S_P(0.625, 0.2), S_p(0.75, 0.3)\right), S_p(0.25, 0)\right)$$
$$= T_L(T_L(0.7, 0.825), 025) = T_L(0.525, 0.25) = 0$$
$$\alpha_{Fiat} = T_L\left(T_L\left(S_P(0.75, 0.2), S_p(0.25, 0.3)\right), S_p(0.85, 0)\right)$$
$$= T_L(T_L(0.8, 0.475), 0.85) = T_L(0.275, 0.85) = 0.125$$
$$\alpha_{Volkswagen} = T_L\left(T_L\left(S_P(0.875, 0.2), S_p(0.5, 0.3)\right), S_p(0, 0.5)\right)$$
$$= T_L(T_L(0.215, 0.65), 0.5) = T_L(0, 0.5) = 0$$

Na osnovu dobijenih rezultata sledi da su samo dva automobila delimično zadovoljila zadate kriterijume, Fiat 500L sa 12,5% i Renault Clio sa 7,5% poklapanja.

Vidimo da se zadati upit lako može postaviti upotrebom nekog standardnog upitnog jezika. Tako bi zadati upit upotrebom XQuery upitne sintakse izgledao kao u listingu 2.29.

```
for $item in /automobili
      let $naziv:=$item/@naziv
      let $potrosnja:=$item/potrosnja
      let $garancija:=$item/garancija
      let $cena:=$item/cena
where $potrosnja = "mala potrošnja" AND
      $garancija = "duza garancija" AND
      $cena = "niska cena"
return
<lokacija>{$naziv}</lokacija>
```

Listing 2.29. Upotreba XQuery sintakse u postavljaju upita.

Glava III

PRETHODNI RADOVI I PRAVCI ISTRAŽIVANJA

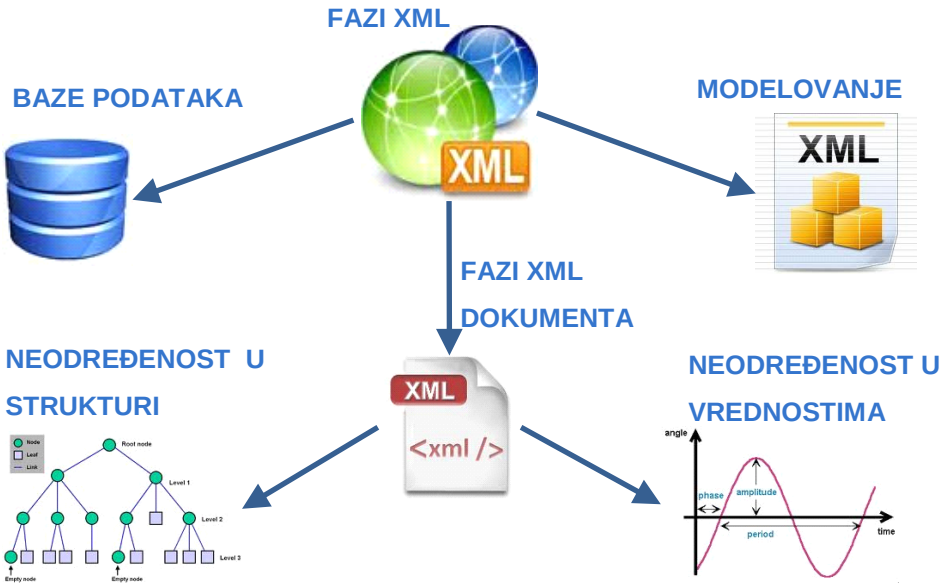
Poglavlje daje pregled dosadašnjih istraživanja i analizu radova iz oblasti upotrebe fazi logike u definisanju neodređenosti u XML dokumentima.

Glavni cilj petogodišnjeg istraživanja sprovedenog na Univerzitetu u Novom Sadu od strane više autora je implementacija sistema sposobnog da koristi fazi logiku sa prioritetima u relacionim bazama podataka. Štaviše, definisano je i implementirano kompletno aplikativno rešenje nad fazi relacionom bazom podataka (Škrbić, 2008), (Škrbić, et al., 2013), (Škrbić, et al., 2011), (Takači & Škrbić, 2008) i (Škrbić, et al., 2011).

Korišćenjem koncepta GPFCSP-a iz radova (Takači, 2005), (Pap & Takači, 2005), (Takači & Škrbić, 2008) i (Takači, et al., 2009), pronađen je način za predstavljanje upita sa prioritetima u fazi relacionim bazama podataka, što je rezultiralo PFSQL (Priority Fuzzy Structured Query Language) upitnim jezikom. PFSQL omogućava postavljanje uslova u WHERE klauzulama upita, koji imaju različite stepene prioriteta. GPFCSP daje teorijsku osnovu za realizaciju upita sa prioritetima. Autori su takođe uspeli da formalizuju PFSQL upite dobijanjem interpretacije u postojećoj fazi logici. Ustanovljeno je da ŁΠ½ logika pruža dovoljno elemenata (Perović, et al., 2009) i (Perović, et al., 2012).

U nastavku ovog istraživanja proučavane su mogućnosti upotrebe fazi logike u definisanju neodređenosti u XML dokumentima. Na ovu temu objavljena su dva naučna rada *Fuzzy XML with Implementation* (Panić, et al., 2012) i *Fuzzy XML and prioritized fuzzy XQuery with implementation* (Panić, et al., 2013), koji ukratko definišu dostignuća istraživanja, a koja će biti detaljno opisana u samoj disertaciji.

Grupa naučnih radova koji obuhvataju istraživanja iz oblasti upotrebe fazi logike u XML-u, a koji su objavljeni u poslednjih desetak godina, analizirani su u nastavku. U najuticajnije pravce istraživanja ubrajaju se definisanje neodređenosti u strukturi XML-a, definisanje neodređenosti u vrednostima XML-a, upotreba fazi logike u XML bazama podataka i modelovanje fazi XML dokumenata. Na slici 3.1 dat je prikaz navedenih pravaca.



Slika 3.1. Najuticajniji pravci istraživanja upotrebe fazi logike u XML-u.

Da bi neka aplikacija mogla da koristi XML dokumenta, potrebno je da poseduje preciznu definiciju strukture dokumenata koje obrađuje. Kako se XML dokumenti razvijaju kao i svi ostali delovi informacionog sistema, uvođenje nove izmenjene verzije dokumenta prouzrokovalo bi nemogućnost njegove upotrebe od strane aplikacija koje ga čitaju. To je samo jedan od problema koji se može rešiti upotrebom fazi logike pri određivanju strukture neprecizno definisanog dokumenta. U definisanju neodređenosti u strukturi XML-a autori najčešće proširenju XPath i XQuery jezik. Rešenja omogućavaju pronalaženje elemenata u XML dokumentu čije pozicije nisu precizno definisane.

Tako Campi i ostali u radu (Campi, et al., 2009) opisuju XML upitni okvir nazvan *FuzzyXPath*. On je baziran na teoriji fazi skupova i oslanja se na fazi uslove za definisanje fleksibilnih ograničenja nad podacima. Uvodi se funkcija *deep-similar*, koja bi zamenila XPath *deep-equal* funkciju. Ideja je određivanje stepena sličnosti između dva XML stabla, tačnije procene da li su dva stabla slična po pitanju strukture i sadržaja. Sličnu stvar rade Damiani, Marrara i Pasi u radu (Damiani, et al., 2007) gde proširuju XPath elementima fazi logike, u cilju omogućavanja rukovanja neodređenom strukturom XML-a. MA, Liu i Yan u radu (Ma, et al., 2010) predlažu novi fazi XML model zasnovan na XSD šemi**.** Upotreba ovog modela omogućava predstavljanje fazi informacije u XML dokumentu. Lo, Kianmehr, Kaya i Alhajj u radu (Lo, et al., 2007) dodaju fazi logiku kao proširenje VRXQuery upitnog jezika, koji podržava interfejs VIREX. Verovatno najznačajniji rad iz ove grupe je rad (Yan, et al., 2009) autora Yan, Ma i Liu, koji daju predlog za modelovanje neodređenosti u XML strukturi upotrebom XSD dokumenata.

Osnovne ideje koje predlažu Yan, Ma i Liu, a odnose se na grupisanje i verovatnoću pojavljivanja, proširuju se i primenjuju u disertaciji.

Vrednosti u XML dokumentima takođe mogu biti neodređene. Informacije koje se upotrebljavaju u svetu oko nas obično nisu precizno definisane. Klasična logika zahteva precizno definisanje svojih elemenata i nije u stanju da radi sa vrednostima koje nisu definisane na taj način. Korisnik često nije u mogućnosti da precizno preslika vrednosti iz

realnog sveta u vrednosti XML dokumenta. Za definisanje takvih vrednosti XML-a obično se upotrebljavaju XML šeme, a ponekad i šema grafovi.

Oliboni i Pozzani u radu (Oliboni & Pozzani, 2008) predlažu generalnu XML šemu za prikaz fazi informacija u XML-u. Za implementaciju se koriste XSD dokumenti, pomoću kojih se definišu fazi tipovi podatka. Rad se fokusira isključivo na definisanje fazi informacija, pod pretpostavkom da korisnik već ima opšte XML šeme koje definišu druge delove XML dokumenta. Uvode se fazi tipovi podataka, koji se klasifikuju u četiri kategorije *classicType*, *fuzzyOrdType*, *FuzzyNonOrdSimilarityType* i *fuzzyNonOrdType*, koje se dalje obrađuju na različite načine. Tsang, Khamisy i Vu u radu (Tseng, et al., 2005) implementiraju XSD i DTD dokumenta, čiji je cilj opisivanje sintakse za predstavljanje generalnih XML fazi sistema. Dat je skup elemenata, sa opisom funkcionalnosti, koji se mogu koristiti u delovima fazi sistema. Lee i Fanjiang u radu (Lee & Fanjiang, 2003) se bave razvojem FOOM (Fazi Objektno Orjentisano Modeliranje) tehnika zasnovanih na XML šemama. FOOM šeme se takođe automatski transformišu u skup interfejsa za programiranje aplikacija. Transformacija ovih šema u skup za proveru validnosti sadržaja i pristup podacima vrši se preko šema grafa. Šema graf je ekstenzija DTD grafa sa tipovima informacija koji služe kao posredni prikaz za opisivanje strukture XML šeme. Herrera-Viedma i ostali, u radu (Herrera-Viedma, et al., 2007) predstavljaju model na bazi fazi logike, za merenje kvaliteta informacija XML dokumenata na Web-u. Model vrši procenu informacija o kvalitetu Web lokacija koristeći stavove korisnika. Sastoji od dve glavne komponente, evaluacione šeme za analiziranje kvaliteta podataka i metoda za rangiranje kvaliteta. Fazi lingvističke tehnike su uključene u proces evaluacije kvaliteta. Radovi (Damiani, et al., 2000), (Hailong, et al., 2008), (Combi, et al., 2005) i (Damiani, et al., 2001) se takođe mogu koristiti kao dobar izvor informacija na ovu temu.

U poređenju sa navedenom grupom radova, ova disertacija se bazira na upotrebi XML šema, kao provereno uspešnog načina za definisanje neodređenih vrednosti. Neke od opisanih ideja primenjene su i usavršene u ovom istraživanju.

Neretko se XML dokumenti skladište u baze podataka, koje sada uglavnom podržavaju rad sa XML tipovima podataka. Tako na primer u radu (Barranco, et al., 2005) Barranco, Campaña i Medina uvode XFSQL (XML Fuzzy Structured Query Language) upitni jezik, koji predstavlja strukturirani format za brzi upitni prikaz podataka i izbegava specifičnosti različitih implementacija upitnih jezika. Komunikacija sa različitim bazama podataka ostvaruje se uz pomoć transformacija. Transformacija XFSQL sintakse u posebnu sintaksu baze podataka može se uraditi pomoću XSL transformacije ili programa specijalno dizajniranih za tu svrhu. Proučavanjem različitih FDBMS predloga i njihovih modela utvrđen je osnovni skup zajedničkih fazi karakteristika, koji je potom implementiran u XFSQL jeziku. XFSQL je definisan u obliku XML šeme, koja fazi sintaksom proširuje standardne mogućnosti jezika. Gaurav i Alhajj u radu (Gaurav & Alhajj, 2006) opisuju mapiranje fazi relacionih baze podataka na fazi XML. Dodatne informacije o modelovanju baze podataka uvođenjem fazi tipova mogu se pronaći u Ma-ovoj knjizi *Fuzzy Database Modeling with XML* (Ma, 2005) i radovima (Ma & Shen, 2006) i (Ma, et al., 2012).

Ponekad autori proširuju SQL jezik i XQuery upitnu sintaksu fazi elementima. Na primer Rodrigues, Cruz i Cavalcante u svom radu (Rodrigues, et al., 2009) predstavljaju novu *Alianca* fazi arhitekturu baze podataka i obrađuje FSQL sintaksi. Seto, Clement, Duong, Kianmehr i Alhajj u radu (Seto, et al., 2009) predstavljaju sistem sa *GUI Query Builder*-om koji generiše XQuery izraze sa podrškom za klasičnu i fazi logiku. Tomson i Radhamani u radu (Thomson & Radhamani, 2009) opisuju prirodan način za izdvajanje podataka iz XML

dokumenata upotrebom XQuery jezika i primenom tehnika fazi logike, za izdvajanje podataka iz XML dokumenata. Oni definišu fleksibilan XQuery upitni jezik baziran na fazi logici. Predloženi jezik obrađuje tehnike koje dozvoljavaju upotrebu lingvističkih termina baziranih na korisnički definisanim funkcijama jezika. Rad pokazuje kako fleksibilan XQuery pruža bolje rezultate od neproširenog XQuery jezika. Disertacija usvaja neke ideje iz prethodnog rada i proširuje ih prioritetima i pragovima zadovoljenja. Chan, Situ, Wong, Kianmehr and Alhajj u radu (Chan, et al., 2008) opisuju dizajn i implementaciju fazi upitnog ugnježdenog sistema za XML baze podataka. Dok Jin i Shidlagatta u radu (Jin & Shidlagatta, 2009) definišu upotrebu fazi logike u trigerima. Postoji još zanimljivih radova koji obrađuju ovu temu, kao na primer (Koyuncu, 2011), (Zhang, et al., 2011) i (Zhang, et al., 2012).

Disertacija koristi bazu podataka za skladištenje XML dokumenata i kao osnovu za XQuery upite. Ona se neće baviti proširenjem SQL jezika.

Dešava se da u realnim sistemima nedostaju informacije. Codd u svom radu (Codd, 1986) predlaže da se uvedu dve različite vrste Null vrednosti, nepoznata vrednost i neprimenljiva vrednost. Da bi se preciznije opisala razlika između ova dva tipa Null vrednosti Tré i ostali u radu (Tré, et al., 2008) daju primer skladištenja podataka o brzini leta ptice i opisuju razlike u nastalim Null vrednostima. Prade i Testemale u radu (Prade & Testemale, 1984) zaključuju da je u fazi bazama podataka neophodno da postoji bar jedna vrsta Null vrednosti, koja bi definisala vrednost neprimenljiva. Obrada Null vrednosti zasnovana na tro-vrednosnoj Kleene logici tema je Rescher-ovog rada (Rescher, 1969). Upotreba tro-vrednosne Kleene logike u definisanju Null vrednosti obrađene u radu Tré-a (Tré, 2002), nazvana je EPTV (Extended Possibilistic Truth Values). Ona pruža iste mogućnosti, kao rešavanje problema pomoću Codd-ove četvoro-vrednosne logike. Tré & Caluwe u radu (Tré & Caluwe, 2003) opisuju upotrebu EPTV u fazi upitima.

U poređenju sa navedenom grupom radova, ova disertacija se ograničava na upotrebu dva tipa Null vrednosti: nepoznata i neprimenljiva. Vrednost tipa nedostupna, koja je opisana u EPTV tro-vrednosnoj logici, svodi se na vrednost tipa nepoznata.

Tabelarni prikaz poređenja doktorske disertacije sa značajnim radovima drugih autora dat je u tabeli 3.1.

Radovi koji se navode u tabeli 3.1 su najznačajniji radovi iz ove oblasti, a porede se po najbitnijim funkcionalnostima. Na osnovu tabele se lako može zaključiti da doktorska disertacija pokriva mnogo veću oblast od bilo kog pojedinačnog rada. Fazi SQL nije mogao da bude implementiran jer se disertacija bavi XML dokumentima, a ne relacionim bazama podataka. Razvoj fazi XPath-a je izvan domena ovog rada. Fokus rada je stavljen na definisanje i određivanje neodređenosti u vrednostima XML elemenata. Definisanje neodređenosti u strukturi elemenata je podržano i implementirano, ali je njihova obrada izostavljena iz implementacije zbog obimnosti rada. Fazi XPath obuhvata upravo deo koji se bavi izračunavanjem verovatnoće pojavljivanja nekog elementa u XML dokumentu i njegov razvoj biće tema narednih radova. Podržano je i implementirano definisanje Null vrednosti u XML dokumentima. Mehanizam obrade Null vrednosti je delimično podržan, o čemu će kasnije biti više reči. Fazifikacija teksta je funkcionalnost koja do sada nije bila obrađivana u naučnim radovima i koja će predstavljati jedan od pravaca budućih istraživanja. U poglavlju koje obrađuje mogućnosti praktične primene, ukratko je objašnjena problematika fazifikacije teksta.

RADOVI	(Škrbić, et al., 2013)	(Campi, et al., 2009)	(Yan, et al., 2009)	(Tseng, et al., 2005)	(Barranco, et al., 2005)	(Thomson & Radhamani, 2009)	(Seto, et al., 2009)	(Tré, et al., 2008)	Disertacija
Fazi XML	+	-	+	+	+	-	-	-	+
Neodređenost u strukturi	-	+	+	-	-	-	-	-	+
Neodr. u vrednostima	+	-	-	+	+	+	+	-	+
DTD podrška	-	-	-	+	-	-	-	-	+
XSD podrška	-	-	-	+	-	-	-	-	+
Fazi SQL	+	-	-	-	+	-	-	+	-
Fazi XQuery	-	-	-	-	-	+	+	-	+
Fazi XPath	-	-	+	-	-	-	-	-	-
Fazifikacija funkcija	+	-	-	+	+	-	+	-	+
Fazifikacija teksta	-	-	-	-	-	-	-	-	-
Prioriteti	+	-	-	-	-	-	-	-	+
Null vrednosti	-	-	-	-	-	-	-	+	+ / -
Implementacija	+	-	+	+	+	-	+	-	+
Testiranje aplikacije	-	-	-	-	-	-	-	-	+
Primeri primene u praksi	-	-	-	-	-	-	-	-	+

Tabela 3.1. Poređenje sa drugim značajnim radovima

Glava IV

FAZI XML SINTAKSA

Poglavlje daje predlog fazi XML sintakse koja predstavlja proširenje standardnog XML-a i omogućava skladištenje neodređenosti u vrednostima i strukturi XML dokumenta. Sintaksa je definisana pomoću XSD i DTD dokumenata. U prvom delu poglavlja fazi XML je ograničen samo na upotrebu neodređenosti u vrednostima XML elemenata, dok drugi deo sadrži predlog definisanja neodređenosti u strukturi XML-a.

4.1 Neodređenost u XML vrednostima

Da bi se definisao fazi skup neophodno je poznavati njegovu funkciju pripadnosti. Međutim, definisanje funkcija pripadnosti nije nimalo lak zadatak, a njihovo skladištenje i obrada zahtevaju visok nivo složenosti pri implementaciji. Iz ovog razloga javljaju se dva moguća pravca realizacije ove ideje.

Prvi način za definisanje fazi skupova obuhvata upotrebu određenog broja jednostavnih i dobro poznatih tipova funkcija (na primer rastuća, opadajuća, trougaona ili trapezoidna). Pri odabiru grupe fazi skupova koje je potrebno implementirati, posebnu pažnju treba obratiti na izbor skupova koji će pokriti što veći broj realnih sistema. Problem je u tome što upotreba ograničenog broja predefinisanih funkcija pripadnosti ne daje mogućnost definisanja slučajeva čije funkcije nisu opisane. Drugi način obuhvata definisanje sintakse koja ima mogućnosti da opiše proizvoljnu funkciju pripadnosti. Upravo ovaj pristup je primenjen u realizaciji ovog istraživanja.

Kako je XML po definiciji proširiv meta jezik za uvođenje fazi logike u XML nije potrebno dodatno proširenje sintakse samog XML jezika. Ipak, potrebno je osmisliti XML strukturu koja je dovoljno fleksibilna da omogući definisanje najrazličitijih primera fazi skupova. Najjednostavniji način za definisanje fazi tipa je upotreba neke XML šeme. Predlog takve sintakse dat je u listingu 4.1, a prikazuje XSD dokument koji definiše fazi tip XML elementa.

```
<xs:schema id="Fuzzy"
    elementFormDefault="qualified"
    xmlns:xs="http://www.w3.org/2001/XMLSchema">

    <!-- Fazi tip -->
    <xs:complexType name="fuzzyType">
        <xs:choice minOccurs="0" maxOccurs="1">
            <xs:element fer="fuzzyValue"/>
            <xs:element fer="fuzzyLabel"/>
            <xs:element fer="fuzzyCrisp"/>
        </xs:choice>
    </xs:complexType>

    <!-- Fazi vrednost -->
    <xs:element name="fuzzyValue">
        <xs:complexType>
```

```
            <xs:sequence>
               <xs:element name="fuzzy" type="fuzzy"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>

    <!-- Lingvisticka promenljiva -->
    <xs:element name="fuzzyLabel">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="fuzzy" type="xs:string"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>

    <!-- Diskretna vrednost -->
    <xs:element name="fuzzyCrisp">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="fuzzy" type="xs:decimal"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>

    <!-- Fazi skup -->
    <xs:complexType name ="fuzzy">
        <xs:sequence>
            <xs:element ref="function" minOccurs ="1"
            maxOccurs="unbounded"/>
        </xs:sequence>
        <xs:attribute name="name" type="xs:string"/>
    </xs:complexType>

    <!-- Funkcija pripadnosti ili njen deo -->
    <xs:element name="function">
        <xs:complexType>
           <xs:simpleContent>
              <xs:extension base="xs:string">
                  <xs:attribute name="minValue" type="xs:string" />
                  <xs:attribute name="maxValue" type="xs:string" />
               </xs:extension>
           </xs:simpleContent>
        </xs:complexType>
    </xs:element>
</xs:schema>
```

Listing 4.1. XSD definicija fazi tipa XML elementa – Fuzzy.xsd.

Prethodni listing definiše *fuzzyType* element, koji može sadržati tri tipa pod-elemenata: fazi vrednost *fuzzyValue*, fazi lingvističku promenljivu *fuzzyLabel* i diskretnu (crisp) vrednost *fuzzyCrisp*. **Fazi vrednost** se definiše imenom pomoću atributa *name*, na primer *<fuzzy name="hladno">*. On sadrži element *function*, koji matematički predstavlja fazi funkciju pripadnosti, čiji izgled opisuje vrednost elemenata, a interval njegovi atributi. Bitno je napomenuti da se zbog lakšeg definisanja funkcija pripadnosti može sastojati iz više delova, od kojih je svaki deo definisan sa po jednim *function* elementom. Spajanjem tako definisanih delova, dobija se jedinstvena funkcija pripadnosti. Granične vrednosti svakog dela funkcije definisane su atributima *minValue* i *maxValue* i opisuju interval nad kojim važi funkcija, na primer *<function minValue="10" maxValue="15">*. Granične vrednosti

pojedinačnih funkcija unutar jednog *fuzzy* elementa ne smeju se preklapati. U slučaju da neki od atributa nije definisan, tada se smatra da njegova vrednost teži beskonačno za *maxValue* ili minus beskonačno za *minValue*. Rezultat funkcije pripadnosti označava stepen pripadanja, koji može imati vrednost iz skupa [0,1]. Intervali za koje nije definisana funkcija pripadnosti imaju vrednost 0. Tekst kojim se definiše vrednost mora biti naveden u odgovarajućem formatu, u zavisnosti od toga koji se parser koristi za prevođenje u ekvivalentnu matematičku funkciju. Na primer, u implementaciji baziranoj na MATLAB-u, funkcije se definišu u formatu *A*x + B*, gde su A i B realni brojevi, a *x* promenljiva koja uzima vrednosti iz zadatog intervala.

Isto ovo je moguće uraditi upotrebom drugih tipova XML šema. Ekvivalentna definicija listinga 4.1, ali ovog puta upotrebom DTD datoteke, data je u listingu 4.2.

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT fuzzyType ANY>
<!ELEMENT fuzzyValue (fuzzy)>
<!ELEMENT fuzzyLabel (fuzzy)>
<!ELEMENT fuzzyCrisp (fuzzy)>
<!ELEMENT fuzzy ANY>
<!ATTLIST fuzzy
          name CDATA #REQUIRED>
<!ELEMENT function (#PCDATA)>
<!ATTLIST function
          minValue CDATA #REQUIRED
          maxValue CDATA #REQUIRED>
```

Listing 4.2. DTD definicija fazi tipa.

Da bi se definisala struktura nekog fazi XML dokumenta upotrebom XSD šeme potrebno je u nju uključiti *Fuzzy.xsd* dokument, a potom definisati tip elementa kao *fuzzyType*. Primer upotrebe *fuzzyType* tipa u XSD dokumentu prikazana je u listingu 4.3.

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema id="Test"
    elementFormDefault="qualified"
    xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:include schemaLocation ="Fuzzy.xsd"/>
    <xs:element name="merenje">
        <xs:complexType>
            <xs:sequence>
              <xs:element name="naziv" type="xs:string"/>
              <xs:element name="lokacija" type="xs:int"/>
              <xs:element name="vreme" type="xs:dateTime"/>
              <xs:element name="temperatura" type="fuzzyType"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
</xs:schema>
```

Listing 4.3. Upotreba fazi tipa u XSD dokumentima.

Primer fazi XML dokumenta koji je definisan u skladu sa opisanom sintaksom i prikazuje rezultate merenja temperature vazduha, dat je u listingu 4.4.

```
<merenje>
      <naziv>Meteorološka stanica Novi Sad</naziv>
      <lokacija>1</lokacija>
      <vreme>2011-09-04</vreme>
      <temperatura>
            <fuzzy name="hladno">
                  <function maxValue="10">1</function>
                  <function minValue="10" maxValue="20">
                        -1/(Max-Min)*x + Max/(Max-Min)
                  </function>
            </fuzzy>
      </temperatura>
</merenje>
```

Listing 4.4. Primer fazi XML dokumenta.

Listing 4.4 je definisan u skladu da XSD definicijom iz listinga 4.3. Fazi tip *temperatura* je opisan fazi skupom *hladno*, koji sadrži funkciju pripadnosti sastavljenu iz više delova. Na primer funkcija *hladno* ima stepen pripadnosti 1 (apsolutno tačan) dok temperatura ne dostigne vrednost 10. Nakon toga vrednost linearno opada do temperature 20, gde ima vrednost 0. Linearno smanjenje je opisano sledećom funkcijom *-1/(Max-Min)*x + Max/(Max-Min)* čije su konstante *Min* i *Max* definisane u atributima *minValue = "10"* i *maxValue = "20"*, a promenljiva x uzima vrednost iz navedenog intervala. Deo funkcije koji opisuje temperaturu preko 20 nije definisan i podrazumeva se da funkcija pripadnosti nad tim intervalom ima vrednost 0. Spajanjem svih delova dobijamo funkciju koja nosi fazi informaciju o tome koliko je hladno.

Prethodni primer opisuje slučaj gde fazi tip sadrži fazi vrednost. Međutim, fazi tip može sadržati i takozvanu **fazi lingvističku promenljivu**, kao tekstualnu vrednost koja započinje znakom #. Tako definisana promenljiva predstavlja predefinisanu funkciju pripadnosti, koja mora sadržati bar jedan *function* element, nalik onom iz listinga 4.4. Primer upotrebe fazi lingvističke promenljive dat je u listingu 4.5.

```
<fuzzy>#hladno</fuzzy>
```

Listing 4.5. Fazi lingvistička promenljiva u fazi tipu.

Poslednji tip vrednosti koju može sadržati fazi tip je **diskretna (crisp) vrednost**, a primer je dat u listingu 4.6. Crisp vrednost predstavlja svaka strogo definisana vrednost (konstanta). U konkretnom primeru iz listinga 4.6, to je vrednost 6.1.

```
<fuzzy>6.1</fuzzy>
```

Listing 4.6. Diskretna vrednost u fazi tipu.

Crisp vrednost predstavlja svaku strogo definisanu vrednost (konstantu). To je recimo vrednost 6.1 u primeru iz listinga 4.6.

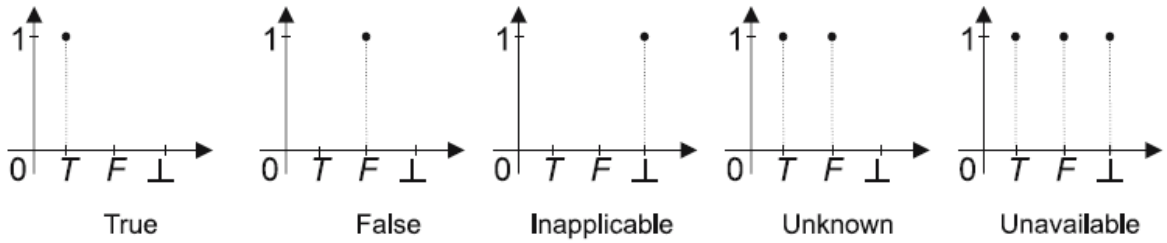
4.1.1 Problem Null vrednosti

Nedostatak informacija je česta pojava, posebno u realnim sistemima, te je potrebno definisati vrednosti elemenata kojima se mogu opisati nedostajući podaci. Tradicionalni sistemi baza podataka problem rešavaju upotrebom Null vrednosti i logike bazirane na tri stanja.

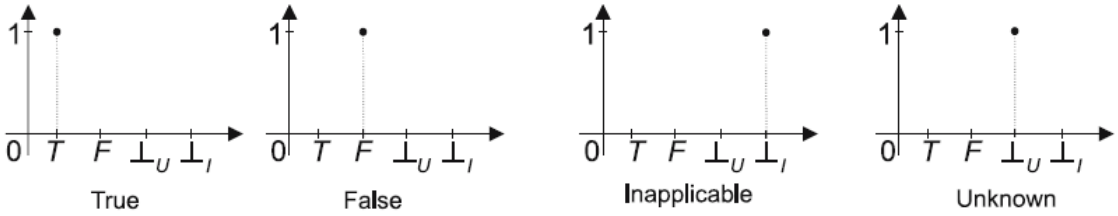
Codd u svom radu (Codd, 1986) predlaže da se uvedu dve različite vrste Null vrednosti, *nepoznata* vrednost i *neprimenljiva* vrednost. Nepoznata vrednost je vrednost koja je nepoznata korisnicima, ali su podaci primenljivi i mogu se uneti kada budu bili dostupni. Vrednost je neprimenljiva onda kada nije moguće definisati takve podatke za posmatrani objekat. Da bi se preciznije opisala razlika između ova dva tipa Null vrednosti Tré i ostali u radu (Tré, et al., 2008) daju primer skladištenja podataka o brzini leta ptice. Tako ukoliko se posmatra ptica koja leti i ako nije moguće utvrditi njenu brzinu, koristimo vrednost *nepoznata*, dok ukoliko bi posmatrali recimo pingvina, koji ne može da leti, u polje brzina leta bi trebalo upisati vrednost *neprimenljiva*.

Autori zapažaju da se pristup definisanju Null vrednosti kod fazi baza podataka dosta razlikuje u odnosu na klasične baze podataka. Razlog za to je što skoro svi fazi modeli omogućavaju definisanje nepoznatih informacija bez potrebe za dodatnim obrađivanjem Null vrednosti. Zaključak je primenljiv nad tipom *nepoznata*, ali modeliranje tipa *neprimenljiva* zahteva dodatni element za njeno definisanje. To znači da je u fazi bazama podataka neophodno da postoji bar jedna vrsta Null vrednosti, koja bi definisala vrednost *neprimenljiva* (Prade & Testemale, 1984). Adekvatna logika za ovaj slučaj može se dobiti pomoću tro-vrednosne logike sa vrednostima tačno (T), netačno (F) i neprimenjiva (⊥). Takva logika, zasnovana na tro-vrednosnoj Kleene logici (Rescher, 1969), razvijena je u radu Tré-a (Tré, 2002) i nazvana EPTV. Navedeni pristup kod fazi skupa podataka pruža iste mogućnosti, kao rešavanje problema pomoću Codd-ove četvoro-vrednosne logike, koja sadrži dva tipa Null vrednosti. Slika 4.1 preuzeta je iz rada (Tré, et al., 2008) i daje uporedne prikaze mogućnosti tro-vrednosne EPTV logike i Codd-ove četvoro-vrednosne logike.

a) Specijalni slučajevi EPTV tro-vrednosne logike



b) Specijalni slučajevi Codd-ova četvoro-vrednosne logike



Slika 4.1. Poređenje između EPTV i Codd-ove logike.

Kao što je prikazano u radu Tré & Caluwe (Tré & Caluwe, 2003), EPTV se može koristiti i u fazi upitima. Tačnije, logički okvir baziran na osnovu EPTV-a proširuje pristup koji nude Prade i Testemale (Prade & Testemale, 1984) i eksplicitno se bavi neprimenljivim informacijama u uslovima upita. Ukoliko je neki deo uslova upita neprimenljiv, to se mora odraziti na rezultat samog upita. Bitno je zapaziti postojanje interakcije među informacijama, jer često se na osnovu nje mogu odrediti okvirne vrednosti elementa koji nedostaje. Na primer, ako nedostaje informacija o maksimalnoj brzini automobila, a postoje informacije o njegovom tipu i snazi motora, tada se na osnovu dostupnih informacija može odrediti približna vrednost za nedostajuću maksimalnu brzinu. Rukovanje interakcijama između informacija je izvan opsega ovog rada i samog predmeta istraživanja.

Za potrebe ove disertacije ograničavamo se na upotrebu dva osnovna tipa Null vrednosti u fazi XML elementima, a to su **nepoznata** i **neprimenljiva**. Dok se vrednosti tipa **nedostupna**, definisane u EPTV tro-vrednosnoj logici, svode na vrednosti tipa nepoznata. Elementi XML dokumenta po pravilu ne moraju biti navedeni, ukoliko je u njihovoj šemi definisano da je minimalan broj pojavljivanja jednak nuli (*minOccurs="0"*). Tako definisana šema omogućila bi definisanje neodređenosti tipa *nepoznata*. Međutim, to još ne omogućava definisanje Null tipa *neprimenljiva*. Taj tip se može definisati upotrebom eksplicitnog definisanja Null elementa, što se postiže postavljanjem atribut elementa XML šeme na *nillable="true"*. Kada bi se navedeni zaključci primenili na XSD definiciju iz listinga 4.1, definicije fazi tipa *fuzzyType*, *fuzzyValue*, *fuzzyLabel* i *fuzzyCrisp* bile bi proširene i izgledale bi kao u listingu 4.7, dok bi poslednja dva elementa *fuzzy* i *function* ostala ista.

```
<!-- Fazi tip -->
<xs:complexType name="fuzzyType">
      <xs:choice minOccurs="0" maxOccurs="1">
            <xs:element fer="fuzzyValue"/>
            <xs:element fer="fuzzyLabel"/>
            <xs:element fer="fuzzyCrisp"/>
      </xs:choice>
</xs:complexType>

<!-- Fazi vrednost -->
 <xs:element name="fuzzyValue">
      <xs:complexType>
            <xs:sequence>
               <xs:element name="fuzzy" type="fuzzy" nillable="true"/>
            </xs:sequence>
      </xs:complexType>
</xs:element>

<!-- Lingvisticka promenljiva -->
<xs:element name="fuzzyLabel">
      <xs:complexType>
            <xs:sequence>
                <xs:element name="fuzzy" type="xs:string" nillable="true"/>
            </xs:sequence>
      </xs:complexType>
</xs:element>

<!-- Diskretna vrednost -->
<xs:element name="fuzzyCrisp">
      <xs:complexType>
            <xs:sequence>
```

```
            <xs:element name="fuzzy" type="xs:decimal" nillable="true"/>
        </xs:sequence>
    </xs:complexType>
</xs:element>

<!-- Fazi skup -->
<xs:complexType name ="fuzzy">
    <xs:sequence>
        <xs:element ref="function" minOccurs ="1"
        maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="name" type="xs:string"/>
</xs:complexType>

<!-- Funkcija pripadnosti ili njen deo -->
<xs:element name="function">
    <xs:complexType>
        <xs:simpleContent>
          <xs:extension base="xs:string">
              <xs:attribute name="minValue" type="xs:string" />
              <xs:attribute name="maxValue" type="xs:string" />
           </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
</xs:element>
```

Listing 4.7. XSD definicija fazi elementa koji može sadržati Null vrednost.

Ukoliko se primeni proširena XSD definicija *fuzzyType* elementa iz prethodnog listinga nad XML dokumentom iz listinga 4.4, tada je moguće kreirati XML dokumente čija bi fazi vrednost *temperatura* bila tipa *nepoznata* (recimo iz nekog razloga nije upisana temperatura u dokument, ali je izmerena) kao u listingu 4.8 ili *neprimenljiva* (praktično neizvodljivo nad datim primerom, potrebno bi bilo da temperatura na zadatom odredištu ne postoji, te se iz tog razloga ne bi mogla izmeriti) kao u listingu 4.9.

```
<merenje>
    <naziv>Meteorološka stanica Novi Sad</naziv>
    <lokacija>1</lokacija>
    <vreme>2011-09-04</vreme>
    <temperatura/>
</merenje>
```

Listing 4.8. Fazi element koji ima vrednost Null tipa *nepoznata*.

```
<merenje>
    <naziv>Meteorološka stanica Novi Sad</naziv>
    <lokacija>1</lokacija>
    <vreme>2011-09-04</vreme>
    <temperatura>
        <fuzzy nillable="true"/>
    </temperatura>
</merenje>
```

Listing 4.9. Fazi element koji ima vrednost Null tipa *neprimenljiva*.

4.2 Neodređenost u XML strukturi

Za XML dokumenta koja imaju promenljivu strukturu kažemo da su neodređena po strukturi. Potrebno je osmisliti sintaksu za definisanje neodređenosti strukture koja će biti u skladu sa XML standardom. Samu sintaksu XML jezika nije potrebno proširivati pošto je po definiciji proširiva. Sintaksa mora da obezbedi jednostavno definisanje stepena pojavljivanja elemenata ili grupe elemenata u XML dokumentu. Isto tako u slučaju definisanja grupe elemenata neophodno je omogućiti definisanje međusobnih odnosa između elemenata u istoj grupi. Sintaksa koju predlaže ovaj rad data je u nastavku.

Svi elementi nekog XML dokumenta mogu sadržati atribut *possibility* koji opisuje verovatnoću njegovog pojavljivanja i uzima vrednost iz intervala [0,1]. Ukoliko ovaj atribut nije definisan u elementu podrazumevana vrednost je 1. Grupisanje elemenata obavlja se pomoću elementa <*group*>, a svi elementi obuhvaćeni ovim elementom pripadaju istoj grupi. *Group* element mora sadržati atribut *dependence* koji opisuje tip povezanosti među elementima u grupi. Dozvoljene vrednosti su AND (konjunkcija), NAND (negirano i), XAND (kontradikcija), XNAND (tautologija), OR (disjunkcija), NOR (negirano ili), XOR (ekskluzivna disjunkcija), XNOR (ako i samo ako). Logičke operacije su date u tabeli 4.1, pri čemu je ⊤ tačno, a ⊥ netačno.

p	q	AND	NAND	XAND	XNAND	OR	NOR	XOR	XNOR
⊤	⊤	⊤	⊥	⊥	⊤	⊤	⊥	⊥	⊤
⊤	⊥	⊥	⊤	⊥	⊤	⊤	⊥	⊤	⊥
⊥	⊤	⊥	⊤	⊥	⊤	⊤	⊥	⊤	⊥
⊥	⊥	⊥	⊤	⊥	⊤	⊥	⊤	⊥	⊤

Tabela 4.1. Logičke operacije.

Jedan od načina za implementaciju navedenog predloga je upotreba XML šema. Na početku je potrebno definisati atribut *possibility* koji opisuje **verovatnoću pojavljivanja elementa** u nekom XML dokumentu. Atribut se može definisati XSD dokumentom datom u listingu 4.10.

```
<xs:schema id="Possibility" xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:attribute name="possibility">
          <xs:simpleType>
                <xs:restriction base="xs:decimal">
                      <xs:minExclusive value="0" />
                      <xs:maxExclusive value="1" />
                </xs:restriction>
            </xs:simpleType>
    </xs:attribute>
</xs:schema>
```

Listing 4.10. XSD definicija atributa verovatnoće pojavljivanja – Possibility.xsd.

Dokument *Possibility.xsd* definiše atribut koji nosi naziv *possibility* i uzima vrednost iz intervala [0,1], na primer *possibility="0.7"*. Kada definisani element uključimo u šemu iz listinga 4.7, dobija se nova *fuzzyType* definicija data u listingu 4.11.

```
<xs:schema id="Fuzzy2"
    elementFormDefault="qualified"
    xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:include schemaLocation ="Possibility.xsd"/>

    <!-- Fazi tip -->
    <xs:complexType name ="fuzzyType">
        <xs:choice minOccurs ="0" maxOccurs="1">
            <xs:element fer="fuzzyValue"/>
            <xs:element fer="fuzzyLabel"/>
            <xs:element fer="fuzzyCrisp"/>
        </xs:choice>

    </xs:complexType>

    <!-- Fazi vrednost -->
    <xs:element name="fuzzyValue">
        <xs:complexType>
            <xs:sequence>
               <xs:element name="fuzzy" type="fuzzy" nillable="true"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>

    <!-- Lingvisticka promenljiva -->
    <xs:element name="fuzzyLabel">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="fuzzy" type="xs:string" nillable="true"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>

    <!-- Diskretna vrednost -->
    <xs:element name="fuzzyCrisp">
        <xs:complexType>
            <xs:sequence>
               <xs:element name="fuzzy" type="xs:decimal" nillable="true"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>


    <!—Fazi skup -->
    <xs:complexType name ="fuzzy">
        <xs:sequence>
            <xs:element ref="function" minOccurs ="1"
            maxOccurs="unbounded"/>
        </xs:sequence>
        <xs:attribute name="name" type="xs:string"/>
        <xs:attribute ref="possibility"/>
    </xs:complexType>

    <!—Funkcija pripadnosti ili njen deo -->
    <xs:element name="function">
        <xs:complexType>
```

```
            <xs:simpleContent>
               <xs:extension base="xs:string">
                   <xs:attribute name="minValue" type="xs:string" />
                   <xs:attribute name="maxValue" type="xs:string" />
                </xs:extension>
            </xs:simpleContent>
        </xs:complexType>
    </xs:element>
</xs:schema>
```

Listing 4.11. XSD definicija fazi tipa XML elementa – Fuzzy2.xsd.

Prethodno definisani dokument proširuje fazi tip novim atributom *possibility*, koji definiše verovatnoću pojavljivanja elementa u XML dokumentu. Atribut je opcioni i u slučaju da nije naveden, podrazumevana vrednost je 1 (postoji 100%).

Drugi element koji je potrebno definisati je **element grupisanja** *group*, a njegova definicija pomoću XSD dokumenta data je u listingu 4.12.

```
<xs:schema id="Group" xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:include schemaLocation ="Fuzzy2.xsd"/>
    <xs:element name="group">
        <xs:complexType>
            <xs:sequence>
                <xs:element ref="fuzzyType" maxOccurs="unbounded"/>
            </xs:sequence>
            <xs:attribute name="dependence" use="required">
                <xs:simpleType>
                    <xs:restriction base="xsd:string">
                        <xs:enumeration value="AND" />
                        <xs:enumeration value="NAND" />
                        <xs:enumeration value="XAND" />
                        <xs:enumeration value="XNAND" />
                        <xs:enumeration value="OR" />
                        <xs:enumeration value="NOR" />
                        <xs:enumeration value="XOR" />
                        <xs:enumeration value="XNOR" />
                    </xs:restriction>
                </xs:simpleType>
            </xs:attribute>
        </xs:complexType>
    </xs:element>
</xs:schema>
```

Listing 4.12. XSD definicija elemenata grupisanja – Group.xsd.

Dokument *Group.xsd* definiše element grupisanja *group*, koji može sadržati sekvencu *fuzzyType* elemenata. Uključena je šema iz listinga 4.11, u kojoj je definisan *fuzzyType*, koji može sadržati atribut *possibility*. Elemenat *group* ima obavezan atribut *dependence*, koji opisuje tip logičke veze između njegovih pod-elemenata.

Na primer, ukoliko bi se XSD dokument iz listinga 4.3 prepravio tako da koristi mogućnost neodređenosti strukture fazi elemenata definisane u listingu 4.12, tada bi dokument izgledao kao u listingu 4.13.

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema id="Test"
    elementFormDefault="qualified"
    xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:include schemaLocation ="Group.xsd"/>
    <xs:element name="merenje">
        <xs:complexType>
            <xs:sequence>
              <xs:element name="naziv" type="xs:string"/>
              <xs:element name="lokacija" type="xs:int"/>
              <xs:element name="vreme" type="xs:dateTime"/>
              <xs:element name="temperatura">
                  <xs:element ref="group" use="required"/>
              </xs:element>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
</xs:schema>
```

Listing 4.13. Upotreba grupisanja fazi tipa u XSD dokumentima.

U prethodnom primeru dokument sadrži element tipa temperatura, koji ima obavezan pod-element tipa *group* i definisan je u *group.xsd* dokumentu. XML dokumenat koji bi odgovarao gore navedenoj šemi naveden je u listingu 4.14.

```
<merenje>
      <naziv>Meteorološka stanica Novi Sad</naziv>
      <lokacija>1</lokacija>
      <vreme>2011-09-04</vreme>
      <temperatura>
            <group dependence = "XOR">
                  <fuzzy name="hladno" possibility="0.7">
                        <function maxValue="10">1</function>
                        <function minValue="10" maxValue="20">
                              -1/(Max-Min)*x + Max/(Max-Min)
                        </function>
                  </fuzzy>
                  <fuzzy name="umereno" possibility="0.2">
                        <function minValue="10" maxValue="15">
                              1/(Max-Min)*x - Min/(Max-Min)
                        </function>
                        <function minValue="15" maxValue="20">
                              -1/(Max-Min)*x + Max/(Max-Min)
                        </function>
                  </fuzzy>
                  <fuzzy name="toplo" possibility="0.9">
                        <function minValue="10" maxValue="20">
```

```
                        1/(Max-Min)*x - Min/(Max-Min)
                    </function>
                    <function minValue="20">1</function>
                </fuzzy>
            </group>
        </temperatura>
</merenje>
```

Listing 4.14. Primer neodređenosti u strukturi fazi XML dokumenta.

Element *temperatura*, sadrži pod-element grupisanja *group*, koji grupiše pod-elemente logičkom vezom tipa XOR (ekskluzivno ILI). Svaka funkcija pripadnosti sada sadrži dodatni atribut *possibility*, koji definiše verovatnoću pojavljivanja elementa u XML dokumentu.

Glava V

FAZI XQUERY UPITI SA PRIORITETIMA

Poglavlje daje razloge za proširenje standardne XQuery upitne sintakse fazi elementima. Definisana je fazi XQuery upitna sintaksa, koja predstavlja proširenje standardne XQuery sintakse. Nakon toga je fazi XQuery proširen sa prioritetima i pragovima zadovoljenja, čija upotreba je omogućena zahvaljujući GPFCSP-u. Sintaksa je precizno opisana pomoću EBNF gramatike. Na kraju poglavlja date su preporuke za obradu upita u slučaju pojave Null vrednosti.

Definisanje sintakse fazi XML jezika predstavlja značajan korak u njegovom razvoju. Da bi takav jezik bio praktično upotrebljiv poželjna je i podrška dodatnih tehnologija, kao na primer XML upitnih sintaksi. XML upitne sintakse omogućavaju jednostavno rukovanje XML dokumentima i jednako su zaslužne za uspeh XML standarda koliko i sam jezik. Da bi upitna sintaksa bila u mogućnosti da podrži rad sa fazi XML elementima, mora biti definisana tako da ih jednostavno može prepoznati i obraditi. Jednostavan način za definisanje fazi XML upitne sintakse je proširenje neke standardne upitne sintakse fazi elementima. Najznačajniji primer XML upitne sintakse je **XQuery** upitna sintaksa. Proširenje XQuery upitne sintakse fazi elementima omogućuje upotrebu fazi vrednosti u definisanju upita. Takva sintaksa naziva se **fazi XQuery upitna sintaksa**. Upiti definisani pomoću nje mogu da daju mnogo preciznija rešenja.

Radi lakšeg razumevanja prednosti navedenog pristupa, u nastavku je naveden jednostavan primer analize rezultata meteoroloških merenja.

Primer 5.1: Pretpostavimo da postoji standardni XML dokument, dat u listingu 5.1, koji sadrži rezultate merenja meteoroloških stanica sa četiri lokacije. Zadatak bi mogao da glasi: "*Odrediti gradove u kojima je u trenutku merenja bilo lepo vreme*".

```
<merenje naziv="Primer_1" vreme="2010-06-02 12:00">
      <lokacija naziv="Beograd">
            <temperatura>21.4</temperatura>
            <vlaznost>0.44</vlaznost>
            <pritisak>1005</pritisak>
            <vetar>
                  <smer>SI</smer>
                  <brzina>7.4</brzina>
            </vetar>
      </lokacija>
      <lokacija naziv="London">
```

```
            <temperatura>8.2</temperatura>
            <vlaznost>0.86</vlaznost>
            <pritisak>1001</pritisak>
            <vetar>
                  <smer>S</smer>
                  <brzina>20.2</brzina>
            </vetar>
      </lokacija>
      <lokacija naziv="Pariz">
            <temperatura>22.0</temperatura>
            <vlaznost>0.84</vlaznost>
            <pritisak>988</pritisak>
            <vetar>
                  <smer>I</smer>
                  <brzina>28.2</brzina>
            </vetar>
      </lokacija>
      <lokacija naziv="Moskva">
            <temperatura>2.2</temperatura>
            <vlaznost>0.74</vlaznost>
            <pritisak>1010</pritisak>
            <vetar>
                  <smer>JI</smer>
                  <brzina>24.9</brzina>
            </vetar>
      </lokacija>
</merenje>
```

Listing 5.1. XML dokument.

Ovaj problem se može rešiti postavljanjem XQuery upita sa odgovarajućim uslovima. Primer takvog upita naveden je u listingu 5.2.

```
<rezultat>
{
      for $item in doc("merenje.xml")/merenje/lokacija
      let $naziv := $item/@naziv
      let $temperatura := $item/temperatura/text()
      let $vlaznost := $item/vlaznost/text()
      let $pritisak :=$item/pritisak/text()
      let $smer :=$item/vetar/smer/text()
```

```
        let $brzina:=$item/vetar/brzina/text()
        where $temperatura > 17 and $temperatura < 25
              and $vlaznost > 0.45 and $vlaznost < 0.85
              and $brzina < 30.0
        order by $naziv
        return
              <lokacija naziv="{$naziv}">
              <temperatura> {$temperatura} </temperatura>
              <vlaznost>{$vlaznost}</vlaznost>
              <pritisak>{$pritisak}</pritisak>
              <vetar>
              <smer>{$smer}</smer>
              <brzina>{$brzina}</brzina>
              </vetar>
              </lokacija>
}
</rezultat>
```

Listing 5.2. Standardni XQuery upit.

Kada se upit iz listing 5.2 postavi nad XML dokumentom iz listinga 5.1, dobije se sledeći rezultat, listing 5.3.

```
<rezultat>
      <lokacija naziv="Pariz">
      <temperatura>22.0</temperatura>
      <vlaznost>0.84</vlaznost>
      <pritisak>988</pritisak>
      <vetar>
            <smer>I</smer>
            <brzina>28.2</brzina>
      </vetar>
      </lokacija>
</rezultat>
```

Listing 5.3. Rezultat koji vraća standardni XQuery upit.

Zanimljivo je da kao dobro rešenje nije prošao Beograd, prikazan u listingu 5.4, koji realno ima bolje rezultate od Pariza. Razlog za njegovo izbacivanje je neodgovarajuća vlažnost, koja je strogo ograničena uslovom *$vlaznost > 0.45*.

```
<lokacija naziv="Beograd">
	<temperatura>21.4</temperatura>
	<vlaznost>0.44</vlaznost>
	<pritisak>1005</pritisak>
	<vetar>
		<smer>SI</smer>
		<brzina>7.4</brzina>
	</vetar>
</lokacija>
```

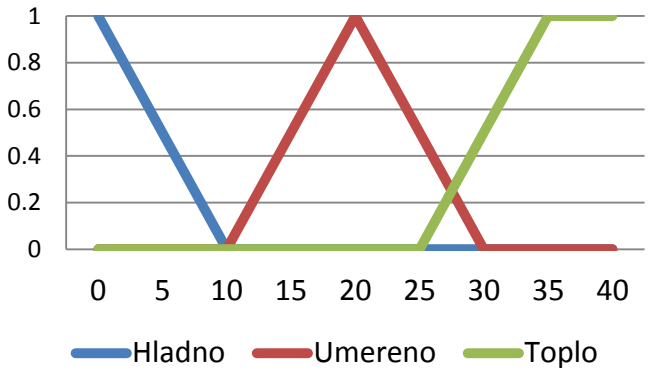
Listing 5.4. Element koji nije zadovoljio uslove.

Problem kod standardnog XQuery upita je potreba za zadavanjem preciznih uslova upita. Na taj način se iz krajnjeg rezultata mogu izostaviti dobra rešenja. Bolji način za određivanje skupa rezultata ostvaruje se upotrebom fazi vrednosti u definisanju uslova upita. Primer fazi XQuery upita prikazuje listing 5.5.

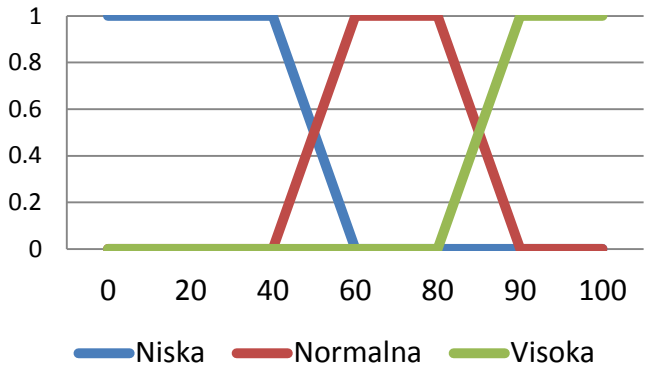
```
<rezultat>
{
	for $item in doc("merenje.xml")/merenje/lokacija
	let $naziv := $item/@naziv
	let $temperatura := $item/temperatura/text()
	let $vlaznost := $item/vlaznost/text()
	let $pritisak :=$item/pritisak/text()
	let $smer :=$item/vetar/smer/text()
	let $brzina:=$item/vetar/brzina/text()
	where $temperatura = #umereno
		and $vlaznost = #normalna
		and $brzina < #jak
	order by $naziv
	return
		<lokacija naziv="{$naziv}"/>
		<temperatura>{$temperatura}</temperatura>
		<vlaznost>{$vlaznost}</vlaznost>
		<pritisak>{$pritisak}</pritisak>
		<vetar>
			<smer>{$smer}</smer>
			<brzina>{$brzina}</brzina>
		</vetar>
		</lokacija>
}
</rezultat>
```

Listing 5.5. Fazi XQuery upit.

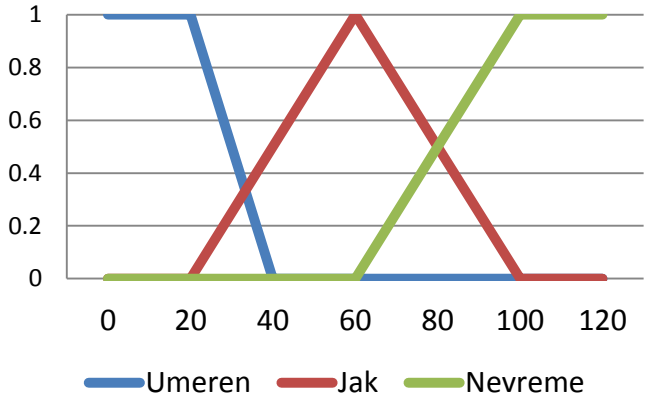
Pre pokretanja fazi XQuery upita potrebno je definisati funkcije pripadnosti za fazi lingvističke promenljive koje se sada koriste u upitu. Dijagrami funkcija pripadnosti za navedene fazi vrednosti prikazani su na slikama 5.1, 5.2 i 5.3.



Slika 5.1. Funkcije pripadnosti fazi skupa temperatura.



Slika 5.2. Funkcije pripadnosti fazi skupa vlažnost vazduha.



Slika 5.3. Funkcije pripadnosti fazi skupa brzina vetra.

Rezultat koji vraća upit iz listinga 5.5, dat je u listingu 5.6. Ovaj put je i Beograd prihvaćen kao zadovoljavajuće rešenje. To znači da je rezultat dobijen na ovaj način precizniji u odnosu na rezultat dobijen standardnim XQuery upitom.

```
<rezultat>
	<lokacija naziv="Beograd">
		<temperatura>21.4</temperatura>
		<vlaznost>0.44</vlaznost>
		<pritisak>1005</pritisak>
		<vetar>
			<smer>SI</smer>
			<brzina>7.4</brzina>
		</vetar>
	</lokacija>
	<lokacija naziv="Pariz">
		<temperatura>22.0</temperatura>
		<vlaznost>0.50</vlaznost>
		<pritisak>988</pritisak>
		<vetar>
			<smer>I</smer>
			<brzina>28.2</brzina>
		</vetar>
	</lokacija>
</rezultat>
```

Listing 5.6. Rezultat koji vraća fazi XQuery upit.

5.1 Prioriteti i pragovi zadovoljenja

U realnom svetu nemaju svi parametri nekog objekta isti uticaj na donošenje krajnjeg zaključka o njemu. Često nije dovoljno podeliti parametre na bitne i one manje bitne, nego je poželjno definisati stepen uticaja na krajnji rezultat, koji bi se izražavao u procentima. Tako se za svaki parametar precizno može odrediti njegov **prioritet** (*priority*) u donošenju zaključka. Ponekad parametri, posebno oni manje bitni (jer je njihov uticaj na sam rezultat uglavnom zanemarljiv), imaju granične vrednosti koje ne smeju biti probijene. Na primer, ukoliko je vlažnost vazduha preko 95%, bez obzira što je toplo i ne duva vetar, ne može se reći da je vreme lepo jer verovatno pada kiša. Granica ispod koje ne sme da se ide, naziva se **prag zadovoljenja** (*threshold*) i poput prioriteta izražava se u procentima.

Da bi se rešili navedeni problemi uvodi se GPFCSP koji omogućava upotrebu prioriteta i pragova zadovoljenja u fazi XQuery upitima. Pronalaženje rešenja koje zadovoljava sva ograničenja za optimalno vreme vrši se pomoću CSP-a. Kada CSP ima vrednost iz intervala [0,1], tada može da se upotrebi fazi logika, što daje FCSP. Pošto su neka ograničenja važnija od drugih (nemaju isti prioritet), FCSP se može proširi prioritetima i dobija se PFCSP. Međutim, PFCSP sistemi dozvoljavaju samo korišćenje konjunkcije, tačnije logičkih operatora tipa I. Iz tog razloga je neophodno generalizovati PFCSP sisteme

kako bi se omogućilo korišćenje disjunkcije i negiranje. Rezultat ove generalizacije je GPFCSP, čime je obezbeđena formalna osnova za upotrebu prioriteta u ograničenjima, a samim tim i razvoj fazi jezika sa prioritetima.

Stepen zadovoljenja ograničenja izračunava se upotrebom t-konormi Sp, gde je $S_P\left(\mu_{R_x^f}(\vartheta), 1-\rho\left(R_x^f\right)\right)$. ϑ je posmatrana vrednost, sa stepenom pripadanja $\mu_{R_x^f}(\vartheta)$ i prioritetom $\rho\left(R_x^f\right)$. Globalni stepen zadovoljenja ograničenja α u slučaju konjunktivne zavisnosti izračunava se kao t-norma T_L, a u slučaju disjunktivne zavisnosti kao S_L dualna t-konorma od T_L.

Na primer, formula za izračunavanje stepena zadovoljenja sa tri skupa ograničenja povezana na sledeći način $(R_1^f \wedge R_2^f) \vee R_3^f$, izgledala bi:

$$\alpha = S_L\left(T_L\left(S_P\left(\mu_{R_1^f}(\vartheta), 1-\rho\left(R_1^f\right)\right), S_P\left(\mu_{R_2^f}(\vartheta), 1-\rho\left(R_2^f\right)\right)\right), S_P\left(\mu_{R_3^f}(\vartheta), 1-\rho\left(R_3^f\right)\right)\right),$$

formula se dalje može rešavati:

$$S_P(x,y) = x + y - xy$$
$$T_L(x,y) = max(x + y - 1, 0)$$
$$S_L(x,y) = min\ (x + y, 1)$$

Primenom navedenih ideja na upit iz listinga 5.5, može se dodatno uticati na kvalitet rešenja. Proširenjem navedenog fazi XQuery upita prioritetima i pragovima zadovoljenja dobijamo prioritizovani fazi XQuery upit, koji je dat u listingu 5.7.

```
<rezultat>
{
      for $item in /merenje/lokacija
            let $naziv:=$item/@naziv
            let $temperatura:=$item/temperatura/text()
            let $vlaznost:=$item/vlaznost/text()
            let $pritisak:=$item/pritisak/text()
            let $pravac:=$item/vetar/pravac/text()
            let $brzina:=$item/vetar/brzina/text()
            let $oblacnost:=$item/oblacnost/text()
      where $temperatura = #optimalna_temperatura[P0.8,T0.3] AND
            $vlaznost = #optimalna_vlaznost[P0.3] AND
            $brzina = #optimalna_brzina_vetra[P0.5,T0.2]
      order by $naziv
      return
      <lokacija naziv="{$naziv}">
```

```
            <temperatura>{$temperatura}</temperatura>
            <vlaznost>{$vlaznost}</vlaznost>
            <pritisak>{$pritisak}</pritisak>
            <vetar>
                  <pravac>{$pravac}</pravac>
                  <brzina>{$brzina}</brzina>
            </vetar>
            <oblacnost>{$oblacnost}</oblacnost>
      </lokacija>
}
</rezultat>
```

Listing 5.7. Primer prioritizovanog fazi XQuery upita.

Prioriteti i prag zadovoljenja navode se neposredno posle fazi vrednosti u uglastim zagradama [*Pvrednost*, *Tvrednost*], a odnose se na samu fazi vrednost. *Pvrednost* i *Tvrednost* mogu sadržati vrednosti iz intervala [0,1]. Ako prioritet za neku fazi vrednost nije naveden, podrazumevana vrednost je 1. Ukoliko ne postoji nikakvo ograničenje, prag zadovoljenja se ne navodi, a podrazumevana vrednost je 0.

Fazi XQuery upit iz listinga 5.7 može se tumačiti na sledeći način. Uslov *#optimalna_temperatura[P0.8,T0.3]* se odnosi na fazi lingvističku promenljivu koja ima prioritet 80% i prag zadovoljenja 30%. Uslovi *#optimalna_vlaznost[P0.3]* ima prioritet 30% i prag zadovoljenja 0%, dok *#optimalna_brzina_vetra[P0.5,T0.2]* ima prioritet 50% i prag zadovoljenja 20%.

Upotrebu prioriteta najlakše je objasniti na praktičnom primeru.

Primer 5.2: Neka tekst zadatka glasi kao u prethodnom primeru, "*Odrediti gradove u kojima je u trenutku merenja bilo lepo vreme*", i neka se definisani upit izvršava nad malo izmenjenim XML dokumentom navedenim u listingu 5.8.

```
<merenje naziv="Primer_2" vreme="2013-06-29 12:00">
    <lokacija naziv="Beograd">
        <temperatura>21.4</temperatura>
        <vlaznost>0.44</vlaznost>
        <pritisak>1005</pritisak>
        <vetar>
            <pravac>SI</pravac>
            <brzina>1.4</brzina>
        </vetar>
        <oblacnost>0.10</oblacnost>
    </lokacija>
    <lokacija naziv="London">
        <temperatura>8.2</temperatura>
```

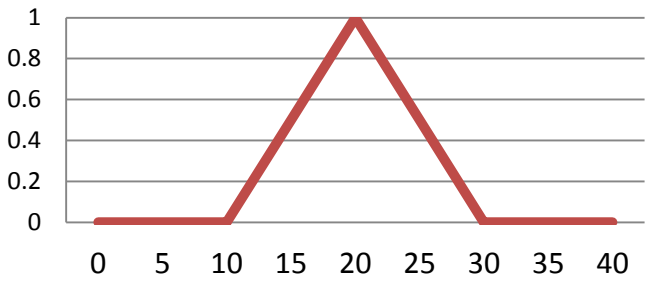
```
        <vlaznost>0.675</vlaznost>
        <pritisak>1001</pritisak>
        <vetar>
            <pravac>S</ pravac>
            <brzina>16.2</ brzina>
        </vetar>
        <oblacnost>0.90</oblacnost>
    </lokacija>
    <lokacija naziv="Pariz">
        <temperatura>11.0</temperatura>
        <vlaznost>0.54</vlaznost>
        <pritisak>988</pritisak>
        <vetar>
            <pravac>I</pravac>
            <brzina>11</brzina>
        </vetar>
        <oblacnost>0.47</oblacnost>
    </lokacija>
    <lokacija naziv="Moskva">
        <temperatura>6.2</temperatura>
        <vlaznost>0.75</vlaznost>
        <pritisak>1010</pritisak>
        <vetar>
            <pravac>S</pravac>
            <brzina>17</brzina>
        </vetar>
        <oblacnost>0.30</oblacnost>
    </lokacija>
</merenje>
```

Listing 5.8. Testni XML dokument.

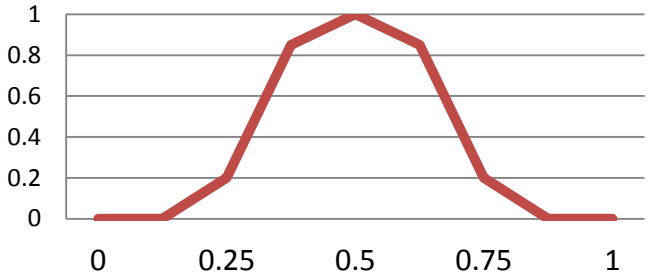
Zainteresovani smo za sledeći skup promenljivih i njihove domene:

- Promenljiva X_1 predstavlja temperaturu i ima domen $d_1 = (-\infty, +\infty)$
- Promenljiva X_2 predstavlja vlažnost vazduha i ima domen $d_2 = [0,1]$
- Promenljiva X_3 predstavlja brzinu vetra i ima domen $d_3 = [0, +\infty)$

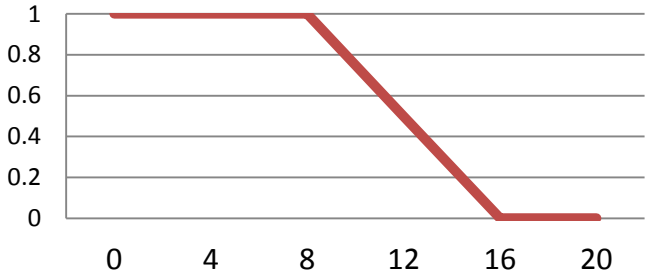
Pretpostavimo da je dat skup ograničenja $R_1^f = "optimalna\ temperatura"$, $R_2^f = "optimalna\ vlaznost"$ i $R_3^f = "optimalna\ brzina\ vetra"$. Data ograničenja mogu se definisati kao lingvističke fazi promenljive *#optimalna_temperatura*, *#optimala_vlaznost* i *#optimalna_brzina_vetra*, a njihove funkcije pripadnosti odgovaraju funkcijama sa slika 5.4, 5.5 i 5.6.



Slika 5.4. Funkcija pripadnosti – Optimalna temperatura.



Slika 5.5. Funkcija pripadnosti – Optimalna vlažnost.



Slika 5.6. Funkcija pripadnosti – Optimalne brzine vetra.

Globalni stepen zadovoljenja ograničenja za konkretni primer, može se izračunati pomoću sledeće jednačine.

$$\alpha = T_L\left(T_L\left(S_P\left(\mu_{R_1^f}(\vartheta), 1-\rho\left(R_1^f\right)\right), S_P\left(\mu_{R_2^f}(\vartheta), 1-\rho\left(R_2^f\right)\right)\right), S_P\left(\mu_{R_3^f}(\vartheta), 1-\rho\left(R_3^f\right)\right)\right)$$

U prethodnoj jednačini $S_P(x,y)$ je t-konorma, a $T_L(x,y)$ je t-norma, i za njih važe sledeće jednačine:

$$S_P(x,y) = x + y - xy$$
$$T_L(x,y) = max(x + y - 1, 0)$$

Za svaku vrednost ϑ izračunava se njen stepen pripadanja $\mu_{R_x^f}(\vartheta)$ koji se dobija kao presek funkcije pripadnosti i trenutne vrednosti. Vrednosti za ovaj primer se mogu videti u tabeli 5.1.

ϑ	X_1-Temper.	$\mu_{R_1^f}(\vartheta)$	X_2-Vlažnost	$\mu_{R_2^f}(\vartheta)$	X_3-Vetar	$\mu_{R_3^f}(\vartheta)$
Beograd	21.4	0.86	0.44	0.9328	1.4	1
London	8.2	0	0.675	0.85	16.2	0
Pariz	11.0	0.1	0.54	0.952	11	0.63
Moskva	6.2	0	0.75	0.2	17	0

Tabela 5.1. Proračuni stepena pripadanja za vrednosti iz primera.

Dalje, prioriteti $\rho(R_x^f)$ koji su dati u upitu iz listinga 5.8 imaju vrednosti 0.8 za temperaturu, 0.3 za vlažnost i 0.5 za brzinu vetra. Ako ove vrednosti uključimo u jednačinu za izračunavanje globalnog stepena zadovoljenja ograničenja α, dobijamo sledeće rezultate.

$$\alpha_{Beograd} = T_L\left(T_L\left(S_P(0.86, 1-0.8), S_P(0.9328, 1-0.3)\right), S_P(1, 1-0.5)\right)$$
$$= T_L(T_L(0.888, 0.97984), 1) = T_L(0.86784, 1) = 0.86784$$

$$\alpha_{London} = T_L\left(T_L\left(S_P(0, 1-0.8), S_P(0.85, 1-0.3)\right), S_P(0, 1-0.5)\right)$$
$$= T_L(T_L(0, 0.955), 0) = T_L(0, 0) = 0$$

$$\alpha_{Pariz} = T_L\left(T_L\left(S_P(0.1, 1-0.8), S_P(0.952, 1-0.3)\right), S_P(0.63, 1-0.5)\right)$$
$$= T_L(T_L(0.28, 0.9856), 0.815) = T_L(0.2656, 0.815) = 0.0806$$

$$\alpha_{Moskva} = T_L\left(T_L\left(S_P(0, 1-0.8), S_P(0.2, 1-0.3)\right), S_P(0, 1-0.5)\right)$$
$$= T_L(T_L(0.2, 0.76), 0) = T_L(0, 0) = 0$$

Može se videti da Beograd i Pariz imaju stepen pripadanja veći od 0, dok Pariz ne ispunjava uslov praga zadovoljenja za optimalnu temperaturu (ima 0.28, a prag je 0.5). Krajnji rezultat prikazan je u listingu 5.9.

```
<rezultat>
      <lokacija naziv="Beograd">
            <temperatura>21.4</temperatura>
            <vlaznost>0.44</vlaznost>
            <pritisak>1005</pritisak>
            <vetar>
                  <pravac>SI</pravac>
                  <brzina>7.4</brzina>
            </vetar>
            <oblacnost>0.10</oblacnost>
      </lokacija>
</rezultat>
```

Listing 5.9. Rezultat prioritizovanog fazi XQuery upita.

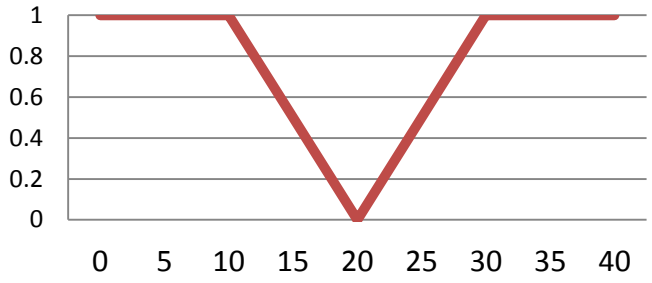
Bez prioriteta i pragova zadovoljenja oba rezultata (Pariz, Beograd) bi bila prihvatljiva. Ovim je pokazano da se sa prioritizovanim fazi XQuery upitom dobilo preciznije rešenje.

Primer 5.3: Prethodni primer pokazao je upotrebu konjunktivnih zavisnosti u povezivanju skupova ograničenja, za prikaz veza disjunktivnih zavisnosti i načina izračunavanja potrebno je malo izmeniti tekst zadatka iz primera 5.1 i 5.2. Novi zadatak mogao bi da glasi: "*Odrediti gradove u kojima je u trenutku merenja bilo loše vreme*". Da bi na nekoj lokaciji bilo loše vreme dovoljno je da bar jedan od parametara koji se posmatraju zadovolji uslov za loše vreme. Primer prioritizovanog fazi XQuery upita koji rešava ovaj zadatak dat je u listingu 5.10.

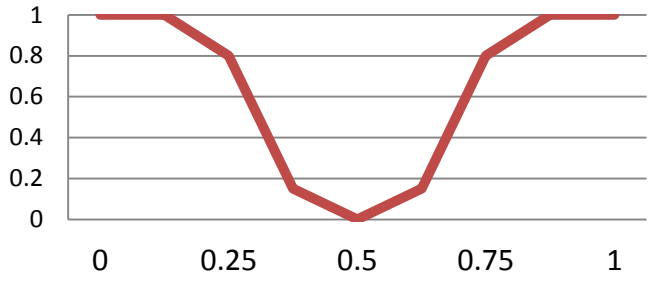
```
<rezultat>
{
      for $item in /merenje/lokacija
            let $naziv:=$item/@naziv
            let $temperatura:=$item/temperatura/text()
            let $vlaznost:=$item/vlaznost/text()
            let $pritisak:=$item/pritisak/text()
            let $pravac:=$item/vetar/pravac/text()
            let $brzina:=$item/vetar/brzina/text()
            let $oblacnost:=$item/oblacnost/text()
      where $temperatura = #losa_temperatura[P0.8,T0.3] OR
            $vlaznost = #losa_vlaznost[P0.3] OR
            $brzina = #velika_brzina_vetra[P0.5,T0.2]
      order by $naziv
      return
      <lokacija naziv="{$naziv}">
            <temperatura>{$temperatura}</temperatura>
            <vlaznost>{$vlaznost}</vlaznost>
            <pritisak>{$pritisak}</pritisak>
            <vetar>
                  <pravac>{$pravac}</pravac>
                  <brzina>{$brzina}</brzina>
            </vetar>
            <oblacnost>{$oblacnost}</oblacnost>
      </lokacija>
}
</rezultat>
```

Listing 5.10. Primer prioritizovanog fazi XQuery upita.

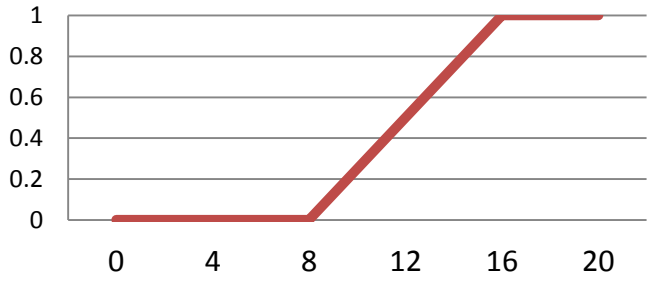
Funkcije pripadnosti navedene u upitu predstavljaju negaciju funkcija iz prethodnog primera i izgledaju kao na slikama 5.7, 5.8 i 5.9.



Slika 5.7. Funkcija pripadnosti – Loša temperatura.



Slika 5.8. Funkcija pripadnosti – Loša vlažnost.



Slika 5.9. Funkcija pripadnosti – Velika brzina vetra.

Takođe, primer se može iskoristiti za prikaz obrade XML dokumenata koji sadrže fazi vrednosti. Zbog toga će se kao testni dokumenat koristiti fazi XML dokument mešovitog sadržaja, dat u listingu 5.11. Navedeni dokumenat pored diskretnih vrednosti sadrži fazi funkcije pripadnosti i fazi lingvističke promenljive.

```
<merenje naziv="Primer_3" vreme="2013-06-30 12:00">
    <lokacija naziv="Beograd">
        <temperatura>
            <fuzzy>#umereno<fuzzy>
```

```
    </temperatura>
    <vlaznost>0.44</vlaznost>
    <pritisak>1005</pritisak>
    <vetar>
        <pravac>SI</pravac>
        <brzina>1.4</brzina>
    </vetar>
    <oblacnost>0.10</oblacnost>
</lokacija>
<lokacija naziv="London">
    <temperatura>8.2</temperatura>
    <vlaznost>0.675</vlaznost>
    <pritisak>1001</pritisak>
    <vetar>
        <pravac>S</ pravac>
        <brzina>16.2</ brzina>
    </vetar>
    <oblacnost>
        <fuzzy name="visoka">
              <function minValue="60" maxValue="80">
                    1/(Max-Min)*x - Min/(Max-Min)</function>
              <function minValue="80">1</function>
        </fuzzy>
    </oblacnost>
</lokacija>
<lokacija naziv="Pariz">
  <temperatura>11.0</temperatura>
    <vlaznost>0.54</vlaznost>
    <pritisak>988</pritisak>
    <vetar>
        <pravac>I</pravac>
        <brzina>11</brzina>
    </vetar>
    <oblacnost>0.47</oblacnost>
</lokacija>
<lokacija naziv="Moskva">
    <temperatura>
```

```
            <fuzzy>#hladno<fuzzy>
        </temperatura>
        <vlaznost>0.75</vlaznost>
        <pritisak>
            <fuzzy name="normalan">
                  <function minValue="970" maxValue="1000">
                        1/(Max-Min)*x - Min/(Max-Min)</function>
                  <function minValue="1000" maxValue="1030">
                        -1/(Max-Min)*x + Max/(Max-Min)</function>
            </fuzzy>
        </pritisak>
        <vetar>
            <pravac>S</pravac>
            <brzina>17</brzina>
        </vetar>
        <oblacnost>0.30</oblacnost>
    </lokacija>
</merenje>
```

Listing 5.11. Testni fazi XML dokument mešovitog sadržaja.

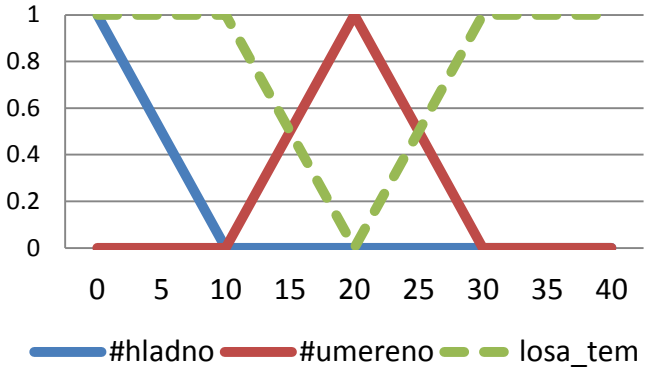
Globalni stepen zadovoljenja ograničenja za navedeni upit iz listinga 5.10 izračunava se pomoću jednačine:

$$\alpha = S_L\left(S_L\left(S_P\left(\mu_{R_1^f}(\vartheta), 1-\rho(R_1^f)\right), S_P\left(\mu_{R_2^f}(\vartheta), 1-\rho(R_2^f)\right)\right), S_P\left(\mu_{R_3^f}(\vartheta), 1-\rho(R_3^f)\right)\right)$$

, gde je $S_P(x,y)$ t-konorma, a $S_L(x,y)$ dualna t-konorma:

$$S_P(x,y) = x + y - xy$$
$$S_L(x,y) = min\,(x+y, 1)$$

Pošto nisu sve vrednosti iz listinga 5.11 diskretnog tipa, određivanje stepena pripadanja se dobija kao presek funkcije pripadnosti koja opisuje fazi element i funkcije pripadnosti koja opisuje uslov. Tako se za lingvističku promenljivu *#umereno*, koja opisuje temperaturu uzima presek sa uslovom *#losa_temperatura*, funkcije su date na slici 5.10. Fazi vrednost *#umereno* ima vrednosti veće od 0 nad intervalu (10, 30), a funkcija pripadnosti nad tim intervalom uzima vrednost iz intervala (0, 1). Fazi funkcija pripadnosti *#hladno* ima vrednost iznad nule za interval manje od 10, uslov *#losa_temperatura* nad navedenim intervalom ima vrednost 1.



Slika 5.10. Funkcije pripadnosti.

Stepeni pripadanja dati su tabeli 5.2.

ϑ	X_1-Temper.	$\mu_{R_1^f}(\vartheta)$	X_2-Vlažnost	$\mu_{R_2^f}(\vartheta)$	X_3-Vetar	$\mu_{R_3^f}(\vartheta)$
Beograd	**10 - 30**	**0 - 1**	0.44	0.0672	1.4	0
London	8.2	1	0.675	0.15	16.2	1
Pariz	11.0	0.9	0.54	0.048	11	0.37
Moskva	< **10**	**1**	0.75	0.8	17	1

Tabela 5.2. Vrednosti stepena pripadanja za dati primer.

Dalje izračunavamo globalne stepene zadovoljenja ograničenja za zadate lokacije :

$$\alpha_{Beograd\ max} = S_L\left(S_L\left(S_P(1,1-0.8), S_P(0.0672,1-0.3)\right), S_P(0,1-0.5)\right)$$
$$= S_L(S_L(1,0.72016),0.5) = S_L(1,0.5) = 0.5$$
$$\alpha_{Beograd\ min} = S_L\left(S_L\left(S_P(0,1-0.8), S_P(0.0672,1-0.3)\right), S_P(0,1-0.5)\right)$$
$$= S_L(S_L(0.2,0.72016),0.5) = S_L(0.92,0.5) = 0.5$$

$$\alpha_{London} = S_L\left(S_L\left(S_P(1,1-0.8), S_P(0.15,1-0.3)\right), S_P(1,1-0.5)\right)$$
$$= S_L(S_L(1,0.745),1) = S_L(1,1) = 1$$

$$\alpha_{Pariz} = S_L\left(S_L\left(S_P(0.9,1-0.8), S_P(0.048,1-0.3)\right), S_P(0.37,1-0.5)\right)$$
$$= S_L(S_L(0.92,0.7144),0.685) = S_L(1,0.685) = 0.685$$

$$\alpha_{Moskva\ min/max} = S_L\left(S_L\left(S_P(1,1-0.8), S_P(0.8,1-0.3)\right), S_P(1,1-0.5)\right)$$
$$= S_L(S_L(1,0.94),1) = S_L(1,1) = 1$$

London i Moskva u potpunosti zadovoljavaju navedena ograničenja, te se može reći da je tamo sigurno loše vreme, dok Beograd (u oba slučaja ima vrednost 50%) i Pariz (68,5%) delimično zadovoljavaju zadate kriterijume.

5.2 Fazi XQuery EBNF gramatika

Bekusova normalna forma ili **BNF** (Backus-Naur Form) je formalni meta jezik za predstavljanje kontekstno slobodnih gramatika, odnosno gramatika programskih jezika.

Proširena Bekusova normalna forma ili **EBNF** (Extended Backus-Naur Form) je meta jezik za predstavljanje kontekstno slobodnih gramatika i ima istu izražajnu moć kao BNF, samo je zapis lakši za razumevanje. Listing 5.12 daje potpunu definiciju sintakse fazi XQuery jezika definisanu EBNF gramatikom. Izrazi koji opisuju *FuzzyLiteral* (označeni crvenom bojom) predstavljaju proširenje u odnosu na standardnu XQuery sintaksu i omogućavaju upotrebu fazi promenljivih u XQuery upitima.

```
Module ::= VersionDecl? (MainModule | LibraryModule)

VersionDecl ::= <"xquery" "version"> StringLiteral ("encoding"
StringLiteral)? Separator

MainModule ::= Prolog QueryBody

LibraryModule ::= ModuleDecl Prolog

ModuleDecl ::= <"module" "namespace"> NCName "=" URILiteral Separator

Prolog ::= ((Setter | Import | NamespaceDecl | DefaultNamespaceDecl)
Separator)* ((VarDecl | FunctionDecl | OptionDecl) Separator)*

Setter ::= BoundarySpaceDecl | DefaultCollationDecl | BaseURIDecl |
ConstructionDecl | OrderingModeDecl | EmptyOrderDecl | CopyNamespacesDecl

Import ::= SchemaImport | ModuleImport

Separator ::= ";"

NamespaceDecl ::= <"declare" "namespace"> NCName "=" URILiteral

BoundarySpaceDecl ::= <"declare" "boundary-space"> ("preserve" | "strip")

DefaultNamespaceDecl ::= (<"declare" "default" "element"> | <"declare"
"default" "function">) "namespace" URILiteral

OptionDecl ::= <"declare" "option"> QName StringLiteral

OrderingModeDecl ::= <"declare" "ordering"> ("ordered" | "unordered")

EmptyOrderDecl ::= <"declare" "default" "order"> (<"empty" "greatest"> |
<"empty" "least">)

CopyNamespacesDecl ::= <"declare" "copy-namespaces"> PreserveMode ","
InheritMode

PreserveMode ::= "preserve" | "no-preserve"

InheritMode ::= "inherit" | "no-inherit"

DefaultCollationDecl ::= <"declare" "default" "collation"> URILiteral

BaseURIDecl ::= <"declare" "base-uri"> URILiteral

SchemaImport ::= <"import" "schema"> SchemaPrefix? URILiteral (<"at"
URILiteral> ("," URILiteral)*)?

SchemaPrefix ::= ("namespace" NCName "=") | (<"default" "element">
"namespace")

ModuleImport ::= <"import" "module"> ("namespace" NCName "=")? URILiteral
(<"at" URILiteral> ("," URILiteral)*)?

VarDecl ::= <"declare" "variable" "$"> VarName TypeDeclaration? ((":="
ExprSingle) | "external")

ConstructionDecl ::= <"declare" "construction"> ("preserve" | "strip")

FunctionDecl ::= <"declare" "function"> <QName "("> ParamList? (")" | (<")"
"as"> SequenceType)) (EnclosedExpr | "external")

ParamList ::= Param ("," Param)*

Param ::= "$" VarName TypeDeclaration?
```

EnclosedExpr ::= "{" Expr "}"

QueryBody ::= Expr

Expr ::= ExprSingle ("," ExprSingle)* **(FuzzyLiteral)?**

ExprSingle ::= FLWORExpr | QuantifiedExpr | TypeswitchExpr | IfExpr | OrExpr

FLWORExpr ::= (ForClause | LetClause)+ WhereClause? OrderByClause? "return" ExprSingle

ForClause ::= <"for" "$"> VarName TypeDeclaration? PositionalVar? "in" ExprSingle ("," "$" VarName TypeDeclaration? PositionalVar? "in" ExprSingle)*

PositionalVar ::= "at" "$" VarName

LetClause ::= <"let" "$"> VarName TypeDeclaration? ":=" ExprSingle ("," "$" VarName TypeDeclaration? ":=" ExprSingle)*

WhereClause ::= "where" ExprSingle

OrderByClause ::= (<"order" "by"> | <"stable" "order" "by">) OrderSpecList

OrderSpecList ::= OrderSpec ("," OrderSpec)*

OrderSpec ::= ExprSingle OrderModifier

OrderModifier ::= ("ascending" | "descending")? (<"empty" "greatest"> | <"empty" "least">)? ("collation" URILiteral)?

QuantifiedExpr ::= (<"some" "$"> | <"every" "$">) VarName TypeDeclaration? "in" ExprSingle ("," "$" VarName TypeDeclaration? "in" ExprSingle)* "satisfies" ExprSingle

TypeswitchExpr ::= <"typeswitch" "("> Expr ")" CaseClause+ "default" ("$" VarName)? "return" ExprSingle

CaseClause ::= "case" ("$" VarName "as")? SequenceType "return" ExprSingle

IfExpr ::= <"if" "("> Expr ")" "then" ExprSingle "else" ExprSingle

OrExpr ::= AndExpr ("or" AndExpr)*

AndExpr ::= ComparisonExpr ("and" ComparisonExpr)*

ComparisonExpr ::= RangeExpr ((ValueComp | GeneralComp | NodeComp) RangeExpr)?

RangeExpr ::= AdditiveExpr ("to" AdditiveExpr)?

AdditiveExpr ::= MultiplicativeExpr (("+" | "-") MultiplicativeExpr)*

MultiplicativeExpr ::= UnionExpr (("*" | "div" | "idiv" | "mod") UnionExpr)*

UnionExpr ::= IntersectExceptExpr (("union" | "|") IntersectExceptExpr)*

IntersectExceptExpr ::= InstanceofExpr (("intersect" | "except") InstanceofExpr)*

InstanceofExpr ::= TreatExpr (<"instance" "of"> SequenceType)?

TreatExpr ::= CastableExpr (<"treat" "as"> SequenceType)?

CastableExpr ::= CastExpr (<"castable" "as"> SingleType)?

CastExpr ::= UnaryExpr (<"cast" "as"> SingleType)?

UnaryExpr ::= ("-" | "+")* ValueExpr

ValueExpr ::= ValidateExpr | PathExpr | ExtensionExpr

GeneralComp ::= "=" | "!=" | "<" | "<=" | ">" | ">="

ValueComp ::= "eq" | "ne" | "lt" | "le" | "gt" | "ge"

NodeComp ::= "is" | "<<" | ">>"

ValidateExpr ::= (<"validate" "{"> | (<"validate" ValidationMode> "{")) Expr "}"

ExtensionExpr ::= Pragma+ "{" Expr? "}"

Pragma ::= "(#" S? QName PragmaContents "#)"

PragmaContents ::= (Char* - (Char* '#)' Char*))

PathExpr ::= ("/" RelativePathExpr?) | ("//" RelativePathExpr) | RelativePathExpr

RelativePathExpr ::= StepExpr (("/" | "//") StepExpr)*

StepExpr ::= AxisStep | FilterExpr

AxisStep ::= (ForwardStep | ReverseStep) PredicateList

ForwardStep ::= (ForwardAxis NodeTest) | AbbrevForwardStep

ForwardAxis ::= <"child" "::"> | <"descendant" "::"> | <"attribute" "::"> | <"self" "::"> | <"descendant-or-self" "::"> | <"following-sibling" "::"> | <"following" "::">

AbbrevForwardStep ::= "@"? NodeTest

ReverseStep ::= (ReverseAxis NodeTest) | AbbrevReverseStep

ReverseAxis ::= <"parent" "::"> | <"ancestor" "::"> | <"preceding-sibling" "::"> | <"preceding" "::"> | <"ancestor-or-self" "::">

AbbrevReverseStep ::= ".."

NodeTest ::= KindTest | NameTest

NameTest ::= QName | Wildcard

Wildcard ::= "*" | <NCName ":" "*"> | <"*" ":" NCName>

FilterExpr ::= PrimaryExpr PredicateList

PredicateList ::= Predicate*

Predicate ::= "[" Expr "]"

PrimaryExpr ::= Literal | VarRef | ParenthesizedExpr | ContextItemExpr | FunctionCall | Constructor | OrderedExpr | UnorderedExpr

Literal ::= NumericLiteral | StringLiteral

NumericLiteral ::= IntegerLiteral | DecimalLiteral | DoubleLiteral

VarRef ::= "$" VarName

ParenthesizedExpr ::= "(" Expr? ")"

ContextItemExpr ::= "."

OrderedExpr ::= <"ordered" "{"> Expr "}"

UnorderedExpr ::= <"unordered" "{"> Expr "}"

FunctionCall ::= <QName "("> (ExprSingle ("," ExprSingle)*)? ")"

Constructor ::= DirectConstructor | ComputedConstructor

DirectConstructor ::= DirElemConstructor | DirCommentConstructor | DirPIConstructor

DirElemConstructor ::= "<" QName DirAttributeList ("/>" | (">" DirElemContent* "</" QName S? ">"))

DirAttributeList ::= (S (QName S? "=" S? DirAttributeValue)?)*

DirAttributeValue ::= ('"' (EscapeQuot | QuotAttrValueContent)* '"') | ("'" (EscapeApos | AposAttrValueContent)* "'")

QuotAttrValueContent ::= QuotAttrContentChar | CommonContent

AposAttrValueContent ::= AposAttrContentChar | CommonContent

DirElemContent ::= DirectConstructor | ElementContentChar | CDataSection | CommonContent

CommonContent ::= PredefinedEntityRef | CharRef | "{{" | "}}" | EnclosedExpr

DirCommentConstructor ::= "<!--" DirCommentContents "-->"

DirCommentContents ::= ((Char - '-') | <'-' (Char - '-')>)*

DirPIConstructor ::= "<?" PITarget (S DirPIContents)? "?>"

DirPIContents ::= (Char* - (Char* '?>' Char*))

CDataSection ::= "<![CDATA[" CDataSectionContents "]]>"

CDataSectionContents ::= (Char* - (Char* ']]>' Char*))

ComputedConstructor ::= CompDocConstructor | CompElemConstructor | CompAttrConstructor | CompTextConstructor | CompCommentConstructor | CompPIConstructor

CompDocConstructor ::= <"document" "{"> Expr "}"

CompElemConstructor ::= (<"element" QName "{"> | (<"element" "{"> Expr "}" "{")) ContentExpr? "}"

ContentExpr ::= Expr

CompAttrConstructor ::= (<"attribute" QName "{"> | (<"attribute" "{"> Expr "}" "{")) Expr? "}"

CompTextConstructor ::= <"text" "{"> Expr "}"

CompCommentConstructor ::= <"comment" "{"> Expr "}"

CompPIConstructor ::= (<"processing-instruction" NCName "{"> | (<"processing-instruction" "{"> Expr "}" "{")) Expr? "}"

SingleType ::= AtomicType "?"?

TypeDeclaration ::= "as" SequenceType

SequenceType ::= (ItemType OccurrenceIndicator?) | <"void" "(" ")">

OccurrenceIndicator ::= "?" | "*" | "+"

ItemType ::= AtomicType | KindTest | <"item" "(" ")">

AtomicType ::= QName

KindTest ::= DocumentTest | ElementTest | AttributeTest | SchemaElementTest | SchemaAttributeTest | PITest | CommentTest | TextTest | AnyKindTest

AnyKindTest ::= <"node" "("> ")"

DocumentTest ::= <"document-node" "("> (ElementTest | SchemaElementTest)? ")"

TextTest ::= <"text" "("> ")"

CommentTest ::= <"comment" "("> ")"

PITest ::= <"processing-instruction" "("> (NCName | StringLiteral)? ")"

AttributeTest ::= <"attribute" "("> (AttribNameOrWildcard ("," TypeName)?)? ")"

AttribNameOrWildcard ::= AttributeName | "*"

```
SchemaAttributeTest ::= <"schema-attribute" "("> AttributeDeclaration ")"
AttributeDeclaration ::= AttributeName
ElementTest ::= <"element" "("> (ElementNameOrWildcard ("," TypeName
"?"?)?)?")"
ElementNameOrWildcard ::= ElementName | "*"
SchemaElementTest ::= <"schema-element" "("> ElementDeclaration ")"
ElementDeclaration ::= ElementName
AttributeName ::= QName
ElementName ::= QName
TypeName ::= QName
IntegerLiteral ::= Digits
DecimalLiteral ::= ("." Digits) | (Digits "." [0-9]*)
DoubleLiteral ::= (("." Digits) | (Digits ("." [0-9]*)?)) [eE] [+-]? Digits
URILiteral ::= StringLiteral
StringLiteral ::= ('"' (PredefinedEntityRef | CharRef | ('"' '"') | [^"&])*
'"') | ("'" (PredefinedEntityRef | CharRef | ("'" "'") | [^'&])* "'")
PITarget ::=
[http://www.w3.org/TR/REC-xml#NT-PITarget]XML
VarName ::= QName
ValidationMode ::= "lax" | "strict"
Digits ::= [0-9]+
PredefinedEntityRef ::= "&" ("lt" | "gt" | "amp" | "quot" | "apos") ";"
CharRef ::= [http://www.w3.org/TR/REC-xml#NT-CharRef]XML
EscapeQuot ::= '""'
EscapeApos ::= "''"
ElementContentChar ::= Char - [{}<&]
QuotAttrContentChar ::= Char - ["{}<&]
AposAttrContentChar ::= Char - ['{}<&]
Comment ::= "(:" (CommentContents | Comment)* ":)"
CommentContents ::= (Char+ - (Char* ':)' Char*))
NCName ::= [http://www.w3.org/TR/REC-xml-names/#NT-NCName]Names
S ::= [http://www.w3.org/TR/REC-xml#NT-S]XML
Char ::= [http://www.w3.org/TR/REC-xml#NT-Char]XML
FuzzyLiteral ::= "#" Literal (GPFCSPExpr)?
GPFCSPExpr ::= ("[" (PriorityExpr) ("," ThresholdExpr)? "]")?
ThresholdExpr ::= "T" PercentLiteral
PriorityExpr ::= "P" PercentLiteral
PercentLiteral ::= (ValueOne) | (ValueZero ("." ([0-9]*)? ([1-9]))?)
ValueOne :: = "1" | "1.0"
ValueZero ::= "0"
```

Listing 5.12. Fazi XQuery EBNF definicija sintakse.

Element *Expr* standardnog XQuery upitnog jezika proširen je opcionim elementom *FuzzyLiteral*, koji predstavlja fazi tip. Svaki element *FuzzyLiteral* počinje znakom #, nakon čega slede ime fazi promenljive i opcioni element *GPFCSPExpr*. Fazi promenljiva mora biti prethodno definisana kao fazi lingvistička promenljiva, na primer kao u listingu 4.5. *GPFCSPExpr* navodi prioritete i pragove zadovoljenja u uglastim zagradama i oni se odnose na samu fazi vrednosti. Prioritet *PriorityExpr* je obavezan element, a sastoji se od slova *P* i vrednosti u nastavku. Prag zadovoljenja *ThresholdExpr* je opcioni element i sastoji se od slova *V* i vrednosti u nastavku. Vrednost *PercentLiteral* predstavlja decimalnu vrednost iz intervala [0, 1]. Ukoliko fazi vrednost ne sadrži svoj prioritet, podrazumevana vrednost je 1. Ukoliko postoje ograničenja, a pragovi zadovoljenja nisu navedeni, podrazumevana vrednost praga zadovoljenja je 0.

5.3 Fazi XQuery upiti nad Null vrednostima

Prethodno poglavlje obrađuje mogućnost definisanja dva tipa Null vrednosti (*nepoznata* i *neprimenljiva*) u XML dokumentima. Za uvođenje Null vrednosti u fazi XQuery upite potrebno je detaljnije obraditi takve slučajeve i definisati određene preporuke koje će se koristiti u procesu implementacije.

Kako ne postoji mogućnost da se izmeni izvorni kod Microsoft SQL Servera, zbog njegove zatvorenosti, odabran je pristup da se ne implementira matematički formalizam nego da se pokuša da se u procesu obrade dobiju neki intuitivni rezultati. Za te intuitivne slučajeve definišemo sledeće preporuke.

Slučaj pojave Null vrednosti tipa **nepoznata** može se razbiti na dva pod-slučaja:

1. Postavljanje upita koji agregiraju podatke nad kolekcijom dokumenata slične sadržine.
2. Postavljanje ostalih pojedinačnih upita.

Pretpostavimo da postoji kolekcija dokumenata, koja sadrži dokumenta slične sadržine i neka se nad takvom kolekcijom izvršava fazi XQuery upit koji agregira neke podatke iz dokumenata. Pod kolekcijom slične sadržine može se smatrati skup XML dokumenata koji sadrži informacije o pojedinačnom učinku igrača na nekoj košarkaškoj utakmici. Svaki od navedenih dokumenata mora sadržati informacije kao što su procenti šuta, broj faula, broj skokova… Šta se dešava ako u nekim dokumentima iz kolekcije nedostaju određeni elementi (na primer nije izbrojan broj faulova nekog igrača na nekoj utakmici)? Takvi slučajevi predstavljaju pojavu Null vrednosti tipa *nepoznata*. Pristup rešavanju ovog problema koji bi podrazumevao izbacivanje kompletnog dokumenata koji sadrže nedostajuće elemente nije dobro rešenje. Bolji pristup je da se prilikom izračunavanja agregatnih funkcija (na primer prosečan broj skokova i faulova) eliminišu samo one vrednosti koje nedostaju (broj faulova).

Za drugi slučaj nedostajućeg elementa pretpostavimo da postoji upit koji ne obuhvata agregaciju elementa slične sadržine, već predstavlja pojedinačan upit nad nekim dokumentom. Na osnovu drugih elemenata iz tog dokumenta, u slučaju postojanja međuzavisnosti, potrebno je nadomestiti nedostajuću vrednost. Primer takvog upita mogao bi da bude određivanje nedostajućeg elementa minutaža, za određenog igrača na određenoj utakmici, na osnovu dostupnih elemenata kao što su broj poena, skokova, faulova… Ukoliko

je igrač na utakmici zabeležio velik broj poena, skokova…, može se pretpostaviti da je i minutaža bila veća. U ovom slučaju neće biti pokušana implementacija jer ne postoji mogućnost menjanja izvornog koda aplikacije Microsoft SQL Server.

Drugi slučaj pojave neodređene informacije predstavlja pojava Null vrednosti tipa **neprimenljiva.** Pretpostavimo da ne postoji procenat šuta za slobodna bacanja, jer ih igrač nije ni izvodio. To predstavlja klasičan primer pojave Null vrednosti tipa *neprimenljiva*. Kao i kod Null vrednosti tipa *nepoznata* i ovde možemo nezavisno razmatrati dva pomenuta slučaja. Ukoliko se radi o postavljanju upita koji agregiraju podatke nad kolekcijom dokumenata slične sadržine, tada je intuitivno potrebno izbaciti nedostajuću vrednost i agregaciju računati bez nje. Miscrosoft SQL Server radi na ovaj način, tako da nije potrebno posebno implementirati ovaj slučaj, ukoliko se implementacija bazira na upotrebi navedene baze podataka. Što se tiče drugog slučaja i obrade pojedinačnih upita, ne može se na osnovu vrednosti drugih elemenata nagovestiti vrednost koja ne postoji (tip vrednosti neprimenljiva baš označava da ta vrednost ne može da postoji). Miscrosoft SQL Server u ovom slučaju neće vratiti vrednost, što je opet u skladu sa intuitivnim pristupom i nije potrebna nikakva dodatna implementacija.

Za navedena rešenja nije potrebna izmena postojeće prioritizovane fazi XQuery sintakse, ali ih treba prihvatiti kako preporuke u procesu implementacije i izračunavanja stepena pripadnosti.

Glava VI

RAZVOJ APLIKACIJE

Na osnovu sintaksi definisanih u prethodnim poglavljima razvijen je softverski paket pod nazivom *Fuzzy XML Editor*. Poglavlje daje prikaz modelovanja aplikacije upotrebom UML dijagrama. Opisane su tehnologije korišćene u razvoju aplikacije. Na kraju poglavlja objašnjen je proces implementacije upotrebom WPF tehnologije i .NET faramework-a i dati delovi koda u XAML-a i C# programskom jeziku koji su bitni za razumevanje procesa implementacije.

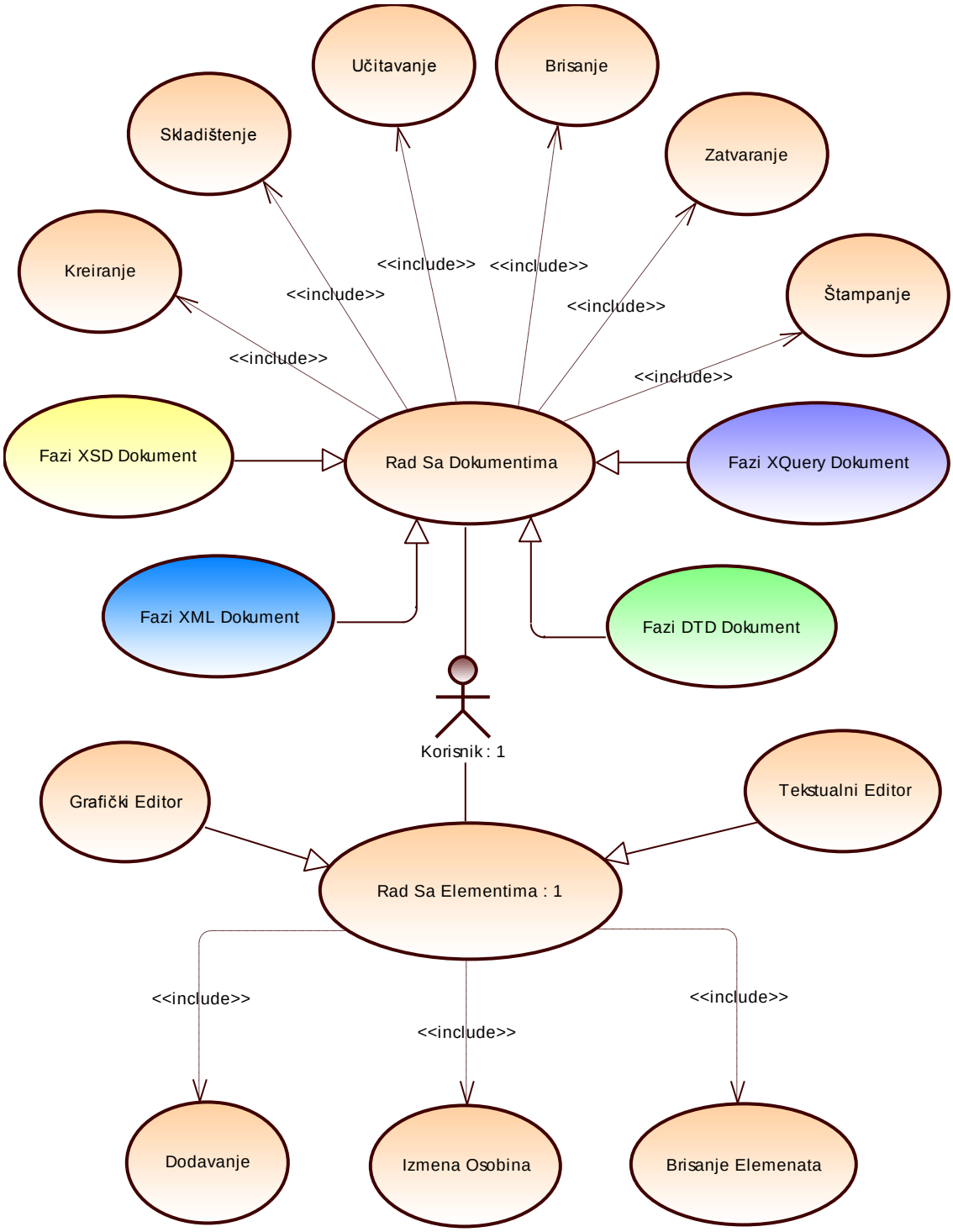
6.1. Modelovanje aplikacije

UML (Unified Modeling Language) je standardizovani opšte namenski jezik, pogodan za modelovanje u oblasti objektno orijentisanog softverskog inženjerstva. Upotrebom osnovnih UML dijagrama definisane su funkcionalnosti i dizajn aplikacije koja se kreira.

6.1.1 UML dijagrami

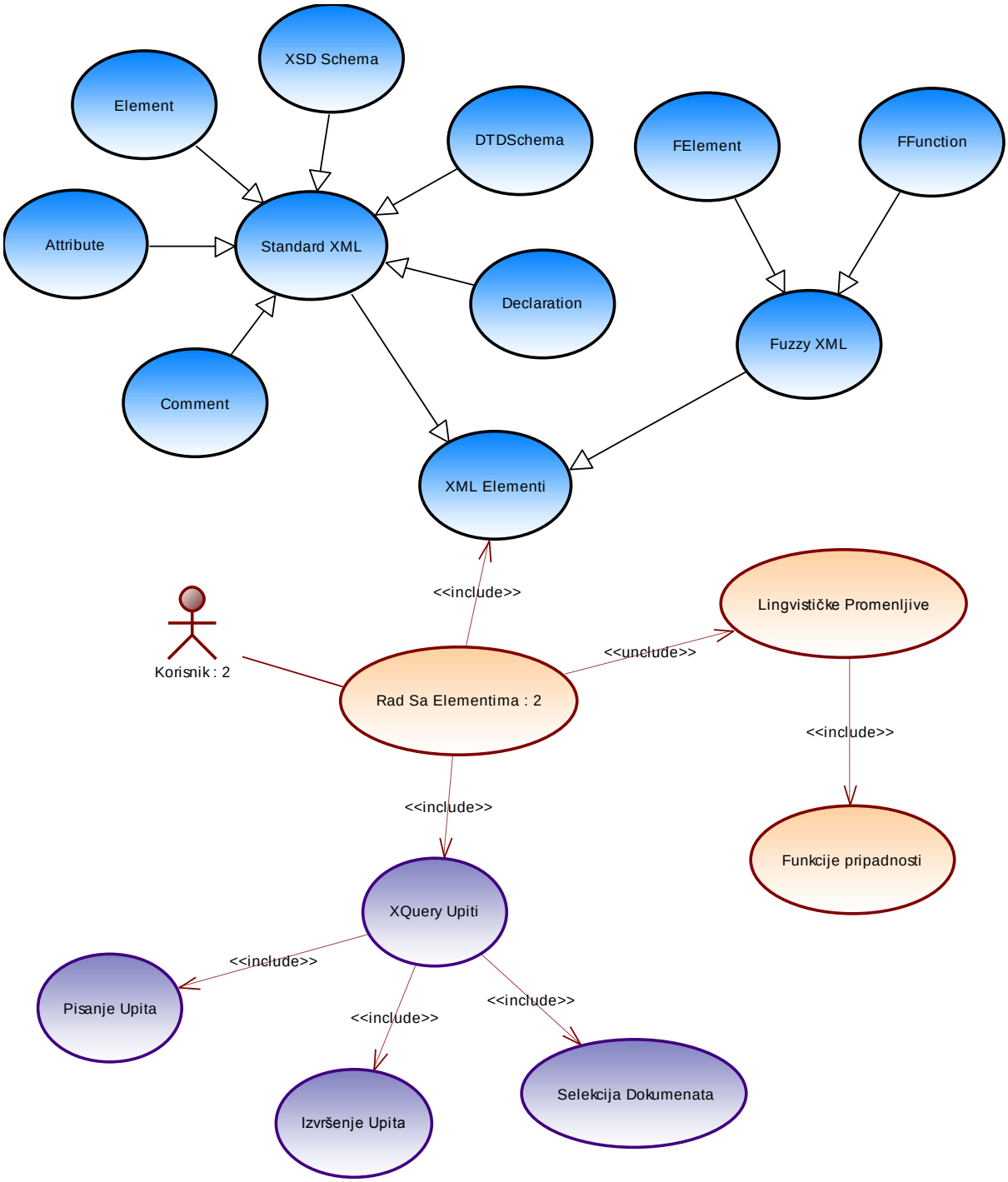
Dijagram slučajeva korišćenja (Use Case): Aplikacija bi trebalo da podrži slučajeve korišćenja prikazane na slikama 6.1, 6.2 i 6.3. Zbog obimnosti dijagrama prikazani su samo delovi koji su od suštinske važnosti za razumevanje softverskog paketa koji se razvija. Kompletna specifikacija priložena je na pratećem CD-u.

Slika 6.1 daje prikaz osnovnih funkcionalnosti alata i zajedno sa slikom 6.2 i 6.3 daje detaljne informacije o akcijama koje softverski paket mora da podrži u radu sa standardnim i fazi XML, DTD i XSD dokumentima i XQuery upitima.



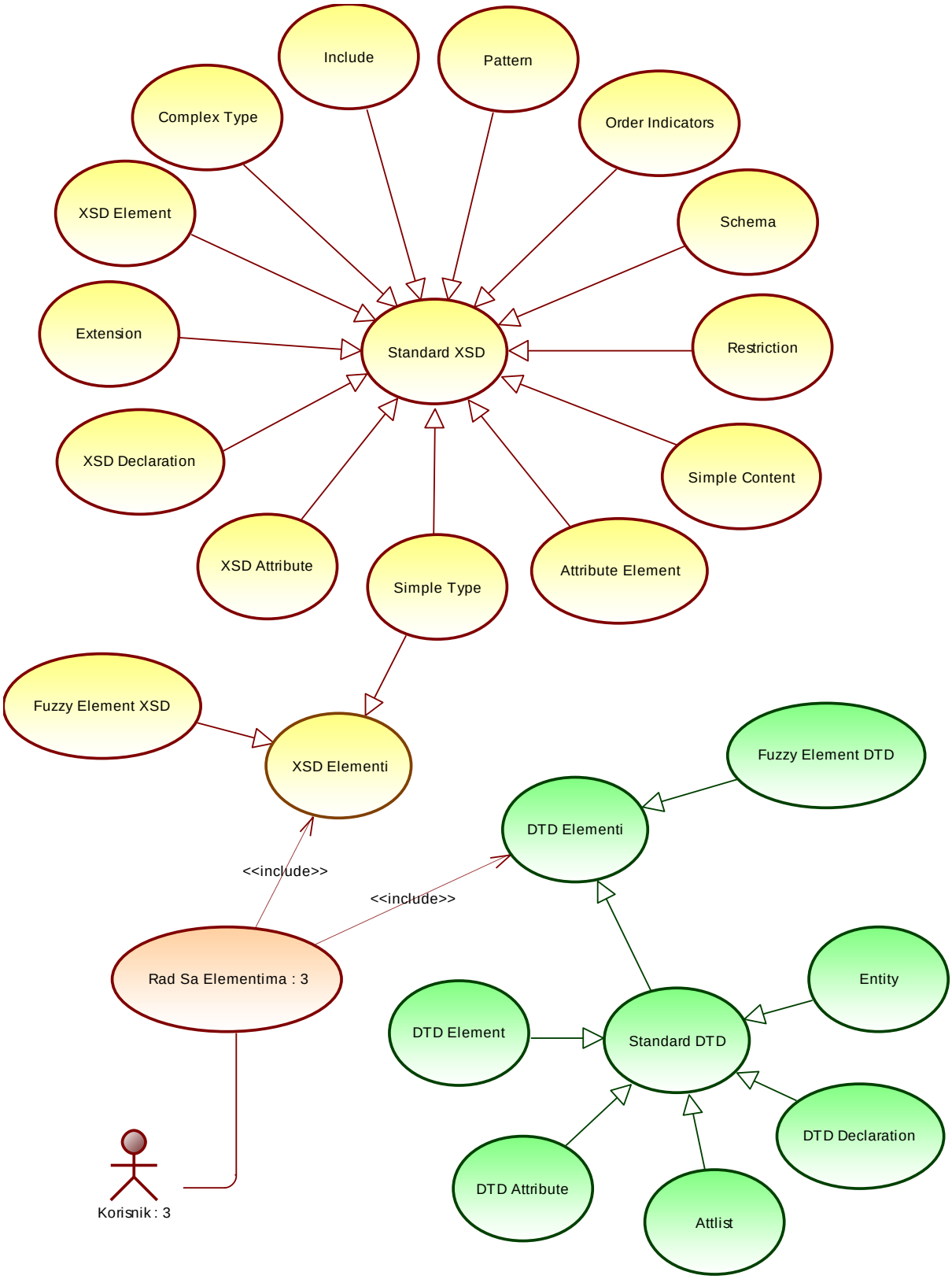
Slika 6.1. Osnovne funkcionalnosti.

Slika 6.2 pruža pojašnjenje akcije *Rad sa elementima* sa prethodnog dijagrama i obuhvata pod-elemente za rad sa XML elementima (*Standard XML, Fuzzy XML, Lingvističke promenljive*…) i XQuery upitima (*XQuery upiti*…).



Slika 6.2. Rad sa XML elementima i XQuery upitima.

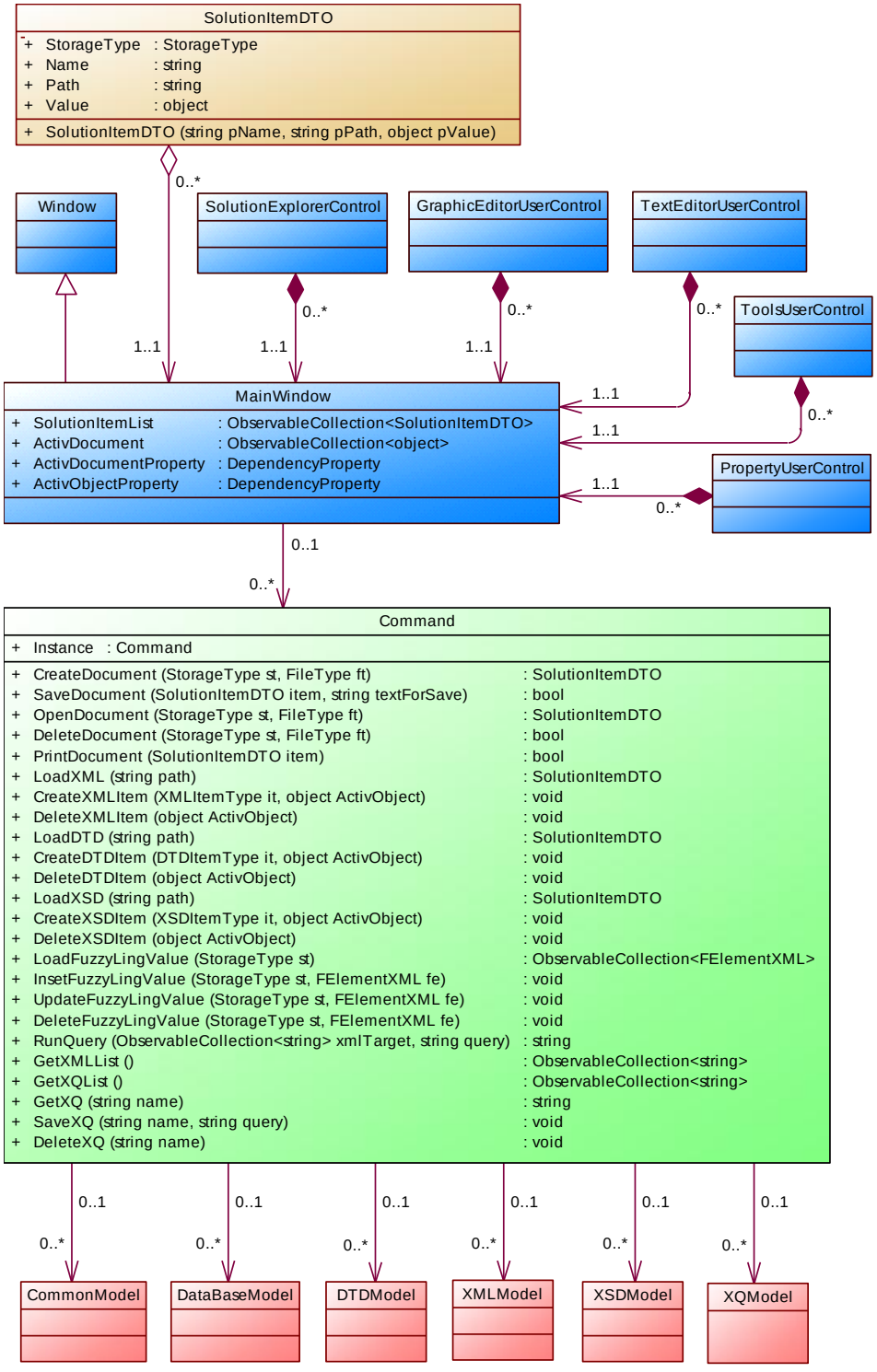
Dodatno pojašnjenje akcije *Rad sa elementima* prikazano je na slici 6.3 i obuhvata rad sa DTD (*DTD elementi*, *Standard DTD*…) i XSD elementima (*Standard XSD*, *Fuzzy Element XSD*…).



Slika 6.3. Rad sa DTD i XSD elementima.

Dijagram klasa (Class Diagram): Zbog velikog broja klasa koje su implementirane u projektu, u nastavku su prikazane samo najosnovnije, a koje predstavljaju osnovu modela koji

se razvija. U klasama su navedeni samo najbitniji parametri i metode, kako bi se olakšala čitljivost dijagrama. Dijagram klasa dat na slici 6.4 opisuje osnovne klase zadužene za funkcionisanje same aplikacije.

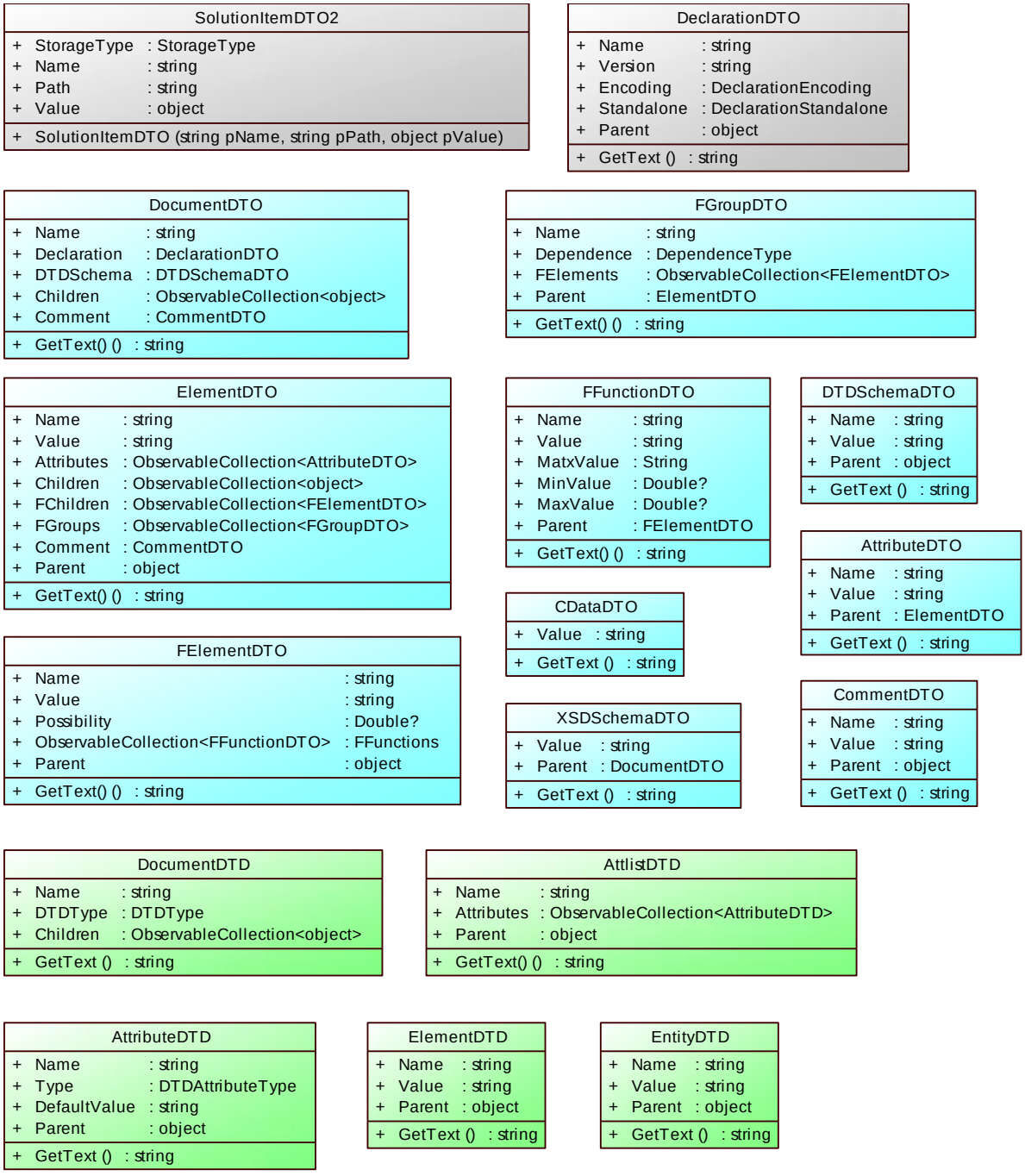


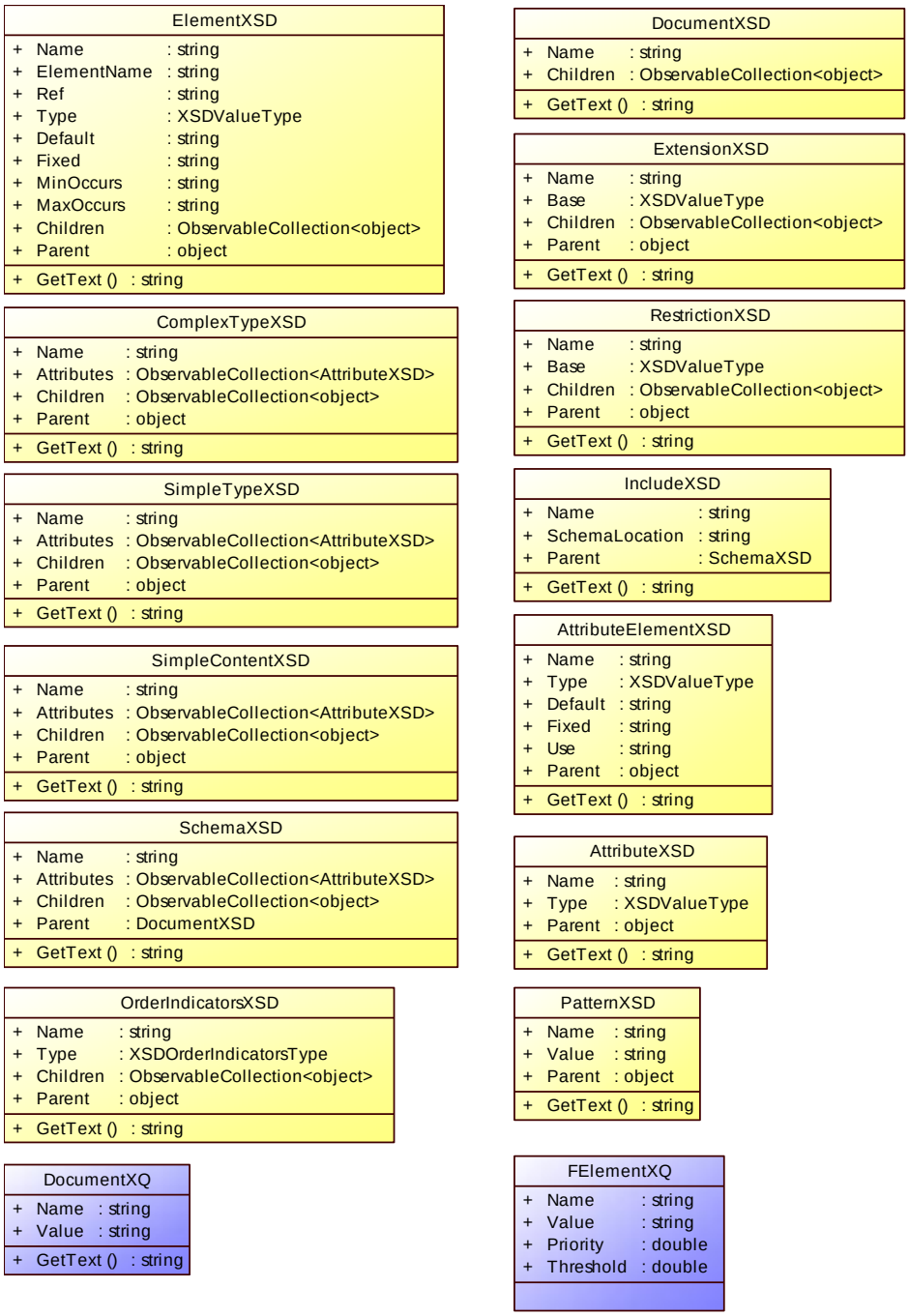
Slika 6.4. Osnovna struktura.

MainWindow je inicijalna klasa koja povezuje grafički interfejs sa komandama iz *Command* klase. *Command* klasa sadrži logiku za izdavanje komandi drugim klasama, od kojih se svaka brine o određenom tipu podataka opisanom u *Model* klasama. Kontrole

opisane u *Control* klasama, koje se uključuju u *MainWindow* klasu, predstavljaju grafički interfejs. *SolutionItemDTO* je klasa koja sadrži podatke, a kolekcija takvih objekata čuva se u *MainWindow* klasi.

Dijagram klasa na slici 6.5 daje skup najbitnijih kompleksnih tipova kojima rukuje aplikacija.

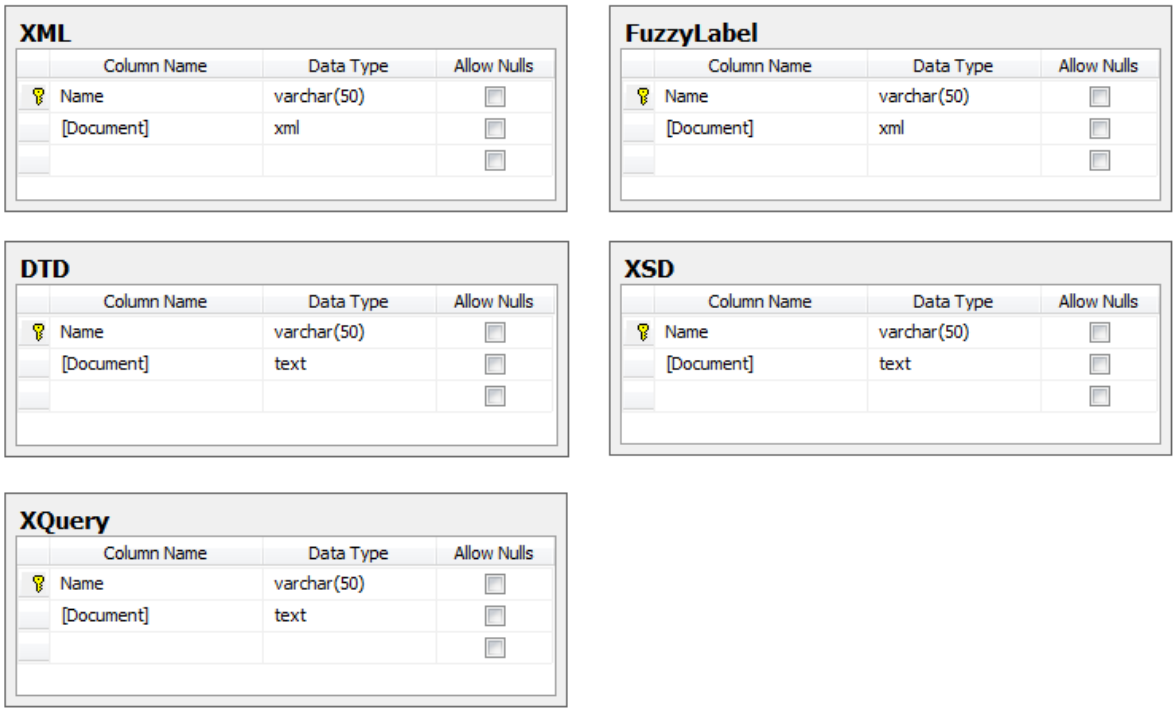




Slika 6.5. Kompleksni tipovi - XML, DTD, XSD i XQuery klase.

Klasa *SolutionItemDTO* služi za definisanje dokumenata. *DeclarationDTO* predstavlja zajedničku klasu. Klase *DocumentDTO*, *ElementDTO*, *CDataDTO*, *AttributeDTO*, *CommentDTO*, *XSDSchemaDTO* i *DTDSchemaDTO* opisuju elemente u XML dokumentu, dok klase *FGroupDTO*, *FElementDTO* i *FFunctionDTO* služe za definisanje fazi elemenata. DTD dokumenti se opisuju klasama *DocumentDTD*, *ElementDTD*, *AttibuteDTD*, *EntityDTD* i *AttlistDTD*. XSQ dokumenti se definišu klasama *DocumentXSD*, *ElementXSD*, *ComplexTypeXSD*, *SympleTypeXSD*, *SimpleContexXSD*, *AttributeXSD*, *IncludeXSD*, *AttributeElementXSD*, *RestrictionXSD*, *SchemaXSD*, *OrderIndicatorsXSD*, *PatternXSD* i *ExtensionXSD*. Klase *DocumentXQ* i *FElementXQ* opisuju fazi XQuery upite.

Fizički model baze podataka: Pored rada sa fajl sistemom, dokumente je moguće skladištiti u bazi podataka. Za razvojnu bazu je odabran Microsoft SQL Server 2008 R2, kao prirodna sprega sa ostalim .NET tehnologijama korišćenim u razvoju aplikacije. Ovaj server omogućava rad sa XML tipovima podataka i postavljanje XQuery upita nad XML sadržajem. Fizički model baze podataka sa kojom radi aplikacija prikazan je na slici 6.6.



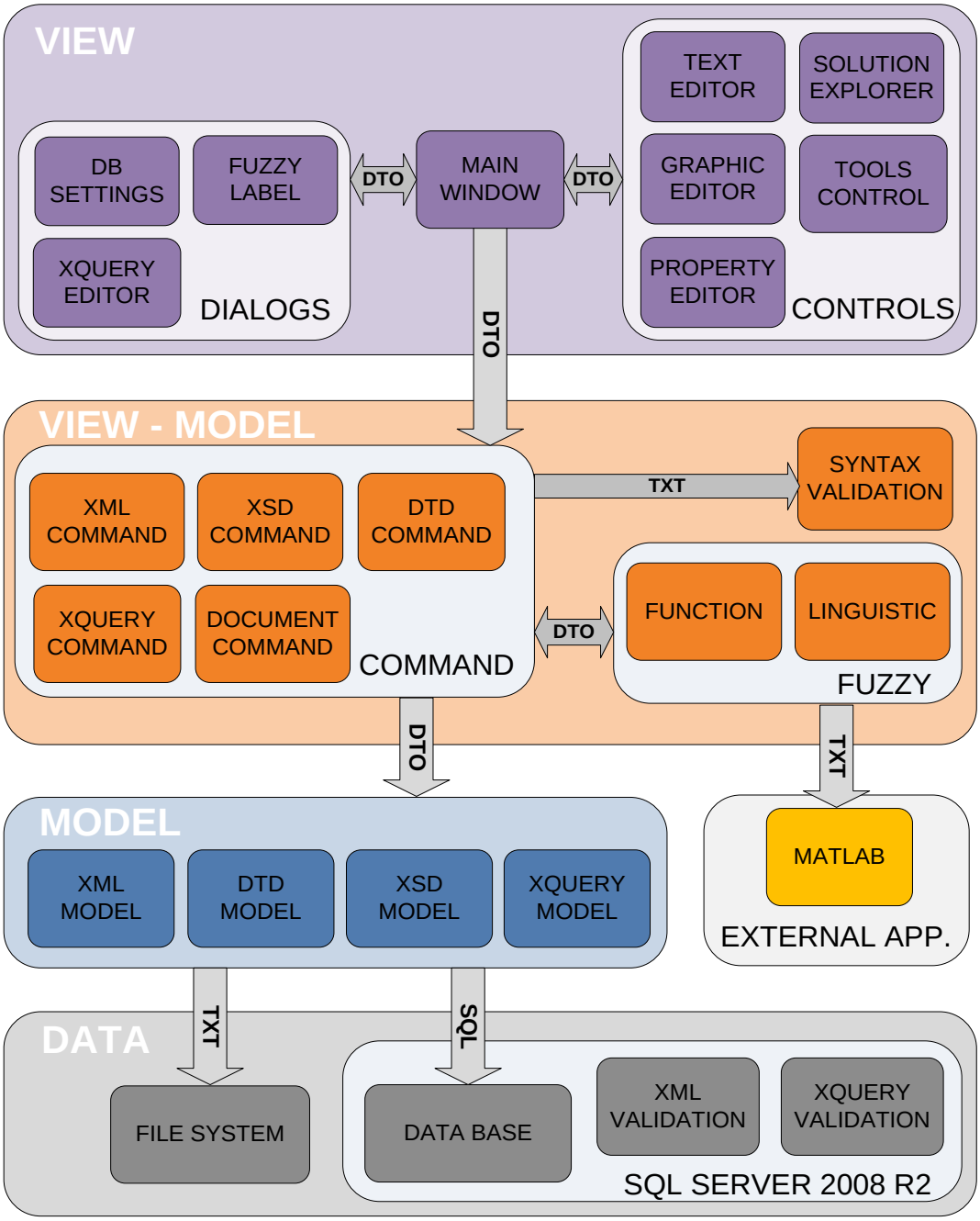
Slika 6.6. Fizički model baze podataka.

Tabela *XML* sadrži fazi XML dokumenta u polju *Document* i njihova imena u polju *Name*. Isto važi i za ostale tabele, sa tom razlikom što *DTD* tabela sadrži fazi DTD dokumenata, *XSD* tabela sadrži fazi XSD dokumenta, a *XQuery* tabela sadrži prioritizovane fazi XQuery upite. *FuzzyLabel* tabela u polju *Name* sadrži nazive fazi lingvističkih promenljivih, dok su polju *Document* njihove predefinisane funkcije pripadnosti.

6.1.2 Arhitektura aplikacije

U nastavku je objašnjen način implementacije softverskog paketa i dat grafički prikaz uz objašnjenje logičkih komponenti i njihovih veza.

Sama aplikacija se sastoji iz više logičkih slojeva, organizovanih po principima **MVVM** (Model View ViewModel) paterna. Raspored komponenata aplikacije i njihove međusobne veze prikazane su na slici 6.7.



Slika 6.7. Blok dijagram strukture aplikacije.

Data sloj je najniži nivo koji se brine o skladištenju podataka, a sastoji se iz fajl sistema i baze podataka. Sva dokumenta je moguće uskladištiti kako u standardne fajlove tako i u bazu podataka. Za rad sa bazom implementirana je podrška za Microsoft SQL Server 2008 R2.

Sloj **Model** sadrži nekoliko komponenata, od kojih svaki ima zadatak da direktno komunicira sa određenim tipom objekata iz *Data* sloja. *XML model* podržava rad sa XML objektima, *DTD model* i *XSD model* su zaduženi za rad sa XML šemama, a *XQuery model* omogućava rad sa XQuery upitima. Svaki od navedenih modela pored mogućnosti rada sa standardnim sintaksama jezika, implementira i deo zadužen za rad sa odgovarajućim fazi sintaksama.

Sloj **ViewModel** poseduje mogućnost komunikacije sa slojem ispod i sastoji se iz nekoliko logičkih celina. *Command* oblast grupiše komponente za rad sa komandama, koji implementiraju interfejse za pristupanje modelu. *Syntax Validation* komponenta omogućuje validaciju sintaksi dokumenata. Dok *Fuzzy* oblast omogućava rad sa fazi logikom što obuhvata funkcije pripadnosti i fazi lingvističke promenljive. Sama *Fuzzy* oblast se oslanja na ***Matlab*** komponentu, koji predstavlja eksternu aplikaciju zaduženu za izračunavanje kompleksnih matematičkih funkcija pri određivanju nivoa pripadnosti. Upotreba MATLAB-a kao matematičkog modula, omogućila je definisanje proizvoljnih funkcija pripadnosti i maksimalne performanse pri radu.

Poslednji sloj **View**, implementira korisnički interfejs, putem kojeg korisnici izvršavaju funkcionalnosti definisane u *Command* oblasti *ViewModel* sloja. *Main Window* je glavni prozor aplikacije, koji uključuje kontrole iz oblasti *Controls* i omogućuje pozivanje ostalih dijalog prozora iz oblasti *Dialogs*.

6.2. Korišćene tehnologije

Tehnologije koje se koriste u izradi aplikacije navedene su i opisane u nastavku teksta. Jedna od ideja je da se celokupan razvoj bazira na upotrebi tehnologija razvijenih od strane *Microsoft*-a, koje su međusobno kompatibilne. Koriste se *Express* verzije pomenutih programa koje su besplatne i javno dostupne, tako da se može reći da je kompletna aplikacija razvijena na besplatnom softveru.

Programski jezik C# (C sharp) je objektno orjentisan programski jezik razvijen od strane *Microsoft*-a i lansiran 2002. godine kao sastavni deo *Microsoft .NET Framework 1.0*. Kasnije je prihvaćen kao standard od ECMA i ISO. Razvija se kao proceduralni, objektno orjentisani jezik, sintaksno baziran na C++ jeziku. C# jezik uključuje aspekte nekoliko drugih programskih jezika (na primer Delphi i Java) i posebnu pažnju posvećuje uprošćenosti. Jezik je namenjen za opštu primenu i izradu aplikacija za .NET Framework platformu. Više detalja o ovom programskom jeziku može se pronaći u knjizi (Schildt, 2010).

Microsoft .NET Framework je programska komponenta koja predstavlja deo *Microsoft Windows* operativnog sistema. Ona obezbeđuje veliki broj predefinisanih kod rešenja za opšte programerske zahteve i upravljanje izvršnim programa napisanim posebno za to okruženje. Kod rešenja javljaju se u obliku biblioteka klasa i poseduju veliki izbor programa neophodnih u oblastima: Korisničkog interfejsa, pristupa podacima, konekcija na bazu, kriptografije, kreiranja Web aplikacija, numeričkih algoritama i mrežne komunikacije. Klas-biblioteke koriste programeri koji ih kombinuju sa sopstvenim kodom i na taj način kreiraju aplikacije. Aktuelna verzija je 4.5, a instalacija se besplatno može preuzeti sa zvaničnog sajta *Microsoft*-a.

WPF (Windows Presentation Foundation) je grafički podsistem za renderovanje korisničkog interfejsa na Windows baziranim sistemima. Razvila ga je kompanija *Microsoft*, a inicijalno je lansiran kao deo *.NET 3.0 Framework*-a. WPF korisni DirectX, umesto GDI (Graphics Device Interface) sistema. Takođe pokušava da obezbedi konzistentan programski model za izgradnju aplikacija i omogućava razdvajanje korisničkog interfejsa i poslovne logike. Tehnologija koristi XAML (eXtensible Application Markup Language), koji je produkt XML-a i služi za definisanje i povezivanje različitih elementa korisničkog interfejsa. WPF aplikacije mogu biti samostalni desktop programi, ili postavljeni kao ugrađeni objekti na nekom Web sajtu. Ovom tehnologijom ujedinjeni su brojni zajednički elementi korisničkog interfejsa, kao što su 2D/3D renderovanje, fiksna i adaptivna dokumenata, tipografija, vektorska grafika i animacija u realnom vremenu. Navedeni elementi mogu biti

povezani i obrađivani na osnovu različitih događaja (events), korisničkih interakcija i povezanih podataka (data bindings). Više detalja o WPF tehnologiji može se pronaći u knjigama (Sells & Griffiths, 2007) i (Nathan, 2010).

MSSQL Server (Microsoft SQL Server) je savremena relaciona baza podataka, razvijena od strane *Microsoft*-a. Predstavlja program čija je osnovna funkcija skladištenje i preuzimanje podataka po zahtevu drugih programa. Postoji najmanje desetak različitih izdanja sistema *Microsoft SQL Server* optimizovanih za različite grupe korisnika i za različite zadatke. Primarni upitni jezici SQL Servera su TSQL (Transact-SQL) i ANSI (American National Standards Institute) SQL.

Visual Studio je integrisano razvojno okruženje razvijeno od strane *Microsoft* korporacije. Koristi se za razvijanje konzolnih i grafičkih aplikacija što uključuje Windows form aplikacije, Web aplikacije, servise, itd. Aplikacije razvijene u njemu mogu da rade sa platformama Microsoft Windows, Windows Mobile, Windows CE, .NET Framework, .NET Compact Framework i Microsoft Silverlight. Visual Studio podržava različite programske jezike u kojima je moguće pisati kod i otklanjati greške. Ugrađeni jezici obuhvataju C, C++, VisualBasic.NET, C# i F#. Podrška za druge jezike kao što su M, Python i Ruby dostupna je preko dodatnih jezičkih servisa. Podržan je rad sa skript jezicima kao što su XML/XSLT, HTML/XHTML, JavaScript i CSS.

MATLAB (MATrix LABoratory) je okruženje za numeričke proračune i programski jezik koji proizvodi kompanija MathWorks**.** Omogućava lako manipulisanje matricama, prikazivanje funkcija, implementaciju algoritama, stvaranje grafičkog korisničkog interfejsa kao i povezivanje sa programima pisanim u drugim jezicima. Izumeo ga je kasnih 1970-ih Cleve Moler, šef katedre za informatiku na Univerzitetu "Novi Meksiko". Za njega su namenski razvijeni brojni dodaci za: matematiku i optimizacije, statistiku i analizu podataka, kontrolu i analizu sistema, procesiranje signala i komunikaciju, obradu slika, testiranje i merenje, finansijsko modelovanje i analizu, povezivanje sa drugim aplikacijama, povezivanje baza podataka i izveštavanje, distribuirano računanje i fazi logiku.

6.3 Implementacija

Aplikacija je implementirana upotrebom WPF tehnologije i .NET framework-a. Delovi XAML koda i C# programskog jezika bitni za razumevanje procesa implementacije su prikazani i objašnjeni u nastavku teksta.

6.3.1 Inicijalizacija aplikacije

Za inicijalizaciju aplikacije koristimo *FuzzyPMF.app* klasu, koja postavlja *MainWindow.xaml* kao početni prozor i uključuje stilove postavljanjem *Dictionary.xaml* u resurse aplikacije. Klasa je navedena u listingu 6.1.

```
<Application x:Class="FuzzyPMF.App"
             xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
             xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
             StartupUri="MainWindow.xaml">
    <Application.Resources>
        <ResourceDictionary>
            <ResourceDictionary.MergedDictionaries>
                <ResourceDictionary Source="Dictionary.xaml" />
            </ResourceDictionary.MergedDictionaries>
```

```
        </ResourceDictionary>
    </Application.Resources>
</Application>
```

Listing 6.1. Inicijalna klasa FuzzyPMF.app.

6.3.2 Konfiguracija aplikacije

Eksterna konfiguracija aplikacije obavlja se putem *App.config* datoteke. Pomoću nje je moguće podesiti konekciju na bazu podataka bez pokretanja same aplikacije. Datoteka je u obliku XML dokumenta i data je u listingu 6.2.

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
   <appSettings>
      <add key="ConnectionString" value="Data Source=GORAN;Initial
            atalog=PMF;Integrated Security=True" />
   </appSettings>
</configuration>
```

Listing 6.2. Konfiguraciona datoteka App.config.

6.3.3 Glavni prozor

Glavni prozor aplikacije je *MainWindow*, a svi ostali prozori se pokreću iz njega. Prozor uključuje nekoliko kontrola, iz *View namespace*-a:

- *SolutionExplorerControl* - Prikaz aktivnih dokumenata
- *ToolsUserControl* - Skup alatki
- *PropertyUserControl* - Podešavanje opcija
- *GraphicEditorUserControl* - Grafički editor
- *TextEditorUserControl* - Tekstualni editor

U listingu 6.3 su navedeni bitni delovi XAML koda glavnog prozora, koji su potrebni za razumevanje njegove strukture. Pozadina *MainWindow* prozora rađena je u C# programskom jezik i opisana klasom *MainWindow.cs*, koja neće biti navedena zbog veličine.

```
<Window x:Class="FuzzyPMF.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:local="clr-namespace:FuzzyPMF"
        xmlns:view="clr-namespace:FuzzyPMF.View"
        Title="Fuzzy XML Editor" Width="1000" Height="700"
        Background="{DynamicResource BackgroundColor}">
    <Grid>
        <Grid.RowDefinitions>
            <RowDefinition Height="30"></RowDefinition>
            <RowDefinition></RowDefinition>
            <RowDefinition Height="20"></RowDefinition>
        </Grid.RowDefinitions>
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="300"></ColumnDefinition>
            <ColumnDefinition Width="3"></ColumnDefinition>
            <ColumnDefinition Width="*"></ColumnDefinition>
        </Grid.ColumnDefinitions>
        <Menu Grid.Row="0" Grid.ColumnSpan="5">
```

```
            . . .
        </Menu>
        <Grid Grid.Row="1" Grid.Column="0">
            <Grid.RowDefinitions>
                <RowDefinition Height="Auto"></RowDefinition>
                <RowDefinition Height="3"></RowDefinition>
                <RowDefinition Height="Auto"></RowDefinition>
                <RowDefinition Height="3"></RowDefinition>
                <RowDefinition Height="Auto"></RowDefinition>
            </Grid.RowDefinitions>
            <view:SolutionExplorerControl x:Name="test" Grid.Row="0"
DataContext="{Binding ActivDocument, RelativeSource={RelativeSource
FindAncestor, AncestorType={x:Type local:MainWindow}}, Mode=TwoWay}"/>
            <GridSplitter Grid.Row="1" Background="LightGray" Width="Auto"
HorizontalAlignment="Stretch" VerticalAlignment="Stretch"/>
            <view:ToolsUserControl Grid.Row="2" DataContext="{Binding
ActivObject, RelativeSource={RelativeSource FindAncestor,
AncestorType={x:Type local:MainWindow}}, Mode=TwoWay}"/>
            <GridSplitter Grid.Row="3" Background="LightGray" Width="Auto"
HorizontalAlignment="Stretch" VerticalAlignment="Stretch"/>
            <view:PropertyUserControl Grid.Row="4" DataContext="{Binding
ActivObject, RelativeSource={RelativeSource FindAncestor,
AncestorType={x:Type local:MainWindow}}, Mode=TwoWay}"/>
        </Grid>
        <GridSplitter  Grid.Row="1" Grid.Column="1" Background="LightGray"
Width="Auto"  HorizontalAlignment="Stretch"/>
        <Grid  Grid.Row="1" Grid.Column="2">
            <Grid.RowDefinitions>
                <RowDefinition Height="*"></RowDefinition>
                <RowDefinition Height="3"></RowDefinition>
                <RowDefinition Height="*"></RowDefinition>
            </Grid.RowDefinitions>
            <view:GraphicEditorUserControl Grid.Row="0"
DataContext="{Binding ActivObject, RelativeSource={RelativeSource
FindAncestor, AncestorType={x:Type local:MainWindow}}, Mode=TwoWay}"/>
            <GridSplitter  Grid.Row="1" Background="LightGray" Width="Auto"
HorizontalAlignment="Stretch" VerticalAlignment="Stretch"/>
            <view:TextEditorUserControl Grid.Row="2" x:Name="TextEditor" />
        </Grid>
        <StatusBar Grid.Row="2" Grid.ColumnSpan="5"/>
    </Grid>
</Window>
```

Listing 6.3. Glavni prozor MainWindow.xaml.

6.3.4 Komandna klasa

Za izvršenje svih akcija u aplikaciji zadužena je klasa *Command.cs*. Klase višeg sloja (*View*) pozivaju metode klase *Command*, a sama klasa određuje koje će module pozvani ili kombinovati, da bi izvršila zahtev. Metode bitne za razumevanje funkcionisanja klase date su u listingu 6.4.

```
public sealed class Command
{
   #region Singleton
   public static readonly Command instance = new Command();
   static Command() { }
   Command() { }
   public static Command Instance
```

```
    {
        get{ return instance;}
    }
    #endregion

    public SolutionItemDTO CreateDocument(StorageType st, FileType ft)...
    public bool SaveDocument(SolutionItemDTO item, string textForSave="")...
    public SolutionItemDTO OpenDocument(StorageType st, FileType ft)...
    public bool DeleteDocument(StorageType st, FileType ft) ...
    public bool PrintDocument(SolutionItemDTO item) ...

    #region FuzzyLing
    public ObservableCollection<FElementDTO> LoadFuzzyLing(StorageType st)..
    public void InsetFuzzyLingValue(StorageType st, FElementDTO fe)...
    public void UpdateFuzzyLingValue(StorageType st, FElementDTO fe)...
    public void DeleteFuzzyLingValue(StorageType st, FElementDTO fe)...
    #endregion

    #region XQuery
    public string RunQuery(string query, string xmlTarget)...
    public ObservableCollection<string> GetXMLList()...
    public ObservableCollection<string> GetXQList()...
    public string GetXQ(string name)...
    public void SaveXQ(string name, string query)...
    #endregion

    #region File
    public SolutionItemDTO LoadXML(string path)...
    public void CreateXMLItem(XMLItemType it, object ActivObject)...
    public void DeleteXMLItem(object ActivObject)...

    public SolutionItemDTO LoadDTD(string path)...
    public void CreateDTDItem(DTDItemType it, object ActivObject)...
    public void DeleteDTDItem(object ActivObject)...
    #endregion

    #region Database
    public SolutionItemDTO DBLoadXML(string path)...
    public void DBCreateXMLItem(XMLItemType it, object ActivObject)...
    public void DBDeleteXMLItem(object ActivObject)...
    public SolutionItemDTO DBLoadDTD(string path)...
    public void DBCreateDTDItem(DTDItemType it, object ActivObject)...
    public void DBDeleteDTDItem(object ActivObject)...
    #endregion
}
```

Listing 6.4. Komandna klasa Command.cs.

6.3.5 Enumeracija

Kako bi se očuvala konzistentnost samih podataka, poželjna je upotreba enumeracije u definisanju tipova koji sadrže ograničen skup precizno određenih vrednosti. WPF omogućava direktno povezivanje (Binding) elemenata sa forme i određenih tipova podataka, u ovom slučaju enumeracija. Zbog toga se u enumeracijama upotrebljava atribut *StringValue* za opis elemenata enumeracije. Ovaj atribut omogućava da se na element forme direktno poveže određena enumeracija, pri čemu elementi na formi prikazuju vrednosti iz *StringValue* polja, koje može sadržati znakove čija upotreba nije dozvoljena u samoj enumeraciji. Enumeracije i klase *StringValue* i *StringEnum* dati su u listingu 6.5.

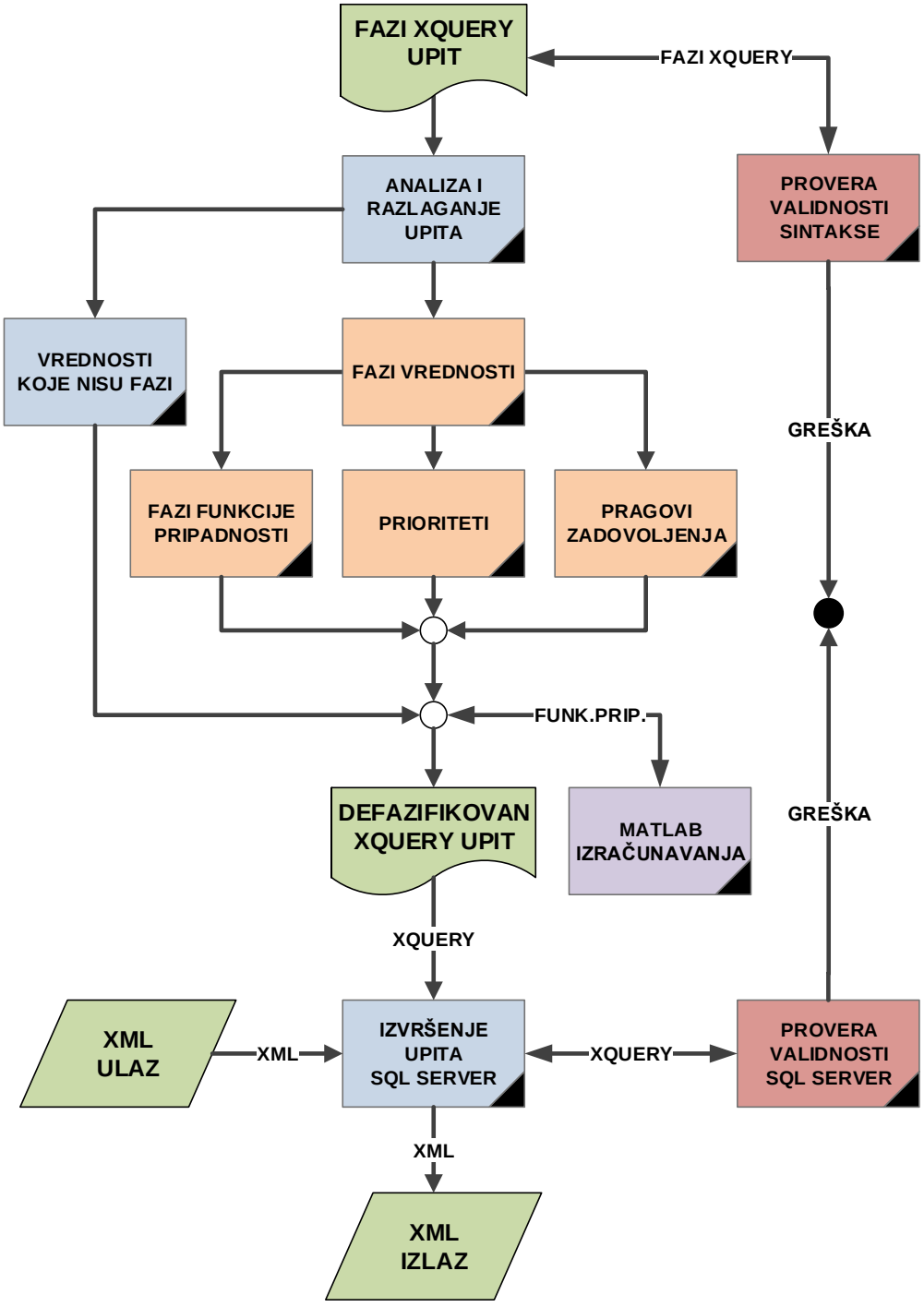
```
namespace FuzzyPMF
{
    public enum StorageType {
        File,
        DB
}

    public enum FileType {
        XML,
        DTD,
        XSD,
        XQuery
}

    public enum XMLItemType {
        Attribute,
        Comment,
        Document,
        Declaration,
        Element,
        DTDSchema,
        XSDSchema,
        FGroup,
        FElement,
        FFunction
    }

    public enum DependenceType {
        [StringValue("AND")] AND,
        [StringValue("NAND")] NAND,
        [StringValue("XAND")] XAND,
        [StringValue("XNAND")] XNAND,
        [StringValue("OR")] OR,
        [StringValue("NOR")] NOR,
        [StringValue("XOR")] XOR,
        [StringValue("XNOR")] XNOR
    }

    public enum DeclarationVersion {
        [StringValue("1.0")] v1
    }

    public enum DeclarationEncoding {
        [StringValue("")] Nothing,
        [StringValue("UTF-8")] UTF_8,
        [StringValue("UTF-16")] UTF_16,
        [StringValue("ISO-10646-UCS-2")] ISO_10646_UCS_2,
        [StringValue("ISO-10646-UCS-4")] ISO_10646_UCS_4,
        [StringValue("ISO-8859-1")] ISO_8859_1,
        [StringValue("ISO-8859-2")] ISO_8859_2,
        [StringValue("ISO-8859-3")] ISO_8859_3,
        [StringValue("ISO-8859-4")] ISO_8859_4,
        [StringValue("ISO-8859-5")] ISO_8859_5,
        [StringValue("ISO-8859-6")] ISO_8859_6,
        [StringValue("ISO-8859-7")] ISO_8859_7,
        [StringValue("ISO-8859-8")] ISO_8859_8,
        [StringValue("ISO-8859-9")] ISO_8859_9,
        [StringValue("ISO-2022-JP")] ISO_2022_JP,
        [StringValue("Shift_JIS")] Shift_JIS,
        [StringValue("EUC-JP")] EUC_JP
    }
```

```
    public enum DeclarationStandalone {
        [StringValue("")] Nothing,
        [StringValue("Yes")] Yes,
        [StringValue("No")] No
    }

    public enum DTDItemType {
        Attlist,
        Attribute,
        Declaration,
        Document,
        Element,
        Entity,
        FElement
    }

    public enum DTDType {
        Internal,
        External
    }

    public enum DTDAttributeType {
        CDATA,      //  The value is character data
        ELIST,      //  The value must be one from an enumerated list
        ID,         //  The value is a unique id
        IDREF,      //  The value is the id of another element
        IDREFS,     //  The value is a list of other ids
        NMTOKEN,    //  The value is a valid XML name
        NMTOKENS,   //  The value is a list of valid XML names
        ENTITY,     //  The value is an entity
        ENTITIES,   //  The value is a list of entities
        NOTATION    //  The value is a name of a notation
    }

    public enum DTDDefaultValue {
        [StringValue("value")] value,//  The default value of the attribute
        [StringValue("#REQUIRED")] REQUIRED, //  The attribute is required
        [StringValue("#IMPLIED")] IMPLIED,//  The attribute is not required
        [StringValue("#FIXED value")] FIXED//  The attribute value is fixed
    }

    public class StringValue : System.Attribute {
        private string _value;
        public StringValue(string value){ _value = value; }
        public string Value{ get { return _value; }}
    }

    public static class StringEnum {
        public static string GetStringValue(Enum value)
        {   string output = null;
            Type type = value.GetType();
            FieldInfo fi = type.GetField(value.ToString());
            StringValue[] attrs = fi.GetCustomAttributes(
            typeof(StringValue), false) as StringValue[];
            if (attrs.Length > 0){ output = attrs[0].Value; }
            return output;
        }
    }
}
```

Listing 6.5. Enumeracije i klase StringValue i StringEnum.

6.3.6 Obrada fazi XQuery upita

Microsoft SQL Server 2008 R2 omogućava izvršavanje standardnih XQuery upita nad XML podacima u bazi podataka. Proširenje standardnih XQuery funkcionalnosti na nivou same baze nije moguće zbog zatvorenosti koda. Ideja je da aplikacija prvo analizira, a potom razloži fazi XQuery upit na nezavisne celine. Nakon toga izvršava se defazifikacija fazi delova upita, a defazifikovan upit se dalje prosleđuje bazi podataka kao standardni XQuery upit. Jedan od načina da se ovo postigne je dat na slici 6.8, koja prikazuje način izvršenja fazi XQuery upita nad XML dokumentom upotrebom Matlab modula.



Slika 6.8. Postupak izvršenja fazi XQuery upita.

Pri postavljanju fazi XQuery upita proverava se validnost sintakse, a u slučaju neispravnosti, šalje se poruka o grešci i prekida izvršavanje. U slučaju da je upit ispravan, vrši se njegova analiza i razlaganje. Vrednosti koje nisu fazi se automatski prosleđuju dalje, dok se fazi vrednosti razlažu na više delova: funkcije pripadnosti, prioritete i pragove zadovoljenja. Dobijene funkcije pripadnosti se konvertuju u odgovarajući format i šalju na obradu Matlab aplikaciji, koja izračunava i vraća vrednosti za zadate tačke. Razložen upit se sada spaja u defazifikovan XQuery upit i prosleđuje Miscrosoft SQL Serveru, gde se proverava njegova validnost. Ukoliko je upit validan izvršava se nad selektovanim XML dokumentom, a rezultat se vraća aplikaciji.

Način postavljanja standardnog XQuey upita pomoću SQL upita u Miscrosoft SQL Serveru, dat je u listingu 6.6.

```
SELECT
	Document.query('
		<result>
		{
		for $item in /merenje/lokacija
		let $naziv := $item/@naziv
		let $temperatura := $item/temperatura/text()
		let $vlaznost := $item/vlaznost/text()
		let $pritisak := $item/pritisak/text()
		let $smer :=$item/vetar/smer/text()
		let $brzina:=$item/vetar/brzina/text()
		let $oblacnost:=$item/oblacnost/text()
		where $temperatura > 17 and $temperatura < 25
			and $vlaznost > 0.45 and $vlaznost < 0.85
			and $brzina < 30.0
			and $oblacnost < 0.50
		order by $naziv
		return
		<lokacija naziv="{$naziv}">
			<temperatura>{$temperatura}</temperatura>
			<vlaznost>{$vlaznost}</vlaznost>
			<pritisak>{$pritisak}</pritisak>
			<vetar>
				<smer>{$smer}</smer>
				<brzina>{$brzina}</brzina>
			</vetar>
			<oblacnost>{$oblacnost}</oblacnost>
		</lokacija>
		}
		</result>
		')
FROM XML
WHERE Name = 'Merenje'
```

Listing 6.6. Postavljanje XQuery upita Miscrosoft SQL Serveru.

Prethodni listing prikazuje kako se u Miscrosoft SQL Serveru, nad tabelom koja nosi ima *XML*, zadaje specifičan selekcioni upit nad dokumentom pod imenom *Merenje*. Kao polje za selekciju navedeno je *Document.query(...)*, koje sadrži standardni XQuery upit.

Za kompleksne matematičke proračune, koji se koriste pri određivanju pripadnosti za fazi vrednosti koristi se **Matlab** aplikacija. Statička metoda za komunikaciju sa Matlab

aplikacijom koja prima matematički izraz i vraća rezultat u obliku teksta, navedena je u listingu 6.7.

```
private static string RunMatlab(string expression)
{
      MLAppClass matlab = new MLAppClass();
      string result = matlab.Execute(expression);
      result = result.Substring(result.IndexOf("ans =") + 5).Trim();
      return result;
}
```

Listing 6.7. Pozivanje Matlab aplikacije.

Aplikacija omogućava definisanje i rad sa Null vrednostima tipa nepoznata i neprimenljiva. XML sintaksa već podržava definisanje Null vrednosti i nije bilo potrebe za posebnom obradom u procesu implementacije. Na osnovu smernica datih u prethodnom poglavlju implementirana je obrada Null vrednosti u slučaju postavljanja XQuery upita. Kako se aplikacija oslanja na Miscrosoft SQL Server pri radu sa XML dokumentima dosta toga je već odrađeno. Ukoliko se Miscrosoft SQL serveru prosledi upit za obradu XML dokumenata koji sadrže Null vrednosti, server obrađuje samo one vrednosti koje ne sadrže nedostajuće informacije, dok Null vrednosti ignoriše. Ta osobina u kombinaciji sa odgovarajućim načinom postavljanja upita (za primer videti poglavlje 9 i listing 9.4), omogućava pravilnu obradu oba tipa Null vrednosti.

Glava VII

OPIS FUNKCIONALNOSTI

Poglavlje daje pregled mogućnosti razvijenog softverskog paketa. Obrađuje proces instalacije aplikacije i njeno inicijalno pokretanje, skladištenje dokumenata, rad kako sa standardnim tako i sa fazi XML, DTD i XSD dokumentima. Takođe prikazuje upotrebu fazi XQuery editora u izvršavanju fazi XQuery upita sa prioritetima.

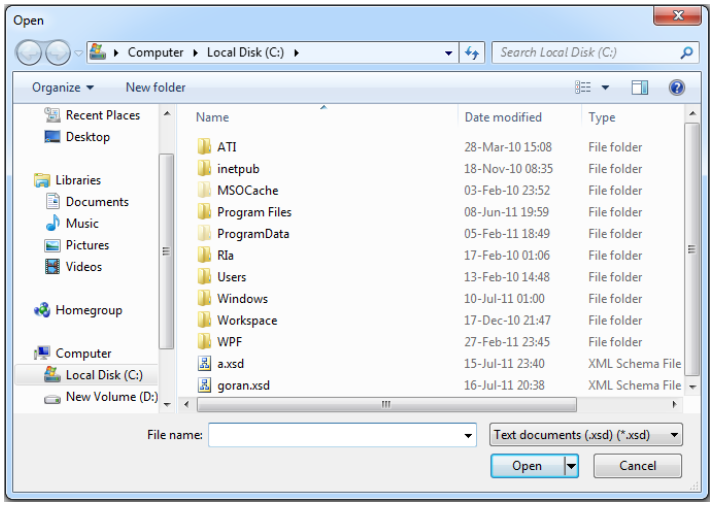
Izvršna verzija aplikacije, video datoteke i dodatne datoteke nalaze se na pratećem CD-u.

7.1 Pokretanje

Aplikacija radi pod Windows operativnim sistemom i koristi .NET 4 framework. Ukoliko korisnik želi da dokumenta skladišti u bazi podataka, neophodno je da na računaru bude instaliran i startovan SQL Server. Proces instalacije aplikacije se vrši pokretanjem datoteke *Setup.exe*. Korisniku preostaje da odabere lokaciju za snimanje aplikacije, prava pristupa i po startovanju aplikacije podesi konekciju na aktuelnu bazu podataka. U slučaju da na računaru ne postoje .NET 4 framework ili SQL Server, korisniku će u procesu instalacije biti ponuđene opcije za preuzimanje i instaliranje komponenata koje nedostaju.

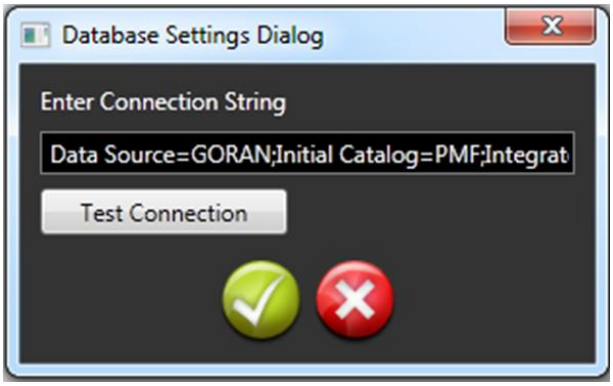
7.2 Skladištenje dokumenata

Dokumenti se u aplikaciji mogu čuvati kao standardni fajlovi ili u bazi podataka. Oba tipa skladištenja omogućavaju kreiranje dokumenata, njihovo skladištenje, učitavanje, izmenu i brisanje. Najčešće se XML, DTD i XSD dokumenti čuvaju u datotekama koje standardno nose ekstenzije *.xml, *.dtd i *.xsd. Fajl sistem predstavlja osnovni način za skladištenje pomenutih dokumenata, a prozor za učitavanja dokumenata prikazan je na slici 7.1.



Slika 7.1. Prozor za učitavanje dokumenata u aplikaciju.

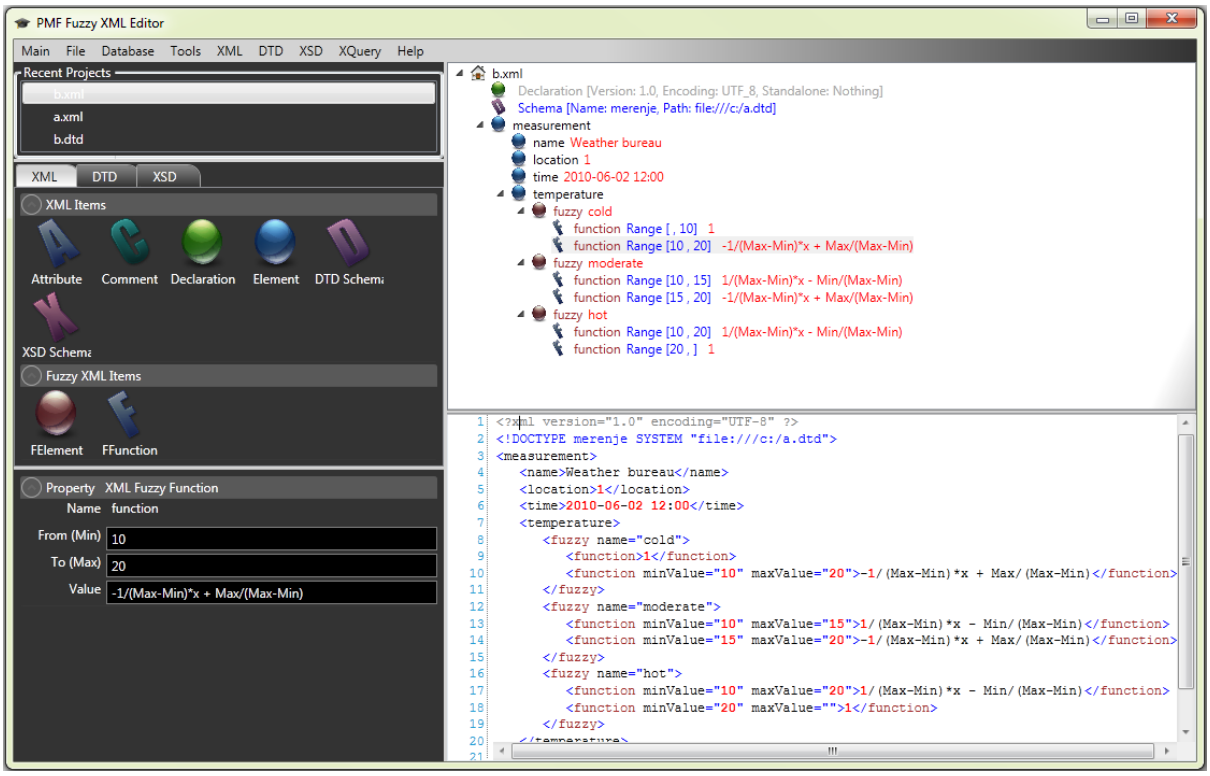
Drugi način za skladištenje dokumenata je upotreba baze podataka, koja je specijalno prilagođena za rad sa XML dokumentima. Prednost ovog pristupa u odnosu na rad sa fajlovima je postojanje ugrađenih upitnih jezika koji omogućuju jednostavnu i brzu pretragu dokumenata. Razvijena aplikacija se oslanja na Microsoft SQL Server 2008 R2 bazu podataka, zbog kompatibilnosti sa .NET tehnologijom. Ova verzija servera pored rada sa relacionim podacima nudi i mogućnost rada sa XML tipovima podataka. Upiti nad XML podacima mogu se vršiti upotrebom standardnog XQuery upitnog jezika. Da bi se omogućila veza sa bazom podataka potrebno je uneti konekcioni string u polje prikazano na slici 7.2.



Slika 7.2. Prozor za podešavanje konekcije na bazu podataka.

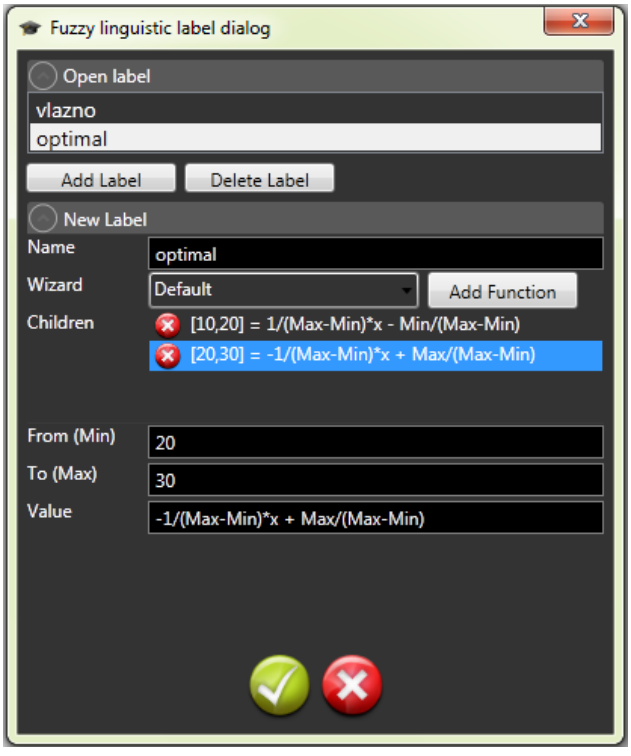
7.3 Fazi XML editor

Aplikacija olakšava kreiranje standardnih i fazi XML dokumenata koji koriste fazi XML sintaksu definisanu u prethodnim poglavljima. Korisniku se pruža mogućnost da upotrebom grafičkog okruženja definiše kako standardne XML elemente tako i fazi tipove elemenata. Primer kreiranja fazi XML dokumenta dat je na slici 7.3. Pored grafičkog prikaza (gore desno) postoji i tekstualni editor (dole desno) koji preslikava grafičku strukturu XML dokumenta u njen tekstualni oblik. Tekstualni prikaz se takođe može menjati po potrebi. Komponente koje je dozvoljeno koristiti u kreiranju dokumenta date su u *XML Items* i *Fuzzy XML Items* skupu alatki (levo). Osobine pojedinačnih elemenata moguće je podešavati u *Property* panelu (dole levo).



Slika 7.3. Kreiranje fazi XML-a.

Fazi vrednosti koje se često upotrebljavaju mogu biti definisane kao **lingvističke promenljive**. Da bi korisnik definisao lingvističku promenljivu potrebno je da u glavnom meniju pod stavkom XML, izabere *Fuzzy Linguistic Label* i otvara se dijalog prikazan na slici 7.4.



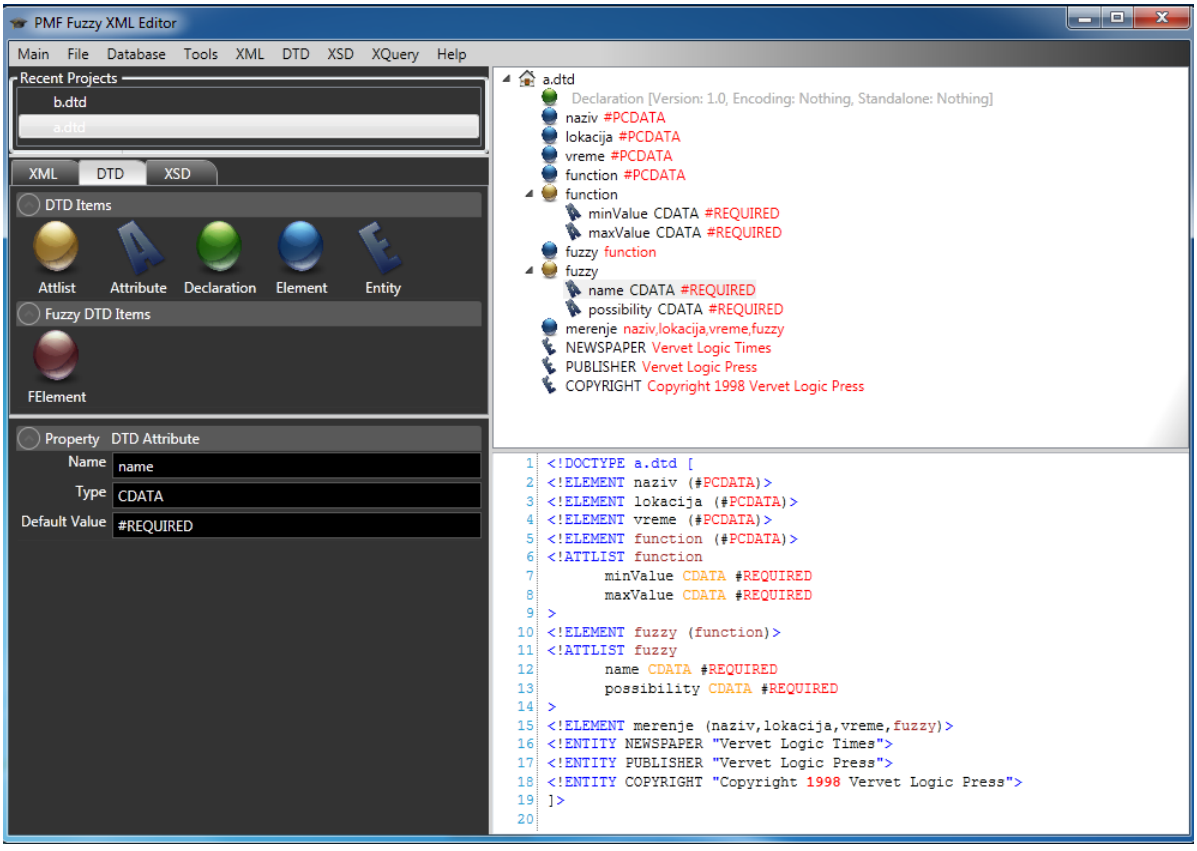
Slika 7.4. Dijalog za definisanje fazi lingvističke promenljive.

Korisniku se pruža mogućnost da sam definiše oblik funkcije pripadnosti fazi promenljive ili da koristi polje *wizard* sa predefinisanim funkcijama. Od predefinisanih tipova fazi vrednosti ponuđene su rastuća (*increasing*), opadajuća (*decreasing*), trougaona (*triangle*), trapezoidna (*trapez*) i interval (*interval*). Sve predefinisane vrednosti je moguće dodatno korigovati.

Za definisanje svake funkcije pripadnosti potrebno je odrediti njen minimum (polje *Min*), maksimum (polje *Max*) i funkciju nad tim intervalom (polje *Value*). Ukoliko minimum ili maksimum teže beskonačnostima polje ostaje prazno. Funkcija nad intervalom mora da zadovolji oblik *A*x + B*, pri čemu su *A* i *B* vrednosti koje mogu sadržati predefinisane konstante *Min* i *Max* koje predstavljaju minimalnu i maksimalnu vrednost intervala (videti polja *Min*, *Max* i *Value* na slici 7.4).

7.4 Fazi DTD editor

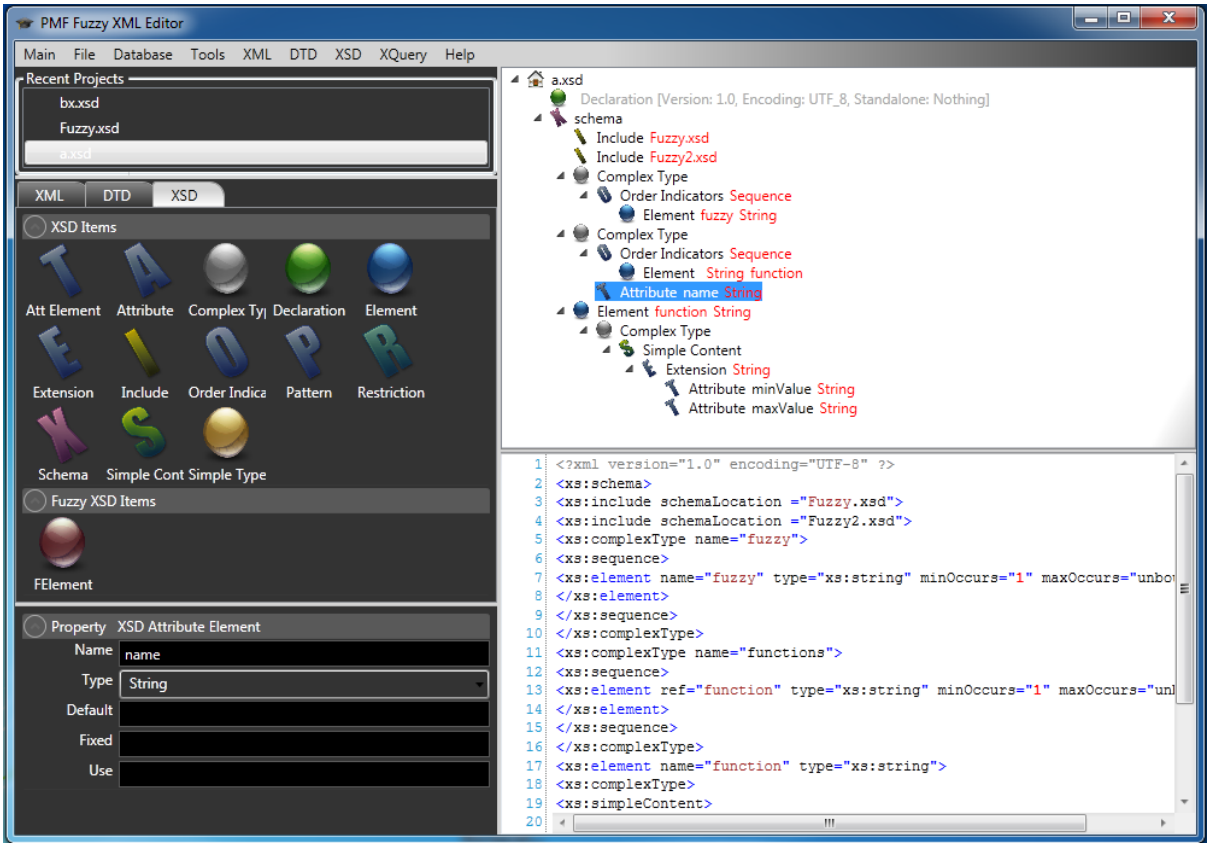
Aplikacija podržava rad sa standardnim i fazi DTD šema jezikom, čija je sintaksa opisana u prethodnim poglavljima. Kreiranje fazi DTD dokumenta radi se na sličan način kao kreiranje fazi XML elementa, sa tom razlikom što se u ovom slučaju koristi DTD skup alatki. U odnosu na standardnu DTD sintaksu, aplikacija omogućava definisanje novog tipa elementa, fazi elementa. Izgled aplikacije koja radi u modu za kreiranje fazi DTD dokumenta dat je na slici 7.5.



Slika 7.5. Kreiranje fazi DTD dokumenta.

7.5 Fazi XSD editor

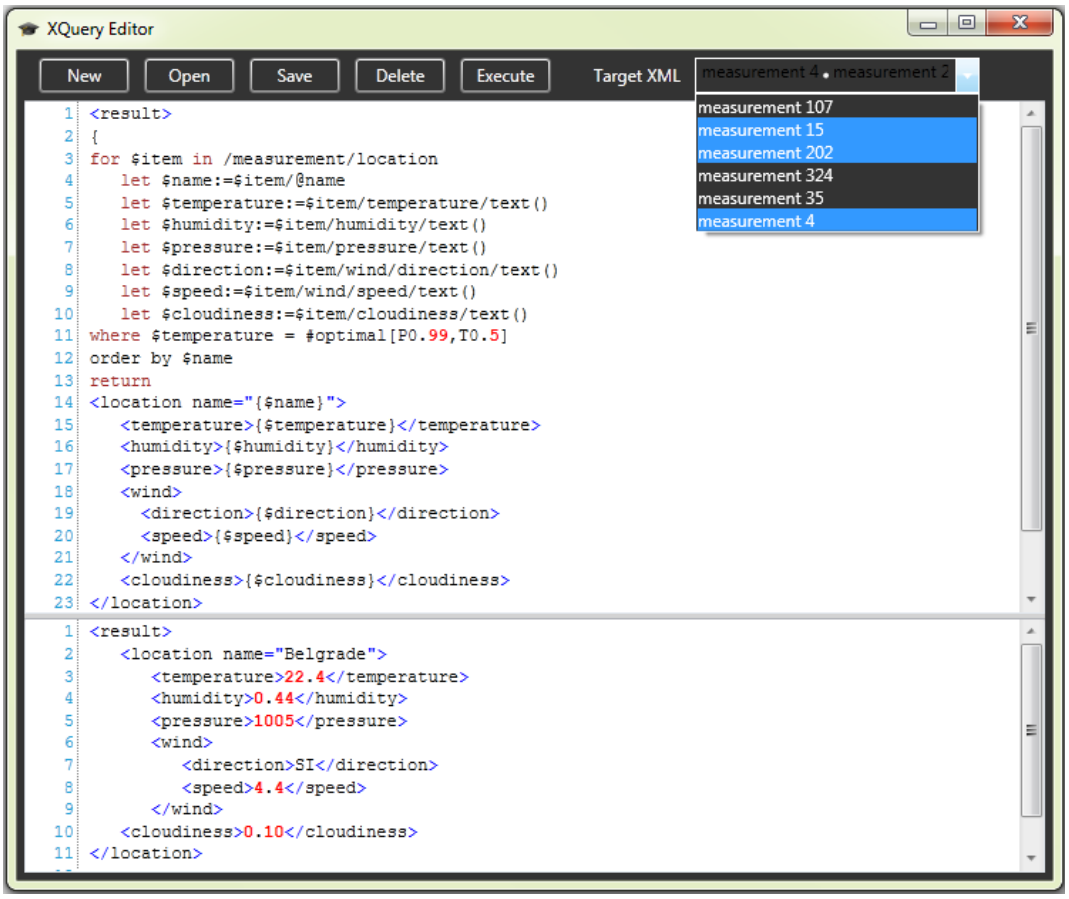
XSD predstavlja XML baziranu alternativu DTD dokumentima, sa nešto unapređenim mogućnostima. Aplikacija omogućava rad kako sa standardnim tako i sa fazi XSD jezikom, definisanim u prethodnim poglavljima. Kreiranje fazi XSD dokumenata radi se kao u prethodna dva slučaja, sa tom razlikom da se ovaj put koristi XSD skup alata. Postojanje *FElement* alatke za definisanje fazi elemenata predstavlja razliku u odnosu na standardnu XSD sintaksu. Izgled aplikacije pri kreiranju fazi XSD dokumenta dat je na slici 7.6.



Slika 7.6. Kreiranje fazi XSD dokumenta.

7.6 Fazi XQuery editor

Aplikacija nudi mogućnost rada sa standardnim i fazi XQuery upitima. Fazi XQuery jezik omogućava postavljanje upita koji sadrže neodređenost. Pokretanje modula vrši se klikom na *XQuery* stavku u glavnom meniju aplikacije, nakon čega se otvara prozor za rad sa XQuery izrazima. Prozor se sastoji iz dva dela, polja za unos upita (gore) i polja za ispis rezultata (dole). Fazi XQuery upite je moguće kreirati, učitavati, snimati, brisati i što je najbitnije izvršavati nad selektovanim XML dokumentima. Slika 7.7 prikazuje izvršavanje fazi XQuery upita u XQuery Editor-u.



Slika 7.7. Izvršavanje fazi XQuery upita u XQuery Editor-u.

Bitno je napomenuti da se upiti mogu izvršavati isključivo nad XML dokumentima uskladištenim u bazi podataka. Izvršavanje je moguće kako nad pojedinačnim dokumentima, tako i nad kolekcijom selektovanih XML dokumenata. Upiti se postavljaju u skladu sa prethodno definisanom fazi XQuery sintaksom, koja uključuje rad sa prioritetima i pragovima zadovoljenja.

Glava VIII

ANALIZA APLIKACIJE

Poglavlje se bavi analizom korisničkog interfejsa sa aspekta funkcionalnosti, testiranjem performansi i proverom stabilnosti aplikacije.

8.1 Evaluacija korisničkog interfejsa

Naučna oblast koja proučava interakcije između korisnika i računara naziva se HCI (Human Computer Interaction). HCI analizira performanse zadataka koje zajednički obavljaju ljudi i kompjuteri, strukturu komunikacije čovek-kompjuter, sociološku i organizacionu interakciju tokom projektovanja sistema, čovekove mogućnosti da koristi računar, algoritme i programiranje samog interfejsa, inženjerske probleme koji se pojavljuju tokom projektovanja i izgradnje interfejsa i procese specifikovanja, projektovanja, evaluacije i implementacije interfejsa.

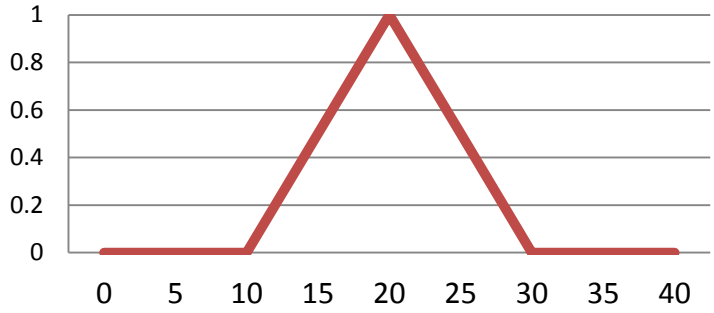
Interakcija između korisnika i računara definiše se kao korisnički interfejs, a može predstavljati kako hardver tako i softver. Cilj evaluacije korisničkog interfejsa je dobijanje informacija o tome koliko je aplikacija dobra u interakciji sa korisnicima. Neke od metoda za razvoj i analizu korisničkog interfejsa su:

- Modeliranje interakcije pomoću DTI, XUAN, GOMS i KLM
- Ekspertska revizija - heuristička evaluacija
- Testiranje utilitarnosti - test scenario, opservacija test subjekta, analiza i interpretacija dobijenih podataka, upitnik, analiza i interpretacija odgovora na upitnik i njihovo poređenje sa opažanjima sa testiranja
- Rezime svih rezultata dobijenih u prethodnim koracima i zaključak o samom programu

8.1.1 Modelovanje interakcije

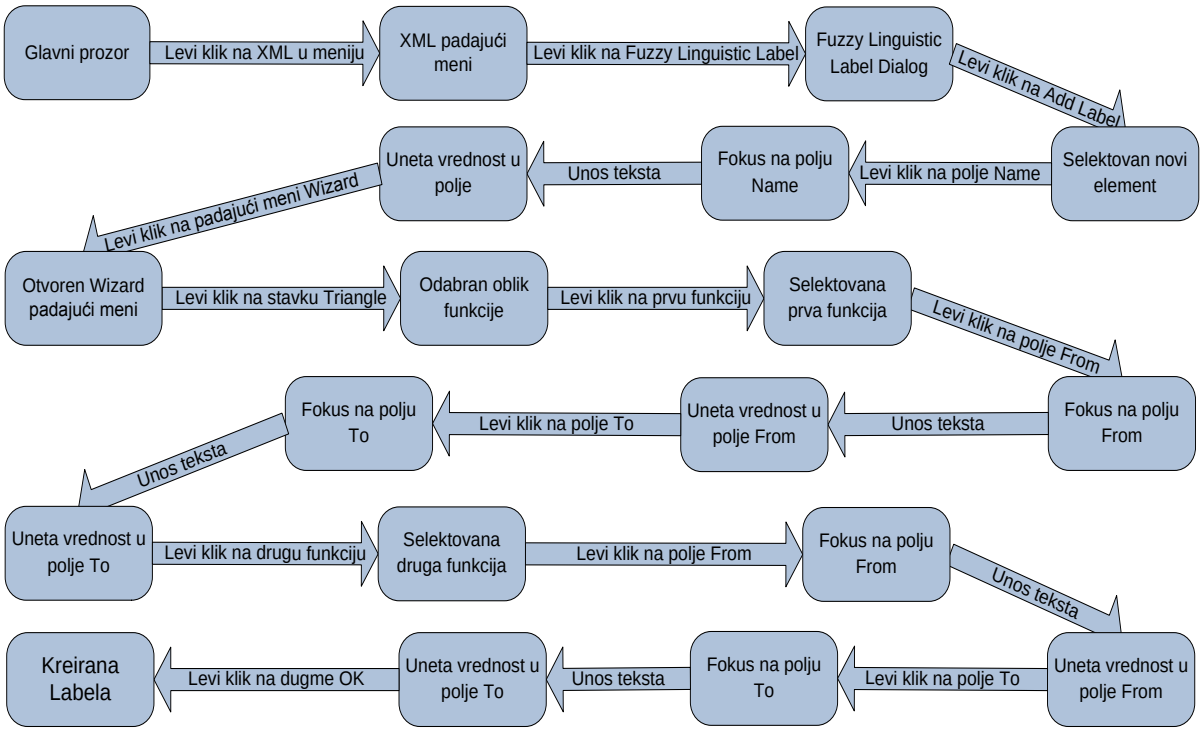
U nastavku je analizirano modelovanje interakcije DTI i KLM metodama. Zbog kompleksnosti modelovanja svih interakcija aplikacije, definisan je jedan karakterističan primer nad kojim je modelovana interakcija.

Primer za koji treba modelovati interakciju: "*Kreirati fazi lingvističku promenljivu pod imenom Test, čija je funkcija pripadnosti prikazana na slici 8.1*".



Slika 8.1. Funkcija pripadnosti fazi lingvističke promenljive *Test*.

DTI (Dijagrama toka interfejsa) predstavlja grafički prikaz toka interfejsa. Sastoji se od objekata i veza koje povezuju objekte. Objekti dijagrama toka interfejsa opisuju trenutno stanje interfejsa aplikacije. Veze sadrže opis akcija koje korisnik izvršava, a koje dovode do prelaska iz jednog stanja interfejsa u drugo. Zbog jednostavnog grafičkog prikaza pogodan je za prikaz toka interfejsa programa. Veličina dijagrama raste, a preglednost opada sa porastom broja mogućih stanja programa i načina izazivanja istih, zato nije pogodan za obimnije prikaze toka interfejsa. DTI dijagram za kreiranje fazi lingvističke promenljive *Test* prikazan je na slici 8.2.



Slika 8.2. DTI dijagram kreiranja fazi lingvističke promenljive *Test*.

Za dobijeni dijagram upotrebom KLM tehnike biće određena brzina kojom prosečan korisnik izvršava zadatak. Vreme koje je potrebno da bi korisnik izvršio neku standardnu operaciju, dosta govori o kvalitetu interakcije.

KLM (Keystroke Level Model) je uprošćena verzija CMN-GOMS varijacije. Šest standardnih operacija je predviđeno u osnovnoj teoriji, čije je vreme empirijski određeno. Operacije su organizovane u serijske nizove, a ukupno vreme izvršenja zadatka se jednostavno izračunava. Vrši se komparacija rešenja tokom izvršenja na nivou motorike, a postiže se operatorima za modelovanje koji mogu biti motorni, mentalni i odgovor sistema. Standardne vrednosti KLM operatora date su u tabeli 8.1.

Operator - Akcija	Napomena	Vreme u sekundama
K - pritisak na dati taster tastature	dobar (90 reči/min) srednji (40 reči/min) početnik	0,12 0,28 1,20
B - pritisak na taster miša	dole ili gore klik	0,10 0,20
P - pomeranje pointer uređaja na cilj	Fitt-ov zakon prosečno vreme	log2((rastojanje / veličina cilja)+0,5) 1,10
H - ruka sa/na tastature na/sa miša	na/sa tastature	0,40
D - crtanje pomoću pointer uređaja	crtanje	-
M - mentalna priprema za fizičku akciju	mentalna priprema	1,35
R- reakcija sistema	reakcija sistema	-

Tabela 8.1. Standardne vrednosti KLM operatora.

KLM akcije i vreme izvršenja za zadati primer:

- Postavljanje ruke na miša H<MIŠ>
- Pozicioniranje na meni *XML* u glavnom meniju MP
- Levi klik na stavku B
- Pozicioniranje na stavku *Fuzzy Linguistic Label* MP
- Levi klik na stavku B
- Pozicioniranje na dugme *Add Label* u otvorenom dijalogu MP
- Levi klik na dugme *Add Label* B
- Pozicioniranje na polje *Name* za unos imena MP
- Levi klik za dobijanje fokusa B
- Postavljanje ruke na tastaturu H<TAST>
- Unos teksta “Test” MK<4 znaka>
- Postavljanje ruke na miša H<MIŠ>
- Pozicioniranje na padajući meni *Wizard* MP
- Desni klik za otvaranje menija B
- Pozicioniranje na *Triangle* stavku u padajućem meniju MP
- Levi klik *Triangle* stavku B
- Pozicioniranje na prvu funkciju MP

- Levi klik na taster za selekciju elementa B
- Pozicioniranje na *From* polje za unos vrednosti MP
- Levi klik za selekciju fokusa B
- Postavljanje ruke na tastaturu H<TAST>
- Unos teksta “10” MK<2 znaka>
- Postavljanje ruke na miša H<MIŠ>
- Pozicioniranje na *To* polje za unos vrednosti MP
- Levi klik za selekciju fokusa B
- Postavljanje ruke na tastaturu H<TAST>
- Unos teksta “20” MK<2 znaka>
- Postavljanje ruke na miša H<MIŠ>
- Pozicioniranje na drugu funkciju MP
- Levi klik na taster za selekciju elementa B
- Pozicioniranje na *From* polje za unos vrednosti MP
- Levi klik za selekciju fokusa B
- Postavljanje ruke na tastaturu H<TAST>
- Unos teksta “20” MK<2 znaka>
- Postavljanje ruke na miša H<MIŠ>
- Pozicioniranje na *To* polje za unos vrednosti MP
- Levi klik za selekciju fokusa B
- Postavljanje ruke na tastaturu H<TAST>
- Unos teksta “30” MK<2 znaka>
- Postavljanje ruke na miša H<MIŠ>
- Pozicioniranje na taster *OK* MP
- Levi klik na taster *OK* B

$T_{H<MIŠ>} + T_{MP} + T_B + T_{MP} + T_B + T_{MP} + T_B + T_{MP} + T_B + T_{H<TAST>} + 4\,T_{MK} + T_{H<MIŠ>} + T_{MP} + T_B + T_{MP} + T_B + T_{MP} + T_B + T_{MP} + T_B + T_{H<TAST>} + 2\,T_{MK} + T_{H<MIŠ>} + T_{MP} + T_B + T_{H<TAST>} + 2\,T_{MK} + T_{H<MIŠ>} + T_{MP} + T_B + T_{MP} + T_B + T_{H<TAST>} + 2\,T_{MK} + T_{H<MIŠ>} + T_{MP} + T_B + T_{H<TAST>} + 2\,T_{MK} + T_{H<MIŠ>} + T_{MP} + T_B$

$= 13\,T_{MP} + 13\,T_B + 12\,T_{MK} + 6\,T_{H<MIŠ>} + 5\,T_{H<TAST>}$

$= 25\,T_M + 13\,T_P + 12\,T_K + 13\,T_B + 11\,T_H$

= 25*1,35 + 13*1,10 + 12*0,28 + 13*0,20 + 11*0,40

= 33.75 + 14.3 + 3.36 + 2,6 + 4,4

= **58.41 s**

Zaključak: Prosečnom korisniku je potrebno manje od minuta da bi kreirao fazi lingvističku promenljivu opisanu slikom 8.1. Dobijeno vreme očigledno predstavlja dobar rezultat, sa obzirom da kreiranje lingvističke promenljive prema odgovarajućem opisu predstavlja jednu od najkomplikovanijih akcija aplikacije.

8.1.2 Ekspertska revizija

Ekspertska revizija predstavlja vršenje ekspertize u korisničkom interfejsu ili aplikacionom domenu. Sprovode je stalni članovi tima ili konsultanti u ranim ili kasnim fazama evaluacije programa. Sadrži formalne izveštaje sa identifikovanim problemima i

preporukama za izmene, a projektantima se ostavlja da tragaju za konkretnim rešenjem problema. Izvršenje ekspertske revizije traje od pola dana do jedne nedelje što zavisi od domena zadatka i operativnih procedura.

Metode ekspertske revizije:

- **Heuristička evaluacija** - Predstavlja određivanje slaganja sa kratkom listom projektantskih heuristika (npr. 8 zlatnih pravila). Poželjno je da eksperti poznaju heuristike i da su u stanju da ih interpretiraju i primene. Sprovodi se u timu od 3-5 ljudi. Jedan član je u stanju da otkrije do 35% grešaka, a 5 članova i do 75% grešaka. Vreme predviđeno za heurističku evaluaciju jednostavnog interfejsa iznosi sat vremena.
- **Revizija po smernicama** - Određivanje slaganja sa dokumentom smernica organizacione ili neke druge prirode. Revizija je vremenski zahtevna i traje do par nedelja za velike sisteme.
- **Inspekcija konzistentnosti** - Određivanje konzistentnosti između više interfejsa, materijala za obuku i sistema pomoći (korišćenih terminologija, boja, izgleda, ulaznih i izlaznih formata)
- **Cognitive walkthrough** - Eksperti su korisnici u tipičnom (ali i kritičnom) zadatku. Karakterističan je samostalni *explore walkthrough,* a kasnije i grupni sa ostalim ekspertima ili projektantima.
- **Formalna inspekcija utilitarnosti** – Karakterističan je stil sastanaka nalik sudnici, sa moderatorom kao sudijom. Vrši se predstavljanje interfejsa, radi diskutovanja o prednostima i slabostima aplikacije. Edukativno je za početnike u projektovanju i menadžmentu, ali je potrebno puno vremena za pripremu i više ljudi nego u drugim metodama.

Metoda ekspertske revizije koja će biti korišćena je metoda heurističke evaluacije, a biće sprovedeno određivanje slaganja sa listom projektantskih heuristika, u ovom slučaju metodom "8 zlatnih pravila".

Analiza slaganje sa 8 zlatnih pravila:

Težiti konzistentnosti: Aplikacija koristi identičnu terminologiju u svim elementima interfejsa. Konzistentnost boja je ostvarena, prevladavaju identične nijanse sive u svim dijalozima. Oblik i boja prozora formiraju se na osnovu podešavanja operativnog sistema koja je korisnik postavio. Fontovi su takođe konzistentni i čitljivi, ali ne postoji mogućnost promene veličine slova na nivou same aplikacije.

Omogućiti korisnicima upotrebu prečica: Od prečica aplikacija podržava samo one za rukovanje glavnim menijem. Sa obzirom da se radi o nekoj vrsti tekstualnog editora i da su korisniku ruke uglavnom na tastaturi, uvođenje dodatnih prečica dovelo bi do povećanja brzine interakcije.

Davati povratnu informaciju: Za svaku akciju korisnika predviđena je povratna informacija. Frekventnije akcije se predstavljaju vizuelnom promenom na dijagramu, dok se za manje frekventne akcije prikazuje obaveštenje. Primetan je nedostatak odgovarajuće povratne informacije u nekim dijalozima aplikacije.

Projektovati dijaloge naglašene zatvorenosti: Sekvence akcija su dobro grupisane, a dijalozi intuitivno odrađeni bez suvišnog pretrpavanja informacijama. Zatvorenost je bilo lako postići zbog jednostavnosti operacija koje se izvršavaju. Snimanje trenutnog stanja je

moguće u svakom trenutku, kako skladištenjem u bazi podataka, tako i snimanjem u nezavisne datoteke.

Ponuditi prevenciju i rukovanje greškom: Prevencija grešaka je odrađena nad poljima za unos vrednosti i omogućena je jednostavna korekcija u slučaju pogrešnog unosa.

Dozvoliti poništavanje efekata akcije: Aplikacija ne poseduje takvu mogućnost. Na prethodna stanja se može vraćati brisanjem prethodne akcije ili učitavanjem dokumenta iz memorije. Ovo predstavlja ozbiljniji nedostatak aplikacije.

Interno podržavati kontrolu: Korisniku je omogućena potpuna kontrola nad aplikacijom. Za prosečnu upotrebu ne postoje operacije za koje je potrebna veća procesorska snaga, a samim tim i više vremena za obradu. Izuzetak iz ove priče je izvršenje XQuery upita, što u slučaju obrade nad velikom kolekcijom dokumenata može da potraje. Korisniku se u tom slučaju menja status pokazivača u ikonicu za čekanje.

Redukovati opterećenje radne memorije: Glavni meniji aplikacije deluju rasterećeno (broj elemenata ponekad ide ispod pravila 7 ± 2), dobro je uravnotežen i prilagođen je zahtevima softverskog paketa. Skup alatki organizovan je po grupama u kojima su elementi sortirani po abecednom redosledu, čime se olakšava njihovo pronalaženje. Celokupna aplikacija je podeljena na više radnih površina, čije se dimenzije mogu menjati.

Zaključak: Na osnovu svega navedenog može se reći da aplikacija zadovoljava dobar deo pravila. Najveći problem je nedostatak mogućnosti poništavanja efekata akcije (*undo*). Takođe, prostora za poboljšanje ima u definisanju novih prečica za ubrzanje rada iskusnijih korisnika.

8.1.3 Testiranje utilitarnosti

Utilitarnost predstavlja stepen kojim interfejs olakšava rešavanje zadataka i inkorporira kriterijume kao što su lakoća učenja, lakoća upotrebe, zaštita od katastrofalnih grešaka i mera podrške korisniku. Određivanje nivoa utilitarnosti vrši se jednostavnim testiranjem korisnika. Na ovaj način se mogu uočiti metode kojima se služi korisnik pri rešavanju zadataka i probleme sa kojima se sreće u interakciji sa programom. U metode za testiranje korisnika ubrajaju se: razvoj i implementacija scenarija ili prototipa, snimanje ponašanja korisnika, intervjuisanje o subjektivnim impresijama i analiza ponašanja korisnika. Prilikom samog testiranja potrebno je poštovati principe etike: ne sme biti fizičkih i psihičkih povređivanja, pravo na privatnost, potpis na saglasnost, anonimnost i mogućnost odustajanja u svakom trenutku.

Testiranje korisnika je izvršeno u nekoliko standardnih koraka i obrazloženo u nastavku.

Podesiti ambijent za opservaciju: Testiranje se vrši u kancelariji, računar se nalazi na stolu. Temperatura je oko 25C^{o}, za osvetljenje se koristi dnevna svetlost, nivo buke je nizak, ispod 25dB.

Opisati svrhu opservacije i saopštiti korisniku da ima pravo odustati u svakom trenutku: Test-korisniku je saopšteno da je svrha opservacije da se odredi faktor intuitivnosti u radu sa softverskim paketom *Fuzzy XML Editor*. Test-korisniku se nudi opcija da odustane od testiranja i saopštava mu se da to isto može da učini u toku testiranja bez posledica i suvišnih pitanja.

Opisati i demonstrirati opremu u prostoriji: Test-korisnik se upoznaje sa računarom. Računar je laptop HP-Pavilion sa Core2Duo P8600 procesorom na 2.40GHz i 4GB memorije. Na računaru je instaliran Microsoft Windows 7 operativni sistem i pokrenuta aplikacija koja se testira.

Objasniti kako se "misli naglas" i da neće biti pružana nikakva pomoć tokom testa: Test-korisnik se upoznaje sa činjenicom da bi bilo poželjno da svoja razmišljanja iznosi naglas. Pre nego što nešto uradi, naglas treba da izgovori šta želi i šta očekuje od te akcije. Skrenuta je pažnja da neće biti pružena nikakva pomoć tokom testiranja.

Opisati zadatak i uvesti korisnika u sistem: Zadatak koji se zadaje korisniku ima za cilj da proveri sposobnost njegovog snalaženja u softverskom paketu *Fuzzy XML Editor*. Zadaci su grupisani u tri grupe, po težini. Svi test korisnici dobijaju iste test zadatke. Naravno, ne očekuje se od svih isto. U slučaju da se ne kompletira zadatak pod A, ne može se preći na zadatak pod B itd. Tekst zadatka se navodi u celosti u nastavku teksta.

A) Početnički

- kreirati novi fazi XML dokument
- dodati nekoliko elemenata
- elementima promeniti imena i vrednost
- proizvoljnom elementu dodati atribut
- proizvoljnom elementu dodati pod-elemente
- sačuvati dokument

B) Osnovni

- učitati kreirani fazi XML dokument
- dodati fazi element i nad njim definisati trougaonu funkciju pripadnosti
- izmeniti interval funkcije pripadnosti
- sačuvati dokument
- kreirati DTD element i dodati par elemenata, od kojih bar jedan mora da bude *fuzzy* element
- sačuvati DTD dokument
- kreirati XSD dokument i dodati par elemenata, od kojih bar jedan mora da bude *fuzzy* element
- sačuvati XSD dokument

C) Napredni

- pokrenuti *XQuery editor*, klikom na *XQuery* stavku u glavnom meniju
- kreirati jednostavan XQuery upit koji će se izvršiti nad kreiranim XML dokumentom
- kreirati fazi lingvističku promenljivu pomoću odgovarajućeg dijaloga, kome se pristupa klikom na stavku *XML > Fuzzy Linguistic Label* u glavnom meniju
- upotrebiti kreiranu fazi lingvističku promenljivu u *WHERE* uslovu

Insistirati da vas korisnik pita pre opservacije: Korisniku je skrenuta pažnja, da ukoliko nešto nije jasno pita pre opservacije, kako se ne bi narušavala pravila o pružanju pomoći tokom same opservacije.

Opservacija korisnika: Vršena je opservacija nad tri korisnika, sa dobrim poznavanjem rada na računaru. Prvi korisnik ne poznaje ni XML tehnologiju ni XQuery upitni jezik, dok ostali poznaju XML, ali ne i XQuery.

Prvi korisnik je imao problema na samom početku, kreirao je XML dokument, dodavao je nasumično elemente, ali nije razumeo njihov smisao. Prvu grupu zadataka je rešio u potpunosti, ali uz dosta muke. U drugoj grupi nije znao kako da definiše trougaonu funkciju pripadnosti i tu je odustao.

Drugi korisnik je iskusan programer i sa lakoćom je rešio prvu grupu zadataka. U drugoj grupi imamo je problema u radu sa DTD i XSD šemom, pošto se nije najbolje sećao sintakse. Kod treće grupe zadataka je odustao pošto ne poznaje XQuery sintaksu i nije umeo da napiše ni najprostiji upit.

Treći korisnik je programer početnik i takođe je uspeo da reši prve dve grupe zadataka, dok mu je u trećoj grupi problem predstavljalo nepoznavanje XQuery jezika. Odustao je zbog nemogućnosti da napiše upit.

Analizirati dobijene podatke: Svi korisnici su ispunili očekivanja. Od korisnika koji ne poznaje XML nije se ni moglo očekivati da uradi nešto više. Kod ostalih korisnika nije postojala nikakva šansa da napišu XQuery upit jer ne poznaju sintaksu.

Organizovati rezultate tabelarno: Procentima je predstavljena vrednost rešenih zadataka. Oznaka (–) znači da testiranja nije ni bilo, pošto korisnik nije prošao prethodni test.

	Prvi	**Drugi**	**Treći**
A - Početni	100%	100%	100%
B - Osnovni	12.5%	100%	100%
C - Napredni	-	25%	25%
Ukupno	35.5%	75%	75%
Očekivano	< 33%	> 66%	> 66%
Zadovoljio	Da	Da	Da

Tabela 8.2. Analiza rezultata.

Interpretirati rezultate u kontekstu drugih rezultata: Presudnu ulogu u rešavanju zadataka odigrala je sličnost softverskog paketa sa ostalim programerskim alatima. Činjenica da aplikacija sadrži grafički i tekstualni editor, listu projekata, skup alatki i prozor za podešavanje osobina selektovanog elementa, pomogla je programerima da se lako orijentišu.

Uočeno je da je za rad sa aplikacijom neophodno da korisnik poznaje sintakse XML, DTD, XSD i XQuery. Ne postoji odgovarajuća pomoć na nivou same aplikacije koja bi pomogla korisnicima da na licu mesta savladaju pomenute sintakse. Sve ovo govori da je aplikacija namenjena za grupu korisnika koji dobro poznaju navedene sintakse, a koji u stvari i jesu ciljna grupa.

8.2 Testiranje performansi sistema

Izvršeno je testiranje performansi aplikacije radi boljeg sagledavanja mogućnosti kreiranih sintaksi i kvaliteta implementacije.

Da bi uporedili brzinu izvršavanja fazi XQuery upita u odnosu na standarde XQuery upite, izvršeno je **testiranje performansi sistema**. Testovi su ponovljeni više puta, a prosečni rezultati upisani u tabele. Za testiranje se koristi kolekcija test XML dokumenata, sličnog sadržaja kao dokument iz listinga 5.1. Dokumenti imaju veličinu od 1KB do 1GB i smešteni su u bazi podataka. Nad njima se izvršavaju fazi XQuery upiti sa prioritetima nalik onom iz listinga 5.8. Testiranja su vršena na računaru sa procesorom Intel Core2 Duo na 2.4GH sa 4GB RAM memorije.

Performanse aplikacije nisu poređenje u odnosu na druge slične aplikacije, pošto nismo uspeli da pronađemo odgovarajuće implementacije. Vrši se poređenje brzine izvršenja standardnog XQuery upita sa brzinom izvršenja fazi XQuery upita sa prioritetima po nekoliko kriterijuma. Oba tipa upita se izvršavaju pomoću aplikacije koja se testira. Standardni XQuery upiti sadrže diskretne vrednosti dok fazi XQuery upiti sadrže identične vrednosti, samo su one ovog puta fazi tipa.

Vreme izvršenja upita u odnosu na broj različitih fazi vrednosti u njemu prikazano je u tabeli 8.3. Opadanje performansi aplikacije sa porastom broja promenljivih izraženije je kod fazi XQuery upita. To se dešava zbog procesa defazifikacije fazi promenljivih.

1MB veličina fajla	1 fazi prom.	3 fazi prom.	5 fazi prom.	10 fazi prom.
Standardni XQuery	0.5s	0.5s	0.5s	1s
Fazi XQuery	1.5s	2s	3s	5s

Tabela 8.3. Složenost upita / vreme izvršavanja.

Tabela 8.4 daje uporedni prikaz vremena potrebnog za izvršenje upita u odnosu na veličinu XML dokumenta nad kojim se upit izvršava. Standardni XQuery je brži, ali se ta razlika značajno smanjuje sa porastom veličine XML dokumenta.

1 fazi promenljiva	1KB	100KB	1MB	20MB	100MB
Standardni XQuery	<0.1s	<0.1s	0.5s	11s	Timeout
Fazi XQuery	1s	1.2s	1.5s	15s	Timeout

Tabela 8.4. Veličina datoteka / vreme izvršavanja.

U tabeli 8.5 dat je prikaz vremena potrebnog za izvršenje upita nad kolekcijom XML dokumenata. Standardni XQuery je brži, ali razlika postaje zanemarljiva sa porastom broja dokumenata.

1MB veličina, 1 fazi prom.	1 Dokument	10 Dok.	100 Dok.	1000 Dok.
Standardni XQuery	0.5s	7s	62s	630s
Fazi XQuery	1.5s	8s	68s	643s

Tabela 8.5. Broj dokumenata / vreme izvršavanja.

Testiranjem performansi aplikacije pokazano je da je fazi XQuery upit sporiji od standardnog XQuery upita, ali ima zadovoljavajuće performanse. Rad sa većim brojem dokumenata i datotekama veće veličine su očekivani u praksi, a u tom segmentu fazi XQuery upiti rade neznatno sporije od standardnih. Ovim je pokazano da je implementacija izvedena dobro, a vreme koje se troši na defazifikaciju upita ne utiče puno na brzinu njihovog izvršavanja.

8.3 Provera stabilnosti

Load (stres) testiranjem došlo se do zaključka da aplikacija radi stabilno pri radu sa standardnim i fazi XML, DTD i XSD dokumentima i XQuery upitima. Učitavanje većih dokumenata usporava aplikaciju što je očekivano. Izvršavanje XQuery upita nad XML dokumentima, čija veličina pojedinačno prelazi 100MB, dovodi do pojave *Timeout* greške. Ovo se može smatrati problemom i trebalo bi uraditi neku vrstu optimizacije i deljenja velikih dokumenata u procesu obrade.

Glava IX

MOGUĆNOSTI PRAKTIČNE PRIMENE

Poglavlje prikazuje primere koji na slikovit način opisuju mogućnosti primene sintaksi i razvijenog softverskog paketa na realnim sistemima. Kao primeri iz prakse uzeti su:

- Košarka – selekcija igrača za košarkašku utakmicu
- Medicina – određivanje stanja pacijenata analizom XML–CCR dokumenata

9.1 Košarka - selekcija igrača za košarkašku utakmicu

Pretpostavimo da trener želi da odabere kandidate koji će se najbolje uklopiti u planiranu taktiku za predstojeću utakmicu. Na raspolaganju ima XML dokumenat koji sadrži informacije o pojedinačnom učinku za igrača na utakmicama u toku sezone. Podaci su preuzeti u toku sezone sa zvaničnog sajta KK Partizan, nakon 20 odigranih kola. Pitanje može da glasi: "*Pronaći igrače koji mogu igrati na poziciji centra, u situaciji kada tim bazira svoju taktiku na dominaciji pod košem, kao i sigurnom napadu preko centarske pozicije. Igrači koji bi zadovoljili navedene karakteristike moraju da budu dobri u skoku u odbrani i napadu. Takođe, poželjno bi bilo da igrač ima dobar procenat šuta za dva poena i sa linije slobodnih bacanja*".

Deo XML dokumenta, koji sadrži podatke o učinku igrača na utakmicama u dosadašnjem toku sezone, dat je u listingu 9.1.

```
<ABALiga datum="2013-02-05">
    ...
    <KKPartizan>
        ...
        <igrac ime="Gordon Drew Edward" pozicija="KC" kolo="16"
            domacin="true" protivnik="KKCibona">
            <minutaza>31</minutaza>
            <poena>
                <ukupno>11</ukupno>
                <sutza1>0.75</sutza1>
                <sutza2>0.80</sutza2>
                <sutza3>0</sutza3>
            </poena>
            <skok>
                <napad>2</napad>
                <odbrana>5</odbrana>
            </skok>
```

```
            <asistencije>2</asistencije>
            <osvojenelopte>3</osvojenelopte>
            <izgubljenelopte>0</izgubljenelopte>
            <licnegreske>3</licnegreske>
            <blokade>1</blokade>
        </igrac>
        ...
    </KKPartizan>
    ...
</ABALiga>
```

Listing 9.1. Struktura XML dokumenta.

Prethodni primer u delu dokumenta koji je prikazan daje učinak za samo jednog igrača (u ovom slučaju Gordon Drew Edward) na jednoj utakmici (KK Partizan – KK Cibona, kolo 16), dok je ostatak dokumenta analogan. Radi lakšeg praćenja rezultata upita u tabeli 9.1 dat je sumirani pregled rezultata postignutih u posmatranom delu sezone, dobijen analizom dokumenta iz listinga 9.1. Vrednost *N* znači da nije ni bilo pokušaja.

KK PARTIZAN	Pozicija	Broj utakmica	Minutaža	Poena ukupno	Slobodna bacanja	Šut za dva poena	Šut za tri poena	Skok napad	Skok odbrana	Asistencija	Osvojene lopte	Izgubljene lopte	Lične greške	Blokade
Anđušić Danilo	B	12	128	69	87%	43%	32%	4	6	13	5	12	11	1
Bertans Davis	K	20	434	166	72%	49%	43%	13	38	15	18	20	47	9
Bogdanović Bogdan	B	16	255	66	67%	47%	18%	5	26	19	14	20	35	2
Čakarević Marko	K	17	90	19	62%	54%	0%	4	16	4	3	8	13	3
Đekić Branislav	KC	7	65	20	80%	37%	25%	4	3	2	1	1	7	2
Gagic Đorđe	C	18	220	121	69%	60%	0%	26	28	3	9	33	65	6
Gordon Drew	KC	20	536	198	67%	49%	9%	51	108	23	23	56	44	16
Lučić Vladimir	K	17	463	215	79%	52%	35%	17	59	15	23	27	41	0
Milosavljević Dragan	B	20	453	144	70%	57%	33%	20	42	32	9	35	51	3
Milutinov Nikola	C	20	202	43	56%	45%	N	18	18	2	3	9	40	1
Musli Dejan	C	19	329	138	73%	56%	N	19	76	21	11	33	43	23
Thomas Torey	P	6	107	37	60%	44%	27%	7	9	15	5	8	12	0
Westermann Leo	P	20	480	138	71%	45%	21%	11	42	43	24	52	56	2

Gordić Nemanja	P	12	199	73	69%	40%	43%	4	8	19	6	12	24	0
Cekovsky Michal	KC	0	0	0	N	N	N	0	0	0	0	0	0	0
Lauvergne Joffrey	C	5	39	14	75%	80%	0%	2	5	3	0	3	7	1

Tabela 9.1. Sumirani pregled kolekcije XML dokumenata.

Standardni XQuery upit, koji bi odgovarao postavljenom pitanju, dat je u listingu 9.2.

```
<rezultat>
{
for $item in /ABALiga/KKPartizan/igrac
            let $ime:=$item/@ime
            let $pozicija:=$item/@pozicija
            let $sutza1:=
                  avg( for $x2 in /ABALiga/KKPartizan/igrac
                  let $i2:= $x2/@ime
                  let $j2:= $x2/poena/sutza1
                  where $i2 = $ime
                  return $j2)
            let $sutza2:=
                  avg( for $x3 in /ABALiga/KKPartizan/igrac
                  let $i3:= $x3/@ime
                  let $j3:= $x3/poena/sutza2
                  where $i3 = $ime
                  return $j3)
            let $napad:=
                  avg( for $x4 in /ABALiga/KKPartizan/igrac
                  let $i4:= $x4/@ime
                  let $j4:= $x4/skok/napad
                  where $i4 = $ime
                  return $j4)
            let $odbrana:=
                  avg( for $x5 in /ABALiga/KKPartizan/igrac
                  let $i5:= $x5/@ime
                  let $j5:= $x5/skok/odbrana
                  where $i5 = $ime
                  return $j5)
      where ($pozicija = "C" OR $pozicija = "KC") AND
            $napad > 1 AND
```

```
            $odbrana > 1 AND
            $sutza1 >= 0.6 AND
            $sutza2 >= 0.6
      order by $ime
      return
      <igrac>{$ime}</igrac>
}
</rezultat>
```

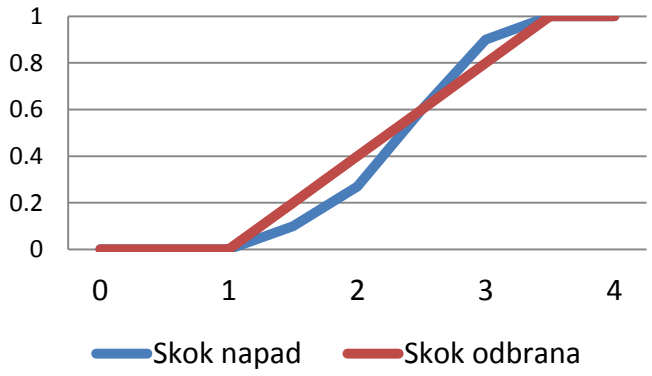
Listing 9.2. XQuery upit.

Izvršavanjem upita iz listinga 9.2 nad dokumentom iz listinga 9.1 dobija se rezultat dat u listingu 9.3.

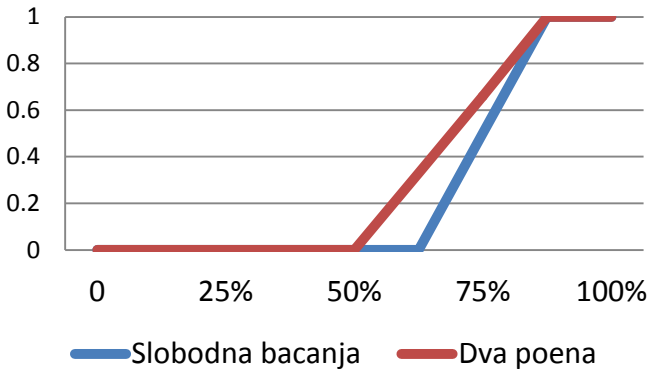
```
<rezultat>
      <igrac>Gagic Đorđe</igrac>
</rezultat>
```

Listing 9.3. Rezultat standardnog XQuery upita.

Postavljeni problem se preciznije može rešiti upotrebom fazi XQuery upita. Prvo je neophodno da korisnik definiše potrebne funkcije pripadnosti za promenljive iz upita. Funkcije pripadnosti bi mogle izgledati kao na slikama 9.1 i 9.2.



Slika 9.1. Funkcije pripadnosti za dobru skok igru.



Slika 9.2. Funkcije pripadnosti za dobar šut.

Nakon toga na osnovu definisanog pitanja postavlja se fazi XQuery upit sa prioritetima. Primer takvog upita dat je u listingu 9.4.

```
<rezultat>
{
for $item in /ABALiga/KKPartizan/igrac
            let $ime:=$item/@ime
            let $pozicija:=$item/@pozicija
            let $sutza1:=
                  avg( for $x2 in /ABALiga/KKPartizan/igrac
                  let $i2:= $x2/@ime
                  let $j2:= $x2/poena/sutza1
                  where $i2 = $ime
                  return $j2)
            let $sutza2:=
                  avg( for $x3 in /ABALiga/KKPartizan/igrac
                  let $i3:= $x3/@ime
                  let $j3:= $x3/poena/sutza2
                  where $i3 = $ime
                  return $j3)
            let $napad:=
                  avg( for $x4 in /ABALiga/KKPartizan/igrac
                  let $i4:= $x4/@ime
                  let $j4:= $x4/skok/napad
                  where $i4 = $ime
                  return $j4)
            let $odbrana:=
                  avg( for $x5 in /ABALiga/KKPartizan/igrac
                  let $i5:= $x5/@ime
```

```
                let $j5:= $x5/skok/odbrana
                where $i5 = $ime
                return $j5)
        where ($pozicija = "C" OR $pozicija = "KC") AND
              $napad = #dobar_skok_napad AND
              $odbrana = #dobar_skok_odbrana AND
              $sutza1 = #solidan_sut1[P0.6,T0.8] AND
              $sutza2 = #solidan_sut2[P0.6,T0.8]
        order by $ime
        return
        <igrac>{$ime}</igrac>
}
</rezultat>
```

Listing 9.4. Fazi XQuery upit sa prioritetima.

Kada se definisani upit izvrši nad dokumentom iz listinga 9.1, dobije se rezultat dat u listingu 9.5.

```
<rezultat>
        <igrac>Gagic Đorđe</igrac>
        <igrac>Gordon Drew</igrac>
        <igrac>Musli Dejan</igrac>
</rezultat>
```

Listing 9.5. Rezultat fazi XQuery upita.

Ukoliko bi postavili klasičan upit bez uzimanja u obzir prioriteta i pragova zadovoljenja, dobijeni rezultat ne bi sadržao neka relativno dobra rešenja. Tako bi iz rezultata bili izbačeni *Gordon* i *Musli*, zbog lošijeg šuta za dva poena. Preostali igrači sa centarskih pozicija *Đekić*, *Milutinov* i *Lauvergne* nisu prošli ni u fazi XQuery upitu, jer nisu ispunili uslove vezane za skok igru, koji su definisani kao uslovi sa najvišim prioritetima i moraju biti zadovoljeni u potpunosti (100%).

9.1.1 Obrada Null vrednosti

Šta bi se desilo kada bi se u razmatranje uveli mogućnost pojave Null vrednosti tipa *nepoznata* i *neprimenljiva*? Pretpostavimo da nedostaju neke informacije za određene igrače na određenim utakmicama. Recimo da je igrač igrao utakmicu, ali iz nekog razloga pojedine informacije nisu upisane. Takve vrednosti se mogu smatrati Null vrednostima tipa **nepoznata**. Primer elementa tipa *igrac* koji sadrži takve informacije dat je u listingu 9.6.

```
<igrac ime="Gordon Drew Edward" pozicija="KC" kolo="16"
domacin="true" protivnik="Cibona">
    <minutaza>31</minutaza>
    <poena>
        <ukupno>11</ukupno>
        <sutza1>0.75</sutza1>
        <sutza2>0.80</sutza2>
        <sutza3>0</sutza3>
    </poena>
    <skok>
        <napad></napad>
        <odbrana></odbrana>
    </skok>
    <asistencije>2</asistencije>
    <osvojenelopte>3</osvojenelopte>
    <izgubljenelopte>0</izgubljenelopte>
    <licnegreske>3</licnegreske>
    <blokade>1</blokade>
</igrac>
```

Listing 9.6. Primer statistike igrača na utakmici sa nekompletnim podacima.

Vidi se da u listingu 9.6 nedostaju informacije o broju skokova u odbrani i napadu za navedenog igrača. Kako je aplikacija bazirana na Misrosoft SQL Server-u i proširenju XQuery jezika implementiranog u njemu, prosečne vrednosti se mogu dobiti upotrebom standardne *avg* funkcije, na način prikazan u XQuery upitu iz listinga 9.4. Upotrebom navedenog upita, SQL Server XQuery implementacija iz dalje obrade sama automatski izbacuju vrednosti koje nedostaju, ali ne i ceo element koji sadrži takve pod-elemente, što je u skladu sa preporukama definisanim u poglavlju 5.3, i nije potrebno vršiti nikakve dodatne izmene. Na navedenom primeru to znači da su prilikom računanja prosečnih vrednosti skokova u napadu i odbrani, nedostajući podaci sa ove jedne utakmice zanemareni (računat je prosek na osnovu preostalih utakmica), dok su prilikom računanja ostalih prosečnih vrednosti (šut za jedan i dva poena) podaci sa ove utakmice uzeti u proračun.

Kada se podaci iz listinga 9.6 ubace u testnu kolekciju iz prethodnog primera, na izlazu se dobija rezultat kao u listingu 9.5. To znači da je Misrosoft SQL Server implementacija uspela pravilno da obradi nedostajuće informacije.

Drugi tip Null vrednosti je vrednost tipa **neprimenljiva**. Pretpostavimo da je igrač igrao utakmicu, ali nije izvodio slobodna bacanja (vrednost N u tabeli 9.1). Elemenat koji opisuje procenat šuta sa linije slobodnih bacanja u ovom slučaju ne bi smeo biti 0 procenata, zbog velikog negativnog uticaja na celokupan zaključak o učinku igrača po tom kriterijumu, već ga treba tretirati kao Null vrednost tipa *neprimenljiva*. To znači da bi XML dokumenat iz

listinga 9.1 trebao izgledati kao u listingu 9.7. Atribut *nillable="true"* označava da element ima vrednost tipa Null - *neprimenljiva*.

```
<igrac ime="Gordon Drew Edward" pozicija="KC" kolo="16"
domacin="true" protivnik="Cibona">
    <minutaza>31</minutaza>
    <poena>
        <ukupno>11</ukupno>
        <sutza1 nillable="true"/>
        <sutza2>0.80</sutza2>
        <sutza3>0</sutza3>
    </poena>
    <skok>
        <napad>2</napad>
        <odbrana>5</odbrana>
    </skok>
    <asistencije>2</asistencije>
    <osvojenelopte>3</osvojenelopte>
    <izgubljenelopte>0</izgubljenelopte>
    <licnegreske>3</licnegreske>
    <blokade>1</blokade>
</igrac>
```

Listing 9.7. Primer statistike igrača na utakmici sa neprimenljivim podacima.

Kada se podaci iz listinga 9.7 ubace u testnu kolekciju, na izlazu se dobija rezultat kao u listingu 9.5. Misrosoft SQL Server implementacija je ignorisala nedostajuće vrednosti i pravilno obradila ostatak dokumenta.

9.2 Medicina - određivanje stanja pacijenata analizom XML – CCR dokumenata

Pretpostavimo da lekar pregleda pacijenata i želi da na osnovu njegovih rezultata i simptoma bolesti dobije preporuku kakvu terapiju da mu prepiše. Za odlučivanje će koristiti dostupne podatke od ostalih pacijenata sa sličnim simptomima. Podaci su skladišteni u obliku XML – CCR (Continuity of Care Record) dokumenata što je standard za beleženje podataka o zdravstvenom stanju pacijenta. Standard je zasnovan na XML jeziku, sa predefinisanim elementima opisanim u XSD šemi (Google, 2013). Dokument sadrži različite sekcije kao što su biografija pacijenta, informacije o osiguranju, dijagnoza i spisak problema, prepisani lekovi, alergije, plan zbrinjavanja, itd. Za potrebe ovog zadatka preuzeti su realni primeri XML – CCR dokumenta (Google, 2013) i na osnovu njih je generisana kolekcija pregleda, koja se dalje koristi u analizi.

Neka pitanje lekara glasi: "*Koju terapiju prepisati pacijentu, ako ima visoku temperaturu, osećaj iscrpljenosti, a povremeno mu se javljaju bolovi u mišićima i kašljuca*".

Ideja je da se prođe kroz dostupna XML – CCR dokumenta i da se pokušaju pronaći ona u kojima pacijenti imaju povišenu temperaturu i bar još jedan od preostala tri simptoma.

Karakterističan primer XML – CCR dokumenat dat je u listingu 9.8. Dokument zbog veličine nije dat u celosti već samo njegovi najbitniji delovi. Podaci o osobama su delimično izmenjeni, ali je struktura dokumenata ostala ista kao u preuzetim primerima.

```
<?xml version="1.0"?>
<ContinuityOfCareRecord xmlns='urn:astm-org:CCR'>
<CCRDocumentObjectID>//www.google.com/h9/feeds/scrapbook/jb3JrxYqJHg</CCRDo
cumentObjectID>
   <Language>
      <text>Srpski</text>
      <Code>
        <Value>srb</Value>
        <CodingSystem>ISO-639-1</CodingSystem>
      </Code>
    </Language>
    <Version>V1.0</Version>
    <DateTime>
      <ExactDateTime>2013-04-07</ExactDateTime>
    </DateTime>
    <Patient>
      <ActorID>Pacient</ActorID>
    </Patient>
    <Body>
      <FunctionalStatus>...</FunctionalStatus>
      <Problems>
        <Problem>
          <DateTime>
            <Type><Text>Datum pojave problema</Text></Type>
            <ExactDateTime>2013-04-04</ExactDateTime>
          </DateTime>
          <Description>
            <Text>Telesna temperatura</Text>
            <Code>
              <Value>38</Value>
              <Units>
                <Unit>C</Unit>
              </Units>
            </Code>
          </Description>
          <Status>
            <Text>Active</Text>
          </Status>
        </Problem>
        <Problem>
          <DateTime>
            <Type><Text>Datum pojave problema</Text></Type>
            <ExactDateTime>2013-04-04</ExactDateTime>
          </DateTime>
          <Description>
            <Text>Iscrpljenost</Text>
            <Code>
              <Value>40</Value>
              <Units>
                <Unit>%</Unit>
              </Units>
            </Code>
          </Description>
```

```
          <Status>
            <Text>Active</Text>
          </Status>
        </Problem>
      </Problems>
      <SocialHistory>...</SocialHistory>
      <Alerts>...</Alerts>
      <Medications>
        <Medication>
          <Status><Text>Activan</Text></Status>
          <DateTime>
            <Type>
              <Text>Datum recepta</Text>
            </Type>
            <ExactDateTime>2013-04-07</ExactDateTime>
          </DateTime>
          <Results>
            <Result>
              <Description><Text>Grip</Text></Description>
            </Result>
          </Results>
          <Product>
            <ProductName>
              <Text>Vitamin C</Text>
              <Code>
                <Value>30461</Value>
                <CodingSystem>RxNorm</CodingSystem>
              </Code>
            </ProductName>
            <Strength>
              <Value>500</Value>
              <Units>
                   <Unit>mg</Unit>
              </Units>
            </Strength>
            <Form><Text>Tableta</Text></Form>
          </Product>
          <Directions>
            <Direction>
              <Dose><Value>1</Value>
               <Units><Unit>tableta</Unit></Units></Dose>
              <Route><Text>Oralno</Text></Route>
              <Frequency><Value>3 na dan</Value></Frequency>
            </Direction>
          </Directions>
        </Medication>
      </Medications>
      <Immunizations>...</Immunizations>
      <VitalSigns>...</VitalSigns>
      <Results>...</Results>
      <Procedures>...</Procedures>
    </Body>
  <Actors>
    <Actor>
      <ActorObjectID>Patient</ActorObjectID>
      <Person>
        <DateOfBirth>
          <ExactDateTime>1937-09-21</ExactDateTime>
        </DateOfBirth>
        <Gender>
          <Text>Muško</Text>
```

```
          <Code>
            <Value>248152002</Value>
            <CodingSystem>SNOMED</CodingSystem>
          </Code>
        </Gender>
      </Person>
    </Actor>
  </Actors>
  </ContinuityOfCareRecord>
```

Listing 9.8. Primer XML – CCR dokumenta.

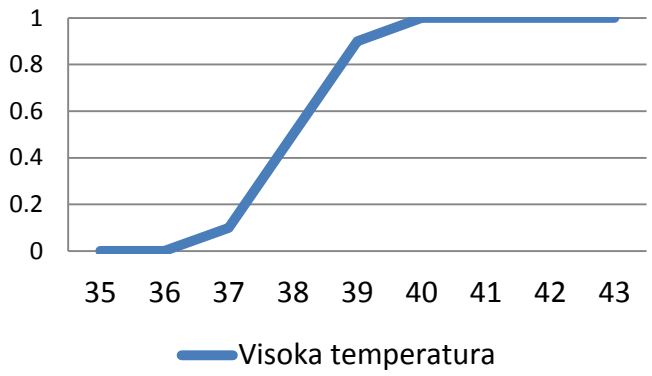
Upit koji se postavlja prikazan je u listingu 9.9

```
<rezultat>
{
      for $item in /ContinuityOfCareRecord/CCRDocumentObjectID/Body
            let $temperatura:= (
                  for $i1 in $item/Problems/Problem
                  let $Description:=($i1/Description/Text)
                  where $Description = "Telesna temperatura"
                  return $i1)[1]
            let $iscrpljenost:= (
                  for $i2 in $item/Problems/Problem
                  let $Description:=($i2/Description/Text)
                  where $Description = "Iscrpljenost"
                  return $i2)[1]
            let $bolovi_misica:= (
                  for $i3 in $item/Problems/Problem
                  let $Description:=($i3/Description/Text)
                  where $Description = "Bolovi u misicima"
                  return $i3)[1]
            let $kasalj:= (
                  for $i4 in $item/Problems/Problem
                  let $Description:=($i4/Description/Text)
                  where $Description = "Kasalj"
                  return $i4)[1]

            let $medications:= (for $m in $item/Medications/Medication
                  return $m)
            let $dijagnoza:= ($medications/Results/Result/Description)
            let $naziv:= ($medications/Product/ProductName/Text)
            let $oblik:=($medications/Product/Form/Text)
            let $doza:= ($medications/Product/Strength/Value)
            let $jedinicaMere:= ($medications/Product/Strength/Units/Unit)
            let $nacin_upotrebe:= ($medications/Directions/Frequency/Value)

      where $temperature = #visoka_temperatura AND(
            $iscrpljenost = #iscrpljenost[P0.8,T0.3] OR
            $bolovi_misica = #bolovi_u_misicima[P0.5,T0.3] OR
            $kasalj = #kasalj[P0.3,T0.3])
```

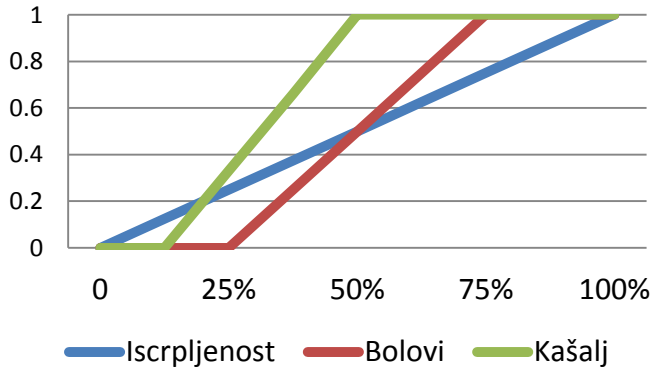
```
        order by $naziv
        return
        <dijagnoza>{$dijagnoza}</dijagnoza>
        <lek>
              <naziv>{$naziv}</naziv>
              <forma>{$forma}</forma>
              <doza>{$doza}-{$jedinicaMere}</doza>
              <nacin_upotrebe>{$nacin_upotrebe}</nacin_upotrebe>
        </lek>
}
</rezultat>
```

Listing 9.9. Postavljanje fazi XQuery upita nad XML - CCR dokumentom.

Neka postoji skup ograničenja $R_1^f = "visoka\_temperatura"$, $R_2^f = "iscrpljenost"$, $R_3^f = "bolovi\_misica"$ i $R_4^f = "kasalj"$. Navedena ograničenja se mogu definisati kao lingvističke fazi promenljive *#visoka_temperatura*, *#iscrpljenost*, *#bolovi_u_misicima* i *#kasalj*, a njihove funkcije pripadnosti odgovaraju funkcijama sa slika 9.3 i 9.4.



Slika 9.3. Funkcija pripadnosti – Visoka temperatura.



Slika 9.4. Funkcije pripadnosti – Iscrpljenost, bolovi i kašalj.

Primer formule za izračunavanje stepena zadovoljenja sa četiri skupa ograničenja povezana na sledeći način $R_1^f \wedge (R_2^f \vee R_3^f \vee R_4^f)$, izgledao bi kao u nastavku:

$$\alpha = T_L\left(S_P\left(\mu_{R_1^f}(\vartheta), 1-\rho\left(R_1^f\right)\right), S_L\left(S_L\left(S_P\left(\mu_{R_2^f}(\vartheta), 1-\rho\left(R_2^f\right)\right), S_P\left(\mu_{R_3^f}(\vartheta), 1-\rho\left(R_3^f\right)\right)\right), S_P\left(\mu_{R_4^f}(\vartheta), 1-\rho\left(R_4^f\right)\right)\right)\right)$$

Dalje se vrši izračunavanju kao u prethodnim primerima, a dobijeni rezultat je dat u listingu 9.10.

```
<rezultat>
      <dijagnoza>Grip</dijagnoza>
      <lek>
            <naziv>Vitamin C</naziv>
            <forma>Tableta</forma>
            <doza>500-mg</doza>
            <nacin_upotrebe>3 na dan</nacin_upotrebe>
      </lek>
</rezultat>
```

Listing 9.10. Rezultat – Dijagnoza i terapija.

Upotreba fazi logike u sistemima za dijagnostikovanje i donošenje zaključaka u medicini nije novina. Cadiag (Computer Assisted DIAGnosis) je ekspertski sistem zasnovan na pravilima, koji ima za cilj pružanje podrške u dijagnostičkom donošenju odluka u oblasti interne medicine. Razvio ga je profesor K.P. Adlassnig na Medicinskom univerzitetu u Beču, videti radove (Adlassnig, et al., 1985), (Adlassnig, et al., 1986), (Adlassnig, 1986) i (Leitich, et al., 2002) .

Sistem se sastoji od sistema za odlučivanje (inference engine) i baze znanja. Sistem za odlučivanje u prvobitnom sistemu Cadiag 1 zasnovan je na klasičnoj tro-vrednosnoj logici (tačno, netačno i neodređeno). Upotrebljivost ekspertskog sistema zasnovanog na takvoj logici je ograničena, često i nemoguća u praksi. Unapređeni sistem Cadiag 2 zamenjuje klasične vrednosti fazi vrednostima. Baza znanja se sastoji od skupa IF – THEN pravila koja opisuju odnose između različitih medicinskih entiteta: simptoma i rezultata ispitivanja s jedne strane i bolesti i terapija sa druge. Primeri pravila u Cadiag 2 sistemu dati su u listingu 9.11.

```
IF suspicion of liver metastases by liver palpation
THEN pancreatic cancer
with the confirmability degree 0.55

IF positive rheumatoid factor
THEN NOT seronegative rheumatoid arthritis
```

```
IF NOT (rheumatoid arthritis AND splenomegaly AND leukopenia ≤ 4000 /µl)
THEN NOT Felty's syndrome
```

Listing 9.11. Primeri Cadiag 2 pravila, preuzeto iz rada (Ciabattoni & Vetterlein, 2010).

Stepen poklapanja predstavlja stepen istinitosti predviđene dijagnoze (u prvom primeru to je 0.55 ili 55%).

Cadiag sistemi nisu predviđeni za rad i obradu XML datoteka (u obliku dijagnoza, recepata…) i upotrebu XQuery upita nad njima. Za razliku od rešenja predstavljenog u disertaciji, a koje koristi postojeće dijagnoze u obliku XML – CCR dokumenata, Cadiag sistemi sadrže bazu znanja koju je prethodno bilo potrebno razviti. U definisanju baze znanja učestvovali su medicinski stručnjaci, koji su preneli svoje znanje na skup pravila, nalik onima iz listinga 9.11. Rešenje u disertaciji nije bazirano na skupovima pravila, već se rezultat dobija analizom baze XML – CCR dokumenata i donošenjem zaključka na osnovu sličnosti simptoma sa simptomima kod drugih pacijenata. Što je skup dokumenata veći, a dijagnoze u njima tačnije, to će rezultat biti precizniji.

9.2.1 Postavljanje tekstualnih upita

Prethodni primeri su pokazali kako korisnik na osnovu zadatog pitanja kreira funkcije pripadnosti za odgovarajuće fazi skupove, fazi XQuery upit, određuje prioritete i nakon toga izvršava upit nad potrebnim dokumentima, kako bi dobio željeni rezultat. Šta ukoliko lekar ne zna fazi XQuery sintaksu ili jednostavno ne želi da prevodi svoje pitanje u fazi XQuery upitni jezik? Potrebno je omogućiti postavljanje upita klasičnim pitanjem.

Razvijena sintaksa ne podržava takve mogućnosti, ali daje dobru osnovu za njen razvoj. Ideja je da se za specifične potrebe kao što su medicina, ekonomija, pravo…, razviju specijalizovani softverski alati koji bi tekstualne upite pretvarali u odgovarajuće fazi XQuery upite. Za razvoj takvih tipova softverskih alata, neophodno je stručno znanje iz pomenutih oblasti i poznavanje teorije fazi skupova. Mogao bi se razviti grafički alat koji bi bio u stanju da funkcije pripadnosti iz upita prebaci u softverski paket na dalju obradu.

Glava X

ZAKLJUČAK

Rad daje prikaz XML sintakse i XML tehnologija, sa posebnim osvrtom na XQuery upitnu sintaksu. Date su osnovne teorijske definicije iz oblasti fazi skupova i fazi logike. Opisani su CSP-a, FCSP-a, PFCSP-a i GPFCSP-a, što omogućava upotrebu prioriteta. Opis sintaksi, tehnologija i potrebnih teorijskih definicija bio je neophodan zbog lakšeg razumevanja suštine rada.

Dat je pregled dosadašnjih istraživanja i analiza naučnih radova iz oblasti upotrebe fazi logike u definisanju XML neodređenosti, čime je dobijena precizna slika o trenutnim dostignućima u ovoj oblasti.

Razvijena je fazi XML sintaksa, koja predstavlja proširenje standardnog XML-a i omogućava definisanje neodređenosti u vrednostima i strukturi XML dokumenta. Sintaksa je definisana upotrebom XSD i DTD dokumenata. Data je definicija i postupak obrade dva tipa Null vrednosti: nepoznata vrednost i neprimenljiva vrednost.

Rad navodi razloge za razvoj fazi upitne sintakse, a potom i definiše fazi XQuery upitnu sintaksu, koja predstavlja proširenje standardne XQuery sintakse. Dobijeni fazi XQuery se proširuje prioritetima i pragovima zadovoljenja upotrebom GPFCSP-a. Navedena je precizna definicija fazi XQuery upitne sintakse pomoću EBNF gramatike, kao i opis izvršenih proširenja. Obrada upita u slučaju pojave Null vrednosti data je u formi preporuka kod procesa implementacije.

Na osnovu razvijenih sintaksi upotrebom UML dijagrama modelovan je softverski paket pod imenom *Fazi XML Editor*. Opisan je način implementacije i tehnologije korišćene u razvoju same aplikacije.

Aplikacija omogućava skladištenja i rad sa standardnim i fazi XML, DTD i XSD dokumentima, kao i kreiranje i izvršavanje standardnih i prioritizovanih fazi XQuery upita.

Dat je pregled funkcionalnosti razvijene aplikacije kroz opis procesa instalacije, inicijalnog pokretanja, skladištenja dokumenata, rada sa standardnim i fazi XML, DTD i XSD dokumentima i upotrebom *Fazi XQuery editora* u izvršenju prioritizovanih fazi XQuery upita. Postoji prateći disk koji sadrži izvršnu verziju aplikacije, video datoteke i dodatne datoteke za testiranje.

Urađena je detaljna analiza kreiranog softverskog paketa i to analizom korisničkog interfejsa sa aspekta funkcionalnosti, testiranjem performansi sistema i proverom stabilnosti aplikacije.

Na kraju su navedeni praktični primeri koji pokazuju mogućnosti primene definisanih sintaksi i razvijenog softverskog paketa u rešavanju nekoliko realnih problema.

U ključnim referencama iz ove oblasti, što je detaljno prikazano u trećem poglavlju, više autora je uspelo da uradi neke od stavki, ili više njih, koje su realizovane u ovoj disertaciji. Dobre ideje iz drugih radova su analizirane i unapređene (kao na primer definisanje neodređenosti u vrednostima XML-a, definisanje neodređenosti u strukturi XML-

a, podrška za fazi DTD i fazi XSD dokumenta, podrška za predefinisane funkcije pripadnosti, definisanje fazi XQuery upitne sintakse i implementacija), a dodate su i neke nove (definisanje proizvoljnih funkcija pripadnosti, obrada više tipova Null vrednosti, uvođenje prioriteta i pragova zadovoljenja u fazi XQuery upite, detaljno modeliranje, ozbiljna implementacija, testiranje razvijenog softverskog paketa i pronalaženje i analiza primera praktične primene), da bi na kraju sve bilo objedinjeno u jedno celovito rešenje. Zbog svega navedenog, a prema saznanjima autora, disertacija se može smatrati **izrazito sveobuhvatnim rešenjem** sa stanovišta problematike koju pokriva iz oblasti definisanja neodređenosti u XML dokumentima.

Funkcije pripadnosti koje se koriste za definisanje fazi skupova i opis neodređenosti, ne ograničavaju se na skup dobro poznatih predefinisanih funkcija, što je slučaj u drugim radovima, već se korisnicima pruža mogućnost **definisanja proizvoljnih funkcija pripadnosti**. Tu mogućnost drugi autori po pravilu izbegavaju zbog komplikovane sintakse i procesa implementacije, kao i odsustva zadovoljavajućih performansi sistema. Razlog zbog čega je ova implementacija u stanju da radi sa proizvoljnim funkcijama pripadnosti u realnom vremenu je upotreba eksterne aplikacije MATLAB za izračunavanje kompleksnih matematičkih izraza. Korišćenje eksternog matematičkog modula je pristup kojem se često pribegava u programiranju zarad osiguranja vrhunskih performansi sistema i brže implementacije.

Uvođenju **prioriteta** i **pragova zadovoljenja** u fazi XQuery upite, predstavlja još jednu specifičnost ove disertacije. Prioriteti i pragovi zadovoljenja, iako bitni, do sada nisu korišćeni u XML upitnim sintaksama. Bazirani su na upotrebi GPFCSP-a, a tako definisani upiti mogu sadržati neodređenosti različitih prioriteta i pragova zadovoljenja. To bi značilo da svaki uslov u upitu u zavisnosti od svog prioriteta i praga zadovoljenja ima odgovarajući uticaj na krajnji rezultat. Tako definisani uslovi mogu da daju preciznija rešenja u odnosu na fazi XQuery upite, koji ne podržavaju rad sa prioritetima i pragovima zadovoljenja.

Prema dostupnim saznanjima autora, obrada više tipova **Null vrednosti** u fazi XML i fazi XQuery upitima prvi put je definisana u ovom radu. Definisana su dva osnovna tipa Null vrednosti, nedostupna vrednost i neprimenljiva vrednost. Obrada dokumenata koji sadrže Null vrednosti urađena je na intuitivan način.

Velika prednost ove disertacije u odnosu na ostale radove je uspešno sprovedena **implementacija** definisanih sintaksi u celovito aplikativno rešenje. Drugi autori se uglavnom fokusiraju na definisanje sintaksi, dok se proces njihove implementacije najčešće zanemaruje. Ukoliko bi i postojao neki vid implementacije, mogućnosti takvog rešenja su toliko ograničene da je bilo kakva upotreba u praksi nad realnim skupom dokumenata praktično neizvodljiva. Takođe, detaljna **analiza** i **testiranje** kreiranog softverskog paketa, iako bitne stavke u potvrdi kvaliteta definisanih sintaksi i izvršene implementacije, ne sprovode se u drugim radovima.

Još jedna dobra osobina disertacije, a zanemarivana od strane drugih autora, je pronalaženje **primera upotrebe** razvijenog rešenja u realnim sistemima. Prikazom takvih primera čitalac dobija bolju sliku o tome kako se upotrebom definisanih sintaksi i implementiranog aplikativnog rešenja mogu rešiti neki praktični problemi.

Razvijena sintaksna rešenja i implementirani softverski paket treba da posluže kao osnova za dalja istraživanja u oblasti definisanja neodređenosti u XML dokumentima. U budućim istraživanjima posvetiće se pažnja usavršavanju sintaksnih rešenja i poboljšanju implementacije sa ciljem proširenja postojećih funkcionalnosti aplikacije i podizanju

performansi sistema. Usavršavanje sintakse i implementacija neodređenosti u strukturi XML dokumenata (osnovna sintaksa je opisana u ovom radu, ali nije urađena implementacija), biće jedan od pravaca budućih istraživanja. Razvoj i implementacija fazi XPath jezika, koji bi se bavio izračunavanjem verovatnoće pojavljivanja nekog elementa u XML dokumentu, takođe može biti tema nekog narednog rada iz oblasti neodređenosti u strukturi XML dokumenata. Sledeći pravac obuhvataće fazifikaciju tekstualnog sadržaja, što podrazumeva razvoj specifičnih sintaksi i izradu odgovarajućih kompajlera za generisanje fazi XQuery upita. Navedene sintakse i kompajleri trebali bi da budu specijalizovani za različita područja primene (na primer medicina, ekonomija ili analiza sportskih rezultata).

LITERATURA

1. Adlassnig, K., 1986. Fuzzy set theory in medical diagnosis. *IEEE Transactions on Systems, Man and Cybernetics,* 16(2), p. 260–265.
2. Adlassnig, K., Kolarz, G., Effenberger, W. & Grabner, H., 1985. Cadiag: Approaches to computer assisted medical diagnosis. *Computers in Biology and Medicine,* Volume 15, p. 315–335.
3. Adlassnig, K., Kolarz, G., Scheithauer, W. & Grabner, H., 1986. Approach to a hospital-based application of a medical expert system. *Informatics for Health and Social Care,* 11(3), p. 205–223.
4. Barranco, C., Campaña, J. & Medina, J., 2005. *Towards a XML fuzzy structured query language.* Barcelona, Spain, C. Barranco, J. Campaña and J. Medina, Towards a XML fuzzy structured query language, in: Proc. EUSFLAT Conf., p. 1188–1193.
5. Bistarelli, S. et al., 1999. Semiring-Based CSPs and Valued CSPs: Frameworks, Properties, and Comparison. *Constraints,* 4(3), p. 199–240.
6. Boag, S. et al., 2011. *XQuery 1.0: An XML query language, second ed.*. [Online] Available at: http://www.w3.org/TR/xquery/ [Accessed 13 8 2013].
7. Bray, T. et al., 2009. *Namespaces in XML 1.0 (Third Edition).* [Online] Available at: http://www.w3.org/TR/REC-xml-names/ [Accessed 13 8 2013].
8. Campi, A. et al., 2009. A fuzzy extension of the XPath query language. *Journal Intell. Inf. Syst.,* 33(3), p. 285–305.
9. Chan, A. et al., 2008. *Fuzzy querying of nested XML.* Las Vegas, USA, Proc. Inform. Reuse Integrat. IEEE Inter. Conf., p. 238–243.
10. Ciabattoni, A. & Vetterlein, T., 2010. On the (fuzzy) logical content of CADIAG-2. *Fuzzy Sets and Systems,* 161(14), p. 1941–1958.
11. Codd, E., 1986. Missing information (applicable and inapplicable) in relational databases. *ACM SIGMOD Record,* 15(4), p. 53–78.
12. Combi, C., Oliboni, B. & Rossato, R., 2005. *Evaluating fuzzy association rules on XML documents.* Springer, Heidelberg, Computational Intelligence, Theory and Applications, Advances in Soft Computing 33, p. 435–448.
13. Damiani, E., Marrara, S. & Pasi, G., 2007. *Fuzzy XPath: Using fuzzy logic an IR features to approximately query XML documents.* Cancun, Mexico, Proc. 12th Int. Fuzzy Syst. Assoc., p. 199–208.
14. Damiani, E., Oliboni, B. & Tanca, L., 2001. *Fuzzy techiques for XML data smushing.* Dortmund, Germany, Proc. Inter. Conf. 7th Fuzzy Days Computational Intell., Theor. App., p. 637–652.
15. Damiani, E., Tanca, L. & Fontana, F., 2000. Fuzzy XML queries via context-based choice of aggregations. *Kybernetika,* 36(6), p. 635–655.
16. Dubois, D., Fargier, H. & Prade, H., 1996. Possibility theory in constraint satisfaction problems: Handling priority, preference and uncertainty. *Applied Intelligence,* Volume 6, pp. 287-309.

17. Dubois, D. & Fortemps, P., 1999. Computing Improved Optimal Solutions to max-min Flexible Constraint Satisfaction Problems. *D. Dubois, P. Fortemps, Computing Improved Optimal Solutions to max-min FlexibleEuropean Journal of Operational Research,* Volume 118, p. 95–126.

18. Gaurav, A. & Alhajj, R., 2006. *Incorporating fuzziness in XML and mapping fuzzy relational data into fuzzy XML.* New York, USA, Proc. ACM Symp. Appl. Comput., p. 456–460.

19. Google, 2013. *Googlecode, CCR.* [Online] Available at: http://java-algorithm-education.googlecode.com/svn/CCR/xsd/CCR.xsd [Accessed 12 5 2013].

20. Google, 2013. *Googlehealthsamples, Source.* [Online] Available at: http://code.google.com/p/googlehealthsamples/source/browse/trunk/CCR_samples/ [Accessed 12 5 2013].

21. Hailong, W., Ma, Z., Li, Y. & Jingwei, C., 2008. *A unified formalism for fuzzy data types representation.* Shandong, China, FSKD '08. Fifth Inter. Conf., p. 167–171.

22. Harold, E. & Means, W., 2004. *XML in a Nutshell, Third Edition.* s.l.:O'Reilly Media.

23. Herrera-Viedma, E. et al., 2007. A fuzzy linguistic model to evaluate the quality of Web sites that store XML documents. *Int. J. Approx. Reason,* 46(1), p. 226–253.

24. Jin, Y. & Shidlagatta, S., 2009. *A framework of fuzzy triggers for XML database systems.* Las Vegas, USA, Proc. Inform. Reuse Integrat. IEEE International Conference, p. 296–299.

25. Kay, M., 2007. *XSL Transformations (XSLT) Version 2.0.* [Online] Available at: http://www.w3.org/TR/xslt20/ [Accessed 13 8 2013].

26. Koyuncu, M., 2011. Intelligent fuzzy queries for multimedia databases. *International Journal of Intelligent Systems*, 26(10), p. 930–951.

27. Kril, G. & Yuan, B., 1995. *Fuzzy Sets and Fuzzy Logic: Theory and Applications.* New Jersey: Prentice Hall.

28. Lee, J. & Fanjiang, Y., 2003. Modeling imprecise requirements with XML. *Information and Software Technology,* Volume 45, p. 445–460.

29. Leitich, H., Adlassnig, K. & and Kolarz, G., 2002. Evaluation of two different models of semiautomatic knowledge acquisition for the medical consultant system cadiag2/rheuma. *Artificial Intelligence in Medicine,* Volume 25, p. 215–225.

30. Lin, C. & Lee, C., 1996. *Neural Fuzzy Systems: A Neuro-Fuzzy Synergism to Intelligent Systems.* New Jersey: Prentice Hall.

31. Lo, A., Kianmehr, K., Kaya, M. & Alhajj, R., 2007. *Wrapping VRXQuery with self-Adaptive fuzzy capabilities.* Fremont, CA, Web Intelligence, IEEE/WIC/ACM Inter. Conf., p. 750–756.

32. Luo, X., Lee, J., Leung, H. & Jennings, N., 2003. Prioritised Fuzzy Constraint Satisfaction Problems: Axioms, Instantiation and Validation. *Fuzzy Sets and Systems,* 136(2), p. 151–188.

33. Ma, Z., 2005. *Fuzzy database modeling with XML.* New York: Springer.

34. Ma, Z., Liu, J. & Yan, L., 2010. Fuzzy data modeling and algebraic operations in XML. *Int. J. Intell. Syst.,* 25(9), p. 925–947.

35. Ma, Z. & Shen, D., 2006. Modeling fuzzy information in the IF2O and object-oriented data models. *Journal of Intelligent and Fuzzy Systems,* Volume 17, pp. 597-612.

36. Ma, Z., Yan, L. & Zhang, F., 2012. Modeling fuzzy information in UML class diagrams and object-oriented database models. *Fuzzy Sets and Systems,* 186(1), p. 26–46.

37. Meseguer, P. & Larrosa, J., 1997. *Solving fuzzy constraint satisfaction problems.* Barcelona, Proceedings of the 6th IEEE International Conference on Fuzzy Systems, pp. 1233-1238.

38. Nathan, A., 2010. *WPF 4 Unleashed.* USA: Sams.

39. Oliboni, B. & Pozzani, G., 2008. *Representing Fuzzy information by using XML schema.* Turin, Italy, Database and Expert Systems Application, DEXA Proc. 19th Inter. Conf. Database Expert Syst. App., p. 683–687.

40. Panić, G., Racković, M. & Škrbić, S., 2012. *Fuzzy XML with Implementation.* Novi Sad, Serbia, BCI'12 September 16–20, p. 58–62.

41. Panić, G., Racković, M. & Škrbić, S., 2013. Fuzzy XML and prioritized fuzzy XQuery with implementation. *Journal of Intelligent and Fuzzy Systems. DOI: 10.3233/IFS-120739.*

42. Pap, E. & Takači, A., 2005. *An Application of Schur-concave t-norms in PFCSP.* Barcelona, Spain, Proceedings of the 4th Conference of the EUSFLAT, p. 380–384.

43. Perović, A., Takači, A. & Škrbić, S., 2009. Towards the formalization of fuzzy relational database queries. *Acta Polytech. Hun.,* 6(1), p. 185–193.

44. Perović, A., Takači, A. & Škrbić, S., 2012. Formalising PFSQL queries using ŁΠ½ fuzzy logic. *Math. Struct. Comput. Sci.,* 22(3), p. 533–547.

45. Prade, H. & Testemale, C., 1984. Generalizing database relational algebra for the treatment of incomplete or uncertain information and vague queries. *Information Sciences,* 34(2), p. 115–143.

46. Rescher, N., 1969. *Many-valued Logic.* New York, USA: McGraw-Hill Book Co..

47. Rodrigues, R., Cruz, A. & Cavalcante, R., 2009. A proposal for a fuzzy database architecture incorporating XML. *Fuzzy Sets and Systems,* 160(2), p. 269–279.

48. Rosenfeld, A., Hummel, R. A. & Zucker, S., 1976 . Scene Labeling by Relaxation Operations. *IEEE Systems, Man, and Cybernetics Society,* 6(6), pp. 420 - 433 .

49. Ruttkay, Z., 1994. *Fuzzy constraint satisfaction.* Orlando, FL , In Proc. 3rd IEEE International Conference on Fuzzy Systems, pp. 1263-1268.

50. Schiex, T., 1992. *Possibilistic constraint satisfaction problems or how to handle soft.* Stanford, CA, In Proc. of the 8th Conf. on Uncertainty in Artificial Intelligence, pp. 268-275.

51. Schildt, H., 2010. *C# 4.0 The Complete Reference.* 1 ed. USA: McGraw-Hill Osborne Media.

52. Sells, C. & Griffiths, I., 2007. *Programming WPF.* 2 ed. USA: O'REILLY Media.

53. Seto, J. et al., 2009. Fuzzy query model for XML documents. *Lect. Notes Comput. Sc.,* Volume 5788 , p. 333–340.

54. Škrbić, S., 2008. *Upotreba fazi logike u relacionim bazama podataka (Doktorska disertacija).* Novi Sad: Prirodno-matematički fakultet.

55. Škrbić, S., Racković, M. & Takači, A., 2011. The PFSQL query execution process. *Novi Sad J. Math.,* 41(2), p. 161–179.

56. Škrbić, S., Racković, M. & Takači, A., 2011. Towards the methodology for development of fuzzy relational database applications. *Comput. Sci. Inf. Syst.,* 8(1), pp. 27-40.

57. Škrbić, S., Racković, M. & Takači, A., 2013. Prioritized fuzzy logic based information processing in relational databases. *Knowledge-Based Systems,* Volume 38, p. 62–73.

58. Takači, A., 2005. Schur-concave triangular norms: Characterization and application in PFCSP. *Fuzzy Sets and Systems,* 155(1), p. 50–64.

59. Takači, A., 2006. *Towards Priority Based Logics.* s.l., Proceedings of the 11th International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems (2006), pp. 651-657.

60. Takači, A., 2006. *Trougaone norme prioriteta i njihova primena na modeliranje ispunjenja fazi ograničenja (Doktorska disertacija).* Novi Sad: Prirodno-matematički fakultet.

61. Takači, A. & Škrbić, S., 2008. *Data Model of FRDB with different data types and PFSQL.* Hershey, PA, in: J. Galindo (Ed.), Handbook of research on fuzzy information processing in databases, Information Science Reference, p. 407–434.

62. Takači, A. & Škrbić, S., 2008. Priority, weight and threshold in fuzzy SQL systems. *Acta Polytech. Hun.,* 5(1), p. 59–68.

63. Takači, A., Škrbić, S. & Perović, A., 2009. *Generalised prioritised constraint satisfaction problem.* Subotica, Serbia, Proc. 7th Serbian-Hungarian Joint Symposium Intell. Syst. Inf., p. 145–148.

64. Tanaka, K. & Nimura, T., 1996. *An Introduction to Fuzzy Logic for Practical Applications.* Crashing Rocks Books ed. Punta Gorda, FL, U.S.A.: Springer.

65. Thomson, E. & Radhamani, G., 2009. Fuzzy logic based XQuery operations for native XML database Systems. *International Journal of Database Theory and Application,* 2(3), p. 13–20.

66. Tré, G., 2002. Extended possibilistic truth values. *International Journal of Intelligent Systems 17 (2002), 427–446.,* 17(4), p. 427–446.

67. Tré, G. & Caluwe, R., 2003. Modelling uncertainty in multimedia database systems: An extended possibilistic approach. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems,* 11(1), p. 5–22.

68. Tré, G., Caluwe, R. & Prade, H., 2008. Null values in fuzzy databases. *Journal of Intelligent Information Systems,* 30(2), p. 93–114.

69. Tseng, C., Khamisy, W. & Vu, T., 2005. Universal fuzzy system representation with XML. *Journal Computer Standards & Interfaces,* 28(2), p. 218–230..

70. Yan, L., Ma, Z. & Liu, J., 2009. *Fuzzy data modeling based on XML schema.* New York, USA, Proc. ACM Symp. Appl. Comput., p. 1563–1567.

71. Zadeh, L., 1965. Fuzzy Sets. *Information and Control,* Volume 8, p. 338–353..

72. Zhang, F., Z. Ma, L. Y. & Cheng, J., 2011. Storing Fuzzy Ontology in Fuzzy Relational Database. *Database and Expert Systems Applications - Lecture Notes in Computer Science,* Volume 6861 , p. 447–455.

73. Zhang, F., Z. Ma, L. Y. & Wang, Y., 2012. A description logic approach for representing and reasoning on fuzzy object-oriented database models. *Fuzzy Sets and Systems,* 186(1), p. 1–25.

SKRAĆENICE

ANSI - American National Standards Institute
BNF - BackusNaur Form
CADIAG - Computer Assisted DIAGnosis
CCR - Continuity of Care Record
CSP - Constraint Satisfaction Problem
DTD - Document Type Definition
DTI - Dijagrama Toka Interfejsa
EBNF - Extended BackusNaur Form
EPTV - Extended Possibilistic Truth Values
FCSP - Fuzzy Constraint Satisfaction Problem
FOOM - Fazi Objektno Orjentisano Modeliranje
GML - Generalized Markup Language
GPFCSP - Generalized Priority Fuzzy Constraint Satisfaction Problem
GDI - Graphics Device Interface
HCI - Human Computer Interaction
HTML - HyperText Markup Language
KLM - Keystroke Level Model
MSSQL - Microsoft SQL Server
MATLAB - MATrix LABoratory
MVVM - Model View ViewModel
XAML - eXtensible Application Markup Language
PFCSP - Prioritised Fuzzy Constraint Satisfaction Problem
PFSQL - Priority Fuzzy Structured Query Language
SGML - Standard Generalize Markup Language
TSQL - Transact-SQL
UML - Unified Modeling Language
W3C - World Wide Web Consortium
WPF - Windows Presentation Foundation
XFSQL - XML Fuzzy Structured Query Language
XML - eXstensible Markup Language
XPath - XML Path Language
XSD - XML Schema Language

BIOGRAFIJA

Goran Panić je rođen 5. januara 1981. godine u Kikindi. Osnovnu školu završio je u Iđošu. Srednju tehničku školu, smer automatika, završava u Kikindi. Fakultet tehničkih nauka u Novom Sadu, odsek Elektrotehnika, smer Računarstvo i automatika, upisao je školske 2000/01. godine. Na istom fakultetu 2008. godine na usmerenju Računarske Nauke, odbranio je Master rad iz oblasti Informacionih sistema, na temu "Razvoj aplikacije za podršku finansijskom knjigovodstvu".

Doktorske studije na Prirodno-matematičkom fakultetu u Novom Sadu, odsek za matematiku, smer Informatika, upisuje školske 2009/10. godine.

U periodu od 2001. do 2002. radi u ustanovi Medijski centar, na poziciji predavača. Od 2003. počinje da radi kao honorarni programer na raznim komercijalnim projektima. Godine 2008. zaposlio se u kompaniji ViaOne na mestu C# programera gde radi na razvoju poslovnih aplikacija. Juna 2010. prelazi u kompaniju RT-RK i radi kao softver inženjer na razvoju GUI-ja za digitalnu televiziju. Oktobra 2010. prelazi u firmu Vavei gde radi kako WPF programer na razvoju softvera za KNX sisteme. U firmu STS prelazi avgusta 2011. gde radi na poziciji programera i razvija softver za pametne kuće. Od aprila 2012. počinje paralelno da radi na projektima za kompaniju RapidTrade kao programer, gde radi na razvoju poslovnih informacionih sistema. Januara 2013. prelazi u firmu Service Plus gde radi na razvoju drajvera za fiskalne uređaje do maja 2013. Trenutno radi u firmi The Logos Vision na razvoju aplikacija za analizu berzanskih transakcija.

Živi u Novom Sadu, nije oženjen, od stranih jezika govori engleski.

Novi Sad, 21. August 2013. Goran Panić

Goran Panic

UNKIVERZITET U NOVOM SADU

PRIRODNO - MATEMATIČKI FAKULTET

KLJUČNA DOKUMENTACIJSKA INFORMACIJA

Redni broj: RBR	
Identifikacioni broj: IBR	
Tip dokumentacije: TD	Monografska dokumentacija
Tip zapisa: TZ	Tekstualni štampani materijal
Vrsta rada (dipl., mag., dokt.): VR	Doktorska disertacija
Ime i prezime autora: AU	Goran Panić
Mentor (titula, ime, prezime, zvanje): MN	dr Miloš Racković, redovni profesor
Naslov rada: NR	Razvoj namenskog sistema fazi logike za primenu u sistemima za upravljanje XML dokumentima
Jezik publikacije: JP	srpski (latinica)
Jezik izvoda: JI	srp. / eng.
Zemlja publikovanja: ZP	Srbija
Uže geografsko područje: UGP	Vojvodina
Godina: GO	2013

Izdavač: IZ	Autorski reprint
Mesto i adresa: MA	Prirodno-matematički fakultet, Trg Dositeja Obradovića 4, Novi Sad
Fizički opis rada: FO	(broj poglavlja / stranica / slika / grafikona / referenci / priloga) 10/156/49/0/73/0
Naučna oblast: NO	Informatika
Naučna disciplina: ND	Računarske nauke
Predmetna odrednica, ključne reči: PO	XML, XQuery, fazi logika
UDK	
Čuva se: ČU	Biblioteka Departmana za matematiku i informatiku PMF-a u Novom Sadu
Važna napomena: VN	Nema
Izvod: IZ	U stvarnom životu većina informacija dolazi kao neprecizne ili nepotpune vrednosti. XML tehnologija je napravila veliki napredak u oblasti skladištenja i prenosa podataka. Doktorska disertacija definiše fazi XML sintaksu koja kombinuje neodređenosti u vrednostima XML-a i neodređenosti u strukturi XML dokumenata. Takođe, rad proširuje standardnu XQuery upitnu sintaksu fazi elementima i uvodi prioritete i pragove zadovoljenja pomoću GPFCSP-a. Za razliku od drugih radova koji se fokusiraju na postavljanje teorija i definisanje sintaksi, rad posebnu pažnju posvećuje praktičnoj upotrebi definisanih sintaksi. Za potrebe istraživanja razvijen je softverski paket koji omogućava rad kako sa standardnim, tako i sa fazi XML, XSD i DTD dokumentima, kao i postavljanje i izvršavanje prioritizovanih fazi XQuery upita. Alat je testiran nad primerima iz prakse.
Datum prihvatanja teme od strane NN veća: DP	05.04.2013.

Datum odbrane: DO	
Članovi komisije: (ime i prezime / titula / zvanje / naziv organizacije / status) KO	predsednik: dr Zoran Budimac, red. prof, PMF, Novi Sad član: dr Miloš Racković, red. prof., PMF, Novi Sad, mentor član: dr Srđan Škrbić, docent, PMF, Novi Sad član: dr Ivan Luković, red. prof., FTN, Novi Sad član: dr Andreja Tepavčić, red. prof., PMF, Novi Sad

UNIVERSITY OF NOVI SAD

FACULTY OF SCIENCE AND MATHEMATICS

KEY WORD DOCUMENTATION

Accession number: ANO	
Identification number: INO	
Document type: DT	Monograph documentation
Type of record: TR	Textual printed material
Contents code: CC	Doctoral dissertation
Author: AU	Goran Panić
Mentor: MN	Miloš Racković, Ph. D., full professor
Title: TI	The development a dedicated system for the application fuzzy logic in systems for manage of XML documents
Language of text: LT	Serbian (Latin)
Language of abstract: LA	eng. / srp.
Country of publication: CP	Serbia
Locality of publication: LP	Vojvodina
Publication year: PY	2013

Publisher: PU	Author's reprint
Publication place: PP	Faculty of Science and Mathematics, Trg Dositeja Obradovića 4, Novi Sad
Physical description: PD	10/156/49/0/73/0
Scientific field SF	Informatics
Scientific discipline SD	Computer Science
Subject, Key words SKW	XML, XQuery, fuzzy logic
UC	
Holding data: HD	Library of Department of Mathematics and Informatics, Trg Dositeja Obradovića 4
Note: N	None
Abstract: AB	In real life, as opposed to virtual, most information comes in the form of imprecise or incomplete values. XML technology has made great progress in the field of storage and data transfer. This doctoral dissertation developed XML extension which combines indefiniteness in the values of XML and indefiniteness in the structure of XML into a single fuzzy XML extension. Furthermore, dissertation expands XQuery syntax with fuzzy values and includes priorities and thresholds in fuzzy XQuery extension using GPFCSP for the first time. Unlike other papers that focus on setting up theories and defining syntax, in this dissertation, special attention is turned to their practical use. A tool for working with XML, XSD and DTD documents and prioritized fuzzy XQuery extension queries has been developed. The tool has been tested on practical examples.
Accepted on Scientific Board on: AS	05.04.2013.

Defended: DE	
Thesis Defend Board: DB	president: Zoran Budimac, Ph.D., full prof., Faculty of Science and Mathematics, Novi Sad. member: Miloš Racković, Ph.D., full prof., Faculty of Science and Mathematics, Novi Sad, menthor member: Srđan Škrbić, Ph.D., assist. prof., Faculty of Science and Mathematics, Novi Sad member: Ivan Luković, Ph.D., full prof., Faculty of Engineering, Novi Sad. member: Andreja Tepavčić, Ph.D., full prof., Faculty of Science and Mathematics, Novi Sad