

Sveučilište u Zagrebu  
**Fakultet elektrotehnike i računarstva**

Hrvoje Šimić

**MODELIRANJE USTROJA  
WWW SJEDIŠTA**

Magistarski rad

Zagreb, 2002.

Rad je izrađen na Zavodu za telekomunikacije Fakulteta elektrotehnike i računarstva  
Sveučilišta u Zagrebu

Mentor rada: doc. dr. sc. Maja Matijašević

Radnja ima: 102 stranice

Rad br.

Povjerenstvo za ocjenu u sastavu:

1. Prof. dr. sc. Ignac Lovrek
2. Doc. dr. sc. Maja Matijašević
3. Prof. dr. sc. Vlatko Čerić

Povjerenstvo za obranu u sastavu:

1. Prof. dr. sc. Ignac Lovrek
2. Doc. dr. sc. Maja Matijašević
3. Prof. dr. sc. Vlatko Čerić

Datum obrane: 10. svibnja 2002.

# Sadržaj

Sadržaj .....	4
1 Uvod .....	8
2 Resursi i identifikatori.....	10
2.1 Web resurs.....	10
2.1.1 Pojam resursa.....	10
2.1.2 Resursi koji predstavljaju pojmove.....	11
2.1.3 Odnosi među resursima.....	11
2.1.4 Dinamičnost resursa informacija.....	12
2.1.5 Resursi na Internetu.....	12
2.2 <i>Uniform Resource Identifier</i> .....	13
2.2.1 Standardni URI .....	14
2.2.2 HTTP-ov URI .....	14
2.3 Identitet Web resursa.....	15
2.3.1 Ugled Web resursa .....	15
2.3.2 Truljenje veza ( <i>linkrot</i> ).....	16
2.4 Zahtjevi na dizajn URI-ja.....	16
2.4.1 Smislenost .....	16
2.4.2 Trajnost.....	16
2.4.3 Dobra ustrojenost .....	17
2.4.4 Kratkoća.....	17
2.4.5 Čitljivost .....	18
2.4.6 Pamtljivost .....	18
2.4.7 Izgovorljivost .....	18
2.5 Preporuke za dizajn URI-ja.....	18
2.5.1 Leksička ograničenja .....	18
2.5.2 Osjetljivost na mala/velika slova .....	19
2.5.3 Sadržaj URI-ja .....	19
2.5.4 Dizajn puta .....	20
2.5.5 Dizajn upita .....	20
2.5.6 Višejezičnost .....	21
3 Arhitektura Web sjedišta.....	22
3.1 Stranica i sjedište.....	22
3.1.1 Hipermehija.....	22
3.1.2 Web stranica .....	22
3.1.3 Web sjedište .....	23
3.1.4 Sastočci Web sjedišta.....	25
3.1.5 Web aplikacija.....	25
3.2 <i>Hypertext Transfer Protocol</i> .....	26
3.2.1 Poruke .....	26
3.2.2 Metode GET i POST .....	27
3.2.3 Uspješan ishod .....	27
3.2.4 Preusmjeravanje na drugi URI.....	28
3.2.5 Greške kod klijenta .....	30
3.3 Korisnički doživljaj Weba .....	30
3.3.1 Računarska korisnička sučelja.....	30

3.3.2 Metafore Weba .....	31
3.3.3 Kontekst.....	33
3.3.4 Uporaba i značenje stranice .....	33
3.4 Navigacija .....	34
3.4.1 Gdje sam? .....	34
3.4.2 Gdje sam bio?.....	35
3.4.3 Kamo mogu ići? .....	35
3.5 Ustroj Web sjedišta .....	36
3.5.1 Identitet resursa .....	36
3.5.2 Identifikator resursa .....	36
3.5.3 Položaj resursa .....	37
3.5.4 Sastav resursa .....	37
<b>4 Pregled modela ustroja Web sjedišta .....</b>	<b>39</b>
4.1 Klasični model ustroja.....	39
4.1.1 Datotečni sustav .....	39
4.1.2 Web poslužitelji temeljeni na datotečnom sustavu.....	39
4.1.3 Uporaba datotečnog sustava pri ustroju sjedišta .....	40
4.2 <i>Structured Graph Format</i> .....	41
4.2.1 Opis .....	41
4.2.2 XML zapis .....	41
4.3 WebML – <i>Web modeling language</i> .....	42
4.3.1 Perspektive .....	42
4.3.2 Hipertekstualni model .....	43
4.3.3 Primjer WebML-a .....	44
4.4 SiteBrain .....	44
4.4.1 Model " <i>The Brain</i> " .....	45
4.4.2 Primjena pri navigaciji.....	45
4.4.3 Navigacijske kvalitete.....	45
4.5 Oracle Internet Application Server.....	46
4.5.1 Oracle Portal.....	46
4.5.2 Portleti.....	46
4.6 Usporedba i ocjena odabralih modela .....	47
4.6.1 Sveobuhvatnost .....	47
4.6.2 Model identifikacije resursa.....	48
4.6.3 Model identiteta resursa.....	48
4.6.4 Model položaja resursa .....	48
4.6.5 Model sastava resursa .....	48
4.6.6 Povezanost s poslužiteljem .....	49
4.6.7 Višejezičnost .....	49
4.6.8 Pregled .....	50
<b>5 Model UriGraph .....</b>	<b>52</b>
5.1 Načela UriGraph-a .....	52
5.1.1 Model identifikacije resursa.....	52
5.1.2 Ulaz i izlaz modela .....	52
5.1.3 Slojevi modela .....	53
5.2 Topološki sloj .....	53
5.2.1 Matematička definicija.....	53
5.2.2 Grafički prikaz .....	54
5.2.3 Prolaz kroz graf .....	55
5.3 Sloj analize zahtjeva .....	55

5.3.1 Sastav zahtjeva .....	55
5.3.2 Obrada zahtjeva u čvoru.....	56
5.3.3 Odabir odredišnih čvorova .....	56
5.3.4 Vrste prijelaza .....	57
5.3.5 Propusnice kroz prijelaze .....	57
5.3.6 Prvenstva grana.....	58
5.3.7 Model prvenstva HNL .....	58
5.3.8 Iznimni slučajevi pri analizi.....	60
5.4 Sloj sinteze odgovora .....	60
5.4.1 Identitet resursa .....	60
5.4.2 Sastav resursa .....	61
5.4.3 Natuknice o identitetu komponente.....	61
5.5 Uzorci u grafu.....	62
5.5.1 Granajući prijelazi .....	62
5.5.2 Povratni prijelaz .....	63
5.5.3 Uzastopni prijelazi.....	63
5.6 Ocjena UriGraph-a.....	64
5.6.1 Sveobuhvatnost .....	64
5.6.2 Identifikacija .....	64
5.6.3 Identitet .....	64
5.6.4 Položaj .....	65
5.6.5 Sastav resursa .....	65
5.6.6 Povezanost s poslužiteljem.....	66
5.6.7 Višejezičnost.....	66
5.6.8 Ponovna usporedba modela .....	66
<b>6 Programska izvedba .....</b>	<b>68</b>
6.1 Okruženje.....	68
6.1.1 <i>Listener</i> .....	69
6.1.2 Glavni poslužiteljski modul.....	69
6.1.3 Webspot.....	69
6.1.4 Izvor izgleda.....	70
6.2 Izvedba ustroja po UriGraph-u u Javi .....	70
6.2.1 Klase sučelja ustroja .....	70
6.2.2 Vrste čvorova i topologija.....	70
6.2.3 Modeliranje propusnica.....	71
6.2.4 Modeliranje natuknica .....	72
6.2.5 Izvedba analize.....	73
6.2.6 Višejezična analiza .....	74
6.2.7 Modeliranje komponenti.....	74
6.2.8 Izvedba sinteze.....	75
6.2.9 Rukovanje iznimkama.....	75
6.3 XML shema za opis modela .....	76
6.3.1 Osnovni element.....	76
6.3.2 Topologija .....	77
6.3.3 Propusnice .....	77
6.3.4 Komponente .....	78
6.3.5 Natuknice .....	78
6.3.6 Primjer XML dokumenta .....	78
6.4 Ispitivanje ispravnosti.....	79
<b>7 Studijski slučaj .....</b>	<b>81</b>
7.1 Zahtjevi .....	81

7.1.1 Predmetni sustav .....	81
7.1.2 Identitet resursa .....	82
7.1.3 Identifikacija resursa .....	82
7.1.4 Položaj resursa .....	83
7.1.5 Sastav resursa .....	83
7.2 Izvedba u UriGraph-u .....	84
7.2.1 Predmetni sustav .....	84
7.2.2 Topološki sloj .....	84
7.2.3 Sloj analize zahtjeva .....	85
7.2.4 Sloj sinteze odgovora.....	85
7.2.5 Zapis ustroja u XML-u.....	86
7.3 Primjeri rada poslužitelja.....	88
7.3.1 Jednostavan zahtjev sadržaja.....	89
7.3.2 Zahtjev za transformiranim odgovorom.....	90
7.3.3 Zahtjev za izvršnim resursom .....	91
7.3.4 Zahtjev za slikom .....	92
8 Zaključak .....	94
Literatura .....	95
Životopis autora .....	98
Kratki sažetak .....	99
Short summary .....	100
Ključne riječi .....	101
Keywords .....	102

# 1 Uvod

U razvoju složenih *World Wide Web* usluga posebna pažnja se mora posvetiti ustrojavanju, odnosno povezivanju korisničkog sučelja, sadržaja i funkcionalnosti Web sjedišta. Dobro promišljen i izведен model ustroja olakšava razvoj i održavanje svake Web usluge, povećava usklađenost dijelova i smanjuje unutarnju zalihost. Svi ti efekti imaju jači učinak kod složenijih Web aplikacija.

Danas korištene tehnologije Web aplikacija i njihovi modeli ustroja uglavnom se temelje na datotečnom sustavu. Takav klasičan pristup, kao i neki drugi opisani u ovom radu, ne uspijevaju ispuniti sve kriterije za proglašavanje potpunim modelom ustroja. U tom cilju se u ovom radu predlaže novi model ustroja, "UriGraph".

Rad se uvelike oslanja na:

- W3C standarde u normativu i duhu, naročito URI, HTTP i XML obitelji standarda;
- preporuke znanstvenih i stručnih autoriteta o teoriji i praksi Weba;
- programski jezik Java i općenito J2EE razvojnu platformu.

Rad je organiziran u ukupno osam poglavlja, uključujući ovaj uvod i zaključak. Sljedeća dva poglavlja daju teorijsku podlogu, zatim slijedi pregled postojećih modela ustroja. U drugoj polovici rada se predstavlja autorovo rješenje modela ustroja.

Drugo poglavlje "Resursi i identifikatori", pokušava razjasniti pojmove vezane uz Web resurse. Počinje se od osnovnog pojma resursa, njegovog identifikatora (URI-ja) i identiteta. Zatim slijede zahtjevi i preporuke na dizajn URI-ja.

Treće poglavlje govori o arhitekturi Web sjedišta. Sjedište je prvo definirano i analizirano, zatim je opisan protokol za komunikaciju na Webu (HTTP).

Četvrto poglavlje predstavlja pet odabranih modela ustroja sjedišta: klasičan model, SGF, WebML, SiteBrain i iAS. Svaki od njih se opisuje i na kraju se uspoređuju prema unaprijed zadanim kriterijima za model ustroja.

Peto poglavlje opisuje model "UriGraph". Model je rastavljen na tri sloja: topološki sloj, sloj analize zahtjeva i sloj sinteze odgovora koji se redom definiraju. Zatim je objašnjeno

nekoliko čestih uzoraka u takvom grafu i na kraju poglavlja je dana ocjena modela i usporedba sa ostalim modelima ustroja iz četvrтog poglavlja.

Šesto poglavlje prikazuje osnove programske izvedbe UriGraph-a u Javi i odgovarajuće XML sheme. Prvo je predstavljen ukupan sklop Web aplikacije, a zatim njegov dio zadužen za ustroj sjedišta.

Sedmo poglavlje pokazuje primjenu UriGraph-a kroz jedan studijski slučaj (*case study*). Kroz izradu tog slučaja i primjere rada poslužitelja ilustrira se praktična funkcija modela.

## 2 Resursi i identifikatori

*Središnja ideja ovog rada je da se ustrojavanje Web sjedišta svodi na definiranje, identifikaciju i određivanje međuodnosa i sastava u njemu sadržanih resursa. Zato na početku rada uvodimo pojmove o resursima na Webu.*

### 2.1 Web resurs

#### 2.1.1 Pojam resursa

"Resurs" (engleski "*resource*") je česta, ali mnogima nejasna riječ. Kaže se: "šuma je prirodni resurs", "fakultetska knjižnica se pokazala vrijednim resursom u mom radu", "tvrtka nema dovoljno ljudskih i finansijskih resursa za dovršetak posla".

Stožerni dokument po pitanju resursa na Webu [rfc2396] kaže da "resurs može biti sve što ima identitet (...) Resurs je konceptualno preslikavanje na entitet ili skup entiteta". Ovaj opis je u literaturi često kritiziran i po mom mišljenju beskoristan za razumijevanje pojma resursa općenito, kao i Web resursa posebno. Stoga evo moje definicije:

Resurs je **trajno dostupan izvor koristi**.

"**Trajno**" ne znači nužno vječno – svi materijalni resursi su ograničenog kapaciteta i vijeka. Trajnost se odnosi na željeni vremenski raspon, praktično predvidivo vrijeme korištenja. Od nekih resursa se može tražiti da traju stoljećima, kod drugih se "trajno" može odnositi na svega nekoliko sati (npr. pozornica postavljena samo za vrijeme trajanja priredbe).

Biti "**dostupan**" znači biti prisutan i pripravan za uporabu.

Riječ "**izvor**" odvaja resurs od izravnog čimbenika koristi (manifestacije resursa, kako ga naziva [wca99]), bila ona tvar, energija ili informacija. Resurs je more, a ne riba; elektrana, a ne električka energija; knjiga, a ne izjava. U stručnoj literaturi o Internetskim resursima ovo se svojstvo resursa često propušta.

"**Korist**" je vjerojatno najproblematičnija riječ u ovoj definiciji. Engleski rječnici koriste "*help*", "*aid*", "*support*" i "*consolation*", koje bi bolje odgovarali hrvatskoj riječi "pomoći". Međutim, "pomoći" implicira nevolju i nuždu, a resursi se ne koriste samo u takvim slučajevima. Korisnost je inherentno subjektivna: što je jednoj civilizaciji lokva

smrdljivog blata drugoj je energetski resurs, a zvjezdano nebo je navigacijski resurs samo pojedincima.

Vidimo da je prozivanje objekta "resursom" subjektivna procjena, vremenski varljiva. Resursu je teško egzaktno utvrditi granice, kao i razložiti ga na "elementarne" resurse.

### 2.1.2 Resursi koji predstavljaju pojmove

Iako se u običnom govoru češće spominju resursi materije, u dalnjem radu ograničit ćemo se na **resurse informacija**<sup>1</sup>, trajno dostupne izvore korisnih informacija. Primjer takvog resursa je knjiga, knjigovodstveni softver, meteorološki zavod.

Važna uloga resursa informacija je **predstavljanje pojma**. Za svaki pojam možemo zamisliti i resurs koji daje informacije vezane uz taj pojam. Pojam  $X$  se klasično određuje preko dva skupa:

- **opseg** (*extension*) – skup objekata koji su  $X$ ;
- **sadržaj** (*intension*) – skup nužnih uvjeta dovoljnih da bi neki predmet bio  $X$ .

Ova dva skupa su međusobno potpuno ovisna: što se sadržaj pojma povećava, opseg mu je manji, i obrnuto. Pojam  $Y$  je **uži** ili konkretniji od pojma  $X$  ako mu je opseg podskup  $X$ -ovog opsega, tj. ako mu je sadržaj nadskup  $X$ -ovog sadržaja.

### 2.1.3 Odnosi među resursima

Resurs koji daje informacije o nekom širem pojmu očito bi trebao dati informacije i o svim njemu užim pojmovima da bi bio potpun. Praktičnije, možemo reći da resurs koji predstavlja širi pojam (nazovimo ga nadresursom) koristi resurse koje predstavljaju odgovarajuće uže pojmove (podresurse) kako bi pružio potpunu informaciju.

Šire shvaćanje odnosa nadređenosti među pojmovima dovodi do pojma **agregatne relacije**. Agregatna relacija jednom pojmu  $X$  pridružuje skup više međusobno bliskih pojmovevih  $\{Y_i\}$  – takav odnos se često opisuje kao "jedan na više" ili "jedan na  $N$ ". Koristit ćemo izraz "pojam  $X$  je agregatno nadređen pojmu  $Y_i$ ".

Relacija konkretizacije je jedan poseban oblik aggregatne relacije: ona zadanom pojmu pridružuje skup njemu užih pojmovevih. Na primjer, pojam "roman Marka Twaina" agregatno je nadređen pojmovima "roman 'Adventures of Tom Sawyer'", "roman

---

<sup>1</sup> Alternativni naziv bi bio "informacijski resurs", kao suprotno od "materijalni resurs". Problem je što je npr. knjiga materijalna, pa bi je trebalo zvati materijalnim resursom informacija, a ne informacijskim resursom koji je materijalan.

*'Adventures of Huckleberry Finn'* i tako dalje. Druga česta agregatna relacija je pridruživanje cjeline skupu njegovih dijelova – tako je pojmu "roman *'Adventures of Tom Sawyer'*" pridružen skup poglavlja tog romana. Postoje mnogi posebni oblici ovakve relacije – na primjer, jednom mentoru može biti pridružen skup njegovih studenata.

Važno je istaknuti kako pojam *X* iako nadređen pojmu *Y* po jednoj aggregatnoj relaciji, može se po nekoj drugoj naći u obrnutoj situaciji. Jednom studentu (*A*) se tako može pridružiti skup profesora koji mu trenutno predaju i taj skup može uključivati i njegovog mentora (*B*). Druga relacija, koja povezuje mentora *B* sa svim njemu dodijeljenim studentima, sadrži i pojam *A*, ovaj put kao podređen pojam.

### 2.1.4 Dinamičnost resursa informacija

Sam podatak koji pribavljamo od resursa informacija s vremenom se može mijenjati. Sat je ekstreman primjer – vremenska promjenjivost je upravo njegova bit. Od drugih resursa očekujemo da daju nepromjenjivu informaciju: video kazeta sa snimkom rođendana djeteta ili primjerak zbirke Shakespeareovih soneta. Promjenjivost sama može varirati kod konkretizacije resursa: Ustav Republike Hrvatske je ustanova koja je po svojoj prirodi živa i promjenjiva sadržaja; međutim, formulacija Ustava objavljena u "Narodnim novinama" broj 28/2001 stalan je povjesni i pravni dokument.

Dakle, sam resurs možemo smatrati promjenjivim ili nepromjenjivim. Češće mijenjani resursi se obično nazivaju **dinamičnim**, a rjeđe mijenjani **statičnim** resursima. Iako se ova dva naziva često koriste kao absolutne kategorije, očito je riječ o kontinuiranoj skali.

### 2.1.5 Resursi na Internetu

Resursi o kojima govori ovaj rad su dio Interneta i spadaju u kategoriju **mrežno dohvatljivih** (*network-retrievable*) resursa informacija. Govorit ćemo o Internetskim ili Web resursima, jer su definirani za cijeli Internet, ali ih u ovom radu uvijek gledamo kao dio Weba. Budući da je Web zasnovan na arhitekturi klijent-poslužitelj, koristiti ćemo izraz "resurs *X* poslužuje *Y*" u slučaju kad je *X* izvor informacija vrste *Y*.

Praktično, možemo primijetiti da se na Web sjedištima slični resursi često grupiraju u **klase** resursa. Na primjer, svaka knjiga u sjedištu knjižare može biti opisana i predstavljena na sličan način, varirajući samo u podacima (naslovu, autoru, cijeni i sl.). Zbog tih sličnosti, tehničko ostvarenje sjedišta će sadržavati funkcionalnost vezanu uz samu klasu resursa, za razliku od svakog resursa posebno.

## 2.2 Uniform Resource Identifier

*Uniform Resource Identifier* (URI) je općeprihvaćeni Internetski standard za identifikaciju resursa (iako bi ih vjerojatno bilo bolje nazivati referencama nego identifikatorima). Osnovna sintaksa i semantika je definirana u [rfc2396]. Više URI-ja može referencirati isti resurs, ali da bi se resurs zvao Internetskim, potrebno je da ga barem jedan URI referencira. Isti URI ne može referencirati više resursa odjednom.

URI je niz znakova iz ograničenog skupa, uglavnom slova latinice sa znamenkama i nekoliko posebnih znakova, koji podliježe posebnim sintaktičnim pravilima. Osnovna struktura (apsolutnog) URI-ja je jednostavna:

*<naziv sheme> : <ostatak specifičan za tu shemu>*

**Shema** je potprostor imena koji ima svoju posebnu sintaksu oblikovanja ostatka URI-ja. Središnja shema za Web je "http", nazvana po istoimenom protokolu.

Dvije posebne vrste URI-ja su **lokatori** (URL-ovi) i **imena** (URN-ovi). URL-ovi identificiraju resurs navodeći način dohvatanja resursa (npr. mrežnu "lokaciju"). URN-ovi imaju dodatno ograničenje na trajnost: kad se jednom ustanovi što URN referencira, taj odnos se više ne smije mijenjati [rfc2141]. Iako su URI-ji na Webu obično URL-ovi, raspravljati ćemo o identifikatorima resursa općenito i stoga koristiti odgovarajući pojam.

Važna značajka URI-ja je njegova **prepisivost** (*transcribability*), sposobnost prenošenja informacije u URI-ju preko različitih sredstava. Važno je da ga ljudi mogu zapisati u nedigitalnom obliku (npr. olovkom na papir) ili ga zapamtiti, da bi ga naknadno unijeli (npr. utipkali) u mrežni uređaj priključen na Web.

Druga važna značajka je **neprozirnost** (*opacity*) dijelova URI-ja [tbl96]. Standardima je definirano koji dijelovi URI-ja sadrže informacije za određeni aktivni element mreže, a koji dijelovi bi za njih trebali biti "neprozirni", "crne kutije". Elementi mreže ne bi trebali pokušavati izvlačiti informacije iz tih dijelova URI-ja niti bi ga trebali mijenjati, jer značenje tih dijelova URI-ja nije utvrđeno u njihovom kontekstu.

Svaka shema za sebe može utvrditi pravila po kojem su dva URI-ja smatrali jednakima, iako možda nemaju jednak leksički zapis (znak po znak). Dodatno se može u shemi utvrditi i **normalizacija**, postupak utvrđivanja normalnog (kanonskog) URI-ja iz skupa svih međusobno jednakih URI-ja u toj shemi.

## 2.2.1 Standardni URI

Unutar prostora URI-ja moguće je definirati neograničen broj shema i svaka može imati svoj način strukturiranja ostatka URI-ja. Međutim, većina shema (uključujući i shemu "http") ostatak strukturira ovako:

//<autoritet><put>?<upit>

**Autoritet** predstavlja prvu razinu hijerarhije u prostoru sheme. On upravlja prostorom imena (*namespace*) koji je definiran ostatkom URI-ja.

**Put** je sastavljen od **segmenata** međusobno odvojenih kosom crtom, zamišljenih da određuju put kroz hijerarhiju (poput puta kroz stablo direktorija u klasičnom datotečnom sustavu).

Posljednji dio, **upit** (*query*), definiran je kao "informacija koju će interpretirati resurs" [rfc2396].

Uzmimo jednostavan primjer URI-ja:

`http://www.hr/search?Zagreb`

Prvo se utvrđuje shema ("http"). Unutar te sheme traži se autoritet identificiran imenom "www.hr" i prenosi mu se ostatak URI-ja: put "search" i upit "Zagreb".

Primijetimo da iz definicije URI-ja proizlazi da je upit neproziran za sve resurse na Webu osim ciljnog. Nedostatak ovakve definicije je implikacija da upit nije potreban za identifikaciju resursa, te bi se moglo zaključiti da ne bi trebao biti u identifikatoru, što čini proturječje.

## 2.2.2 HTTP-ov URI

*Hypertext Transfer Protocol* [rfc2616] koristi opću sintaksu URI-ja, gdje se autoritetom smatra *host name* s portom. Ne definira se detaljnija struktura upita, ali klijenti i poslužitelji obično koriste sljedeću notaciju. Upit se sastoji od segmenata međusobno odvojenih znakom "&". Svaki segment se može sastojati od dva dijela, naziva i vrijednosti, međusobno odijeljenih znakom "=".

Na primjer, u upitu "`id=13&detailed`" postoje dva segmenta: prvi ima naziv "`id`" i vrijednost "`13`", dok drugi ima samo naziv "`detailed`".

Specifikacija HTTP/1.1 preporuča da cijeli URI bude osjetljiv na mala/velika slova (*case-sensitive*), osim naziva poslužitelja i sheme, koji moraju biti neosjetljivi za HTTP-ovskog

klijenta. Standard se odnosi na jednakost URI-ja sa strane klijenta. Normalizacija nije definirana.

## 2.3 Identitet Web resursa

Svaki Web resurs ima svoj **identitet** (*identity*), odgovor na općenita pitanja "Što ovaj resurs predstavlja?" i/ili "Što ovaj resurs poslužuje?". Budući da je resurs samo subjektivna apstrakcija konkretnog objekta, i doživljaj tog identiteta ovisi o promatraču. Resursu identitet zadaje njegov vlasnik.

### 2.3.1 Ugled Web resursa

Svaki resurs može imati **ugled** (*reputation*). Ugled je popularni doživljaj identiteta (i nekih drugih svojstava) resursa sa strane njegovih korisnika. Resurs može steći ugled na tri načina:

- tako da neki autoritet (vlasnik resursa ili ugledni Web katalog) proglaši njegov identitet;
- tako da se stvore neautoritarne veze na njega koje sugeriraju njegov identitet;
- tako da resurs poslužuje sadržaj kojim sugerira svoj identitet.

Na primjer, na URI-ju "[http://prognoza.hr/hrdanas\\_n.html](http://prognoza.hr/hrdanas_n.html)" nalazi se Web stranica koja poslužuje dnevnu prognozu Državnog hidrometeorološkog zavoda za cijelu Hrvatsku. Pretpostavimo, dakle, da je vlasnik ovom resursu zadao identitet "vremenska prognoza za Hrvatsku za danas". To još uvijek ne znači da je identitet te stranice jasan svakom Web korisniku.

Na odgovarajućoj stranici vlasnika (DHMZ-a) postoji veza "Hrvatska danas" u kontekstu "Prognoze" na ovu stranicu, što poprilično jasno prenosi zamišljeni identitet. Sama stranica poslužuje sadržaj koji u trenutku pisanja ovog teksta je naslovljen s: "Prognoza vremena za Hrvatsku za 18.12.2001", što očito sugerira drugačiji (krivi) identitet. Općenito, nigdje u sadržaju se trenutno ne spominje ključna riječ "danasa", što znatno smanjuje šanse kvalitetnog kotiranja URI-ja među Web pretraživačima.

Kako se korisnici upoznaju s novim resursom (i njegovim URI-jima), s vremenom se njegov ugled **utvrđuje**. URI se zapiše, zapamti, postave se veze na drugim Web stranicama, pretraživači registriraju URI i dobiveni entitet u svoje baze, korisnici ga dodaju u *bookmark*-e, poneki se čak objave u medijima.

### 2.3.2 Truljenje veza (*linkrot*)

Uz ugled URI-ja je usko povezana "bolest" Weba zvana *linkrot*, doslovno "truljenje veza" [nie98]. Nakon što je ugled resursa već utvrđen, odgovori na njegovom URI-ju postanu nespojivi s popularno utvrđenim ugledom. Veze na taj resurs postanu neupotrebljive i otud naziv "trula veza".

Web resurs može prestati posluživati na utvrđenom URI-ju i takav se URI tada naziva "slomljenom vezom" (*broken link*). Kad resurs prestane ažurirati svoj sadržaj naziva se "mrtvim". Ponekad, URI-ji promijene vlasnika koji im potpuno predefinira identitet. Sve su to uzroci truljenja veza.

## 2.4 Zahtjevi na dizajn URI-ja

Iako se sam standard [rfc2396] pobrinuo za ispunjenje nekih konstrukcijskih zahtjeva (npr. prepisivost), na Web arhitektu još ostaje veliki posao dodjeljivanja pojedinih URI-ja resursima na svom sjedištu odnosno smišljanja pravila za stvaranje URI-ja koji se dodjeljuju klasi resursa. Takav postupak se u zadnje vrijeme naziva **dizajnom** URI-ja, odnosno dizajnom prostora imena URI-ja [tbl92] [nie00]. U ovom poglavlju predlažem željena dizajnerska svojstva URI-ja, navedena od najvažnijih prema manje važnima.

### 2.4.1 Smislenost

Smislenost (*meaningfulness*) je jedino dizajnersko svojstvo navedeno (i preporučeno) u [rfc2396], a odnosi se na logičke veze s referenciranim pojmom. Ljudi su skloni u referencama tražiti logičke veze s referenciranim pojmom u nekom (jezičnom, organizacijskom, tehničkom) kontekstu. Na primjer, "**fer.hr**" je URI blizak pojmu Fakulteta elektrotehnike i računarstva, jer je "FER" uobičajena kratica za Fakultet, a CARNet registrira takve ustanove po shemi "*naziv*.hr". Zato je takav URI smislen.

### 2.4.2 Trajnost

Tim Berners-Lee kao jedan od aksioma URI-ja navodi da URI treba neprestano referencirati "istu stvar", a da početno značenje što URI referencira određuje vlasnik URI-ja [tbl96]. Ovo svojstvo se u literaturi obično naziva **trajnost** (*persistence*). Trajnom URI-ju bit će lakše i točnije utvrđen ugled.

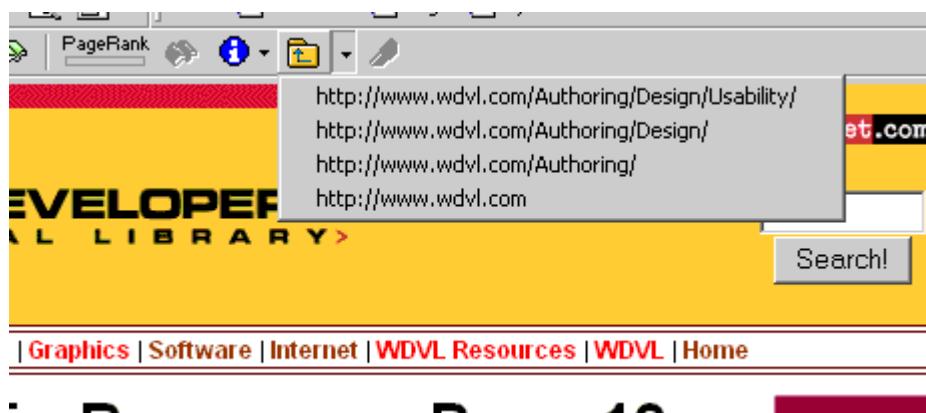
Netrajnost URI-ja može biti posljedica više događaja. U prvom slučaju, sam resurs može biti nedovoljno trajan (npr. ukinuta usluga radi pomanjkanja financija), što nadilazi ovlasti dizajnera URI-ja.

U drugom slučaju resurs i dalje postoji, ali nije dobavljen starim (nego možda nekim drugim) URI-jem. Klasičan razlog da resurs promjeni URI je njegov loš dizajn.

U trećem slučaju resurs postoji i dobavljen je istim URI-jem, ali njegov novi identitet više ne odgovara starom ugledu. Izmjena zadanog identiteta ili loše upravljanje stečenim ugledom očito dovode do ovakvih problema. Primjer: URI "university.tld/~joe" može dobiti ugled kao "osobna stranica studenta Joe D.", kao što npr. piše na samoj stranici. S druge strane, stvarni identitet resursa s tim URI-jem kako ga je zamislio vlasnik (sveučilište) može biti "stranica našeg korisnika s *username*-om **joe**". Promjena u osobi koja ima takvo korisničko ime dovesti će do odstupanja između starog ugleda stranice i njegovog novog sadržaja.

### 2.4.3 Dobra ustrojenost

Ustroj dobro dizajniranih URI-ja resursa u sjedištu treba odgovarati ustroju sjedišta. Put URI-ja obično predstavlja put kroz hijerarhiju ustroja. Korisnici često odsijecaju kraj URI-ju kako bi prešli na višu razinu u hijerarhiji sjedišta. Popularni dodatak preglednicima *Google Toolbar* ima alat "*Up one level*" koji omogućuje brzo odsijecanje zadnjeg segmenta puta trenutnog URI-ja (Slika 1).



Slika 1: Primjena alata "*Up one level*" s *Google Toolbar*-a na stranici  
["http://www.wdvl.com/Authoring/Design/Usability/use3\\_3.html"](http://www.wdvl.com/Authoring/Design/Usability/use3_3.html)

### 2.4.4 Kratkoća

Uz to što prirodno ide uz neka druga dobra svojstva (npr. pamtljivost), kratkoća ima još prednosti:

- kratki URI-ji se lakše utipkuju, npr. u samo adresno polje preglednika;
- manja je vjerojatnost da će se URI prelomiti i njegov dio prijeći u novi red teksta, što je pogotovo važno kod e-pošte (takve veze neće raditi), ali i u novinama, časopisima i drugim publikacijama s užim stupcem teksta;

- kratki URI-ji se općenito bolje uklapaju u skučene uvjete unosa, prikaza i spremanja kod manjih digitalnih uređaja: mobilnih telefona (npr. SMS), ručnih računala i slično.

Nažalost, kratkoća je često u sukobu sa smislenošću i dobrom ustrojenosti.

### **2.4.5 Čitljivost**

Sposobnost točne vizualne percepcije zapisanog URI-ja (na zaslonu ili na papiru) naziva se "čitljivost". Čitljivost smanjuje greške prilikom pretipkavanja i drugih načina umnažanja URI-a.

### **2.4.6 Pamtljivost**

Pamtljivost vrlo ovisi o drugim svojstvima: jasno je da se lakše pamte kraći i smisleniji identifikatori.

### **2.4.7 Izgovorljivost**

Ljudi često spominju URI-je koje uspiju zapamtiti u razgovoru, čitaju ih drugima preko telefona ili ih izgovaraju na radiju ili televiziji.

## **2.5 Preporuke za dizajn URI-ja**

Na temelju poželjnih svojstava u nastavku su navedene neke preporuke koje bi trebale poboljšati dizajn URI-ja, prvenstveno značajno ograničavajući njegovu strukturu.

### **2.5.1 Leksička ograničenja**

Leksička ograničenja su dodatna ograničenja skupa dozvoljenih znakova koja povećavaju čitljivost, izgovorljivost i pamtljivost.

U putu URI-ja preporučam korištenje samo sljedećih znakova:

- mala slova engleske abecede (od 'a' do 'z');
- velika slova engleske abecede (od 'A' do 'Z');
- dekadske znamenke (od '0' do '9');
- kosa crta ('/') – za razdvajanje segmenata puta;
- crtica ('-') – za razdvajanje riječi unutar segmenta;
- točka ('.') – za posebne potrebe (npr. oblikovanje datuma), kao i za oznake vrste dokumenta (u produžetku imena stranice, npr. ".pdf").

Podvlaku ('\_') ne bi trebalo koristiti zbog neobičnosti (ne koristi se u klasičnoj tipografiji) i sličnosti s crticom. Podvlaka se i inače ne smije koristiti u nazivu poslužitelja, pa bi bilo konzistentno ne koristiti je niti drugdje u URI-ju.

Zarez (',') je sličan točki i nije uobičajen u URI-jima.

U upitu se slična ograničenja mogu koristiti kod naziva segmenta. Kod vrijednosti segmenta upita ograničenja nisu praktična, jer se u njima ponekad prenosi proizvoljni niz znakova (npr. vrijednost polja iz HTML obrasca).

### 2.5.2 Osjetljivost na mala/velika slova

Ozbiljna prepreka pamtljivosti i izgovorljivosti URI-ja proizlazi iz osjetljivosti URI-ja na mala i velika slova (*case-sensitivity*) [nie99]. Budući da standardi već propisuju neosjetljivost za dijelove URI-ja (poput sheme i naziva domaćina), sustavno je i korisno pravilo proširiti na što veće područje URI-ja. Ponovo su vrijednosti segmenta upita izuzeta, zbog svoje posebne funkcije.

### 2.5.3 Sadržaj URI-ja

Temeljni postulat ovog rada je da svaki URI, kao i svaki dobar identifikator, treba prenositi samo informaciju o identitetu resursa; ništa manje i ništa više.

Postoje tri osnovne vrste pogrešnog oblikovanja sadržaja URI-ja:

- **suvišna** informacija – ona koja nije o identitetu resursa;
- **zalihosna** (redundantna) informacija o identitetu resursa;
- **nepostojeća** informacija o identitetu resursa.

Uvođenje suvišne, nerelevantne informacije u identifikator je najčešći i najbolniji grijeh dizajnera koji u pravilu dovodi do netrajinih, dugih i općenito loših URI-ja. URI ne bi smio sadržavati:

- **tehničke** detalje: sufikse (npr. "`html`", "`asp`") ili segmente ("`cgi-bin`", "`pgsql`") koji su podložni promjeni iz tehničkih razloga;
- informacije o **internoj organizaciji**, poput naziva odjela ili imena osobe koja se brine o tom resursu;
- promjenjiva **svojstva** resursa: često se pronalaze riječi poput "novo", "povjerljivo" i slično u URI-jima resursa koji očito neće vječno biti takvi.

Zalihosna informacija u URI-ju znači da se dio identiteta resursa predstavlja dijelom URI-ja više nego što je neophodno, odnosno više nego jednom. Kao primjer navedimo zamišljeni put "/1998/log/27.05.98" koji sadrži zalihosnu informaciju o godini. Bolje bi bilo "/log/1998/5/27". URI kojemu je uklonjena nepotrebna zalihost očito je kraći i može biti bolje osmišljen.

Konačno, nedostatak potrebne informacije u URI-ju može se pokazati kao ozbiljan uzrok netrajnosti. Uzmimo za primjer izmišljeno sveučilište koje održava konferenciju "WebConf 2002". Dizajner je za konferenciju te godine neprimjereno odabrao URI "university.tld/webConf", da bi dogodine isti resurs premjestio na URI "university.tld/webConf/2002" jer je morao nekamo smjestiti i "WebConf 2003". Prvi URI nije bio primjereno smislen, a dizajner kratkovidan.

#### **2.5.4 Dizajn puta**

Prvenstvena namjena puta je predstavljanje agregatnih veza među resursima. Za primjer u poglavlju 2.1.2 možemo dizajnirati sljedeće putove URI-ja:

```
/mark-twain/roman  
/mark-twain/roman/tom-sawyer  
/mark-twain/roman/tom-sawyer/1.poglavlje
```

Iste agregatne veze su prirodan put za ustroj sjedišta.

#### **2.5.5 Dizajn upita**

Upit je namijenjen za detaljnije atribute koji opisuju vrlo konkretnе resurse i rijetko postaju dostupni ljudima. U slučajevima kad je upit vidljiv ljudima, s njim treba postupati oprezno.

Zbog manje preglednih znakova odvajanja ("?", "&" i "=") u upitu, dizajner treba biti naročito oprezan i štedljiv s brojem i duljinom segmenata – u protivnom, URI neće biti ni kratak, ni čitljiv, ni izgovorljiv, niti dobro ustrojen.

Zbog proturječnosti originalne definicije upita (preformulirano: "upit je dio identifikatora resursa koji taj resurs interpretira"), razumno je odustati od definicije iz [rfc2396] i prihvati upit kao ravnopravan dio URI-ja s putem. Upit je nužan i u slučajevima kada je pogodno da URI sadrži informaciju koja nema određen smještaj u nekom nizu i može se pojaviti neovisno o agregatnim vezama sjedišta.

## 2.5.6 Višejezičnost

Jezik u kojem je URI zapisan vrlo je važan za njegovu kvalitetu. Ako ljudi bolje razumiju riječi od kojih je URI sastavljen, lakše će ga pročitati, zapamtiti i izgovoriti. Prirodno je da resurs koji poslužuje sadržaj isključivo na hrvatskom jeziku ima i URI sastavljen od hrvatskih riječi.

Kod sjedišta koja poslužuju sadržaj na više jezika usporedno pojavljuju se slični resursi, međusobno različiti samo po jeziku prikazivanja. Stoga bi i njihovi URI-ji trebali biti na odgovarajućim jezicima, npr.:

```
/dalmatia/hvar/accomodation?currency=EUR  
/dalmacija/hvar/smjestaj?valuta=EUR
```

Odgovarajući segmenti u oba URI-ja imaju jednako značenje, samo su zapisani u drugačijem jeziku. Kao rezultat, sam oblik URI-ja sugerira na kojem jeziku se traži sadržaj. Neki dijelovi URI-ja ostaju isti na svim jezicima, poput nekih zemljopisnih imena i međunarodnih oznaka.

*Drugo poglavlje je razjasnilo temeljni pojam ovog rada, pojam Web resursa, i objasnilo tehničke i ljudske osobine URI-ja kao identifikatora Web resursa. U nastavku ćemo upotrijebiti to znanje pri izradi i ustrojavanju Web sjedišta.*

# 3 Arhitektura Web sjedišta

*Ovo poglavlje objašnjava osnove arhitekture Web sjedišta, počevši od definiranja osnovnih pojmova, stranice i sjedišta; dajući pregled protokola HTTP; uvid u razumijevanje korisničkog doživljaja Weba, posebno se zadržavajući na pojmovima navigacije i ustroja sjedišta.*

## 3.1 Stranica i sjedište

Stariji modeli Web arhitekture počinju od stranice kao osnovnog, čak samodostatnog koncepta, žarišta pozornosti svih struka koje se bave izradom Web materijala. U novije vrijeme sve popularniji postaju pristupi po kojima sjedište preuzima glavnu ulogu, a stranice su samo različiti pogledi na informacijski sustav koji se preko mreže doživljava kao Web sjedište. Usvajajući ovaj pristup, pokušati ćemo precizno opisati osnovne pojmove arhitekture sjedišta, naročito stranicu, sjedište i Web aplikaciju.

### 3.1.1 Hipermehdija

Hipertekstualni dokument je tekstualni dokument s aktivnim vezama (tzv. hipervezama, engleski *hyperlinks*) prema drugim dokumentima. Postepeno se taj pojam proširio na upotrebu multimedije umjesto samog teksta, tako da je skovan naziv "hipermedija".

"Hipermedijski stranični entitet" je naziv za hipermedijski dokument koji intuitivno odgovara općem pojmu stranice. Stoga:

- obična slika (npr. GIF) ne može biti hipermedijski stranični entitet ako njen format ne dozvoljava ugradnju hiperveza;
- višestrajni PDF dokument ili prostor prividne stvarnosti, iako oba imaju ugrađene hiperveze na druge resurse, ne mogu biti hipermedijski stranični entiteti jer ne odgovaraju doživljaju stranice u općem smislu.

### 3.1.2 Web stranica

Temeljni i najpopularniji Web resurs je stranica. U svakodnevnom govoru se naziv "**Web stranica**" koristi u više vrlo različitih značenja:

- **Internetski resurs**, obično predstavljen svojim identifikatorom (URI-jem);
- **hipermedijski entitet** u obliku npr. HTML tekstualne datoteke, bez ugrađenih entiteta;

- **slika** hipermedijske stranice predočene u pregledniku iz pribavljenog hipermedijskog entiteta i ugrađenih entiteta.

Iako svako značenje ima svoje opravdanje, u ovom radu koristiti ćemo definiciju koja odgovara prvom značenju:

Web stranica je Web resurs koji poslužuje hipermedijske stranične entitete.

### 3.1.3 Web sjedište

Pojam **Web sjedišta** je nešto teži pojam za definiranje, ali ne i manje važan. Prva izjava kojom ćemo pokušati opisati sjedište je najvažnija i najopćenitija:

Web sjedište je logička zbirka Web resursa.

Očito je da svaka zbirka Web resursa ne mora biti sjedište, pa ćemo dodati neka ograničenja na takvu zbirku. Za opis tih ograničenja uvodimo novi pojam: **osnovni URI (base URI)** sjedišta. Taj HTTP-ovski URI ne sadrži upit (a često niti put) i svako sjedište ima jedan takav URI.<sup>2</sup>

Definicijom osnovnog URI-ja dolazimo do prvog ograničenja (ograničenje postojanja naslovne stranice):

Osnovni URI Web sjedišta identificira Web stranicu.

Takva stranica se zove naslovnom stranicom ili **naslovnicom** (engleski izraz je *home page*) sjedišta. Dakle, kad klijent izvede zahtjev GET nad osnovnim URI-jem, mora dobiti hipermedijski stranični entitet, a ne npr. slikovni entitet ili poruku o nepronađenom resursu.

Sljedeće ograničenje nazivam **ograničenjem potprostora URI-ja**:

Svaki resurs u Web sjedištu se može identificirati proširenjem osnovnog URI-ja sjedišta.

URI se može **proširiti** na dva načina:

- dodavanjem segmenata na kraj puta; i/ili
- dodavanjem segmenata bilo gdje u upitu URI-ja.

<sup>2</sup> [wca99] i Internet Explorer sužavaju pojam osnovnog URI-ja tako da ne dozvoljavaju da sadrži put. Time se osnovni URI sjedišta svodi na HTTP-ovski autoritet (naziv domaćina i možda port).

Na primjer, "[www.hr/fakultet/fer?god=2001&hr](http://www.hr/fakultet/fer?god=2001&hr)" je proširenje URI-ja "[www.hr/fakultet?hr](http://www.hr/fakultet?hr)" dodavanjem jednog segmenta puta "**fer**" i jednog segmenta upita "**god=2001**" na prvo mjesto.

Ako je osnovni URI sjedišta "[www.hr](http://www.hr)", resurs identificiran samo URI-jem "[www.fer.hr/o\\_nama](http://www.fer.hr/o_nama)" ne može biti dio tog sjedišta, jer njegov URI nije proširenje osnovnog URI-ja. Međutim, ovo ograničenje ne nameće da resurs identificiran s "[www.hr/fakultet/fer](http://www.hr/fakultet/fer)" mora biti dio istog sjedišta, samo zato što mu je URI proširenje osnovnog URI-ja. Drugim riječima, osnovni URI određuje svoj potprostor u kojem se nalaze svi URI-ji resursa sjedišta, ali ne nužno i samo oni.

Ograničenje **unutrašnje povezanosti** sjedišta:

Za svaki resurs Web sjedišta postoji niz veza koji počinje na naslovnoj stranici sjedišta, prelazi preko nula ili više drugih stranica sjedišta i završava na tom resursu.

Dakle, do svakog resursa u sjedištu možemo doći iz naslovne stranice sjedišta, a da nikad ne napustimo sjedište. Kad se kaže "postoji niz veza", to može značiti da se neke od veza pojavljuju samo u rijetkim slučajevima (npr. u određeno vrijeme, za određene korisnike), ali ta veza teorijski mora biti ostvariva.

Ovaj pokušaj opisa je očito tehnički i teorijski. S korisničke strane se mora pripomenuti da bi svi resursi koji su dio sjedišta trebali imati zajednički identitet: vizualni, sadržajni i funkcionalni.

Korisno je definirati i **podsjedište** (engleski *subsite*):

Podsjedište nekog Web sjedišta *A* je takvo sjedište *B* čiji je skup resursa pravi podskup skupa resursa sjedišta *A*.

Iz ograničenja sjedišta možemo zaključiti da bi svako podsjedište od *A* imalo svoj osnovni URI koji bi bio proširenje *A*-ovog osnovnog URI-ja, samim tim i svoju naslovnu stranicu i svoj potprostor URI-ja<sup>3</sup>. Identitet ovog sjedišta bi očito trebao biti varijacija šireg identiteta, tako da ujedno ukazuje na pripadnost širem kontekstu i ističe svoju posebnost.

<sup>3</sup> [wca99] traži da podsjedište ima drugačijeg izdavača, iako izdavač "domaćinskog sjedišta" može nametnuti neke "široke smjernice" vezane uz uređenje podsjedišta.

### 3.1.4 Sastojci Web sjedišta

U svakom se sjedištu mogu prepoznati četiri "sastojka": sadržaj, izgled, funkcionalnost i ustroj. Svaki od tih sastojaka je relativno nezavisan i može se donekle nezavisno održavati i mijenjati.

**Sadržaj (content)** je osnovna informacija koja se predaje klijentu, razlog zbog kojeg se sjedište postavilo. Glavni izazov u vođenju dinamičnog sjedišta (sjedišta dinamičnih resursa) je česta promjena sadržaja na način koji ne zahtijeva izmjene ostalih sastojaka. Na većem sjedištu sadržaj unosi i ažurira veći broj osoba koji ga stvaraju ili prikupljaju (tzv. akviziteri informacija). Sustav upravljanja sadržajem može biti organiziran tako da je svaki akviziter posebno nadležan za ograničeni dio korpusa sadržaja.

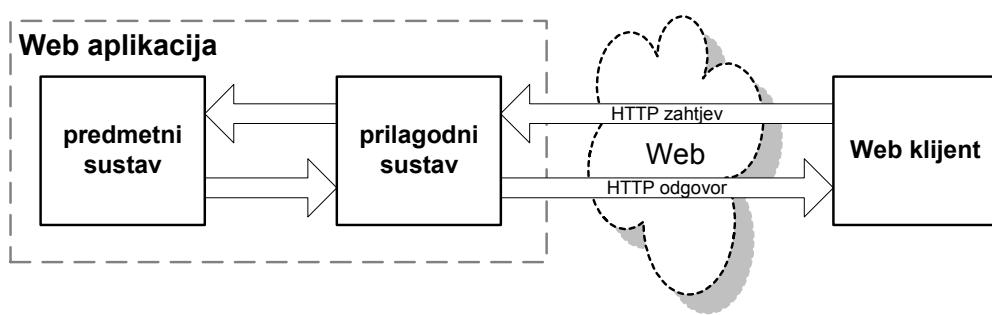
Često je sadržaj potrebno korisniku predočiti na pregledan i ugodan način. Taj sastojak sjedišta nazivamo **izgledom (look)** sjedišta. Osmišljavanjem i izvedbom izgleda se obično bave grafički dizajneri i takav jedinstven izgled se proteže na sve stranice čineći njegov vizualni identitet.

**Funkcionalnost** sjedišta pokriva obradu informacija koje se preko Web sučelja unose i prikazuju. Funkcionalnost je onaj sastojak koji od "običnog" sjedišta čini Web aplikaciju. Funkcionalnost sjedišta izgrađuju i mijenjaju programeri.

Konačno, četvrti sastojak služi kao poveznica ostalih sastojaka i sučelje prema sjedištu: **ustroj (structure)**. O njemu će više riječi biti pri kraju ovog poglavlja i u ostatku rada.

### 3.1.5 Web aplikacija

Web aplikacija je uobičajeni naziv za poslužiteljski program koji pruža uslugu na Webu. U ovom radu gledati ćemo na takvu Web aplikaciju kao na sklop dvaju sustava: predmetnog i prilagodnog sustava (Slika 2).



Slika 2: Podjela Web aplikacije na предметni i прilagodni sustav.

**Predmetni sustav** je dio Web aplikacije koji se odnosi na njen predmet bavljenja, stvari "posao" aplikacije. Podaci i operacije vezane uz taj predmet (često nazivani poslovnim

podacima i poslovnom logikom) sadržane su u potpunosti unutar takvog sustava. Predmetni sustav definira općenito sučelje preko kojeg se može koristiti, ali sam nije "svjestan" da se koristi preko Weba. Bolje rečeno, predmetni sustav ne bi trebao biti ograničen na korištenje putem Weba niti bi tehničke pojedinosti Weba trebale utjecati na njegovu funkcionalnost.

Nad već definiranim predmetnim sustavom nadograđuje se **prilagodni sustav** koji omogućuje njegovo korištenje putem Weba. On je potreban kako bi se predmetni sustav prilagodio korištenju putem Weba i sadrži potrebne tehničke i organizacijske informacije svojstvene za Web. S druge strane, prilagodni sustav ne bi smio preuzeti ili izmijeniti osnovne funkcije predmetnog sustava.

Sadržaj i funkcionalnost sjedišta prirodno se crpe iz predmetnog sustava, dok su izgled i ustroj locirani u prilagodnom sustavu.

## 3.2 Hypertext Transfer Protocol

*Hypertext Transfer Protocol* (HTTP) je aplikacijski protokol na TCP/IP složaju. On uz URI-je predstavlja tehnološku osnovicu Weba. U nastavku se ukratko navode i objašnjavajući neki dijelovi specifikacije, ne ulazeći detaljno u sintaksu i značajke protokola koje nisu važne za raspravu. Za potpunu i normativnu specifikaciju vidi [rfc2616].

### 3.2.1 Poruke

Klijent i poslužitelj komuniciraju putem poruka (*messages*). Postoje dvije vrste poruka: **zahtjev** (*request*) je poruka klijenta poslužitelju, a **odgovor** (*response*) je poruka poslužitelja klijentu uzrokovana klijentovim zahtjevom.

Poruke su oblikovane kao običan tekst. Oba tipa poruke imaju istu strukturu: poseban početni redak (*start line*), polja zaglavlja (*header fields*) i tijelo poruke (*message body*). Početni redak zahtjeva sadrži metodu i URI, dok početni redak odgovora sadrži statusni kod, a obje vrste poruke sadrže još i verziju protokola. Svako polje zaglavlja se sastoji od para (naziv, vrijednost). Konačno, moguće tijelo poruke sadrži entitet koji se prenosi (npr. HTML dokument).

Osnovna informacija HTTP **odgovora** je statusni kôd (*status code*). Statusni kôd je troznamenkasti broj koji označava ishod pokušaja razumijevanja i ispunjenja zahtjeva na koji se odgovor odnosi.

### 3.2.2 Metode GET i POST

Prva informacija u zahtjevu je naziv metode (*method*) koja se koristi, a neposredno zatim slijedi URI resursa nad kojim klijent zahtjeva izvršenje operacije. U ovom radu usredotočiti ćemo se na dvije najčešće korištene i općenito najvažnije metode HTTP-a, GET i POST. Metoda **GET** koristi se kad klijent želi dohvatiti trenutne informacije od identificiranog mrežnog resursa. **POST** služi za slanje podataka s klijenta odgovarajućem resursu na poslužitelju.

U praksi, metode se često razlikuju samo po nekim tehničkim detaljima. Prvo, GET prenosi sve podatke koje će se predati resursu u upitu URI-ja, dok POST metoda omogućuje prijenos podataka i u tijelu zahtjeva. To je dobro jer se tako mogu prenijeti veće količine podataka, ali i radi skraćenja URI-ja. S druge strane, u HTML-u aktiviranje svake obične hiperveze u dokumentu nalaže isključivo GET zahtjev, dok za pokretanje POST-a treba koristiti složenije HTML obrasce.

Važna razlika između ove dvije metode je u konceptu **nuspojava** (*side-effects*) na poslužitelju. Pojednostavljeno rečeno, zahtjev uzrokuje nuspojave ako poslužitelj zbog njega promijeni svoje stanje na način koji će se odraziti na bitan sadržaj budućih odgovora poslužitelja.

Prema [rfc2616], GET ne smije uzrokovati nikakve nuspojave – njegova jedina funkcija je da dohvati informacije. Takve metode se nazivaju **sigurnim** (*safe*) metodama za korisnika. Korisnik se korištenjem sigurnih metoda na Web-u ograda od odgovornosti za bilo kakvu promjenu važnih informacija, tj. stanja poslužitelja. GET metodom bi korisnik mogao npr. zatražiti informacije o ponudi dućana, ali ne i pokrenuti kupnju.

S druge strane, POST metoda bi se trebala koristiti kad zahtjev treba uzrokovati nuspojave. Tipični primjeri takvog zahtjeva su predaja narudžbe, unos podataka u bazu, pridodavanje komentara postajećem tekstu i slično. Takvi zahtjevi su potrebni i korisni, ali ih korisnik ne može same po sebi smatrati sigurnim. Prema standardu, korisnički agenti (preglednici) se obvezuju posebno prikazati nesigurne metode, kako bi korisnik bio upozoren na moguće posljedice svojeg zahtjeva.

### 3.2.3 Uspješan ishod

Kôdovi HTTP odgovora koji predstavljaju uspješan ishod su oblika 2xx (prva znamenka mora biti dvojka). Najčešći statusni kôd HTTP odgovora uopće je kôd **200 (OK)**. U slučaju da se odgovor odnosi na zahtjev GET, u tijelu odgovora se nalazi entitet koji

predstavlja traženu informaciju od resursa. Ta kombinacija – zahtjev GET i odgovor 200 – predstavlja najčešći oblik komunikacije u protokolu.

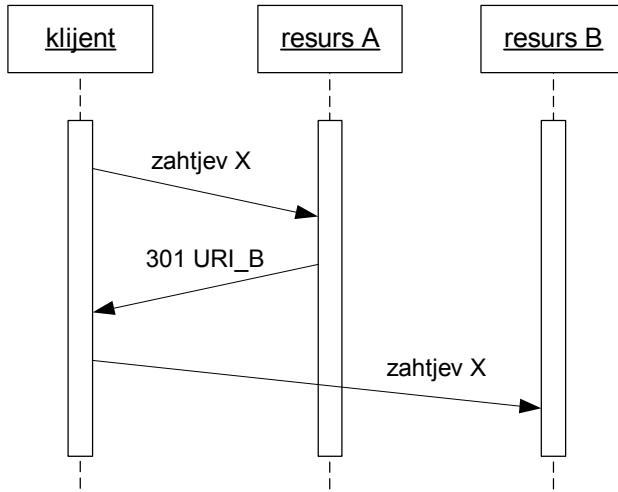
Poseban kôd je 204 (*No Content*). Prema specifikaciji ovakav odgovor znači da je poslužitelj ispunio zahtjev, ali nema potrebe da vraća ikakav entitet u tijelu odgovora (iako možda vraća ažuriranu meta-informaciju). Korisnički agent ne bi trebao promijeniti prikaz trenutnog dokumenta. Proširenje ove funkcionalnosti je definirano za kôd 205 (*Reset Content*). U ovom slučaju postojeći dokument se zadržava ali se i njegov sadržaj postavlja na početne vrijednosti (npr. kod višestrukog unosa podataka preko obrasca). Nažalost, stariji preglednici ne ispunjavaju ove dijelove specifikacije.

### 3.2.4 Preusmjeravanje na drugi URI

Poruka sa statusnim kôdom preusmjeravanja (3xx) sadrži novi URI u polju zaglavlja **Location**. Korisnički agent bi trebao (ponekad nakon dozvole korisnika, a ponekad automatski) podnijeti novi zahtjev prema tom URI-ju kako bi ispunio originalni zahtjev. Tijelo preusmjeravajućeg odgovora (ako postoji) obično sadrži samo poruku koja mu omogućuje da zahtjev podnese sam ukoliko automatsko preusmjeravanje zakaže.

Kod klasičnih Web poslužitelja statusni kôdovi preusmjeravanja se tipično koriste nakon premještanja resursa. Ovakav scenarij je spomenut pri raspravi o trajnosti URI-ja. Resurs se jedno vrijeme nalazi na URI-ju A. Drugdje na Webu i izvan njega stvore se veze prema tom resursu preko URI-ja A. Iz nekog razloga, resurs promjeni svoju adresu i sada je ona B. Međutim, stare veze još uvijek šalju zahtjeve prema A. Jedno od rješenja takvog problema je da se poslužitelj konfigurira da na zahtjeve prema A odgovara preusmjeravanjem na URI B.

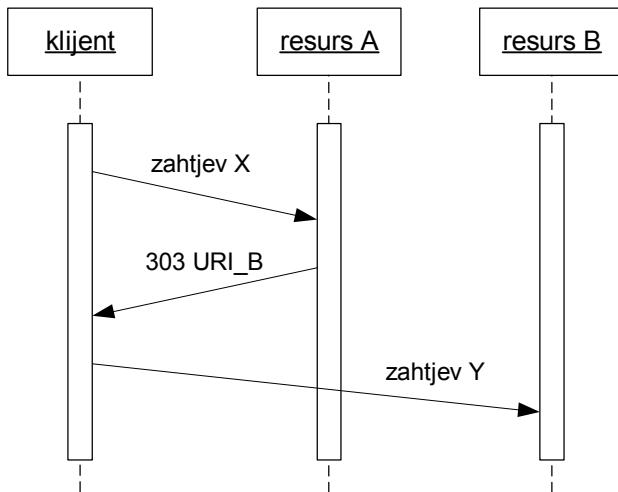
U navedenom scenariju pogodan je konkretni statusni kôd **301 (Moved Permanently)**. Takav odgovor podrazumijeva da je resurs trajno premješten i da bi sve veze trebale ubuduće biti usmjereni na novi URI. Korisnički agent bi trebao ponoviti identičan (osim ciljnog URI-ja, naravno) zahtjev nad novim URI-jem. Ukoliko je metoda nesigurna, korisnički agent bi prije preusmjeravanja trebao tražiti dozvolu korisnika.



Slika 3: Komunikacija pri odgovoru 301 (*Moved Permanently*) ili 307 (*Temporary Redirect*). Traženi resurs ne koristi više staru adresu i klijent je upućen da ponovi isti zahtjev nad novom adresom.

Ipak, umjesto statusnog kôda 301 u navedenom scenariju, kao i drugim njemu sličnim scenarijima, obično se koristi statusni kôd **302 (Found)**. Originalna namjena ovog kôda je bila da signalizira privremenu promjenu adrese resursa i stoga se veze prema staroj adresi ne bi trebale mijenjati. Inače, ovaj statusni kôd ima isto značenje kao i 301.

Sljedeći statusni kôd ove klase je **303 (See Other)**. U načelu, ovaj odgovor znači da resurs ne vraća entitet kao odgovor na zahtjev i da bi korisnički agent trebao postaviti novi GET zahtjev nad drugim resursom kao sljedeći korak u navigaciji. Budući da je GET metoda sigurna, preusmjeravanje se može obaviti automatski.



Slika 4: Komunikacija pri odgovoru 303 (*See Other*). Zahtjev X je uredno ispunjen, ali je klijent upućen da se za nastavak komunikacije obrati resursu B s novim GET zahtjevom.

Kôd 303 je vrlo različit od prije spomenutih 301 i 302, pogotovo ako promatramo slučaj kad je originalni zahtjev POST. Kôdovi 301 i 302 kažu da je resurs kojemu se podaci

trebaju poslati premješten, te da je taj zahtjev (POST) trebalo uputiti na novu adresu. Kôd 303 kaže da je željeni resurs nije premješten – on je ostao na staroj adresi i primio je podatke – ali i da će korisnički klijent entitet koji treba sada prikazati naći na drugoj adresi (koristeći GET). Prednost ovakvog preusmjeravanja biti će objašnjene kasnije.

Ključni problem s preusmjeravanjima u HTTP-u je što najvažniji preglednici (*Internet Explorer* i *Netscape Navigator*) ne poštuju standard. Osnovna greška je što se kôd 302 interpretira kao 303, tj. nad novom adresom se uvijek izvršava GET, a ne originalna metoda. HTTP/1.1 dokumentira taj *de facto* standard i uvodi dva nova kôda: već objašnjeni 303 i 307 (*Temporary Redirect*) koji je nova, neokaljana inačica 302.

Problem s "čistim" kôdovima 303 i 307 je što ih stariji klijenti ne prepoznaju. Najbolje što se može učiniti kod takvih slučajeva je odgovarajuća dodatna poruka.

### 3.2.5 Greške kod klijenta

Ova klasa poruka (4xx) je namijenjena za slučajeve kad se čini da je pogreška nastupila na klijentovoj strani. Odgovori trebaju sadržavati poruke namijenjene korisniku u kojem se opisuje situacija i nude rješenja.

Najčešća i najopćenitija poruka ove klase je kôd **404** (*Not Found*). Klijentu se želi naznačiti da resurs nije dostupan, ali se ne može ili ne želi ulaziti u razloge. Tipičan razlog je tipografska pogreška u URI-ju.

Još jedan čest i koristan kôd je **401** (*Unauthorized*). Njime se korisničkom agentu sugerira da je u zahtjevu potrebna valjana autorizacija kako bi se pristupilo želenom resursu (obično korisničko ime i zaporka). Kôd **405** (*Method Not Allowed*) služi za obaveštavanje klijenta da metoda zahtjeva nije primjenjiva na traženi resurs.

## 3.3 Korisnički doživljaj Weba

Osnova dobre Web arhitekture mora biti istinsko razumijevanje načina na koji korisnik doživljava Web i neki njegov pojedini dio. Taj doživljaj je zajednički učinak svih sjedišta na Webu, ali i tehničke podloge Weba, naročito Web preglednika.

### 3.3.1 Računarska korisnička sučelja

Korisničko sučelje (*user interface*, UI) je onaj dio sustava koji je namijenjen interakciji s korisnikom. U našoj raspravi ograničavamo se na programska (softverska) sučelja.

Povijesno najznačajniji oblici korisničkog sučelja i danas su najpopularniji. Stariji oblik se naziva **kriptičnim** sučeljem i sastoji se od upisivanja znakova u računalo i čitanje njegovog odgovora u obliku teksta, poput dijaloga. Primjer za takvo sučelje su tzv. *command prompt* sučelja DOS-a i Unix-a.

Šira primjena grafičkih mogućnosti računala dovela je do pojave **grafičkih korisničkih sučelja** (GUI-a). Jedno od "svetih pisama" GUI-a je "*Macintosh Human Interface Guidelines*" [app92], knjiga uputa za aplikacijska korisnička sučelja na Macintosh-u koje je izdao Apple. Za sučelje poput Macintosh-a često se koristi naziv WIMP (od ključnih naziva *windows, icons, menus and a pointing device*). U takvo sučelje spadaju i Windows sučelja, Motif i dr.

**Web** predstavlja novu platformu, ugrađenu u sustav WIMP<sup>4</sup>, ali ne sasvim ovisnu o njemu. Osnovne razlike su u uporabi metafore stranice umjesto prozora kao osnovnog gradivnog elementa sučelja i mreže umjesto osobnog računala kao objekta korištenja [per98].

### 3.3.2 Metafore Weba

Web je razmjerno nov, složen i važan sustav s vrlo širokom bazom korisnika, od kojih većina nije za to obučena niti ima posebno tehničko znanje. Zato obilato koristi **metafore** u svom nazivlju i funkciji.

Pojednostavljeno rečeno, metafora je preslikavanje nekog novog koncepta na sličan a korisniku poznatiji koncept. Ta metoda pomaže pri razumijevanju (pa samim tim i korištenju) novih i složenih sustava na razne načine:

- omogućava brže učenje o novom sustavu;
- pomaže kod prisjećanja već naučenih činjenica;
- omogućava bolje razumijevanje ponašanja novog sustava u nepredviđenim (nenaučenim) situacijama analogijom s poznatim konceptom.

Kako bi prednosti ove metode što više došle do izražaja, potrebno je odabrati metaforu koja je vjerna i sama po sebi poznata korisnicima. Ponekad je teško nov sustav vjerno opisati poznatim konceptom, jer je sam razlog zanimljivosti tog sustava njegova različitost od postojećih. Ponekad se različiti dijelovi sustava opisuju različitim

---

<sup>4</sup> Precizno govoreći, Web sučelje je bilo i još uvijek se koristi ugrađeno i u kriptične sustave (npr. Lynx).

metaforama. Na taj način se može govoriti da je sustav predočen složenom (*composite*) metaforom [smi96].

Originalna metafora Weba je vjerojatno bila **knjižnica** referentnih tekstova. Bibliografska utopija, jedinstvena svjetska mreža svega znanja skupljenog na jednom mjestu, skup suvislih dokumenata povezanih referencama ("za detalje vidi..."). Dokumenti su ispočetka bili tekstualni i odgovarali su datotekama trajno smještenim na diskovima poslužitelja. Međutim, nakon nekog vremena Web je prešao granice sveučilišta i izišao na ulicu, u poduzeća i domove, počeo se baviti politikom, zabavom, trgovinom i u tom procesu nadrastao svoju originalnu metaforu.

Suvremenija metafora je **časopis**. Promatran na taj način, Web je ogromno skladište časopisa, a pojedino sjedište predstavlja pojedini časopis. Ljudi listaju časopise, letimično pregledaju članke, skeniraju kroz tekst, preskaču s jednog na drugi članak, s jednog na drugi časopis, s vremena na vrijeme provjeravaju nova izdanja poznatih časopisa i slično.

Za razliku od znanstveno-stručnih publikacija koje na pojedinim stranicama nose samo osnovni sadržaj (tekst članka), Web stranice novijeg dizajna imaju i mnogo usputnog: reklame, okvire, nazive srodnih članaka, ankete i slično. Naslovica znanstvene periodike je obično vrlo jednostavna i uz naziv nerijetko sadrži samo popis članaka. Naslovica Web sjedišta više sliči naslovnicama popularne periodike sadrži veliku sliku ili više njih, nekoliko istaknutih naslova koje privlače pažnju, možda čak kraće tekstove – sam detaljan sadržaj ili indeks je sakriven. Pojam **obrasca** je u skladu s tiskovnim metaforama. Ljudi su se navikli na papirnate obrasce, čak i u časopisima: psihotestovi, narudžbenice...

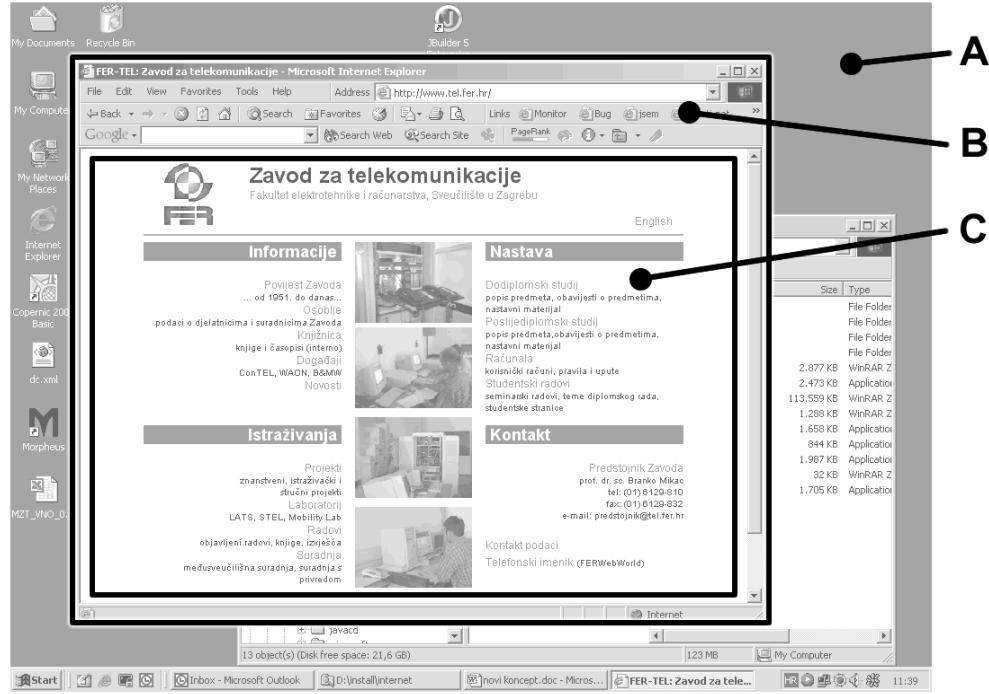
Možda je podjednako popularna i metafora **putovanja**. Web se zamišlja kao zaseban virtualni svijet kroz koji korisnik putuje.

Nazivi iz tiskovne metafore: stranica (*page*), objavlјivanje (*publishing*) stranice, *bookmark*, naslovna stranica, otvaranje (*open*) stranice. Iz metafore putovanja slijede nazivi poput sjedište (*site*), posjet (*visit*) stranici ili sjedištu, adresa (*address*), nazad (*back*), naprijed (*forward*) i navigacija (*navigation*).

Iz nazivlja i drugih elemenata sučelja možemo vidjeti kako je za Web izgrađena složena metafora sastavljena uglavnom od metafora tiska i putovanja, ali uključujući povremeno i manje važne metafore i originalne nazive.

### 3.3.3 Kontekst

Kod WIMP sučelja kontekst se prenosi otvaranjem prozora unutar postojećih prozora (Slika 5). Preglednik (B) je u kontekstu operativnog sustava (A). Web stranice (C) su u kontekstu preglednika. Unutar korisničkog prostora preglednika (gdje se prikazuju stranice), kontekst se mora generirati iznova.



Slika 5: Web stranica (C) se nalazi unutar preglednika (B), a on unutar GUI-a operativnog sustava (A).

Budući da (u pravilu) svaka nova stranica popunjava cijeli korisnički prostor preglednika, kontekst se unutar sjedišta prenosi ponavljanim dijelovima stranice. Ako je vizualno prenesen identitet šireg pojma, pojma koji predstavlja cijelo sjedište ili neka nadstranica, podstranica treba samo naglasiti razliku između drugih pojmovima u istom kontekstu.

### 3.3.4 Uporaba i značenje stranice

Korisnici doživljavaju stranicu kao mjesto na kojem mogu dobiti informacije prema ugledu stranice, a njen URI kao sredstvo pribavljanja stranice.

Osnovni problem pojavljuje se kad se nesigurni (obično POST) zahtjevi koriste za prikaz, koji se onda pogrešno povezuje s identitetom, kao što je pokazano sljedećim primjerom.

Zamislimo da resurs na URI-ju  $X$  prima narudžbu preko zahtjeva POST kao odgovor vraća kôd 200 (*OK*) i hipermedijski entitet sa sadržajem narudžbe kao potvrdu. Korisnik vidi stranicu kao informaciju o svojoj narudžbi i URI  $X$  kao sredstvo pribavljanja te informacije.  $X$  se može zapisati, spremiti među omiljene veze (*bookmarks*, *favorites*) ili

na drugi način sačuvati dok ne poželi ponovo vidjeti sadržaj svoje narudžbe i ponoviti zahtjev resursu na adresi  $X$ . Problem je što je takav ugled pogrešan, i resurs na adresi  $X$  ustvari prima novu narudžbu, a ne pruža podatke o staroj.

Rješenje je da resurs na adresi  $X$  ne bude Web stranica, nego funkcija koja po uspješnom okončanju kao odgovor vraća kôd 303 (*See Other*) s adresom  $Y$  stranice koja treba prikazati informaciju o novoj narudžbi. Korisnik na ovaj način ne bi ni primijetio prijelaz preko resursa na adresi  $X$ , dok resurs na adresi  $Y$  može utvrditi ugled koji odgovara njegovom identitetu.

## 3.4 Navigacija

Navigacija je umijeće prelaska s jedne na drugu Web stranicu aktiviranjem postojećih hiperveza na tim stranicama. Arhitekt sjedišta pomaže svojim korisnicima pri navigaciji osmišljavajući stranice od kojih se sjedište sastoji. Prema Nielsenu [nie00], svaka stranica bi trebala korisniku ponuditi odgovor na tri pitanja:

- "Gdje sam?" – identitet i položaj trenutne stranice;
- "Gdje sam bio?" – informacije o dotadašnjoj navigaciji;
- "Kamo mogu ići?" – trenutne navigacijske mogućnosti.

### 3.4.1 Gdje sam?

Korisno je ovo pitanje razdvojiti na dva komplementarna aspekta: predložavanje **identiteta** ("Što ova stranica poslužuje?") i **položaja** ("Koji je odnos ove stranica i drugih stranica na Webu?"). O prikazivanju identiteta govorili smo u prethodnom poglavljju, pa ćemo se usredotočiti na problematiku prikazivanja položaja na Web stranici.

S obzirom na postojanje dvaju stožernih okvira konteksta, pojednostavljeni možemo reći da se pitanje dalje rastavlja na dva potpitanja: "Na kojem se Web sjedištu nalazim?" (prepoznavanje identiteta i položaja sjedišta) i "Gdje sam unutar ustroja tog sjedišta?". Ovo posljednje pitanje dovodi do središnjeg pojma ustroja sjedišta. Očito da pitanje implicira postojanje ustroja sjedišta koji se sastoji od međusobno povezanih komponenti te postojanje načina da se jednostavno prikaže lokacija komponente unutar tog ustroja.

**Karta sjedišta** (*site map*) je Web stranica čija je osnovna namjena vizualizacija ustroja sjedišta na kojem se nalazi, pogotovo prikaz položaja. Ako je sjedište preveliko da bi se predstavili svi resursi na jednoj stranici, karta prikazuje samo najvažnije dijelove, odnosno ulazi u dubinu ustroja samo do određene razine.

### 3.4.2 Gdje sam bio?

Informaciju o protekloj navigaciji obično daje sam preglednik (npr. tipkom "Back" i promjenom boje na vezama prema već posjećenim resursima). Ponekad, Web aplikacija sama može dati dodatne mogućnosti kontrole sesije: npr., popularno Web sjedište "amazon.com" ima vrlo uspјelu uslugu "*The page you made*" koja prikazuje pregledane artikle i srodne informacije.

### 3.4.3 Kamo mogući?

Svaka veza se ne mora koristiti za navigaciju. Osim veza između samostalno prikazanih resursa, postoje i veze na umetnute resurse, npr. veza iz HTML dokumenta prema umetnutoj slici. Zbog toga ćemo koristiti uži pojam **navigacijske veze**. Po uzoru na Nielsena [nie00] razlučio bih ukupno tri vrste navigacijskih veza:

- **ustrojne (structural)** veze koje povezuju ustrojno bliske resurse unutar sjedišta (ponekad i između sjedišta, ako postoji odgovarajući ustroj u skupu tih sjedišta);
- **asocijacijske** veze, kojim se identitet stranice veže uz identitet nekog ustrojno nevezanog drugog resursa, obično zbog srodnosti pojmove koje predstavljaju;
- **sadržajne veze**<sup>5</sup>, ugrađene u sadržajni sastojak stranice, pokazuju na resurse bliske tematiki sadržaja, ne nužno vezane uz identitet stranice.

Kao primjer, zamislimo stranicu koja predstavlja forum za raspravu o programiranju u Javi. Ona se nalazi na sjedištu **forums.tld** koje poslužuje mnoge forme o različitim temama. URI stranice je: <http://forums.tld/computers/java>. Stranica je sastavljena od navigacijske ploče, nekih tematskih veza i poruka koje čine raspravu, a koje mogu same sadržavati veze na proizvoljne Web resurse (Tablica 1).

URI	opis veze	vrsta veze
<a href="http://forums.tld/">http://forums.tld/</a>	naslovna stranica sjedišta	ustrojna
<a href="http://forums.tld/computers/java/servlet">/computers/java/servlet</a>	rasprava specijalizirana za tehnologiju servleta	ustrojna
<a href="http://forums.tld/entertainment">/entertainment</a>	dio sjedišta s forumima o zabavi	ustrojna
<a href="http://java.sun.com">http://java.sun.com</a>	Sun-ovo sjedište o Javi	asocijacijska
<a href="http://java.tld/forum">http://java.tld/forum</a>	drugi forum o Javi	asocijacijska
<a href="http://news.tld/19423">http://news.tld/19423</a>	vijest o temi spomenutoj u raspravi	sadržajna
<a href="http://people.tld/mark">http://people.tld/mark</a>	osobna stranica autora jedne poruke	sadržajna

Tablica 1: Primjeri vrsta navigacijskih veza na zamišljenoj stranici.

Dok ustrojne veze gotovo u potpunosti ovise o ustroju sjedišta, sadržajnim vezama je praktički nemoguće upravljati iz stanovišta arhitekture sjedišta jer ih unose ljudi koji

<sup>5</sup> Odgovarajući pojam za Nielsena su "embedded links", veze umetnute u tekst stranice [nie00]

mijenjaju sadržaj sjedišta. Slobodno povezivanje je potrebno barem na nekoj razini, jer jedino tako Web dolazi do punog potencijala. Asocijacijske veze su po stupnju upravlјivosti negdje između.

## 3.5 Ustroj Web sjedišta

Riječ "ustroj" ("structure") u kontekstu Web sjedišta se koristi na vrlo različite načine, i često se izbjegava definiranje tog pojma. Obično se govori o "ustroju informacijskog prostora" [nie00], što je dosta općenit pojam. Konkretniji smisao može biti konceptualni ustroj podataka koji čine sadržaj sjedišta (kao kod [cer00]). Takav ustroj može biti prikazan npr. dijagramom entitet-odnos (*Entity-Relationship Diagram – ERD*) ili UML-om (*Unified Modeling Language*) [umlrr].

Prema načelu odvajanja prilagodnog od predmetnog sustava, trebalo bi razlikovati ustroj sjedišta od ustroja predmetnog sustava. Iako je prvi u mnogome zavisан o drugom, ipak koristi bitno drugačije koncepte.

Definicija ustroja Web sjedišta korištena u ovom radu je:

**Ustroj** Web sjedišta je skupna informacija o **identitetu, identifikatorima, položaju i sastavu** svakog **resursa** u sjedištu.

U nastavku su pojašnjene navedene četiri vrste informacija o resursima sjedišta.

### 3.5.1 Identitet resursa

Identitet resursa je pojam koji resurs predstavlja (vidi poglavlje 2.3).

### 3.5.2 Identifikator resursa

Budući da je za identifikaciju resursa u sjedištu odabran standard URI, problem modeliranja identifikatora resursa postaje problem dizajniranja URI-ja u potprostoru URI-ja sjedišta.

Model identifikacije resursa pruža važno sučelje prema sjedištu, koje koriste i ljudi i strojevi. Važno je da se to sučelje ne naruši, tj. da se ne naruši trajnost uglednih URI-ja. Bilo bi najbolje kad se URI-sučelje ne bi sužavalо za vrijeme javnog korištenja, nego samo po potrebi proširivalо.

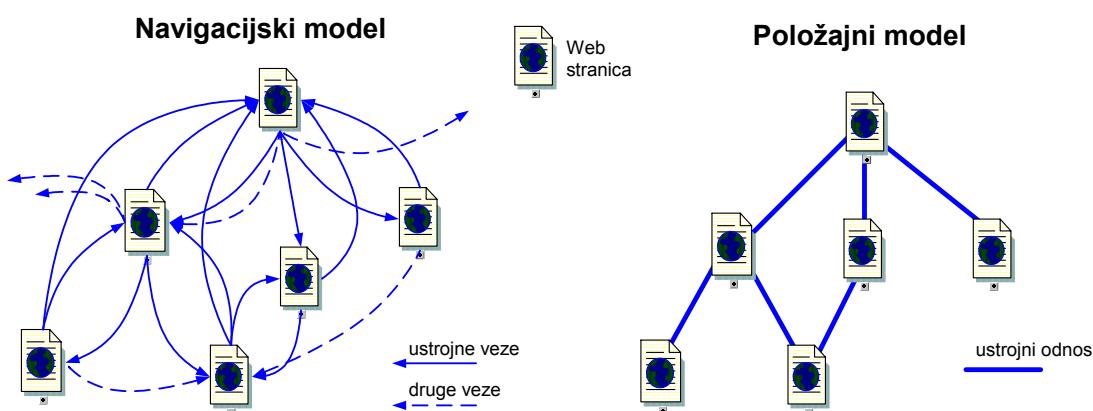
Ako model ustroja iz tehnoloških razloga nameće znatna ograničenja na plan identifikacije resursa, trajnost tih URI-ja će se ograničiti na trajnost tehnologije, što je

gotovo uvijek prekratko. Resursi često mijenjaju URI-je s, na primjer "/cgi-bin/odjel.pl?carape" na "/odjel.asp?ime=carape", samo da bi ih dogodine opet promjenili, a Web će biti prisutan i dulje od te godine.

### 3.5.3 Položaj resursa

Položaj resursa na sjedištu je određen međuodnosima s drugim resursima u sjedištu. Najvažniji su međuodnosti stranica.

Ustrojne veze (poglavlje 3.4.3) u navigacijskom modelu odražavaju ustrojne odnose među stranicama. Mnogostrukе veze u navigacijskom modelu mogu se prikazati s nekolicinom ustrojnih odnosa (Slika 6).



Slika 6: Navigacijski i položajni model istog sjedišta nacrtani usporedo. Iz svake stranice postoji ustrojna veza prema naslovnicima, svim neposredno nadređenim i podređenim stranicama.

### 3.5.4 Sastav resursa

Resurs može biti sastavljen od više zasebnih, samodostatnih logičnih cjelina. One čine detaljniji, finiji uvid u resurs, jer navode identitet dijelova sadržaja resursa. Te pojedine logične cjeline možemo zvati **komponentama** resursa, a njihov smještaj unutar nekog resursa kompozicijom ili **sastavom** tog resursa.

Sastav i identitet resursa nisu u čvrstoj vezi, štoviše resurs može razmjerno često mijenjati svoj sastav kako bi se prilagodio korisniku i vremenu, zadržavajući isti identitet.

Najvažniji primjer sastava resursa je sastav Web stranice. Stranica je nezavisni blok korisničkog sučelja i sadrži komponente koje se isporučuju odjednom. Odgovor stranice se šalje kao entitet u HTTP-ovom odgovoru, dok se njena komponenta ne može tako samostalno isporučiti. Gledajući na taj način, stranica nije ništa drugo nego spremnik za svoje komponente.

*Treće poglavlje govori o temeljima Weba, definirajući osnovne pojmove (stranicu i sjedište), tehničku podlogu (protokol HTTP), korisnički i autorski aspekt Web sjedišta, s posebnim naglaskom na navigaciju i ustroj. U sljedećim poglavljima se više govori o ustroju sjedišta, počevši od pregleda nekih odabranih modela.*

# 4 Pregled modela ustroja Web sjedišta

Ovo poglavlje predstavlja postojeće modele ustroja Web sjedišta. Uz daleko najpopularniji, takožvani klasični model ustroja temeljen na datotečnom sustavu, predstavljena su i četiri suvremenija modela: Structured Graph Format (SGF), WebML, SiteBrain i iAS, svaki od njih na svoj način karakterističan i zanimljiv, ali u praksi mnogo manje korišten.

## 4.1 Klasični model ustroja

Osnova klasičnog modela ustroja sjedišta je datotečni sustav (engleski *file system*). Web i URI-ji su nastali na proširenju pojmove iz datotečnog sustava, što dodatno opravdava naziv "klasični".

### 4.1.1 Datotečni sustav

Najpopularnije porodice datotečnih sustava danas su Unix i DOS/Windows. I one i druge popularne vrste datotečnih sustava temelje se na tri temeljna načela [nie96]:

- postoji samo jedna vrsta spremišta za podatke: **datoteke** (*files*);
- datoteke su postavljene u strogu hijerarhiju **direktorija** (*directories* ili *folders*);
- identifikacija datoteka i direktorija se vrši isključivo putem njihovih **imena** (*name*) i položaja u hijerarhiji.

Svaki od tih načela dovodi do klasičnih problema vezanih uz sve datotečne sustave:

- ograničenje jedne univerzalne vrste spremišta dovodi do jednoslojnog predstavljanja sadržaja, za razliku od višeslojnog (npr. razdvajanje sadržaja i izgleda);
- ograničenje jedne hijerarhije direktorija onemogućuje višestruke (složene) klasifikacije;
- identifikacija putem jedne tekstualne vrijednosti (imena) dovodi do problema osiromašenja i gubljenja konteksta.

### 4.1.2 Web poslužitelji temeljeni na datotečnom sustavu

Web poslužitelji temeljeni na datotečnom sustavu koriste datoteke kao resurse. Postoje dva osnovna načina korištenja datoteke kao resursa: pasivni i aktivni.

**Pasivni** resursi su datoteke čiji sadržaj poslužitelj doslovno prenosi kao entitet odgovora onako kako je zapisan na mediju za pohranu. Preslikavanje je klasično "jedan-na-jedan": pasivni resurs se poistovjećuje s datotekom. Pasivni resursi ne moraju nužno biti nepromjenjivi (čak niti statični) ako se sadržaj datoteke (često) mijenja. Promjena se obično radi ručno, ali postoje i sustavi koji automatski generiraju i/ili ažuriraju sadržaj tih datoteka. U tom slučaju se datoteke koriste kao svojevrstan međuspremnik objavljenih podataka.

**Aktivni** resursi sadrže informacije o načinu generiranja entiteta za odgovor. To su obično izvršne ili skriptne datoteke, koje se pokreću na HTTP-ovski zahtjev, a njihov izlaz (obično tekst formatiran kao HTML) se vraća u HTTP-ovom odgovoru kao traženi entitet. Ovo načelo je primijenjeno kod skoro svih klasičnih platformi za izvedbu Web aplikacija, od CGI-a, preko ASP-a, *servlet*-a, JSP-a itd. Takva datoteka može predstavljati klasu resursa, tako da put URI-ja predstavlja izvršnu datoteku, dok je upit informacija koja se prenosi toj datoteci.

**Direktoriji** se koriste za grupiranje resursa, na isti način kao što su korišteni za grupiranje datoteka. U Web poslužitelju odredi se neki direktorij kao temeljni i on postaje korijen hijerarhije Web sjedišta.

#### **4.1.3 Uporaba datotečnog sustava pri ustroju sjedišta**

Budući da je Web originalno modeliran prema datotečnom sustavu, preslikavanje na podprostor URI-ja je prirodno: put iz URI-ja se preslikava na jednaki put datotečnog sustava (određen imenima direktorija). Put do direktorija se obično preslikava na datoteku predviđenog imena u tom direktoriju (npr. "`index.html`" ili "`default.asp`").

Ako put identificira aktivni resurs, sadržaj upita mu se obično prenosi kao jedan parametar ili popis parametara.

Temeljni direktorij odgovara korijenu Web sjedišta. Naprednije tehnike preslikavanja podprostora URI-ja uključuju dodavanje tzv. "virtualnih direktorija", kada se Web put preslika na neki drugi put koji se ne nalazi nužno u temeljnog direktoriju.

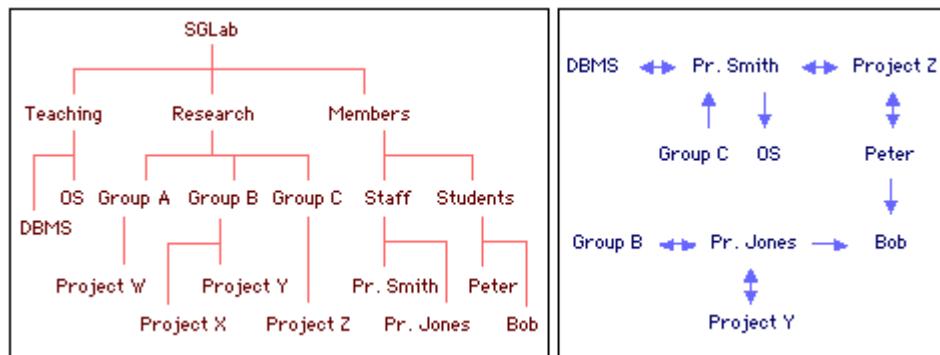
Preusmjeravanja (vidi poglavje 3.2.4) se izvode podešavanjem poslužitelja: obično se doslovno unose početni i ciljni URI-ji, ponekad i vrsta preusmjeravanja.

## 4.2 Structured Graph Format

Structured Graph Format (SGF) je XML gramatika za opisivanje ustroja Web sjedišta [lie98].

### 4.2.1 Opis

SGF razlikuje dvije osnovne vrste veza među čvorovima grafa: **hijerarhijske** i **asocijativne** (Slika 7). Hijerarhijske veze ostvaruju agregatne relacije i razapinju stablo između svih čvorova. Asocijativne veze su slobodne usmjerene veze između bilo koja dva čvora, i same tvore mrežu. Asocijativne veze se zatim dijele na dvije vrste: **eksplicitne** i **implicitne**. Dok su eksplizitne zadate, implicitne veze se podrazumijevaju između dvaju čvorova  $p$  i  $q$  ako postoji bar jedna eksplizitna asocijativna veza među čvorovima  $s$  i  $d$ , koji su podređeni čvoru  $p$  odnosno  $q$ .



Slika 7: Primjer hijerarhijske (lijevo) i mrežne (desno) organizacije prema SGF-u [lie98].

### 4.2.2 XML zapis

XML zapis SGF-a (Slika 8) se sastoji od tri dijela: čvorovi, hijerarhijske i asocijativne veze

Element **NODES** sadrži popis čvorova i njihovih atributa. Osnovni atributi su nametnuti identifikator, naziv čvora i URL, zadan kao proizvoljni atribut u posebnom elementu **SGATT**.

Popisi hijerarhijskih odnosno asocijativnih veza nalaze se u elementima **HIERARCHY** i **NETWORK**. Navode se samo identifikatori ishodišnog i odredišnog čvora.

```
<!DOCTYPE STRUCTUREDGRAPH SYSTEM "SGF.dtd">
<STRUCTUREDGRAPH>

<NODES>
  <NODE NODEID="N001" LABEL="SGLab_Home_Page">
    <SGATT NAME="URL" VALUE="http://www.sglab.edu" />
  </NODE>
  <NODE NODEID="N002" LABEL="Teaching">
    <SGATT NAME="URL" VALUE="http://www.sglab.edu/teaching.html"/>
  </NODE>
```

```

<NODE NODEID="N003" LABEL="Research">
    <SGATT NAME="URL" VALUE="http://www.sglab.edu/research.html"/>
</NODE>
...
<NODE NODEID="N016" LABEL="Project_Z">
    <SGATT NAME="URL"
        VALUE="http://www.sglab.edu/projects/project_z.html"
    />
</NODE>
<NODE NODEID="N019" LABEL="Peter">
    <SGATT NAME="URL" VALUE="http://www.sglab.edu/~peter" />
</NODE>
</NODES>

<HIERARCHY>
    <LINK SOURCE="N001" DEST="N002"> </LINK>
...
</HIERARCHY>

<NETWORK>
    <LINK SOURCE="N019" DEST="N020"> </LINK>
...
</NETWORK>

</STRUCTUREGRAPH>

```

Slika 8: Djelomični prikaz ustroja sjedišta u SGF-u [lie98].

## 4.3 WebML – *Web modeling language*

WebML [cer00] [bon00] [ros01] je jezik za opisivanje ključnih značajki sjedišta na višoj razini. Ostvaren je kao XML gramatika vezana uz intuitivno grafičko predstavljanje.

### 4.3.1 Perspektive

Specifikacija sjedišta u WebML-u sastoji se od četiri međusobno ortogonalne perspektive:

- **ustrojnog** (*structural*) modela, koji opisuje konceptualni model podataka predmetnog sustava;
- **hipertekstualnog** modela, koji opisuje složaj stranica u sjedištu;
- **prezentacijskog** modela, koji opisuje izgled stranica u sjedištu;
- **personalizacijskog** modela, koji služi prilagođavanju dijelova sjedišta pojedinim korisnicima ili grupama korisnika.

Ove četiri perspektive pokrivaju sve dijelove opisa sjedišta. Jasan je rastav na sadržaj (čiji se kostur daje u tzv. ustrojnom modelu), izgled (prezentacijski model) i ustroj sjedišta koji je, kombiniran s navigacijskim značajkama, sadržan u hipertekstualnom modelu. Stoga ćemo posljednji model pogledati pobliže.

### 4.3.2 Hipertekstualni model

Ovaj model se u WebML-u dijeli na dva zasebna podmodela: kompozicijski model i navigacijski model.

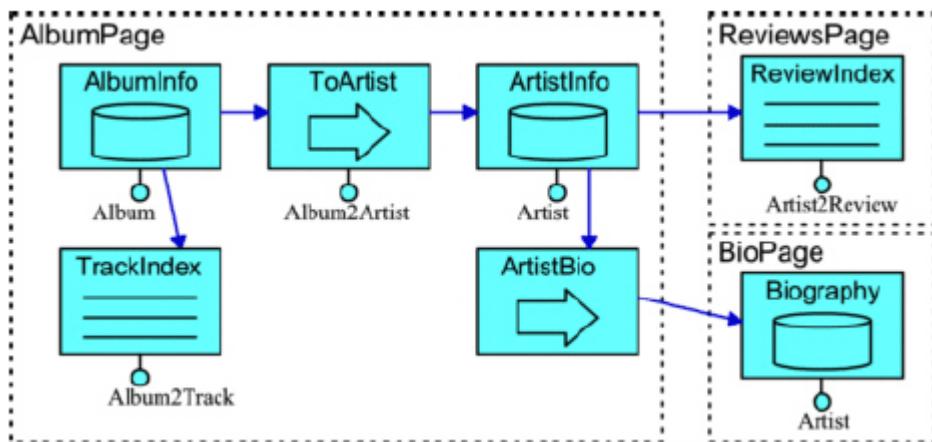
**Kompozicijski** (*composition*) model navodi koje se komponente koriste u sjedištu. WebML-ov naziv je "**sadržajna jedinica**" (kraće "jedinica") koje se onda slažu u stranice. WebML definira šest vrsta takvih jedinica:

- **podatkovna** jedinica – prikazuje detaljne podatke o jednom objektu;
- **višepodatkovna** (*multidata*) jedinica – detaljni podaci o skupu objekata;
- **indeksna** jedinica – popis objekata;
- **skrolna** jedinica – prikaz kontrola za pristupanje uređenom skupu objekata;
- **jedinica filtriranja** (*filter unit*) – prikazuje obrazac za unos vrijednosti kojima se može filtrirati skup objekata;
- **usmjerivačka** jedinica (*direct unit*) – ne prikazuje podatke, nego označava semantički međuodnos dvaju objekata.

**Navigacijski** model određuje na koji se način jedinice i stranice povezuju u hipermedijski sustav. Zato služe usmjerene **veze**, koje se dijele na:

- **kontekstne** veze – povezuju jedinice tako da se iz ishodišne jedinice prenese informacija (kontekst) do odredišne jedinice, kako bi se utvrdilo koji se objekt ili skup objekata prikazuje;
- **nekontekstne** veze – slobodno povezuje stranice, neovisno o njihovom sadržaju ili semantici.

#### 4.3.3 Primjer WebML-a



Slika 9: Primjer WebML dijagrama

Primjer dijagrama (Slika 9) prikazuje koncept tri stranice (označene isprekidanim linijama):

- **AlbumPage** – prikaz zadanog albuma; prikazuje informacije o tom albumu (**AlbumInfo**), popis pjesama na albumu (**TrackIndex**) i informacije o izvođaču (**ArtistInfo**);
- **ReviewsPage** – prikazuje samo popis recenzija za zadanog izvođača;
- **BioPage** – prikaz biografije za zadanog izvođača.

Vidimo da sličica ilustrira vrstu sadržajne jedinice (**AlbumInfo**, **ArtistInfo** i **Biography** su podatkovne jedinice, a tu su i po dvije indeksne i usmjerivačke jedinice). Veze (usmjerene linije) su sve kontekstualne i prenose potreban kontekst ciljnoj jedinici. Usmjerivačke jedinice pretvaraju informaciju o jednoj vrsti entiteta na informaciju o drugoj vrsti. Na primjer, **ToArtist** prenosi kontekst iz informacije o albumu na informaciju o izvođaču tog albuma.

Važno je primjetiti da ove tri stranice nisu samodostatne, jer nije dovedena kontekstualna informacija u jedinicu **AlbumInfo**.

#### 4.4 SiteBrain

SiteBrain je proizvod temeljen na tzv. "The Brain" modelu grafa, izvorno zamišljenog kao alata za naprednu organizaciju osobnih bilješki, datoteka i drugih resursa.

#### 4.4.1 Model "The Brain"

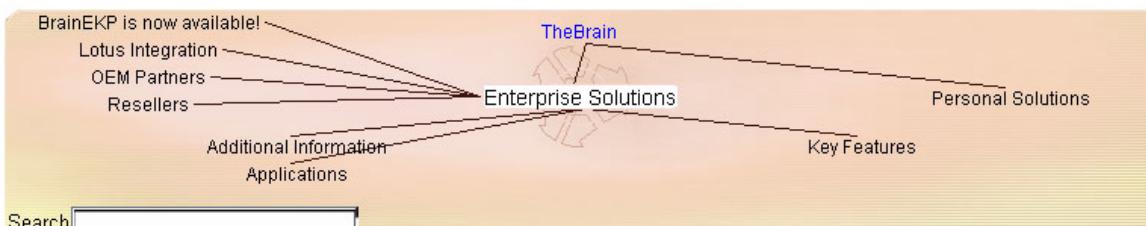
U modelu grafa "The Brain" cijela mreža se naziva "mozgom" (*brain*), a jedan čvor u mreži "misao" (*thought*). Postoje dvije vrste veza koje se mogu uspostaviti između misli:

- usmjereni veza koja predstavlja odnos roditelj-dijete (*parent-child*);
- neusmjereni veza koja povezuje slične misli (*jump link*).

Model sliči SGF-u, s tim da je veza roditelj-dijete izvedena tako da ne stvara nužno hijerarhiju, jer se njima može zatvoriti petlja. Svaka misao može imati više "roditelja" i "djece".

#### 4.4.2 Primjena pri navigaciji

Namjena SiteBrain-a je da takvim modelom prikaže ustroj sjedišta. Odgovarajući *applet* u Javi se pokrene u svom okviru (*frame*), obično na vrhu stranice i omogućuje korisniku pregled ustroja sjedišta. Pogled na trenutno aktivnu misao i njene izravne "susjede" se zove "*plex*" (Slika 10). Dok korisnik navigira "mozgom", šalje se HTTP-ovski zahtjev za prikazivanje Web stranice vezane uz tu misao u drugom okviru u pregledniku.



Slika 10: *Applet* koji prikazuje jedan *plex* u SiteBrain-u na sjedištu [www.thebrain.com](http://www.thebrain.com). Vidimo trenutno aktivnu misao u sredini, jednu roditeljsku misao iznad, troje misli-djece ispod, četiri bliske misli lijevo i jednu "sestru" desno.

#### 4.4.3 Navigacijske kvalitete

Osnovne prednosti SiteBrain-a su:

- predstavlja jednoobrazan navigacijski sustav;
- ne treba učitati stranicu kako bi se prikazao njen položaj;
- ima ugrađeno brzo pretraživanje po nazivima misli.

S druge strane, nedostaci SiteBrain-a su:

- korištenje okvira na stranici koji ruše metaforu stranice; Web dizajneri ih često izbjegavaju;

- koristi Java *applet*, što predstavlja veliki zahtjev na klijenta i u praksi često dovodi do tehničkih problema;

U vrijeme pisanja ovog rada čini se da je tvrtka The Brain Technologies prekinula razvoj SiteBrain-a (iako ga još koristi, npr. na svom vlastitom sjedištu), ali je nastavila razvoj izvornog modela u proizvodima PersonalBrain i BrainEKP.

## 4.5 Oracle Internet Application Server

Oracle-ov *Internet Application Server* (iAS) [ora9ias] predstavlja sloj aplikacijskog poslužitelja velike arhitekture koja se temelji na Oracle-ovoj relacijskoj bazi. Oracle 9 iAS uključuje podršku za PL/SQL Web toolkit, servlete, JSP i mnoge druge Web tehnologije.

Karakterističnost iAS-a je što se resursi čuvaju kao objekti relacijske baze podataka, slično kao što se kod klasičnog modela resurs čuva u datotečnom sustavu kao datoteka. Taj objekt je u pravilu procedura (tzv. *stored procedure*) koja vraća entitet (obično HTML tekst). Dijelovi iAS-ovog paketa, poput **Portala**, sami generiraju te procedure (a po potrebi i druge objekte baze) na osnovu vlastitih modela.

### 4.5.1 Oracle Portal

Oracle Portal je jedan od proizvoda iz iAS-ovog paketa namijenjen lakom kreiranju Web sjedišta povezanih s bazom podataka, naročito korporativnih intranetskih portala. U osnovi se sastoji od upravljanja sadržajem (*content management*), izrade korisnički prilagodljivih stranica i izrade Web aplikacija nad objektima baze podataka.

Portal smješta sadržaj u tzv. **sadržajna područja** (*Content Area*). Sadržajno područje je organizirano preko jedinstvenog stabla direktorija (*folder-a*), kojem je pridruženo još jedno stablo kategorija i dodatna višestruka klasifikacijama po tzv. perspektivama.

Stranice se izgrađuju od unaprijed definiranih komponenti koje prikazuju sadržaj iz sadržajnog područja ili su same komponente aplikacije (obrasci, izvještaji i sl.). Nažalost, ne postoji nikakva organizacija stranica usporediva s organizacijom sadržajnog područja.

### 4.5.2 Portleti

Portal komponente stranice naziva **portletima**. iAS sam dolazi s velikom zbirkom upotrebljivih portleta, ali nudi i *Portlet Development Kit* (PDK) za jednostavan razvoj korisnički prilagođenih portleta [orapc]. Podržan je razvoj u Javi i PL/SQL-u.

Predviđeni su uobičajeni načini rada (*mode*) za sve portlete. Postoji ukupno sedam različitih načina rada:

- *Shared Screen Mode* – izgled portleta kad dijeli stranicu s drugim komponentama;
- *Full Screen Mode* – izgled portleta kao jedine komponente na stranici;
- *Preview Mode* – prikaz kako bi izgledao portlet kad ga se primjeni;
- *Edit Mode, Edit Defaults Mode* – izgled obrazaca za izmjenu parametara portleta;
- *About Mode, Help Mode* – informacije o portletu korisniku.

## 4.6 Usporedba i ocjena odabralih modela

Nakon pojedinačnog opisa svakog od modela postaviti ćemo ispred svakog nekoliko kriterija koji jamče da model odgovara teoriji ustroja sjedišta. Za početak valja reći da nijedan od prikazanih modela nije izvorno zamišljen kao model ustroja sjedišta u tako širokom smislu kao što je definirano u ovom radu, pa stoga ne čudi što nijedan nije zadovoljio sve kriterije. Postoje i druge metodologije koje bi mogle biti usporedive, ali nisu ušle u ovaj odabir (npr. OOHMD [ros99], W3DT [bic96]...)

Glavni kriteriji su zasnovani na definiciji ustroja. Osnovni preduvjet je da model obuhvaća sve resurse u sjedištu. Za svaki od njih treba zatim opisati četiri svojstva: identitet, identifikator, položaj i sastav. Ako model prepoznaže komponente resursa, onda ulazi u obzir i opisivanje identiteta svake komponente. Na kraju, ocjenjuje se modelova povezanost sa samim poslužiteljem i podržavanje višejezičnosti.

### 4.6.1 Sveobuhvatnost

Sveobuhvatnost je svojstvo modela da opisuje svaki resurs koji je dio sjedišta.

Većina promatranih modela (SGF, WebML i SiteBrain) ograničava se samo na stranice. Ostali prikazni resursi (slike, pasivni dokumenti) prepuštaju se obično izvedbi prema klasičnom modelu.

Klasični model čuva sve resurse u datotečnom sustavu na jedinstven način i stoga ima jednak pristup svim resursima: pasivnim dokumentima, skriptama, programima, slikama i svim ostalima. Takav pristup nužno garantira sveobuhvatnost.

iAS sadrži mogućnost čuvanja svih resursa u bazi, putem posebnih rješenja interMedia (za pohranu multimedijalnih sadržaja) i iFS (*Internet File System*, izvedba napredne vrste

datotečnog sustava u bazi). Međutim, u praksi se uglavnom koristi smještanje u usporedni klasični model.

#### **4.6.2 Model identifikacije resursa**

Iz opisa odabralih modela očito je zanemarivanje važnosti dizajna identifikatora – URI-ja. SGF, WebML i SiteBrain primaju proizvoljne URI-je za svaku stranicu koju opisuju.

Klasični model obično veže URI-je uz ustroj direktorija, tako da često put ili dio puta URI-ja odgovara datotečnom putu. iAS iz URI-ja očitava naziv procedure koju treba pokrenuti i njene parametre, što često dovodi do za ljudi sasvim neprepoznatljivih i neupotrebljivih URI-ja.

#### **4.6.3 Model identiteta resursa**

Nijedan od predstavljenih modela nema rješenje problema zapisivanja ili izgrađivanja identiteta određenog resursa.

#### **4.6.4 Model položaja resursa**

Položaj je najpopularnije svojstvo ustroja među ispitivanim modelima. Većina modela kombinira dva pristupa:

1. uspostavljanje stroge hijerarhije (stabla);
2. nadilaženje hijerarhije dodavanjem posebnih veza ili dodatnih hijerarhija.

Dok se klasični model i SGF ograničavaju na jedno stablo, iAS-ov model organizacije sadržajnog područja ima najsloženiji i najmoćniji sustav višestruke klasifikacije i višestrukog povezivanja. Činjenica što takav model nije primjenjiv na resurse izvan sadržajnog područja diskvalificira taj model kao model ustroja Web sjedišta.

#### **4.6.5 Model sastava resursa**

Neki oblik podjeli resursa na samostalne elemente, komponente, vidljiv je samo u dva slučaja: sadržajne jedince u WebML-u i portleti u Oracle-ovom Portal-u. Sastav je kod oba modela predviđen samo za stranice.

Razlike između ova dva načina uporabe komponenti su zнатне. WebML koristi zatvoren skup teoretski definiranih komponenti i usredotočuje se na identifikaciju i prenošenje kontekstne informacije s jedne komponente na drugu. Portleti, s druge strane, čine otvoreni skup s naglaskom na platformu za razvoj.

Pravilna uporaba načina rada stvarno ističe portlete kao samostalne komponente u stranici. Nažalost, međusobno povezivanje portleta, naročito pri izgradnji njihovog identiteta je zanemareno. S druge strane, WebML ima predviđeno povezivanje komponenti teorijski važnim kontekstnim vezama.

#### **4.6.6 Povezanost s poslužiteljem**

Klasični model i iAS-ov model su izravno izvedeni u Web poslužitelju: klasični model s poslužiteljem zasnovanim na datotečnom sustavu i Oracle iAS s poslužiteljem zasnovanim na procedurama smještenim u bazi.

Drugi modeli su praktički nepovezani s poslužiteljem. WebML predviđa povezivanje s poslužiteljem putem generiranja Web aplikacije u obliku ASP skripti. Takvo jednosmjerno i asinkrono povezivanje očito dovodi do nesigurne logičke veze između zamišljenog (konceptualnog) i izvedbenog sustava.

#### **4.6.7 Višejezičnost**

Nijedan od opisanih modela ne podržavaju višejezičnost sami po sebi. Uobičajeni način rješavanja problema višejezičnih sjedišta svodi se na izgradnju više sličnih jednojezičnih sjedišta koji se trebaju održavati manje-više zasebno. Ovakav pristup povećava nepotrebnu redundanciju, čineći održavanje skupljim i manje pouzdanim.

#### 4.6.8 Pregled

	klasični model	SGF	WebML	SiteBrain	Oracle iAS
obuhvaća sve resurse?	✓	✗	✗	✗	✓ (teorijski)
modelira identifikatore resursa?	✓ djelomično put	✗	✗	✗	✗
identitet resursa?	✗	✗	✗	✗	✗
položaj resursa?	✓ jedno stablo	✓ stablo i asocijativne veze	✗	✓ mreža s dvije vrste veza	✓ samo unutar sadržajnog područja
sastav resursa?	✗	✗	✓ zatvoren skup komponenti	✗	✓ samo stranice, otvoren skup portleta
neposredno povezan s poslužiteljem?	✓	✗	✗	✗	✓
podržava višejezičnost?	✗	✗	✗	✗	✗

legenda: ✓ - ispunjeno ili djelomično ispunjeno; ✗ - nije ispunjeno

Tablica 2: Pregledna usporedba opisanih modela ustroja.

Već opisane ocjene svakog modela prema kriterijima prikazane su sada na pregledan način (Tablica 2). Promatrajući ove modele usporedno, možemo prepoznati tri skupine:

- **Poslužiteljski** modeli, poput klasičnog i iAS-a, neposredno su povezani s poslužiteljem – sve promjene u modelu bi se trebale odmah odraziti na rad poslužitelja. Neposredna posljedica je barem potencijalna sveobuhvatnost resursa na poslužitelju.
- **Položajni** modeli koji svoju funkciju svode uglavnom na prikaz položajnog modela stranica u sjedištu. Očiti primjeri su SGF i SiteBrain. Klasični model se djelomično uklapa u ovu skupinu sa svojom jednostavnom organizacijom resursa u hijerarhiju direktorija.
- **Sastavni** modeli usredotočeni su na komponente stranice i njihovu funkciju. Takav je WebML i (djelomično) iAS. WebML kao potpuniji sastavni model ima izravan način prenošenja kontekstne informacije o identitetu komponente. iAS s druge strane ima općenitiju platformu za izradu komponenti.

Poslužiteljski modeli pokušavaju proširiti postojeće razvojne tehničke platforme na Web. Klasični model je zasnovan na izvornom poimanju Weba: jedinstven sustav za lagan pristup lokalno spremljenim dokumentima preko mreže. Stoga on proširuje već postojeći

datotečni sustav omogućivši mu jednu vrstu mrežnog pristupa. iAS koristi isto načelo, ali bitno drugačiju platformu – procedure smještene u Oracle-ovoj bazi.

Opisani položajni modeli krenuli su drugim putem. Već postojeći model, SGF odnosno The Brain, koji nisu bile izvorno u vezi s Webom, preuređeni su kao model ustroja Web sjedišta. Taj način se uglavnom pokazuje slabo primjenjivim i previše ograničenim za sve mogućnosti i popularne prakse sadašnjih rješenja Web sjedišta.

Treći pristup (sastavni modeli) izbjegava pogrešku pokušavanja nametanja postojećeg modela na ustroj sjedišta i prepoznaje koncepte i uzorke svojstvene za teoriju i praksu Weba. WebML prihvata takav pristup i uspijeva u mnogim svojim ciljevima, ali je ograničen na model komponenti stranica i ne pokriva druge aspekte ustroja sjedišta.

*Nakon pregleda i usporedbe nekih odabranih postojećih modela ustroja, definira se novi model nazvan "UriGraph" s ciljem zadovoljenja svih aspekata definicije ustroja sjedišta.*

# 5 Model UriGraph

*Na osnovu definicija ustroja iz prethodnih poglavlja, u ovom poglavljju predlaže se novi model ustroja Web sjedišta, nazvan "UriGraph". Izvorno osmišljen kao model za opisivanje identifikatora resursa (konkretno, URI-ja) u sjedištu, ovakav model se pokazao korisnim za opis ostalih triju svojstava ustroja: identiteta, položaja i sastava svih resursa. Takru univerzalnu primjenu omogućuju tehnike dobrog dizajniranja potprostora URI-ja, koji se onda može koristiti u sve potrebne svrhe. Na kraju poglavљa UriGraph je ocijenjen kao model ustroja sjedišta i uspoređen s prethodno opisanim drugim modelima ustroja.*

## 5.1 Načela UriGraph-a

Za razliku od WebML-a, koji počinje od sastava stranice i posebnu pažnju posvećuje uređenju i povezivanju komponenti, zanemarujući stranicu kao cjelinu, model UriGraph počinje od modela identifikacije resursa (uključujući i sve stranice).

### 5.1.1 Model identifikacije resursa

UriGraph predstavlja prvenstveno model **identifikacije** resursa, konkretno model složaja URI-ja unutar sjedišta. Dizajn potprostora URI-ja čini osnovno sučelje prema svim Web resursima. Kako je potrebno da dijelovi URI-ja svakog resursa vjerno prenose njegov identitet, slijedi da se iz modela identifikacije resursa može prikupiti potrebna informacija i o **identitetu** svakog resursa.

Budući da je poželjno da URI odražava i položaj resursa na sjedištu, prirodno je da isti odnosi među resursima postavljeni preko identifikatora posluže kao **polozajni** model. Kroz takav model resursi nižeg položaja mogu naslijediti neka svojstva od resursa na višem položaju, poput komponenti, čineći model **sastava** resursa u sjedištu.

Slijedeći ovaku logiku, možemo zaključiti da iz modela identifikacije resursa klijaju ostala tri svojstva ustroja. UriGraph istražuje tu ovisnost, nudeći kompaktni, ali potpuni model ustroja.

### 5.1.2 Ulaz i izlaz modela

Ulaz u ustrojni model se naziva ustrojnim **zahtjevom**. Zahtjev treba sadržavati samo identifikator resursa u određenom kontekstu (tj. informaciju o sjedištu).

Izlaz iz modela je ustrojni **odgovor**. U njemu treba biti sadržan opis identiteta pronađenog resursa, kao i informacije o njegovom položaju i sastavu. Iznimno, odgovor može biti poruka o neregularnom ispunjenju zahtjeva (nema takvog resursa).

### 5.1.3 Slojevi modela

U cilju smanjenja složenosti, model ustroja je u ovom radu razložen na tri funkcionalna sloja: topološki sloj, sloj analize zahtjeva i sloj sinteze odgovora.

**Topološki sloj** je temeljni sloj koji određuje sve čvorove u grafu i njihove povezanosti.

Nad topološkim slojem je izgrađen **sloj analize zahtjeva**. On definira uvjete pri kojima može doći do izvršenja prijelaza, očitava informaciju iz segmenata URI-ja i zatim uklanja očitane segmente.

Usporedno s analizom zahtjeva provodi se i **sinteza odgovora** – informacije koje se očitaju pri analizi mogu se ugraditi u odgovor u obliku natuknica, zajedno s komponentama odgovora koje mogu biti ugrađene u mesta grafa.

## 5.2 Topološki sloj

Prvi, temeljni sloj grafa ustroja tiče se samo popisivanja čvorova i veza medu čvorovima, bez funkcionalnosti obrade samog zahtjeva. Slijedi matematička definicija i prijedlog grafičkog prikaza topološkog sloja.

### 5.2.1 Matematička definicija

Osnova topološkog sloja modela je usmjereni graf (tzv. digraf). Definira se preko dva skupa čvorova, jednog istaknutog čvora i jedne relacije.

- Postoji skup **čvorova**  $N$ , partitioniran na dva konačna skupa: neprazan skup  $P$  (koji se naziva skupom **mesta - places**) i skup  $T$  (skup **prijelaza - transitions**).

$$N = P \cup T$$

$$P \neq \emptyset$$

$$P \cap T = \emptyset$$

- Jedan element skupa  $P$  je posebno istaknut i naziva se **korijenom** grafa.

Označavamo ga s  $r$ :

$$r \in P$$

3. Postoji binarna relacija  $E$  nad skupom  $N$ . Ta relacija predstavlja svih **skup usmjerenih grana** između čvorova, tako da u svakom paru  $(a, b)$   $a$  predstavlja ishodišni a  $b$  ciljni čvor:

$$E \subseteq (N \times N)$$

4. Graf je ograničen tako da ne postoje povratne grane u ishodišni čvor, tzv. petlje.

To znači da je relacija  $E$  potpuno nerefleksivna, tj. ne sadrži parove  $(x, x)$  za nijedan  $x$  iz  $N$ :

$$\forall x \in N : (x, x) \notin E$$

5. Relacija  $E$  ne sadrži parove dvaju mesta – iz mesta mogu biti grane samo prema prijelazima:

$$\forall p_1, p_2 \in P : (p_1, p_2) \notin E$$

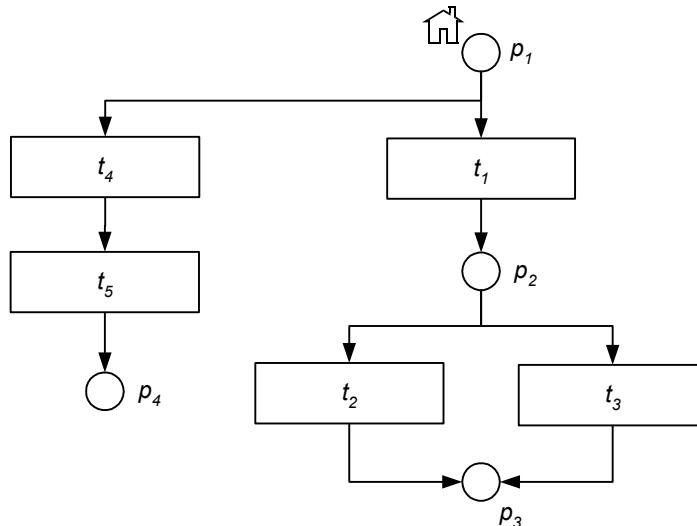
6. Iz svakog prijelaza postoji najviše jedna grana koja ide u neko mjesto.

$$\forall t \in T; \forall p_1, p_2 \in P : ((t, p_1) \in E \wedge (t, p_2) \in E) \Rightarrow p_1 = p_2$$

Uređena četvorka  $(P, T, r, E)$  koja udovoljava svim navedenim ograničenjima se onda može zvati **grafom ustroja po UriGraph-u**.

### 5.2.2 Grafički prikaz

Prema općem grafičkom prikazu, svako mjesto je prikazano krugom a prijelaz širokim pravokutnikom. Čvorovi su međusobno povezani strelicama koje označuju grane. Korijenski čvor je označen simbolom kućice (prema engleskom izrazu *home page*).



Slika 11: Primjer topološkog sloja grafa ustroja s označenim čvorovima

Na primjeru (Slika 11) vidimo graf sa sljedećim karakterističnim vrijednostima:

$$\begin{aligned}
P &= \{p_1, p_2, p_3, p_4\} \\
T &= \{t_1, t_2, t_3, t_4, t_5\} \\
r &= p_1 \\
E &= \{(p_1, t_1), (p_1, t_4), (t_1, p_2), (p_2, t_2), (p_2, t_3), (t_2, p_3), (t_3, p_3), (t_4, t_5), (t_5, p_4)\}
\end{aligned}$$

### 5.2.3 Prolaz kroz graf

Definirati ćemo **šetnju** (*walk*) kroz graf ustroja  $G = (P, T, r, E)$  kao  $z$ -torku ( $z > 0$ ) čvorova iz  $N(n_1, n_2, \dots, n_z)$ , takvu da za svaka dva susjedna čvora u popisu postoji grana u grafu koja ta dva čvora povezuje u istom redoslijedu:

$$\forall k \in \{1, 2, \dots, z-1\}: (n_k, n_{k+1}) \in E$$

Posebna vrsta šetnje je **prolaz** (*passage*) kroz graf ustroja. Prolaz je svaka šetnja koja udovoljava sljedećim ograničenjima:

1. Prvi čvor u prolazu je uvijek korijen grafa.

$$n_1 = r$$

2. Posljednji čvor u prolazu je uvijek mjesto.

$$n_z \in P$$

Na primjeru grafa (Slika 11) možemo uočiti ukupno pet mogućih prolaza:  $(p_1)$ ,  $(p_1, t_1, p_2)$ ,  $(p_1, t_1, p_2, t_2, p_3)$ ,  $(p_1, t_1, p_2, t_3, p_3)$  i  $(p_1, t_4, t_5, p_4)$ .

U slučaju da graf sadrži čvor koji nije dio nijednog mogućeg prolaza kroz graf, takav čvor se naziva **nefunkcionalnim**.

## 5.3 Sloj analize zahtjeva

Svrha analize zahtjeva je očitavanje njegovih dijelova konstruiranjem prolaza kroz graf. Tijekom prolaza zahtjev se smanjuje izbacivanjem po jednog segmenta prolaskom kroz svaki prijelaz. Kada se zahtjev posve isprazni, smatra se da je zahtjev ispunjen i nema ga potrebe više obrađivati. Budući da to čini trenutni čvor posljednjim u prolazu, on bi trebao biti mjesto.

### 5.3.1 Sastav zahtjeva

Zahtjev prenosi identifikator resursa preuzet iz svog okruženja, u našem slučaju HTTP-ovog URI-ja. Budući da je identifikacija sjedišta izbačena, konačan zahtjev se sastoji od dva dijela:

- puta  $P$ , popisa segmenata puta;

- upita  $Q$ , popisa segmenata upita.

Zahtjev  $R$  je, dakle, uređeni par  $(P, Q)$ .

Segment puta je niz znakova određen pravilima segmenata puta HTTP-ovog URI-ja.

Segment upita je nešto složeniji. Svaki segment upita ima naziv, a može imati i vrijednost. I naziv i vrijednost mogu biti prazni.

Zahtjev je **prazan** kad mu liste i puta i upita ne sadrže segmente.

### 5.3.2 Obrada zahtjeva u čvoru

Za svaki čvor  $n$  u grafu ustroja  $G$  definiran je skup njegovih **odredišnih čvorova**  $D_n$ , tj. svih čvorova  $m$  za koje postoji grana  $(n, m)$ . Čvor  $n$  se u odnosu na taj skup naziva **ishodišnim čvorom**.

$$D_n = \{m \in N \mid (n, m) \in E\}$$

**Obrada** zahtjeva  $R$  za ishodišni čvor  $n$  je naziv koji znači utvrđivanje skupa odredišnih čvorova  $C_n$  (podskup od  $D_n$ ). Prolaz se može konstruirati tako da se počne obradivati korijen grafa i u prolaz se dodaje svaki sljedeći obrađeni čvor. Pri svakoj obradi razlikujemo tri slučaja:

1.  $C_n$  sadrži točno jedan čvor,  $m$ . Taj čvor treba biti sljedeći čvor u prolazu i on se dalje obrađuje.
2.  $C_n$  je prazan skup. Ne postoji čvor kojim bi se nastavio prolaz – kaže se da je zahtjev **neobradiv**.
3.  $C_n$  sadrži više od jednog čvora. Nastavak prolaza nije jednoznačno određen – kaže se da je zahtjev **višeznačan**.

### 5.3.3 Odabir odredišnih čvorova

Pri obradi zahtjeva u čvoru, osnovna zadaća je odabir odredišnih čvorova. Ako je neki odredišni čvor mjesto, automatski je odabran (po definiciji grafa ustroja postoji najviše jedan takav odredišni čvor). Svaki odredišni čvor koji je prijelaz je odabran ako i samo ako se ispitivanjem prijelaza utvrdi njegova **prohodnost**. Ispitivanje prohodnosti prijelaza je operacija koja može biti posebno definirana za svaki prijelaz.

Utvrdjivanjem prohodnosti prijelaza odabire se i točno jedan segment iz zahtjeva koji je potreban za **izvršenje** prijelaza. Ako se prohodni prijelaz tijekom obrade odabere za

nastavak šetnje, taj segment će se izbaciti iz zahtjeva, a informacija koju je nosio se može ugraditi u odgovor pri sintezi, kao što je opisano kasnije.

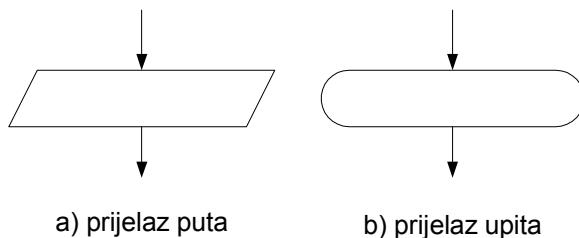
### 5.3.4 Vrste prijelaza

Razlikujemo dvije osnovne vrste prijelaza:

- **prijelaz puta**, koji predstavlja očitavanje jednog segmenta puta;
- **prijelaz upita**, koji očitava segment upita.

Osim po vrsti segmenta koje ispituju, ova dva prijelaza se razlikuju i po redoslijedu ispitivanja. Prijelaz puta ispituje prvi neobrađeni segment puta, dok prijelaz upita ispituje sve segmente upita dok ne nađe na prvi koji odgovara. Stoga se pri izvršenju prijelaza puta uvijek izbacuje prvi segment iz puta, dok se kod prijelaza upita može izbaciti bilo koji (ali točno jedan) segment iz upita zahtjeva.

U grafičkom prikazu, posebne vrste prijelaza se crtaju drugačije. Prijelaz puta se crta kao paralelogram (kose stranice podsjećaju na znak kose crte "/") a prijelaz upita kao oval (oble stranice podsjećaju na obične zagrade koje ograju par ime-vrijednost) (Slika 12).



Slika 12: Simbol prijelaza u grafu ustroja.

### 5.3.5 Propusnice kroz prijelaze

Ispitivanjem prijelaza se utvrđuje njegova prohodnost. Ispitivanje prijelaza se svodi na ispitivanje njegove **propusnice (pass)**.

Postoje osnovna podjela propusnica u UriGraph-u na **elementarne i složene** propusnice. Složena propusnica sadržava više drugih propusnica i na osnovu ispitivanja njih donosi odluku o vlastitoj propusnosti. Elementarna propusnica se ne može dalje razložiti na druge propusnlice.

Uobičajeni primjeri složenih propusnica su **konjuktivna** propusnica, koja propušta samo ako svaka od sadržanih propusnica propušta zahtjev (logička operacija "I"); te **disjunktivna** propusnica, koja propušta ako bilo koja od sadržanih propusnica propušta (logička operacija "ILI").

Ispitivanje propusnosti ovisi o vrsti propusnice (kako je definirana), o segmentu koji se ispituje, kao i dosadašnjoj analizi (tzv. kontekstu analize).

### 5.3.6 Prvenstva grana

U poglavlju 5.3.2 koje opisuje obradivanje čvora u postupku konstruiranja prolaza kroz graf za zadani zahtjev, kao jedan od mogućih slučajeva naveden je problem višezačnosti u određivanju sljedećeg čvora u prolazu. Ta višezačnost se može u većini slučajeva riješiti uvođenjem pravila prvenstva (prioriteta) između izlaznih grana.

Formalno rečeno, nad skupom izlaznih grana iz ishodišnog čvora definirana je **relacija prvenstva** " $\geq$ ", sa svojstvima refleksivnosti, tranzitivnosti i usporedivosti. Relaciju tumačimo na način da " $x \geq y$ " znači "grana  $x$  je višeg ili istog prvenstva od grane  $y$ ". Na osnovu te relacije možemo definirati i sljedeće relacije:

1. relacija **jednakosti po prvenstvu**:

$$x \approx y \Leftrightarrow x \geq y \wedge y \geq x$$

2. relacija **pravog prvenstva**:

$$x > y \Leftrightarrow x \geq y \wedge \neg(x \approx y)$$

Obrada čvora dovodi do odabira skupa odredišnih čvorova  $C_n$  nad kojim je definirana ista relacija prvenstva. Ishodišni čvor se sada obrađuje tako da se odabire neprazan podskup  $C_n$  odredišnih čvorova koji imaju najviše prvenstvo u  $C_n$ . Ako je skup  $C_n$  jednočlan, obrada je uspjela.

Pri ovakvoj obradi analiza se još uvijek može blokirati, i to u sljedeća dva slučaja:

- skup  $C_n$  još uvijek može biti prazan (ne postoji  $C_n$ ) što dovodi do neobradivosti zahtjeva;
- $C_n$  još može biti višečlan, što dovodi do višezačnosti zahtjeva.

U grafičkom prikazu, oznaka prvenstva grane se zapisuje uz strelicu na kraju grane, tako da je vidno nedvosmisleno koju granu označava.

### 5.3.7 Model prvenstva HNL

Modelu UriGraph je u ovom radu pridružen tzv. **model HNL** pisanja oznaka koje određuju prvenstva grana. Taj model je zamišljen sa sljedećim zahtjevima na umu:

1. postojanje podrazumijevanog (*default*) prvenstva koje se ne navodi;

2. lako proširivanje oznake prvotno zadano prvenstva na proizvoljno više ili niže prvenstvo, iz čega slijedi da je skup prvenstava otvoren (nema apsolutno najmanjeg i najvećeg prvenstva);
3. gust skup svih prvenstava – tako da između svaka dva različita prvenstva po tom modelu postoji prvenstvo manje od jednog, a veće od drugog.

Prvo, definiran je tročlani skup simbola  $\mathbf{S} = \{\mathbf{H}, \mathbf{N}, \mathbf{L}\}$ . U skupu je definirana relacija pravog prvenstva  $\mathbf{S} \times \mathbf{S} " > "$  tako da vrijedi  $\mathbf{H} > \mathbf{N} > \mathbf{L}$ . Iz te relacije se može uočiti da  $\mathbf{H}$  predstavlja visoko (*high*),  $\mathbf{N}$  srednje (*normal*) a  $\mathbf{L}$  nisko (*low*) prvenstvo.

Model HNL čini skup oznaka s relacijama pravog prvenstva i jednakosti po prvenstvu. Oznaka prvenstva je uređena  $n$ -torka simbola:  $p = (a_1, a_2, a_3, \dots, a_n)$ . Za svaku oznaku  $p$  definirana je i **indeksna funkcija**  $s_p$ . Ona vraća simbol na  $i$ -tom mjestu oznake prvenstva, odnosno simbol srednjeg prvenstva  $\mathbf{N}$  za sve indekse veće od najvišeg definiranog indeksa ( $n$ ).

$$s_p : \mathbf{N} \rightarrow \mathbf{S}$$

$$s_p(i) = \begin{cases} a_i & \text{za } i \leq n \\ \mathbf{N} & \text{za } i > n \end{cases}$$

Sada možemo definirati odnose među oznakama prvenstva. Dvije oznake  $p$  i  $q$  su **iste** (i jednake po prvenstvu) ako imaju iste indeksne funkcije:

$$p = q \Leftrightarrow \forall i \in \mathbf{N} : s_p(i) = s_q(i)$$

Ako su oznake različite, očito postoji razlika u nekom indeksu. Najmanji indeks za koji postoji razlika u indeksnim funkcijama označimo s  $k$ :

$$p \neq q \Rightarrow \exists k \in \mathbf{N} : \forall i \in \mathbf{N} : s_p(i) \neq s_q(i) \Rightarrow i \geq k$$

Konačno, relacija pravog prvenstva među oznakama prvenstva se može definirati kao odgovarajuća relacija prvenstva između prvih različitih simbola:

$$p > q \Leftrightarrow s_p(k) > s_q(k)$$

Ovakva relacija prvenstva među oznakama se preslikava na skup grana tako da je svakoj grani pridružena točno jedna oznaka. Ukoliko oznaka nije izrijekom navedena, podrazumijeva se oznaka normalnog prvenstva (  $\mathbf{N}$  ).

Ukupno gledajući, zahtjevi su zadovoljeni na sljedeći način:

1. postoji podrazumijevano prvenstvo (  $\mathbf{N}$  );

2. svako zadano prvenstvo  $p$  može se proširiti simbolom **L** za niže, odnosno **H** za više prvenstvo;
3. za dva zadana različita prvenstva  $p > q$ , može se konstruirati prvenstvo  $r$  sa svojstvom  $p > r > q$  tako da se npr.  $p$  proširi simbolom **L**, ili  $q$  simbolom **H** iza posljednjeg mesta na kojem se  $p$  i  $q$  razlikuju.

### 5.3.8 Iznimni slučajevi pri analizi

Već smo naveli da u tijeku analize zahtjeva može doći do više iznimnih slučajeva. Ukratko ćemo ih ovdje navesti:

- **nepotpun** zahtjev – zahtjev je ispraznjen između prijelaza, tj. analiza nije završila u mjestu;
- **neobradiv** zahtjev – nema odredišnog čvora s kojim se može nastaviti prolaz;
- **višezačan** zahtjev – postoji više od jednog prolaza koja bi odgovarala zadanim zahtjevima.

Ovi iznimni slučajevi mogu se razriješiti na dva načina:

- izvan sustava modela ustroja, tako da odgovarajući modul primi iznimku;
- izmjenom modela ustroja, tako da se unesu nove grane nižeg prvenstva s prohodnjim prijelazima putem kojih se može sintetizirati odgovarajući odgovor (obično kôd 404 u nekom kontekstu).

## 5.4 Sloj sinteze odgovora

Sloj sinteze odgovora, kao najviši sloj modela UriGraph predstavlja ugradnju informacije proistekle iz analize zahtjeva, stanja poslužiteljskog sustava i samog ustroja u odgovor modela. Odgovor se sastoji od identiteta i sastava resursa (ako je resurs složen od više komponenti).

### 5.4.1 Identitet resursa

Identitet je skup **natuknica** (*clues*), elementarnih informacija o identitetu resursa, koji odgovaraju općenitim atributima pojma koji taj resurs predstavlja. Pojedine natuknice se dodaju u odgovor uz potvrđne propusnice u prijelazima grafa. Formalno rečeno, ako analiza zahtjeva konstruira prolaz  $\Pi$ , točno one propusnice koje se nalaze u prolazima iz  $\Pi$  će dodati svoje natuknice u odgovor. U slučaju da se u prijelazu nalazi složena

propusnica, preuzimaju se natuknice iz svih propusnica koje se pri ispitivanju prijelaza pokažu prohodnjima.

Natuknica se može stvoriti iz informacije koju nosi segment puta ili upita koji je bio izbačen u tom prijelazu. Na taj način informacije iz zahtjeva prelaze u odgovor.

### 5.4.2 Sastav resursa

Sastav resursa je skup komponenti. Komponente koje će odgovor sadržavati se također nalaze u čvorovima konstruirane šetnje, ovaj put u mjestima. Oblik komponente ne ovisi o analiziranom zahtjevu, ali može imati svojstva **lokalnosti** (djelovanja u čvoru u kojem je smještena) i **nasljednosti** (djelovanja u čvorovima nakon onog u kojem je smještena). Lokalna komponenta se dodaje u odgovor samo ako je njen čvor posljednji u šetnji, dok se nasljedna komponenta dodaje ako njen čvor nije posljednji u šetnji. Komponenta može biti istovremeno i lokalna i nasljedna.

Formalno rečeno: neka je  $\Pi$  konstruirani prolaz ( $n_1, n_2, \dots, n_z$ ), a  $\Lambda_n$  odnosno  $\mathbf{H}_n$  skup lokalnih odnosno nasljednih komponenti u čvoru  $n$  ( $\Lambda_n$  i  $\mathbf{H}_n$  su prazni skupovi za prijelaze, a u mjestima nisu nužno disjunktni). Tada je  $C$ , skup komponenti koje sadrži odgovarajući odgovor, definiran kao unija svih nasljednih komponenti čvorova koji nisu posljednji u šetnji i lokalnih komponenti posljednjeg čvora u šetnji.

$$C = \left( \bigcup_{i=1}^{z-1} \mathbf{H}_{n_i} \right) \cup \Lambda_{n_z}$$

Iznimno, komponenta može biti niti lokalna niti nasljedna. Budući da takva komponenta ne može biti nikada uključena u odgovor, takav odabir ima ulogu (privremenog) isključenja komponente iz korištenja, za razliku od brisanja i gubitka informacije o toj komponenti.

### 5.4.3 Natuknice o identitetu komponente

Kao što se utvrđuje identitet samog resursa, svaka pojedina komponenta tog resursa ima svoj vlastiti identitet. Na primjer, naslovница sjedišta o filmskim novostima može imati popis trenutno najpopularnijih filmova kao svoju komponentu. Informacija o vrsti komponente ("popis najpopularnijih filmova") očito je najvažnija pri određivanju kako se komponenta treba prikazati, ali se u modelu mogu zadati i neki dodatni parametri ("točno pet filmova iz proteklog tjedna").

Identitet resursa nužno ulazi u identitet komponente, budući da se komponenta nalazi u kontekstu resursa. Drugim riječima, skup natuknica koji čini identitet resursa je podskup

svim skupovima natuknica koji čine identitet svake komponente tog resursa. Na primjer, stranica specijalizirana za trilere može filtrirati samo taj žanr među filmovima u opisanoj komponenti.

Lokalnost ili nasljednost komponente nužno se odražava i na njegove natuknice o identitetu.

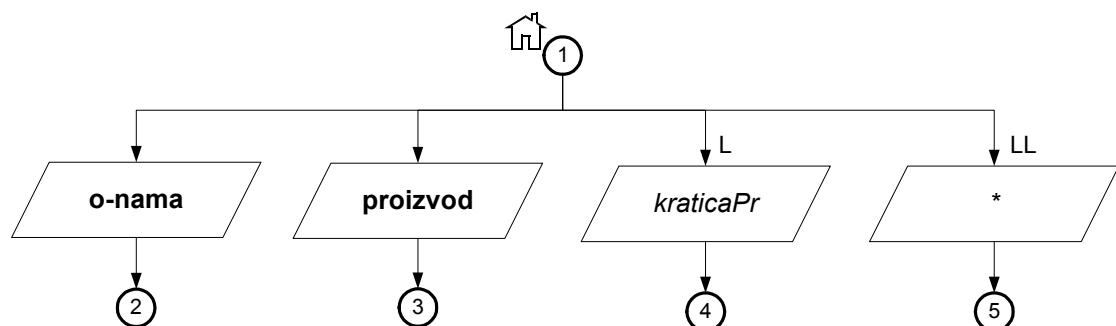
## 5.5 Uzorci u grafu

Opisati ćemo nekoliko jednostavnih a čestih uzoraka koji se pojavljuju u grafovima modeliranim po UriGraph-u. Kroz njihov opis pokazati ćemo i jednostavne primjere korištenja UriGraph-a u modeliranju ustroja sjedišta.

### 5.5.1 Granajući prijelazi

Uobičajeni način stvaranja hijerarhije u grafu ustroja je grananje više različitih prijelaza koji se povezuju na različita mjesta.

Čvorove iz primjera (Slika 13) možemo podijeliti u tri skupine.



Slika 13: Primjer za uzorak granajućih prijelaza.

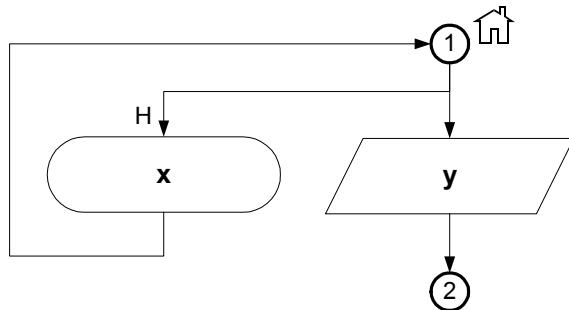
Prva skupina su dvije glavne grane ustroja sjedišta, koje počinju s prijelazima "o-nama" i "proizvod" i nastavljaju se dalje u mrežu čvorova, što na slici nije prikazano. U realnom slučaju ovakvih grana bi bilo više od dvije.

Drugu skupinu čini klasa resursa predstavljena mjestom 4, a to su ustvari preusmjeravanja na primarne URI-je nekih popularnih URI-ja. Uzmimo na primjer da je Ledov proizvod King (identificiran s "<ledo.hr/proizvod/sladoled/king>") popularan toliko da poduzeće želi kraći URI koji lakše može propagirati. Na ovaj način može se uvesti kratica koja će zahtjeve na URI "<ledo.hr/king>" preusmjerivati na primaran, dulji URI. Ovaj prijelaz vezan je s nižim prvenstvom, kako se ne bi dogodilo da nepomišljeno uvedena kratica zasjeni glavne grane ustroja.

Krajnji desni prijelaz propušta bilo kakav segment puta (simbol zvjezdice). Može se iskoristiti da presretne prvi segment puta u zahtjevu koji bi inače vratio iznimku.

### 5.5.2 Povratni prijelaz

Povratni prijelaz je uzorak u grafu kod kojeg se prijelaz spaja ponovno na mjesto od kojeg je spojen.

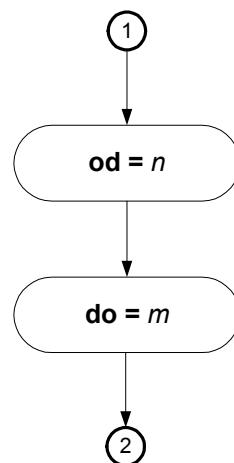


Slika 14: Primjer grafa s povratnim prijelazom.

Iz primjera (Slika 14) se vidi da svaki zahtjev može, ali ne mora imati segment upita "x". Regularni su prolazi "/y", "/y?x" i "/y?x&x", između ostalih. U ovom primjeru je nužno je da povratni prijelaz ima ulaznu granu višeg prvenstva, jer bi inače zahtjev "/y?x" bio više značan.

### 5.5.3 Uzastopni prijelazi

Izravnim spajanjem dva prijelaza tako da ishodišni prijelaz nije spojen na neko mjesto stvara se uzorak uzastopnih prijelaza. Redoslijed nizanja prijelaza je važan za prijelaze puta, ali ne i prijelaze upita.



Slika 15: Primjer uzastopnih prijelaza.

Primjer (Slika 15) pokazuje prijem parametara putem dva segmenta upita koja uvijek moraju dolaziti u paru. Oba kao vrijednost segmenta primaju cijeli broj ( $n$  i  $m$ ). Ispravni zahtjevi su "`?od=1&do=3`" i "`?do=0&od=100`" ali ne i "`?od=1`" ili "`?do=3`".

Uzastopni prijelazi se navode kao obavezni uvjeti za prelazak iz jednog mesta u drugo, dok povratni prijelazi predstavljaju neobavezne uvjete.

## 5.6 Ocjena UriGraph-a

Ponoviti ćemo ocjenu modela ustroja prema kriterijima ustanovljenim u 4.6, ovaj put nad UriGraph-om.

### 5.6.1 Sveobuhvatnost

Sveobuhvatnost je očito ispunjena. Budući da je UriGraph ugrađen u poslužitelj, a postojanje resursa se doživjava na razini protokola, slijedi da ako svako posluživanje zahtjeva prolazi kroz UriGraph, on mora sadržavati svaki resurs. Zbog toga postaje važno da UriGraph može modelirati svaku praktično potrebnu vrstu resursa.

UriGraph je pogodan za modeliranje stranica, izvršnih i drugih resursa. Budući da je u osnovama UriGraph-a resurs definiran vrlo općenito, teorijski ga je moguće primijeniti na bilo kakvu vrstu resursa.

### 5.6.2 Identifikacija

UriGraph je prvenstveno model analize valjanih URI-ja za Web sjedište, pa je zbog toga kvaliteta modela identifikacije osnovni kriterij po kojem bi se trebao ocjenjivati. Očito je da je UriGraph-ov način modeliranja URI-ja vrlo općenit. To je prednost kad se želi na jedinstven način oblikovati različite moguće uzorke, ali može biti nedostatak ako se ograničava na slaganje modela po nekoliko jednostavnih načela (npr. stroga hijerarhija). Ovaj put je odabran svjesno zbog mišljenja autora da su načini ustroja sjedišta nedovoljno istraženi i da bi se to istraživanje trebalo potaknuti razvojem moćnijih alata. Kad (odnosno ako) se pokaže da su neki načini i uzorci u ustroju znatno bolji od ostalih, ili čak dovoljni za slaganje po volji složenih sjedišta, UriGraph se može prepraviti tako da mu se upgrade ograničenja koja će olakšati korištenje.

### 5.6.3 Identitet

Identitet resursa se utvrđuje tako da se iz zahtjeva (URI-ja) očitavaju dijelovi korisnih informacija i prevode u prikladan način. UriGraph-ov način je skup Javinih objekata s određenim svojstvima. Spremnik u obliku skupa je važan da prikrije različit redoslijed

kojim su natuknice prikupljane. Klase natuknica moraju biti pomno oblikovane tako da bude jasno pod kojim uvjetima su dvije natuknice iste, a kad su različite (koristeći Javinu metodu `equals`). Udovoljavanjem tih tehničkih zahtjeva postiže se usklađenost između intuitivnog doživljaja identiteta resursa s jedne i egzaktne, tehničke provjere s druge strane.

#### 5.6.4 Položaj

Položajni model je vjerojatno najslabija točka UriGraph-a, ne zbog internih problema ili nesuglasja, nego zbog njegove implicitnosti. Položajni model u UriGraph-u se u potpunosti oslanja na model identifikacije i kod složenijih grafova ustroja s velikom povezanosti čvorova teško je utvrditi veze između resursa. Na primjer, ako su dva mesta povezana prijelazom, može se reći da je ishodišni resurs na jedan način nadređen odredišnom. Međutim, kod višestrukih, nehijerarhijskih veza između šireg skupa resursa postaje teško utvrditi najvažnije odnose.

Osnovni problem je nedostatak semantike o važnosti pojedinih resursa. Za položajni model je od presudne važnosti prepoznavanje da li je pojedini resurs stranica, pomoćni prikazni resurs ili nešto treće. Zatim je važno prepoznati istaknute resurse (tzv. *landmark*) i podgrafove koji čine "niše" ili "logičke domene" u sjedištu. UriGraph, kako je definiran u ovom radu, za sada nema takvih mogućnosti.

Kod iole složenijih Web sjedišta/aplikacija, a takvima je UriGraph prilagođen, gotovo rješenje za problem generiranja jednostavnog položajnog modela je, iz današnje točke gledišta, utopija. Primjena jednostavnih položajnih modela, poput opisanih SGF-a i SiteBrain-a na primjer, može samo djelomično pomoći.

#### 5.6.5 Sastav resursa

Model sastava je u UriGraph-u prisutan uglavnom kroz ideju nasljeđivanja komponenti kroz graf ustroja. Funkcionalnost komponenti nije dio definicije UriGraph-a, već ovisi o drugim dijelovima aplikacije, kao što će biti pokazano kasnije u radu.

Uspoređujući ga s kvalitetnim modelom sastava kod WebML-a i iAS-ovog Portala, možemo uočiti najvažnije razlike:

- WebML nudi vrlo ograničen skup komponenti usko vezanih uz predmetni sustav. UriGraph i Portal ne ograničavaju značajno broj niti mogućnosti komponenti koje se mogu izraditi i ravnopravno koristiti s ugrađenima.

- U WebML-u, komponente se izravno povezuju kontekstnim vezama kako bi prenijele informaciju o identitetu, iako su na odvojenim stranicama. UriGraph prenosi tu informaciju preko identiteta stranica, slično kao Portal (iako Portal nema ugrađen koncept predstavljanja identiteta stranice).
- UriGraph komponente ne koristi samo za stranice, nego za sve resurse, za razliku od WebML-a i Portala.

### **5.6.6 Povezanost s poslužiteljem**

UriGraph je zamišljen kao modul poslužitelja, kao što je objašnjeno u 5.6.1, pa je stoga izravna povezanost s poslužiteljem očita. Ovaj put je arhitektura poslužitelja izrađena prema unaprijed smisljenoj teoriji korištenja Weba, za razliku od npr. klasičnog Web poslužitelja ili iAS-a.

### **5.6.7 Višejezičnost**

Kombinacijom opisanih načina višejezične analize zahtjeva (6.2.6) i odgovarajuće višejezičnosti predmetnog sustava može se izvesti istinski višejezičan ustroj sjedišta. On je s jedne strane konzistentan za svaki pojedini jezik, a s druge strane daje jedinstven opis svih ključnih svojstava resursa, bez obzira na jezik.

### **5.6.8 Ponovna usporedba modela**

Preglednu usporedbu odabralih modela ustroja (Tablica 2) proširujemo dodavanjem stupca s ocjenama UriGraph-a prema istim kriterijima (Tablica 3).

	<b>klasični model</b>	<b>SGF</b>	<b>WebML</b>	<b>SiteBrain</b>	<b>Oracle iAS</b>	<b>UriGraph</b>
obuhvaća sve resurse?	✓	✗	✗	✗	✓*	✓
identifikatori resursa?	✓*	✗	✗	✗	✗	✓
identitet resursa?	✗	✗	✗	✗	✗	✓
položaj resursa?	✓*	✓	✗	✓	✓*	✓
sastav resursa?	✗	✗	✓	✗	✓	✓ otvoren skup komponenti
neposredno povezan s poslužiteljem?	✓	✗	✗	✗	✓	✓
podržava višejezičnost?	✗	✗	✗	✗	✗	✓

legenda: ✓ - ispunjeno; ✓\* - djelomično ispunjeno; ✗ - nije ispunjeno

Tablica 3: Pregledna usporedba odabranih modela ustroja s UriGraph-om.

UriGraph ispunjava sve kriterije koje ovaj rad postavlja za potpun model ustroja Web sjedišta.

Rekapitulirajući razlike između UriGraph-a i ostalih opisanih modela ustroja, naglasio bih nekoliko točaka:

- UriGraph je prvenstveno namijenjen razvojnim inženjerima, a ne krajnjim korisnicima – mnoga njegova svojstva se mogu iskoristiti tek programiranjem;
- UriGraph je predviđen da se uklopi u odgovarajuću platformu za razvoj Web aplikacija velikog opsega – za skromnije potrebe može se pokazati učinkovitije korištenje klasičnih sredstava;
- za optimalno korištenje UriGraph-a (kao i za većinu drugih modela) potrebno je razviti odgovarajuće vizualne alate;
- ispravno korištenje UriGraph-a zahtijeva dobro poznavanje odgovarajuće teorije ustroja Web sjedišta (opisane u ovom radu).

*Nakon definicije modela, u sljedeća dva poglavља posvetiti ćemo pažnju njegovoj konkretnoj izvedbi u programskom jeziku Java i evaluaciji modela u eksperimentalnom okruženju.*

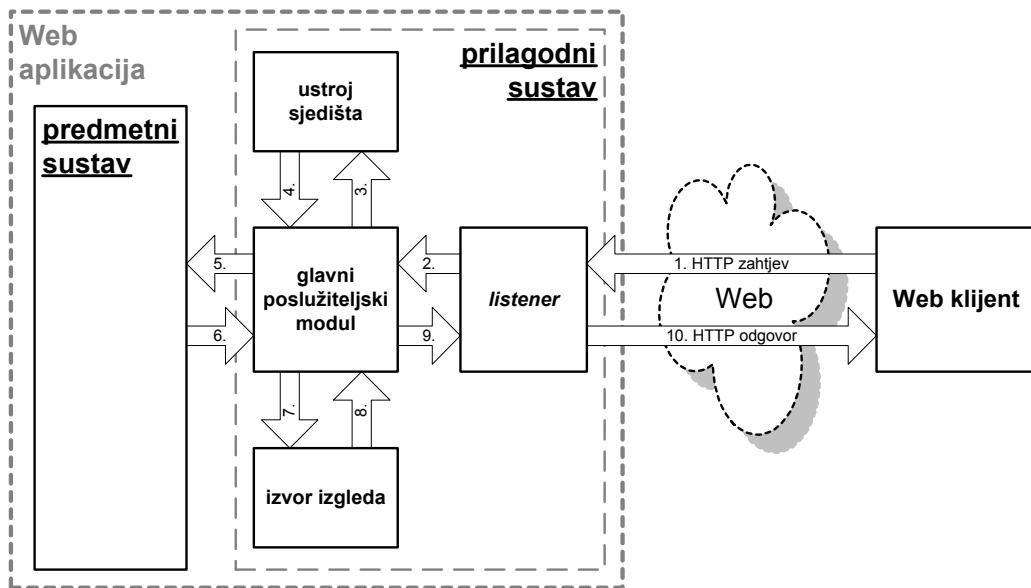
# 6 Programska izvedba

Ovo poglavlje opisuje način na koji je UriGraph izведен kao program, kao i Web poslužiteljska platforma čiji je dio. Opisana je i XML shema za opis grafa ustroja, a posebna pažnja je dana ispitivanju ispravnosti programa.

## 6.1 Okruženje

Programska izvedba UriGraph-a je smještena u šire programsko okruženje koje odgovara prilagodnom sustavu za određeno sjedište. Korištena je sljedeća programska podrška:

- Java2 verzija 1.3;
- Xerces i JDOM, XML parser za Javu;
- Xalan, XSLT [xslt10] podrška za Javu.



Slika 16: Skica programskog okruženja s približnim redoslijedom komunikacije između dijelova.

U grubim crtama, evaluacijska platforma se sastoji od sljedećih dijelova (Slika 16):

- **listener**-a koji sluša HTTP-ovski zahtjev, prevodi ga iz običnog teksta u Javin objekt i proslijeđuje glavnom poslužiteljskom modulu;
- **glavni poslužiteljski modul** koji povezuje ostatak prilagodnog sustava i komunicira s *listener*-om;
- **izvor izgleda** stranica;
- **ustroj sjedišta** – opisan u UriGraph-u;

- **predmetni sustav** koji je izvor sadržaja i funkcionalnosti – ima isključiv pristup sadržaju sjedišta.

### **6.1.1 Listener**

Za potrebe evaluacije UriGraph-a razvijen je poseban *listener* u Javi. Na taj način omogućena je potpuna kontrola platforme nad HTTP-ovim porukama. Komunikacija je uspostavljena preko Javine tehnologije *socket-a*.

### **6.1.2 Glavni poslužiteljski modul**

Funkcija glavnog poslužiteljskog modula se sastoji od:

- primanja HTTP-ovog zahtjeva pretvorenog u objekt;
- oblikovanja ustrojnog zahtjeva;
- na osnovu ustrojnog odgovora, izvršavanje određenih transakcija nad predmetnim sustavom;
- ovisno o odgovoru pojedinih komponenti, pokretanje transformacija sadržaja prema izvoru izgleda;
- konačnog sastavljanja odgovora poslužitelja i proslijedivanja *listener-u*.

### **6.1.3 Webspot**

"Webspot" je koncept u evaluacijskom okruženju koji povezuje Internetsku lokaciju s izvorištima sastojaka sjedišta. Internetska lokacija određuje osnovni URI sjedišta određujući naziv domaćina, port i put. Svaki izvor sastojka se pojedinačno određuje, kako bi se svaki mogao promjeniti nezavisno od ostalih.

U našem primjeru XML dokumenta (Slika 17) jedini parametar Internetske lokacije je port, pa je osnovni URI probnog sjedišta "<http://localhost:4772/>". Navedena su i tri izvora sastojaka za probno sjedište. Ustroj se čita iz XML datoteke, a sadržaj i izgled iz odgovarajućih Javinih klasa. Izvor sastojka funkcionalnosti je ujedno izvor sadržaja u ovoj izvedbi, pa nije naveden posebno.

```
<?xml version="1.0" ?>
<webspot port="4772">
  <structure-source file="w:\magisterij\xml\struct\docnotes.xml"/>
  <content-source class="hr.fer.tel.wance.test.notes.NotesContentSource"/>
  <look-source class="hr.fer.tel.wance.test.notes.NotesLook"/>
</webspot>
```

Slika 17: Primjer XML dokumenta koji definira *Webspot*.

### 6.1.4 Izvor izgleda

Sastojak izgleda je ugrađen u platformu tako da se uz svaku komponentu definira predložak koji pretvara sadržaj te komponente u oblik pogodan za korisnika – gotovu Web stranicu.

Pretvorba sadržaja u gotovu stranicu izvedena je uporabom tehnologije XSLT (*Extensible Stylesheet Language Transformations*) iz porodice XML tehnologija. Nad originalnim XML dokumentom u kojem se nalazi je sadržaj stranice izvršava se transformacija koja ga pretvara u XHTML, ograničenje HTML-a napravljeno da odgovara XML sintaksi.

Osim transformacije pojedinih komponenti, postoji poseban dio izvora koji se tiče rasporeda već transformiranih komponenti u stranici. Taj raspored se utvrđuje na osnovu straničnog konteksta.

## 6.2 Izvedba ustroja po UriGraph-u u Javi

U nastavku je dan pregled klasne strukture izvedbe UriGraph-a u Javi.

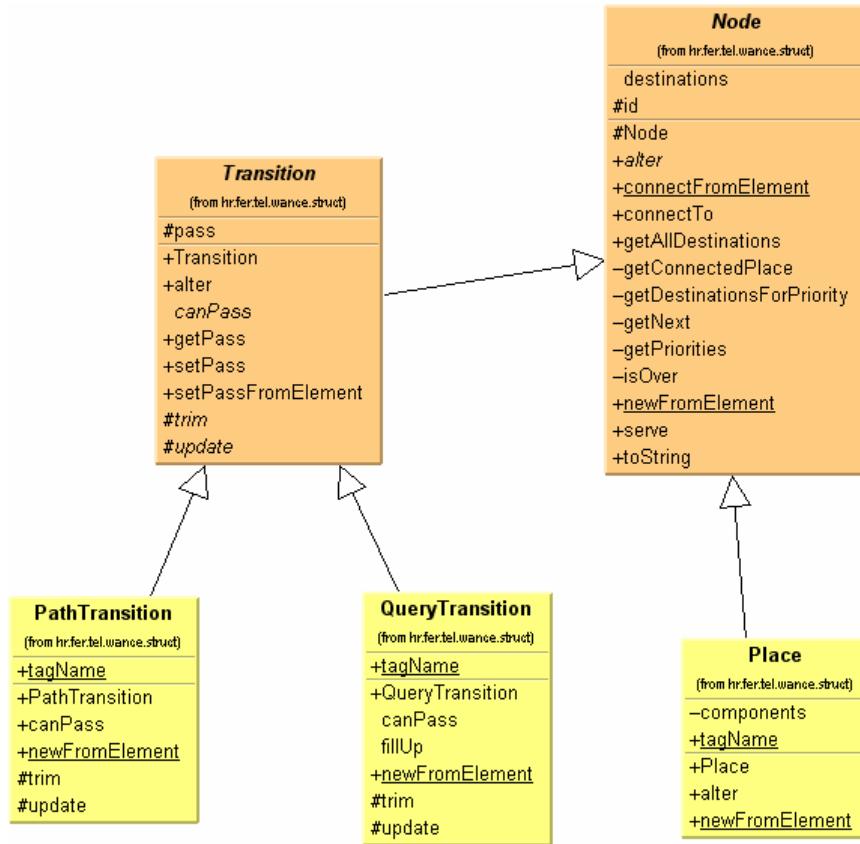
### 6.2.1 Klase sučelja ustroja

Pojedini graf ustroja predstavlja objekt klase **Structure**. Graf ustroja je određen korijenskim čvorom, koji je odgovarajuće povezan s ostatkom grafa.

Formiranje zahtjeva se svodi na instanciranje klase **Request**, zadavanjem puta i upita URI-ja. Obrada ustroja koristi se pozivom njegove metode **getResponse** koja prima zahtjev kao argument, klonira ga i unosi u obradu korijenskog čvora, čija metoda rekurzivno poziva sebe za sljedeće (odredišne) čvorove u grafu. Konačan odgovor se vraća kao objekt klase **Response**.

### 6.2.2 Vrste čvorova i topologija

Hijerarhija klase koja predstavlja vrste čvorova definirane osnovnim modelom UriGraph-a. Općeniti čvor u grafu predstavlja apstraktna klasa **Node** iz koje se izvode klase **Place** (predstavlja mjesto) i apstraktna klasa **Transition** (prijelaz). **PathTransition** i **QueryTransition** su dalje izvedene iz **Transition**-a. Ovakva klasna hijerarhija jasno prati odnose među pojmovima (Slika 18).

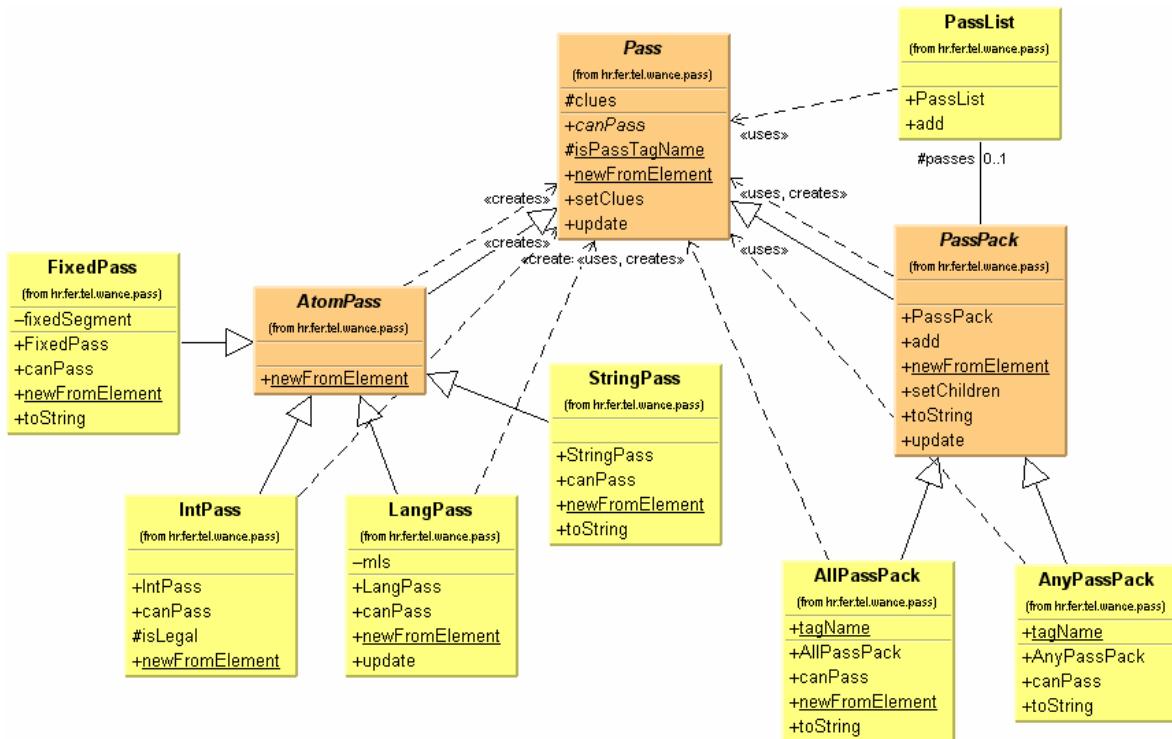


Slika 18: UML-ov klasni dijagram vrsta čvorova prema modelu UriGraph.

Čvorovi se dodaju instanciranjem tri neapstraktne klase čvorova, i povezivanjem metodom **connectTo** s ostatkom grafa.

### 6.2.3 Modeliranje propusnica

Klasna hijerarhija propusnica počinje općom, apstraktnom klasom **Pass**, iz koje se izvode apstraktne klase koje predstavljaju elementarnu (**AtomPass**) i složenu (**PassPack**) propusnicu. Funkcionalnost složene propusnice implementirana je u dvije klase: **AllPassPack** (konjuktivna složena propusnica) i **AnyPassPack** (disjuktivna složena propusnica).



Slika 19: UML-ovski dijagram izvedbe propusnica u Javi.

Iz osnovne klase za elementarne propusnice izvedeno je više konkretnih propusnica. Sve moraju implementirati metodu **canPass**, koja za zadani segment i kontekst (skup dotad prikupljenih natuknica o identitetu resursa) vraća odgovor na pitanje je li propusnica prohodna ili nije.

Propusnice su univerzalne za oba vrsta segmenta, puta i upita. Većina je implementirana da provjerava segment kroz dvije vrijednosti: ključ i podatak. Segment puta za obje vrijednosti vraća svoj sadržaj. Segment upita vraća svoj ključ za ključ segmenta, dok za podatak vraća svoju vrijednost, ako postoji, inače svoj ključ.

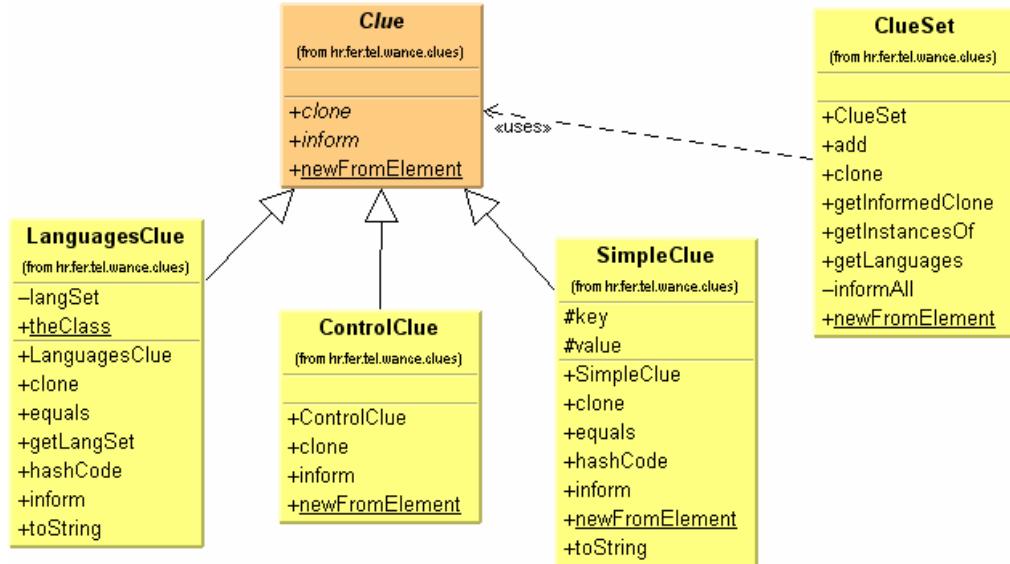
## 6.2.4 Modeliranje natuknica

Sve natuknice su izvedene iz apstraktne klase **Clue**. Osnovna metoda koju sve konkretnе klase moraju implementirati je **inform**, spremanje informacije prikupljene iz očitanog segmenta. Svaka pojedina natuknica mora odlučiti da li i kako tu informaciju obraditi i pohraniti.

Natuknice se skupljaju u objekte klase **Clueset**, izvedene iz `java.util.HashSet`. Stoga svaka **Clue**-ovska klasa mora implementirati metode `equals` i `hashCode`.

Druga važna osobina klase **Clueset** je metoda **getInformedClone** koja vraća duboko klonirani skup natuknica s već prikupljenim informacijama. Stoga klasa **Clue**

implementira sučelje `Cloneable` i svaka njena podklasa mora odgovarajuće implementirati duboko kloniranje.



Slika 20: Hjerarhija klasa za izvedbu natuknica s nekim konkretnim izvedbama natuknica.

### 6.2.5 Izvedba analize

Analiza zahtjeva počinje od korijenskog čvora grafa ustroja. Obradu čvora predstavlja metoda `serve`. Ona prima objekte klase `Request` i `Response`, zahtjev i odgovor, te ih mijenja (u pravilu smanjujući zahtjev i povećavajući odgovor) rekursivno pozivajući istu metodu nad objektima dalje u prolazu kroz graf.

Obrada čvora ima dvije istaknute operacije: **promjenu zahtjeva i odgovora** metodom `alter` (zahtjev se smanjuje, a odgovor povećava ugrađujući informaciju iz zahtjeva i samog ustroja) te **odabir sljedećeg čvora** metodom `getNext`.

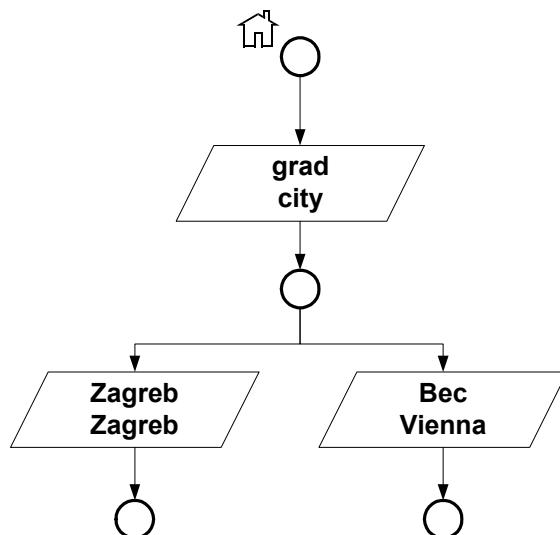
Promjena zahtjeva i odgovora delegira se dalje na klase `Place` i `Transition`. Kod mjesta promjena se svodi na dodavanje komponenti u odgovor (naslijednih prije kraja i lokalnih na kraju). Kod prijelaza promjena se dijeli na ažuriranje odgovora (`update`) i skraćenje zahtjeva (`trim`). Skraćenje zahtjeva je definirano u klasama `PathTransition` i `QueryTransition`, a ažuriranje odgovora se dalje delegira na propusnice, pozivajući već pojašnjenu metodu `getInformedClone`.

Druga operacija, odabir sljedećeg čvora za analizu, se također svodi na pozivanje odgovarajućih metoda `canPass` u podklasama. Svako određeno mjesto automatski je odabранo, dok svaki prijelaz mora proći provjeru prolaznosti. Prijelaz puta ispituje prvi segment puta, a prijelaz upita svaki segment upita nad svojom propusnicom. Složene propusnice mogu isto pitanje delegirati svojim podpropusnicama.

## 6.2.6 Višejezična analiza

Za potrebe višejezičnih prijelaza napravljene su posebne klase ugrađene u postojeći okvir:

- **LanguagesClue** – posebna, ugrađena natuknica o skupu jezika u kojem bi mogao biti odgovor;
- **LangPass** – propusnica za višejezične segmente koji imaju nepromjenjive ključne vrijednosti segmenta za svaki jezik.



Slika 21: Primjer jednostavnog višejezičnog sjedišta. U prvom retku su nazivi na hrvatskom, u drugom na engleskom.

Za primjer (Slika 21), pretpostavimo da sjedište podržava dva jezika: hrvatski i engleski. Sustav sâm na početku analize dodaje objekt klase **LanguagesClue** koji sadrži skup ova dva jezika. Za zahtjev "/" ta će natuknica ostati ista i neće biti određeno iz zahtjeva na kojem od dva jezika se treba prikazati naslovnička. Za zahtjev "/grad/Zagreb" jezični skup se izmijeni tako da sadrži samo hrvatski jezik već nakon prvog prijelaza. Zahtjev "/city/Bec" je neispunjiv jer se skup jezika sveo na samo engleski nakon prvog prijelaza i stoga je sljedeći zahtjev "/Bec" neispunjiv među propusnicama koje propuštaju samo "Zagreb" i "Vienna".

## 6.2.7 Modeliranje komponenti

Komponenta resursa je instanca klase **Component** koja koristi klasu **ClueSet** za vlastiti skup natuknica. Izrada vlastite komponente ovisi o njenoj funkcionalnosti, što nadilazi domenu ustroja sjedišta. Obično komponenta vraća sadržaj u obliku XML elementa i/ili provodi transakciju nad predmetnim sustavom i vraća kod preusmjeravanja.

## 6.2.8 Izvedba sinteze

Sinteza se sastoji od:

- prikupljanja natuknica za resurs;
- prikupljanja komponenti i natuknica za tu komponentu;

Osnovna metoda za prikupljanje natuknica je `update`, ažuriranje odgovora, spomenut u 6.2.5. Sastoji se od kloniranja natuknica u određenom čvoru, "informiranja" tih natuknica i uključivanja u skup natuknica iz odgovora. "Informiranje" natuknica (metoda `inform`) prima segment i očitava korisnu informaciju iz njega, kako je opisano u 6.2.4.

Prikupljanje komponenti se odvija u metodi `alter` klase `Place`. Ona jednostavno dodaje nasljedne komponente u odgovor dok se zahtjev ne isprazni, onda dodaje lokalne komponente.

## 6.2.9 Rukovanje iznimkama

Nadklasa svim iznimkama u ustrojnom dijelu je `StructureException`, izvedena iz osnovne klase iznimaka platforme `hr.tel.fer.wance.WanceException`, i posredno iz `java.lang.Exception`.

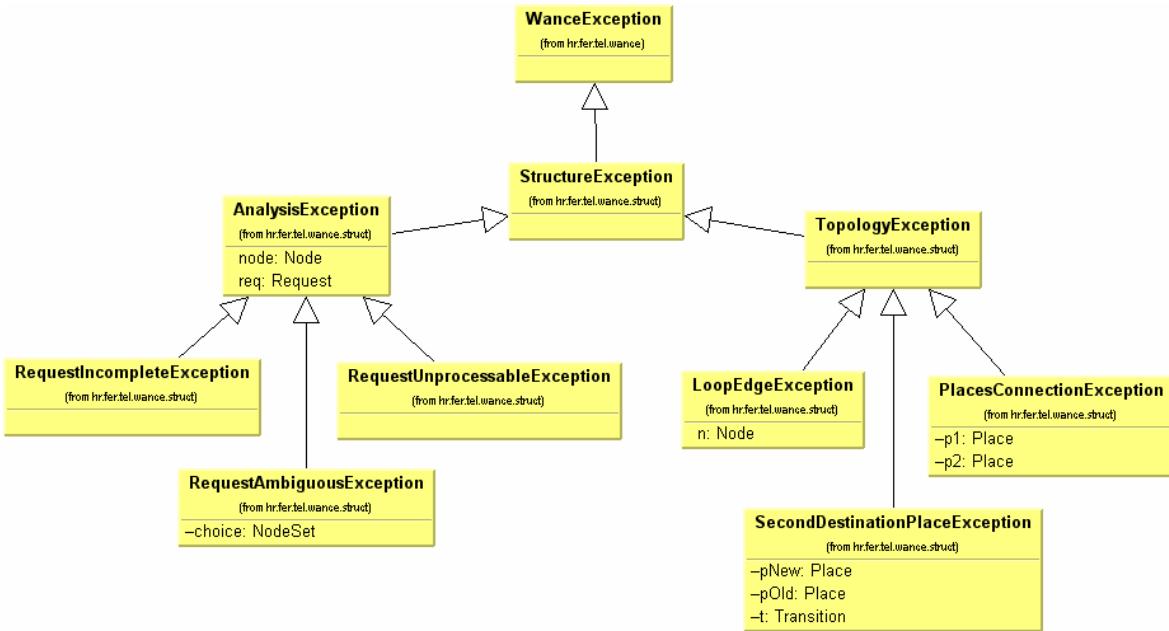
Nelegalni slučajevi pri izgradnji topologije predstavljeni su klasom `StaticStructureException`. Pojedini slučajevi uporabe ove iznimke:

- `LoopEdgeException` – pokušaj spajanja čvora sa samim sobom (petlje);
- `PlacesConnectionException` – pokušaj spajanja mjesta s mjestom;
- `SecondDestinationPlaceException` – pokušaj spajanja prijelaza sa još jednim mjestom.

Klasa koja predstavlja općenitu iznimku pri analizi zahtjeva je `AnalysisException`.

Osnovne podklase za iznimne slučajeve prema 5.3.8 su:

- `RequestIncompleteException` za nepotpun zahtjev;
- `RequestUnprocessableException` za neobradiv zahtjev;
- `RequestAmbiguousException` za više značan zahtjev.



Slika 22: Osnovna organizacija ustrojnih iznimaka. Svaka klasa sadrži objekte potrebne za detaljan opis iznimne situacije.

## 6.3 XML shema za opis modela

Za zapis grafa ustroja po modelu UriGraph osmišljena je XML-ova shema zapisa. XML se danas udomaćio kao sintaktički okvir za ovakve formate i pruža veliku softversku podršku (pogotovo u Javi) i kompatibilnost. Za više informacija o XML-u vidi [xml00].

Osnovno načelo pri izradi ove sheme bila je **proširivost**. Za svaki entitet u modelu za koji je predviđena otvorena funkcionalnost u smislu pisanja novih klasa u Javi koje ostvaruju željeni učinak, predviđeno je da i definicija odgovarajućeg elementa u shemi bude otvorena za unos naziva klase i potrebnih parametara. Imenovane klase se u programu koji čita dokument dinamički učitavaju i instanciraju.

### 6.3.1 Osnovni element

Osnovni, korijenski element koji sadrži sve druge elemente u XML dokumentu opisuje jedan graf ustroja i naziva se **structure**. Sadrži sljedeće atribute:

- **root-id** – Identifikator čvora koji čini korijen grafa (identifikatori čvorova su opisani dalje u poglavlju).
- **java-base** – Znakovni niz koji se koristi kao osnovica pri nazivanju Java paketa, prefiks svakog naziva klase koja dolazi dalje u dokumentu, a počinje s tri točke (npr. ako je osnovica "hr.fer.tel.wance", naziv klase dalje u dokumentu "...pass.LangPass" biti će pretvoren u pun naziv klase "hr.fer.tel.wance.pass.LangPass"). Na ovaj način se može skratiti

tekst XML dokumenta i učiniti preglednijim. Nazivi klase koji ne odgovaraju osnovici jednostavno se pišu bez početne tri točke.

Element **structure** sadržava više elemenata čvorova, opisanih u nastavku.

### 6.3.2 Topologija

Topologija je opisana nizom elemenata za opis čvora, poredanih bez posebnog reda, ali s označenim vezama (granama). Nazivi tih elemenata su:

- **place** za mesta;
- **path-transition** za prijelaze puta;
- **query-transition** za prijelaze upita.

Svaki element za opis čvora, bez obzira o vrsti, sastoji se od sljedećeg:

- Atribut **id** predstavlja identifikator čvora. Za svaki čvor to je jedinstveni niz znakova koji služi kao referenca pri povezivanju čvorova.
- Nula ili više elemenata za povezivanje s drugim čvorovima "**connect-to**". Ti elementi sadrže samo atribut **id** koji predstavlja identifikator odredišnog čvora i neobavezni atribut **priority** s oznakom prvenstva te grane.

Ostatak elementa varira ovisno o vrsti čvora. Mesta mogu imati elemente koji opisuju komponente, dok prijelazi mogu sadržavati elemente za opis propusnica.

### 6.3.3 Propusnice

Elementi koji predstavljaju propusnice se postavljaju u prijelaze, tj. u elemente **path-transition** i **query-transition**. Definirana su tri elementa, dva za posebne vrste složenih propusnica i jedan općeniti:

- **pass-all** predstavlja konjuktivnu složenu propusnicu;
- **pass-any** predstavlja disjuktivnu složenu propusnicu;
- **pass** predstavlja općenitu propusnicu čija je klasa definirana atributom.

U slučaju da se u elementu prijelaza nalazi više elemenata propusnice, podrazumijeva se da su vezani u konjuktivnu složenu propusnicu.

Elementi složenih propusnica nemaju posebnih elemenata niti atributa, osim što sadrže druge elemente propusnica. Međusobna sintaktička razlika je samo u imenu.

Element općenite propusnice može sadržavati više atributa. Atribut "**class**" sadrži naziv Javine klase opisane propusnice. Ostali atributi, elementi i njihove vrijednosti su proizvoljne, i koristi ih ta klasa.

Zajednička osobina svake propusnice je što mogu sadržavati više podelemenata natuknica, kao što će biti objašnjeno kasnije.

### 6.3.4 Komponente

Komponenta se opisuje elementom **component**, koji se smješta u element mesta. On sadrži atribut **class** (kao i propusnica), ali i dva posebna logička atributa: **local** i **hereditary**. Prvi atribut navodi da li komponenta ima svojstvo lokalnosti (podrazumijeva se **true**), a drugi da li ima svojstvo nasljedivosti (podrazumijevana vrijednost je **false**).

Svaka komponenta može sadržavati natuknice kao podelemente.

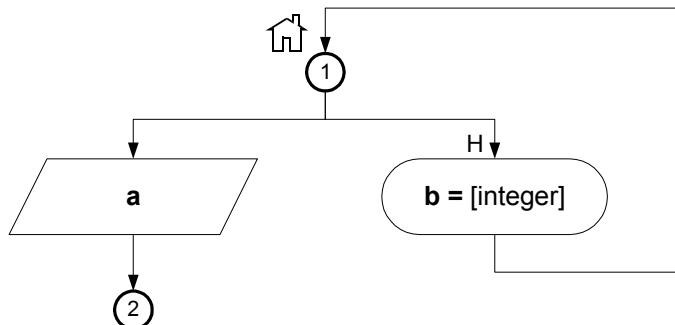
### 6.3.5 Natuknice

Natuknice se opisuju u posebnom elementu "**clue**". Taj element se može nalaziti kao podelement u svakom elementu propusnice (kao natuknica o identitetu resursa) i u svakom elementu komponente (kao natuknica o identitetu komponente resursa).

Element natuknice ima također atribut **class** i proizvoljne ostale atribute, ovisno o navedenoj klasi.

### 6.3.6 Primjer XML dokumenta

Ilustrirati ćemo uporabu sheme na jednostavnom i reprezentativnom primjeru grafa (Slika 23).



Slika 23: UriGraph za primjer XML dokumenta.

Zapis (Slika 24) uključuje i neke dodatne informacije. Dodane su dvije komponente u mjestu. U prvu komponentu je ugrađena natuknica, kao i u sve propusnice. Semantika ovih komponenti i natuknica nije važna i zadana je samo kao ilustracija sintakse.

```
<?xml version="1.0" ?>
<structure root-id="p-root" java-base="hr.fer.tel.wance">

  <place id="p1">
    <connect-to id="ta" />
    <connect-to id="tb" priority="H" />
    <component class="...test.TestRootComp" hereditary="true">
      <clue class="...clues.SimpleClue" key="root" />
    </component>
  </place>

  <path-transition id="ta">
    <connect-to id="p2" />
    <pass class="...pass.FixedKeyPass" key="a" />
    <clue class="...clues.SimpleClue" key="a" />
  </path-transition>

  <place id="p2">
    <component class="...test.TestAComp" />
  </place>

  <query-transition id="tb">
    <connect-to id="p1" />
    <pass class="...pass.FixedKeyPass" key="b" />
    <pass class="...pass.IntPass">
      <clue class="...test.IdClue" />
    </pass>
  </query-transition>
</structure>
```

Slika 24: Primjer zapisa UriGraph-a u XML-u.

Za složeniji primjer XML zapisa vidi 7.2.5.

## 6.4 Ispitivanje ispravnosti

Ispitivanje ispravnosti paketa koji implementiraju ustroj prema UriGraph-u sastoji se od više razina:

1. preliminarnog ispitivanja klase pri kodiranju, obično provizornom izradom metode **main** s pokušnim kodom u ispitivanoj klasi;
2. formalnog ispitivanja funkcionalnosti paketa ustroja preko posebne klase, nazvane **StructTest**;
3. operativnog ispitivanja u evaluacijskom okruženju koje uključuje izradu XML-ovih dokumenata za opis ustroja i drugih ulaznih podataka, te ručnog ispitivanja očekivane funkcionalnosti sjedišta korištenjem korisničkog sučelja (Web preglednika).

Najpomnije je izrađena druga razina ispitivanja, čiji je cilj brzo ispitivanje osnovne funkcionalnosti modula ustroja s praktično dovoljno visokom pouzdanosti. Za svaki pojedini test u kodu su navedeni uvjeti, ulaz i očekivani izlaz, kao i funkcionalnost koja izvodi testnu operaciju, uspoređuje dobiveni izlaz s očekivanim i prijavljuje odstupanje.

Ispitivanje se sastoji od više uzastopnih zbirki testova (*test suites*). Svaka predstavlja jedan korak dalje u složenosti modela, prema njegovim slojevima, od topologije, preko analize zahtjeva do sinteze odgovora. To su redom zbirke za:

- instanciranje čvorova probnog grafa;
- provjera ispravnog i neispravnog povezivanja tih čvorova;
- postavljanje raznovrsnih propusnica u sve prijelaze grafa;
- provjera ispravnih i neispravnih prolaza kroz oformljeni graf;
- postavljanje novih i prepravljanje nekih starih grana tako da sadrže informaciju o prvenstvu;
- provjera ispravnih i neispravnih prolaza kroz ovako izmijenjen graf s granama s prvenstvom;
- postavljanje različitih natuknica u propusnice u grafu;
- provjera točnosti natuknica u odgovorima na zadane zahtjeve;
- provjera točnosti komponenti u odgovorima na zadane zahtjeve.

*Nakon opisa modela, njegove izvedbe okruženja, Javinih klasa, XML sheme i metoda provjere ispravnosti, pokazati ćemo na jednom studijskom slučaju kako se UriGraph može primjeniti.*

# 7 Studijski slučaj

Posljednje poglavlje prikazuje primjenu UriGraph-a u odabranom studijskom slučaju. Opisano je jedno pokušno sjedište te izrada njegovog grafa ustroja i ostali sastojaka. Na kraju su pokazani detalji rada poslužitelja za tako izrađeno sjedište u nekoliko karakterističnih primjera zahtjeva.

## 7.1 Zahtjevi

Za potrebe ilustracije i procjene UriGraph-a predstavljeno je sjedište koje prikazuje podatke, prima ih i izvršava transakcije nad predmetnim sustavom.

### 7.1.1 Predmetni sustav

Za probno sjedište odabran je jednostavan ali reprezentativan predmetni sustav bilježenja tekstualnih poruka. Jedna bilješka (*note*) se sastoji od naslova, samog teksta, datuma stvaranja i identifikatora autora. Svakoj bilješki je nametnut njen identifikator, prirodan broj.

Nad bilješkama su omogućene osnovne operacije vraćanja podataka o identificiranoj bilješci, stvaranja nove bilješke, izmjene podataka postojeće i brisanja bilješke. Svi tekstualni podaci se čuvaju usporedno na engleskom i hrvatskom jeziku.

Kao primjer podataka u predmetnom sustavu (sadržaja sjedišta) unijeli smo nekoliko citata poznatih ljudi (Tablica 4). Prikazani su samo engleski prijevodi, iako postoje i hrvatski. Datum stvaranja bilješke je predstavljen (približnom) godinom nastanka citata.

<b>id</b>	<b>naslov</b>	<b>tekst bilješke</b>	<b>autor</b>	<b>datum</b>
1	Education	I have never let my schooling interfere with my education.	Twain	1905
2	Classics	Something that everybody wants to have read and nobody wants to read.	Twain	1900
3	On Twain	Mark Twain and I have to put things in such a way as to make people, who would otherwise hang us, believe that we are joking.	Shaw	1940
4	Last words	Even in the valley of the shadow of death, two and two do not make six.	Tolstoy	1910
5	Logic	Contrariwise, if it was so, it might be; and if it were so, it would be: but as it isn't, it ain't. That's logic.	Carroll	1871
6	Mathematics and reality	As far as the laws of mathematics refer to reality, they are not certain, and as far as they are certain, they do not refer to reality.	Einstein	1975

Tablica 4: Sadržaj probnog sjedišta na engleskom jeziku.

### **7.1.2 Identitet resursa**

Početi ćemo utvrđivanjem klasa resursa koje će sadržavati naše sjedište i njihovo temeljno svojstvo, identitet. Svaku klasu resursa označiti ćemo rednim brojem. Osnovni resurs je:

1. naslovница, stranica koja predstavlja cijeli sustav.

Resursi vezani uz prikaz i unos podataka o bilješkama su također **stranice**:

2. bilješka općenito;
3. određena bilješka;
4. skup izabranih bilježaka prema opsegu identifikatora (od-do), odnosno pojedinačno navedenih bilježaka;
5. nova bilješka – obrazac za unos podataka radi stvaranja nove bilješke
6. izmjena postojeće bilješke – stranica koja omogućuje prepravku nekih podataka bilješke, npr. putem istog obrasca kao i za unos nove.

Osim stranica, potrebni su i **izvršni** resursi koji izvršavaju određenu transakciju nad predmetnim sustavom i preusmjeravaju klijenta na drugi, prikazni resurs:

7. stvaranje nove bilješke s primljenim podacima;
8. izmjena podataka određene bilješke;
9. brisanje određene bilješke ili skupa bilježaka.

Konačno, slike su primjer **pomoćnog** resursa u sjedištu, jer se koriste pri oblikovanju stranica:

10. prikaz slike.

### **7.1.3 Identifikacija resursa**

Sučelje prema probnom sjedištu je predstavljeno preko oblika URI-ja (Tablica 5).

<b>klasa resursa</b>	<b>URI</b>	<b>značenje</b>
1	/	naslovница
2	/note	o svim bilješkama
3	/note/ <i>noteID</i>	bilješka br. <i>noteID</i>
4	/note?from=x&to=y	bilješke od br. x do br. y
	/note?id=x&id=y&...	bilješke br. x, y...
5	/note/new	nova bilješka
6	/note/ <i>noteID</i> /edit	uredi bilješku br. <i>noteID</i>
7	/note/create	stvori novu bilješku
8	/note/ <i>noteID</i> /update	izmijeni bilješku br. <i>noteID</i>
9	/note/ <i>noteID</i> /delete	izbriši bilješku br. <i>noteID</i>
	/note/delete?from=x&to=y	izbriši bilješke od br. x do br. y
	/note/delete?id=x&id=y&...	izbriši bilješke br. x, y...
10	/image- <i>imgID</i>	slika br. <i>imgID</i>

Tablica 5: Oblici identifikatora u probnom sjedištu.

Uz navedene oblike URI-ja postoje i odgovarajući URI-ji na hrvatskom jeziku.

Osim navedenih zahtjeva na URI-je, postoji potreba za uvođenjem posebnog, globalnog segmenta upita "**content-only**" ("samo sadržaj"). Ovaj prijelaz ima posebnu funkciju reguliranja entiteta koji se vraća na HTTP-ov zahtjev GET. Ako ovaj segment upita nije naveden sadržaj stranice se transformira izgledom stranice i takav dostavlja klijentu. Međutim, ako je navedeno da se traži samo sadržaj, nema transformacije i klijentu se dostavlja neobrađen sadržaj u XML formatu.

#### 7.1.4 Položaj resursa

Položaj svakog od resursa u sjedištu prati organizaciju svog identifikatora.

#### 7.1.5 Sastav resursa

Postoje različiti načini izrade komponenata za svako zadano sjedište. Osnovni izbor se svodi na omjer informacije: koliko se prenosi informacijom o vrsti komponente a koliko natuknicama. Naš izbor je bio manji broj komponenti: dvije osnovne i dvije pomoćne. Jedna osnovna komponenta predstavlja individualnu bilješku, a druga skup bilježaka.

Poimence, to su:

- **bilješka** – prikazuje bilješku detaljno (sa svim podacima), može je stvoriti, izmijeniti i izbrisati;
- **odabir bilježaka** – prikazuje popis bilježaka samo s naslovom, briše ih;
- **obrazac za bilješke** – prikazuje obrazac za unos ili izmjenu podataka o bilješci;

- **slika** – prikazuje sliku u odgovarajućem formatu.

Komponente su raspoređene po resursima na sljedeći način:

klasa resursa	vrsta komponente	lokalna	nasljedna	natuknice komponente
1	odabir bilježaka	da	ne	posljednjih pet
2	odabir bilježaka	da	ne	(sve)
3	bilješka	da	da	-
4	odabir bilježaka	da	da	-
5	obrazac za bilješke	da	ne	-
6	obrazac za bilješke	da	ne	-
7	bilješka	da	ne	-
10	slika	da	ne	-

Tablica 6: Raspored komponenti na probnom sjedištu.

## 7.2 Izvedba u UriGraph-u

Uz pojašnjenje izvedbe predmetnog sustava, izvedba ustroja sjedišta je opisana prema slojevima UriGraph-a.

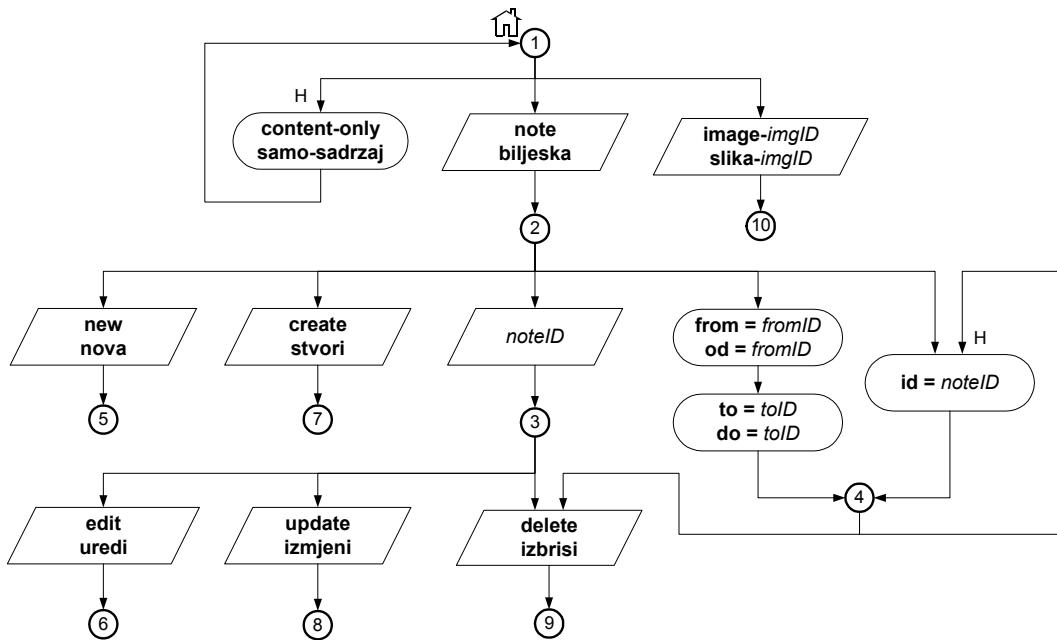
### 7.2.1 Predmetni sustav

Osnovni pojam bilješke implementira klasa **Note**. Međutim, najrazrađenija klasa je **Repository**, koja predstavlja spremnik svih bilješki u sustavu. Sučelje prema njemu je **NotesContentSource**, izведен iz klase **ContentSource** koja predstavlja sve izvore sadržaja. Spremnik pruža mogućnost izravnog rukovanja bilješkama putem metoda, dok izvor sadržaja omogućuje dodatno sučelje koje vraća gotove XML-formatirane dokumente.

Prije rada na prilagodnom sustavu (tj. Web sjedištu) važno je da predmetni sustav bude dobro definiran, kvalitetno implementiran i da prema njemu postoji objektno sučelje. On ne sadrži funkcionalnost niti podatke koji su posebni za probno sjedište ili za Web općenito, ali s druge strane prikuplja sav sadržaj i obavlja svu funkcionalnost sjedišta.

### 7.2.2 Topološki sloj

Priložen je grafički prikaz UriGraph-a na kojem je vidljiv topološki sloj rješenja zajedno sa slojem analize zahtjeva (Slika 25).



Slika 25: Graf ustroja probnog sjedišta s naznačenim segmentima. U mjestima su brojevima označene pojedine klase resursa.

### 7.2.3 Sloj analize zahtjeva

Identifikatori za navedene resurse dani su na dva jezika, engleskom i hrvatskom. Model identifikacije (Slika 25) izrađen je prema zahtjevima (Tablica 5). Fiksne vrijednosti pojedinih segmenata su zapisane podebljano, a varijabilne kurzivom. U prvom retku je engleski prijevod fiksnih vrijednosti, a u drugom hrvatski.

Gotovo sve propusnice su iz klase **LangPass** (vidi 6.2.6). Kod prijelaza slika upotrijebljena je posebna složena propusnica **CompositePass** koja svoje podpropusnice ispituje nad dijelovima segmenta. U ovom slučaju znak razdvajanja je crtica ("—"). Dio segmenta prije crticice ispitana je višejezičnom propusnicom, a ostatak posebnom propusnicom **ImageIdPass** koja provjerava identifikator slike.

Slična propusnica, **NoteIdPass**, koristi se na dva mesta u grafu gdje se provjerava sadrži li segment identifikator postojeće bilješke. Propusnica postavlja to pitanje spremniku bilješki iz predmetnog sustava.

### 7.2.4 Sloj sinteze odgovora

Najsloženiji dio modeliranja probnog sjedišta je izrada klase za sloj sinteze odgovora. Za nju valja izraditi klase koje služe kao komponente. Komponente prepoznaju što trebaju raditi na osnovu dobivenih natuknica i koriste predmetni sustav preko izvora sadržaja kako bi ispunile zadaću.

Tipičan primjer takve komponente je **NoteComp**, komponenta "bilješka" kako je definirana u 7.1.5. Ona ukupno ima četiri funkcije:

1. prikaz detalja određene bilješke – prima jedan **NoteClue**, a vraća XML element;
2. stvara novu bilješku – prima odgovarajuće podatke o novoj bilješci iz tijela HTTP-ovog zahtjeva, a vraća preusmjeravanje na informacije o toj bilješki
3. mijenja sadržaj postojeće bilješke – prima jedan **NoteClue** i odgovarajuće izmjenjene podatke o bilješci iz tijela HTTP-ovog zahtjeva, a vraća preusmjeravanje na informacije o toj bilješki;
4. briše postojeću bilješku – prima jedan **NoteClue**, a vraća preusmjeravanje na resurs o svim bilješkama.

Za razliku od **NoteComp**-a, koji prima svu informaciju o svom identitetu iz jedne natuknice, **NoteSelectionComp** može primiti više natuknica, od kojih neki upućuju na raspon bilježaka, a ne na pojedinačnu bilješku.

### 7.2.5 Zapis ustroja u XML-u

Za formiranje XML zapisa treba uvesti jedinstvene identifikatore za svaki čvor grafa. Na našem primjeru (Slika 26) identifikatori čvorova su opisni, svako mjesto počinje slovom "p", a prijelaz slovom "t".

```
<?xml version="1.0" ?>
<structure
  root-id="p-root"
  java-base="hr.fer.tel.wance"
  clue-default-class="...clues.simpleClue"
  pass-default-class="...pass.StringPass"
>

<place id="p-root">
  <connect-to id="t-content-only" priority="H" />
  <connect-to id="t-note" />
  <connect-to id="t-image-id" />
  <component class="...test.notes.NoteSelectionComp">
    <clue class="...clues.OrderBy" name="date" order="descending" />
    <clue class="...clues.OrdinalRange" from="1" to="5" />
  </component>
</place>

<query-transition id="t-content-only">
  <connect-to id="p-root" />
  <pass class="...pass.LangPass">
    <translation lang="EN" text="content-only" />
    <translation lang="HR" text="samo-sadrzaj" />
    <clue class="...clues.ControlClue" key="transform" value="false" />
  </pass>
</query-transition>

<path-transition id="t-note">
  <connect-to id="p-any-note" />

```

```

<pass class="...pass.LangPass">
    <translation lang="EN" text="note" />
    <translation lang="HR" text="biljeska" />
    <clue key="note" />
</pass>
</path-transition>

<place id="p-any-note">
    <connect-to id="t-new-note" />
    <connect-to id="t-note-id" />
    <connect-to id="t-create-note" />
    <connect-to id="t-from-note" />
    <connect-to id="t-multiple-note-id" />
    <component class="...test.notes.NoteSelectionComp" />
</place>

<path-transition id="t-new-note">
    <connect-to id="p-new-note" />
    <pass class="...pass.LangPass">
        <translation lang="EN" text="new" />
        <translation lang="HR" text="nova" />
        <clue class="...test.notes.NoteClue" key="new" />
    </pass>
</path-transition>

<place id="p-new-note">
    <component class="...test.notes.NoteFormComp" />
</place>

<path-transition id="t-create-note">
    <connect-to id="p-create-note" />
    <pass class="...pass.LangPass">
        <translation lang="EN" text="create" />
        <translation lang="HR" text="stvori" />
        <clue class="...test.notes.NoteClue" key="create" />
    </pass>
</path-transition>

<place id="p-create-note">
    <component class="...test.notes.NoteComp" />
</place>

<path-transition id="t-note-id">
    <connect-to id="p-specific-note" />
    <pass class="...test.notes.NoteIdPass">
        <clue class="...test.notes.NoteIdClue" key="id" />
    </pass>
</path-transition>

<place id="p-specific-note">
    <connect-to id="t-edit-note" />
    <connect-to id="t-update-note" />
    <connect-to id="t-delete-note" />
    <component class="...test.notes.NoteComp" hereditary="true" />
</place>

<path-transition id="t-edit-note">
    <connect-to id="p-edit-note" />
    <pass class="...pass.LangPass">
        <translation lang="EN" text="edit" />
        <translation lang="HR" text="uredi" />
        <clue class="...test.notes.NoteClue" key="edit" />
    </pass>
</path-transition>

<place id="p-edit-note">
    <component class="...test.notes.NoteFormComp" />
</place>

<path-transition id="t-update-note">
    <connect-to id="p-update-note" />
    <pass class="...pass.LangPass">
        <translation lang="EN" text="update" />
        <translation lang="HR" text="izmjeni" />
        <clue class="...test.notes.NoteClue" key="update" />
    </pass>
</path-transition>

```

```

<place id="p-update-note" />

<path-transition id="t-delete-note">
  <connect-to id="p-delete-note" />
  <pass class="...pass.LangPass">
    <translation lang="EN" text="delete" />
    <translation lang="HR" text="izmjeni" />
    <clue class="...test.notes.NoteClue" key="delete" />
  </pass>
</path-transition>

<place id="p-delete-note" />

<query-transition id="t-from-note">
  <connect-to id="t-to-note" />
  <pass class="...pass.LangPass">
    <translation lang="EN" text="from" />
    <translation lang="HR" text="od" />
  </pass>
  <pass class="...test.notes.NoteIdPass">
    <clue class="...test.notes.NoteIdClue" key="from" />
  </pass>
</query-transition>

<query-transition id="t-to-note">
  <connect-to id="p-selected-notes" />
  <pass class="...pass.LangPass">
    <translation lang="EN" text="to" />
    <translation lang="HR" text="do" />
  </pass>
  <pass class="...test.notes.NoteIdPass">
    <clue class="...test.notes.NoteIdClue" key="to" />
  </pass>
</query-transition>

<query-transition id="t-multiple-note-id">
  <connect-to id="p-selected-notes" />
  <pass class="...pass.FixedKeyPass" key="id" />
  <pass class="...test.notes.NoteIdPass">
    <clue class="...test.notes.NoteIdClue" key="id" />
  </pass>
</query-transition>

<place id="p-selected-notes">
  <connect-to id="t-multiple-note-id" priority="H" />
  <connect-to id="t-delete-note" />
  <component class="...test.notes.NoteSelectionComp" hereditary="true" />
</place>

<path-transition id="t-image-id">
  <connect-to id="p-specific-image" />
  <pass class="...pass.CompositePass" delimiter="-" >
    <pass class="...pass.LangPass" >
      <translation lang="EN" text="image" />
      <translation lang="HR" text="slika" />
    </pass>
    <pass class="...test.notes.ImageIdPass">
      <clue class="...test.notes.ImageIdClue" />
    </pass>
  </pass>
</path-transition>

<place id="p-specific-image">
  <component class="...test.notes.ImageComp" />
</place>
</structure>

```

Slika 26: XML opis probnog sjedišta u UriGraph-u.

## 7.3 Primjeri rada poslužitelja

U nastavku su pokazani detalji rada poslužitelja u sljedećim primjerima zahtjeva:

- zahtjev samo sadržaja resursa u XML-u;
- zahtjev za stranicom u XHTML-u;
- zahtjev za izvršnim resursom (izvršavanje transakcije);
- zahtjev za slikom.

### 7.3.1 Jednostavan zahtjev sadržaja

Kao prvi primjer promotrit ćemo HTTP-ovski zahtjev GET na resurs identificiran s **"/note/1?content-only"**.

Odgovor ustrojnog modula sadrži sljedeće natuknice (u vrlo slobodnoj Javinoj sintaksi):

- **LanguagesClue( { english } )** – na engleskom;
- **ControlClue( "transform", "false" )** – samo sadržaj;
- **NoteClue( "note" )** – bilješka;
- **NoteIdClue( 1 )** – ID bilješke je 1.

Osim natuknica, odgovor sadrži i jednu komponentu bez svojih natuknica:

- **NoteComp()** – komponenta individualne bilješke.

Budući da nije navedena posebna akcija (npr. brisanje, ažuriranje), komponenta **NoteComp** inicira povrat sadržaja. Komponenta poziva odgovarajuće metode predmetnog sustava kako bi saznala podatke o bilješci br. 1 i na osnovu njih stvara XML-ov element **<note>** s odgovarajućim sadržajem.

Modul Web poslužitelja očitava iz natuknice **ControlClue** da sadržaj ne treba transformirati u XHTML. Budući da je sadržaj stvoren kao DOM stablo, serijalizira se u tekst i stavlja u tijelo HTTP-ovog odgovora 200 (*OK*) (Slika 27).

```
<?xml version="1.0" ?>
<page-content>
<note id="1">
  <heading>Education</heading>
  <body>I have never let my schooling interfere with my
education.</body>
  <author>Twain</author>
  <date>1905</date>
</note>
</page-content>
```

Slika 27: Tijelo HTTP-ovog odgovora probnog sjedišta na zahtjev **"/note/1?content-only"**.

### 7.3.2 Zahtjev za transformiranim odgovorom

Drugi primjer ima najjednostavniji zahtjev: GET nad naslovnicom "/".

Ustrojni odgovor sadrži samo jednu natuknicu:

- `LanguagesClue( { english, hrvatski } )` – oba jezika.

Odgovor sadrži i jednu komponentu s dvije natuknice:

- `NoteSelectionComp()` – komponenta odabira bilježaka;
- `orderBy( "date", "descending" )` – natuknica za najnovije bilješke;
- `ordNumRange( 1, 5 )` – natuknica za samo prvih pet.

Budući da nije navedena posebna akcija, komponenta `NoteSelectionComp` inicira povrat sadržaja. Stvara popis svih bilješki u predmetnom sustavu i prosljeđuje ih prvom objektu natuknice, koji sortira popis po datumu, od novijih prema starijima. Druga natuknica filtrira samo one s rednim brojem od 1 do 5. Konačno, stvara se XML-ov element `<note-list>` (Slika 28). Radi jednostavnosti, prepostavili smo da je problem višeiznačnosti jezika riješen tako da se engleski jezik podrazumijeva.

```
<note-list>
  <note id="6">
    <heading>Mathematics and reality</heading>
    <author>Einstein</author>
    <date>1975</date>
  </note>
  <note id="3">
    <heading>On Twain</heading>
    <author>Shaw</author>
    <date>1940</date>
  </note>
  <note id="4">
    <heading>Last words</heading>
    <author>Tolstoy</author>
    <date>1910</date>
  </note>
  <note id="1">
    <heading>Education</heading>
    <author>Twain</author>
    <date>1905</date>
  </note>
  <note id="2">
    <heading>Classics</heading>
    <author>Twain</author>
    <date>1900</date>
  </note>
</note-list>
```

Slika 28: XML-ov element koji predstavlja pet najnovijih bilježaka na sjedištu.

Budući da se ovaj odgovor transformira, u izvoru izgleda traži se XSLT-ov paket koji transformira tu komponentu. Prepostavimo da je u našem slučaju taj paket vrlo skroman, sastavljen od dva XSLT-ova predloška koji prikazuju podatke tablično (Slika 29).

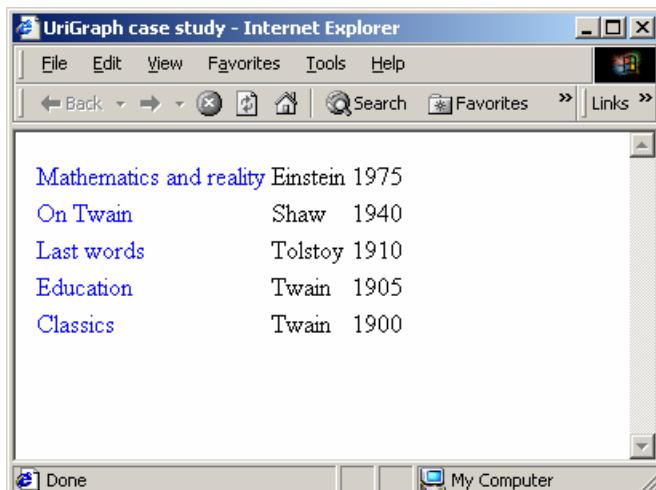
Identifikator bilješke se koristi kako bi se formirala hiperveza usidrena u naslovu bilješke koja pokazuje na odgovarajuću stranicu o bilješki.

```
<xsl:template match="/note-list">
  <table>
    <xsl:apply-templates />
  </table>
</xsl:template>

<xsl:template match="/note-list/note">
  <tr>
    <td>
      <a>
        <xsl:attribute name="href">
          /note/<xsl:value-of select="@id" />
        </xsl:attribute>
        <xsl:value-of select="heading" />
      </a>
    </td>
    <td><xsl:value-of select="author" /></td>
    <td><xsl:value-of select="date" /></td>
  </tr>
</xsl:template>
```

Slika 29: XSLT fragment koji definira predloške za transformaciju popisa bilježaka.

Budući da je to jedina komponenta resursa, transformirani dokument se slaže u odgovor 200 (*OK*) i preglednik ga prikazuje (Slika 30).



Slika 30: Prikaz naslovne stranice probnog sjedišta u pregledniku.

### 7.3.3 Zahtjev za izvršnim resursom

Ovo je primjer zahtjeva nad izvršnim resursom: "**POST /biljeska/stvorи**".

Ovakav zahtjev treba nositi u svom tijelu odgovarajuće podatke potrebne za stvaranje nove bilješke na dva jezika.

Odgovor ustrojnog modula sadrži sljedeće natuknice:

- `LanguagesClue( { hrvatski } )` – na hrvatskom;
- `NoteClue( "note" )` – bilješka;

- `NoteClue( "create" )` – stvori novu bilješku.

Odgovor sadrži i jednu komponentu bez svojih natuknica:

- `NoteComp()` – komponenta individualne bilješke.

Ovaj put je navedena akcija, pa komponenta `NoteComp` učitava podatke iz tijela HTTP-ovog zahtjeva i inicira preusmjeravanje. Nova poruka dobiva identifikator 7, pa je URI na koji se klijent usmjerava `"/biljeska/7"`. Statusni kôd preusmjeravanja je 303 (*See Other*) kao što je objašnjeno u 3.3.4, pa će preglednik prikazati stranicu sa sadržajem nove bilješke, bez da korisnik primijeti kako je to odgovor na drugi zahtjev.

### 7.3.4 Zahtjev za slikom

Resursi koji vraćaju sliku u našem studijskom slučaju su posebni zbog tri razloga:

1. obično su **pomoćni** resursi, koji se ne traže sami za sebe, nego samo kako bi bili dio prikaza drugog resursa, obično stranice;
2. poslužuju **gotove** entitete, jer slike obično ne zahtijevaju posebnu obradu (npr. XML-ove transformacije), nego se isporučuju u obliku zapisanom na poslužitelju;
3. entitet nije u obliku XML-ovog stabla, niti tekstualan, nego u svom posebnom formatu, tj. s njim se barata kao sa binarnim sadržajem.

Primjer zahtjeva za slikom može biti "**GET /slika-1**". Odgovor ustrojnog modula sadrži sljedeće natuknice:

- `LanguagesClue( { hrvatski } )` – na hrvatskom;
- `ImageIdClue( 1 )` – slika br. 1;

Odgovor sadrži i jednu komponentu bez svojih natuknica:

- `ImageComp()` – komponenta individualne slike.

Komponenta slike čita samo broj potrebne slike i inicira povrat gotovog, binarnog entiteta. Entitet je bez daljnjih provjera vraćen *listener-u* koji formulira odgovor 200 (*OK*).

Poseban repozitorij slika kojeg smo predvidjeli u ovom primjeru (od njega komponenta dobiva sadržaj slike) može biti izrađen na način da sadrži i više entiteta koji odgovaraju jednom resursu slike, ovisno npr. o jeziku, širini komunikacijskog pojasa i sl.

## 8 Zaključak

U ovom radu promotrili smo ključni dio u razvoju složenih Web usluga, ustroj Web sjedišta. Definirali smo sjedište kao skup Web resursa. Ustroj tog sjedišta definiramo tako da svakom resursu se pridružuje:

- identifikator, prema standardu *Uniform Resource Identifier-a*;
- identitet, ideja vlasnika sjedišta o tome što resurs predstavlja;
- položaj u odnosu na druge resurse sjedišta;
- sastav resursa koji čine modularne komponente.

U radu su istraženi postojeći postupci i modeli ustroja Web sjedišta. Posebna pažnja je posvećena najpopularnijem, klasičnom modelu ustroja koji se zasniva na datotečnom sustavu. Dodatno su obrađeni SGF, WebML, SiteBrain i iAS. Provedena je usporedna analiza opisanih modela s motrišta primjene u razvoju i održavanju Web usluga i aplikacija.

Na osnovu stečenog znanja i iskustva osmišljen je unaprijeđen model ustroja nazvan "UriGraph". UriGraph je prvenstveno model identifikacije resursa, tj. opisuje dizajn podprostora URI-ja na sjedištu. Iz tog modela se može prikupiti potrebna informacija i o identitetu, položaju i sastavu svakog resursa.

Kao praktična verifikacija rezultata istraživanja, programski je izvedena cijelokupna platforma u koju je ugrađena programska izvedba modela. Izrađen je i XML format za opis modela. U laboratorijskim uvjetima model je iskušan u više karakterističnih slučajeva, od kojih je jedan detaljno prikazan u radu.

Buduća istraživanja bi svakako trebala detaljnije proučiti primjenu UriGraph-a u razvoju različitih Web sjedišta. Praktične promjene nad modelom bi mogle sadržavati uvođenje konstrukcije podgrafa, ili brojanje i ograničenje broja prolaza kroz jedan prijelaz.

## Literatura

- [app92] Apple Computer: "*Macintosh Human Interface Guidelines*", Addison-Wesley, Reading, SAD, 1992.
- [bic96] Bichler, M., Nusser, S.: "*W3DT - The Structured Way of Developing WWW Sites*", Proceedings of the ECIS 1996, Lisbon, Portugal, 1996.
- [bon00] Bongio, A., Ceri, S., Fraternali, P., Maurino, A.: "*Modeling Data Entry and Operations in WebML*", Proceedings WebDB, p. 201-214, Dallas, SAD, 2000.
- [cer00] Ceri, Stefano; Piero Fraternali; Aldo Bongio: "*Web Modeling Language (WebML): a modeling language for designing Web sites*", 9<sup>th</sup> International World Wide Web Conference, svibanj 2000, Amsterdam, Nizozemska,  
<http://www9.org/w9cdrom/177/177.html>
- [cha01] Champin, Pierre-Antoine; Jérôme Euzenat; Alain Mille: "*Why URLs are good URIs, and why they are not*", svibanj 2001., <http://www710.univ-lyon1.fr/~champin/urls/>
- [lie98] Liechti, Olivier; Mark J. Sifer; Tadao Ichikawaielsen: "*Structured graph format: XML metadata for describing Web site structure*", 7<sup>th</sup> International World Wide Web Conference, kolovoz 1998, Brisbane, Australija,  
<http://www7.scu.edu.au/programme/fullpapers/1853/com1853.htm>
- [nie00] Nielsen, Jakob: "*Designing Web Usability: The Practice of Simplicity*", New Riders Publishing, Indianapolis, 2000., ISBN 1-56205-810-X,  
<http://useit.com/jakob/webusability/>
- [nie96] Nielsen, Jakob: "*Death of file system*", <http://www.useit.com/papers/filedeath.html>
- [nie98] Nielsen, Jakob: "*Fighting linkrot*", <http://www.useit.com/alertbox/980614.html>
- [nie99] Nielsen, Jakob: "*URL as UT*", Alertbox, ožujak 1999.,  
<http://www.useit.com/alertbox/990321.html>

- [orapc] Oracle Portal Center, <http://portalstudio.oracle.com>
- [ora9ias] Oracle9i Application Server, <http://www.oracle.com/ip/deploy/ias>
- [per98] Perkins, Ron: "*Web Navigation vs. Desktop Application Navigation*", CHI 98 Workshop on Web Navigation, kolovoz 1998, Los Angeles, SAD
- [rfc2141] Moat, R.: "*URN Syntax*", RFC 2141, svibanj 1997., <http://www.ietf.org/rfc/rfc2141.txt>
- [rfc2396] Berners-Lee, T.; R. Fielding; L. Masinter: "*Uniform Resource Identifiers (URI): Generic Syntax and Semantics*", RFC 2396, kolovoz 1998., <http://www.ietf.org/rfc/rfc2396.txt>
- [rfc2616] Fielding, R.; J. C. Mogul; H. Frystyk; L. Masinter; P. Leach; T. Berners-Lee: "*Hypertext Transfer Protocol -- HTTP/1.1*", RFC 2616, srpanj 1999., <http://www.ietf.org/rfc/rfc2616.txt>
- [ros99] Rossi, Gustavo; Schwabe, Daniel; Lyrdet, Fernando: "*Web application models are more than conceptual models*", Proceedings of ER'99 (Paris, France, November 1999), Springer, 239-252. <http://citeseer.nj.nec.com/rossi99web.html>
- [ros01] Rossi, G., D. Schwabe, L. Esmeraldo, F. Lyardet, "*Engineering Web Applications for Reuse*", IEEE Multimedia, Vol. 8, No. 1, 20-31, siječanj 2001.
- [smi96] Smilowitz, Elissa D.: "*Do Metaphors Make Web Browsers Easier to Use?*", Designing for the Web: Practices and Reflections, Microsoft Campus, listopad 1996. <http://www.baddesigns.com/mswebcnf.htm>
- [tbl92] Berners-Lee, T.: "*Style guide for online hypertext*", <http://www.w3.org/Provider/Style/>
- [tbl96] Berners-Lee, T.: "*Universal Resource Identifiers – Axioms of Web Architecture*", *W3C Design Issues*, 19.12.1996., <http://www.w3.org/DesignIssues/Axioms.html>
- [umlrr] Rational Rose UML Resource Center, <http://www.rational.com/uml/>

- [wca99] Lavoie, Brian; Henrik Frystyk Nielsen (editors): "*Web Characterization Terminology & Definitions Sheet*", radni nacrt W3C-a, 24. 5. 1999.,  
<http://www.w3.org/1999/05/WCA-terms/01>
- [xml00] Bray, T.; J. Paoli; C. M. Sperberg-McQueen; E. Maler (editors): "*Extensible Markup Language (XML) 1.0 (Second Edition)*", preporuka W3C-a, listopad 2000.,  
<http://www.w3.org/TR/2000/REC-xml-20001006>
- [xslt10] Clark, James: "*XSL Transformations (XSLT) Version 1.0*", preporuka W3C-a, studeni 1999., <http://www.w3.org/TR/xslt>

## **Životopis autora**

Rođen sam 27. svibnja 1975. godine u Tučepima. Srednju školu, matematičko-informatički smjer završio sam 1993. godine u Makarskoj. Iste godine upisujem Fakultet elektrotehnike i računarstva. Diplomirao sam 1998. godine na smjeru Telekomunikacije i informatika radom pod nazivom "Racionalizacija razvoja multimedijskih aplikacija". Iste godine zapošljavam se na Zavodu za telekomunikacije u svojstvu zavodskog suradnika i upisujem poslijediplomski magistarski studij. Radim na projektima suradnje s privredom na razvoju Web aplikacija u području elektroničkog poslovanja. Sudjelujem u nastavi na predmetima "Telematičke usluge" i "Seminar".

## Kratki sažetak

Ovaj rad definira Web sjedište kao kolekciju Web resursa, uključujući Web stranice. Svakom resursu se pridružuje (1) identifikator u sintaksi *Uniform Resource Identifier-a*; (2) identitet, ideja vlasnika sjedišta o tome što resurs predstavlja; (3) položaj u odnosu na druge resurse sjedišta i (4) sastav resursa koji čine modularne komponente. Skup ove četiri osobine za svaki resurs u sjedištu čini ustroj Web sjedišta. Rad definira model nazivan UriGraph za opisivanje identifikatora resursa sjedišta i preko njega i ostale tri osobine, što ga čini potpunim modelom ustroja Web sjedišta. Model se može predstaviti grafički ili kao XML dokument koji odgovara zadanoj shemi. Rad opisuje i izvedbu modela u Javi, integraciju u pokušni Web poslužitelj i jednostavni studijski slučaj.

## Short summary

### *Modeling of WWW Site Structure*

This thesis defines Web site as a collection of Web resources, including Web pages. Each resource is assigned (1) an identifier in the syntax of Uniform Resource Identifier; (2) an identity, site owner's notion of what a resource is representing; (3) a position relative to other resources in the site; and (4) a composition of the resource consisting of modular components. The set of these four qualities for every resource in the site makes the site structure. The thesis introduces the model called UriGraph for describing identifiers for site's resources and through them the other three qualities, making it a complete Web site structure model. The model can be represented graphically or as an XML document conforming to the specified schema. The thesis describes the implementation of the model in Java, integration in an experimental Web server, and a simple case study.

## Ključne riječi

World Wide Web

razvoj

ustroj

Web sjedište

Web resurs

Web aplikacija

XML

URI

UriGraph

## Keywords

World Wide Web

development

structure

Web site

Web resource

Web application

XML

URI

UriGraph