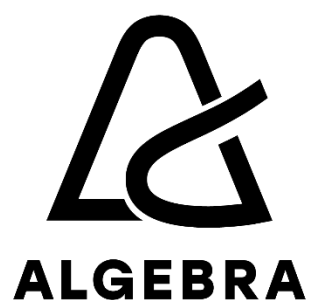




Priručnik

Interoperabilnost informacijskih sustava

Aleksander Radovan
Branko Mihaljević

Zagreb, 2020.

autori:

v. pred. Aleksander Radovan, dipl. ing.
dr. sc. Branko Mihaljević, dipl. ing.

urednica:

Barbara Ćosić

naslov:

Interoperabilnost informacijskih sustava

stručni recenzenti:

Milan Korać, dipl. ing.
dr. sc. Branko Mihaljević, dipl. ing.
mr. sc. Igor Ljubi, dipl. ing.
doc. dr. sc. Martin Žagar, dipl. ing., EMBA

lektorica:

Iva Lednicki

grafički urednik:

Krešimir Pletikosa, ACE

nakladnik:

Algebra d.o.o., 2020.

za nakladnika:

doc. dr. sc. Mislav Balković

mjesto i godina izdavanja:

Zagreb, 2020.

Sva prava pridržana. Niti jedan dio ove knjige ne smije se reproducirati ili prenositi u bilo kojem obliku, niti na koji način. Zabranjeno je svako kopiranje, citiranje te upotreba knjige u javnim i privatnim edukacijskim organizacijama u svrhu organiziranih školovanja, a bez pisanog odobrenja nositelja autorskih prava.

Copyright © Algebra d.o.o.

CIP zapis dostupan u računalnom katalogu
Nacionalne i sveučilišne knjižnice u Zagrebu pod brojem

ISBN 978-953-322-369-8

Sadržaj

1. Poglavlje: Uvod u interoperabilnost.....	5
1.1 Uvod	6
1.2 Definicije	7
1.3 Elektroničko poslovanje.....	14
1.4 Čimbenici, koristi, razine i vrste interoperabilnosti	18
1.5 Područja usklađivanja interoperabilnosti	26
1.6 Pravni okvir vezan uz interoperabilnost u Hrvatskoj	30
1.7 Strategije interoperabilnosti.....	40
1.8 Program Interoperabilna rješenja za europsku javnu upravu ISA	55
1.9 Okviri za interoperabilnost	56
2. Poglavlje: Norme komunikacije i razmjene podataka.....	69
2.1 Osnovni pojmovi normizacije	70
2.2 Tipovi normi.....	74
2.3 Načini izrade norma.....	77
2.4 Važnija normirna tijela.....	78
2.5 Norme za razmjenu elektroničkih podataka.....	93
2.6 Norme u elektroničkom poslovanju	96
2.7 Norme za zapis znakova i hrvatski grafemi.....	106
3. Poglavlje: Označni jezici.....	119
3.1 Uvod u označne jezike	120
3.2 Označni jezik SGML.....	123
3.3 Označni jezik XML	124
3.4 YAML.....	132
3.5 JSON.....	134
3.6 Validacije dokumenata	139
4. Poglavlje: Rukovanje i upravljanje podacima u obliku XML	151
4.1 Rukovanje podacima u obliku XML.....	152
4.2 Poznatije primjene označnih jezika	170
5. Poglavlje: Softverske arhitekture, tehnologije i protokoli	187
5.1 Općenito o uslugama, definicije i stanja usluga	188
5.2 Arhitekture raspodijeljenih sustava.....	190
5.3 Arhitekture raspodijeljenih sustava.....	201
5.4 Interoperabilnost kod višeslojnih arhitektura	203
5.5 Tehnologije izvedbe interoperabilnosti	212
6. Poglavlje: Arhitektura orijentirana uslugama (SOA).....	219
6.1 Osnovna načela orijentiranosti usluzi.....	220

6.2	Poslovanje i orijentiranost usluzi	221
6.3	Sprega	227
6.4	Otkrivanje usluga	229
6.5	Slaganje usluga	238
6.6	Upravljanje i nadzor nad sustavom	240
6.7	Primjeri sustava orijentiranih uslugama	242
7.	Poglavlje: Usluge Web (Web Services).....	253
7.1	Uvod u usluge Web.....	254
7.2	Arhitektura usluga Web.....	256
7.3	Simple Object Access Protocol – SOAP	261
7.4	Web Services Description Language	268
7.5	REST API	273
7.6	Razvoj standarda usluga Web.....	280
8.	Poglavlje: Sigurnost interoperabilnosti i usluga.....	287
8.1	Osnovni pojmovi sigurnosti	288
8.2	Korištenje kriptografije	289
8.3	Elektronički potpis	292
8.4	Elektronička (digitalna) oмотnica	298
8.5	Napadi.....	300
8.6	Sigurnost usluga.....	302
9.	Poglavlje: Primjer interoperabilnosti	309
9.1	Primjer implementacije interoperabilnosti – NIAS autentikacija	310

1. Poglavlje: Uvod u interoperabilnost

U ovom poglavlju naučit ćete:

- ✓ razumijevanje pojmova interoperabilnost, semantika i ontologija
- ✓ razloge, koristi, ciljeve, čimbenike, razine, vrste i područja interoperabilnosti
- ✓ tehnologije i usluge interoperabilnosti
- ✓ osnove pravnog okvira vezanog uz interoperabilnost
- ✓ osnove Hrvatskog okvira za interoperabilnost (HROI)

1.1 Uvod

Cilj ovoga kolegija je upoznati studente s raznolikim oblicima interoperabilnosti prisutnim u modernom računarstvu. Već na početku bismo htjeli napomenuti da se u sadržaju kolegija obrađuje mnogo različitih pojmova, koncepata i aktualnih tehnologija te će se postupnim upoznavanjem s njima doći do konkretne upotrebe tih tehnologija i raznolikih izvedbi njihove međusobne suradnje, odnosno interoperabilnosti. Neka će se načela, metode, mehanizme i tehnologije koje su rasprostranjenije, popularnije ili bolje normirane obraditi potanje, a druge ćemo samo ukratko objasniti. No, bitno je razumjeti da kako bi određeni sustavi, aplikacije i tehnologije, ali i posljedično čitavi sustavi organizacija, odnosno osobe u tim organizacijama mogle surađivati, prvo treba potanje upoznati obje strane koje žele surađivati, različite aspekte njihovog poslovanja i poslovnih procesa, uočiti mogućnosti međusobne suradnje, prepoznati njihove sličnosti i razlike te na kraju uvažanjem svih tih značajki ponuditi rješenje u obliku interoperabilnih aplikacija, usluga ili sustava. Iz tog će se razloga i u ovom priručniku prvo objašnjavati opće pojmove, zatim pregledati niz raznolikih tehnologija i na kraju prikazati suradnja dviju ili više strana korištenjem određenih protokola, metoda i tehnika na primjerima.

Priručnik je podijeljen na više dijelova. U prvom se dijelu objašnjava što je interoperabilnost, zašto je korisna, koji joj je cilj te na kojim razinama i područjima djeluje. Važno je razumjeti da interoperabilnost nije samo tehnički pojam, već mnogo dublje zadire u suradnju dviju strana te sadržava mnoge organizacijske, pravne, socijalne i druge aspekte. U drugom dijelu ulazi se u problematiku komunikacije i razmjene podataka, s posebnim naglaskom na norme. Prikazani su najvažniji načini dolaska do komunikacije kao preduvjeta suradnje te se navode najbitnije hrvatske i međunarodne norme i normizacijska tijela. Zatim se objašnjavaju pojmovi iz interoperabilnosti u računarstvu vezani uz jezik i semantiku. U trećem se dijelu obrađuju označni jezici, s posebnim naglaskom na jezik XML, oblik zapisa podataka JSON i jezik za serijalizaciju podataka YAML. Iako ste se s jezikom XML već susreli u kolegiju „Standardi u primjeni internetske tehnologije“, ovdje mu se pristupa s razine komunikacije i interoperabilnosti te se vaša znanja proširuju drugim načinom pogleda na tehnologije koje prate jezik XML, kao i drugim načinima zapisa u obliku YAML i JSON. Mogućnost korištenja jezika XML za manipulaciju, prijenos i prikaz podataka obrađuje se u četvrtom dijelu, a navode se i najbitnije upotrebe jezika XML. Peti dio definira najpoznatije i najčešće korištene softverske arhitekture, tehnologije i protokole vezane uz interoperabilnost.

Objašnjavaju se i potrebni pojmovi i tehnologije potrebni za razumijevanje raznolikih postojećih usluga, a prikazuju se i drugi mehanizmi koje je moguće usporediti s uslugama. U šestom se dijelu opisuje arhitektura zasnovana na uslugama (SOA) definira se pojam usluge u računarstvu te objašnjava stanja i vrste usluga. Objašnjava se kako se uslugama pristupa i koji su najpoznatiji primjeri takvih usluga. Sedmi dio donosi najpoznatiji skup tehnologija kada se govori o uslugama, a to su Web Services i REST. U nastavku se obrađuju pojmovi WSDL, SOAP i UDDI. U osmom se dijelu objašnjavaju različiti pojmovi vezani uz sigurnost kod interoperabilnosti, komunikacije na Internetu i usluga. S nekim ste se od navedenih načela već susreli na kolegiju „Sigurnost informacijskih sustava“, a ovdje se naglašavaju sigurnosni aspekti vezani uz komunikaciju na Internetu i interoperabilnost. Određene se tehnike i metode proširuju dodatnim razumijevanjem potrebnim za ostvarivanje sigurnih interoperabilnih usluga. Navode se i osnovni oblici napada na sigurnost, najčešće korištene metode zaštite te se referencira na pravne aspekte i norme u Hrvatskoj, zajedno sa stvarnom praksom korištenja. U devetom poglavlju dani su različiti primjeri interoperabilnosti iz prakse.

U prvom ćete se poglavlju upoznati s raznim aspektima interoperabilnosti, ciljevima, razinama, vrstama i područjima kako bi u sljedećim poglavljima ovog priručnika došli do toga kako

tehnički izvesti interoperabilnost. Krenut ćemo od definicija i razumijevanja osnovnih pojmova te podjela potrebnih za razumijevanje interoperabilnosti i njene svrhe.

1.2 Definicije

U ovom će se potpoglavlju proći kroz niz definicija i objašnjenja određenih pojmova potrebnih za razumijevanje ovog kolegija, kao što su interoperabilnost, semantika, ontologija i drugi.

1.2.1 Interoperabilnost

Za početak trebamo definirati što znači pojam **interoperabilnost** (engl. *interoperability*). Budući da je taj pojam dosta apstraktan i katkad nejasan, u ovom će se potpoglavlju pokušati razjasniti njegov smisao i šire značenje. Predlažemo zajedničko proučavanje definicije pojma *interoperabilnost*. Pritom nemojte smetnuti s uma da je značenje pojma interoperabilnost za svaku osobu ipak većinom definirano njenim vlastitim iskustvom, uvjetovano okruženjem iz kojeg dolazi te vlastitim preferencijama, jer interoperabilnost je ipak apstraktni pojam.



Ako postavite pitanje osobama iz različitih struka što podrazumijevaju pod pojmom interoperabilnosti, svaka će vam dati prilično različito objašnjenje tog pojma. Primjerice, filozof će pričati o razumijevanju dvaju subjekata, bankar o razmjeni financijskih podataka, jezikoslovac (lingvist) o sličnosti jezika potrebnoj da se dvije osobe sporazumiju, psiholog će pronalaziti kompatibilnosti dvije osobe u nekom odnosu, a poslovni čovjek o suradnji dviju tvrtki i prenošenju poslovnih informacija između dviju strana. Pokušate li isto pitanje postaviti djetetu, najvjerojatnije nećete dobiti nikakav odgovor, jer se ono, osim što često ni ne poznaje tu riječ, još nikad nije susrelo s takvim oblikom apstraktnog pojma, odnosno nema iskustva koje bi oblikovalo taj pojam. Razlog je u tome što je interoperabilnost, kao i niz drugih sličnih riječi, potpuno apstraktni pojam čije se razumijevanje stječe vlastitim iskustvom i predstavljen nam je vlastitom percepcijom. Stoga pokušajte sami definirati interoperabilnost, uzmite nekoliko minuta i pokušajte objasniti što je za vas interoperabilnost. Kada završite, razmislite kako bi vaše razumijevanje i objašnjenje pojma interoperabilnost objasnili djetetu.

Pogledamo li rječnike, svaki će rječnik imati malo drukčije objašnjenje pojma interoperabilnost u općem smislu. Neki konzervativniji izvori kažu da je to „mogućnost sustava (npr. naoružanja) da radi s ili koristi dijelove drugog sustava“¹, a jedna od definicija kaže da je to „mogućnost sustava, jedinica ili snaga da pružaju i primaju usluge drugih sustava, jedinica ili snaga i da ih koriste tako da mogu zajedno djelovati“². Očigledno je osnovni smisao pojma interoperabilnost proizašao iz vojne upotrebe, pa je i definicija preuzeta iz vojnog vokabulara gdje se pojam prvotno upotrebljavao³.

¹ Interoperability, Merriam-Webster Dictionary, <http://www.merriam-webster.com/dictionary/interoperability>

² Interoperability, TheFreeDictionary, Farlex, <http://www.thefreedictionary.com/interoperability>

³ The Oxford Essential Dictionary of the U.S. Military

U kontekstu računarstva i informatike, neki od poznatijih engleskih rječnika kažu da je interoperabilnost „**stupanj u kojem se dva proizvoda, programa i sl. mogu zajedno koristiti, ili kvaliteta mogućnosti da se koriste zajedno**“⁴, a neki kao definiraju interoperabilnost kao izvedenicu pridjeva **interoperabilan** (engl. *interoperable*), koji pak definiraju kao „**mogućnost razmjene i korištenja informacija**“⁵.



Kako bismo bolje razumjeli pojam interoperabilnosti, rastavit ćemo ga na dijelove. Prvi dio, prefiks „inter“ dolazi od riječi latinskog jezika „in“ i „terra“, što doslovno znači „u zemlji“ (imalo veze s ukopom), ali se ovdje koristi u smislu „između“, odnosno „u sredini“, „tijekom“, „zajednički“, „recipročni“ i slično. Zanimljivo je da se isti prefiks koristi se i kod pojma „internet“. Pojam „operabilnost“ (engl. *operability*) rastavlja se na riječi „djelovati“ (engl. *operate*) i „mogućnost“ (engl. *ability*) te u načelu ne postoji u rječnicima našeg jezika, no postoji srodna riječ „operabilan“ koji označava onog koji radi ili djeluje. Stoga se taj drugi dio može definirati kao mogućnost da određena oprema ili sustav funkcionira tako da zajednički odrađuje određeni zadatak.

Kada složimo pojmove „inter“ i „operabilnost“, dolazimo do niza uobičajenih korištenih hrvatskih prijevoda pojma interoperabilnost, kao „**sudjelatnost**“, „**međudjelovanje**“, „**mogućnost međusobnog djelovanja**“, „**mogućnost zajedničkog rada**“, „**mogućnost uzajamnog funkcioniranja**“ i slično, a katkad se koristi i „**interoperativnost**“.



U stručnim krugovima se najčešće koristi doslovni prijevod „interoperabilnost“, no znanstvena zajednica pokušava uvesti i druge primjerenije oblike hrvatskih prijevoda izvornoga engleskog termina „interoperability“, pa se i gore navedeni termini znaju koristiti kao sinonimi pojma interoperabilnost. Iako većina rječnika hrvatskog jezika ne sadrži riječ „interoperabilnost“, u smislu sve češćeg uvođenja engleskih riječi u hrvatski standardni jezik kao posljedice korištenja u praksi, očekujemo da će se ovo promijeniti, a trenutno je baš riječ „interoperabilnost“ najčešći prijevod na internetskim stranicama na hrvatskom jeziku⁶, kao i najčešći izbor pri automatskom prevođenju.

Kako bi uže definirali ovaj apstraktan pojam u nama zanimljivom području, možemo pogledati niz raznolikih **definicija interoperabilnosti u računalnom smislu**, odnosno u polju računarstva, računarske znanosti ili inženjerstva te informatike, a ovdje ćemo navesti neke najpoznatije i označiti bitne pojmove:

„*Interoperabilnost je mogućnost dva ili više sustava ili komponente da **razmjenjuju** informacije i **koriste** razmijenjene informacije.*“⁷

⁴ Interoperability Cambridge Business English Dictionary, <http://dictionary.cambridge.org/dictionary/business-english/interoperability>

⁵ Interoperable, Oxford Dictionaries, <http://www.oxforddictionaries.com/definition/interoperable>

⁶ Pretraživač Google daje cca 256.000 rezultata za pretragu pojma „interoperabilnost“ na hrvatskom jeziku na datum 1.1.2020.

⁷ Interoperability, IEEE Standard Computer Dictionary, 1990.

„Interoperabilnost je mogućnost **različitih vrsta** računala, mreža, operacijskih sustava i aplikacija da učinkovito **surađuju bez prethodne komunikacije**, na način da razmjenjuju informacije na koristan i suvisao način.“⁸

„Interoperabilnost je sposobnost **raznolikih sustava i organizacija** da zajedno **surađuju**.“⁹

„Interoperabilnost je mogućnost komunikacije i prijenosa podataka između funkcionalnih jedinica koje se izvode na **različitim platformama, implementiranim u različitim tehnologijama**, korištenjem industrijskih **normi** ili široko prihvaćenih opisa podataka i komunikacijskih protokola.“¹⁰

„Interoperabilnost je mogućnost **komunikacije, izvršavanja** programa ili **razmjene** podataka između različitih funkcijskih jedinica na način koji zahtijeva **vrlo malo znanja korisnika** o jedinstvenim karakteristikama tih jedinica.“¹¹

„Interoperabilnost je **sposobnost zasebnih i različitih organizacija** da međusobno djeluju u smjeru zajednički korisnih i dogovorenih ciljeva, što uključuje **razmjenu podataka, informacija i znanja** kroz poslovne procese koje podržavaju i **korištenje informacijsko-komunikacijskih tehnoloških sustava**.“

Zadnja navedena definicija interoperabilnosti je iz Hrvatskog okvira za interoperabilnost (HROI)¹², usuglašenog i s Europskim okvirom za interoperabilnost (EIF). Prema ovoj izjavi interoperabilnost se postiže **dogovorom organizacija** o zajedničkim ciljevima i interesima te **konsenzusom** na razini upravljanja, a provodi se **usklađivanjem** u relevantnim poslovnim područjima te **praktičnim zajedničkim djelovanjem** kroz razmjenu podataka, informacija i znanja u poslovnim procesima, dok se krajnji ciljevi u pravilu postižu **pružanjem usluga**. Pritom se interoperabilnost ne smije poistovjetiti s integracijom, kompatibilnošću ili prilagodljivošću jer su to različiti pojmovi.



Ako iz svih ovih definicija izuzmemo tehnološki aspekt, vidljivo je da se interoperabilnost u povijesti postizala i na načine koji nužno ne uključuju tehnologiju. Stoljećima se odvijao određen oblik poslovanja bez tehnologije, a začeci moderne tehnologije kakvu danas poznajemo mjere se tek u desetljećima. Nekad se poslovalo uporabom određenih metoda i tehnika koje su uključivale zapise na papiru, pisanje perom ili olovkom, tiskanje ili štampanje, potpisivanje, pečatiranje i stavljanje žiga na dokumente. Prijenos dokumenata bio je ostvaren raznim oblicima dostave i pošte, od glasnika i golubova, preko poštanskih kočija, do poštanskih ureda. Poslovanje u smislu ugovaranja i robno-novčane razmjene obično se odvijalo kada su sudionici bili u neposrednoj fizičkoj blizini te su mogli i usmeno komunicirati, a kada su bili udaljeni, uz neke oblike pismene komunikacije. Tek korištenje telefona donosi prve oblike interoperabilnosti na velike udaljenosti, a sve se dodatno pospješuje uvođenjem teleksa i telefaksa. Danas, u doba sveprisutnih računala, elektroničkih telekomunikacija i elektroničkog poslovanja, smisao interoperabilnosti potpuno se mijenja i značajno dobiva na važnosti. Interoperabilnost koja se objašnjava u ovom priručniku podrazumijeva napredne i moderne oblike poslovanja i komunikacije.

⁸ Interoperability, DCMI Glossary, Dublin Core Metadata Initiative, <http://dublincore.org/documents/usageguide/glossary.shtml>

⁹ Interoperability, AFUL, <http://interoperability-definition.info/en/>

¹⁰ Application Interoperability: MS.NET and J2EE, Microsoft Press, 2003.

¹¹ Interoperability, ISO/IEC 2382-01, Information Technology Vocabulary, Fundamental Terms

¹² Odrednice Hrvatskog okvira za interoperabilnost, ver. 1.0, Središnji državni ured za e-Hrvatsku, 24.6.2010.

Svrha i cilj interoperabilnosti je **povećanje učinkovitosti** koja ima različite utjecaje na poslovanje. Najočiti su **brzina, nepotrebnost fizičke blizine i poslovanje na velike udaljenosti**, te **automatizacija procesa** i konačna posljedica – **postizanje poslovanja tamo gdje prije nije bilo moguće ili je bilo znatno otežano uz smanjenje troškova poslovanja**.

Osnovni **preduvjeti** moderne interoperabilnosti su razni oblici elektroničkih komunikacija, najčešće kao infrastrukture u obliku elektroničke komunikacijske mreže, računalni sustavi za poslovanje u kojima se ističu sustavi za izradu i pohranu strukturiranih podataka o poslovanju uporabom novih tehnologija, metoda i mehanizama transformacija, slanja i primanja te sigurne pohrane podataka. Svi ovi sustavi, protokoli i tehnologije koriste informacije u obliku elektronički zapisanih podataka, pa se može zaključiti da moderna interoperabilnost zapravo počiva na naprednoj tehnologiji.

1.2.2 Sintaksa, semantika i ontologija

U ovom ćemo potpoglavlju definirati još neke pojmove vezane uz značenje pojedinog podatka ili informacije u cjelini. Naime, da bi dvije strane mogle komunicirati, osim što moraju govoriti istim jezikom, one moraju za pojmove iz tog jezika imati isto značenje. Primjerice, ako dvije osobe komuniciraju istim jezikom, a jedna nikad nije vidjela automobil, onda će teško shvatiti što joj druga pokušava reći koristeći riječ „automobil“. Jednako tako, ako dvije osobe komuniciraju istim jezikom, obje znaju riječ „jabuka“, ali jedna osoba nikad nije vidjela crvenu jabuku, već samo zelene, tada druga osoba koja govori o boji nekog predmeta koji je „*crven poput jabuke*“ neće biti shvaćena. Sličan je problem kada ista riječ jednoj osobi znači jedno, a drugoj ima potpuno drugo značenje, kao što je primjer s riječi „karta“ koja može imati višestruko značenje (zemljopisna, igrača, astrološka, za prijevoz i sl.). Kroz ovakve primjere dolazimo do problematike načina komuniciranja koji se zasnivaju na sintaksi i semantici.

Iako ste se već sigurno susreli s pojmom sintakse, ovdje ćemo još jednom razjasniti značenje tog pojma. **Sintaksa** dolazi od grčke riječi „sintaksis“, koja se dijeli na „sin“, što znači „zajedno“, i „taksis“, što znači „poredak“. U jezičnom smislu sintaksa označava pravila i načela konstruiranja rečenica. U računarstvu se sintaksa najčešće veže uz programske i druge jezike, gdje označava **skup pravila koja definiraju kombinacije znakova**. Te se kombinacije znakova zapisane programskim ili drugim jezikom u nekom dokumentu (datoteci), te se, ako su ispravno strukturirane, nazivaju programom ili skriptom. Kada govorimo o sintaksi nekog **podatka**, onda se to odnosi na **propisanu strukturu kako taj podatak treba biti zapisan i pohranjen**. Isto tako sintaksa **protokola** označava **propisanu strukturu određenog načina razmjene podataka**.

Sljedeći bitan pojam je **semantika**, koja dolazi od grčke riječi „semantikos“, što znači značajan, odnosno „onaj koji daje znakove“, jer u grčkom „sema“ označava znak. Pojam semantika odnosi se na aspekte značenja izražene u jeziku ili nekom drugom obliku. Semantika postoji i kao nekoliko grana znanosti, konkretno grana lingvistike, grana psihologije i grana etike, a nas ovdje zanima semantika u smislu računarstva. Semantika u računarstvu se ponajprije odnosi na programske i druge jezike, gdje označava **značenje pojedinih izraza**. No, postoji i drugi smisao riječi semantika, a to je **značenje pojedinih podatkovnih struktura** koje se koriste za **definiranje određenih oblika zapisa podataka** i predstavljaju **sadržajni oblik informacije**.



Sintaksa programskog jezika definira da je program ispravno strukturiran (engl. *structured*) ili dobro oblikovan (engl. *well-formed*), ali ne govori ništa o značenju pojedinih dijelova programa ni njegovom značenju kao cjeline. Semantika definira značenje, a sintaksa ne. Može se zaključiti da sintaktički ispravni programi ne moraju biti i semantički ispravni, jer ispravna struktura ne znači i smislenu značenje. Primjer iz jezika za sintaktički ispravnu, ali semantički besmislenu rečenicu bio bi: „*Ovaj kolegij na svijet donosi ružičaste usluge.*“, koja samo u pjesničkom ili prenesenom smislu može imati neko značenje.

Sljedeći važan pojam je **ontologija** koja se također zasniva na grčkoj riječi, gdje je „on“ particip glagola biti, tj. biće, a „logos“ označava riječ ili učenje, pa se ontologija može nazvati općom naukom o biću. Iako je bitno napomenuti da se danas riječ ontologija primarno koristi u filozofskom značenju, gdje ontologija znači granu metafizike koja se bavi prirodom bića, bivanja ili bitka, ovdje nas zanima njeno značenje u kontekstu računarstva i interoperabilnosti. U računarstvu i informacijskim znanostima ontologija se definira kao **formalna reprezentacija znanja korištenjem skupa koncepata unutar neke domene i odnosima između tih koncepata**. U praksi to znači da se ontologija koristi za razumijevanje objekata u određenoj domeni. Kao oblik reprezentacije znanja o nekom dijelu stvarnosti ontologije se koriste u umjetnoj inteligenciji, semantičkom webu, softverskom inženjerstvu i drugim granama računarskih znanosti. Za bolje razumijevanje ovog apstraktnog pojma pogledajmo nekoliko definicija:

„*Ontologija određuje dijeljeni vokabular koji se koristi za modeliranje domene, odnosno tipova objekata i koncepata, njihovih značajki i relacija.*“¹³

„*Ontologija je hijerarhijska struktura znanja o stvarima ostvarena kategoriziranjem prema njihovim značajkama (ili barem relevantnim i/ili kognitivnim) kvalitetama.*“¹⁴

„*U računarstvu i informacijskoj znanosti, ontologija formalno predstavlja znanje kao skup koncepata unutar domene i veze (odnose) između tih koncepata. Može se koristiti za modeliranje domene i podršku rasuđivanju (razumijevanju) koncepata.*“¹⁵

„*U kontekstu računalnih i informacijskih znanosti, ontologija definira skup reprezentacijskih primitiva koji modeliraju domenu znanja ili komunikaciju.*“¹⁶

U prethodnim se definicijama spominju tipovi objekata, koncepti i na kraju reprezentacijski primitivi, koji se mogu definirati kao objekti koji sadržavaju informaciju o svom značenju i ograničenja za logički konzistentnu primjenu. **Osnovni reprezentacijski primitivi su klase** (razredi), **atributi** (značajke) i **odnosi** između članova razreda.

Promotrit ćemo **primjer ontologije kod baza podataka**. Ontologija je razina apstrakcije podatkovnih modela, analogna hijerarhijskom ili relacijskom modelu baze podataka, ali namijenjena modeliranju znanja o objektima, atributima i odnosima. S obzirom na to da se

¹³ Ontology, Arvidsson, F. i Flycht-Eriksson, A., Ontologies I, <http://www.ida.liu.se/~janma/SemWeb/Slides/ontologies1.pdf>

¹⁴ Ontology, Free On-Line Dictionary of Computing, <http://foldoc.org/ontology>

¹⁵ Ontology, Wikipedia, http://en.wikipedia.org/wiki/Ontology_%28information_science%29

¹⁶ Ontology, Encyclopedia of Database Systems, Liu, L. i Özsu T. (izd.), Springer-Verlag, 2009. <http://tomgruber.org/writing/ontology-definition-2007.htm>

ontologije uobičajeno specificiraju jezicima koji dopuštaju razdvajanje apstrakcije od podatkovnih struktura i načina implementacije, ontologije su bliže logici nego jezicima za modeliranje baze podataka. Zato se kaže da su ontologije modeli „na semantičkoj razini“, dok su sheme baza podataka modeli „na logičkoj ili fizičkoj razini“.



Budući da su ontologije neovisne o nižim, fizičkim razinama modela baza podataka, mogu se koristiti za integriranje heterogenih baza podataka ili, općenito, bilo kakvih izvora podataka, kako bi omogućile interoperabilnost između inače nespojivih podataka, specificiranjem sučelja neovisnih usluga temeljenih na znanju.

Komponente ontologije uključuju:

- **klase (razrede)** – skupove, koncepte, kolekcije, tipove objekata, vrste stvari,
- **individue** – instancije ili objekte,
- **atribute** – aspekte, značajke, osobine, karakteristike ili parametre objekata (i razreda tj. klasa),
- **odnose** – načine kako se klase tj. razredi i individue odnose jedno prema drugima,
- **funkcijske pojmove** – složene strukture izvedene iz određenih odnosa koje se mogu koristiti na mjestu termina u izjavi,
- **ograničenja** – formalno izjavljeni opisi što mora biti istina kako bi se tvrdnja prihvatila,
- **pravila** – izjave u obliku „ako – onda“ koje opisuju logičke zaključke izvedene iz tvrdnji,
- **aksiome** – tvrdnje i pravila u logičkom obliku koje zajedno obuhvaćaju teoriju koju ontologija opisuje u domeni primjene,
- **događaje** – promjene atributa i odnosa.

Od osnovnih oblika ontologija najvažnije su domenske, formalne i više ili temeljne ontologije.

Domenske ontologije (engl. *domain ontologies*) modeliraju određenu domenu i reprezentiraju određena značenja pojmova u toj domeni. Domenske se ontologije fokusiraju na određeni aspekt svijeta ili stvarnosti, odnosno našega ljudskog shvaćanja znanja i njegove reprezentacije. Tako je određeni pojam definiran samo unutar te domene, a sva se druga značenja istog pojma zanemaruju.



Primjerice, semantičko značenje pojma „karta“ različito je u domeni prijevoza prometnim sredstvom, zemljopisnoj domeni, domeni igara i domeni astrologije, dok se značenje umanjećenice istog pojma, „kartica“, također semantički različito percipira u domeni računalnih dijelova, domeni sredstava plaćanja, domeni telekomunikacija ili domeni identifikacijskih sredstava.

Formalne ontologije (engl. *formal ontologies*) sadržavaju strukturu definiranu putem aksioma i cilj im je omogućiti nepristran opis stvarnosti. Njihove su osnovne značajke neograničena proširivost, neovisnost sadržaja i konteksta te prilagodba različitim razinama potankosti. Danas

postoji vrlo velik broj računalnih formalnih ontologija, a možete pogledati pregledne popise objavljenih ontologija¹⁷.

Više ili temeljne ontologije (engl. *upper ontologies*) su modeli zajedničkih objekata koji su generički upotrebljivi u širokom skupu domenskim ontologija i sadržavaju rječnike pojmova u skupovima domena.



Neke od najpoznatijih viših ontologija su *skup metapodatkovnih elemenata Dublin Core*¹⁸ koji je i norma ISO 15836:2009, zatim *opća formalna ontologija GFO*¹⁹ (engl. *General Formal Ontology*) koja opisuje procese i objekte, *ontologija Cyc* korištena u sustavima umjetne inteligencije, *ontologija SUMO*²⁰ (engl. *Suggested Upper Merged Ontology*) za računalne informacijske procesne sustave te leksička baza i ontologija *Wordnet*²¹, od kojih su neke bile predložene i za odabir norme tzv. standardne gornje ontologije (engl. *standard upper ontology*).

Glavna uloga ontologije je specificiranje modela reprezentacije podataka na razini apstrakcije iznad konkretnih logičkih ili fizičkih modela (npr. kod baza podataka) kako bi se podaci mogli uvoziti, izvoziti, prevoditi, pronalaziti i unificirati u nizu neovisnih sustava i usluga. Uspješne primjene uključuju interoperabilnost na razini baza podataka kroz različite integrirane usluge, kao što su križna pretraživanja. Svaka ontologija koristi **rječnik** ili vokabular koji je definira.

Formalne se ontologije zapisuju pomoću **ontoloških jezika** ili formalnih jezika za zapis ontologija.



Neki od najpoznatijih jezika za pisanje ontologija su *zajednički algebarski jezik za specifikacije CASL*²² (engl. *Common Algebraic Specification Language*), *Common Logic* koji je i norma ISO/IEC 24707:2007²³ koja ima i otvoreni repozitorij COLORE²⁴, formalni jezik *Gelish*²⁵ i porodica jezika OWL 2²⁶ (engl. *Web Ontology Language*) koji je i W3C preporuka.

Ontologije, kao dio normi konzorcija W3C, čine pojam semantičkog weba²⁷ (engl. *Semantic Web*), gdje se koriste za specifikaciju normativnih rječnika koji omogućavaju razmjenu podataka između sustava. **Semantički web**²⁸ je pojam koji opisuje različite metode i tehnologije koje računalima omogućuju razumijevanje značenja, odnosno semantike gomile

¹⁷ Ontology (information science), Examples of published ontologies, Wikipedia, https://en.wikipedia.org/wiki/Ontology_%28information_science%29#Published_examples

¹⁸ Dublin Core Metadata Initiative, <http://dublincore.org/>

¹⁹ General Formal Ontology (GFO), <http://savannah.nongnu.org/projects/gfo>

²⁰ Suggested Upper Merged Ontology (SUMO), <http://www.ontologyportal.org>

²¹ WordNet, a lexical database of English, <http://wordnet.princeton.edu>

²² Common Algebraic Specification Language (CASL), <http://www.informatik.uni-bremen.de/cofi/index.php/CASL>

²³ ISO Common Logic Tools, <http://sourceforge.net/projects/common-logic/>

²⁴ COLORE, Semantic Technologies Laboratory, University of Toronto, <http://stl.mie.utoronto.ca/colore/>

²⁵ Gellish@work Association, <http://www.gellish.net>

²⁶ OWL 2 Web Ontology Language Document Overview, <http://www.w3.org/TR/owl2-overview>

²⁷ 30 Berners-Lee, Tim, James Hendler i Ora Lassila, The Semantic Web, *Scientific American Magazine*, 2001

²⁸ W3C Semantic Web, <http://www.w3.org/standards/semanticweb/>

informacija koje se svakodnevno pojavljuju na Webu. Inicijalna vizija je bila stvaranje strojno čitljivih metapodataka koji opisuju postojeće i buduće informacije i sadržaje, odnosno podatke na webu, kako bi računalni programi mogli pristupati tim podacima i izvršavati automatske zadatke, poput pronalaženja odgovarajućeg sadržaja na inteligentan način, različit od pukog pretraživanja na razini sličnosti riječi traženog upita. Tehnologije semantičkog weba uključuju radne okvire, različite oblike razmjene informacije, notacije, jezike i sheme koji omogućavaju formalni opis pojmova i koncepata te njihovih odnosa u određenoj domeni znanja. Ontologije se kod semantičkog weba koriste za specifikaciju standardnih konceptualnih rječnika kako bi se omogućile usluge interoperabilnosti sustava i baza znanja.

1.3 Elektroničko poslovanje

Kada se govori o interoperabilnosti, jedan od osnovnih povezanih pojmova je elektroničko poslovanje, kao pokretač moderne interoperabilnosti, te nešto uži pojam elektroničke trgovine.

Elektroničko poslovanje (engl. *electronic business*), **ePoslovanje** ili **e-Poslovanje** (engl. *eBusiness* ili *e-Business*) u današnje je doba općepoznati pojam koji označava poslovanje u elektroničkom obliku, korištenjem moderne informacijsko-komunikacijske tehnologije, izvedeno uporabom računalnih mreža i elektroničkog oblika zapisa dokumenata. Većinom se pod računalnom mrežom primarno podrazumijeva Internet, iako nije nužno da se Internet koristi za elektroničko poslovanje. Neke od definicija elektroničkog poslovanja koje mogu dodatno približiti razumijevanje ovog pojma su sljedeće:

*„Elektroničko poslovanje je **primjena** informacijskih i komunikacijskih **tehnologija za podršku poslovnih aktivnosti**.”²⁹*

*„Elektroničko poslovanje je **skup određenih poslovnih aktivnosti uz uporabu suvremene tehnologije** kao npr. automatizacija prodaje, upravljanje on-line kupovinom, upravljanje opskrbom, upravljanje narudžbama, www reklamiranje, potpora ostvarenju dobiti, uporaba elektroničkih platnih sustava ili ostvarenje kvalitetne komunikacije između poslovnih partnera.”³⁰*

*„E-poslovanje je svaki **oblik poslovne transakcije** koja **koristi elektronički medij** (većinom Internet) kako bi provela transakciju”³¹*

Opet možemo primijetiti kako se radi o različitim, ali sukladnim definicijama, koje sve spominju poslovanje korištenje informacijskih i komunikacijskih tehnologija (ICT) i elektronički medij ili način. Poslovne transformacije temelje se najčešće na integraciji (udruživanju), kolaboraciji (suradnji) i globalnom povezivanju poduzeća (umrežavanju) te korištenju Interneta.

Drugi bitan pojam je **elektronička trgovina** (engl. *electronic commerce*), **eTrgovina** ili **e-Trgovina**, (engl. *eCommerce* ili *e-Commerce*), a odnosi se na kupoprodaju proizvoda ili usluga uporabom elektroničkih sustava i računalnih mreža. I opet se pod računalnom mrežom najčešće misli na Internet.

²⁹ Beynon-Davies P., E-Business, Palgrave, Basingtoke, 2004.

³⁰ VEdran Batoš i Damir Kalpić, Elektroničko poslovanje, materijal za predavanja, FER, 2005.

³¹ e-Poslovanje za konkurentnost malih i srednjih poduzetnika, MINPO, 2012

Navest ćemo još neke definicije:

„E-trgovina je **trgovina** koja se izvodi putem **Interneta**.“³²

„E-trgovina je **kupovina, prijenos ili razmjena** proizvoda, usluga i/ili informacija **putem računalnih mreža**.“

S obzirom da se u moderno doba trgovina provodi najčešće korištenjem računalnih mreža, a Internet je najraširenija globalna mreža, ovo su sve prikladne definicije. Smatra se da je pojam e-Trgovine došao u upotrebu 1993. godine s pojavom prvih trgovina na webu.

Najčešći svakodnevni primjeri e-Trgovine, koje i sami najčešće koristimo su korištenje web stranica za kupoprodaju proizvoda i usluga putem Interneta, te financijske transakcije korištenjem Internet bankarstva i različitih posrednika u plaćanju (uključujući i mobilno plaćanje).



Iz prethodnih je definicija također vidljivo da je elektronička trgovina u uskoj vezi s elektroničkim poslovanjem, slično kao što je i trgovina u uskoj vezi s poslovanjem. No, elektronička je trgovina uži pojam od elektroničkog poslovanja jer podrazumijeva samo odnos koji se zasniva na trgovini, a elektroničko poslovanje podrazumijeva bilo kakav oblik poslovanja na „elektronički način“.

1.3.1 Scenariji interakcija

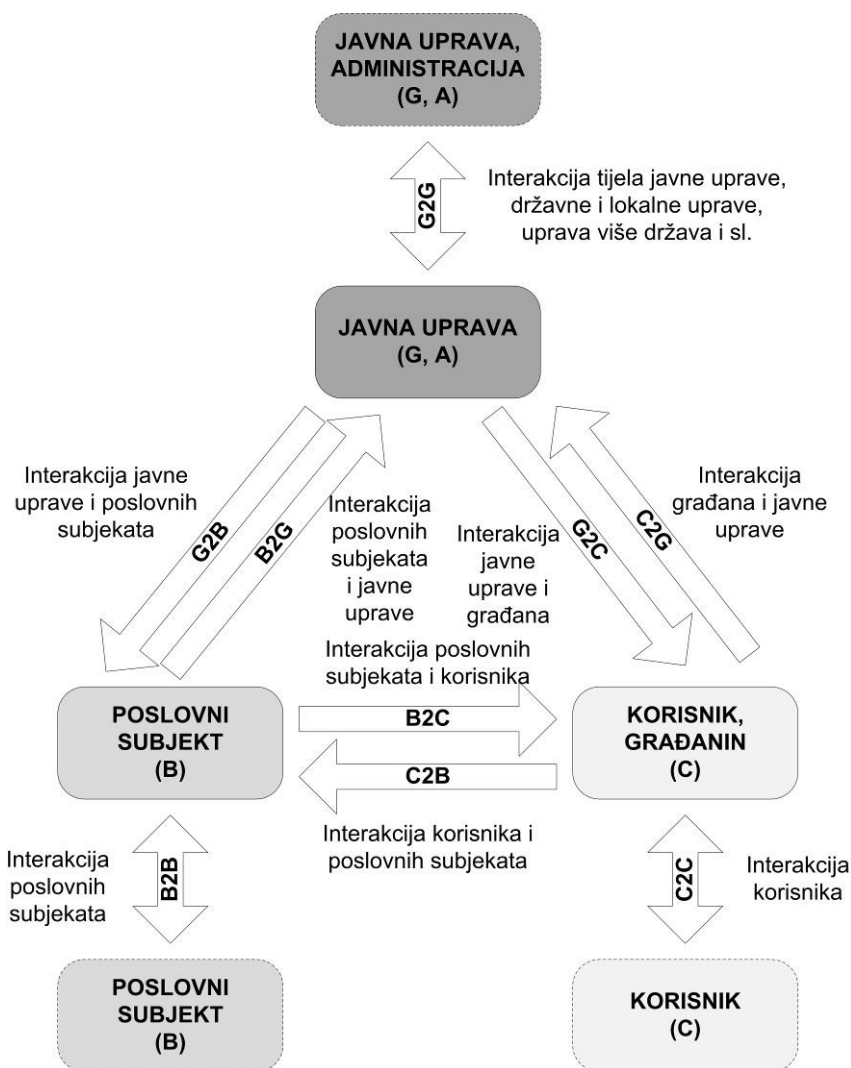
U odnosu dviju strana u elektroničkom poslovanju dolazi do određenih scenarija interakcija. Te scenarije interakcija ponajprije određuju sami subjekti odnosa, jer određeni subjekti imaju mogućnosti pružiti određene usluge, a drugi subjekti trebaju te usluge.

Ovisno o **tipu subjekta** koji sudjeluje u interakciji, razlikujemo tri osnovne vrste:

- a) **javnu upravu**, koja uključuje državnu upravu, lokalnu samoupravu, administrativni aparat države i slično,
- b) **poslovne subjekte**, što se odnosi na poduzeća – trgovačka društva (javna trgovačka društva, komanditna društva, dionička društva i društva s ograničenom odgovornošću), razne organizacije i institucije, te obrtnike i slično,
- c) **korisnike**, bilo da su građani države, zaposlenici u nekom poslovnom subjektu ili jednostavno predstavljaju „društvo“ kao krajnjeg korisnika.

Slika 1.1 prikazuje moguće kombinacije scenarija interakcije.

³² e-commerce, Merriam-Webster Dictionary, <http://www.merriam-webster.com/dictionary/e-commerce>



Slika 1.1: Uobičajeni scenariji interakcije elektroničkom poslovanju

U označavanju scenarija interakcije koriste se sljedeće znakovne oznake:

- **G** (engl. *Government*) – državna uprava, javna uprava
 - može se pojaviti i oznaka **A** (engl. *Administration*) koja označava administraciju
 - u oznakama se često poistovjećuju javna i državna uprava, a javna uprava može biti pojam koji uključuje i tijela centralne državne i lokalne samouprave
- **B** (engl. *Business*) – poslovni sektor, poslovni subjekt, organizacija, trgovačko društvo
- **C** (engl. *Citizen* ili *Consumer* ili *Civil Society*) – građanin, krajnji korisnik, civilno društvo
 - može se pojaviti i oznaka **E** (engl. *Employee*) koja označava zaposlenika ili djelatnika u odnosu poslovnog subjekta prema korisniku kojeg zapošljava
- **2** (engl. *To*) – označava odnos
 - ovo je i igra riječi jer se broj 2 (engl. *two*) u engleskom izgovara isto kao i pojam „odnos“, „prema“ ili „k“ (engl. *to*)

Navedene oznake se mogu kombinirati, a sljedeća tablica 1.1 prikazuje kratice scenarija interakcije sa značenjima i primjerima odnosa.

Oznaka	Engleski termin	Značenje	Primjer odnosa
B2B	<i>Business To Business</i>	interakcija dvaju poslovnih subjekata	odnos dviju tvrtki ili institucija, najčešće dobavljač i kupac
G2G	<i>Government To Government</i>	interakcija dvije javne uprave ili administracije	odnos uprava dviju država, državne uprave i lokalne samouprave, međusobno između tijela javne uprave i slično
C2C	<i>Consumer To Consumer</i>	interakcija dvaju korisnika	odnos dviju osoba najčešće kroz komunikaciju, kupoprodaje, društvene mreže i slično
B2G	<i>Business To Government</i>	Interakcija poslovnog subjekta i javne uprave, primarno smjer od poslovnog subjekta prema javnoj upravi	odnos poslovnog subjekta prema javnoj upravi, najčešće kroz pružanje određenih usluga
G2B	<i>Government To Business</i>	interakcija javne uprave i poslovnog subjekta, primarno smjer od javne uprave prema poslovnom subjektu	odnos javne uprave prema poslovnim subjektima, najčešće kroz pružanje usluga oblika eUprava
B2C	<i>Business To Consumer</i>	interakcija poslovnog subjekta i korisnika, primarno označava smjer od poslovnog subjekta prema korisniku	odnos poslovnog subjekta i korisnika, najčešće kroz trgovinu na Webu i eTrgovinu u obliku maloprodaje
C2B	<i>Consumer To Business</i>	interakcija korisnika i poslovnog subjekta, primarno smjer od korisnika prema poslovnom subjektu	odnos korisnika i poslovnog subjekta, najčešće kada korisnici nude svoje usluge tvrtkama
G2C	<i>Government To Citizen</i>	interakcija javne uprave i korisnika/građanina, primarno smjer od javne uprave prema korisniku/građaninu	odnos javne uprave i korisnika, najčešće kroz korištenje usluga javne uprave od strane građana
C2G	<i>Citizen To Government</i>	interakcija korisnika i javne uprave, primarno smjer od građanina/korisnika prema javnoj upravi	odnos građanina i javne uprave, relativno rijedak slučaj

Tablica 1.1: Mogućnosti scenarija interakcije

1.4 Čimbenici, koristi, razine i vrste interoperabilnosti

Kada govorimo o interoperabilnosti, prvo treba razlučiti **zašto uopće trebamo interoperabilnost**. Naravno da će svatko od nas pronaći barem nekoliko dobrih razloga, općepoznatih široj javnosti, ali to je daleko od organiziranog pristupa i stvarnog promišljanja o razlozima.



Krenimo od jednostavnog svakodnevnog primjera. Ako nekoga upitate što je to vožnja automobila i kako se vozi, većina će početi objašnjavati od uključivanja motora, ubacivanja u prvu brzinu i mijenjanja brzina. Rijetki će pri objašnjavanju (osim u autoškoli) obratiti pozornost na to da su za vožnju potrebni određeni preduvjeti, poput prometnice u određenom stanju i niz drugih uvjeta kojima tijekom vožnje treba udovoljiti, primjerice da treba vožnju prilagoditi uvjetima, da nas u okruženju očekuje niz opasnosti, da treba paziti na prometnu signalizaciju i slično, jer se to jednostavno podrazumijeva. Problem kod interoperabilnosti je baš u tome što se **unaprijed ne zna što se sve podrazumijeva**, odnosno, u prenesenom značenju našeg primjera, kakve nas prometnice, opasnosti, uvjeti vožnje, izazovi i problemi očekuju. Tek kad se utvrde neki osnovni preduvjeti, možemo govoriti o tome „kako voziti“. Ovo potpoglavlje djelomično govori baš o tome, o čimbenicima, koristima, razinama, vrstama i područjima interoperabilnosti, a sve to kako bi se u sljedećim poglavljima mogli više orijentirati na ono „kako“ ostvariti interoperabilnost.

1.4.1 Čimbenici uspostave interoperabilnosti

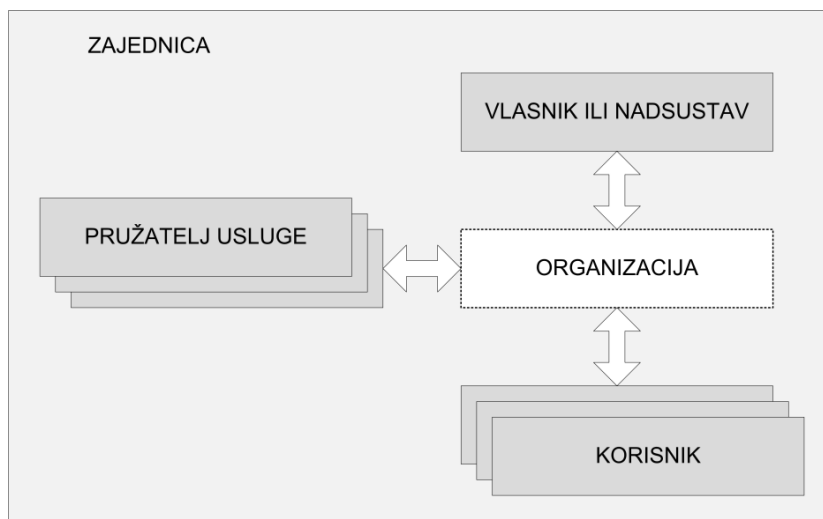
Kada se promatra interoperabilnost neke organizacije ili se ona želi uspostaviti, treba proučiti sve **čimbenike** koji mogu utjecati, a njih dijelimo na vanjske i unutarnje. Ova je podjela u skladu s područjima interoperabilnosti i potencijalnim partnerima u interoperabilnosti Hrvatskog okvira za interoperabilnost.

Vanjski čimbenici su razlog postojanja same organizacije jer je na temelju njih organizacija nastala u određenom obliku, djeluje i mijenja se. Pod vanjske čimbenike ubrajamo sve druge organizacije, okolinu i subjekte u okruženju s kojima je organizacija u nekom izravnom odnosu. Može se reći da su vanjski čimbenici **kontekst djelovanja organizacije u okruženju** koji prikazuje odnose s drugim organizacijama i subjekte.

Slika 1.2 prikazuje neposredne vanjske čimbenike:

- vlasnik ili nadsustav,
- korisnik,
- pružatelj usluge,
- zajednica,
- ostali vanjski čimbenici.

Odnos sa svakim neposrednim vanjskim čimbenikom je različit, a svaki taj odnos promotrit će se potanje, jer su neposredni vanjski čimbenici prvi potencijalni subjekti, odnosno „one druge strane“ kod uspostave interoperabilnosti.



Slika 1.2: Djelovanje organizacije u okolini

Odnos organizacije s vlasnikom ili nadsustavom je najprirodniji za uspostavu interoperabilnosti jer su interesi i ciljevi organizacije i vlasnika povezani i sukladni. Vlasnik organizacije postavlja uvjete o interoperabilnosti, a na oblik odnosa može značajnije utjecati ako postoji i neka druga povezanost s organizacijama s kojima želi uspostaviti sukladne oblike interoperabilnosti, primjerice pripadnost istoj grupaciji, konzorciju ili klasteru. Ovo se može odnositi i na trgovačka društva u privatnom vlasništvu i na organizacije unutar javne uprave i javnog sektora koji imaju zakonski definiran hijerarhijski ili resorni sustav odgovornosti.

Odnos korisnika i organizacije je također vrlo poželjan za uspostavu interoperabilnosti jer se temelji na usluzi koju organizacija pruža korisniku. Bez obzira na to je li korisnik fizička ili pravna osoba, usluga se zasniva na nizu internih procesa u organizaciji te se rezultat tih procesa prezentira korisniku. Poželjno je interoperabilne usluge graditi s detaljnim promišljanjem oblika i načina usluge koja može služiti širem krugu ili različitim skupinama korisnika, a moguće je i da se usluga inicijalno građena za jednu skupinu korisnika kasnije nadogradi i za druge skupine. Interoperabilnost prema tijelima javne uprave organizacije najčešće započinju kroz usluge automatskog izvješćivanja i dostave zakonski propisanih dokumenata, a kasnije se proširuje novim funkcionalnostima.

Budući da organizacija u svom poslovanju koristi i **usluge drugih organizacija**, bez obzira na to je li riječ o tijelima javne uprave ili poslovnim subjektima, primjerice trgovačkim društvima, te su usluge jedno od najlogičnijih područja za izgradnju interoperabilnosti. Pružatelji usluga, pogotovo ako istu (ili sličnu) uslugu pružaju nizu različitih subjekata, već vjerojatno imaju detaljno razvijen proces pružanja odgovarajuće, kvalitetne usluge, pa je provedba interoperabilnosti utoliko jednostavnija. S obzirom na dobro definiranu uslugu i jasan odnos pružatelja usluge i organizacije koja ju prima, ovo je u tehničkom provedbenom smislu jedan od jednostavnijih oblika provedbe interoperabilnosti.

Zajednica, kao okolina u kojoj organizacija djeluje, definira niz uvjeta poslovanja na političkoj, pravnoj, društvenoj i poslovnoj razini. U toj je okolini prisutna javna uprava, kao i niz drugih poslovnih subjekata. Svi su oni potencijalni kandidati za provedbu interoperabilnosti. Ako se razmišlja u skladu sa sveprisutnom globalizacijom i Hrvatskom kao dijelom EU, zajednica se može sagledati lokalno i regionalno, ali i na europskoj, odnosno globalnoj razini, jer su, praktički, i organizacije i uprava iz EU odnosno iz cijelog svijeta potencijalni partneri u interoperabilnosti.

S druge strane, **unutarnji čimbenici** su zapravo **dijelovi same organizacije i njena poslovna djelatnost**. U smislu interoperabilnosti, podjela poslovnih područja dijeli se na način sukladan područjima interoperabilnosti potanje opisanim kasnije. Za svako područje njegova uloga definira niz zakonitosti i ograničenja koje donosi, a koja se iskazuju u obliku normi i pravila.

Unutarnji čimbenici u obliku poslovnih područja dijele se na:

- političko područje i/ili područje odlučivanja,
- pravno područje,
- organizacijsko-procesno područje,
- semantičko područje,
- tehničko područje.

Političko područje (i područje odlučivanja) definira svrhu i ciljeve postojanja organizacije, odnosno volju vlasnika koja se definira u obliku akta o osnivanju organizacije, registracije djelatnosti i slično. Drugi dio ovog područja vezan je uz vođenje organizacije i procese odlučivanja. Ovdje se donosi niz internih akata organizacije koji govore o misiji, viziji, ciljevima i vrijednostima organizacije te raspolaganju resursima, djelovanju i odlučivanju. Svi propisi koji se donose unutar organizacije moraju biti u skladu s propisima zajednice u kojoj djeluje organizacija, kako bi ona bila interoperabilna.

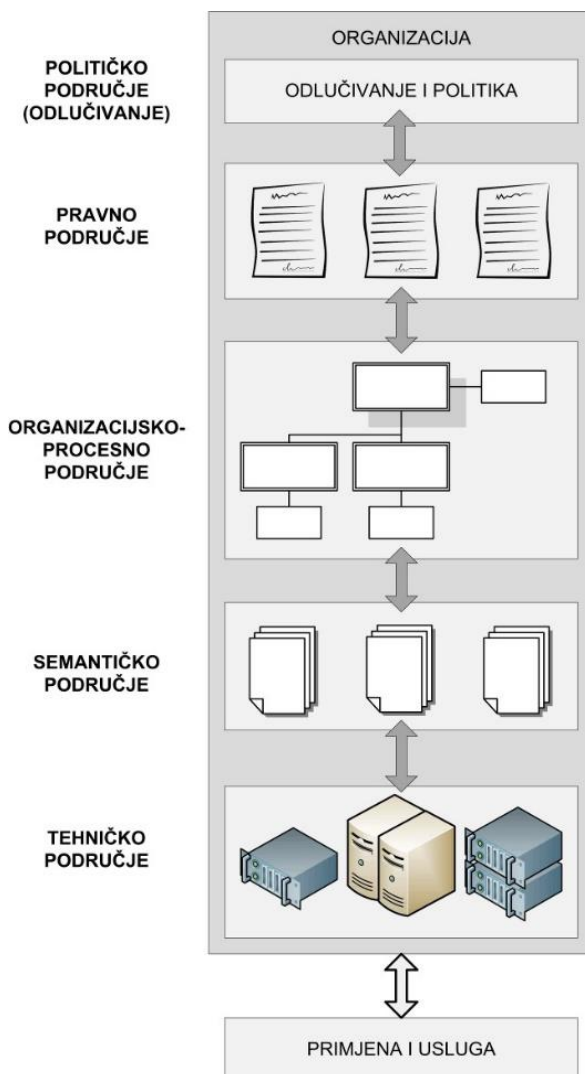
Pravno područje usko je vezano uz političko područje i područje odlučivanja te se odnosi na sve formalne propise i posljedične interne dokumente organizacije koji definiraju funkcioniranje organizacije u zakonskim okvirima te druge propise zajednice koji imaju određenu pravnu snagu. U ovom se području s političkim područjem preklapaju i neki dokumenti, kao što su inicijalni akt o osnivanju (najčešće društveni ugovor) i registracija djelatnosti, no postoji niz drugih odluka i posljedičnih dokumenata koji više pripadaju pravnom području, primjerice pravilnici, sistematizacije, ugovori i slično, a zadiru u sferu interoperabilnosti.

Organizacijsko-procesno područje odnosi se na konkretne poslovne procese u organizaciji i njihove međusobne odnose. Organizacijska struktura, definicija procesa te prava, obveze i odgovornosti u organizaciji čine operativni temelj organizacije, kojoj je cilj pružanje usluga definiranih političkim i pravnim okvirom. Upravljanje organizacijom i poslovne odluke, pogotovo srednje i niže razine, dio su svakodnevne operative i poslovnih procesa, što za posljedicu ima poslovanje u sadašnjem trenutku, ali i prikupljanje znanja koje se iskorištava u budućem poslovanju organizacije. Sami procesi organizacije, kao i usluge koje organizacija pruža, predmet su za uspostavu interoperabilnost sa drugim sudionicima.

Semantičko područje opisuje na jednoznačan, dosljedan i razumljiv način sve pojmove, elemente, sudionike i procese u organizaciji. Ovo bi se moglo usporediti s rečenicom u kojoj je jasno značenje svih riječi, što je subjekt, što objekt, a što predikat, ali je jasan i **kontekst** te **značenje** cijele rečenice. Sva iskustva, prakse i znanja prikupljena u organizaciji trebaju se semantički definirati kako bi se mogli pohraniti i dokumentirati u obliku koji nije samo znanje u umovima djelatnika koje se usmeno prenosi, već fizički eksplicitno i nepromjenjivo zapisano u različitim oblicima na odabranim medijima za pohranu. Tako je znanje organizacije moguće zapisati u obliku pravila, rječnika, priručnika i drugih dokumenata, ali i tehnički naprednijih baza podataka, XML shema i drugih računalnih oblika za pohranu. Razumijevanje znanja u organizaciji zapravo je ključno za uspješno poslovanje i učinkovito odvijanje procesa koji su po svojoj prirodi interoperabilni.

Tehničko područje je najniža razina poslovanja u organizaciji koja se ostvaruje kroz udovoljavanje tehničkim preduvjetima korištenjem različitih tehnologija. Jasno je da se interoperabilnost može postići i procesima koji ne uključuju naprednu tehnologiju, pa se poslovanje može, kao i stotinama godina prije, odvijati uporabom papira, olovke, potpisa, žiga, pošte i poslovanja u kojem su sudionici u neposrednoj fizičkoj blizini i usmeno komuniciraju. No, oni oblici interoperabilnosti na koje se fokusira ovaj priručnik podrazumijevaju moderne oblike poslovanja i komunikacije koji nadilaze sve navedeno te unaprjeđuju, automatiziraju, ubrzavaju i povećavaju učinkovitost poslovanja. **Osnovni tehnički preduvjeti** su razni oblici elektroničkih komunikacija, infrastrukture računalne mreže, sustavi za pohranu podataka, metode i mehanizmi dohvata, transformacija, slanja i primanja te pohrane informacija u elektroničkom obliku, i svi drugi aspekti koji prate ovakvo moderno poslovanje. Tek se u slučaju uporabe raznih oblika modernog elektroničkog poslovanja može govoriti o interoperabilnosti.

Sljedeća slika prikazuje pet područja poslovanja organizacije i njihov međusobni odnos. Bitno je uočiti da područja poslovanja nisu neovisna, već promjene u jednom području uzrokuju potrebu prilagodbe u drugim područjima kako bi sustav u cjelini mogao učinkovito funkcionirati. Također, interoperabilnost se ostvaruje usklađivanjem djelovanja organizacija na svih ovih pet područja poslovanja, kako je opisano kasnije.



Slika 1.3: Područja poslovanja organizacije

1.4.2 Koristi od interoperabilnosti

Iako bez puno analize možemo navesti neke koristi od uvođenja i primjene interoperabilnosti, često zanemarujemo neke aspekte neizravnih koristi koje nam interoperabilnost može donijeti. Stoga u sljedećem popisu navodimo niz općenitih koristi, od kojih se u konkretnom slučaju izvedbe mogu uočiti sve ili samo određene, kao što su:

- povećana kvaliteta usluga,
- smanjenje troškova usluga,
- smanjenje administrativnih usluga i birokracije,
- smanjenje aparata javne uprave, a time i troškova,
- izbjegavanje vezanja uz određene dobavljače (tzv. *vendor lock-in*) omogućava slobodan odabir proizvoda i usluga na tržištu,
- povećanje broja dobavljača normiranih proizvoda i usluga,
- veća kompetitivnost tržišnog natjecanja,
- migracija prema otvorenim normama,
- oslobađanje kreativnosti zbog povećanja tržišnog natjecanja (što ubrzava razvoj tehnologija),
- mogućnost jednostavnijeg ispunjenja različitih zakonskih obveza,
- ubrzanje razvoja novih usluga,
- otvaranje novih mogućnosti i poslova te rast tržišta,
- ujednačavanje mogućnosti sudjelovanja tržišnih sudionika unutar i izvan lokalnog tržišta,
- mogućnost naplate provizije unutarnjeg, vanjskog ili međunarodnog korištenja usluga,
- povećan i transparentan protok informacija između javne uprave, organizacija i krajnjih korisnika (osigurava točnije i potpunije informacije te bolju informiranost),
- pojednostavljenje korištenja usluga javne uprave i organizacija,
- spajanje usluga i *One-Stop-Shop* usluge kroz razne kanale u skladu s orijentacijom prema korisnicima,
- korištenje usluga uporabom elektroničkog identiteta i elektroničkih isprava,
- povećano sudjelovanje korisnika u korištenju javnih usluga, što povećava demokraciju,
- povećana mobilnost i transparentno korištenje usluga bez obzira na fizičku lokaciju,
- smanjenje troškova informacijsko-telekomunikacijskih tehnologija,
- bolja koordinacija usluga,
- povećana i poštenija kompetitivnost tržišnih subjekata otvorenog tržišta (pogotovo važno za manje tvrtke),
- rast novih tržišta i povećanje postojećih,
- omogućavanje ponovnog korištenja podataka i funkcionalnosti,
- bolje upravljanje i odlučivanje na temelju agregiranih podataka i pokazatelja,
- promicanje međunarodne suradnje.

Gledajući područja interoperabilnosti možemo zaključiti koje sve koristi interoperabilnost donosi. Na tehničkom području interoperabilnost svakako povećava interakciju različitih informacijsko-komunikacijskih tehnoloških sustava u razmjeni i dijeljenju podataka. Najveća su korist **velike uštede** u vremenu i novcu izbjegavanjem nenormiranih, zatvorenih, *ad hoc* i pojedinačnih rješenja komunikacije. Interoperabilnost u konačnici donosi **sigurnije, pouzdanije i učinkovitije** sustave koje zahtijevaju **manje održavanja**. Na semantičkom području interoperabilnost omogućuje **izravnu razmjenu podataka s jasnim i definiranim značenjem**. S organizacijskog gledišta sudionici trebaju prilagoditi svoje poslovne procese i organizaciju, što omogućava **bolju povezanost**, ali često i **iskoristivost**. Na pravnom i

političkom području dolazi do jasnog izražavanja **zajedničkih interesa i prioriteta** te **identifikacije zajedničkih** ciljeva uz prilagodbu upravljačkog, ali i administrativnog sustava pojedinih sudionika.

No, s obzirom na to da dolazi do velikog broja interakcija između velikog broja sudionika te se razmjenjuje velik broj osjetljivih informacija, time se povećavaju i rizici vezani uz **sigurnost i zaštitu podataka i privatnosti**. Jedan od glavnih imperativa pri provođenju interoperabilnosti, pogotovo na nacionalnoj ili međunarodnoj razini, je sigurnost podataka, a provodi se raznim oblicima zaštite podataka koji se razmjenjuju, zaštitom komunikacijskog kanala te autentikacijom i autorizacijom korisnika usluga sustava, o kojoj ćemo više govoriti kasnije.

Ukratko, može se zaključiti da interoperabilnost u konačnici omogućava **bržu izgradnju kvalitetnijih usluga prilagođenih potrebama korisnika, uz veću učinkovitost i nižu cijenu troška poslovanja**.

1.4.3 Razine interoperabilnosti

Pogledamo li pitanje jednu od definicija interoperabilnosti koja glasi: „*Interoperabilnost je mogućnost dva ili više sustava ili komponente da razmjenjuju informacije i koriste razmijenjene informacije*“³³, uočiti ćemo dva pojma vezana uz glavne procese: **razmjena** i **korištenje**. Ta dva pojma definiraju i dvije osnovne razine interoperabilnosti: sintaksnu i semantičku interoperabilnost. No, razina interoperabilnosti postoji više, a ovdje ćemo sagledati tehničku, sintaksnu, semantičku, pragmatičnu, dinamičku i konceptualnu razinu interoperabilnosti.

Krenimo od najniže razine koja se često preskače jer se podrazumijeva ako postoji bilo kakav oblik interoperabilnosti. To je **tehnička interoperabilnost** (engl. *technical interoperability*) koja se zasniva na tehničkim preduvjetima komunikacije. Ona može označavati bilo kakav spoj dvaju sustava ili računala kroz neki oblik sučelja. Danas je to ipak najčešće neki oblik računalne mreže (žične ili bežične), ali u širem smislu to je bilo kakav oblik komunikacije sustava. Ako nisu ispunjeni uvjeti tehničke interoperabilnosti, tada moderna interoperabilnost sigurno uopće ne postoji.

Sintaksna interoperabilnost (engl. *syntactic interoperability*) ili interoperabilnost na razini sintakse je sposobnost dvaju ili više sustava da **komuniciraju i razmjenjuju** podatke. Sintaksna se interoperabilnost zasniva na **tehničkoj sličnosti sustava, oblika podataka** koji se razmjenjuju i **komunikacijskih protokola**. Norme jezika koji se koriste zapis podataka, kao što su primjerice jezici XML ili SQL, omogućavaju sintaksnu interoperabilnost. Sintaksna interoperabilnost zapravo definira način komuniciranja u tehničkom smislu pa se općenito pod pojmom sintaksne interoperabilnosti često podrazumijeva i tehnička interoperabilnost, kao što su **programska sučelja, oblici zapisa podataka, oblici poruka** i slično. No, kao što ćete vidjeti malo kasnije, sintaksa i tehnika ipak su dva prilično različita pojma. Sintaksna interoperabilnost je preduvjet daljnjih, viših razina interoperabilnosti.

Semantička interoperabilnost (engl. *semantic interoperability*) ili interoperabilnost na razini semantike omogućuje, osim razmjenjivanja informacija dvaju ili više računalnih sustava, da se razmijenjene informacije automatski, smisljeno i točno **tumače** kako bi se proizveli korisni rezultati u odnosu na svrhu svih sustava koji komuniciraju. To znači da je semantička

³³ Definicija pojma *interoperability* iz IEEE (Institute of Electrical and Electronics Engineers) Standard Computer Dictionary – A Compilation of IEEE Standard Computer Glossaries, 1990.

interoperabilnost izravno vezana uz **korištenje** podataka, a ono je pak povezano sa značenjima tih podataka. Semantička se interoperabilnost može definirati i kao sposobnost svih strana da na isti način tumače **značenja** informacija koje razmjenjuju. Za postizanje semantičke interoperabilnosti obje strane u komunikaciji moraju poštovati zajednički referentni model razmjene informacija, odnosno moraju uskladiti semantička područja poslovanja, odnosno moraju se „razumjeti“. Sadržaj zahtjeva za razmjenu informacija treba biti jednoznačno definirani tako da je „**ono što je poslano jednako onome što se razumjelo**“.

Ovdje se pojavljuje i pojam **semantičke imovine** (engl. *semantic assets*), koji obuhvaća sve resurse potrebne za ostvarivanje semantičke interoperabilnosti. Semantičku imovinu zapravo čine različiti klasifikacijski sustavi, nomenklature, šifarnici, pojmovnici, rječnici ili vokabulari, XML sheme i slični popisi pojmova, ali sastavni dio semantičke imovine su i specifikacije međusobnih veza tih popisanih elemenata te pravila pridruživanja ili preslikavanja među njima, što je u uskoj vezi s ontologijama.

No, za korištenje neke semantičke imovine potrebno ju je postaviti u kontekst. Primjerice, pojam „matični broj“ ne znači isto kod studenta, građanina i pravnog subjekta. Postavljanje značenja u kontekst i posljedično korištenje tog određenog podatka s tim određenim značenjem u tom kontekstu definira **pragmatičnu interoperabilnost** (engl. *pragmatic interoperability*). Kao što smo već spomenuli, ontologija se koristi za razumijevanje pojmova i koncepata u određenoj domeni, pa se stoga za postizanje pragmatične interoperabilnosti često koriste ontologije.



Tehnička, sintaksna, semantička i pragmatična razina interoperabilnosti su osnovne razine čiji se preduvjetima treba udovoljiti kako bi se postigla učinkovita interoperabilnost. No, neki autori idu dalje s objašnjenjem sveukupnosti modela interoperabilnosti, pa govore o dinamičkoj i konceptualnoj razini interoperabilnosti, koje podrazumijevaju još više razine međusobnog komuniciranja i razumijevanja dviju strane. Pojmovi dinamičke i konceptualne interoperabilnosti preuzeti su iz teorije modeliranja i simuliranja, gdje se i govori o višim razinama interoperabilnosti, a ovdje se može iskoristiti za bolje razumijevanje stupnjevanja razina interoperabilnosti. Prema tom konceptu razine suradnje dvaju sustava temelje se na višestrukim „razinama modela konceptualne interoperabilnosti“³⁴ LCIM (engl. *Levels of Conceptual Interoperability Model*), koje su kasnije dodatno razrađene³⁵ kao na sljedećoj slici.

Dinamička interoperabilnost (engl. *dynamic interoperability*) odnosi se na promjene sustava u ovisnosti o samoj komunikaciji. Naime, kako se događa proces komunikacije tako se informacije u sustavu se s vremenom mijenjaju, što utječe i na inicijalne pretpostavke te postavlja ograničenja u razmjeni podataka. Učinci promjene podataka u vremenu s vremenom postaju razumljivi svim sudionicima komunikacije te utječu na daljnji tijek same komunikacije.

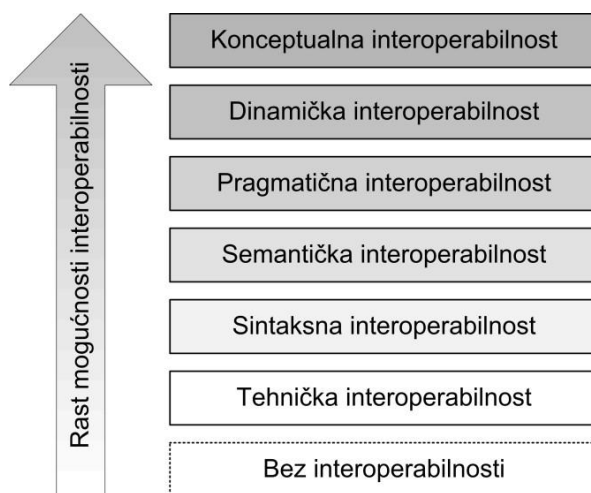
³⁴ Tolk, A. i Muguira, J.A., The Levels of Conceptual Interoperability Model (LCIM). Proceedings IEEE Fall Simulation Interoperability Workshop, IEEE CS Press, 2003.

³⁵ Turnitsa, C.D., Extending the Levels of Conceptual Interoperability Model. Proceedings IEEE Summer Computer Simulation Conference, IEEE CS Press, 2005.

Na taj način nove informacije u obliku odgovora i novih saznanja zapravo mijenjaju sam proces komunikacije u interoperabilnosti.

Najviša razina je **konceptualna interoperabilnost** (engl. *conceptual interoperability*) koja se zasniva na konceptualnom modelu, odnosno pretpostavkama i ograničenjima smislene apstrakcije stvarnosti. Osnovni koncept je da sudionici u komunikaciji u potpunosti razumiju funkcioniranje druge strane, jer je sve ono važno za razumijevanje procesa potpuno jasno specificirano i definirano, te posljedično komunicirano. To znači da nerazumijevanja između strana uopće nema. No, implementacija koja dovodi do te razine može na nižim razinama biti različita (npr. tehnološki).

Slika 1.4 prikazuje razine interoperabilnosti prema modelu LCIM.



Slika 1.4: Razine interoperabilnosti prema proširenom modelu LCIM

Kako bismo bolje razumjeli razine interoperabilnosti objasniti ćemo ih uz analogije dvaju sugovornika iz stvarnog svijeta na sljedeći način:

1. razina, **bez interoperabilnosti** – samostalni sustavi bez interoperabilnosti
 - primjer: dvije strane uopće ne komuniciraju
2. razina, **tehnička interoperabilnost** – postoji komunikacijski protokol ili razmjena podataka između dvaju sustava, uspostavljena je komunikacijska infrastruktura (mreža, protokol) za razmjenu informacija koja je jednoznačno definirana
 - primjer: dvije strane komuniciraju, jedna govori, druga čuje neke zvukove, ali ne zna točno što
3. razina, **sintaksna interoperabilnost** – postoji zajednička struktura podataka koja se koristi kod razmjene, koristi se zajednički protokol za oblikovanje podataka, a oblik zapisa informacije je jednoznačno definiran
 - primjer: u komunikaciji se razumiju glasovi i riječi, ali se ne zna njihovo značenje
4. razina, **semantička interoperabilnost** – korištenje zajedničkoga referentnog modela razmjene informacije; osim samih podataka u odgovarajućem obliku, razmjenjuje se i smisao, odnosno značenje podataka, a zahtjevi za razmjenom jednoznačno su definirani
 - primjer: u komunikaciji se razumiju riječi i njihovo pojedinačno značenje, ali ne i cijele rečenice – ne poznaje se kontekst riječi u rečenici ni u tekstu
5. razina, **pragmatična interoperabilnost** – sustavi u komunikaciji su svjesni metoda i procedura koje koristi drugi sustav, korištenje podataka u kontekstu aplikacije

razumljivo je od svih uključenih sudionika komunikacije, a podatak u kontekstu je jednoznačno definiran

- primjer: u komunikaciji se razumiju riječi u rečenici i njihov kontekst, dakle čitav tekst
- 6. razina, **dinamička interoperabilnost** – kako se podaci u sustavu s vremenom mijenjaju, to utječe i na pretpostavke i ograničenja u razmjeni podataka, pa učinci promjene podataka u vremenu postaju razumljivi svim sudionicima komunikacije
- primjer: promjene u procesima sudionika utječu na promjene u rečenicama komunikacije
- 7. razina, **konceptualna interoperabilnost** – najviša razina, zasniva se na konceptualnom modelu, odnosno pretpostavkama i ograničenjima smislene apstrakcije stvarnosti; sudionici u komunikaciji u potpunosti razumiju funkcioniranje druge strane jer je ono potpuno specificirano i definirano, ali implementacija može biti različita
- primjer: definirani su i razumiju se svi procesi i promjene kod svih sudionika komunikacije koji se posljedično u potpunosti razumiju – razumijevanje razmišljanja.

1.5 Područja usklađivanja interoperabilnosti

Područja na kojima se ostvaruje interoperabilnost kao unutarnji čimbenici interoperabilnosti već su spomenuti, a ovdje ćemo ih potanje objasniti.

Prema podjeli koja je 2010. godine usvojena u kontekstu Hrvatskog okvira za interoperabilnost (HROI), smatra se da se interoperabilnost ostvaruje usklađivanjem djelovanja organizacija na **pet osnovnih poslovnih područja**³⁶:

- političkom području i području odlučivanja,
- pravnom području,
- organizacijsko-procesnom području,
- semantičkom području,
- tehničkom području.

Svako od ovih područja ima specifični pogled na djelovanje organizacije i njenu interoperabilnost, pa ćemo ih potanje objasniti.

Interoperabilnost u političkom kontekstu zajedničkog odlučivanja dobiva se kao rezultat usklađivanja vizija i ciljeva različitih organizacija. Pretpostavka za interoperabilnost je jasno izražena volja i namjera sudjelovanja u postizanju zajedničkih ciljeva. U političkom kontekstu djeluju prvenstveno čelnici (vlasnici, direktori itd.) organizacija koje žele uspostaviti interoperabilnost, a koji mogu prepoznati zajedničke prioritete, planirati zajedničke aktivnosti i definirati ograničenja. Kako bi se uspješno uspostavila interoperabilnost, u političkom se kontekstu trebaju jasno, precizno i provedivo opisati odluke sudionika o ciljevima i načinu provedbe.

Na razini državne uprave predstavljaju takav opis u političkom kontekstu, koji definira glavna strateška područja djelovanja, postavlja ciljeve i načine njihovog ispunjavanja te nadzire rezultate. Sve to olakšava i ubrzava definiranje političkog konteksta između svih sudionika, koji će se posljedično koristiti za definiciju interesa i područja zajedničkog razvoja cjelovitih usluga,

³⁶ Odrednice Hrvatskog okvira za interoperabilnost, ver.1.0, Središnji državni ured za e-Hrvatsku, 24.6.2010.

dogovor oko ciljeva, uvjeta i sredstava za provedbu zajedničkih projekata te konačnu provedbu konkretnih mjera i projekata.

Pravna interoperabilnost postiže se usklađivanjem pravnih područja sudionika uz utvrđivanje potencijalnih pravnih zapreka za postizanje ciljeva prethodno dogovorenih u političkom kontekstu. Detaljna usporedba i analiza odgovarajućih zakonskih specifičnosti, obveza pojedinih sudionika, internih pravnih akata, formalnih obveza prema trećim organizacijama i svih pravnih situacija koje se otvaraju u političkom kontekstu utvrđuje postojanje formalno-pravnih zapreka i ograničenja koja mogu utjecati na uspostavu interoperabilnosti, a tamo gdje zapreke postoje treba uzeti u obzir njihove posljedice i pronaći načine njihovog uklanjanja. Ako uklanjanje pravnih zapreka pretpostavlja složene, dugotrajne i neizvjesne postupke rješavanja, kao što su izmjene i dopune zakona i podzakonskih akata, onda se to treba uzeti u obzir prilikom pokušaja uspostavljanja interoperabilnosti u pravnom području.



Primjer takvih problema pravne interoperabilnosti iz nedavne prakse je kada se u političkom kontekstu kao zajednički cilj definira usluga koja pretpostavlja razmjenu dokumenta u obliku elektroničke isprave, korištenjem komunikacijskih mreža, uz ovjeravanje, odnosno identifikaciju potpisnika i potvrdu vjerodostojnosti potpisanoga elektroničkog zapisa naprednim elektroničkim potpisom, a zakon primjerice još uvijek podrazumijeva obvezan fizički obrazac, ispravu ispisanu na papiru u točno određenom obliku, uz vlastoručni potpis olovkom. Tada treba razmotriti postoji li kakva pravna alternativa koja bi omogućila takvu uslugu ili zapravo treba mijenjati zakon.

Organizacijsko-procesna interoperabilnost utvrđena je poslovnom usklađenošću sudionika te omogućava kvalitetno i održivo postizanje ciljeva prethodno definiranih u političkim kontekstu, a unutar utvrđenih okvira pravne interoperabilnosti. Učinkovito povezivanje procesa dvaju sudionika radi vlastite koristi ili pružanja zajedničke usluge nekom trećem korisniku postiže interoperabilnost na organizacijsko-procesnom području. S obzirom na to da su određeni poslovni procesi tijela državne uprave uređeni zakonima i podzakonskim aktima, ako se pokušava ostvariti interoperabilnost s javnom upravom, prvo se predviđa potanje opisivanje i dokumentiranje procesne interoperabilnosti s razine državne uprave, koju onda usvajaju pojedini korisnici. Tako bi se trebali definirati generički modeli poslovnih procesa, koji će onda predstavljati referentni obrazac usklađen s propisima. Na osnovi tog referentnog obrasca moći će se brzo i učinkovito razraditi i usklađivati modele poslovnih procesa u bilo kojoj organizaciji za postizanje interoperabilnosti u organizacijsko-procesnom području na nacionalnoj razini.



U Hrvatskoj je Središnji državni ured za e-Hrvatsku bio zadužen da, u okviru Hrvatskog okvira za interoperabilnost (HROI), izradi metodološke priručnike i smjernice s preporukama za modeliranje i prezentaciju modela poslovnih procesa te izgradnju usluga za pohranjivanje, objavljivanje i korištenje referentne dokumentacije i modela. Nakon preustroja 2012. godine osnovana je Uprava za e-Hrvatsku pod Ministarstvom uprave, koja je preuzela većinu tih obveza.

Semantičku interoperabilnost već smo prije definirali kao sposobnost svih sudionika da na jednak način tumače značenja informacija koje razmjenjuju. Moguće ju je postići jedino usklađivanjem semantičkih područja poslovanja između sudionika i definiranjem zajedničke

semantičke imovine. Podsjetimo se, semantička imovina uključuje različite šifrnike, nomenklature, pojmovnike, rječnike, XML sheme i slično te opisuje međusobne veze pojmova i pravila među njima. U javnoj su upravi strukture i značenja podataka često već definirane odgovarajućim propisima i pohranjene u postojećim registrima. Dosljedna i kontinuirana ugradnja svih tih resursa u zajedničku semantičku imovinu preduvjet je za uspješnu interoperabilnost na semantičkom području.



U Hrvatskoj, u okviru provedbe Hrvatskog okvira za interoperabilnost (HROI), utvrđuju se aktivnosti u kojima tijela državne uprave izrađuju prijedloge dokumenata semantičke interoperabilnosti. Posljedično tome svako tijelo državne uprave treba izraditi dokumentaciju o semantičkoj interoperabilnosti svojih procesa, jer ono samo najbolje poznaje svoje procese zasnovane na pozitivnim pravnim propisima iz njihove nadležnosti. Po dokumentiranju procesa uskladit će se te procese između raznih tijela, ali i prema praksi u EU, a to sigurno uključuje i preispitivanje određenih procesa i pravnih propisa na kojima se temelje, uz potencijalno definiranje prijedloga promjena tih propisa i posljedično procesa. U svakom pojedinom projektu nadležna tijela primjenjuju semantičke modele, a za podatke koji još u tom trenutku nisu obuhvaćeni bibliotekom semantičke imovine izradit će se odgovarajući dokumenti semantičke interoperabilnosti koji će se predložiti za uključivanje u semantičku imovinu.

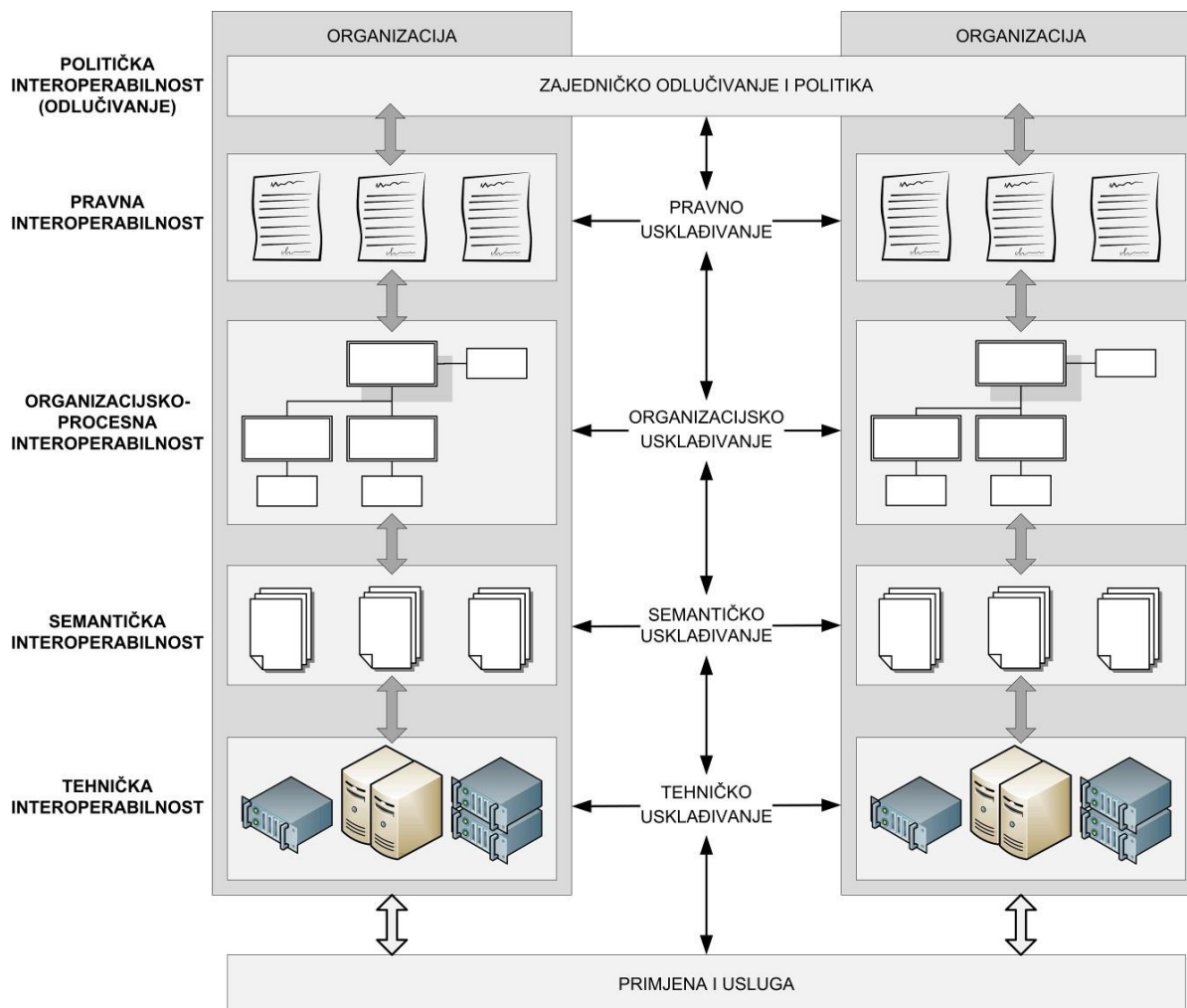
Tehnička interoperabilnost, koju smo također već spomenuli, definira se kao svojstvo informacijskih sustava sudionika u političkom kontekstu da prema korisnicima djeluju kao virtualno jedinstven sustav. Pretpostavka interoperabilnosti u tehničkom području je **katalog tehničkih normi**, koji sadržava popis svih poznatih tehničkih normi koje se primjenjuju u informacijskim sustavima ili se planiraju koristiti. Pojedini zapis u katalogu tehničkih normi sadržava punu referencu norme, što uključuje izvor norme, puni naziv norme, datum usvajanja norme, inačicu i rok korištenja, kako bi se norma jednostavno mogla pronaći, preuzeti i koristiti.

Uz svaku normu treba odrediti i njen status, pa tako norma može biti obvezna, preporučena, nepreporučena ili zabranjena. Samo se obvezne i preporučene norme koriste za ostvarivanje interoperabilnosti u tehničkom području. Norme koje nisu preporučene mogu se koristiti do kraja roka korištenja ako ne predstavljaju zapreku uspostavi tehničke interoperabilnosti, a zabranjene se norme više ne smiju koristiti, odnosno umjesto njih treba koristiti obvezne ili preporučene norme ako postoje. Ako se istovremeno koriste različite inačice iste norme, onda se sve one stavljaju u katalog tehničkih normi.

Očigledno je da se interoperabilnost na svih pet navedenih područja treba uskladiti, jer svako odstupanje na bilo kojem od područja dovodi do neispunjenja načela interoperabilnosti u primjeni. Sva navedena područja interoperabilnosti u izravnoj su vezi i njihovo usklađivanje se može promatrati razinski, slično kao što su promatrane i druge razine interoperabilnosti kasnije u tekstu.

Slika 1.5 prikazuje grafički sva područja ili vrste interoperabilnost te procese usklađivanja među njima. Tehnička je interoperabilnost na najnižoj razini, slijedi semantička interoperabilnost, zatim organizacijsko-procesna interoperabilnost, iznad nje je razina pravne interoperabilnosti, a na vrhu je najviša razina interoperabilnosti u političkom kontekstu zajedničkog odlučivanja. Kao što se može primijetiti, **usklađivanje** se događa **između različitih razina interoperabilnosti**, odnosno područja poslovanja unutar svake organizacije,

te **na pojedinim razinama interoperabilnosti** između dvije organizacije. Tek sve to zajedno može dovesti do odgovarajućih efekata usklađenosti interoperabilnosti u cjelini.



Slika 1.5: Usklađivanje područja interoperabilnosti dvije organizacije



U Hrvatskoj se uspostavljaju mehanizmi usklađivanja strateških dokumenata i propisa na političkom i pravnom području, a glavnu ulogu imao je već spomenuti Središnji državni ured za e- Hrvatsku, a nakon toga Uprava za e-Hrvatsku. Čitav postupak započinje razmatranjem poslovnih područja i interoperabilnosti organizacija u državnoj upravi koja će utvrditi polazište za uspostavu međusobne interoperabilnosti na državnoj razini. Pravila, organizacijske sheme, poslovni procesi, strukture i značenje podataka, a katkad i tehničke norme definirani su nizom propisa u obliku zakona, uredbi, pravilnika i drugih podzakonskih akata. Time ti propisi neizravno definiraju i određene elemente interoperabilnosti. Preispitivanje prilagodbe propisa važna je sastavnica provedbe interoperabilnosti na državnoj razini kod svih relevantnih sudionika, a više o tome govori se nastavku.

1.6 Pravni okvir vezan uz interoperabilnost u Hrvatskoj

Kao pretpostavke razvoju informacijskog društva, elektroničkog poslovanja i usluga, a time i interoperabilnosti informacijskih sustava, Republika Hrvatska donijela je niz zakona, pravilnika, uredbi i drugih podzakonskih akata koji uređuju osnovne zakonske okvire, definiraju pojmove, odgovornosti i drugo. U ovom ćete se potpoglavlju upoznati s osnovnim zakonima, njihovim glavnim dijelovima i pojmovima koje definiraju.

Tako su u narednim potpoglavljima ukratko obrađeni sljedeći zakoni:

- Uredba (EU) br. 910/2014 Europskog parlamenta i Vijeća od 23. srpnja 2014. o elektroničkoj identifikaciji i uslugama povjerenja za elektroničke transakcije na unutarnjem tržištu (eIDAS) i bivši Zakon o elektroničkom potpisu,
- Zakon o elektroničkoj trgovini,
- Zakon o elektroničkoj ispravi,
- Zakon o institucijama za elektronički novac,
- Zakon o elektroničkim medijima,
- Zakon o elektroničkim komunikacijama,
- Zakon o informacijskoj sigurnosti,
- Zakon o tajnosti podataka,
- Zakon o pravu na pristup informacijama i
- Zakon o zaštiti osobnih podataka.

1.6.1 Pravni okvir usluga informacijskog društva

Jedna od najpoznatijih usluga informacijskog društva je elektronička trgovina. Osim elektroničke trgovine, i niz drugih usluga podrazumijeva korištenje raznih oblika elektroničkih isprava, kao i jamstva da je određena isprava izdana od određene osobe, odnosno potpisana. Iz tih su razloga u Hrvatskoj doneseni zakoni i podzakonski akti koji uređuju pravni okvir za ostvarivanje elektroničke trgovine i drugih usluga informacijskog društva.

Relativno rano u odnosu na druge zakonske akte, točnije 24. siječnja 2002. godine, donesen je **Zakon o elektroničkom potpisu**³⁷ koji je uređivao pravo fizičkih i pravnih osoba na uporabu elektroničkog potpisa u upravnim, sudskim i drugim postupcima, poslovnim i drugim radnjama, te prava, obveze i odgovornosti fizičkih i pravnih osoba u vezi s davanjem usluga certificiranja elektroničkog potpisa. 8. lipnja 2017. taj zakon je stavljen izvan snage stupanjem na snagu Zakona o provedbi Uredbe (EU) br. 910/2014 Europskog parlamenta i Vijeća od 23. srpnja 2014. o elektroničkoj identifikaciji i uslugama povjerenja za elektroničke transakcije na unutarnjem tržištu i stavljanju izvan snage Direktive 1999/93/EZ³⁸.

³⁷ Zakon o elektroničkom potpisu, Narodne novine, br. 10/2002, 80/2008. i 30/2014.

³⁸ Zakon o provedbi Uredbe (EU) br. 910/2014 Europskog parlamenta i Vijeća od 23. srpnja 2014. o elektroničkoj identifikaciji i uslugama povjerenja za elektroničke transakcije na unutarnjem tržištu i stavljanju izvan snage Direktive 1999/93/EZ, Narodne novine, br. 62/2017.



Zakon o elektroničkom potpisu je bio jedan od naprednijih zakona s početka stoljeća koji je dobro definirao značenja pojmova, kao što su napredan elektronički potpis, napredan vremenski žig, elektronički zapis, podaci i sredstva za izradu i verificiranje elektroničkog potpisa, certifikat, kvalificirani certifikat i davatelja usluga certifikacije.

Uredba (EU) br. 910/2014 Europskog parlamenta i Vijeća od 23. srpnja 2014. o elektroničkoj identifikaciji i uslugama povjerenja za elektroničke transakcije na unutarnjem tržištu - eIDAS³⁹, odnosi se prvenstveno na izgradnju povjerenja u *online* okruženju, što se smatra ključnim za gospodarski i socijalni razvoj sva članice EU. Ovom Uredbom se nastoji se povećati povjerenje u elektroničke transakcije na unutarnjem tržištu pružanjem zajedničkog temelja za sigurnu elektroničku interakciju između građana, poduzeća i tijela javne vlasti, povećavajući time djelotvornost javnih i privatnih online usluga, elektroničkog poslovanja i elektroničke trgovine u EU.

Uredba eIDAS sadrži 7 poglavlja koja potanje opisuju određena područja: Opće odredbe, Elektronička identifikacija, Usluge povjerenja. Elektronički dokumenti, Delegiranja ovlasti i provedbene odredbe i Završne odredbe. U kontekstu interoperabilnosti najzanimljivije je poglavlje *III. Usluge povjerenja* koje potanje govori o kvalificiranim uslugama povjerenja te potanje o elektroničkim potpisima, elektroničkim pečatima, elektroničkim vremenskim žigovima, uslugama elektronički preporučene dostave i autentikaciji mrežnih stranica. Tako se u Uredbi primjerice jednoznačno definiraju sljedeći pojmovi:

- **elektronička identifikacija** je postupak korištenja osobnim identifikacijskim podacima u elektroničkom obliku koji na nedvojben način predstavljaju bilo fizičku ili pravnu osobu ili fizičku osobu koja predstavlja pravnu osobu, a **osobni identifikacijski podaci** označava skup podataka koji omogućavaju da se utvrdi identitet fizičke ili pravne osobe ili fizičke osobe koja predstavlja pravnu osobu;
- **autentikacija** znači elektronički postupak koji omogućava da elektronička identifikacija fizičke ili pravne osobe ili izvornost i cjelovitost podataka u elektroničkom obliku budu potvrđeni;
- **elektronički potpis** znači podaci u elektroničkom obliku koji su pridruženi ili su logički povezani s drugim podacima u elektroničkom obliku i koje potpisnik koristi za potpisivanje, **napredan elektronički potpis** znači elektronički potpis koji na nedvojben način je povezan s potpisnikom, omogućava identificiranje potpisnika, izrađen je korištenjem podacima za izradu elektroničkog potpisa koje potpisnik može, uz visoku razinu pouzdanja, koristiti pod svojom isključivom kontrolom i povezan je s njime potpisanim podacima na način da se može otkriti bilo koja naknadna izmjena podataka., dok **kvalificirani elektronički potpis** znači napredan elektronički potpis koji je izrađen pomoću kvalificiranih sredstava za izradu elektroničkog potpisa i temelji se na kvalificiranom certifikatu za elektroničke potpise;

³⁹ Uredba (EU) br. 910/2014 Europskog parlamenta i Vijeća od 23. srpnja 2014. o elektroničkoj identifikaciji i uslugama povjerenja za elektroničke transakcije na unutarnjem tržištu i stavljanju izvan snage Direktive 1999/93/EZ

- **certifikat za elektronički potpis** znači elektronička potvrda koja povezuje podatke za validaciju elektroničkog potpisa s fizičkom osobom i potvrđuje barem ime ili pseudonim te osobe
- **usluga povjerenja** znači elektronička usluga koja se u pravilu pruža uz naknadu i koja se sastoji od izrade, verifikacije i validacije elektroničkih potpisa, elektroničkih pečata ili elektroničkih vremenskih žigova, usluge elektroničke preporučene dostave i certifikata koji se odnose na te usluge ili izrade, verifikacije i validacije certifikata za autentikaciju mrežnih stranica ili čuvanja elektroničkih potpisa, pečata ili certifikata koji se odnose na te usluge;
- **sredstvo za izradu elektroničkog potpisa** znači konfigurirani softver ili hardver koji se koristi za izradu elektroničkog potpisa;
- **elektronički pečat** znači podaci u elektroničkom obliku koji su pridruženi drugim podacima u elektroničkom obliku ili su logički povezani s njima radi osiguravanja izvornosti i cjelovitosti tih podataka, **napredan elektronički pečat** znači elektronički pečat koji je na nedvojben način je povezan s autorom pečata, omogućava identificiranje autora pečata, izrađen je korištenjem podacima za izradu elektroničkog pečata koje autor pečata može, uz visoku razinu pouzdanja i pod svojom kontrolom, koristiti za izradu elektroničkog pečata i povezan je s podacima na koje se odnosi na takav način da se može otkriti bilo koja naknadna izmjena podataka, a **kvalificirani elektronički pečat** znači napredan elektronički pečat koji je izrađen pomoću kvalificiranog sredstva za izradu elektroničkog pečata i koji se temelji na kvalificiranom certifikatu za elektronički pečat;
- **elektronički vremenski žig** znači podaci u elektroničkom obliku koji povezuju druge podatke u elektroničkom obliku s određenim vremenom i na taj način dokazuju da su ti podaci postojali u to vrijeme, a **kvalificirani elektronički vremenski žig** znači elektronički vremenski žig koji povezuje datum i vrijeme s podacima na način kojim se u razumnoj mjeri isključuje mogućnost nezapažene promjene podataka, se temelji se na izvoru točnog vremena povezanom s koordiniranim svjetskim vremenom i potpisan je pomoću naprednog elektroničkog potpisa ili pečaćen pomoću naprednog elektroničkog pečata kvalificiranog pružatelja usluga povjerenja ili jednakovrijednom metodom.
- **elektronički dokument** znači svaki sadržaj koji je pohranjen u elektroničkom obliku, a posebno kao tekstualni ili zvučni, vizualni ili audiovizualni zapis;
- **certifikat za autentikaciju mrežnih stranica** znači potvrda pomoću koje je moguće izvršiti autentikaciju mrežnih stranica te kojom se mrežne stranice povezuju s fizičkom ili pravnom osobom kojoj je izdan certifikat;
- **validacija** znači postupak verifikacije i potvrđivanja da su elektronički potpis ili pečat valjani.

Potanje o elektroničkom potpisu i njegovoj vezi sa zakonskim okvirima opisano je u posebnom poglavlju, gdje je prikazano i kako se koristi elektronički potpis. Uobičajeno, uz područje ove Uredbe veže se i niz podzakonskih akata.

Vrlo rano, a samo godinu i pol dana nakon postavljanja zakonskih okvira za elektronički potpis, 2003. godine donesen je **Zakon o elektroničkoj trgovini**⁴⁰, kojim se uređuje pružanje usluga informacijskog društva, odgovornost davatelja usluga te pravila u vezi sa sklapanjem ugovora u elektroničkom obliku. Zakon je 2011. godine usklađen sa Direktivom 2000/31/EZ Europskog parlamenta i Vijeća. U Zakonu o elektroničkoj trgovini definira se značenje sljedećih pojmova:

- **podatak** – informacija, poruka i dokument sastavljen, poslan, primljen, zabilježen, pohranjen ili prikazan elektroničkim, optičkim ili sličnim sredstvom, uključujući prijenos Internetom, elektroničku poštu i komunikaciju telefaksom,
- **usluga informacijskog društva** – usluga koja se uz naknadu pruža elektroničkim putem na zahtjev korisnika, a posebno prodaja robe i usluga, nuđenje podataka, reklamiranje i elektronički pretraživači na Internetu, te mogućnost traženja podataka i usluga koje se prenose elektroničkom mrežom, posreduju u pristupu mreži ili pohranjuju podatke korisnika,
- **davatelj usluga** – pravna ili fizička osoba koja pruža usluge informacijskog društva,
- **korisnik usluge** – fizička ili pravna osoba koja zbog profesionalnih ili drugih ciljeva koristi uslugu informacijskog društva,
- **ugovori u elektroničkom obliku** – ugovori što ih pravne i fizičke osobe u cijelosti ili djelomično sklapaju, šalju, primaju, raskidaju, otkazuju, pristupaju i prikazuju elektroničkim putem koristeći elektronička, optička ili slična sredstva, uključujući prijenos Internetom,
- **komercijalno priopćenje** – priopćenje u bilo kojem obliku, oblikovano da promiče, izravno ili neizravno, robu, usluge ili ugled svake pravne ili fizičke osobe koja obavlja registriranu djelatnost,
- **potrošač** – fizička osoba koja sklapa pravni posao na tržištu u svrhe koje nisu namijenjene njezinu zanimanju niti njezinoj poslovnoj aktivnosti ili poduzetničkoj djelatnosti.



Davatelj usluga mora u obliku i na način koji je neposredno i stalno dostupan **korisnicima pružiti informacije**, kao što su ime i prezime ili naziv tvrtke, sjedište, OIB, ostale podatke na temelju kojih korisnik može brzo i nesmetano stupiti s njim u vezu (elektroničku adresu), broj sudskog ili drugoga javnog registra u koji je davatelj usluga upisan, podatke o registru, porezni broj (ako je davatelj usluga obveznik plaćanja PDV-a) te druge informacije, poput jasno i nedvosmisleno naznačenih cijena, uz naznaku uključenih troškova dostave, manipulativnih troškova, poreza i drugih troškova koji utječu na cijenu.

Zakon definira oblik i valjanost **ugovora u elektroničkom obliku** kao i **ponuda** te prihvata ponuda u obliku elektroničke poruke potpisane elektroničkim potpisom, uz ogradu neprimjenjivanja za određene slučajeve (imovinski, darovni i bračni ugovori i sl.). Kada se kao

⁴⁰ Zakon o elektroničkoj trgovini, Narodne novine, br. 173/2003., 67/2008., 36/2009., 130/2011., 30/2014. i 32/2019.

pretpostavka valjanosti i nastanka ugovora traži potpis osobe, smatra se tom uvjetu odgovara elektronička poruka potpisana elektroničkim potpisom.



U zakonu je izražen i problem **neželjene elektroničke komunikacije** (netraženoga komercijalnog priopćenja), odnosno tzv. *anti-spam* politike, tako da se korištenje elektroničke pošte u svrhu slanja komercijalnog priopćenja koje nije zatraženo (neželjeno telekomunikacijsko priopćenje ili engl. *spam*) dopušta samo uz prethodni pristanak osobe kojoj je takva priopćenje namijenjeno, a u skladu sa zakonom kojim se uređuje područje telekomunikacija (elektroničkih komunikacija).

Osim navedenog definiraju se i odgovornosti posrednika usluga u vezi s porukama elektroničke pošte, privremenom pohranom podataka (engl. *caching*), pohranom podataka (engl. *hosting*) i poveznicama (engl. *links*), a određene stavke ovog zakona stupile su automatski na snagu prijemom Hrvatske u EU.

Treći važan zakon iz područja usluga informacijskog društva je **Zakon o elektroničkoj ispravi**⁴¹, donesen 2005. godine. Ovim se zakonom uređuje pravo fizičkih i pravnih osoba na uporabu elektroničke isprave u svim poslovnim radnjama i djelatnostima te u postupcima pred tijelima javne vlasti, pravna valjanost elektroničke isprave te uporaba i promet elektroničkih isprava. Naravno, to se odnosi na postupke u kojima se elektronička oprema i programi mogu primjenjivati u izradi, prijenosu, pohrani i čuvanju informacija u elektroničkom obliku, a odredbe ovog zakona ne primjenjuju se, nažalost, u potencijalnim slučajevima gdje se drugim zakonima izričito traži uporaba isprava na papiru. U Zakonu o elektroničkoj ispravi dodatno se definira značenje sljedećih pojmova:

- **elektronička isprava** – jednoznačno povezan cjelovit skup podataka koji su elektronički oblikovani (izrađeni pomoću računala i drugih elektroničkih uređaja), poslani, primljeni ili sačuvani na elektroničkom, magnetnom, optičkom ili drugom mediju i koji ima svojstva kojima se utvrđuje izvor (stvaratelj) i vjerodostojnost sadržaja te dokazuje postojanost sadržaja u vremenu, a čiji sadržaj uključuje sve oblike pisanog teksta, podatke, slike i crteže, karte, zvuk, glazbu ili govor,
- **dokumentacijsko svojstvo** – obvezni skup podataka, poput elektroničkog potpisa, vremena izrade, naziva stvaratelja i drugih, koji se ugrađuju u elektroničku ispravu radi zadržavanja vjerodostojnosti, cjelovitosti i valjanosti kroz vremensko razdoblje utvrđeno zakonima i drugim propisima,
- **stvaratelj** – fizička ili pravna osoba koja primjenom elektroničkih sredstava (uređaji i programi) izrađuje, oblikuje i potpisuje elektroničku ispravu svojim naprednim elektroničkim potpisom,
- **pošiljatelj** – fizička ili pravna osoba koja šalje ili u ime koje se primatelju šalje elektronička isprava (ne uključuje informacijskog posrednika),
- **primatelj** – fizička ili pravna osoba koja prima njoj upućenu elektroničku ispravu, što ne uključuje informacijskog posrednika
- **dokumentacijski ciklus** – promet elektroničke isprave od trenutka izrade do arhiviranja, uključujući izradu, slanje, primanje, pohranjivanje i čuvanje s pridruženim postupcima

⁴¹ Zakon o elektroničkoj ispravi, Narodne novine, br. 150/2005.

unošenja i ovjere podataka kojima se potvrđuje stvaratelj, pošiljatelj, primatelj, vrijeme otpreme i vrijeme prijema, vjerodostojnost i cjelovitost elektroničke isprave,

- **informacijski sustav i djelatnik informacijskog sustava** – informacijsko-komunikacijski sklop kao skup programa, informatičkih i telekomunikacijskih uređaja, metoda i postupaka primijenjenih u postupcima izrade, slanja, primanja, provjere i čuvanja elektroničkih isprava te fizička osoba koja izravno rukuje elektroničkim ispravama,
- **informacijski posrednik** – fizička ili pravna osoba koja u ime drugih obavlja otpremu, prijem, prijenos i čuvanje elektroničkih isprava,
- **elektronička arhiva** – skup elektroničkih isprava uređenih u dokumentacijske cjeline u skladu sa zakonom i drugim propisima.

U zakonu stoji da elektronička isprava ima **istu pravnu snagu kao i isprava na papiru** ako se njena uporaba i promet provode u skladu sa zakonskim odredbama i **pravno je valjana** ako je izrađena, otpremljena, primljena, čuvana i pohranjena primjenom dostupne informacijske tehnologije, u potpunosti ispunjava zahtjeve sadržane u zakonu, ima osnovnu građu utvrđenu zakonom te se može prikazati u obliku koji je sukladan obrascu utvrđenom u Zakonu.

Zahtjevi za elektroničku ispravu propisuju da elektronička isprava mora u svim radnjama uključenim u dokumentacijski ciklus osigurati:

- jednoznačno obilježje kojim se nedvojbeno **utvrđuje** pojedinačna elektronička **isprava**,
- jednoznačno obilježje kojim se nedvojbeno **utvrđuje** pojedinačni **stvaratelj** elektroničke isprave,
- informacijsku **cjelovitost** i **nepovredivost** elektroničke isprave,
- **pristup** sadržaju elektroničke isprave kroz cijelo vrijeme dokumentacijskog ciklusa,
- oblik zapisa koji čitatelju omogućuje jednostavno **čitanje** sadržaja.

U Zakonu se kao **građa elektroničke isprave** definiraju dva obvezna neodvojiva dijela: **opći dio**, koji čini predmetni sadržaj (informacije u elektroničkom obliku) isprave, a uključuje naslov primatelja (ako je namijenjena imenovanom primatelju), te **posebni dio**, koji čine jedan ili više ugrađenih elektroničkih potpisa, podaci o vremenu nastajanja elektroničke isprave te druga dokumentacijska svojstva.

Zakon propisuje i **obrazac prikaza elektroničke isprave**, odnosno obvezno sadržavanje unutarnjeg i vanjskog obrasca prikaza koji se koriste u prikazivanju sadržaja i tijekom rukovanja sadržajima ugrađenim u elektroničku ispravu. **Unutarnji obrazac** prikaza sastoji se od tehničko-programskog obrasca zapisivanja sadržaja u elektroničkom obliku na medij koji zadržava ili prosljeđuje elektroničku ispravu, a **vanjski obrazac** sastoji se od vizualnog i razumljivog prikaza sadržaja elektroničke isprave na zaslonu računalnih ili drugih elektroničkih uređaja, na papiru ili drugom materijalnom predmetu proizvedenom iz zapisa u elektroničkom obliku na mediju. Dopuštena je uporaba preslike elektroničke isprave na papiru koja, ako je ispis ovjerala ovlaštena osoba ili javni bilježnik, ima istu pravnu snagu.



Fizička i pravna osoba koja svojom izričito očitovanom voljom prihvaća uporabu i promet elektroničkih isprava za svoje potrebe kao i za potrebe poslovnih i drugih odnosa s drugima ne može odbiti elektroničku ispravu samo zato što je sačinjena, korištena i prometovana u elektroničkom obliku. No, uporaba elektroničkih isprava ne smije niti jednoj strani uključenoj u poslove razmjene elektroničkih isprava ograničavati poslovanje ili ju dovoditi u neravnopravan položaj ako za to nema tehničke i druge preduvjete.

Nadalje, **Zakon o elektroničkom novcu**⁴² koji je danas na snazi, zasnovan je na Zakonu o institucijama za elektronički novac⁴³ iz 2008. godine, Zakonu o elektroničkom novcu⁴⁴ iz 2010. godine, te je usklađen s zakonskim propisima EU. Taj zakon uređuje elektronički novac, izdavanje i izdavatelje elektroničkog novca, te uvjete za osnivanje, poslovanje, prestanak i nadzor rada institucija za elektronički novac sa sjedištem u Republike Hrvatske, kao iz poslovanje institucija za elektronički novac sa sjedištem izvan Republike Hrvatske. U zakonu o elektroničkom novcu definira se značenje sljedećih pojmova:

- **elektronički novac** – elektronički, uključujući i magnetski, pohranjena novčana vrijednost koja je izdana nakon primitka novčanih sredstava u svrhu izvršavanja platnih transakcija u smislu zakona kojim se uređuje platni promet i koju prihvaća fizička ili pravna osoba koja nije izdavatelj tog elektroničkog novca, a koja čini novčano potraživanje prema izdavatelju,
- **institucija za elektronički novac** – pravna osoba koja je dobila odobrenje za izdavanje elektroničkog novca
- **platne usluge** – usluge koje pružatelji platnih usluga obavljaju kao svoju djelatnost, a to mogu biti usluge: **vođenja računa** za plaćanje, **polaganja gotovog novca** na račun za plaćanje, **podizanja gotovog novca** s računa za plaćanje, **izvršenja platnih transakcija** (izravnih terećenja, platnih transakcija putem platnih kartica i sl. te kreditnih transfera), uključujući prijenos novčanih sredstava na račun za plaćanje kod korisnikovog pružatelja platnih usluga ili kod drugog pružatelja platnih usluga, izvršenja platnih transakcija u kojima su novčana sredstva pokrivena kreditnom linijom za korisnika platnih usluga izdavanja i/ili prihvaćanja platnih instrumenata, novčanih pošiljaka i izvršenja platnih transakcija kad se suglasnost platitelja za izvršenje platne transakcije daje nekim telekomunikacijskim sredstvom, digitalnim ili informatičko-tehnološkim uređajem, a plaćanje se obavlja telekomunikacijskom ili mrežnom operatoru ili operatoru informatičko-tehnološkog sustava, koji djeluje isključivo kao posrednik između korisnika platnih usluga i dobavljača robe i usluga,
- **platna transakcija** – transakcija kako je definirana zakonom kojim se uređuje platni promet,
- **potrošač** – fizička osoba koja sklapa ugovor o izdavanju elektroničkog novca izvan područja svoje gospodarske djelatnosti ili slobodnog zanimanja.

⁴² Zakon o elektroničkom novcu, Narodne novine, br. 64/2018.

⁴³ Zakon o institucijama za elektronički novac, Narodne novine, br. 117/2008. – nevažeći

⁴⁴ Zakon o elektroničkom novcu, Narodne novine, br. 139/2010. – nevažeći

Osim navedenih zakona, u određenim slučajevima treba uzeti u obzir i **Zakon o elektroničkim medijima**⁴⁵. Iako se većinom bavi pravima, obvezama i odgovornostima pružanja audio i audiovizualnih medijskih usluga, u jednom dijelu uređuje i usluge **elektroničkih publikacija** putem elektroničkih komunikacijskih mreža i elektroničkih medija. Pod elektroničke se publikacije smatraju urednički oblikovani programski sadržaji koji se objavljuju putem interneta, što uključuje internetske stranice i portale, pa to svakako pripada uslugama informacijskog društva i razmjene podataka, odnosno informacija.

1.6.2 Pravni okvir elektroničkih komunikacija

U kontekstu elektroničkih i telekomunikacija u općem smislu zakonodavac se također brine o pravnim okvirima kojima se definiraju osnovni pojmovi, prava i odgovornosti u **Zakonu o elektroničkim komunikacijama**⁴⁶, koji je u potpunosti zamijenio bivši Zakon o telekomunikacijama⁴⁷. Ovaj Zakon uređuje područje elektroničkih komunikacija, od korištenja elektroničkih komunikacijskih mreža i pružanja elektroničkih komunikacijskih usluga, preko pružanja univerzalnih usluga, korištenja elektroničke komunikacijske infrastrukture, prava i obveza sudionika na tržištu elektroničkih komunikacijskih mreža i usluga, adresiranja, numeriranja i upravljanja radiofrekvencijskim spektrom, digitalnog radija i televizije, zaštite podataka i sigurnosti elektroničkih komunikacija, do djelovanja nacionalnoga regulatornog tijela za elektroničke komunikacije i poštanske usluge.

Kako ne bismo ulazili u potankosti definicija mnogobrojnih pojmova definiranih ovim zakonom, nabrojani su samo neki osnovni pojmovi poznatih naziva, kao što su djelatnost elektroničkih komunikacijskih mreža i usluga, dodjela brojeva i adresa, elektromagnetska kompatibilnost (EMC), elektronička komunikacijska mreža, infrastruktura, oprema i usluga, elektronička pošta, elektroničke komunikacije i vodovi, elektronički programski vodiči (EPG), javna komunikacijska mreža i usluga, javna telefonska mreža i usluga, međupovezivanje, multipleks, radijske komunikacije i televizija. Uz ovaj se Zakon vežu mnogi podzakonski akti iz područja telekomunikacija koje ovdje nećemo eksplicitno navoditi, a kojima možete pristupiti pomoću popisa propisa vezanih uz elektroničke komunikacije na internetskim stranicama Ministarstva mora, prometa i infrastrukture⁴⁸.

1.6.3 Pravni okvir informacijske sigurnosti, zaštite osobnih podataka i zaštite intelektualnog vlasništva

Osnovni propis koji utvrđuje zakonske okvire informacijske sigurnosti donesen je 13. srpnja 2007. godine i zove se **Zakon o informacijskoj sigurnosti**⁴⁹, a utvrđuje osnovne pojmove, mjere, područja i norme informacijske sigurnosti te odgovornosti nadležnih tijela. Tako se u Zakonu o informacijskoj sigurnosti definiraju:

⁴⁵ Zakon o elektroničkim medijima, Narodne novine, br. 153/2009., 84/2011. i 94/2013.

⁴⁶ Zakon o elektroničkim komunikacijama, Narodne novine, br. 73/2008., 90/2011., 133/2012., 80/2013., 71/2014. i 72/2017.

⁴⁷ Zakon o telekomunikacijama, Narodne novine, br. 122/2003., 158/2003., 60/2004. i 70/2005. – nevažeći

⁴⁸ Ministarstvo mora, prometa i infrastrukture, <http://www.mmpi.hr/>

⁴⁹ Zakon o informacijskoj sigurnosti, Narodne novine, br. 79/2007.

- **informacijska sigurnost** – stanje povjerljivosti, cjelovitosti i raspoloživosti podatka, koje se postiže primjenom propisanih mjera i normi te organizacijskom podrškom za poslove planiranja, provedbe, provjere i dorade mjera i normi
- **mjere informacijske sigurnosti** – opća pravila zaštite podataka koja se realiziraju na fizičkoj, tehničkoj ili organizacijskoj razini, a obuhvaćaju nadzor pristupa i postupanja s klasificiranim podacima (klasificirani podaci su objašnjeni u nastavku kod Zakona o tajnosti podataka), postupanje prilikom neovlaštenog otkrivanja i gubitka klasificiranih podataka, planiranje mjera prilikom izvanrednih situacija i ustrojavanje posebnih fondova podataka za klasificirane podatke
- **norme informacijske sigurnosti** – organizacijske i tehničke procedure i rješenja namijenjena sustavnoj i ujednačenoj provedbi propisanih mjera informacijske sigurnosti
- **sigurnosna provjera** – utvrđuje mjere i norme koji se primjenjuju na osobe koje imaju pristup klasificiranim podacima,
- **sigurnost podatka** – utvrđuje mjere i norme koje se primjenjuju kao opće zaštitne mjere za prevenciju, otkrivanje i otklanjanje štete od gubitka ili neovlaštenog otkrivanja klasificiranih i neklasificiranih podataka,
- **sigurnost informacijskog sustava** – utvrđuje mjere i norme klasificiranog i neklasificiranog podatka koji se obrađuje, pohranjuje ili prenosi u informacijskom sustavu te zaštite cjelovitosti i raspoloživosti informacijskog sustava u procesu planiranja, projektiranja, izgradnje, uporabe, održavanja i prestanka rada informacijskog sustava,
- **sigurnost poslovne suradnje** – primjena propisanih mjera i normi za provedbu natječaja ili ugovora s klasificiranom dokumentacijom koji obvezuju pravne i fizičke osobe,
- **sigurnosna akreditacija informacijskog sustava** – postupak u kojem se utvrđuje osposobljenost tijela i pravnih osoba za upravljanje sigurnošću informacijskog sustava, a provodi se utvrđivanjem primijenjenih mjera i standarda informacijske sigurnosti,
- **tijela za informacijsku sigurnost:**
 - **Ured Vijeća za nacionalnu sigurnost** – središnje državno tijelo za informacijsku sigurnost koje koordinira i usklađuje donošenje i primjenu mjera i standarda informacijske sigurnosti u RH te u razmjeni klasificiranih i neklasificiranih podataka između RH i stranih zemalja i organizacija,
 - **Zavod za sigurnost informacijskih sustava** – središnje državno tijelo za tehnička područja sigurnosti informacijskih sustava u tijelima i pravnim osobama, a to su norme sigurnosti informacijskih sustava, sigurnosne akreditacije informacijskih sustava, upravljanje kriptomaterijalima koji se koriste u razmjeni klasificiranih podataka te koordinacija prevencije i odgovora na računalne ugroze sigurnosti informacijskih sustava,
 - **CERT** – nacionalno tijelo za prevenciju i zaštitu od računalnih ugroza sigurnosti javnih informacijskih sustava u RH.

Zakonski okvir pristupa klasificiranim podacima, koji imaju neki oblik „državne tajne“, također je donesen u 2007. godini pod nazivom **Zakon o tajnosti podataka**⁵⁰ te utvrđuje pojmove klasificiranih i neklasificiranih podataka, stupnjeve tajnosti, postupke klasifikacije i deklasifikacije, pristupe klasificiranim i neklasificiranim podacima, njihovu zaštitu i nadzor nad provedbom Zakona. Zakon o tajnosti podataka definira:

- **podatak** – dokument, odnosno svaki napisani, umnoženi, nacrtani, slikovni, tiskani, snimljeni, fotografirani, magnetni, optički, elektronički ili bilo koji drugi zapis podatka, saznanje, mjera, postupak, predmet, usmeno priopćenje ili informacija koja s obzirom na svoj sadržaj ima važnost povjerljivosti i cjelovitosti za svog vlasnika,
- **klasifikaciju podatka** – postupak utvrđivanja jednog od stupnjeva tajnosti podatka s obzirom na stupanj ugroze i područje,
- **deklasifikaciju podatka** – postupak kojim se utvrđuje prestanak postojanja razloga zbog kojih je određen podatak klasificiran odgovarajućim stupnjem tajnosti, nakon čega podatak postaje neklasificirani s ograničenom uporabom samo u službene svrhe,
- **klasificirani podatak** – podatak koji je nadležno tijelo (u propisanom postupku) takvim označilo i za koji je utvrđen stupanj tajnosti, kao i podatak kojeg je RH tako označenog predala druga država, međunarodna organizacija ili institucija, sa stupnjevima tajnosti: **VRLO TAJNO** – nepopravljiva šteta nacionalnoj sigurnosti i vitalnim interesima RH, **TAJNO** – teška šteta, **POVJERLJIVO** – šteta i **OGRANIČENO** – šteta djelovanju i izvršavanju zadataka,
- **neklasificirani podatak** – podatak bez utvrđenog stupnja tajnosti, koji se koristi u službene svrhe, kao i podatak koji je RH predan tako označen,
- **vlasnika podatka** – nadležno tijelo u okviru čijeg je djelovanja nastao klasificirani ili neklasificirani podatak,
- **certifikat** – uvjerenje o sigurnosnoj provjeri koje omogućuje pristup klasificiranim podacima.

Uz područje Zakona o tajnosti podataka vežu se i mnog drugi podzakonski akti.

Treći važniji zakon iz područja sigurnosti i zaštite je **Zakon o zaštiti osobnih podataka**⁵¹, koji uređuje zaštitu osobnih podataka o fizičkim osobama te nadzor nad prikupljanjem, obradom i korištenjem osobnih podataka u Hrvatskoj radi zaštite privatnog života i ostalih ljudskih prava i temeljnih sloboda u prikupljanju, obradi i korištenju osobnih podataka. **Osobnim podacima** smatra se svaka informacija koja se odnosi na identificiranu fizičku osobu, odnosno osobu čiji se identitet može utvrditi izravno ili neizravno, posebno na osnovi identifikacijskog broja ili jednog ili više obilježja specifičnih za njezin fizički, psihološki, mentalni, gospodarski, kulturni ili socijalni identitet.

U Hrvatskoj postoji i **Zakon o pravu na pristup informacijama**⁵², koji uređuje pravo na pristup informacijama i ponovnu uporabu informacija koje posjeduju tijela javne vlasti, a propisuje i načela prava, ograničenja prava i postupak za ostvarivanje i zaštitu prava na pristup informacijama i ponovnu uporabu informacija, te djelokrug, način rada i uvjete za imenovanje i razrješenje Povjerenika za informiranje, kao i inspeksijski nadzor.

⁵⁰ Zakon o tajnosti podataka, Narodne novine, br. 79/2007. i 86/2012.

⁵¹ Zakon o zaštiti osobnih podataka, Narodne novine, br. 103/2003., 118/2006., 41/2008., 130/2011. i 106/2012.

⁵² Zakon o pravu na pristup informacijama, Narodne novine, br. 25/2013. i 85/2015.

Uz jedan dio zaštite prava koja su vezana uz komunikacije, podatke i njihovu razmjenu veže se i **Zakon o autorskom pravu i srodnim pravima**⁵³, koji uređuje autorska prava autorskih djela, među koja pripadaju pisana djela i računalni programi te prava proizvođača baza podataka. **Računalni programi** zaštićeni su kao jezična djela ako su izvorni u smislu da predstavljaju vlastitu intelektualnu tvorevinu svog autora. Pojam računalni program obuhvaća izražaj računalnog programa u bilo kojem obliku, uključujući i pripremni dizajnerski materijal, ali ne i ideje i načela na kojima se zasniva bilo koji element računalnog programa, uključujući i ona na kojima se zasnivaju njegova sučelja. **Bazama podataka** smatraju se zbirke samostalnih djela, podataka ili druge građe u bilo kojem obliku, koje su uređene po određenom sustavu ili metodi i pojedinačno dostupni elektroničkim ili drugim sredstvima, pri čemu postizanje, verifikacija ili predstavljanje sadržaja zahtijeva kvalitativno i/ili kvantitativno znatno ulaganje koje se, primjerice, sastoji u sredstvima, utrošenom vremenu i uloženom trudu.

Bitan je još i **Zakon o patentu**⁵⁴, koji uređuje sustav zaštite izuma patentom. Patent je isključivo pravo koje štiti nositelja patenta u pogledu gospodarskog iskorištavanja izuma. Kao patent se priznaje svaki izum iz bilo kojeg područja tehnike koji je nov, ima inventivnu razinu i koji se može industrijski primijeniti, ali ne i otkrića, znanstvene teorije, matematičke metode, pravila, upute i metode za izvođenje umnih aktivnosti ili za obavljanje poslova, prikazivanje informacija i računalni programi. No, niz izuma koji se mogu međunarodno patentirati ipak dolazi iz domene elektroničkog poslovanja i razmjene informacija, uključujući telekomunikacijske tehnologije i oblike zapisa podataka.

1.7 Strategije interoperabilnosti

Svaka država zadužena je za stvaranje nacionalne strategije informatizacije te posljedično i interoperabilnosti. Hrvatska je od samih početaka bila vrlo aktivna u ovom području, vjerojatno uvjetovano i činjenicom da smo zbog ratnih razaranja kasnili u odnosu na zapadnu Europu, a postojala je realna potreba da se određene usustave i usklade s ostalim zemljama, pa se djelomično djelovalo u skladu s već donesenim preporukama, pogotovo na razini Europske unije.



Još 1991. godine započele su prve naznake informatizacije državne uprave, pravosuđa i javnih djelatnosti, a većinom su spašene i stare državne baze podataka i programska oprema od ratnih razaranja. U 90-tim godinama prošlog stoljeća u državnoj upravi je provedena standardizacija aplikacija, uvedeni su sustavi ERP, uveden je sustav elektroničke pošte i sagrađena moderna nacionalna akademska mreža CARNet te niz drugih aktivnosti.

Početkom stoljeća osnovan je **Ured za internetizaciju**, objedinjen je proces javne nabave, izvedeni su projekti modernizacije carinske uprave, sustava porezne uprave, sustava državne riznice, sustava (nove) osobne iskaznice, sustava graničnih prijelaza, sustava za gospodarenje i upravljanje prostorom RG, sustava interoperabilnosti katastra i zemljišnih

⁵³ Zakon o autorskom pravu i srodnim pravima, Narodne novine, br. 167/2003., 79/2007., 80/2011., 141/2013., 127/2014., 62/2017. i 96/2018.

⁵⁴ Zakon o patentu, Narodne novine, br. 173/2003., 87/2005., 76/2007., 30/2009., 128/2010., 49/2011., 76/2013. i 46/2018.

knjiga, institucionalnog okvira za internetizaciju, računalne i komunikacijske mreže tijela državne uprave, sustava obračuna plaća tijela državne uprave, sustava za upravljanje državnom imovinom te je započet projekt strategije zdravstva i višega stupnja primjene ICT za izgradnju nacionalne mreže institucionalnoga i izvaninstitucionalnoga obrazovanja. Paralelno s navedenim aktivnostima donesen je niz zakona koji su potomje opisani u poglavlju 0, a 2002. godine predstavljena je i prva **strategija "Informacijske i komunikacijske tehnologije – Hrvatska u 21. stoljeću"**⁵⁵.

Nakon osnivanja **Središnjeg državnog ureda za e-Hrvatsku** u 2004. godini usvojen je prvi put **Operativni plan provedbe „Programa e-Hrvatska 2007.“**⁵⁶, a pokrenut je i program **One-Stop-Shop** i otvoren servis **HITRO.HR**, te je započelo povezivanje tijela državne uprave u jedinstvenu komunikacijsku mrežu **HITRONet**. Uslijedilo je usvajanje Nacionalnog programa informacijske sigurnosti u RH, pokrenut je projekt **HITROREZ**, a prikazani su i prvi rezultati Središnjeg državnog ureda za e-Hrvatsku. Tada je prvi put izrađena i studija razvoja informacijskog društva u RH za 2005. na temelju koje su planirani daljnji koraci. 2007. godine je pokrenut Središnji **portal državne uprave "Moja uprava"** i projekt e-Otoci, a Hrvatska pristupa **programu ICT PSP** te se hrvatske ICT tvrtke prvi put zajednički pojavljuju i predstavljaju Hrvatsku na sajmu CeBIT.

U 2008. godini započinje izrada **Strategije razvoja elektroničke uprave za razdoblje 2009.-2012.**, a usvojena je i Strategija prelaska s analognog na digitalno emitiranje televizijskih programa u Republici Hrvatskoj tzv. DVB-T (*Digital Video Broadcasting - Terrestrial*). Krajem prvog desetljeća uključujemo se u projekt Europske komisije "*eGovernment Benchmarking*" kojim se mjeri stupanj razvoja elektroničke uprave u zemljama EU, a izrađuje se i Analiza hrvatske ICT industrije za proteklo desetljeće. Također se pokreće Standardni projekt elektroničkog uredskog poslovanja i program e-Ured, te Hrvatska pristupa **programu ISA**. 2010. godina je posebno zanimljiva jer se tada predlaže i usvaja **Hrvatski okvir za interoperabilnost** kao preduvjet za poboljšanje komunikacije i međusobne suradnje poslovnih sustava tijela državne uprave.

U drugom desetljeću ovog stoljeća započela koordinacija provedbe **Digitalne agende za Europu 2020.**, predstavljen je informacijski sustav Evidencije ulaganja u RH i pokrenuta inicijativa *Going Local*, usvojena Odluka o utvrđivanju ciljeva razvoja elektroničke uprave u tijelima državne uprave za razdoblje do 2015. godine, te predstavljen Koncept arhitekture umrežene uprave. U mandatu Vlade RH 2011.-2015. organizacijski su spojeni Ministarstvo uprave i Središnji državni ured za e-Hrvatsku čime su u nadležnost ministarstva dodani i poslovi informatizacije i modernizacije. Na taj su način u novu **Upravu za e-Hrvatsku** pri Ministarstvu uprave spojene odvojene cjeline reforme državne uprave i informatizacije.

Ustroj Uprave za e-Hrvatsku sastoji se od Sektora za modernizaciju i informatizaciju državne uprave unutar koje djeluje i služba za elektroničke usluge i registre te Sektora za infrastrukturu unutar kojeg su Služba za razvoj i standardizaciju te Služba za podršku korisnicima. Ključnu ulogu imalo je **Povjerenstvo za koordinaciju informatizacije javnog sektora**, osnovano 2012. godine, koje čine najviši dužnosnici Vlade Republike Hrvatske sa zadaćom usmjeravanja razvoja i koordinacije svih poslova i projekata primjene informacijske i

⁵⁵ Informacijske i komunikacijske tehnologije – Hrvatska u 21. stoljeću, MZOŠ, 2004.

⁵⁶ Operativni plan provedbe „Programa e-Hrvatska 2007.“, Središnji državni ured za e-Hrvatsku, 2004.

komunikacijske tehnologije u javnom sektoru, a u cilju racionalizacije sustava i povećanja kvalitete javnih usluga.

Glavne aktivnosti Povjerenstva su:

- optimizacija korištenja elektroničke komunikacijske infrastrukture za potrebe javne uprave, s naglaskom na iskorištavanje resursa u vlasništvu Republike Hrvatske,
- stvaranje registarski uređene države povezivanjem baza podataka,
- objedinjavanje i međusobno dijeljenje informatičkih resursa (hardver, softver i znanje) u javnoj upravi,
- modernizacija zakonodavnog okvira za bržu i efikasniju primjenu informacijske i komunikacijske tehnologije u javnoj upravi,
- provedba objedinjenih postupaka javne nabave za robe i usluge iz područja ICT-a,
- jačanje međunarodnih aktivnosti u području e-uprave i stvaranje uvjeta za bolje iskorištavanje stranih izvora financiranja za ICT projekte.

Naravno, ne očekuje se da Povjerenstvo za koordinaciju informatizacije javnog sektora samostalno radi na tim aktivnostima, već je zaduženo za osnivanje niza **radnih skupina** koje rade na konkretnim projektima, vezano uz temeljne registre i elektroničku razmjenu podataka, elektronički identitet i autentikaciju, standardizaciju rješenja za korisnički pristup, infrastrukturu, primjenu otvorenoga koda i otvorenih normi i definiranje područja i tema za objedinjenu javnu nabavu.

U **Strateškom planu Ministarstva uprave za razdoblje od 2014. do 2016. godine**⁵⁷ eksplicitno je naveden cilj unaprjeđenja kvalitete javnih usluga kroz djelotvornu primjenu ICT-a u radu javne uprave kroz:

- razvoj i standardizaciju informacijske i komunikacijske infrastrukture,
- poboljšanje dostupnosti i kvalitete elektroničkih usluga i rješenja,
- unaprjeđenje sustava podrške za koordinaciju razvoja cjelovitih integriranih usluga,
- učinkovito korištenje instrumenata kohezijske politike i programa EU.

Potanje, planirani načini ostvarenja tih ciljeva ostvaruju se kroz sljedeće aktivnosti:

- optimiran razvoj komunikacijske infrastrukture HITRONet,
- optimiran razvoj računalne infrastrukture u vlasništvu RH („oblak javnog sektora“),
- donošenje strategije za otvorene specifikacije i standarde,
- usklađivanje Hrvatskog okvira za interoperabilnost s EIF 2.0 (potpoglavlja 1.8.1 i 1.8.4),
- omogućavanje korištenja zajedničkih interoperabilnih rješenja na razini EU,
- nadogradnja sustava Mojauprava.hr,
- redovito ažuriranje popisa službenih obrazaca javnopravnih tijela,
- primjena elektroničkih identiteta u elektroničkim javnim uslugama,
- uspostava elektroničke razmjene podataka između registara u javnoj upravi,
- puna primjena elektroničkog uredskog poslovanja,
- priprema popisa usluga javnog sektora, odabir i priprema u obliku integralnih e-usluga,
- priprema pregleda procesa državne uprave i gotovih *e-government* rješenja,

⁵⁷ Strateški plan Ministarstva uprave za razdoblje od 2014. do 2016. Godine, Ministarstvo uprave RH, 2013., <http://www.uprava.hr/UserDocsImages/Strate%C5%A1ki%20plan%202014%20-%202016%20.pdf>

- realizacija pristupa osobnim informacijama iz javne uprave za građane,
- sustavna podrška izgradnju korisnički-orientiranih elektroničkih usluga javne uprave,
- unaprjeđenje organizacijske strukture i razvoja ljudskih potencijala za ICT stručnjake u javnoj upravi,
- uspostava sustava podrške zajedničkih rješenja tijela javne uprave i obrazovnih programa za javnu upravu,
- primjena zajedničkih metodologija za standardizaciju ICT-a (katalozi metapodataka),
- uspostava sustava podrške za ponovno korištenje javnih informacija,
- poticanje osposobljavanja i uključivanja svih društvenih skupina za korištenje elektroničkih usluga i sudjelovanje u pripremu i donošenju odluka javne uprave RH i EU
- međusobno povezivanje sustava za podršku korisnicima javne uprave RH i EU
- priprema Programa razvoja elektroničkih usluga
- uspostavljanje sustava za praćenje projekata u državnoj upravi
- priprema programskih dokumenata i projekata za financiranje putem instrumenata kohezijske politike za financijske perspektive 2014.-2020.
- aktivno sudjelovanje Ministarstva uprave u projektima na razini UE te Programima EU vezanim uz realizaciju Digitalne agende za Europu
- priprema i provedba projekta projekata i pretpristupnog instrumenta IPA te tzv. Prijelaznog instrumenta (engl. *Transition Facility*).

Kao što se može vidjeti popis aktivnosti je bio prilično dugačak, a ovdje ćemo navesti najznačajnije projekte, od kojih smo neke u sljedećim potpoglavljima i potonje opisali:

- projekt **HITRONet** – računalno komunikacijska mreža javnopravnih tijela,
- projekt **Digitalna agenda za Europu**,
- projekt **e-Građani**, uključivo prijedlog projekta **NIAS** (nacionalni identifikacijski i autentifikacijski sustav) i prijedlog projekta **OKP** (sustav osobnog korisničkog pretinca),
- program **Interoperabilna rješenja za europsku javnu upravu ISA**
- **Program podrške politikama za primjenu informacijskih i komunikacijskih tehnologija CIP ICT PSP** (*Competitiveness and Innovation Framework Programme: The Information and Communication Technologies Policy Support Programme*),
- projekt objedinjavanja optičke infrastrukture u tvrtkama u pretežitom vlasništvu RH
- projekti vezani uz temeljne registre i elektroničku razmjenu podataka, uključivo projekt analize stanja sustava OIB, projekt Zakona o registrima i projekt uspostave sustava temeljnih registara i e-razmjene podataka
- projekti vezani za elektronički identitet i autentikaciju te standardizaciju rješenja za korisnički pristup.

Posljednja strategija koja je u trenutku pisanja još uvijek aktualna je "**Strategija razvoja javne uprave za razdoblje od 2015. do 2020. godine**"⁵⁸ koja predstavlja okvir za razvoj javne uprave i usmjerena je na unaprjeđenje upravnih kapaciteta te na bolju organizaciju javne uprave. Vremenski okvir donošenja, a donekle i sadržaj ove strategije, povezan s ispunjenjem preduvjeta za korištenje sredstava europskih fondova za razdoblje 2014. – 2020., tematski cilj 11 – Jačanje institucionalnih kapaciteta javnih tijela i zainteresiranih strana te učinkovite javne uprave. S druge strane strategija ima cilj uskladiti razvoj javne uprave s ciljevima Europe 2020.,

⁵⁸ Strategija razvoja javne uprave za razdoblje od 2015. do 2020. godine, Narodne novine br. 70/2015.

strategijom Europske unije kojom se želi poduprijeti razvoj pametnog, održivog i uključivog gospodarstva do 2020. godine, pa će se tako razvoj javne uprave odvijati u tri glavna smjera:

- pojednostavnjenje i modernizacija upravnog postupanja, kao i osiguravanje pouzdane i brze podrške javne uprave građanima i poslovnim subjektima realizacijom projekata e-uprave
- unaprjeđenje sustava razvoja i upravljanja ljudskim potencijalima radi stvaranja moderne javne službe
- reforma upravnog sustava sukladno najboljoj praksi i iskustvima dobrog upravljanja prema europskim standardima.

Iz navedenog se može vidjeti da se u Hrvatskoj već dulje vrijeme značajno promišlja o problematici interoperabilnosti i puno je toga napravljeno, pogotovo u dijelu javne uprave, no usporedbom s razvijenim zemljama rezultati su ipak ograničenog opsega, a neki koraci nisu izvedeni do kraja ili na vrijeme, te je tek za vidjeti kako će se u budućnosti ove ideje pretvoriti u konkretne aktivnosti i kako će se mjere provoditi i utjecati na svakodnevno poslovanje. U nastavku ćemo potanje obraditi neke od ostvarenih projekata i programa.

1.7.1 HITRONet

HITRONet⁵⁹ je komunikacijska mreža za povezivanje različitih javnopravnih tijela putem zajedničke računalno komunikacijske infrastrukture, a ujedno predstavlja integralni dio projekta e-Uprava i temeljnu infrastrukturu za daljnji razvoj elektroničkih usluga jer omogućava bolju komunikaciju između javnopravnih tijela.

Osnovna **svrha** mreže HITRONet je **integriranje državnih informacijskih resursa kroz sigurnu privatnu širokopojasnu infrastrukturu**, što obuhvaća povezivanje središnjih i udaljenih lokacija javnopravnih tijela na zajedničku podatkovnu mrežu radi efikasnijeg i jeftinijeg rada tijela državne uprave i razmjene elektroničkih podataka. Dodatno, HITRONet korisnicima omogućuje siguran i strogo kontroliran pristup te povezivanje mreže na internet i uspostavu standardnih mrežnih servisa.

Kroz mrežu HITRONet povezano je 46 središnjih lokacija javnopravnih tijela te preko 400 udaljenih lokacija javnopravnih tijela (pravosudnih tijela – sudova, ureda državne uprave u županijama, ispostava HZZO-a itd.), a na mreži su, pored uobičajenih mrežnih usluga pristupa zajedničkom izlazu na Internet putem internetskog priključka kapaciteta od 1 Gb/s, implementirani i posebni servisi isključivo za potrebe javnopravnih tijela, kao što su Sustav državne riznice, OIB sustav, e-Spis i slično. Pored navedenog, u HITRONetu je uspostavljena i razmjena internetskog prometa putem sustava CIX (*Croatian Internet eXchange*) vezom kapaciteta od 1Gb/s i internetskim prometom (engl. *Internet peering*) sa 17 od ukupno 25 članica. Još 2009. godine HITRONet je povezan i sa europskom podatkovnom mrežom sTESTA, posebnom mrežom kojoj je svrha povezivanje tijela na razini Europske unije.

⁵⁹ HITRONet, <https://uprava.gov.hr/o-ministarstvu/ustrojstvo/uprava-za-e-hrvatsku/aktualni-projekti/hitronet/873>

1.7.2 Digitalna agenda za Europu

U okviru Strategije Europa 2020, kao jednu od sedam ključnih inicijativa, Europska komisija je 2010. objavila priopćenje koje sadrži inicijativu pod nazivom **Digitalna agenda za Europu**⁶⁰ ili **DAE** (*Digital Agenda for Europe*). Sveopći je cilj ove inicijative ostvariti održive gospodarske i društvene pogodnosti na jedinstvenom digitalnom tržištu utemeljenom na brzom i ultrabrzom internetu te interoperabilnim aplikacijama.

Procjenjuje se da digitalna ekonomija raste 7 puta brže od ostatka gospodarstva. Projekt DAE ima za cilj pomoću europskim tvrtkama i građanima da maksimalno iskoriste digitalne tehnologije. Inicijalni prijedlog sadržavao je 101 akciju grupiranu u **sedam prioriternih područja djelovanja**:

- **jedinstveno digitalno tržište,**
- **interoperabilnost i normiranje** informacijskih i komunikacijskih proizvoda i usluga,
- **povjerenje i sigurnost** na Internetu,
- **brzi i izuzetno brzi pristup Internetu,**
- **istraživanje i razvoj** (poticanje ulaganja),
- **poboljšanje digitalne pismenosti, vještina i uključenosti,**
- **koristi korištenjem ICT-a za društvo EU** (primjena informacijskih i komunikacijskih tehnologija u rješavanju ključnih izazova društva, kao što su klimatske promjene, povećanje troškova zdravstvene skrbi i starenje stanovništva).

Na europskoj razini, Europska komisija obvezuje se poduzeti slijedeće:

- osigurati stabilan pravni okvir koji će stimulirati ulaganja u otvorenu i konkurentnu brzu internetsku infrastrukturu i pripadajuće usluge,
- razviti učinkovite politike, stvoriti jedinstveno tržište za *online* sadržaj i usluge (prekogranične i sigurne web usluge EU i tržište digitalnih sadržaja, s visokom razinom povjerenja, uravnotežen regulatorni okvir s jasnim pravima režima, poticanje multiteritorijalnih dozvola, odgovarajuću zaštitu i naknadu za nositelje prava i aktivnu podršku za digitalizaciju europske bogate kulturne baštine te oblika globalnog vladanja na internetu),
- reformirati fondove za istraživanje i inovaciju i povećati podršku u području poslovnog ICT sektora, kako bi se povećala tehnološka snaga EU na ključnim područjima i stvorili uvjeti za visok rast SME-a uz nastajanje novih tržišta i ICT inovacija.

Države članice koje su prihvatile Digitalnu agendu za Europu, a među njima je i Hrvatska, dužne su:

- izraditi operativne strategije brzog interneta i osigurati javno financiranje, uključujući strukturalne fondove za područja koja nisu pokrivena privatnim investicijama,
- uspostaviti pravni okvir za koordinaciju javnih radova u vezi s smanjivanjem troškova iskorištenja mreža (engl. *network rollout*),
- promicati implementacije i korištenje modernih *online* usluga (e-Government, online zdravstvo, pametne kuće, digitalne vještine, sigurnost i sl.)

⁶⁰ Digital Agenda for Europe, <http://ec.europa.eu/digital-agenda>

Neki od ambicioznih ciljeva Digitalne agende od 2013. do 2020. godine su:

- 100% povećanje u javnoj potrošnji za ICT
- 99,9% populacije EU pokriveno brzim pristupom Internetu (fiksni i bežični, >30Mbps),
- 75% populacije EU redovno koristi Internet,
- 60% građana EU s posebnim potrebama redovno koristi Internet,
- 50% kućanstava EU koristi brzi pristup Internetu,
- 50% populacije EU koristi *e-Government*,
- 50% populacije EU koristi *online* trgovinu,
- 30% malih i srednjih poduzeća prodaje *online*,
- 20% populacije EU koristi prekograničnu *online* trgovinu,
- *roaming* po nacionalnim cijenama.

Hrvatska sudjeluje u Digitalnoj agendi kroz niz projekata i aktivnosti, a dosad su najistaknutiji projekti povezivanja registara, povećanja transparentnosti informacija, stvaranje novog i stabilnog okruženja regulacije širokopojsnog pristupa Internetu, projekt e-Građani, ali i projekti e-Upisa i e- Zdravstva.

1.7.3 Projekt e-Građani

Projekt **e-Građani** jedan od najvažnijih nacionalnih informatičkih projekata u Republici Hrvatskoj koji omogućava elektroničku komunikaciju građana s javnim sektorom na jednom mjestu na Internetu, putem portala koji objedinjava razne informacije o javnim uslugama i radu Vlade RH i ministarstava te omogućava siguran pristup elektroničkim uslugama korištenjem elektroničkog identiteta posredstvom jedne ili više prihvatljivih vjerodajnica za elektroničku identifikaciju. Sustav je uspostavljen je s ciljem modernizacije, pojednostavljenja i ubrzanja napredne i sigurne komunikacije građana i javnog sektora te povećanja transparentnosti pružanja javnih usluga.



Projekt e-Građani bio je inicijalno predložen od strane Povjerenstva za koordinaciju informatizacije javnog sektora i započeo je s radom 2014. godine Odlukom o pokretanju projekta e-Građani⁶¹, a realiziran je u tri faze, suradnjom Ureda predsjednika Vlade RH i Ministarstva uprave, s agencijama FINA⁶² i APIS-IT⁶³ te tijelima javnog sektora Republike Hrvatske.

Sustav je uspostavljen kako bi tijela javnog sektora putem odgovarajuće web aplikacije i web servisa komunicirala s javnošću te sa svojim korisnicima prilikom:

- objave javnih informacija,
- pružanja korisnički orijentiranih i personaliziranih tzv. "*Single-Sign-On*" e-usluga korisnicima i/ili
- slanja osobnih službenih poruka korisnicima vezanih za javne usluge, postupke i osobne statute.

⁶¹ Odluka o pokretanju projekta e-Građani, Narodne novine, br. 52/2013.

⁶² FINA – Financijska agencija, <http://www.fina.hr/>

⁶³ Agencija za podršku informacijskim sustavima i informacijskim tehnologijama, <http://www.apis-it.hr>

Iz tog razloga definirane su i izvedene tri glavne sastavnice zajedničke infrastrukture javnog sektora sustava e-Građani, a to su:

- gov.hr – sustav Središnjeg državnog portala,
- NIAS – nacionalni identifikacijski i autentikacijski sustav,
- OKP – sustav Osobnog korisničkog pretinca.

1.7.3.1 Elektroničke usluge u sustavu e-Građani

Putem sustava e-Građani svim građanima dostupne su mnoge usluge, primjerice⁶⁴:

- zatražiti elektroničke izvode iz matične knjige rođenih, vjenčanih ili knjige državljana
- zatražiti elektroničke zapise uvjerenja o prebivalištu, boravištu te vlasništvu cestovnih vozila
- predati zahtjev za izradu putovnice, vozačke dozvole
- obaviti prijavu/upis rođenja djeteta u maticu rođenih, evidenciju državljanstva, prijaviti prebivalište novorođenog djeteta, regulirati zdravstveni status novorođenog djeteta
- izvršiti uvid u ocjene svog djeteta u školi
- pregledati recepte koje ste realizirali u ljekarnama u zadnjih 6 mjeseci
- naručiti Europsku karticu zdravstvenog osiguranja
- zatražiti elektroničku radnu knjižicu
- registrirati se kao potencijalni posloprimac
- provjeriti uplaćene doprinose u drugi stup mirovinskog osiguranja
- pribaviti potvrde od REGOS-a
- provjeriti svoju poreznu knjigovodstvenu karticu, JOPPD, pregledati ukupne primitke, obračunate doprinose i poreze po pojedinim isplatiteljima
- zatražiti elektronički izvadak vlasničkog lista, kopiju katastarskog plana, prijepis/izvod iz posjedovnog lista, izvadak iz Baze zemljišnih podataka
- pregledati ukupne primitke, obračunate doprinose i poreze po pojedinim isplatiteljima
- provjeriti svoje podatke u OIB sustavu
- sudjelovati u procesima savjetovanja sa zainteresiranom javnošću
- pribaviti e-potvrdu koja pokazuje da se protiv korisnika ne vodi kazneni postupak za kaznena djela koja se progone po službenoj dužnosti.

Dostupne e-usluge u sustavu e-Građani⁶⁵ su iz sljedećih područja:

- Pravna država i sigurnost (11): e-Zahtjev za izdavanje ePutovnice, Izdavanje elektroničke isprave Grada Zagreba, Uvjerenje da se ne vodi kazneni postupak, Osobni korisnički pretinac, mojID, Uvjerenje iz kaznene evidencije, e-Prijava boravišta hrvatskih državljana, Registar birača - e-Privremeni upis, Suglasnosti i punomoći u postupcima iz djelokruga MUP-a, Registar birača, e-Usluge MUP-a
- Obitelj i život (4): E-usluge socijalna skrb, e-Matične knjige, e-Novorođenče, Kalkulator doplatka za djecu

⁶⁴ Usluge dostupne u sustavu e-Građani na dan 31.1.2020.

⁶⁵ Usluge dostupne u sustavu e-Građani na dan 31.1.2020.

- Odgoj i obrazovanje (7): e-Razmjena studentskih ocjena, ePodnesak Ministarstva znanosti i obrazovanja, Online Tečajevi Srca, e-Dnevnik za roditelje, Home for Homeless servis u sustavu AAI@EduHr, e-Zapis o statusu studenta, Središnja prijava na diplomske studijske programe
- Promet i vozila (5): e-Nautika, e-Plovilo, Registracija operatora bespilotnih zrakoplova za kategorije B2 i C1, e-Zahtjev za izdavanje vozačke dozvole, Porezna prijava za obračun i plaćanje posebnog poreza na motorna vozila, Aktivno građanstvo (4), Moj račun - Gradska plinara Zagreb-Opskrba, e-prijavnice Ministarstva culture, MojZagreb, eSavjetovanja
- Branitelji (1): Predaja zahtjeva hrvatskih branitelja i članova obitelji
- Financije i porezi (3): SKDD e-Ulagatelj, Moj OIB, ePorezna
- Zdravlje (5): Realizirani recepti, Zahtjev za izdavanje Europske kartice zdravstvenog osiguranja (EKZO), Pregled izabranog liječnika, Portal zdravlja, Otvorene narudžbe
- Rad (9): e-Pomorac, Obvezni mirovinski fond (prijava/promjena), e-Osiguranje radničkih tražbina, Burza rada, Elektronički zapis o radno pravnom statusu (e-radna knjižica), Korisničke stranice HZMO-a, e-Potvrde iz mirovinskog sustava, Moj račun – REGOS, Sustav elektroničkih usluga REGOS-a
- Poslovanje (13): START - elektroničko pokretanje poslovanja, Postupci vezani uz članstvo Hrvatske komore arhitekata, e-Obrt, Postupci vezani uz članstvo u Hrvatskoj komori inženjera strojarstva, Postupci vezani uz članstvo u Hrvatskoj komori inženjera građevinarstva, Postupci vezani uz članstvo u Hrvatskoj komori inženjera elektrotehnike, Postupci vezani uz članstvo u Hrvatskoj komori ovlaštenih inženjera geodezije, e-Detektivi, e-Poljoprivreda, ePostupci u području intelektualnog vlasništva, e-Aplikacija za prijavu polaganja stručnog ispita za obavljanje stručnih geodetskih poslova, eVisitor, Registracija objekata koji pružaju uslugu smještaja strancima
- Stanovanje i okoliš (4): eDozvola - predaja zahtjeva za gradnju i prostorno uređenje, Vodne usluge Međimurskih voda, Zajednički informacijski sustav zemljišnih knjiga i katastra - ZIS OSS, Komunalne usluge i naknade

1.7.3.2 Sustav Središnjeg državnog portala gov.hr

Sustav Središnjeg državnog portala pod imenom **gov.hr** predstavlja jedinstveno mjesto za pristup javnim informacijama. Osnovna svrha središnjeg državnog portala je objedinjavanje informacija raznih državnih institucija kako bi građani na što jednostavniji i brži način došli do traženih informacija o javnim uslugama. Informacije su dostupne u različitim standardiziranim oblicima.



Za operativno vođenje i održavanje sustava gov.hr zadužen je APIS-IT.



Moja uprava e-Građani O Vladi i tijelima državne uprave

POSTANI e-GRADANIN - nema više čekanja u redovima

Moja uprava

Zdravlje Zdravstveno osiguranje, prava iz obveznog osiguranja, pravo na povrat troškova, zdravlje na radu, privremena nesposobnost za rad...	Obitelj i život Brak i izvanbračna zajednica, roditeljstvo, pomoć i savjetovanje, socijalna skrb, treća dob, smrt i nasljeđivanje, prava osoba s invaliditetom
Rad Traženje posla, radni odnos, prekid radnog odnosa, mirovine, nezaposleni, međuljudski odnosi...	Hrvatski branitelji Prava branitelja i obitelji, stambeno zbrinjavanje, olakšice i subvencije, zapošljavanje, zdravlje i invalidi, centri za pomoć, udruge i okupljanje
Državljanstvo i isprave Hrvatsko državljanstvo, isprave, potvrde i uvjerenja	Obrazovanje Predškolski odgoj, osnovno, srednje i visoko obrazovanje, e-Usluge...
Pravna država i sigurnost Pravna zaštita, prava potrošača, prevencija i prijava kaznenog djela, žrtve zločina i nestale osobe, javni red i mir, službe za zaštitu i spašavanje...	Stanovanje i okoliš Izgradnja i obnova kuće, kupnja, prodaja i najam nekretnine, vlasnička prava, područja od posebne državne skrbi, održivo gospodarenje otpadom...
Promet i vozila Registracija, dozvole za upravljanje vozilima, sigurnost na cesti, kupnja i prodaja vozila, cestarine	Poslovanje Pokretanje poslovanja, potpore poslodavcima, poljoprivreda, turizam, razvoj otoka, zaštita intelektualnog vlasništva, poslovni prostori
Financije i porezi Porezi i prijava poreza, plaćanje internetom, upravljanje dugovima, štednja i ulaganje, osiguranja	Aktivno građanstvo i slobodno vrijeme Udruge, volontiranje, na putovanju, šport, građani u političkom životu, kulturna događanja, u prirodi

Slika 1.6: Internetske stranice Središnjeg državnog portala sustava gov.hr

1.7.3.3 Nacionalni identifikacijski i autentikacijski sustav (NIAS)

Nacionalni identifikacijski i autentikacijski sustav NIAS je informacijsko-tehnološki sustav središnje identifikacije i autentikacije korisnika elektroničkih javnih usluga. Osnovni zadatak sustava NIAS je sigurna i pouzdana identifikacija i autentikacija korisnika koji putem odgovarajuće vjerodajnice mogu pristupiti javnim elektroničkim uslugama.



Vjerodajnica je sredstvo dokazivanja (prepoznavanja) elektroničkog identiteta koje je zaštićeno tehnološkim protokolima. U praksi vjerodajnica je nešto što korisnik zna i/ili posjeduje, primjerice korisničko ime i lozinka, token, digitalni certifikat i slično, koju korisnik treba pažljivo čuvati i ne povjeravati drugim osobama.

Svaki građanin Republike Hrvatske koji posjeduje prihvatljivu vjerodajnicu za elektroničku identifikaciju, ima u okviru sustava NIAS jedinstveni elektronički identitet kojim može pristupiti elektroničkim javnim uslugama i tako ostvariti elektroničku komunikaciju s tijelima javnog sektora. Osnovni zadatak NIAS-a je sigurna i pouzdana identifikacija i autentikacija korisnika koji putem odgovarajuće vjerodajnice pristupaju javnim elektroničkim uslugama.



Sustav NIAS od izdavatelja vjerodajnice preuzima i razmjenjuje **samo one podatke koji su nužni za jednoznačnu identifikaciju korisnika**, kao što je primjerice **OIB**, a tajnim podacima vezanim uz vjerodajnicu sustav NIAS ne može pristupiti te ih i dalje posjeduje i zna samo korisnik vjerodajnice. Posao uspostave i operativnog vođenja sustava NIAS povjeren je FINA-i koja odrađuje i integraciju sa ostalim sastavnicama.

Svaka elektronička usluga u sustavu e-Građani zahtijeva određenu razinu sigurnosti prilikom autentikacije korisnika. **Razina sigurnosti vjerodajnice** predstavlja stupanj ranjivosti vjerodajnice prilikom njezinog korištenja, a u sustavu NIAS razlikuju se četiri razine sigurnosti, pri čemu je najniža razina 1, a najviša je razina 4 je, što je usklađeno s preporukama i iskustvima stečenim kroz projekte u zemljama EU. Sustav NIAS prilikom prijave na pojedinu uslugu nudi korisniku na izbor samo one vjerodajnice koje udovoljavaju minimalno traženoj razini sigurnosti.



Definirane sigurnosne razine kao i kriteriji za ocjenu razine osiguranja kvalitete autentikacije, preuzeti su radi interoperabilnosti iz pilot **projekta EU STORK**, čija je svrha uspostavljanje paneuropske prekogranične autentikacije između država članica EU.

Lista prihvaćenih vjerodajnica⁶⁶ u sustavu NIAS dana je u sljedećoj tablici.

Tablica 1. Lista prihvaćenih vjerodajnica

Izdavatelj vjerodajnice	Vjerodajnica za NIAS	Sigur. razina	Status
MUP RH - Ministarstvo unutarnjih poslova	Elektronička osobna iskaznica (eOI)	4	Trajna
FINA - Financijska agencija	FinaCertRDC certifikat	4	Trajna
AKD - Agencija za komercijalnu djelatnost d.o.o.	Osobni identifikacijski certifikat NCP+	4	Trajna
AKD - Agencija za komercijalnu djelatnost d.o.o.	kID certifikat	4	Trajna
CARNet - Hrvatska akademska i istraživačka mreža	mToken za e-Građane	3	Trajna
FINA - Financijska agencija	FinaSoft certifikat	3	Trajna
HZZO - Hrvatski zavod za zdravstveno osiguranje	Pametna kartica s certifikatom	3	Trajna
HPB - Hrvatska poštanska banka d.d.	HPB token / mToken	3	Trajna
ZABA - Zagrebačka banka d.d.	ZABA token/mToken	3	Trajna
PBZ - Privredna banka Zagreb d.d.	mToken aplikacija / kartica / mToken	3	Trajna
RBA - Raiffeisenbank Austria d.d.	RBA token/mToken i CAP čitač	3	Trajna
KentBank d.d.	SMS jednokratni pin	3	Trajna

⁶⁶ Lista prihvaćenih vjerodajnica, gov.hr, <https://gov.hr/e-gradjani/lista-prihvacenih-vjerodajnica/1667>

OTP banka d.d.	OTP token/mToken	3	Trajna
Erste&Steiermärkische Bank d.d.	Erste mToken/Display kartica	3	Trajna
ADDIKO - Addiko Bank d.d.	Addiko token/mToken	3	Trajna
SRCE - Sveučilišni računski centar	Korisničko ime i lozinka - AAI@EduHr	2	Trajna
HP - Hrvatska pošta d.d.	ePošta	2	Trajna
FINA - Financijska agencija	e-Građani ePass	2	Trajna
HZMO - Hrvatski zavod za mirovinsko osiguranje	Korisničko ime i lozinka	2	Privremena
REGOS - Središnji registar osiguranika	Korisničko ime i lozinka	2	Privremena
HZZ - Hrvatski zavod za zapošljavanje	Korisničko ime i lozinka	2	Privremena

Popis prihvaćenih vjerodajnica se ažurira i objavljuje sukladno proceduri definiranoj u dokumentu "**Protokol rada NIAS-a**"⁶⁷, što obuhvaća sljedeće postupke:

- Provedba audita kod izdavatelje vjerodajnice – pri čemu se ocjenjuju različiti čimbenici, od kvalitete procesa identifikacije i registracije korisnika do vrste i načina izdavanja te korištenja vjerodajnice u procesu elektroničke autentikacije,
- Izrada stručne podloge „Mišljenje NIAS audit tima o razini osiguranja kvalitete autentikacije za predmetnu vjerodajnicu“
- Donošenje „Odluke o prihvaćanju i ocjeni razine za predmetnu vjerodajnicu u okviru sustava NIAS“
- Uključivanje vjerodajnice u sustav NIAS i njezina javna objava na „Listi prihvaćenih vjerodajnica“.

Zahtjevi koje izdavatelj vjerodajnice treba ispuniti definirani u dokumentu "Kriteriji za određivanje razine osiguranja kvalitete autentifikacije u okviru sustava NIAS"⁶⁸.

Primjer odabira vjerodajnice iz čiste prihvatljivih vjerodajnica u sustavu NIAS za određenu uslugu, prikazan je na sljedećoj slici zajedno s logotipom izdavatelja vjerodajnice, načinom prijave i razinom sigurnosti koju vjerodajnica pruža.

⁶⁷ Protokol rada NIAS-a, Verzija 1.0, od 06.09.2013.

⁶⁸ Kriteriji za određivanje razine osiguranja kvalitete autentifikacije, Verzija 1.2, od 06.12.2013.

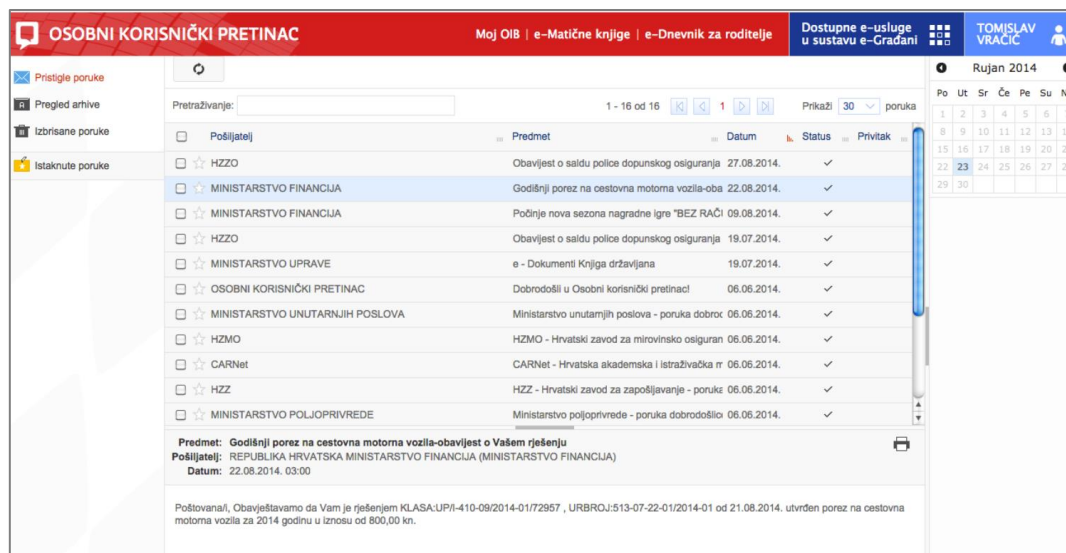
Kao podsustav NIAS-a postoji i usluga **mojID** koja služi za administraciju korisnikovog jedinstvenog elektroničkog identiteta (e-ID) u NIAS-u. usluga je namijenjena krajnjem korisniku za jedinstvenu evidenciju svih korisničkih računa otvorenih kod različitih izdavatelja vjerodajnica uključenih u NIAS.

Lista prihvatljivih vjerodajnica		
Izdavatelj vjerodajnice	Način prijave	Sigurnosna razina
	Osobni certifikat	4
	Token aplikacija	3
	Korisničko ime i lozinka	2
Izdavatelj vjerodajnice	Način prijave	Sigurnosna razina
	Korisničko ime i lozinka	2
	Osobni certifikat	3
	Token uređaj / aplikacija	3
	Korisničko ime i lozinka	2
	Osobni certifikat	3
	Token uređaj / aplikacija	3
	mToken aplikacija / čitač kartice / mobilni token #withKEY	3
	mToken / čitač kartice / token	3
	SMS jednokratni pin	3
	Osobni certifikat	4
	Token uređaj / aplikacija	3
	Korisničko ime i lozinka	2
	Poslovni certifikat	4
	mToken aplikacija / Display kartica	3
	Osobni certifikat	4
	Token uređaj / aplikacija	3
	Poslovni certifikat	4

Slika 1.7: Odabir vjerodajnice iz liste prihvatljivih vjerodajnica

1.7.3.4 Sustav Osobnog korisničkog pretinca (OKP)

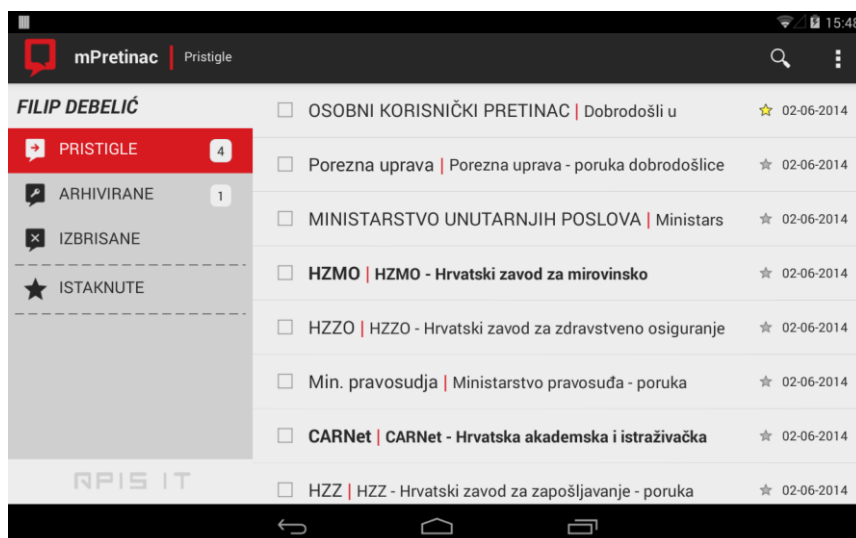
Sustav Osobnog korisničkog pretinca OKP predstavlja realizaciju pristupa osobnim informacijama koje javni sektor upućuje građanima, kao što su primjerice informacije o statusu osobnih predmeta ili informacije o elektroničkoj razmjeni osobnih podataka. Svaki građanin Republike Hrvatske koji ima važeći OIB, posjeduje i svoj vlastiti osobni korisnički pretinac. Sustav OKP omogućava primanje osobnih službenih poruka vezanih za javne usluge, uvid u postupke i njihov tijek te osobne statuse kao i njihov pregled, upravljanje i pohranu. Putem njega svaki građanin je izravno informiran o njemu važnim situacijama i događajima vezanim za osobna zakonska prava i obveze te o korištenju osobnih podataka u javnom sektoru.



Slika 1.8: Primjer poruka korisnika u sustavu OKP putem web sučelja



Pristup osobnom korisničkom pretincu moguće je i putem **nativnih aplikacija za mobilne uređaje** na platformama Android i iOS te Windows. Za uspostavu i operativno vođenje sustava OKP zadužen je APIS-IT.



Slika 1.9: Primjer poruka korisnika u sustavu OKP na mobilnom ili tablet uređaju

U sustavu OKP dostupne su elektroničke poruke iz sljedećih područja⁶⁹:

- Stanovanje i okoliš (1): Jedinstvena uplatnica
- Prava potrošača (1): e-Upitnik o zadovoljstvu elektroničkim uslugama
- Zdravlje (13): Naknade plaće i potpore, Istek dopunskog osiguranja, Saldo dopunskog osiguranja, Promjena izabranog liječnika, Status obveznog zdravstvenog osiguranja, Obavijest o izrađenoj EKZO kartici, Obavijest o isteku EKZO kartice, Podsjetnik o aktivnoj narudžbi, Dodjela MBO-a novorođenom djetetu, Obavijest o otkazanoj narudžbi, Prijava temeljem redovnog školovanja, Obavijest o promjeni osobnih podataka, Privremena nesposobnost za rad
- Financije i porezi (9): Obračunati porez na dohodak, Deblokada računa, Blokada računa, Obavijest o nagradnoj igri, Nova osnova za blokirani račun, Obavijest o stanju blokade, Izvršena osnova bez blokade računa, Dobrodošlica-računi su blokirani, Dobrodošlica-računi nisu blokirani
- Aktivno građanstvo (1): Obavijest o obveznom cijepljenju kućnih ljubimaca
- Poslovanje (2): Obavijest o nepotpunoj/neispravnoj Prijavi za upis, Obavijest o provedenom postupku upisa u Obrtni registar
- Promet i vozila (5): Istek registracije vozila, Rješenje za obračun posebnog poreza na motorna vozila, Obavijest o isteku tehničkog pregleda plovila, Naknada za pomorsko dobro i objekte sigurnosti, Neplaćena naknada za uporabu pomorskog dobra
- Odgoj i obrazovanje (1): Obavijest o izostanku djeteta
- Rad (18): Raspored u OMF, Potvrda članu OMF, Promjena kategorije OMF, Prijenos poslova upravljanja OMF-om, Pripajanje mirovinskih društava, Adresa zatraženog e-dokumenta, Ulazak u evidenciju nezaposlenih, Izlazak iz evidencije nezaposlenih, Rješenje o novčanoj naknadi, Prestanak novčane naknade, Prijava o početku osiguranja, Prijava o prestanku osiguranja, Brisanje prijave o početku osiguranja, Obračunate ustege iz mirovine, Obavijest o isteku dokumenta iz sustava e-Pomorac, Obavijest o prevođenju invalidske na starosnu mirovinu, Obavijest o postupku utvrđivanja dijela mirovine, Rješenje AORT-a
- Obitelj i život (1): Obavijest o provedenoj verifikaciji podataka
- Pravna država i sigurnost (14): Promjena u sudskom predmetu, Istek osobne iskaznice, Istek putovnice, Istek vozačke dozvole, Obavijest o biračkom mjestu, e-oglasna, Rješenje zahtjeva za privremeni upis birača, Zahtjev za prijavu boravišta - obavijest, Obavijest o roku za preuzimanje osobne iskaznice, Elektronički zapis - MUP RH, Kaznena evidencija - neispravan zahtjev, Uvjerjenje iz kaznene evidencije, Uvjerjenje da se ne vodi kazneni postupak, Nevođenje kaznenog postupka - neispravan zahtjev.

⁶⁹ Elektroničke poruke dostupne u sustavu OKP na dan 31.1.2020.



Institucije koje izdaju elektroničke poruke putem sustava OKP su (broj dostupnih e-poruka u zagradi): Agencija za osiguranje radničkih tražbina (1), CARNet (1), Financijska agencija - FINA (7), Hrvatski zavod za mirovinsko osiguranje (6), Hrvatski zavod za zapošljavanje (4), Hrvatski zavod za zdravstveno osiguranje (13), MINPO (2), Ministarstvo financija (3), Ministarstvo mora, prometa i infrastrukture (4), Ministarstvo poljoprivrede (1), Ministarstvo pravosuđa (6). Ministarstvo unutarnjih poslova (7), Ministarstvo uprave (4), REGOS - Središnji registar osiguranika (6) i ZG Holding (1).

1.8 Program Interoperabilna rješenja za europsku javnu upravu ISA

Program Europske Komisije **ISA² - Interoperabilna rješenja za javnu administraciju, poslovanje i građane** (*Interoperability solutions for public administrations, businesses and citizens*)⁷⁰ razvijen je kao podrška razvoju digitalnih rješenja koja omogućava da javne uprave, poduzeća i građani u EU imaju izravne koristi od prekograničnih i međusektorskih interoperabilnih javnih usluga. Prethodni program **Interoperabilna rješenja za europsku javnu upravu ISA⁷¹** (*Interoperability Solutions for European Public Administrations*) imao je za cilj olakšati i ubrzati razmjenu informacija i pružanje usluga elektroničke uprave, kako među upravnim sektorima, tako i među državama Europske unije.



Program ISA pokrenut je krajem 2009. godine, a trajao je do kraja 2015. godine s ukupnom vrijednosti od 164 milijuna eura, dok je program ISA² započeo 2016. godine i traje do kraja 2020. godine, s proračunom u iznosu od 131 milijuna eura. Programom ISA² upravlja Jedinica za interoperabilnost Generalne direkcije za informatiku Europske komisije – DIGIT.D2.

S aspekta interoperabilnosti ovo su vrlo zanimljivi programi, te tako program ISA² podržava čak 54 akcije, organizirane u 9 radnih paketa, usredotočene na razvoj digitalnih rješenja i rješenja koja se mogu besplatno koristiti u području interoperabilnosti.

Program ISA² osigurava da su aktivnosti interoperabilnosti dobro koordinirane na razini EU, potiče razvoj rješenja za javnu upravu na temelju potreba poduzeća i građana, uspostavlja potrebne instrumente za jačanje interoperabilnosti na razini EU i nacionalnoj razini, u što, među najvažnijim, ubrajamo:

- revidirani **Europski okvir interoperabilnosti EIF** (*European Interoperability Framework*)

⁷⁰ ISA² - Interoperability solutions for public administrations, businesses and citizens, <https://ec.europa.eu/isa2/>

⁷¹ Interoperability Solutions for European Public Administrations, <http://ec.europa.eu/isa/>

- revidiranu **Europsku strategiju interoperabilnosti EIS** (*European Interoperability Strategy*)
- arhitekturu u obliku **Europske referentne arhitekture interoperabilnosti EIRA** (*European Interoperability Reference Architecture*)
- kartografiju rješenja u obliku **Europske interoperabilne kartografije EIC** (*European Interoperability Cartography*).

Program se koordinira s drugim relevantnim službama Europske Komisije, kao što primjerice DG DIGIT, DG CNECT i DG GROW te drugim povezanim inicijativama, kao što je:

- kontekst strategije jedinstvenog digitalnog tržišta (*Digital Single Market Strategy*)
- Europski akcijski plan e-uprave (*European eGovernment Action Plan*) 2016.-2020.
- CEF-Digital (*Connecting Europe Facility*)

te sličnim inicijativama vezanim uz interoperabilnost.

Sudjelovanjem u programu ISA i ISA² tijelima državne uprave i organizacijama u Republici Hrvatskoj na raspolaganju stoji niz servisa koje mogu koristiti u cilju povećanja učinkovitosti razmjenom podataka putem pouzdane mreže koja jamči visoku razinu sigurnosti i integritet svih podataka.

1.9 Okviri za interoperabilnost

Kako bi se postigla interoperabilnost i na nacionalnoj i na međunarodnoj razini treba utvrditi pravila i norme za razmjenu i korištenje podataka. Stoga se osmišljavaju **okviri za interoperabilnost** (engl. *interoperability frameworks*), koji se ukratko mogu definirati kao niz pravila koja opisuju dogovorene načine međusobne interakcije organizacija i niz normi koje se trebaju koristiti kod ostvarivanja interoperabilnosti.

1.9.1 Europski okvir za interoperabilnost (EIF)

Europska unija prepoznala je potrebu za donošenjem određenih propisa vezanih uz interoperabilnost još u prošlom stoljeću. Tako je još 1998. godine donese Direktiva o normama 1998/34/EC, a 2003. godine Direktiva o informacijama javnog sektora 2003/98/EC, te 2004. godine Direktiva o javnoj nabavi 2004/18/EC. Usporedo s time događao se i niz političkih inicijativa interesnih strana, odnosno proizvođača određenih platformi i programske podrške, koje su cijelo to vrijeme pokušale utjecati na izgled tih vršnih dokumenata o interoperabilnosti za EU.

Jedna od pozitivnih neovisnih inicijativa bila je ona za usvajanje okvira za interoperabilnost, koja je i ostvarila svoj cilj kada je 2004. godine donesene prva verzija **Europskog okvira za interoperabilnost EIF** (*European Interoperability Framework*), odnosno **EIF 1.0**⁷². Europski okvir za interoperabilnost zapravo opisuje kako su se organizacije sporazumjele ili će se

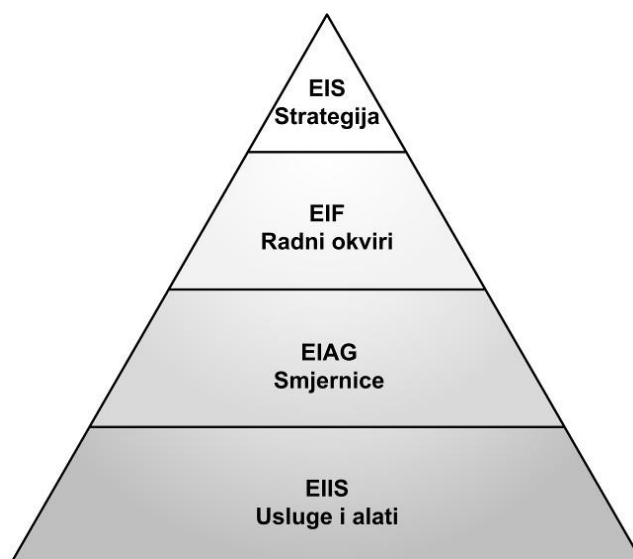
⁷² European Interoperability Framework for Pan-European eGovernment Services, EIF 1.0, IDABC, EC, <http://ec.europa.eu/idabc/en/document/2319/5938.html>

sporazumjeti oko međusobne interakcije i koje će norme koristiti. EIF donosi politike i smjernice koje čine temelj za odabir normi, a prilagođava se pojedinoj gospodarskoj, sociološkoj, političkoj, kulturnoj, jezičnoj, povijesnoj i zemljopisnoj situaciji i primjeni u određenom slučaju. EIF se primarno definira nad paneuropskim uslugama javne uprave PEGS (*Pan-European eGovernment Services*).

Prva verzija EIF-a donesena je u sklopu programa **Interoperabilne isporuke europskih e-usluga javne uprave administraciji, poslovnom sektoru i građanima IDABC** (*Interoperable Delivery of European eGovernment Services to public Administrations, Business and Citizens*) koji je završio 2009. godine. Inicijative programa IDABC, osim EIF-a, uključuju i:

- **Europsku strategiju interoperabilnosti EIS** (*European Interoperability Strategy*),
- **Smjernice za izgradnju arhitekture europske interoperabilnosti EIAG** (*European Interoperability Architecture Guidelines*) i
- **Infrastrukturne usluge europske interoperabilnosti EIIS** (*European Interoperability Infrastructure Services*).

Infrastrukturne usluge i alati EIIS u obliku uobičajenih i generičkih ponovno iskoristivih komponenata, čine temelj **trokuta upravljanja interoperabilnošću** (engl. *interoperability government triangle*). Povrh njega su smjernice za izgradnju arhitekture EIAG, iznad njih je radni okvir EIF, a na vrhu je strategija EIS, kao što se može i vidjeti na sljedećoj slici, a EIF i EIS su i dva ključna elementa Digitalne agende.



Slika 1.10: Razine trokuta upravljanja interoperabilnošću (strategija, okvir, smjernica, usluge i alati)

Europska strategija interoperabilnosti EIS⁷³, izdana kao aneks 1 komunikacije “*Towards interoperability for European public services*” Europske komisije, sadržava općenite smjernice i akcijske prioritete koje trebaju pomoći interakcijama, razmjeni i suradnji europskih javnih uprava za isporuku europskih javnih usluga. Iako je strategija započeta pod programom

⁷³ Annex I to the Commission communication on interoperability - European Interoperability Strategy (EIS), http://ec.europa.eu/isa/documents/isa_annex_i_eis_en.pdf

IDABC, nastavljena je u programu ISA. U strategiji su se izričito naglašavala tri različita klastera područja:

- povjerenje razmjene informacija,
- arhitektura za interoperabilnost i
- procjena implikacija ICT-a na novu legislativu EU,

te dvije pridružene mjere:

- podizanje svijesti o interoperabilnosti i
- dijeljenje najboljih praksi.

Provedbu okvira EIF u prvoj verziji najviše su onemogućavale netehnološke prepreke različitosti članica EU u veličini i strukturi javne uprave, te modela interakcije s građanima. Kao neke od primjera naveli bi razlike u broju i mogućnostima ponuđenih javnih usluga, statusu vlasništva poduzeća, nadzornim operacijama granice od strane carine i policije, centralizaciji i lokalizaciji upravljanja, načinima interakcije, procesima ključnim za građane (rođenje, sklapanje braka, smrt), procedurama izdavanja dokumenata te u podršci za višestruke strane jezike.

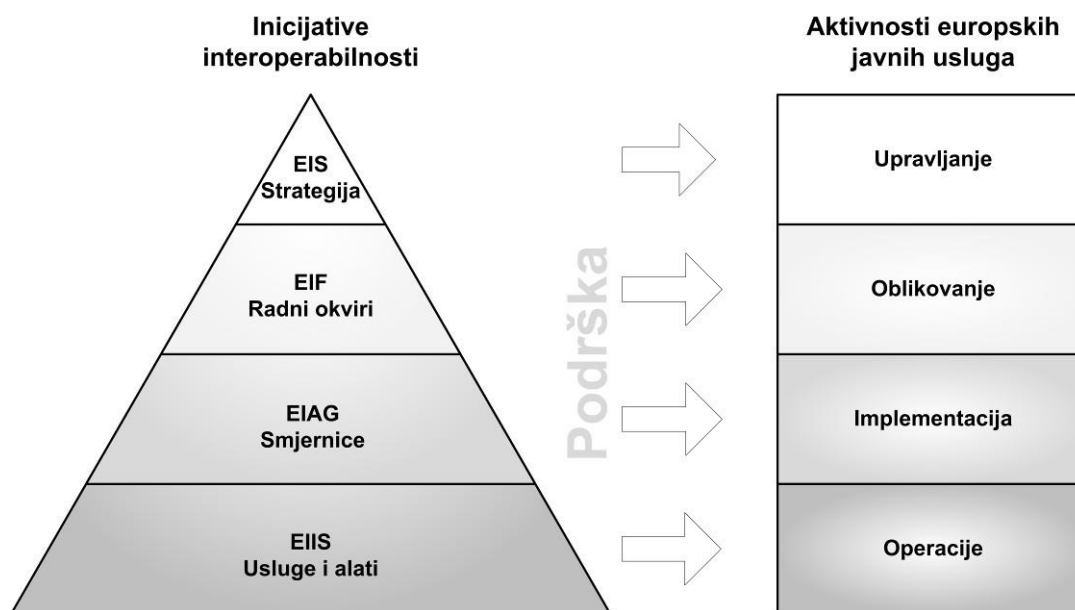
Iz tog razloga pristupilo se osvježavanju inačice, te je stvoren prijedlog **Europskog okvira za interoperabilnost EIF 2**⁷⁴, koji je bio tema niza političkih debata, naročito vezanih uz razloge korištenja određenih tehnologija. Tijekom diskusija o prijedlogu kreirali su se određeni lobiji koji su zagovarali korištenje određenih vlasničkih normi i tehnologija koje su pod komercijalnim licencama, što je bilo u suprotnosti s idejom i smjernicama Europske unije o korištenju otvorenih normi, a po mogućnosti i otvorenog koda te otvorene programske podrške. EIF 2 je ipak usvojen u prosincu 2010. od strane Europske komisije kao aneks 2 iste već spomenute komunikacije Europske komisije. Za početak pročitajmo službenu definiciju okvira za interoperabilnost:

*„Okvir za interoperabilnost je dogovoreni pristup interoperabilnosti za organizacije koje žele **surađivati** u smjeru **zajedničke isporuke javnih usluga**. U okviru svog djelokruga primjene, **specificira skup zajedničkih elemenata**, kao što su vokabular, koncepti, principi, politike, smjernice, preporuke, norme, specifikacije i prakse.“*

Interoperabilnost je pretpostavka i podupirač učinkovite isporuke europskih javnih usluga, tako što rješava suradnju, razmjenu informacija i dijeljenje te ponovno korištenje informacija. Rezultat su poboljšana isporuka javnih usluga i smanjenje troškova.

Dakle, vidljivo je da EIF zapravo ne daje preporuke konkretnih tehničkih norma, već donosi opće smjernice, odnosno preporuke. Neke od preporuka EIF-a odnose se na područja dostupnosti, višezličnosti, privatnosti, korištenje otvorenih norma, programske podrške otvorenog koda i višestranih rješenja.

⁷⁴ Annex II to the Commission communication on interoperability - European Interoperability Framework (EIF), http://ec.europa.eu/isa/documents/isa_annex_ii_eif_en.pdf

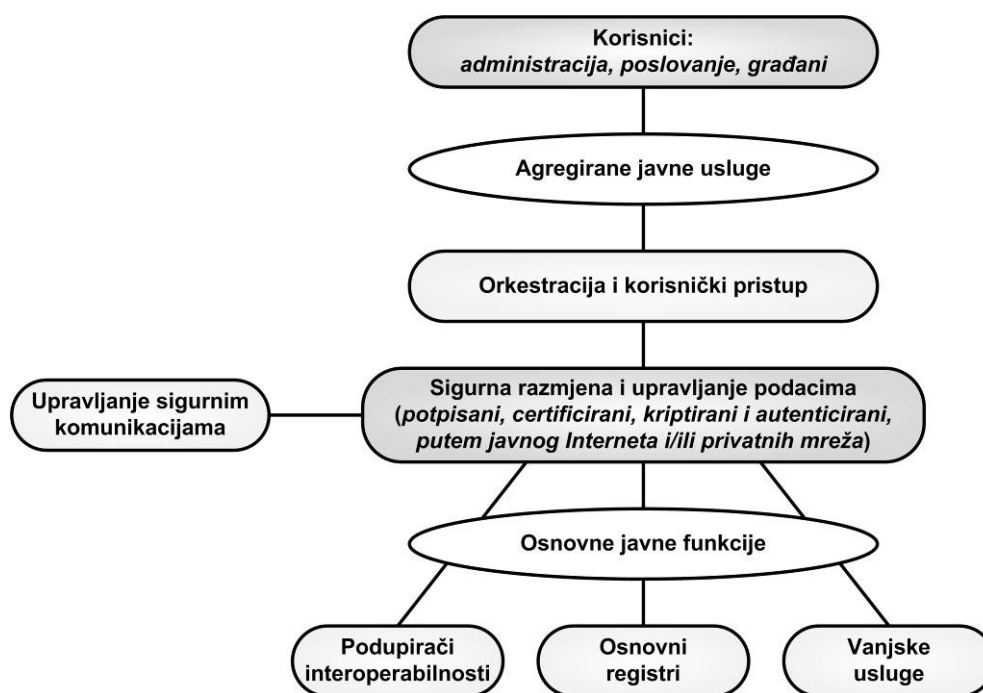


Slika 1.11: Inicijative interoperabilnosti podupiru aktivnosti osnivanja europskih javnih usluga

EIF definira i **12 principa**, koje ćemo ovdje i navesti:

1. supsidijarnost (kao najniže moguće) i proporcionalnost (kao najbliže moguće),
2. usmjerenost korisniku,
3. uključenost i dostupnost,
4. sigurnost i privatnost,
5. višejezičnost,
6. jednostavnost administracije,
7. transparentnost,
8. sačuvanost informacije,
9. otvorenost,
10. ponovna iskoristivost,
11. tehnološka neutralnost i prilagodljivost i
12. djelotvornost i učinkovitost.

EIF definira i **konceptualni model javnih usluga** (engl. *conceptual model for public services*) koji sadrži glavna načela organiziranja konstrukcije i operacije interoperabilnosti (Slika 1.12).



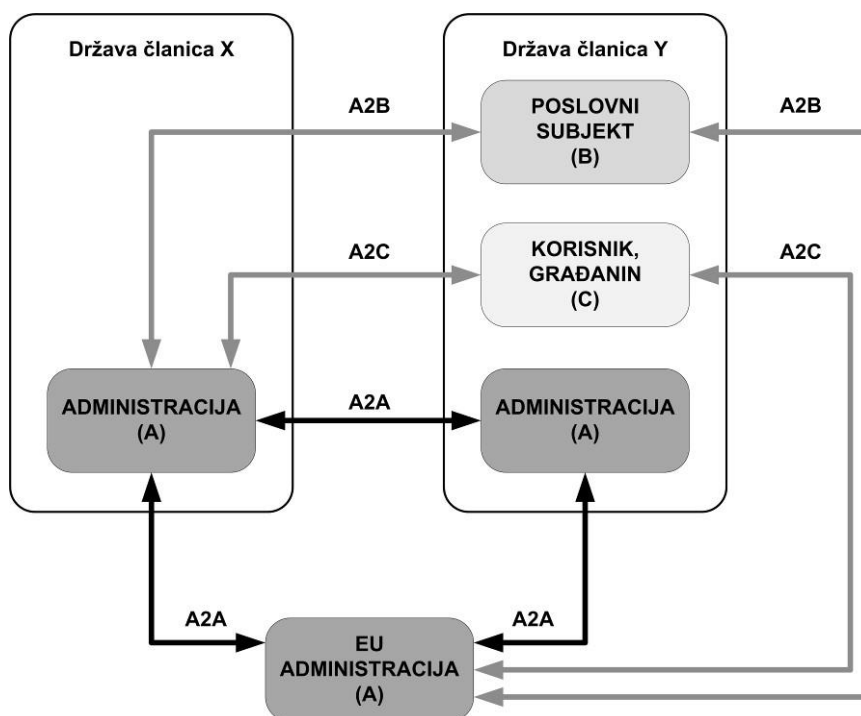
Slika 1.12: Konceptualni model javnih usluga EIF-a

Osnovne javne funkcije uključuju registre i javne baze podataka, te podupirače interoperabilnosti, kao što su razni informacijski brokeri, prevoditelji protokola, formata, jezika, značenja i normi, te druge vanjske usluge, kao što su usluge pristupa Internetu ili usluge plaćanja.

Središnja razina sigurne razmjene podataka je infrastruktura koja omogućava pouzdan i siguran transport i razmjenu podataka zaštićenih raznim mehanizmima, uključujući elektronički potpis, certifikat, kriptiranja te autentikaciju i autorizirani pristup, kao i nadzor te zapisivanje aktivnosti. U ovu razinu pripada i upravljanje uslugama sigurne komunikacije, registracija usluga i zapisivanje događaja.

Razina agregacije usluga grupira se korištenjem određenih osnovnih javnih funkcija i predstavljanjem određenoj vrsti korisnika, a to su javna uprava, gospodarstveni sektor i krajnji korisnici, odnosno građani.

U EIF-u se interoperabilnost definira na tehničkoj, semantičkoj, organizacijskoj, pravno i političkoj razini. Također, EIF je sukladan scenarijima interakcija, samo ovdje ne govorimo o nacionalnoj, odnosno državnoj, već europskoj razini, pa tako možemo pogledati jedan tipičan primjer scenarija interakcije kakvi se žele ostvariti (Slika 1.13).



Slika 1.13: Scenarij interakcije europskih javnih usluga

Na kraju ne zaboravimo da EIF propagira **korištenje otvorenih normi, tehničkih specifikacija i uporabe softvera otvorenog koda**. Korištene norme trebale bi održavati neprofitne organizacije procesom otvorenog donošenja odluka i konsenzusom od strane svih zainteresiranih stranaka. Osim toga norme se trebaju javno objavljivati te biti dostupne besplatno ili uz minimalni trošak, a intelektualna vlasništva u obliku патената koja su dio norme trebaju također biti dostupna besplatno (engl. *royalty-free*). Smatra da je izravna povezanost otvorenosti i interoperabilnosti vrlo važna te da otvorene norme imaju ključnu ulogu u postizanju visoke razine interoperabilnosti. Tako se preporučuje prihvaćanje tržišnih i pravnih normi koje predstavljaju IETF, W3C, OASIS, CEN i ISO, a koje pokrivaju područja komunikacije, sigurnosti, razmjene podataka, otkrivanja usluga, prezentacije i zapisa podataka, metapodataka o procesima i podacima te imenovanja. Osim toga preporučuje se i korištenje softvera otvorenog koda i razvojnih modela otvorenog koda.

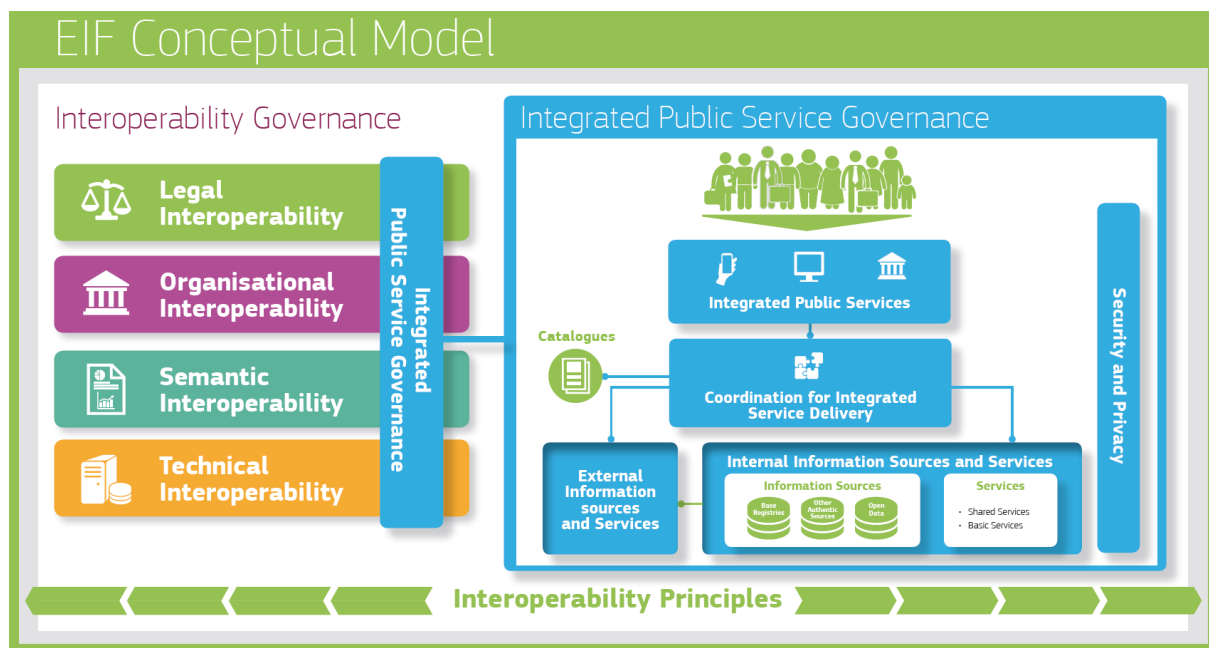
Europska komisija je 2017. godine donijela **novi Europski okvir interoperabilnosti (New EIF)** koji više nije sadržavao broj inačice, a kao dio je komunikacije (COM (2017) 134): Novi EIF nudi **47 konkretnih preporuka** o tome kako poboljšati upravljanje svojim aktivnostima interoperabilnosti, uspostaviti međuorganizacijske odnose, pojednostaviti procese koji podržavaju krajnje digitalne usluge i osigurati da postojeće i novo zakonodavstvo ne dovode u pitanje napore interoperabilnosti. Novi EIF stvoren je u kontekstu prioriteta stvaranje jedinstvenog digitalnog tržišta u Europi. Javni sektor, koji čini preko četvrtine ukupne zaposlenosti i predstavlja otprilike petinu BDP-a EU kroz javne nabave, igra ključnu ulogu na jedinstvenom digitalnom tržištu kao regulator, pružatelj usluga i poslodavac. Uspješna provedba EIF-a poboljšat će kvalitetu europskih javnih usluga i stvorit će okruženje u kojem javne uprave mogu surađivati na digitalni način.

Novi okvir stavlja veći naglasak na to kako se načela i modeli interoperabilnosti trebaju primjenjivati u praksi. Broj preporuka porastao je s 25 na 47, a postojeće preporuke za interoperabilnost su ažurirane pojednostavljene kako bi se olakšala njihova provedba, s jačim

fokusom na otvorenost i upravljanje informacijama, prenosivost podataka, upravljanje interoperabilnošću i integrirano pružanje usluga.

Rezultat je to uzimajući u obzir nove politike EU-a, kao što je revidirana Direktiva o ponovnoj upotrebi informacija u javnom sektoru, Direktiva INSPIRE i Uredba eIDAS. Također se razmatraju nove inicijative EU-a, kao što je Europska inicijativa za oblak, Akcijski plan e-uprave EU-a za razdoblje 2016. - 2020., kao i one koje su predviđene, poput Jedinstvene digitalne propusnice.

Komunikacija vezana uz novi EIF je prevedena na sve jezike pa tako i hrvatski⁷⁵.

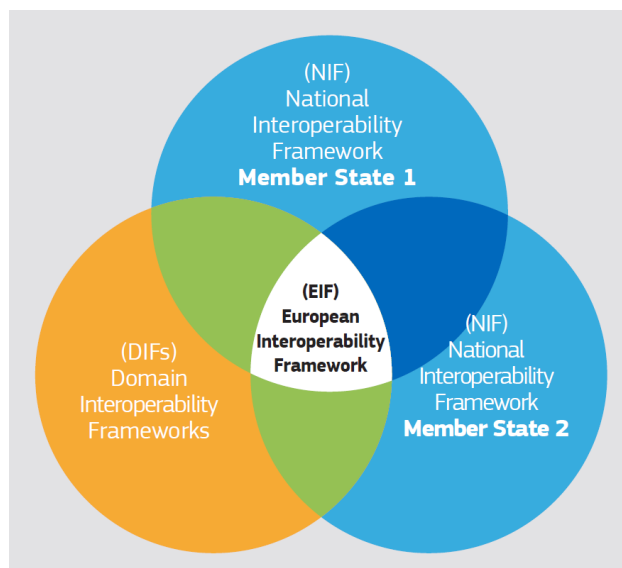


Slika 1.14 Konceptualni model novog EIF-a⁷⁶

Novi EIF popraćen je **Akcijskim planom interoperabilnosti**, u kojem su navedeni prioriteti koji bi trebali podržati provedbu EIF-a od 2016. do 2020. godine. Akcijski plan interoperabilnosti sastoji se od pet fokusnih područja, koji se bave pitanjima koja se odnose na identifikaciju mehanizama za upravljanje interoperabilnošću, suradnju među organizacijama, angažiranje dionika i podizanje svijesti o prednostima interoperabilnosti. Također pokriva razvoj, poboljšanje i promociju ključnih čimbenika interoperabilnosti, uzimajući u obzir potrebe i prioritete krajnjih korisnika.

⁷⁵ Komunikacija Komisije Europskom parlamentu, Vijeću, Europskom gospodarsko i socijalnim odboru i odboru regija Europski okvir za interoperabilnost – strategija provedbe, COM/2017/0134 final, <https://eur-lex.europa.eu/legal-content/HR/TXT/?uri=CELEX:52017DC0134>

⁷⁶ Slika preuzeta s The New European Interoperability Framework, https://ec.europa.eu/isa2/eif_en



Slika.1.15 Presjek primjena nacionalnih i domenskih okvira za interoperabilnost⁷⁷

1.9.2 Hrvatski okvir za interoperabilnost (HROI)

U skladu se Europskim okvirom za interoperabilnost EIF očekuje se da zemlje članice razviju svoje nacionalne okvire za interoperabilnost. Stoga je i u Hrvatskoj pokrenute takva inicijativa, a Vlada Republike Hrvatske je prvo donijela **Strategiju reforme državne uprave i Strategiju razvoja elektroničke uprave**, a zatim 2010. godine donijela **Odluku o uspostavljanju Hrvatskog okvira za interoperabilnost** te usvojila Odrednice Hrvatskog okvira za interoperabilnost.

Hrvatski okvir za interoperabilnost (HROI) definira uvjete te način izgradnje i postizanja interoperabilnosti javne uprave na nacionalnoj razini u skladu s društvenim i korisničkim potrebama i prioritetima i u odnosu na zahtjeve interoperabilnosti EU. HROI je zapravo je skup usklađenih pravila, propisa, definicija, normizacijskih i drugih dokumenata, potpornih usluga i resursa te provedbenih mjera i organizacije namijenjene stvaranju preduvjeta za pružanje objedinjenih korisnički usmjerenih usluga javne uprave.

Konačni **cilj** primjene HROI je uspostava stabilne poslovne i tehnološke interoperabilnosti sustava javne uprave i drugih društvenih čimbenika sukladno načelima korisnički usmjerene uprave. **Svrha** HROI-a je stvaranje preduvjeta za učinkovitu i djelotvornu primjenu informacijske i komunikacijske tehnologije u podršci svim funkcijama javne uprave. HROI treba omogućiti procesnu povezanost, suradnju i razmjenu informacija između tijela javne uprave i svih ostalih sudionika u obavljanju funkcija javne uprave na nacionalnoj i EU razini. Krajnji je cilj izgradnja poslovne i tehnološke okoline koja će stabilno i dugoročno osigurati uvjete za učinkovitu suradnju sustava javne uprave i drugih sudionika na postizanju zajednički utvrđenih ciljeva, sukladno načelima korisnički usmjerene uprave.

Odrednice HROI-a⁷⁸ definiraju strateška polazišta za HROI, kao što su temeljna načela, odnos prema reformi javne uprave i e-upravi, dionike i korisnike, provedbeni cilj, aktivnosti, strukturu,

⁷⁷ Slika preuzeta s The New European Interoperability Framework, https://ec.europa.eu/isa2/eif_en

⁷⁸ Odrednice Hrvatskog okvira za interoperabilnost, ver.1.0, Središnji državni ured za e-Hrvatsku, 24.06.2010.

odnose i obveze u primjeni, prema kojima će se provoditi usklađivanje informacijskih sustava tijela državne uprave. Riječ je o važnim dokumentima koje je u sklopu provedbe Strategije razvoja elektroničke uprave, u suradnji s predstavnicima akademske zajednice, svih tijela državne uprave i domaće IT zajednice, izradio bivši Središnji državni ured za e-Hrvatsku. Dokumenti su izrađeni na temelju analize najbolje prakse zemalja članica EU-a i drugih zemalja s visoko razvijenom elektroničkom upravom te su usklađeni s Europskim okvirom za interoperabilnost, čime se na kraju omogućuje sigurna i učinkovita razmjena podataka ne samo između tijela državne uprave u Republici Hrvatskoj nego i s europskom administracijom.

Prema Odrednicama HROI-a definiraju se uvjeti te način izgradnje i postizanja interoperabilnosti javne uprave na nacionalnoj razini u skladu s društvenim i korisničkim potrebama i prioritetima i u odnosu na zahtjeve interoperabilnosti EU-a. Cilj HROI-a je stvoriti uvjete za komunikaciju između postojećih sustava i bržu razmjenu informacija između građana i službenika, omogućiti pružanje usluga elektroničkim putem te omogućiti automatiziranu razmjenu i korištenje velikih količina podataka pohranjenih u državnim registrima i za komercijalne svrhe. Uspostavljanjem HROI-a povezat će se danas odvojeni informacijski sustavi tijela državne uprave u umreženu upravu, čime će se unaprijediti proces upravljanja informacijama te razmjena podataka između tijela državne uprave, a građanima omogućiti jednostavnije i brže obavljanje javnih usluga elektroničkim putem.

Konačni cilj primjene HROI-a je uspostava stabilne poslovne i tehnološke interoperabilnosti sustava javne uprave i drugih sudionika. Kako bi se to postiglo, treba ostvariti sljedeće operativne ciljeve:

- uspostavu i djelovanje organizacije i sustava upravljanja te koordinacije provedbe na nacionalnoj razini,
- uspostavu nacionalne infrastrukture i podrške za poslovno, procesno i tehničko povezivanje s tijelima EU i njenim članicama te uključivanje u razvoj i isporuku prekograničnih, interoperabilnih i integriranih, korisnički usmjerenih javnih usluga,
- pripremu, usvajanje, primjenu i održavanje pravila i propisa, biblioteke normizacijskih i drugih dokumenata interoperabilnosti usklađenih s važećom europskom dokumentacijom te sustavno praćenje i primjena najbolje međunarodne i domaće prakse,
- priprema i provedba programa, planova i mjera usklađenih s društvenim i korisničkim potrebama i prioritetima
- uspostavu, izgradnju, održavanje i razvoj zajedničkih sustava, servisa i drugih resursa,
- objedinjavanje procesa i usluga javne uprave u korisnički usmjerene usluge,
- podizanje razine zajedničkog i višestrukog korištenja razvijenih javnih resursa i stečenih znanja na nacionalnoj razini,
- pojednostavljenje i podizanje učinkovitosti pripreme, prilagodbe, provedbe i praćenja rezultata strateških dokumenata i akata radi djelotvornog ostvarenja interoperabilnosti i društvenih ciljeva,
- izgradnju mehanizama trajnog obrazovanja i podrške državnim službenicima i namještenicima,
- uspostava sustava trajnog informiranja, podrške i komunikacije s korisnicima i javnosti kroz suvremene kanale komunikacije radi podizanja svijesti, poticanja korištenja, promocije te sudjelovanja u razvoju modernog sustava javne uprave.

1.9.3 Struktura, sudionici i temeljna načela HROI-a

Struktura Hrvatskog okvira za interoperabilnost uključuje:

- dokumente Vlade Republike Hrvatske,
 - Odluka o HROI-u, Odrednice HROI-a, godišnji planovi i izvještaji provedbe HROI-a,
- sustav upravljanja, organizacije i podrške provedbe,
- biblioteku strateških i provedbenih dokumenata interoperabilnosti,
 - programi, planovi, norme, metodološki priručnici, upute, smjernice, katalozi, šifranici itd.,
- zajedničke resurse i potporne usluge podrške provedbi
 - informacijska infrastruktura, provedbena organizacija, funkcije, procesi, servisi i mehanizmi podrške,
- provedbene mjere.

Sudionici HROI-a su sva tijela, organizacije i pojedinci koji sudjeluju u upravljanju i provedbi HROI-a, obvezno uključujući Vladu Republike Hrvatske kao vlasnika, Središnji državni ured za e-Hrvatsku, odnosno Upravu za e-Hrvatsku pri Ministarstvu uprave, različita druga ministarstva i središnje državne urede kao nositelje provedbe, državne upravne organizacije, urede državne uprave u županijama, javni sektor (zavodi, agencije itd.), Agenciju za podršku informacijskim sustavima i informacijskim tehnologijama APIS-IT, Financijsku agenciju FINA i mnoge druge, a poželjno uključujući i tijela lokalne i područne samouprave, trgovačka društva, pravna tijela s javnim ovlastima te znanstvene, obrazovne i razvojne institucije.

Temeljna načela pri provedbi HROI-a su:

- primjena načela Strategije razvoja elektroničke uprave u RH, odnosno novih strategija kako budu donošane,
- praćenje europske strategije i Europskog okvira za interoperabilnost uz uvažavanje najbolje prakse zemalja s razvijenim nacionalnim okvirima za interoperabilnost,
- referenciranje na druge postojeće odgovarajuće nacionalne i/ili međunarodne norme (bez pokretanja paralelnog postupka usvajanja normi),
- građenje kroz kombinirani pristup:
 - **odozgo prema dolje** (engl. *top-down*) – temeljeno na strateškim ciljevima i prioritetima strategije vladinih programa kroz standardizaciju metodologija, uspostavu servisa interoperabilnosti, izgradnju i dostupnost gradbenih komponenti, provedbu strateških projekata i pilota te koordiniranim radom tijela državne uprave u području interoperabilnosti,
 - **odozdo prema gore** (engl. *bottom-up*) – od strane sudionika HROI-a, temeljeno na inicijativama skupina interesno i poslovno povezanih sudionika, tamo gdje postoje uvjeti za uspješnu uspostavu interoperabilnosti te brzu realizaciju i isporuku usluga korisnicima,
- poticanje višekratnog korištenja dostupnih referentnih specifikacija i gotovih komponenti temeljenih na otvorenom kodu uz zadovoljavanje konceptualnog modela javnih usluga, važećih metoda i međunarodnih normi,
- transparentan i otvoren razvoj HROI-a koji omogućuje sudjelovanje svih sudionika u skladu s Kodeksom savjetovanja sa zainteresiranom javnošću u postupcima donošenja zakona, drugih propisa i akata.

1.9.4 Primjena i provedba HROI-a

Kod primjene HROI-a u tijelima javne uprave istovremeno treba djelovati na razvojno-strateškoj i operativnoj razini jer se jedino koordiniranom provedbom na obje razine mogu postići postavljeni ciljevi. Na razvojno-strateškoj razini postavljaju se nacionalni strateški i programski ciljevi Vlade, čija je posljedica postizanje općih društvenih ciljeva na razini cijelog sustava. Na operativnoj se razini postavljaju neposredni ciljevi pojedinih skupina i organizacija, čija je posljedica konkretna interoperabilna usluga za određene korisnike vezane uz te organizacije.

S obzirom na to da primjena HROI-a u javnoj upravi čini samo jedan dio interoperabilnosti na nacionalnoj razini, drugi dio ciljeva treba usmjeriti ka gospodarstvu i civilnom sektoru. Tek kada se interoperabilnost uspostavi između javne uprave, gospodarskog i civilnog sektora moći će se govoriti o punoj interoperabilnosti. Takva interoperabilnost otvara nove načine racionalizacije poslovanja te racionalnog korištenja javnih resursa i proračunskih sredstava, uz prepuštanje dijela razvoja, funkcija i usluga onim organizacijama koje to rade učinkovitije, kvalitetnije i jeftinije.

Na razini gospodarstva identificiraju se opća područja primjene u informacijsko-komunikacijskom tehnološkom sektoru te razvojnim i uslužnim organizacijama. Izgradnja zajedničke informacijske infrastrukture i tehnička realizacija rješenja uz savjetovanje i osposobljavanje drugih sudionika glavna je uloga organizacija informacijsko-komunikacijskog tehnološkog sektora. Izgradnja sadržaja i stručnih elemenata usluga za krajnje korisnike te korištenje javnih informacijskih resursa područja su u kojima će važnu ulogu imati razvojne i uslužne organizacije. Potrebe civilnog sektora donekle su usporedive s javnim sektorom, a najčešće se kreću u područjima kao što su zdravstvo, socijalna skrb, kultura i slično.

Jedan od glavnih proizvoda provedbe je **biblioteka strateških i provedbenih dokumenata interoperabilnosti**. Radi korištenja biblioteke očekuje se razvoj Kataloga važećih dokumenata interoperabilnosti i usluga podrške HROI-a, koji će sadržavati opis dokumenta, namjenu, izvor ili nadležno tijelo, status, referentni izvor, rok primjene i važnosti te poveznice na srodne dokumente. Biblioteka i katalog dokumenata interoperabilnosti sadržavat će:

- dokumente koje HROI referencira (strategije, strateški i ostali planovi koji definiraju pokazatelje uspješnosti državne uprave, propisi koji uređuju organizacijske strukture, procese i podatke u državnoj upravi te dokumenti i direktive EU-a koje utječu na interoperabilnost),
- strateške i planske dokumente (studije, analize, preporuke, programi i planovi provedbe),
- normizacijske dokumente (referentne arhitekture, referentni modeli i specifikacije, standardni projekti, tehničke dokumentacije, katalozi, šifarnici, adresari, rječnici i drugo),
- pomoćne dokumente (metodološki i drugi priručnici, upute, smjernice, obrasci ugovora i drugih dokumenata, radni dokumenti, obrazovni materijal i drugo),
- arhivu prethodnih oblika dokumenata.

U vrijeme donaćanja HROI-a odlučeno je da će Razvoj i primjena HROI-a biti podržani informacijsko-komunikacijskim tehnologijama kroz razvoj **zajedničkih resursa i potpornih usluga** podrške koji osiguravaju provedbenu podršku. Zajednički resursi (gradbene komponente) koji su tada navedeni uključivali su:

- portal Moja uprava – javno objavljivanje dokumenata, pristup objedinjenim i korisnički prilagođenim uslugama,
- sustav za autentikaciju i autorizaciju – pristup i upravljanje uslugama i resursima sukladno pravima i ovlaštenjima,

- sustav izgradnje, povezivanja, upravljanja i objedinjavanja procesa i usluga javne uprave, tzv. upravna sabirnica usluga (engl. *Government Service Bus*), uporabom informacijskog sustava OIB-a,
- riznicu semantičke imovine,
- uslugu pregleda, sistematizacije pretraživanja i dohvata dokumentacije i resursa interoperabilnosti te registar usluga – omogućuje opis, nalaženje i integraciju usluga (više o UDDI-u, engl. *Universal Description, Discovery and Integration* u poglavlju 6),
- izvorne registre temeljnih entiteta (fizičke osobe, poslovni subjekti i prostorne jedinice),
- sustav HITRONet – računalno-komunikacijsko povezivanje sustava sudionika,
- program e-Ured – sustav normi, modela i standardnih projekata za izgradnju informacijskih sustava sudionika,
- kolaboracijsku platformu za suradnju na razvoju dokumenata interoperabilnosti,
- stručnu, operativnu, organizacijsku, projektnu i tehničku podršku razvoju i provedbi te primjeni referentnih arhitektura, modela i specifikacija.

U Strateškom planu Ministarstva uprave za razdoblje od 2014. do 2016. godine eksplicitno je naveden cilj unaprjeđenja kvalitete javnih usluga kroz djelotvornu primjenu ICT-a u radu javne uprave, od čega su naročito važni razvoj i standardizacija informacijske i komunikacijske infrastrukture, poboljšanje dostupnosti i kvalitete elektroničkih usluga i rješenja, te unaprjeđenje sustava podrške za koordinaciju razvoja cjelovitih integriranih usluga.

Iako su praktično svi planirani načini ostvarenja zacrtanih ciljeva Strateškog plana izravno ili neizravno povezani s interoperabilnošću, a time i s HROI, izdvojili bi ipak nekoliko najvažnijih s aspekta HROI:

- usklađivanje Hrvatskog okvira za interoperabilnost s EIF 2.0,
- strategija „oblaka javnog sektora“,
- donošenje strategije za otvorene specifikacije i standarde,
- korištenje zajedničkih interoperabilnih rješenja na razini EU,
- primjena elektroničkih identiteta u elektroničkim javnim uslugama,
- elektronička razmjene podataka između registara u javnoj upravi,
- popis usluga javnog sektora, odabir i priprema u obliku integralnih e-usluga,
- pregled procesa državne uprave i gotovih *e-government* rješenja,
- zajedničke metodologija za standardizaciju ICT-a,
- sustav podrške za ponovno korištenje javnih informacija,
- povezivanje sustava za podršku korisnicima javne uprave RH i EU
- program razvoja elektroničkih usluga.

U posljednjoj donesenoj Strategiji e-Hrvatska 2020. u svibnju 2017. ponovno se naglašavaju svi pozitivni učinci interoperabilnosti. No, ovdje se napominje i da s obzirom na to da je "*Hrvatski okvir interoperabilnosti razvijen je sukladno European Interoperability Framework 1.0 tj. sukladno verziji EIF 1.0*", "*Hrvatska neće raditi novi okvir interoperabilnosti već će u svim aktivnostima voditi računa o načelima Europskog okvira interoperabilnosti EIF 2.0*"⁷⁹. Možemo zaključiti da postoji prikladna strategija, ali kako će se to i provesti u praksi ovisi o nizu čimbenika, od kojih možemo u negativnom kontekstu izdvojiti trenutnu gospodarsku situaciju, a u pozitivnom kontekstu to što je Hrvatska punopravna članica EU, što sigurno doprinosi daljem ubrzanju provedbe.

⁷⁹ Strategija e-Hrvatska 2020., Ministarstvo uprave RH, svibanj 2017.

2. Poglavlje: Norme komunikacije i razmjene podataka

U ovom poglavlju naučit ćete:

- ✓ osnovno o normama i normiranju
- ✓ tipove normi
- ✓ normirna tijela
- ✓ o normama za razmjenu elektroničkih podataka
- ✓ o normama elektroničkog poslovanja
- ✓ o normama za zapis i prikaz znakova

2.1 Osnovni pojmovi normizacije

U ovom je poglavlju obrađen jedan od osnovnih preduvjeta interoperabilnosti, a to je korištenje norma. Jedino pridržavanje normi, bez obzira na to tko ih je donio i kakve su vrste, i međusobni dogovor dviju strana omogućuje da dvije strane s velikom sigurnošću uspostave komunikaciju i razmjenu podataka koje kasnije mogu koristiti. U drugom dijelu poglavlja obrađeni su različiti tipovi i vrste normi. Nakon toga slijedi popis najvažnijih normiranih tijela čije norme koristimo u nizu opisa. Na kraju dolaze potpoglavlja vezana uz praksu normi bitnih za računarstvo, informatiku i telekomunikacije, odnosno informacijsko-telekomunikacijske tehnologije. To su norme za razmjenu elektroničkih podataka, norme elektroničkog poslovanja te norme zapisa i prikaza znakova na računalu, a time i u elektroničkim dokumentima, bazama podataka, datotekama i drugim zapisima.

Norme se definiraju kao pravila, propisi, obrasci ili kriteriji prema kojima se određuje ustaljena praksa, odnosno inicijalni način kako se nešto radi. U kontekstu strukovnih, nacionalnih ili međunarodnih normi definiraju se pravila i upute za određenu aktivnost koji su odredili sami korisnici iz tog područja, opet na strukovnoj, nacionalnoj ili međunarodnoj razini.

Norma se može definirati na sljedeći način: „*Norma je **poznata i priznata mjera za određenu kvantitativnu ili kvalitativnu veličinu u okviru određene socijalne zajednice.***“

Iz navedene definicije možemo zaključiti kako je norma mjera koju je određena socijalna zajednica, koja može biti definirana na razne načine, primjerice od zemljopisne do stručne, prepoznala i priznala, odnosno dogovorila se oko njenog opetovanog korištenja, a može se odnositi na količinu ili kvalitetu.



U engleskom jeziku se uobičajeno koristi termin „*standard*“ koji se na standardni hrvatski jezik najčešće prevodi kao norma, no možemo uočiti kako je u kontekstu globalnog korištenja engleskih termina česta i upotreba termina *standard*. Ovo je leksički zanimljivo jer engleski jezik poznaje i imenicu „*norm*“ koja ima isto značenje. Inače, riječ *standard* u klasičnom smislu je naziv za zastavu, a primijetiti ćete i da smo spomenuli „standardni hrvatski jezik“, a ne „normirani hrvatski jezik“. Kako ne bi dalje uvodili zabune, za potrebe ovog kolegije smatrat ćemo da su **norma i standard istoznačnice**, s time da ćemo, prvenstveno temeljem naziva institucija koje u Hrvatskoj donose norme, ipak nešto više preferirati riječ *norma*. Glagol vezan uz normu ili standard, odnosno svođenje nečega pod normu ili standard je *normirati* ili *standardizirati*, a glagolska imenica je *normiranje*, *normizacija* ili *standardizacija*.

No, kako je red da je i sama norma normirana, na početku ćemo zbog boljeg razumijevanja gradiva, u skladu s još jednom važnom normom HRN EN 45020:2007 Normizacija i srodne djelatnosti -- Rječnik općih naziva⁸⁰, definirati pojmove korištenih u ovom poglavlju s označenim bitnim značajkama.

⁸⁰ HRN EN 45020:2007 Normizacija i srodne djelatnosti -- Rječnik općih naziva (ISO/IEC Guide 2:2004; EN 45020:2006)

U skladu s normom HRN EN 45020:2007, a prema Unutrašnjim pravilima za normizaciju (UPN) Hrvatskog zavoda za norme⁸¹, to su:

- **norma** – dokument donesen **konsenzusom** i odobren od priznatoga tijela, koji za opću i višekratnu uporabu daje **pravila, upute ili značajke** za djelatnosti ili njihove rezultate radi postizanja najboljeg stupnja uređenosti u danome kontekstu, uz sljedeću napomenu:
 - norme bi se trebale temeljiti na provjerenim znanstvenim, tehničkim i iskustvenim rezultatima, i biti usmjerene promicanju najboljih prednosti za društvo,
- **normizacija** – djelatnost uspostavljanja **odredaba** za **opću i višekratnu uporabu** koje se odnose na postojeće ili moguće probleme radi postizanja **najboljeg stupnja uređenosti** u danome kontekstu, uz sljedeće napomene:
 - djelatnost se prvenstveno sastoji od oblikovanja, izdavanja i primjene norma,
 - važne koristi normizacije su poboljšavanje prikladnosti proizvoda, procesa i usluga za njihove predviđene svrhe, otklanjanje zapreka u trgovini te olakšavanje tehničke suradnje,
- **konsenzus** – **opće slaganje** koje se odlikuje **odsutnošću čvrstoga protivljenja** bitnim sadržajima od strane znatnoga dijela interesnih skupina i procesom u kojemu se nastoje uzeti u obzir gledišta svih zainteresiranih strana te uskladiti oprečna stajališta, uz napomenu da:
 - konsenzus nužno ne znači jednoglasnost,
- **propis** – dokument koji sadrži obvezatna zakonska pravila, a donosi ga upravno tijelo.

Normizacija se definira kao djelatnost koja uključuje izradu dokumenata primjenom načela konsenzusa u priznatome tijelu i dragovoljnu uporabu tih dokumenata za zajedničku dobrobit, a norme se pripremaju iz različitih razloga i za različite primjene.

U skladu s definicijama već spomenute norme HRN EN 45020:2007 Normizacija i srodne djelatnosti -- Rječnik općih naziva i Unutrašnjim pravilima za normizaciju (UPN) Hrvatskog zavoda za norme⁸², definirat ćemo i sljedeće važne pojmove:

- **normativni dokument** – dokument koji daje pravila, upute ili značajke za različite djelatnosti ili njihove rezultate, uz sljedeće napomene:
 - naziv „normativni dokument“ rodni je naziv koji obuhvaća dokumente kao što su norme, tehničke specifikacije, kodeksi dobre prakse i propisi,
 - "dokumentom" se smatra svaki medij na kojemu je zapisana određena obavijest,
 - nazivi za različite vrste normativnih dokumenata određuju se tako da uzimaju u obzir dokument i njegov sadržaj kao jedinstvenu cjelinu,
- **nacionalna norma** – norma dostupna javnosti koju je prihvatilo nacionalno normirno tijelo,
- **hrvatska norma (HRN)** – norma dostupna javnosti koju je prihvatio Hrvatski zavod za norme (HZN),

⁸¹ UPN 1:2014, Unutrašnja pravila za normizaciju – 1. dio: Normizacija općenito, ciljevi i osnovna načela, Hrvatski zavod za norme, 2014., https://www.hzn.hr/UserDocImages/pdf/UPN_1_2014-02-20.pdf

⁸² UPN 2:2014, Unutrašnja pravila za normizaciju – 2. dio: Vrste dokumenata i njihovo označivanje, Hrvatski zavod za norme, 2014., https://www.hzn.hr/UserDocImages/pdf/UPN_2_2014-02-20.pdf

- **prednorma** – dokument dostupan javnosti koji je privremeno prihvatio koje normizacijsko tijelo i učinilo dostupnim javnosti kako bi se njegovom primjenom steklo potrebno iskustvo koje je osnova za izradu norme,
- **hrvatska prednorma (HRS ENV)** – dokument koji je privremeno prihvatio Hrvatski Zavod za norme (HZN) i učinio dostupnim javnosti kako bi se njegovom primjenom steklo potrebno iskustvo koje je osnova za izradu hrvatske norme, uz sljedeće napomene:
 - prednorma na europskoj razini ima status tehničke specifikacije, tj. nije postignut konsenzus da se njezin sadržaj objavi kao norma,
 - status hrvatske prednorme može imati samo prihvaćena europska prednorma
- **tehnička specifikacija** – dokument u kojemu se propisuju tehnički zahtjevi koje treba zadovoljiti kakav proizvod, proces ili usluga, uz sljedeće napomene:
 - kad god je to potrebno, tehnička specifikacija treba naznačiti postupke s pomoću kojih se može odrediti jesu li ispunjeni dani zahtjevi,
 - tehnička specifikacija može biti norma, dio norme ili poseban dokument neovisan o normi,
- **hrvatska tehnička specifikacija (HRS)** – normativni dokument Hrvatskog zavoda za norme (HZN) dostupan javnosti u kojemu se propisuju tehnički zahtjevi kojima treba udovoljiti kakav proizvod, proces ili usluga; uz napomenu da:
 - je tehnička specifikacija dokument koji sadržava normativne elemente, ali nije postignut konsenzus da se njegov sadržaj objavi kao norma,
- **tehnički izvještaj – obavijesni dokument** koji nije prikladan za objavu kao norma ili tehnička specifikacija,
- **hrvatski tehnički izvještaj (HRI)** – obavijesni dokument Hrvatskog zavoda za norme (HZN) koji nije prikladan za objavu kao hrvatska norma ili hrvatska tehnička specifikacija,
- **upute** – obavijesni dokument koji daje smjer, savjet ili preporuku za normizacijska načela i politiku te smjernice za pisce norma
- **HZN upute (HRU)** – obavijesni dokument Hrvatskog zavoda za norme (HZN) dostupan javnosti koji daje smjer, savjet ili preporuku za normizacijska načela i politiku te smjernice za pisce norma.

Osim toga, skladu s Unutrašnjim propisima CEN/CENELEC-a⁸³ definirana je i europska norma:

- **europska norma (EN)** – norma prihvaćena od Europskog odbora za normizaciju (CEN) ili Europskog odbora za elektrotehničku normizaciju (CENELEC) koja nosi sa sobom obvezu primjene kao istovjetne nacionalne norme i povlačenje proturječnih nacionalnih norma.



Iz navedenih definicija možemo zaključiti da postoje mnogi drugi dokumenti različiti od norma kao što su tehničke specifikacije, tehnički izvještaji, upute, kodeksi dobre prakse, preporuke, smjernice, izvještaji, tehnički sporazumi i slično. Bitna razlika između norma i ostalih dokumenata je različita razina konsenzusa potrebna za njihovo donošenje.

⁸³ CEN/CENELEC Internal Regulations - Part 2, 2011, definicija 2.5

Neke od **vrsta norma** koje najčešće susrećemo su: osnovna norma, terminološka norma, norma za ispitivanje, norma za proizvod, norma za proces, norma za uslugu, norma za sučelje i norma o potrebnim podacima.

Kao što je vidljivo iz definicija, norme i tehničke specifikacije sadržavaju odredbe kojima treba udovoljiti kakav proizvod, proces ili usluga, koji se nazivaju **normativnim elementima**, dok ostale vrste dokumenata sadržavaju samo obavijesne odredbe koje se nazivaju **obavijesnim elementima**. Sadržaj norme, koja sadržava i normativne i obavijesne elemente, može biti podijeljen u normativni i obavijesni dio.



Normativni i obavijesni elementi se razlikuju kontekstom i izričajem, pa se tako primjerice **preporuke** se izražavaju izrazom „**treba**“, a **zahtjevi** izrazom „**mora**“.

2.1.1 Ciljevi i načela normizacije

Opći ciljevi normizacije proizlaze iz same definicije normizacije, a to su osiguranje prikladnosti kojega proizvoda, procesa ili usluge da u određenim uvjetima služi svojoj namjeni, ograničivanje raznolikosti izborom optimalnoga broja tipova ili veličina, osiguravanje spojivosti različitih proizvoda, zaštita zdravlja, sigurnost, zaštita okoliša itd.

Ciljevi normizacije zapravo se mogu sažeti kao doprinos općemu napretku kroz sljedeće aspekte proizvoda, procesa ili usluge:

- osiguranje prikladnosti,
- povećanje razine sigurnosti,
- očuvanje zdravlja i života ljudi i životinja te zaštitu okoliša,
- poboljšanje proizvodne učinkovitosti i gospodarstveno ponašanje,
- uklanjanje tehničkih zapreka u međunarodnoj trgovini.

Osnovno načelo normizacije je **konsenzus**, no, postoje i druga važna **načela**, kao što su uključivanje svih zainteresiranih strana, javnost rada, stupanj razvoja tehnike i koherentnost zbirke norma. **Konsenzus** možemo definirati kao **opće slaganje**. Ideja općeg slaganja uključuje da ne postoji čvrsto protivljenje važnim sadržajima od strane većine ili znatnoga dijela interesnih skupina. U procesu dolaska do općeg slaganja treba se nastojati uzeti u obzir gledišta svih zainteresiranih strana te, ako je moguće, uskladiti oprečna stajališta. No, isto tako konsenzus nužno ne znači jednoglasnost, već zapravo odluku većine. Demokratski postupak pripreme norma pretpostavlja **uključivanje svih zainteresiranih strana** koje imaju pravo sudjelovati i dati svoj doprinos izradbi norme kako bi je dragovoljno primijenili. Postupak pripreme norma mora biti **dostupan javnosti** od svojega početka i u svim fazama. O početku pripreme koje norme, o tijelu koje je priprema, o dokumentu koji služi kao osnova za njezinu pripremu i o fazama pripreme (rasprava o nacrtu norme i izdavanje norme) javnost mora biti obaviještena na odgovarajući način. Norma definira i „**stanje tehnike**“ kao stupanj razvoja tehnike u danome vremenu utemeljen na provjerenim znanstvenim, tehničkim i iskustvenim spoznajama. Zbirka norma mora biti **koherentna** te norme ne mogu biti proturječne, a donošenjem nove norme se stara norma povlači.

Glavne su **značajke norma** s gledišta javnosti da:

- se temelje na provjerenim znanstvenim, tehničkim i iskustvenim rezultatima
- su usmjerene na promicanje najboljih prednosti za društvo, čime predstavljaju dogovornu osnovu za procjenjivanje proizvoda, procesa ili pružanja usluga, posebno s obzirom na sigurnosne zahtjeve i sprečavanja ozljeđivanja,
- nude nedvosmislene tehničke kriterije,
- su univerzalno prepoznatljive i upotrebljavane.



No, realnost je i da **norme djelomično potiskuju inovativnost**, jer ako se pridržavate nečeg što postoji, nikad nećete osmisлити nešto novo, a ako i hoćete, tržište to tek treba prihvatiti.

U kontekstu interoperabilnosti norme bi trebale:

- olakšati suradnju dvaju subjekata,
- olakšati povezivanje dvaju subjekata,
- omogućiti konkurentnost na tržištu,
- kod potrošača omogućiti neovisnost o proizvođaču.

2.2 Tipovi normi

Do norme se najčešće dolazi tako da određena interesna skupina ima potrebu za dogovorom oko nekog pitanja ili problema pa saziva zajedničko tijelo koje, nakon analize, predlaže neko rješenje o kojem se raspravlja, a kada se postigne dogovor, ono što je dogovoreno izrađuje se u obliku norme i objavljuje kroz neko normirno tijelo. Norme može predlagati određeni proizvođač, udruga proizvođača, interesna udruga ili netko drugi. Primjerice, ako mi sami napišemo svoju normu i objavimo je, recimo na našoj internetskoj stranici, to nam neće previše pomoći da se naša norma prihvati. Prihvaćanje norme se u pravilu događa na dvama razinama, pravnoj i tržišnoj.

2.2.1 Pravne i tržišne norme

Neovisna tijela koja formalno donose norme imaju snagu pravne podrške normi i tada govorimo u tzv. **pravnim normama** ili **de iure normama**. Drugi oblik normi su one koje su široko prihvaćene na tržištu i tada govorimo o tzv. **tržišnim normama** ili **de facto normama**.

Tip norme neizravno uvjetuje i **mogućnost izbora proizvoda** koji se pridržava norme na tržištu i **rizik kod korištenja** te norme i proizvoda usklađenih s normom. Tako pravna norma nudi veliku mogućnost izbora uz relativno mali rizik, dok norma vezana uz proizvod, odnosno proizvođača ili udrugu proizvođača nosi malu mogućnost izbora, jer se praktički bira samo između proizvoda tih proizvođača uz relativno veliki rizik. Norme vezane uz licencije i interesne udruge nude srednju mogućnost izbora i srednju raznu rizika.



Ako se ikad zapitamo što se sve u našoj okolini može normirati, odgovor leži u **Katalogu hrvatskih norma** po ICS-u⁸⁴, koji sadržava sva sljedeća područja normi (navedeno ilustrativno): opći pojmovi, nazivlje, dokumentacija, usluge, poduzeća, uprava, prijevoz, matematika, prirodne znanosti, sociologija, zdravstvena skrb, zaštita zdravlja i okoliša, sigurnost, meteorologija, fizikalne pojave, ispitivanja, mehanički sustavi, fluidni sustavi, proizvodna tehnika, tehnika prijenosa energije i topline, elektrotehnika, elektronika, telekomunikacije, informacijska tehnika, uredski strojevi, precizna mehanika, nakit, cestovna vozila, željeznička tehnika, brodogradnja, zrakoplovna i svemirska tehnika, rukovanje gradivima, pakiranje i raspačavanje roba, tekstilna i kožna tehnologija, odjevna industrija, poljoprivreda, prehrambena tehnologija, kemijska tehnologija, rudarstvo, naftna tehnologija, metalurgija, drvna tehnologija, industrija stakla i keramike, industrija gume i plastike, papirna tehnologija, industrija boja, gradnja, vojna tehnika, oprema za kućanstvo, zabava i šport. Ovaj dugačak popis naravno ne morate znati, ali on nam slikovito govori da se može **normirati praktički bilo koji segment ljudske aktivnosti i današnjeg društva** te da je pregršt normi u uporabi svakodnevno oko nas, iako toga nismo ni svjesni.

2.2.2 Tehničke, semantičke i otvorene norme

Tehničke norme (engl. *technical standards*) su najčešće formalni dokumenti koji utvrđuju određeni uniformni inženjering, tehničke uvjete, metode, procese i prakse. Osnovni oblici tehničkih normi su:

- **specifikacije** – označavaju određeni skup zahtjeva na proizvod, materijal, komponentu, sustav ili uslugu, često kao dogovor konačne izvedbe nekog produkta ili usluge,
- **metode** – označavaju procedure koje proizvode određeni rezultat, a mogu se odnositi i na testne procedure za ispitivanje određenog elementa ili usluge,
- **prakse** – označavaju skup instrukcija koje određuju operacije ili funkcije,
- **smjernice** – označavaju opće informacije ili mogućnosti bez konkretnog opisa akcija,
- **definicije** – označavaju formalno utemeljenu terminologiju ili vokabulare,
- **jedinice** – to su prihvaćene mjere kvantiteta fizikalnih veličina.

Postoje i **semantičke norme** (engl. *semantic standards*), koje definiraju značenja elemenata u komunikaciji, a najviše se odnose na pojedinu struku ili poslovnu djelatnost. Najčešće se koriste za definiranje poslovne semantike za poslovne sadržaje kao što su poruke i poslovni objekti koji se mogu koristiti unutar različitih gospodarskih grana.

Otvorenost normi i tehničkih specifikacija je vrlo važan aspekt interoperabilnosti kao i slobode izbora jer otvorenost spušta granice ulaska u tržište, jača kompetitivnost te time omogućuje veći izbor, višu kvalitetu i niže troškove. Ako se pravilno pozicionira otvorenost može potpomagati inovaciji, primjerice uključivanjem zainteresiranih strana u izradu normi, a ne sputavati je. U svakom slučaju otvorenost jača položaj korisnika i pruža veće mogućnosti pri

⁸⁴ Katalog hrvatskih norma po ICS-u, <http://www.hzn.hr/default.aspx?id=251>

odabiru proizvoda i dobavljača koji su sukladni normama. Na kraju otvorenost potiče interoperabilnost kroz transparentnost razmjene podataka, što često potiče razvoj više razine sigurnosti, a sve to znači izbjegavanje ovisnosti o određenom proizvođaču (engl. *lock-in*) ili proizvodu i potencijalne „pat pozicije“ u budućnosti.

Pod **otvorenom normom** (engl. *open standard*) smatra se norma koje je javno dostupna, s različitim pravilima korištenja koja često podrazumijevaju jednostavnu dostupnost, ali ne postoji jedinstvena definicija pa se interpretacije pojma otvorene norme razlikuju od slučaja do slučaja. Problem ove nedorečenosti je u višeznačnosti pojma „otvorenost“ koji može stavljati naglasak na različite značajke, kao što su otvorenost posljedične specifikacije, otvorenost postupka donošenja norme i sudjelovanja u istome, otvorenost prava i slično. U akademskoj zajednici većine europskih država otvorenost norme znači da ne postoji mogućnost komercijalnog iskorištavanja norme, što se definira i licencijskim pravilima. No, kod nekih otvorenih normi to može podrazumijevati određene razumne i nediskriminatorne cijene, ali ne nužno i korištenje bez plaćanja. Većina međunarodnih normiranih organizacije naplaćuje određene cijene vezane uz licencije pod kojima su norme objavljene. Osim toga, katkad se otvorenost norme povezuje s idejom otvorenosti koda te se smatra da samo referentna implementacija otvorenog koda može definirati i otvorenost norme.



Izjava vezana uz otvorenost normi usvojena od Vlade RH⁸⁵: „Otvorene norme, kao premisa razvitka interoperabilnosti informacijskih sustava tijela državne uprave, javno su dostupne i objavljene na način da je omogućeno njihovo opetovano i slobodno korištenje uz naknade sukladne troškovima objave i dostave, a bez ograničenja u pogledu zaštite intelektualnog vlasništva. Time se otvorene norme pojavljuju kao javno dobro, što pridonosi snažnijem sudjelovanju naših stručnjaka kako u primjeni tako i u donošenju otvorenih normi.“

Postoji niz razloga koji **povećavaju svakodnevnu uporabu otvorenih normi** kod sve većeg broja korisnika⁸⁶. Jedan od glavnih razloga korištenja otvorenih normi je **smanjenje troškova** koji nastaju kao posljedica nabave licencija za određene informacijske sustave i aplikacije, počevši od operacijskih sustava, preko uredskih aplikacija, poslovnih informacijskih sustava kao što su ERP (engl. *Enterprise Resource Planning*) i CRM (engl. *Customer Relationship Management*), specijaliziranih i drugih aplikacija, do troškova samih norma i korištenja određenih vlasničkih (engl. *proprietary*) tehnologija. Drugi važan razlog je **mogućnost slabljenja veze za određenog dobavljača i jednostavnija mogućnost promjene dobavljača, ali i tehnologija**. Razlog je i što se u općem smislu otvorene norme brže razvijaju i unose u svoju primjenu novija, naprednija i sigurnija rješenja, a svaki korisnik može sudjelovati u unaprjeđenju normi i posljedičnih sustava koji ih koriste.

⁸⁵ Odrednice razvitka i uporabe računalnih programa s otvorenim kodom u tijelima državne uprave, e-Hrvatska, Vlada RH, 2006.

⁸⁶ Značaj otvorenih normi u Metodološkom priručniku za evaluaciju, Isporuca 2, Fakultet elektrotehnike i računarstva, Sveučilište u Zagrebu, Zagreb, 2009.

2.3 Načini izrade norma

Hrvatske norme izrađuju **tehnički odbori** Hrvatskoga zavoda za norme (HZN/TO) **postupkom od šest faza**⁸⁷. Takav dokument daje se izravno na odobravanje kao nacrt hrvatske norme na javnu raspravu, bez prolaženja kroz prethodne faze. U nastavku slijedi kratki opis svake od šest faza:

- 1. faza: **Poticaj** – prvi korak u izradi hrvatske norme je potvrda da je određena norma potrebna ili da je potrebno prihvatiti određenu europsku/međunarodnu normu, a prijedlog za novi projekt dostavlja se Tehničkoj upravi (TU) koja donosi odluku o uključivanju novoga projekta u program rada odgovarajućeg HZN/TO-a ili odbija prijedlog,
- 2. faza: **Pripremna faza** – za pripremu radnog nacrt hrvatske norme HZN/TO ili TU obično osniva radnu skupinu sastavljenu od stručnjaka za to područje; mogu se razmatrati uzastopni radni nacrti sve do izrade najboljega tehničkog rješenja za predmet na koji se ono odnosi; u toj se fazi radni nacrt proslijeđuje matičnomu odboru radne skupine ili TU-u radi dobivanja suglasnosti,
- 3. faza: **Nacrt odbora** – nakon što se radni nacrt odobri kao nacrt odbora, tajništvo odbora ga registrira. Nacrt odbora se distribuira članovima HZN/TO-a radi komentiranja i odobravanja. Mogu se razmatrati uzastopni nacrti odbora sve dok se ne postigne konsenzus o tehničkom sadržaju. Kad se konsenzus postigne, tekst se dovršava kao nacrt hrvatske norme
- 4. faza: **Javna rasprava** – nacrt hrvatske norme daje se na javnu raspravu i davanje primjedaba u roku od dva mjeseca; ako pristignu primjedbe tehničke prirode, tekst se vraća HZN/TO-u koji ga je izradio na daljnje razmatranje i prerađeni dokument će se ponovno dati na javnu raspravu kao nacrt hrvatske norme,
- 5. faza: **Odobrovanje** – ako tijekom javne rasprave ne pristignu nikakve primjedbe na nacrt hrvatske norme ili pristignu samo primjedbe uredničke prirode, tekst se uređuje i odobrava kao konačni nacrt hrvatske norme; HZN/TO koji ga je izradio mora odobriti konačni tekst kao hrvatsku normu,
- 6. faza: **Objava** – jednom kad je konačni tekst hrvatske norme već odobren, u njega se smiju unositi samo manje uredničke izmjene, ako i gdje je to potrebno. Konačni tekst se mora odobriti za objavu kao hrvatska norma (HRN).

Ovih šest glavnih faza dodatno se dijeli na podfaze označene kodovima, koji se posljedično koriste u označavanju statusa izrade pojedine norme.



U nekim slučajevima, kad nisu osnovani odgovarajući TO-i, europske se norme smiju, ako je potrebno, prihvatiti kao hrvatske norme po takozvanom "skraćenom postupku". Kod skraćenog postupka moguće je ispustiti određene faze. Dodatno, svakih pet godina HZN/TO-i preispituju hrvatske norme koje su izradili. Većinom glasova članova HZN/TO-a donosi se odluka hoće li hrvatska norma biti potvrđena, prerađena ili povučena.

⁸⁷ UPN 3:2014, Unutrašnja pravila za normizaciju – 3. dio: Izrada i donošenje hrvatskih norma i drugih dokumenata, Hrvatski zavod za norme, 2014., https://www.hzn.hr/UserDocImages/pdf/UPN_3_2014-02-20.pdf

2.4 Važnija normirna tijela

Svjetski sustav normizacije uključuje normizacijsku djelatnost na trima razinama: međunarodnoj, regionalnoj (europskoj) i nacionalnoj (hrvatskoj) razini.

Međunarodne organizacije za normizaciju koje povezuju svjetski sustav normizacije međusobnim sporazumima o suradnji, a u koje se mogu učlaniti nacionalna normirna tijela, su:

- Međunarodna organizacija za normizaciju **ISO** (*International Organization for Standardization*),
- Međunarodno elektrotehničko povjerenstvo **IEC** (*International Electrotechnical Commission*),
- Međunarodna telekomunikacijska unija **ITU** (*International Telecommunication Union*).

Načelno, IEC obuhvaća tehnološka područja elektrotehničke i elektroničke inženjerske tehnologije, ITU telekomunikacije i radiokomunikacije, a sva druga područja obuhvaća ISO. Slične sporazume o suradnji provode se između organizacija za normizaciju na regionalnoj i nacionalnoj razini.

Najpoznatije **regionalne normirne organizacije** na području Europske unije, u koje se učlanjuju nacionalna tijela svake zemlje iz nekog zemljopisnog, političkog ili gospodarskog područja, su:

- Europski odbor za normizaciju **CEN** (*European Committee for Standardization*),
- Europski odbor za elektrotehničku normizaciju **CENELEC** (*European Committee for Electrotechnical Standardization*),
- Europski institut za telekomunikacijske norme **ETSI** (*European Telecommunications Standards Institute*).

Nacionalna normirna organizacija je normirno tijelo priznato na nacionalnoj razini koje ima pravo biti nacionalnim članom odgovarajućih međunarodnih i regionalnih normirnih organizacija. U Hrvatskoj to je **Hrvatski zavod za norme**, dok su neka druga poznatija nacionalna normirna tijela:

- Normirna udruga Francuske AFNOR (franc. *Association Française de Normalisation*)
- Američki nacionalni normirni institut ANSI (engl. *American National Standards Institute*)
- Britanski institut za norme BSI (engl. *British Standards Institution*)
- Njemački institut za norme DIN (njem. *Deutsches Institut für Normung e.V.*)
- Austrijski institut za norme ON (njem. *Österreichisches Normungsinstitut*)
- Slovenski institut za norme SIS (slov. *Slovenski inštitut za standardizacijo*)
- Nacionalna normirna organizacija Italije UNI (tal. *Ente nazionale italiano di unificazione*)

Međunarodne, regionalne i nacionalne normirne organizacije, osim što su međusobno povezane sporazumima o suradnji, prihvaćaju i dogovorene postupke te načine suradnje koji su navedeni u uputama Kodeksa dobrih praksi za normiranje⁸⁸.

⁸⁸ ISO/IEC Guide 59:2019 ISO and IEC recommended practices for standardization by national bodies, <https://www.iso.org/standard/71917.html>

Osim toga postoji i čitav niz **strukovnih normirnih** i drugih **organizacija** važnih u kontekstu interoperabilnosti od kojih ćemo izdvojiti nekoliko **najvažnijih za područje računarstva i informatike**:

- Institut inženjera elektrotehnike i elektronike **IEEE** (engl. *Institute of Electrical and Electronics Engineers*)
- Konzorcij World Wide Weba **W3C** (engl. *World Wide Web Consortium*)
- Međunarodno radno tijelo za inženjering Interneta **IETF** (engl. *Internet Engineering Task Force*)
- Autoritet za brojeve pridružene Internetu **IANA** (engl. *Internet Assigned Numbers Authority*)
- Centar Ujedinjenih naroda za omogućavanje trgovanja i elektroničkog poslovanja **UN/CEFACT** (engl. *United Nations Centre for Trade Facilitation and Electronic Business*), kao podružnica Centra Ekonomske komisije Ujedinjenih naroda za Europu UNECE (*United Nations Economic Commission for Europe*)
- Organizacija za unapređivanje strukturiranih informacijskih normi **OASIS** (engl. *Organization for the Advancement of Structured Information Standards*)
- Međunarodna organizacija GS1

Kako bi olakšali prepoznavanje normirnih organizacija i njihove uloge donosimo i sliku razina te oznaka (logotipa) važnijih normirnih organizacija s aspekta interoperabilnosti.



Slika 2.1: Razine i oznake (logotipi) normirnih organizacija

2.4.1 Hrvatski zavod za norme HZN

Hrvatski zavod za norme (HZN)⁸⁹ je neovisna i neprofitna javna ustanova osnovana kao nacionalno normirno tijelo Republike Hrvatske radi ostvarivanja ciljeva normizacije:

- a) osiguranje prikladnosti kojega proizvoda, procesa ili usluge
- b) povećanje razine sigurnosti proizvoda, procesa ili usluga, čuvanja zdravlja, života ljudi i životinja te zaštite okoliša
- c) poboljšanje proizvodne učinkovitosti i gospodarno ponašanje
- d) uklanjanje tehničkih zapreka u međunarodnoj trgovini.

Hrvatski zavod za norme je osnovala Vlada Republike Hrvatske na sjednici održanoj 27. listopada 2004. godine donijevši Uredbu o osnivanju Hrvatskog zavoda za norme kao javne ustanove za ostvarivanje ciljeva normizacije i obavljanje poslova i zadataka nacionalne normizacije. Službeni skraćeni naziv ustanove je HZN, a naziv na engleskome jeziku je *Croatian Standards Institute*.



Povijesno, kada je nakon osnivanja Hrvatske kao samostalne države ovlaštenost za područje normiranja, certificiranja i mjeriteljstva prenesena s bivše države, utemeljen je prvo Republički zavod za normizaciju i mjeriteljstvo, koji je 1992. godine promijenio naziv u Državni zavod za normizaciju i mjeriteljstvo (DZNM), a kasnije je HZN preuzeo sve djelatnosti iz djelokruga DZNM-a.

U Hrvatskoj, osim HZN-a, djeluju i **Hrvatska akreditacijska agencija (HAA)**⁹⁰ (engl. *Croatian Accreditation Agency*), neovisna i neprofitna javna ustanova koja obavlja poslove nacionalne službe za akreditaciju u Republici Hrvatskoj, te **Državni zavod za mjeriteljstvo (DZM)**⁹¹.

Zakon o normizaciji⁹² donesen 2013. godine, jedan je od pet osnovnih zakona u području tehničkog zakonodavstva, a prema obvezama koje proizlaze iz Ugovora o pristupanju Republike Hrvatske Europskoj uniji. Ovim se zakonom uređuju ciljevi i načela hrvatske normizacije, određuje nadležno tijelo za provedbu uredbi i direktiva Europskoga parlamenta i Vijeća o europskoj normizaciji, kao i osnivanje, ustrojstvo i djelatnost nacionalnog normirnog tijela, pripremanje i izdavanje hrvatskih norma i njihova uporaba i ostala vezana pitanja. Osim njega još su važni i Zakon o tehničkim zahtjevima za proizvode i ocjenjivanju sukladnosti⁹³, Zakon o općoj sigurnosti proizvoda⁹⁴, Zakon o akreditaciji⁹⁵ i Zakon o mjeriteljstvu⁹⁶.

U skladu sa Zakonom o normizaciji i Uredbom o osnivanju HZN-a, dio djelatnosti HZN-a obuhvaća poslove koji su važni za Republiku Hrvatsku i obavlja ih isključivo HZN, a to su:

⁸⁹ Hrvatski zavod za norme, HZN, <http://www.hzn.hr>

⁹⁰ Hrvatska akreditacijska agencija, HAA, <http://www.akreditacija.hr>

⁹¹ Državni zavod za mjeriteljstvo, <https://dzm.gov.hr/>

⁹² Zakon o normizaciji, Narodne novine, br. 80/2013.

⁹³ Zakon o tehničkim zahtjevima za proizvode i ocjenjivanju sukladnosti, Narodne novine, br. 80/2013., 14/2014. i 32/2019.

⁹⁴ Zakon o općoj sigurnosti proizvoda, Narodne novine, br. 30/2009., 135/2010., 14/2014. i 32/2019.

⁹⁵ Zakon o akreditaciji, Narodne novine, br. 158/2003., 75/2009. i 56/2013.

⁹⁶ Zakon o mjeriteljstvu, Narodne novine, br. 74/2014. i 111/2018.

- pripremanje, prihvaćanje i objavljivanje hrvatskih normi i drugih dokumenata iz područja normizacije,
- predstavljanje hrvatske normizacije u međunarodnim i europskim normizacijskim organizacijama,
- održavanje zbirke hrvatskih normi i vođenje registra hrvatskih normi,
- uređivanje, objavljivanje i distribuiranje hrvatskih normi, drugih dokumenata i publikacija iz područja normizacije,
- uspostavljanje i održavanje baze podataka o normama i drugim dokumentima iz područja normizacije te davanje obavijesti o normama i drugim dokumentima,
- objavljivanje obavijesti o hrvatskim normama te obavijesti o drugim dokumentima iz područja normizacije u službenom glasilu,
- osiguravanje informacija o nacionalnim, europskim i međunarodnim normama cjelokupnoj javnosti, a posebno gospodarstvu.

Hrvatski zavod za norme je član Međunarodne organizacije za normizaciju ISO, Međunarodnog elektrotehničkog povjerenstva IEC, Europskog odbora za normizaciju CEN, Europskog odbora za elektrotehničku normizaciju CENELEC i Europskog instituta za telekomunikacijske norme ETSI.

Danas u Hrvatskoj postoji više od 12.000 prihvaćenih normi, od čega je svega dvjestotinjak na hrvatskom jeziku, a ostale su na jeziku izvornika. HZN prodaje norme prema važećem cjeniku, a u HZN se mogu učlaniti pravne i fizičke osobe, kao redovni članovi ili kao promatrači. Stručni rad članova u tehničkim odborima je volonterski.



Normoteka HZN-a je specijalna „knjižnica“ koja pohranjuje i održava zbirke norma i drugih normativnih dokumenata s pratećim katalozima, časopisima i bazama podataka putem kojih se pretražuju i pronalaze odgovarajući podaci o normama i drugim dokumentima iz područja normizacije. Kao referentno mjesto za norme i normativne dokumente, korisnicima norma omogućuje se uvid u fond i pristup svim raspoloživim informacijama. Osiguran je prostor i oprema za rad korisnika te stručna pomoć osoblja pri pretraživanju i definiranju zahtjeva.

Ustrojstvo Hrvatskog zavoda za norme uključuje niz tijela: Upravno vijeće, Stručno vijeće, Ravnatelja, Savjet za norme, Tehničku upravu, Programske odbore i Tehničke odbore. U HZN-u postoji niz **tehničkih odbora (TO)** koji prate organizacijsku strukturu ISO, pa nose i slične oznake, a s razine informatike, telekomunikacija i računarstva te interoperabilnosti važniji su:

- TO Z1 – Informacijska tehnologija – pokriva ista područja kao ISO/IEC JTC 1, ISO/IEC JTC1/SC 34, CEN/TC 224 i CEN/TC 225,
- TU Z1 – Informatika 1 – pokriva isto područje kao CEN TC/365,
- TU TK1 – Telekomunikacije 1,
- TO T3 – Nazivlje u telekomunikacijama,
- TO T4 – Normizacija u telekomunikacijama – pokriva područja niza odbora ETSI/TC.

Normativni dokumenti HZN-a obuhvaćaju raznolike dokumente, kao što su norme, tehničke specifikacije, kodeksi dobre prakse, propisi i drugi. Dokumenti HZN-a mogu biti izvorni hrvatski dokumenti i hrvatski dokumenti koji nastaju prihvaćanjem međunarodnih, europskih i drugih dokumenata normiranih tijela.

U HZN-u se izrađuju i objavljuju sljedeće vrste **dokumenata**:

- **hrvatska norma**, oznaka **HRN**,
- **hrvatska prednorma**, oznaka **HRS ENV**,
- **hrvatska tehnička specifikacija**, oznaka **HRS**,
- **hrvatski tehnički izvještaj**, oznaka **HRI**,
- **amandman**, oznaka **A**,
- **ispravak**, oznaka **Ispr.**,
- **HZN upute**, oznaka **HRU**.

Hrvatski normativni dokumenti podliježu **pravilima označavanja** dokumenata u obliku **referencijske oznake**. Referencijska oznaka izvornoga hrvatskog dokumenta sastavljena je od **slovne i brojčane oznake** hrvatskog dokumenta i godine objave toga dokumenta razdvojene dvotočkom. Referencijska oznaka izvorne hrvatske norme sastoji od prefiksa HRN, slova i brojaka u nizu od 1000 naviše i godine izdanja u obliku **HRN nnnn:gggg**, kao što možete vidjeti u sljedećoj tablici, zajedno s primjerima ostalih izvornih hrvatskih normativnih dokumenata.

Vrsta	Slovna oznaka	Referencijska oznaka	Objašnjenje	Primjer
Hrvatska norma	HRN	HRN nnnn:gggg	nnnn >1000	HRN 1015:2004
Hrvatska tehnička specifikacija	HRS	HRS nnnn:gggg	nnnn >1000	HRS 1111:2005
Hrvatski tehnički izvještaj	HRI	HRI nnnn:gggg	nnnn >1000	HRS 1236:2006
HZN upute	HRU	HRU nnn:gggg	nnn <1000	HRS 10:2005

Tablica 2.2: Oznake izvornih hrvatskih normativnih dokumenata

Hrvatski zavod za norme **prihvaća** kao hrvatske dokumente različite vrste **međunarodnih i europskih normativnih dokumenata**, kao istovjetne ili preinačene hrvatske norme, a to su:

- norme (IEC, ISO, EN, ETS),
- dokumente za usklađivanje (HD),
- prednorme (ENV),
- tehničke specifikacije (TS) i javno dostupne specifikacija (PAS),
- tehničke izvještaje (TR) i upute (EG, Guide).

Referencijska oznaka hrvatskoga normativnog dokumenta nastalog prihvaćanjem međunarodnog, europskog ili dokumenata normiranih tijela drugih država sastavljena je od slovne oznake hrvatskoga dokumenta i slovne i brojčane oznake preuzetog dokumenta te godine objave hrvatskoga dokumenta razdvojene dvotočkom. Dakle, ako je hrvatska norma nastala prihvaćanjem međunarodne ili europske norme od tijela kao što su IEC, ISO, EN ili ETS, onda se prije oznake dokumenta umeće i kratica ustanove koja je izdala normu, npr. HRN ISO 25123:2004, a u sljedećoj tablici možete vidjeti primjere različitih dokumenata.

Vrsta	Slovna oznaka	Referencijska oznaka izvornog dokumenta	Hrvatska referencijska oznaka	Primjer
Norma	IEC ISO EN ETS	IEC nnnn:gggg ISO nnnn:gggg EN nnnn:gggg ETS nnnn:gggg	HRN IEC nnnn:gggg HRN ISO nnnn:gggg HRN EN nnnn:gggg HRN ETS nnnn:gggg	HRN IEC 15671:2005 HRN ISO 25123:2004 HRN EN 1324:1999 HRN ETS 12345:2006
Dokument za usklađivanje	HD	HD nnn:gggg	HRN HD nnn:gggg	HRN HD 123:2003
Prednorma	ENV	ENV nnnn:gggg	HRN ENV nnnn:gggg	HRN ENV 4593:2001
Tehnička spec.	ust/TS	ust/TS nnn:gggg	HRS ust/TS nnn:gggg	HRS IEC/TS 123:2006
Javno dost. spec.	ust/PAS	ust/PAS nnn:gggg	HRS ust/PAS nnn:gggg	HRS IEC/PAS 12876:2006
Tehnički izvještaj	ust/TR CR	ust/TR nnnn:gggg CR nnnn:gggg	HRI ust/TR nnnn:gggg HRI CR nnnn:gggg	HRI ISO/TR 7654:2004 HRI CR 9876:2000
Upute	ust Guide ust EG ust/EG	ust Guide nnn:gggg ust EG nnnn:gggg ust/EG nnnn:gggg	HRU ust Guide nnn:gggg HRU ust EG nnnn:gggg HRU ust/EG nnnn:gggg	HRU ISO Guide 12:2006 HRU ETSI EG 235:2006 HRU CLC/EG

Tablica 2.3: Oznake istovjetno prihvaćenih međunarodnih dokumenata

Referencijska oznaka hrvatskoga dokumenta može se dopuniti dvoslovnom oznakom jezika na kojemu je izvorni tekst objavljen. Oznaka jezika odvojena je od referencijske oznake dokumenta jednim razmakom i stavlja se na kraj, a kod navođenja više jezika oznake se razdvajaju zarezom (npr. HRN EN ISO 472:2007 hr, en, fr, de).



Jedna od zanimljivosti vezana uz norme u Hrvatskoj je da ne postoji hrvatska **norma za zapis datuma i vremena**, bilo u računalu, bilo u općem jeziku ili poslovnim dokumentima, već samo pravopisna pravila, koja su se tijekom vremena već mnogo puta mijenjala. No, zato postoji međunarodna norma **ISO 8601-1:2019**⁹⁷, koja definira niz oblika zapisa datuma, vremena i vremenskih intervala. Tako je primjerice jedan od najčešćih zapisa datuma u obliku „**yyyy-mm-dd**“, vremena u obliku „**hh:mm:ss**“ i intervala u obliku „**yyyy-mm-ddThh:mm:ss**“. Navedeni se oblici zapisa mogu jednostavno sortirati i uspoređivati, a norma ISO 8601 široko je prihvaćena na međunarodnoj razini, iako nije preuzeta kao službena hrvatska norma.

⁹⁷ ISO 8601-1:2019 (en), Date and time — Representations for information interchange — Part 1: Basic rules



Zbog interoperabilnosti se za zapis datuma i vremena preporučuje koristiti međunarodna norma ISO 8601, odnosno zapise datuma, vremena i vremenskih intervala u oblicima „yyyy-mm-dd“, „hh:mm:ss“, „yyyy-mm-ddThh:mm:ss“. No, za zapis prilagođen krajnjim korisnicima u Hrvatskoj prvenstveno se preporučuje pridržavanje pravopisnih pravila.

2.4.2 Međunarodna organizacija za normizaciju ISO

Međunarodna organizacija za normizaciju ISO⁹⁸ (*International Organization for Standardization*) je međunarodna nevladina normirna organizacija, osnovana 1947. godine sa sjedištem u Ženevi. U organizaciji ISO-a uspostavljen je velik broj tehničkih odbora TC (engl. *Technical Committee*) koji su podijeljeni na pododbore SC (engl. *Subcommittee*) i, po potrebi, na radne skupine WG (engl. *Working Group*). ISO u velikoj mjeri surađuje s povjerenstvom IEC i zajedno osnivaju zajedničke tehničke odbore JTC (engl. *Joint Technical Committee*). Članovi ISO-a su nacionalna normirna tijela iz 164 države, a ima više od 249 tehničkih odbora, 504 pododbora i 2714 radnih skupina, te je dosad objavio više od 22.467 normi⁹⁹. Njezini su članovi službena korisnička tijela priznata od njihovih nacionalnih normirnih tijela. Bivši DZNM je bio član ISO-a od 1993. godine, a HZN je postao punopravni član (engl. *member body*) 1. srpnja 2005. godine.

ISO izrađuje i objavljuje niz različitih vrsta dokumenata pa tako razlikujemo:

- međunarodne norme (engl. *International Standard*) označene kao „ISO nnnnn:gggg Naziv“ s mogućnošću korištenja prefiksa IS ispred broja norme,
- tehničke izvještaje (engl. *Technical Report*) označene kao „ISO TR nnnnn:gggg Naziv“,
- tehničke specifikacije (engl. *Technical Specification*) i
- druge dokumente,

gdje, u skladu s referencijskom oznakom, „nnnn“ označava broj norme, „gggg“ godinu izdanja, a „Naziv“ naziv norme.



Ako je norma izdana u suradnji s drugom organizacijom, onda se pišu obje kratice naziva organizacija, pa se tako norme koje zajednički izrađuju ISO i IEC označavaju kao ISO/IEC.

Na informacijskom području i području interoperabilnosti najvažniji je rad sljedećih odbora:

- **JTC 1** – Informacijske tehnologije,
- **TC 46** – Informacije i dokumentacije,
- **TC 154** – Procesi, podatkovni elementi i dokumenti u trgovini, industriji i administraciji,
- **TC 171** – Aplikacije za upravljanje dokumentima.

⁹⁸ International Organization for Standardization, ISO, <http://www.iso.org>

⁹⁹ ISO in figures, <https://www.iso.org/iso-in-figures.html>, dohvaćeno na dan 31.01.2010.

2.4.3 Međunarodno elektrotehničko povjerenstvo IEC

Međunarodno elektrotehničko povjerenstvo IEC¹⁰⁰ (*International Electrotechnical Commission*, odnosno *Commission électrotechnique internationale*) je međunarodna neprofitna nevladina normirna organizacija, osnovana 1906. godine, a danas sa sjedištem u Ženevi. IEC izdaje međunarodne norme iz područja elektrotehnike, elektronike i srodnih tehnologija.

Članovi IEC-a su nacionalni elektrotehnički odbori iz 88 država¹⁰¹, ima više od 208 tehničkih odbora/pododbora, 684 radne skupine, 210 projektnih skupina i 646 skupina za održavanje, a u radu IEC-a sudjeluje više od 35.000 stručnjaka. Do 2018. godine IEC je objavio više od 10.000 publikacija. Referencijska oznaka normi IEC-a je u obliku „IEC nnnn:gggg Naziv“. Bivši DZNM je bio član od 1993. godine, a HZN je član (engl. *full member*) od 1. srpnja 2005. godine.

2.4.4 Međunarodna telekomunikacijska unija ITU

Međunarodna telekomunikacijska unija ITU¹⁰² (*International Telecommunication Union*) je vodeća agencija Ujedinjenih naroda za pitanja informacijske i komunikacijske tehnologije. Osnovana je 1865. godine u Parizu, a danas ima sjedište u Ženevi sa 193 države članice te više od 700 sektorskih i pridruženih članova.

ITU koordinira globalno dijeljeno korištenje radiofrekvencijskog spektra, promovira međunarodnu suradnju odobravanja satelitskih orbita, potiče nadogradnju telekomunikacijske infrastrukture, naročito u zemljama u razvoju, te pomaže u razvoju i koordinaciji različitih tehničkih normi vezanih uz informacijske i komunikacijske tehnologije. Na razini vlada i privatnog sektora središnje je mjesto dogovaranja i odlučivanja o pitanjima razvoja mreža i usluga.

ITU-om upravlja Glavno tajništvo (*General Secretariat*), a postoje tri glavna sektora:

- **Radiokomunikacijski sektor ITU-R** (ranije poznat kao CCIR, *International Radio Consultative Committee*),
 - upravlja međunarodnim radiofrekvencijskim spektrom i resursima vezanim uz satelitske orbite,
 - objavljuje međunarodne tehničke norme u području učinkovite uporabe radiofrekvencijskog spektra za sve radiokomunikacijske sustave,
 - objavljuje Preporuke (engl. *Recommendations*) koje predstavljaju norme, a koje odobravaju države članice, dok njihova primjena nije obvezna,
- **Sektor za normizaciju u telekomunikacijama ITU-T** (ranije poznat kao CCITT, *International Telephone and Telegraph Consultative Committee*, odnosno *Comité consultatif international téléphonique et télégraphique*),
 - objavljuje različite tehničke norme vezane uz informacijske i komunikacijske tehnologije, najviše u području rada telekomunikacijskih mreža,
 - objavio više od 3000 norma u obliku Preporuka koje su u primjeni, ali nisu obvezne,
- **Sektor za razvoj telekomunikacija ITU-D,**

¹⁰⁰ International Electrotechnical Commission, IEC, <http://www.iec.ch>

¹⁰¹ IEC Facts & figures, <https://www.iec.ch/about/activities/facts.htm>, dohvaćeno na dan 31.01.2010.

¹⁰² International Telecommunication Union, ITU, <http://www.itu.int>

- osnovan da pomogne u širenju pravednog, održivog i pristupačnog pristupa informacijsko-komunikacijskim tehnologijama,
- glavna područja rada su sigurna i pouzdana uporaba informacijsko-komunikacijske tehnologije, primjena telekomunikacija u izvanrednim situacijama te povećanje stupnja komunikacijske povezanosti širom svijeta.

2.4.5 Europski odbor za normizaciju CEN

Europski odbor za normizaciju CEN¹⁰³ (*European Committee for Standardization*, odnosno *Comité Européen de Normalisation*) je neprofitna organizacija osnovana 1961. godine sa sjedištem u Bruxellesu, s misijom poticanja europskog gospodarstva u globalnoj trgovini, na dobrobit europskih građana i okoliša, pružajući učinkovitu infrastrukturu zainteresiranim stranama za razvoj, održavanje i distribuciju koherentnog skupa normi i specifikacija. U ovom obliku posluje od 1975. godine, a članovi su 34 nacionalna normirna tijela država članica EU-a i EFTA-e. Ima više od 320 tehničkih odbora, a do 2020. godine objavio je više od 17300 normi.

Nacionalna normirna tijela država za koje su prihvaćene ili su mogući kandidati za pridruživanje Europskoj uniji (EU) ili Europskom udruženju slobodne trgovine (EFTA) i država koje su tehnički, znanstveno, ekonomski, politički i društveno usko povezane s EU-om i EFTA-om imaju status pridruženog člana. Organizacije koje zastupaju posebne društvene ili gospodarske interese na europskoj razini, čiji su statuti ozakonjeni europskim zakonima ili nacionalnim zakonima pojedine države kao nacionalnog člana CEN-a, mogu biti članice CEN-a u statusu suradnika. Bivši DZNM je bio pridruženi član od 1995. godine, a HZN je pridruženi član od 1. srpnja 2005. godine.

CEN donosi norme na europskoj razini, no one nisu obvezne dok ih zakonodavac određene zemlje ne propiše, što se trenutno odnosi na članice CEN-a. Norme koje izdaje CEN podupiru poslovanje na europskoj razini. Norme se imenuju kao EN (engl. *European Norm*), no važne su i tehničke specifikacije TS (engl. *Technical Specification*), tehnički izvještaji TR (engl. *Technical Report*) i dokumenti usvojeni na radionicama CEN-a tzv. CWA (engl. *CEN Workshop Agreement*).

CEN je službeno normirno tijelo Europske unije, prepoznato i priznato kao predstavnik normizacije za sve sektore osim elektrotehnike i telekomunikacija, gdje su nadležni CENELEC i ETSI. CEN blisko surađuje s organizacijama CENELEC, ETSI i ISO, a jedan od primjera je CEN-CENELEC Forum koji je zadužen za područje informacijsko-komunikacijskih tehnologija.

2.4.6 Europski odbor za elektrotehničku normizaciju CENELEC

Europski odbor za elektrotehničku normizaciju CENELEC¹⁰⁴ (*European Committee for Electrotechnical Standardization*, odnosno *Comité Européen de Normalisation Electrotechnique* ili *Europäisches Komitee für elektrotechnische Normung*) nastao je 1973. godine spajanjem CENEL-a i CENELCOM-a, a danas ima sjedište u Bruxellesu. CENELEC je zadužen je za europsku normizaciju u području elektrotehnike.

¹⁰³ European Committee for Standardization, CEN, <http://www.cen.eu>

¹⁰⁴ European Committee for Electrotechnical Standardization, CENELEC, <http://www.cenelec.eu>

Članovi su nacionalni elektrotehnički odbori država članica EU-a i EFTA-e te broji 34 člana. CENELEC je krajem 2012. godine imao više od 77 tehničkih odbora te 277 pododbora i radnih skupina. Do kraja 2019. godine objavio je 7305 normi. Nacionalna tijela, odnosno nacionalni odbori država u susjedstvu EU-a imaju status pridruženog člana u CENELEC-u. Bivši DZNM je bio pridruženi član od 1995. godine, a HZN je pridruženi član od 1. srpnja 2005. godine.

2.4.7 Europski institut za telekomunikacijske norme ETSI

Europski institut za telekomunikacijske norme ETSI¹⁰⁵ (*European Telecommunications Standards Institute*) je nezavisna neprofitna normizacijska organizacija u sektoru telekomunikacija, osnovana 1988. godine na inicijativu Europske konferencije poštanskih i telekomunikacijskih uprava (CEPT). Krajem 2019. godine ETSI je imao više od 850 članova iz 65 zemalja. ETSI ima oko 35 tehničkih tijela (TC, EP, EPP, SC), a do kraja prošlog desetljeća objavio je preko 4.500 europskih norma. Bivši DZNM je bio je član od 1994. godine, a HZN ima status nacionalnoga normirnog tijela (NSO, engl. *National Standards Organization*) od 1. srpnja 2005. godine.

2.4.8 Američki nacionalni normirni institut ANSI

Od velikog skupa nacionalnih normizacijskih organizacija ovdje ćemo izdvojiti samo **Američki nacionalni normirni institut ANSI¹⁰⁶** (*American National Standards Institute*), koja je privatna neprofitna organizacija koja nadgleda razvoj normi u Sjedinjenim Američkim Državama. Budući da smo već dulji niz godina svjedoci opće globalizacije, norme koje donosi ANSI kontinuirano se usklađuju s drugim međunarodnim normama, ali je također primjetno da se određene norme ne mogu jednostavno uskladiti s primjerice europskim normama, najviše zbog razlika u mjernim sustavima te praksi korištenja određenih tehnologija. Danas ANSI ima više od 125.000 članova organizacije i 3.500.000 pojedinačnih članova.

Ova organizacija ima dugu tradiciju jer je u SAD-u još 1918. godine osnovan AESC (*American Engineering Standards Committee*), kao krovno tijelo postojećih pet inženjerskih udruga, koji 1928. godine mijenja naziv u ASA (*American Standards Association*), da bi 1966. godine nastao USASI (*United States of America Standards Institute*), koji tek 1969. godine dobiva današnji naziv ANSI.

2.4.9 Institut inženjera elektrotehnike i elektronike IEEE

Institut inženjera elektrotehnike i elektronike IEEE¹⁰⁷ (*Institute of Electrical and Electronics Engineers*, čiji se naziv čita kao „aj-tripl-i“) je međunarodna neprofitna stručna udruga za unaprjeđenje tehnologija vezanih uz elektrotehniku i elektroniku. Ima više od 422.000 članova u više od 160 zemalja, od čega skoro polovicu u SAD-u. U udruzi IEEE postoji više od 339 sekcija u deset svjetskih regija, oko 2.430 stručnih odjela koji na lokalnim razinama objedinjuju članove sličnih tehničkih interesa, više od 3.284 studentskih ogranka na fakultetima i sveučilištima u stotinjak zemalja, 39 društava i 5 tehničkih odbora koji zastupaju široki raspon

¹⁰⁵ European Telecommunications Standards Institute ETSI, <https://www.etsi.org/>

¹⁰⁶ American National Standards Institute ANSI, <https://www.ansi.org/>

¹⁰⁷ Institute of Electrical and Electronics Engineers, IEEE, <http://www.ieee.org>

tehničkih interesa. Osim toga IEEE izdaje više od 200 zbornika konferencija, časopisa i magazina, pokrovitelj je više od 1.900 konferencija po cijelom svijetu godišnje te razvija i održava oko 1.250 važećih normi i više od 700 normi u razvoju. Sve ove brojke govore kako je IEEE vodeći autoritet na širokom tehničkom području od računalnih znanosti, biomedicinske tehnike i telekomunikacija, preko električne energije, do potrošačke elektronike i mnogih drugih područja. U ovom se trenutku IEEE smatra najvećom svjetskom stručnom udrugom.



Udruga IEEE osnovana je još 1884. godine u obliku AIEE (*American Institute of Electrical Engineers*) idejom nekolicine znanstvenika s ciljem praćenja razvoja elektrotehnike, a jedan od prvih predsjednika bio je Alexander Graham Bell. Kasnije joj je pridružena i organizacije IRE (*Institute of Radio Engineers*).

Danas IEEE nastoji poticati, organizirati i pomagati raznolike tehničke aktivnosti širom svijeta, a glavni cilj je unaprijediti teorijska i praktična znanja u području elektrotehnike i računarstva. Osim toga IEEE je pokrovitelj znatnog broja stručnih i znanstvenih skupova i aktivnosti širom svijeta te objavljuje više od četvrtine svih publikacija vezanih za elektrotehniku i računarstvo. Ugrubo se može procijeniti da IEEE putem svojih znanstvenih i stručnih publikacija, skupova i normi objavljuje oko 30 posto ukupne svjetske literature na području elektrotehnike i računarstva te organizira više od 300 velikih stručnih skupova godišnje iz istih područja. U Hrvatskoj je IEEE prisutan je od 1971. godine, a Hrvatska sekcija IEEE¹⁰⁸ samostalno djeluje od 1992. godine.

2.4.10 Konzorcij World Wide Web W3C

Konzorcij World Wide Web¹⁰⁹ ili **W3C** (*World Wide Web Consortium*) je međunarodna udruga koja potiče razvoj Web-a i donosi norme vezane uz Web. Nema službeno sjedište, ali ima dijelove na MIT-u (*Massachusetts Institute of Technology*) u Cambridgu u SAD-u, zatim u ERCIM-u (*European Research Consortium for Informatics and Mathematics*) u Sophia-Antopolisu u Francuskoj i Sveučilištu Keio blizu Tokija u Japanu. W3C je osnovao **Tim Berners-Lee**, poznatiji i kao „izumitelj Web-a“, a i danas ga vodi uz predsjednika konzorcija Jeffreya Jaffu. Početkom 2020. godine W3C je imao 428 članova.

Misija konzorcija W3C je vođenje evolucije Web-a razvojem protokola i smjernica koje omogućavaju dugotrajni rast Web-a. Načela na kojima se zasniva rad W3C-a uključuju krilaticu „Web za sve“ (engl. *Web for All*) i „Web na svemu“ (engl. *Web on Everything*) koje potiču razvoj alternativnih pristupa Webu, internacionalizaciju, mobilni Web, Web na raznorodnim uređajima, razvoj preglednika i mnoge druge. Vizija Web-a prema W3C-u uključuju sudjelovanje i „bogatu interakciju“ (engl. *Rich Interaction*) svih sudionika na Webu kroz oblikovanje stranica Web-a i aplikacije Web-a, arhitekturu Web-a, podatke u uslugama na Webu kroz osnovne tehnologije temeljene na jeziku XML, semantički Web, usluge Web-a te povjerenje (engl. *trust*) na Webu izraženo kroz sigurnost usluga i privatnost.

¹⁰⁸ IEEE, Hrvatska sekcija, <http://www.ieee.hr>

¹⁰⁹ World Wide Web Consortium, W3C, <http://www.w3.org>

Primarna aktivnost W3C-a je razvoj protokola i smjernica te izrada normi koje definiraju ključne pojmove Web-a. W3C je u suradnji s IETF-om objavio niz normi spomenutih u ovom priručniku, kao što su:

- CSS (Cascading Style Sheets),
- CGI (*Common Gateway Interface*),
- DOM (*Document Object Model*),
- HTML (*HyperText Markup Language*),
- OWL (*Web Ontology Language*),
- RDF (*Resource Description Framework*),
- SVG (*Scalable Vector Graphics*),
- SOAP (*Simple Objects Access Protocol*),
- SMIL (*Synchronized Multimedia Integration Language*),
- VoiceXML,
- WSDL (*Web Services Description Language*),
- XHTML (*Extensible Hypertext Markup Language*),
- XML (*Extensible Markup Language*),
- XML Schema ili XSD,
- XPath (*XML Path Language*),
- XQuery,
- XSLT (*Extensible Stylesheet Language Transformations*),
- i mnoge druge.

2.4.11 Međunarodno radno tijelo za inženjering Interneta IETF

Međunarodno radno tijelo za inženjering Interneta IETF¹¹⁰ (*Internet Engineering Task Force*) je velika međunarodna zajednica istraživača, dizajnera mreža, operatera i proizvođača koji surađuju na razvoju Interneta. Osnovna misija i cilj IETF-a je da „Internet radi bolje“. Iako na prvi pogled IETF može izgledati kaotično, on je dobro organiziran, ali na načelu labave povezanosti, i ima više od 100 radnih skupina.

Upravljačka skupina za inženjering Interneta IESG (*Internet Engineering Steering Group*) je tijelo IETF-a zaduženo za upravljanje aktivnostima IETF-a, a time i radnih skupina, procesima donošenja normi te davanje smjernica članovima. IETF i IESG odgovorni su za odobravanje normi interneta u obliku dokumenata RFC.

Pod donekle čudnovatim nazivom „zahtjev za komentarom“ ili **RFC¹¹¹** (engl. *Request for Comments*) razvio se sustav objavljivanja normi vezanih uz Internet. U početku su RFC-ovi stvarno i bili zahtjevi za komentarima oko rješavanja arhitekturnih problema Interneta. No, RFC danas može biti opis neke komunikacije, novog koncepta, informacije, sažetka neke tehnologije, preporuke, a katkad i računalnog humora¹¹² vezanog uz Internet u obliku memoranduma koji objavljuje IETF. I neke druge organizacije objavljuju dokumente pod nazivom RFC, ali se najčešće ipak misli na IETF. Sadržaj RFC-a su često rezultati istraživanja, opisi metoda i postupaka, protokoli, jezici, algoritmi, preporuke korištenja i inovativne

¹¹⁰ Internet Engineering Task Force (IETF), <http://www.ietf.org>

¹¹¹ Request for Comments (RFC), <https://www.ietf.org/standards/rfcs/>

¹¹² Tradicionalno izdavanje RFC za 1. travnja od 1973. godine do danas, tzv. April Fools' Day RFC

tehnologije. Neke RFC-ove IETF, odnosno njegovo tijelo IESG, prihvaća kao Internetske norme (engl. *Internet Standard*), koje tada dobivaju kraticu STD, ali nisu svi dokumenti RFC norme. Osim toga postoje sljedeći statusi dokumenta: informacijski (engl. *Informational*), eksperimentalni (engl. *Experimental*), najbolja praksa (BCP, engl. *Best Current Practice*), povijesni (engl. *Historic*) i norma (engl. *Standards Track*), koji se dodatno dijeli na predložene norme (engl. *Proposed Standard*), inicijalni prijedlog prve inačice norme (engl. *Draft Standard*) i norme Interneta (engl. *Internet Standard*).

Svaki dokument RFC izrađuje pojedini autor ili skupina autora, koji su navedeni na vrhu memoranduma. Za pisanje dokumenata RFC postoje određena pravila koja određuju urednici, a ne postoje modifikacije i inačice dokumenta, već se može samo proglasiti nevažećim. Kao što se uočava, RFC ne prati klasični sustav izrade norme kroz konzorcije i institucije, već pojedinci mogu izravno doprinijeti sa svojim dokumentima koji prolaze proces revizije i javne dorade predloška prve inačice (engl. *draft*), a zatim ih IETF odobrava. Čitav proces i sam je opisan u RFC 2026¹¹³.

2.4.12 Autoritet za brojeve pridružene Internetu IANA

Autoritet za brojeve pridružene Internetu IANA¹¹⁴ (*Internet Assigned Numbers Authority*) je tijelo odgovorno za koordinaciju ključnih elemenata koji omogućavaju rad Interneta. To se ponajprije odnosi na jedinstvene nazive, kodove i sustave numeriranja koji se koriste od raznih tehničkih normi i protokola Interneta. Načelno se aktivnosti organizacije IANA mogu podijeliti na upravljanje korijenskim DNS-om, domenama .int i .arpa te drugim problematikama domenskih naziva, zatim koordinaciju globalnog skupa adresa IP te informiranje regionalnih registara Interneta i sustavima numeriranja elemenata protokola Interneta u skladu s drugim normama. Organizacijom IANA operira organizacija za pridružene nazive i brojeve Interneta ICANN (*Internet Corporation for Assigned Names and Numbers*). IANA je zadužena za globalne jedinstvene nazive i brojeve koji se koriste u protokolima Interneta koji se koriste u dokumentima RFC.

2.4.13 Centar Ujedinjenih naroda za omogućavanje trgovanja i elektroničkog poslovanja UN/CEFACT

Centar Ujedinjenih naroda za omogućavanje trgovanja i elektroničkog poslovanja UN/CEFACT¹¹⁵ (*United Nations Centre for Trade Facilitation and Electronic Business*) dio je Ekonomske komisije Ujedinjenih naroda za Europu UNECE (*United Nations Economic Commission for Europe*), koja unaprjeđuje poslovanje i trgovanje organizacija te učinkovitu razmjenu proizvoda i usluga. U kontekstu interoperabilnosti UN/CEFACT je važan kao organizacija visokog ugleda za normiranje elektroničkog poslovanja koja je donijela normu EDIFACT, koju je kasnije usvojila i organizacija ISO. Trenutno je UN/CEFACT važan u kontekstu široko prihvaćene norme CC (*Core Components*).

¹¹³ Bradner, S., The Internet Standards Process - Revision 3, RFC 2026, <http://tools.ietf.org/html/rfc2026>

¹¹⁴ IANA Internet Assigned Numbers Authority, ICANN, <http://www.iana.org>

¹¹⁵ United Nations Centre for Trade Facilitation and Electronic Business UN/CEFACT, <http://www.unece.org/cefact>

UN/CEFACT je inače poznat po vrlo detaljnom, ali i sporom postupku donošenja norma u četiri koraka:

- specifikacija poslovnih zahtjeva (BRS, engl. *Business Requirements Specification*),
- mapiranje specifikacije zahtjeva (RSM, engl. *Requirements Specification Mapping*),
- osnovne komponente / osnovni informacijski entiteti CC/BIE (engl. *Core Components/Basic Information Entity*),
- stvaranje XML shema.

2.4.14 Organizacija za unapređivanje strukturiranih informacijskih normi OASIS

Organizacija za unapređivanje strukturiranih informacijskih normi OASIS¹¹⁶ (*Organization for the Advancement of Structured Information Standards*) je neprofitni konzorcij koji pokreće razvoj, konvergenciju i prihvaćanje otvorenih normi za globalno informacijsko društvo. Ovo se osobito odnosi na norme usluga Web-a, elektroničkog poslovanja, sigurnosti i interoperabilnosti. OASIS je osnovan 1993. godine pod nazivom *SGML Open*, a kasnije je promijenio ime u skladu sa svojim djelovanjem, uključujući norme vezane uz jezik XML. Danas ima više od 5.000 sudionika, koji predstavljaju više od 600 organizacija i pojedinaca iz više od 100 zemalja. Neke od najpoznatijih normi vezane uz OASIS su norme DocBook, ebXML, OpenDocument, UDDI i UBL kao i niz normi vezanih uz tehnologije i jezike Web Services, od kojih je većina opisana u drugim poglavljima ovog priručnika.

2.4.15 Međunarodna organizacija GS1

Organizacija **GS1**¹¹⁷ (*Global Standards One*) je međunarodna neprofitna, poslovno neutralna organizacija, usmjerena na razvoj i unapređenje globalnih normi, tehnologija i poslovnih rješenja kojima je cilj unapređenje učinkovitosti i preglednosti lanaca opskrbe i potražnje, globalno u svim sektorima. Organizacija GS1 nastala je 2004. godine spajanjem organizacija **EAN International** (*European Article Number International*) i **UCC** (*Uniform Code Council*), a okuplja više od 110 lokalnih organizacija i prisutna je sa svojim normama u više od 150 država, te ima 1,5 milijuna tvrtki korisnika. Sustav normi koje pruža GS1 podržava više od milijun tvrtki u cijelom svijetu u više od 25 industrijskih sektora, pri čemu se izdvajaju maloprodaja robe široke potrošnje, zdravstvo te transport i logistika.

Istim nazivom se naziva i potpuno integrirani **sustav GS1** koji sadržava:

- **BarCodes** – normu crtičnih kodova (engl. *bar codes*) za automatsku identifikaciju roba (nekadašnjim *EAN.UCC*), koja koristi GS1 identifikacijske ključeve (engl. *Identification Keys*) za jedinstvenu identifikaciju proizvoda, lokacija, usluga i sl.,
- **eCom** – norme elektroničkog poslovanja porukama i XML alati za elektroničku razmjenu podataka i elektroničko poslovanje,
- **GDSN** – globalnu sinkronizacijsku mrežu i norme za sigurnu i pouzdanu razmjenu točnih normiranih podataka,

¹¹⁶ Organization for the Advancement of Structured Information Standards OASIS, <https://www.oasis-open.org/>

¹¹⁷ GS1, <http://www.gs1.org>

- **EPCglobal** – globalnu normu koja kombinira tehnologiju RFID i jedinstvene kodove EPC (engl. *Electronic Product Code*) za trenutnu i automatsku identifikaciju i praćenje u opskrbnom lancu.

Sustav GS1 ja najrasprostranjeniji sustav upravljanja opskrbnim lancem na svijetu, a najviše je korišten u maloprodaji.

Glavni **identifikacijski ključevi** uključuju:

- **globalni broj trgovinskog artikla GTIN** (engl. Global Trade Item Number),
- **globalni broj lokacije GLN** (engl. Global Location Number),
- **serijski broj spremnika isporuke SSCC** (engl. Serial Shipping Container Code)
- i mnoge druge.

Ključevi se mogu prikazivati na različitim medijima, a najpoznatiji su različiti **crtični kodovi** (engl. *bar codes*) kao što su EAN-13, UPC-A, EAN-8, UPC-E, ITF-14, GS1-128, skupina kodova GS1 DataBar, dvodimenzionalni crtični kod GS1 DataMatrix i GS1 QR Code, te **oznake EPC/RFID** (*Electronic Product Code/Radio Frequency Identification*), koje koriste tehnologiju identifikacije radijskim frekvencijama i EPC TDS (*EPC Tag Data Standard*).

Sljedeća slika prikazuje 16-eroznamenasti broj GTIN u obliku EAN-13 i njegove osnovne dijelove: prefiks (a), broj tvrtke (b), referencu artikla (c) i kontrolnu znamenku (d).



Slika 2.2: Globalni broj trgovinskog artikla GTIN (engl. Global Trade Item Number)

Norme za elektroničko poslovanje i razmjenu poruka poznatije pod zajedničkim nazivom **GS1 EDI** (*Electronic Data Interchange*) imaju nekoliko dijelova:

- **GS1 EANCOM** je nešto starija komunikacijska norma temeljena na normi **UN/EDIFACT** (United Nations Electronic Data Interchange for Administration, Commerce and Transport)
- **GS1 XML** je nešto novija norma temeljena na jeziku XML i elektroničkim poslovnim porukama
- **GS1 UN/CEFACT XML profile**
- **GS1 UBL** koji je i dijelom novije strategije **GS1 EDI Strategy 2018-2020**

GS1 EANCOM i GS1 XML se i dalje paralelno razvijaju, ali nisu potpuno kompatibilne. Stariju normu EANCOM polako istiskuje novija norma GS1 XML koja ima znatno bržu stopu prihvaćanja, a koju potanje opisujemo kasnije.

Na temelju Ugovora o licenciji i potpisanog Kodeksa poslovnog ponašanja s međunarodnom GS1 organizacijom, u Hrvatskoj djeluje neprofitna udruga tvrtki korisnika **GS1 Croatia**¹¹⁸,

¹¹⁸ GS1 Croatia, <http://www.gs1hr.org>

Hrvatsko udruženje za automatsku identifikaciju, elektroničku razmjenu podataka i upravljanje poslovnim procesima.

2.5 Norme za razmjenu elektroničkih podataka

Prema potrebama elektroničkog poslovanja, kao posljedice, odnosno evolucije procesa učinkovite razmjene elektroničkih podataka, norme se mogu promatrati višerazinski¹¹⁹.

Na razini međupovezivanja najviše su korišteni protokoli i prateće norme na aplikacijskoj razini, a pogotovo vezani uz otvorene norme. Tako su najviše u uporabi sljedeći općepoznati protokoli, kao što su protokol **HTTP** (*HyperText Transfer Protocol*), protokol **SOAP** (*Simple Object Access Protocol*), te drugi protokoli često korišteni na internetu, primjerice **FTP** (*File Transfer Protocol*), **SMTP** (*Simple Mail Transfer Protocol*) i **MIME** (*Multipurpose Internet Mail Extensions*).

Na razini razmjene podataka koriste se poznate *de facto* i *de iure* norme za definiranje oblika i strukture dokumenata, odnosno poruka. Prije svega to je jezik **XML** i **XML shema** potanje opisani kasnije, **prostori naziva XML-a**, jezici za manipulaciju podacima, kao što je **XSLT**, zatim **norme za prikaz znakova** te uniformni identifikatori i lokatori resursa **URI** i **URL**.

Na razini integracije sustava praktički je riječ o razmjeni informacija između aplikacija u otvorenoj okolini. Načelno postoji nekoliko smjerova uporabe i razvoja normi razmjene podataka, koji su obrađeni u sljedećim potpoglavljima, a to su ponajprije **EDI** i **Web Services**.

Smjernice EU-a upućuju na to da je glavni smjer kod elektroničke razmjene podataka odabir otvorenih norma, ponajprije zasnovanih na jeziku XML i tehnologijama Web Services, pri čemu se polagano napušta postojeća dugogodišnja raširena praksa, kao što je uporaba norme EDIFACT. I u Hrvatskoj se preporučuje korištenje istih norma, tehnologija Web Services, jezika XML i XML shema te normi Unicode za zapis znakova.

2.5.1 Elektronička razmjena podataka EDI

Prvi smjer uključuje **elektroničku razmjenu podataka EDI**¹²⁰ (*Electronic Data Interchange*) s definiranim strukturama podataka koje se između organizacija razmjenjuju elektroničkim putem. EDI nije uopće nova tehnologija, već je zasnovana na načelima razmjene podataka utemeljenim pred pedesetak godina.

EDI je definiran kao računalna razmjena striktno oblikovanih poruka koji predstavljaju dokument ili druge novčane instrumente. No, EDI nužno ne uvjetuje razmjenu podataka elektroničkim mrežama, već je moguć i prijenos bilo kakvim elektroničkim medijem. Podrazumijeva se da se razmjena, odnosno prihvrat poruke može dogoditi isključivo računalno, bez uplitanja ljudskog operatera, pa zato ni podaci u porukama nisu izvedeni kako bi bili jednostavno razumljivi ljudima, već samo računalima. S obzirom na to da EDI ne definira način prijenosa poruke, moguće je koristiti različite načine i protokole, uključujući protokole vezane

¹¹⁹ Studija normizacije u e-Poslovanju, Isporuka 1, ver. 2.4, Fakultet elektrotehnike i računarstva, Sveučilište u Zagrebu, Zagreb, 2009.

¹²⁰ Electronic Data interchange, EDI, FIPS PUB 161-2, NIST, 1996-04-29, <http://www.itl.nist.gov/fipspubs/fip161-2.htm>

uz elektroničke poruke, SMTP, MIME, zatim protokole FTP i HTTP vezane uz Internet i druge mreže s dodanom vrijednošću VAN (engl. *Value-added Network*). Neki od ovih postupaka vezanih uz elektroničku poštu i protokol HTTP opisani su u RFC 3335¹²¹ i RFC 4130¹²².

Postoje **četiri glavne skupine normi EDI** koje su se pojavile 80-ih godina prošlog stoljeća:

- **UN/EDIFACT** – međunarodna norma UN-a, najviše globalno rasprostranjena, osim u SAD-u,
- **ANSI ASC X12** – norma razvijena od ANSI-a, raširena u Sjevernoj Americi,
- **TRADACOMS** – zastarjela norma razvijena od ANA-e, još dijelom postoji u Velikoj Britaniji,
- **ODETTE**¹²³ – norma koja se koristi u europskoj automobilskoj industriji.

Norma **UN/EDIFACT**¹²⁴ (United Nations/Electronic Data Interchange For Administration, Commerce and Transport) prepoznata je kao međunarodna norma pod zajedničkim nazivom serije **ISO 9735-10:2014 Electronic data interchange for administration, commerce and transport (EDIFACT)**, a objedinjuje deset pojedinačnih normi.

U svojoj osnovi norma EDIFACT sadržava:

- skup sintaksnih pravila za oblikovanje podatkovnih struktura,
- protokol interaktivne razmjene I-EDI i
- standardne poruke, koji omogućavaju razmjenu podataka.

Primjer jedne poruke EDIFACT¹²⁵ je sljedeći:

```
UNA:+.? ' UNB+UNOB:4+STYLUSSTUDIO:1+DATADIRECT:1+20051107:1159+6002 '
UNH+SSDD1+ORDERS:D:03B:UN:EAN008 '
BGM+220+BKOD99+9 ' DTM+137:20051107:102 ' NAD+BY+5412345000176::9 '
NAD+SU+4012345000094::9 ' LIN+1+1+0764569104:IB ' QTY+1:25 '
FTX+AFM+1++XPath 2.0 Programmer?'s Reference' LIN+2+1+0764569090:IB '
QTY+1:25 '
FTX+AFM+1++XSLT 2.0 Programmer?'s Reference' LIN+3+1+1861004656:IB '
QTY+1:16 '
FTX+AFM+1++Java Server Programming '
LIN+4+1+0596006756:IB ' QTY+1:10 '
FTX+AFM+1++Enterprise Service Bus '
UNS+S ' CNT+2:4 ' UNT+22+SSDD1 ' UNZ+1+6002 '
```



Iz ove kratke EDIFACT poruke vidljivo koliko je čovjeku teško iščitati smislene podatke pa je ovakav zapis praktički predviđen samo za računalnu obradu.

¹²¹ Harding, T., MIME-based Secure Peer-to-Peer Business Data Interchange over the Internet, RFC 3335, IETF, <http://tools.ietf.org/html/rfc3335>

¹²² Moberg, D. i Drummond, R., MIME-Based Secure Peer-to-Peer Business Data Interchange Using HTTP, Applicability Statement 2 (AS2), RFC 4130, IETF, <http://tools.ietf.org/html/rfc4130>

¹²³ ODETTE, <http://www.odette.org>

¹²⁴ UN/EDIFACT, <http://www.unece.org/trade/untdid/welcome.htm>

¹²⁵ Primjer preuzet s <http://www.stylusstudio.com/edi/edifact-sample.txt>

Sljedeća slika ilustrira strukturu poruke EDIFACT u skladu s normama UN/EDIFACT i ANSI ASC X12:

- **komunikacijska sjednica** – opcionalni dio, predstavlja cjelovitu komunikaciju ili sam dokument EDI,
- **omotnica razmjene** – nužni dio, segment koji omata samu transakciju,
- **funkcionalna grupa** – opcionalni dio, segment koji može omatati funkcionalnu cjelinu,
- **transakcijski skup** – nužni dio, omotač same transakcije,
- **podatkovni segment** – nužni dio, podaci.

	oznake UN/EDIFACT	oznake ASC X12
Komunikacijska sjednica (opcionalno)	UNA	
Omotnica razmjene (nužno)	UNB Interchange Header	ISA
Funkcionalna grupa (opcionalno)	UNG Functional Group Header	GS
Transakcijski skup (nužno)	UNH Message Header	ST
Podatkovni segment (nužno)		
	UNT Message Trailer	SE
	UNE Functional Group Trailer	GE
Funkcionalna grupa	UNG Functional Group Header	GS
Transakcijski skup (nužno)	UNH Message Header	ST
Podatkovni segment (nužno)		
	UNT Message Trailer	SE
Transakcijski skup (nužno)	UNH Message Header	ST
Podatkovni segment (nužno)		
	UNT Message Trailer	SE
	UNE Functional Group Trailer	GE
	UNZ Interchange Trailer	IEA

Slika 2.3: Struktura poruke EDIFACT

Kao što se može vidjeti norme UN/EDIFACT i ANSI ASC X12 su vrlo slične, ali među njima postoje određene razlike u ključnim riječima, terminatorima, indikatoru otpuštanja poruke, kompozitnim elementima, podršci za binarne podatke i podržanim sigurnosnim pravilima, pa se ne može izravno komunicirati između sustava s izvedenim različitim normama.

U Hrvatskoj EDI nažalost nikad nije bio u preširokoj upotrebi, ponajprije zato što je cjelokupna infrastruktura razvijena na temelju manjih i srednjih tvrtki, tzv. SME tržište (engl. *Small and Medium Enterprises*), pa ne postoje veliki informacijskih posrednici i EDI je većini bio preskup za korištenje. No, u slučaju zapadne Europe za očekivati je da jedan dio poduzeća sigurno i dalje koristi EDIFACT.

2.5.2 Tehnologije elektroničke razmjene podataka Web Services

Treći važan smjer normiranja razmjene podataka u elektroničkom obliku kreće se u smjeru normi **tehnologija Web Services**, konkretnije **SOAP**, **WSDL** i **UDDI**. Za pitanja sigurnosti koriste se norme **WS-Security** i **WS-Reliability/WS-Reliable-Messaging**, dok se za pitanja adresiranja koristi norma **WS-Addressing** koju je donio W3C, a koja omogućava transportno neovisni mehanizam adresiranja poruka i usluga kroz identifikaciju krajnjih točaka usluge. S obzirom na to da je uslugama Web i tehnologijama Web Services na različite načine posvećeno čak nekoliko poglavlja ovog priručnika, ovdje nećemo potanje razmatrati same tehnologije ni prateće norme.

2.5.3 Ostale norme elektroničke razmjene podataka

U razvoju tehnologija Web Services, utjecala je većim dijelom i norma **CORBA**¹²⁶ (*Common Object Request Broker Architecture*), koja omogućava interoperabilnost heterogenih programskih komponenti izvedenih u različitim programskim jezicima i na različitim platformama. Norma CORBA koristi jezik za definiciju sučelja IDL (*Interface Definition Language*), za koji postoje preslikavanja za praktički sve aktualne programske jezike. Korištenjem brokera zahtjeva objekta – ORB (*Object Request Broker*) zahtjevi se proslijeđuju lokalnim ili udaljenim objektima uporabom protokola IIOP (*Internet Inter-ORB Protocol*). Norma CORBA ima dosta implementacija, ali se često odustaje od primjene te norme zbog složenosti izvedbe i posljedične veće cijene.

2.6 Norme u elektroničkom poslovanju

Na svjetskoj razini postoji niz različitih normi koje se koriste u elektroničkom poslovanju, od kojih ćemo neke najpoznatije opisati. Kao i u drugim državama, u Hrvatskoj postoji niz pokušaja davanja preporuka vezanih uz elektroničko poslovanje. Pri odabiru preporuka treba obratiti pozornost na što veću vjerojatnost prihvatanja, odnosno da te preporuke budu prihvatljive glavnim sudionicima elektroničkog poslovanja u državi, kao što su banke, velike tvrtke, država i glavni isporučitelji programskih rješenja. Kao i u većini drugih slučajeva, jedan od većih izazova je pokušaj pomirbe otvorenih norma (engl. *open standards*) i vlasničkih norma (engl. *proprietary standards*). Na europskoj se razini primjećuje znatan pomak zahvaljujući angažmanu EU, koja osim što **promovira otvorenost i otvorene norme**, vrši pritisak na korištenje i utjecaj vlasničkih norma, najčešće u obliku kazni za netržišno ponašanje.

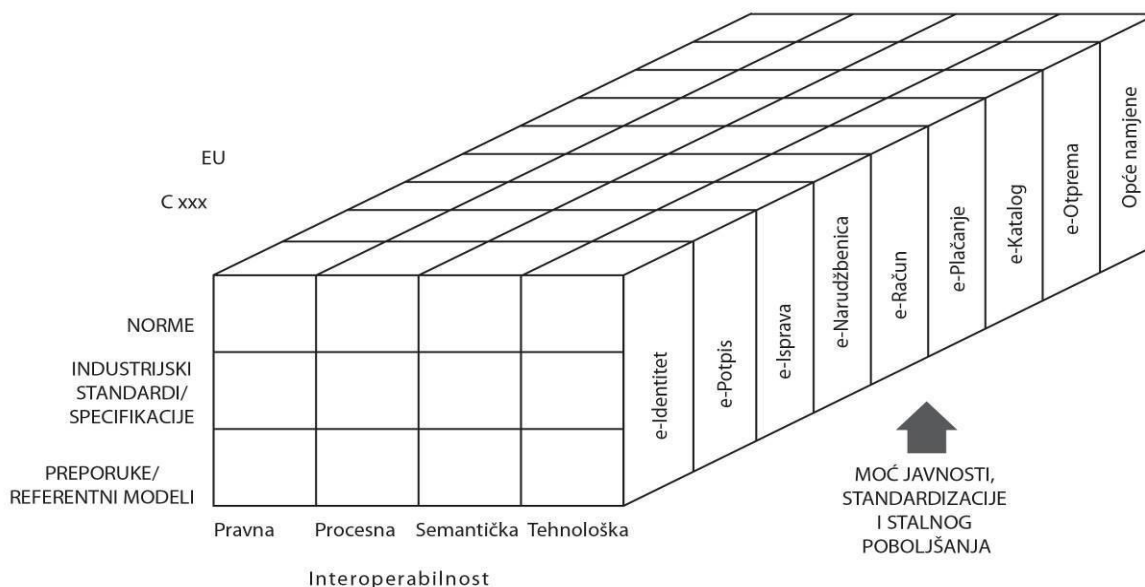
Uvođenje učinkovitog e-Poslovanja znači i rješavanje niza preduvjeta na nacionalnim razinama, kao što su e-Identitet, e-Potpis, e-Isprave i slično. Ovome u Hrvatskoj doprinosi i uvođenje osobnoga identifikacijskog broja OIB, koji je zamijenio jedinstveni matični broj građana JMBG iz niza razloga, kao što su nesklad sa Zakonom o zaštiti osobnih podataka¹²⁷ zbog govoreće šifre u broju, nekompatibilnost s europskom praksom, nemogućnošću iskorištavanja JMBG-a za potrebe obračuna poreza na dodatnu vrijednost u međunarodnoj praksi i slično.

¹²⁶ Common Object Request Broker Architecture (CORBA), OMG, <http://www.omg.org/spec/CORBA/>

¹²⁷ Zakon o zaštiti osobnih podataka, Narodne novine, br. 103/03., 118/06., 41/08., 130/2011. i 106/2012.

Ostvarivanje e-Poslovanja je višedimenzionalni problem koji možemo sagledati unutar okvira za ostvarivanje e-Poslovanja (Slika 2.4) kroz dimenzije:

- **interoperabilnosti** na svim područjima: političkom, pravnom, organizacijsko-procesnom, semantičkom i tehničkom,
- **normi**, specifikacija, preporuka, referentnih modela i slično,
- **primjene tehnologija u izvedbi**, npr. za e-Identitet, e-Potpis, e-Isprava, e-Ugovor, e-Narudžbenica, e-Račun, e-Plaćanje, e-Katalog, e-Otprema, Opće namjene



Slika 2.4: Okvir za ostvarivanje e-Poslovanja¹²⁸

2.6.1 Elektronički identitet

Jedna od prvih pretpostavki za učinkovito e-Poslovanje je uvođenje **elektroničkog identiteta** ili **e- Identiteta**, popularno skraćeno zvanog i **eID**. Zapis o identitetu osobe u elektroničkom obliku osnova je za ostvarivanje bilo kakvog oblika elektroničke trgovine i mnogih usluga elektroničkog poslovanja, jer definira tko je uopće osoba koja sudjeluje u transakciji i poslovnom odnosu, bilo da je riječ o davatelju ili primatelju usluge. Osnovna ideja je korištenje e-Identiteta koji bi bio međunarodno, odnosno globalno prihvaćen i na taj način omogućio elektroničko poslovanje u međunarodnim odnosima, unutar EU-a i globalno. U Europskoj uniji pokrenute su brojne inicijative za uvođenje e-Identiteta, prvenstveno kako bi se realizirao zahtjev za globalizacijom kretanja građana, korištenja javne administracije, provedbe e-Poslovanja te upotrebe zajedničkog tržišta EEA (*European Economic Area*).

Osnovni **zahtjevi** na e-Identitet¹²⁹ uključuju međusobno povjerenje, sigurnost, stvarni identitet kao skup osobnih podataka, autentikaciju kao dokaz identiteta i potpis kao odobrenje određene izjave ili dokumenta.

¹²⁸ Slika iz Studije normizacije u e-Poslovanju, Isporuka 1, FER, Sveučilište u Zagrebu, Zagreb, 2009.

¹²⁹ Studija normizacije u e-Poslovanju, Isporuka 1, ver. 2.4, FER, Sveučilište u Zagrebu, Zagreb, 2009.

Za tehnološko rješavanje ovih zahtjeva koriste se različite **tehnologije**, uključujući sigurnu komunikaciju uporabom kriptiranja i povjerljivih kanala, usluge autentikacije, nacionalne registre stanovništva i pravnih osoba, zaporce u raznim oblicima, od PIN-a preko lozinki do biometrije, zatim pametne kartice (engl. *smart cards*) i tokeni, elektronički potpis i infrastruktura javnog ključa PKI (engl. *Public Key Infrastructure*).

Ako zahtjeve podijelimo na pojedina glavna područja, ističu se identifikacija, autentikacija i potpisivanje, koji zajedno čine tzv. **funkcionalnosti IAS** (*Identification, Authentication, Signature*).

U smislu zahtjeva za **identifikaciju** osobni podaci nositelja moraju biti u elektroničkom obliku, a od konkretnih podataka očekuje se prezime i ime, spol, datum i mjesto rođenja te potencijalno nacionalni identifikacijski broj (u hrvatskom slučaju OIB) i nacionalnost. Ako se podaci nalaze na elektroničkom uređaju, kao što je pametna kartica ili token, moraju biti zaštićeni PIN-om ili nekim drugim sigurnosnim rješenjem, kao što je biometrija, što je i u skladu sa Zakonom o zaštiti osobnih podataka. Dodatno u uređaju treba biti pohranjen naziv izdavatelja uređaja, broj uređaja, zemlja izdavatelja, datum izdavanja i datum isteka važenja.

Ako proučimo **autentikaciju**, osim što mora biti sigurna i pouzdana na razini sustava, postoji niz zahtjeva na potpisni ključ. Zahtjevi na potpisni ključ uključuju nemogućnost višestrukog korištenja, zaštitu od neovlaštene uporabe PIN-om ili biometrijom te uspoređivanje PIN-a ili biometrijskog obrasca na uređaju. Obrada PIN-a opisana je normom ISO/IEC 7816-4¹³⁰, a biometrija normom ISO/IEC 7816-11¹³¹, pri čemu se podatkovni objekti opisuju i normom ISO/IEC 19785-1¹³², a podaci o otisku prsta normom ISO/IEC 19794-2¹³³. Kako bi elektronički potpis bio zakonski valjan, sustav također mora udovoljavati određenim zahtjevima, kao što su sigurna i pouzdana funkcija **potpisivanja** uporabom uređaja koji sadržava potpis i funkcija pozitivnog odobrenja vlasnika potpisa, kao i neporecivosti. Elektronički potpis mora biti u skladu s Uredbom eIDAS.

Norma ETSI TS 101 456¹³⁴ opisuje procedure vezane uz upravljanje uređajem i certifikatima, dok dokumenti EN 14890-1 i EN 14890-2¹³⁵ opisuju problematiku ključeva, algoritama i formate za interoperabilnost na pametnim karticama. Implementacija infrastrukture javnog ključa postavlja zahtjeve za barem dva certifikata, od kojih je jedan za potpisivanje, a drugi za ostale funkcije, te usklađenost s X.509 V3 profilom.

Trenutno u EU mnogo zemalja ima razvijen e-Identitet, a i sve preostale ostale razmatraju uvođenje elektroničkog identiteta. No, postojeći e-Identiteti uvedeni su na različite načine i za potrebe vlastite zemlje, pa je interoperabilnost, iako podržana unutar jedne zemlje, na europskoj razini često nije jednostavna. Stoga se na razini cijele EU održava niz aktivnosti kako bi se poboljšala interoperabilnost. Jedan dio problema odnosi se i na interoperabilnost uređaja, odnosno pametnih kartica. Jedna od ideja je uporaba **kartice europskih građana**

¹³⁰ ISO/IEC 7816-4:2013 Organization, security and commands for interchange

¹³¹ ISO/IEC 7816-11:2017 Personal verification through biometric methods

¹³² ISO/IEC 19785-1:2015 Information technology -- Common Biometric Exchange Formats Framework -- Part 1: Data element specification

¹³³ ISO/IEC 19794-2:2011 Information technology -- Biometric data interchange formats -- Part 2: Finger minutiae data

¹³⁴ ETSI TS 101 456 v1.4.3 (2007-05), Technical Specification, Electronic Signatures and Infrastructures (ESI); Policy requirements for certification authorities issuing qualified certificates

¹³⁵ EN 14890-1 i EN 14890-2, Application interface for smart cards used as secure signature creation devices. Basic services & additional services

ECC (engl. *European Citizen Card*), koji objedinjuje funkciju osobne iskaznice i putovnice u zajedničkom elektroničkom identitetu.

U Hrvatskoj je FINA uspostavila infrastrukturu javnog ključa te je već neko vrijeme dobavljač kvalificiranih certifikata za pravne i fizičke osobe tzv. FINA e-kartica. Za potrebe omogućavanja pristupa i korištenja paneuropskih usluga, sustav treba preuzeti ulogu jedinstvene točke projekta STORK¹³⁶ za međusobno priznavanje elektroničkih identiteta izdanih u svim zemljama članicama EU. Očekivani rezultati tog projekta su:

- uspostavljen **zakonodavni okvir** kojim se propisuje izdavanje i korištenje jedinstvenog elektroničkog identiteta za svakog korisnika sustava umrežene uprave (građani, poslovni subjekti, službenici) na nacionalnoj razini
- proveden pilot **projekt sustava autentikacije** putem postojećih vjerodajnica različitih izdavatelja i razina pouzdanosti autentikacije (prema STORK QAA)
- izgrađen **sustav za upravljanje elektroničkim identitetima i autentikaciju korisnika**, te uspostavljeno njegovo operativno funkcioniranje iz svih aspekata: pravno, organizacijsko-kadrovski, tehnički, sigurnosno, financijski
- uvedena **nova elektronička osobna iskaznica** u skladu s ECC
- provedena **edukacija** službenika u tijelima javne vlasti za korištenje zajedničkog sustava za autentikaciju.

Netom prije stupanja Hrvatske u članstvo EU, u lipnju 2013. godine, donesen je Zakon o izmjenama i dopunama Zakona o osobnoj iskaznici¹³⁷, čime je započelo uvođenje nove redizajnirane osobne iskaznice s unaprijeđenim elementima zaštite i OIB-om. U drugoj fazi projekta izvedeno je i izdavanje osobnih iskaznica s elektroničkim nosačem podataka (čipom) koji omogućuje elektronski potpis nositelja osobne iskaznice, čime je realiziran projekt izrade **nove elektroničke osobne iskaznice**. Hrvatski državljani mogu prelaziti unutarnje granice Europske unije samo s važećom osobnom iskaznicom. Dodatno, s obzirom da je predviđena je mogućnost izdavanja osobnih iskaznica i djeci mlađoj od 16 godina, ovime se pojeftinjuje i pojednostavnjuje putovanja unutar EU i drugih država.

2.6.2 Elektronički potpis

Elektronički potpis ili **e-Potpis** sljedeća je bitna pretpostavka elektroničkom poslovanju. Kao što je to već prije objašnjeno, uredba eIDAS uređuje problematiku elektroničkog potpisa te definira značenja pojmova elektronički potpis, napredan elektronički potpis i kvalificirani napredni elektronički potpis, koji je zapravo napredni elektronički potpis fizičke osobe zasnovan na kvalificiranom certifikatu. U općem slučaju dva su najbitnija postupka vezana uz e-Potpis: **izrada potpisa** koju izvodi potpisnik, odnosno pošiljatelj, i **provjera potpisa** koju izvodi primatelj, a Svi važniji zakoni i podzakonski akti potrebni za provedbu elektroničkog potpisa u Hrvatskoj već postoje.

S obzirom na to da poslove certificiranja za potrebe javne uprave radi FINA, tamo je uspostavljena infrastruktura javnog ključa i centra za registre digitalnih certifikata RDC-FINA, čime je FINA ovlaštena izdavati svu potrebnu opremu za izvođenje i provjeru elektroničkog

¹³⁶ Projekt Secure idenTity acrOss boRders linKed 2.0

¹³⁷ Zakon o osobnoj iskaznici, Narodne novine, br. 62/2015.

potpisa, te izdavanje odgovarajućih certifikata. Za pravne osobe FINA izdaje sljedeće certifikate:

- autentikacijski (normalizirani) certifikat – koristi se za autentikaciju, odnosno enkripciju (zaštitu tajnosti podataka), te za njihovu kombinaciju i osigurava autentičnost, cjelovitost, izvornost i tajnost, izdaje na e-kartici ili USB tokenu ili kao certifikat za aplikacije koji se instalira na poslužitelju,
- potpisni (kvalificirani) certifikat – koristi se za elektroničko potpisivanje dokumenata ili transakcija naprednim elektroničkim potpisom, te jamči autentičnost, cjelovitost i izvornost te priskrbljuje i neporecivost zamjenjujući u cijelosti vlastoručni potpis ili vlastoručni potpis i otisak pečata, a vezan je uz osobu pa se izdaje na e-kartici ili USB tokenu.

Za građane odnosno fizičke osobe FINA može izdati:

- osobni certifikat srednje razine sigurnosti (certifikat na krypto uređaju)o
- osobni *soft* certifikat NCP (engl. *Normalized Certificate Policy*)

Pritom se koristi niz nacionalnih i međunarodnih normi koje uključuju već spomenutu uredbu eIDAS i niz normi vezanih uz poslovanje certifikatima, profil certifikata X.509, infrastrukturu javnog ključa i informacijsku sigurnost, od kojih su najvažnije:

- norme informacijske sigurnosti,
- norme elektroničkog potpisa
- norme vezane uz elektroničku poštu,
- norme vezane uz kriptografiju i kriptografske uređaje.

Elektronički potpis se u Hrvatskoj danas vičnom koristi od strane predstavnika pravnih osoba za potpisivanje dokumenata u odnosu prema tijelima javne uprave B2G, dok se u smislu razmjene potpisanih dokumenata poslovnih subjekata ili u privatne svrhe još uvijek prilično rijetko koristi, čemu je vjerojatno uzrok i u pomanjkanju navike i svjesnosti korištenja.

2.6.3 Elektronički račun

Elektronički račun ili **e-Račun** je elektronička poruka koja sadržava račun ili fakturu. Izdavanje računa ili fakturiranje događa se nakon naručivanja, definiranja isplate, ali prije procesa plaćanja. U praksi se često događa da se fizički račun zaprimi naknadno, nakon obavljenog plaćanja, ali to nije preporučeni, a ni legalan način poslovanja. Osim osnovnog klasičnog načina fakturiranja postoji i samofakturiranje (engl. *self-billing*), gdje kupac izdaje račun samome sebi uz autorizaciju dobavljača, odnosno prodavača.

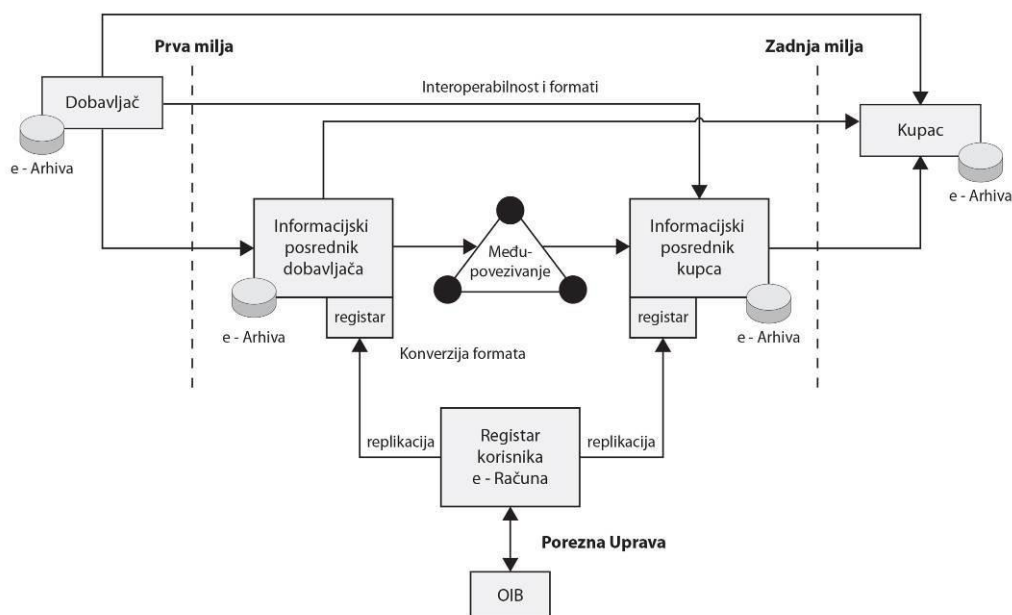
Elektronički je račun najrašireniji oblik elektroničke isprave u elektroničkom poslovanju na svijetu i zaslužan je za razvoj svih drugih procesa. Kao isprava elektronički je račun potpuno istovjetan papirnatom računu, a uz njega je moguće priložiti različite druge dokumente, bilo elektroničke ili papirnate koji su skenirani, kao što su dopisi, prilozi, obrasci, ugovori i slično. Temeljem raširenosti usluge internetskog bankarstva, elektronički je račun znatno dobio na popularnosti. Sadržaj elektroničkog računa definira se u nizu političkih i pravnih dokumenata, pa tako postoji nekoliko direktiva i uredbi EU te niz drugih dokumenata.

Ako se pri izdavanju računa koristi i posrednik, onda takav posrednik može izdati račun „u ime i u korist“ dobavljača, uz eksplicitnu autorizaciju dobavljača. Kupac može koristiti posrednika i za primanje račun i provjeru potpisa. Komunikaciju pojednostavnjuje ako je to jedan te isti posrednik.



U Hrvatskoj su dugo vremena postojale smetnje za uvođenje e-Računa u elektroničko poslovanje zbog neprilagođenog Pravilnika o porezu na dodanu vrijednost, jer je sadržavao odredbu kojom se propisivalo da se računi poslani telekomunikacijskim uređajem ne smatraju računima u smislu Zakona o porezu na dodanu vrijednost, uz neke iznimke. Zapravo se čekala reforma poreznog sustava i zakonodavno usklađivanje s EU, a navedeni Pravilnik o PDV-u¹³⁸ promijenjen je u srpnju 2011. godine.

Sljedeća slika prikazuje očekivanu izvedbenu tehnološku arhitekturu e-Računa na kojoj se može uočiti povezanost dobavljača, kupca, posrednika, registra i sustava OIB.



Slika 2.5: Tehnološka arhitektura e-Računa¹³⁹

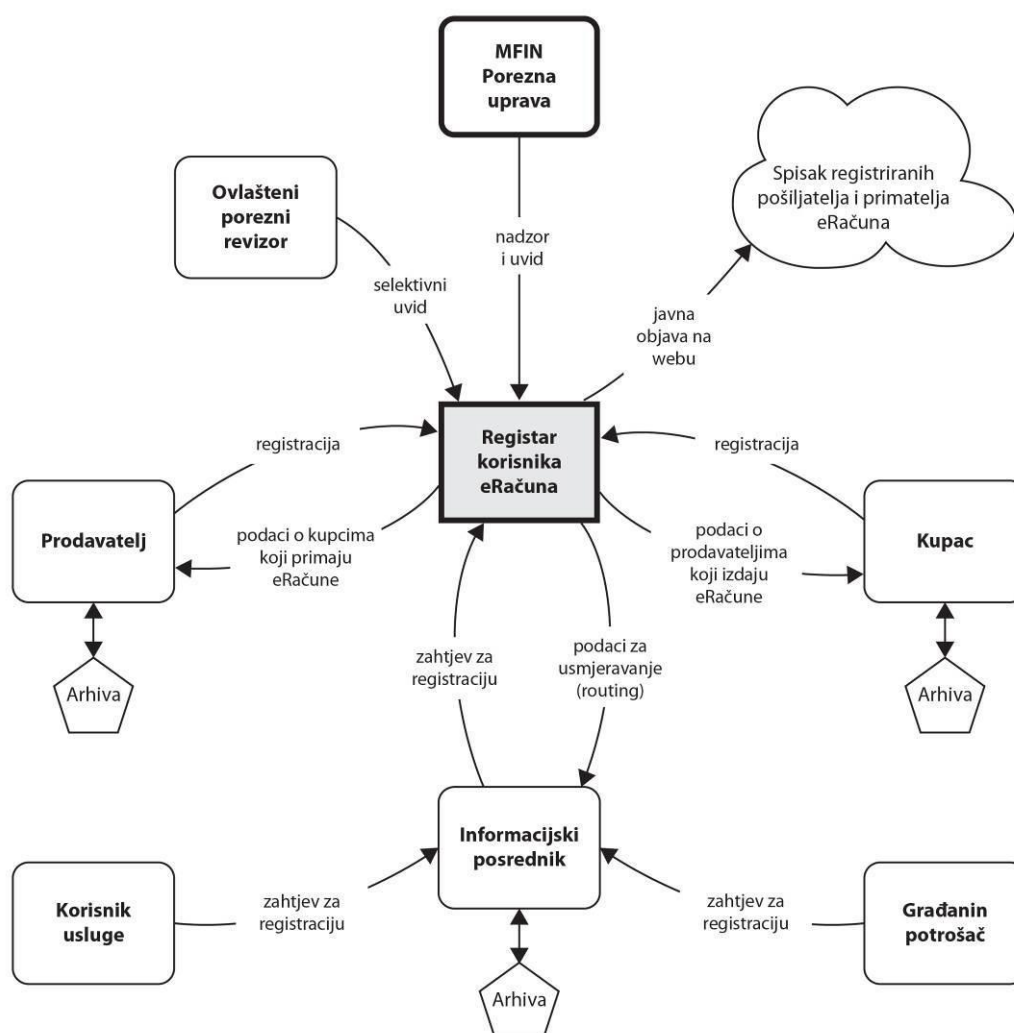
Kao što je vidljivo na slici, za korištenje elektroničkog računa potrebna je i centralna evidencija drugih korisnika koja se naziva **registar korisnika e-Računa**. Ovo nije zakonska obveza u EU, već usluga koja pojednostavljuje dobivanje informacija i korištenje e-Računa. U Hrvatskoj može postojati samo jedan registar pod nadzorom države, a korisnici su poslovni subjekti koji mogu primati i slati elektroničke račune te građani koji mogu samo primati elektroničke račune.

¹³⁸ Pravilnik o izmjenama i dopunama Pravilnika o porezu na dodanu vrijednost, Narodne novine, br. 89/2011.

¹³⁹ Slika iz Studije normizacije u e-Poslovanju, Isporučka 1, FER, Sveučilište u Zagrebu, Zagreb, 2009.

Osnovni procesi vezani uz registar su:

- registracija korisnika,
- dohvat podataka o kupcima (koji primaju e-Račun),
- dohvat podataka o prodavačima (koji šalju e-Račun),
- dohvat podataka za usmjeravanje poruka e-Računa,
- nadzor i uvid,
- javna objava pošiljatelja i primatelja.



Slika 2.6: Registar korisnika e-Računa¹⁴⁰

Registar sadržava podatke o pravnim subjektima i fizičkim osobama, a glavni ključ je osobni identifikacijski broj OIB, uz alternativne identifikatore, kao što su matični broj subjekta MBS, globalni lokacijski broj GLN od organizacije GS1, međunarodni broj bankovnog računa IBAN i slično.

¹⁴⁰ Slika iz Studije normizacije u e-Poslovanju, Isporučka 1, FER, 2009.

Podaci koji se pohranjuju uključuju:

- vrstu subjekta (poslovni, poslovni iz javnog sektora, građanin),
- naziv subjekta ili ime i prezime,
- mjesto i adresu subjekta,
- podatke o kontaktu (ime i prezime, adresa elektroničke pošte, telefon itd.),
- podatke o informacijskom posredniku (ako postoji),
- podatke o vrstama i inačicama dokumenata koje korisnik prima ili šalje,
- podatke o razdoblju korištenja.

Još 2009. godine je u Hrvatskoj bilo imenovano Povjerenstvo za e-Račun pri Ministarstvu gospodarstva, rada i poduzetništva, koje je počelo razvijati hrvatsku normu za e-Račun¹⁴¹ kroz definiranje informacijskog modela. Svi elementi informacijskog modela preslikani su na elemente norme **UBL 2.0** te je specifikacija norme e-Račun zapravo podskup norme UBL. Istovremeno se pratilo i razvoj norme **UN/CEFACT CII 2.0** (engl. *Cross Industry Invoicing*) zasnovane na jeziku XML, koju podržava i organizacija GS1, a koja će se u budućnosti potencijalno također podržati pri preslikavanju elemenata modela e-Računa. Norma CII jedna je od rijetkih koja podržava normiranja koja se mogu koristiti u različitim industrijskim granama, što je prouzročilo spori proces donošenja norme. Kod elektroničkog je računa moguće koristiti i normu **GS1 XML**, koja je prilično kvalitetna i prihvaćena, no svi korisnici moraju biti članovi organizacije GS1 i koristiti šifre GTIN. Više o prijedlogu specifikacije e-Računa, te strukturi i izvedbi e-Računa navedeno je kasnije.

U Hrvatskoj trenutno mnoge financijske institucije nude neke usluge vezane uz e-Račun koje omogućuju razmjenu elektroničkog računa između izdavatelja i primatelja računa te upravljanje cjelokupnim poslovnim procesom izdavanja, zaprimanja te arhiviranja računa.

2.6.4 Elektroničko plaćanje

Pod **elektroničkim plaćanjem** smatra se prijenos novčanih sredstava u skladu s izdanim računom kao završni proces nabave roba ili usluga. Oblik elektroničkog plaćanja koji se provodi korištenjem elektroničkog bankarstva prilično je razvijen u Hrvatskoj, jer praktički sve veće banke nude uslugu **internetskog bankarstva** korištenjem preglednika Weba, a u zadnje vrijeme i tzv. **mobilnog bankarstva**, odnosno aplikacije za pametne telefone, primjerice za Android i iOS te druge operacijske sustave. No, internetsko bankarstvo nije cijeli proces elektroničkog plaćanja, već samo njegov „prozor prema korisniku“, jer cjelovita usluga uključuje i niz drugih procesa, kao što je automatsko zadavanje naloga za plaćanje, provođenje plaćanja i knjiženje.

Zakon o platnom prometu u zemlji postavlja pravni okvir nacionalnih i međunarodnih plaćanja, a čitav je proces nadopunjen normiranjem obrasca naloga za plaćanje od strane Hrvatske udruge banaka. Normirani obrazac HUB 1 i HUB 1-1, a kasnije HUB 3 i HUB 3A, postali su norma, a u učestaloj upotrebi je i zapis podataka na papirnatom obrascu uporabom dvodimenzionalnog crtičnog koda (engl. *2D bar code*), kao i elektronička inačica platnog naloga e-HUB zapisana u jeziku XML. Po ulasku Hrvatske u EU međunarodni platni promet došlo je do daljnjeg usklađivanja prema sustavu SEPA (*Single Euro Payment Area*) Europskog

¹⁴¹ Specifikacija elektroničkog računa, nacrt, verzija 0.1, Povjerenstvo za e-Račun, MINGORP, 2009-12-23

vijeća za platni promet EPC (*European Payments Council*) i njegovim normama i shemama elektroničkog plaćanja.

Za pokretanje procesa elektroničkog plaćanja gospodarski i javni subjekti koriste usluge svojih poslovnih banaka. Sve su banke u Hrvatskoj povezane dvama međubankovnim platnim sustavima:

- **Nacionalnim klirinškim sustavom (NKS)** koji služi za izvršenje malih i masovnih plaćanja s obračunom na neto multilateralnom načelu i
- **Hrvatskim sustavom velikih plaćanja (HSVP)** koji služi za izvršenje međubankovnih plaćanja i knjiženje na računima banaka u realnom vremenu na bruto-načelu.

Pomoću ova dva sustava banke izvode sva međubankarska plaćanja, osim naplata unutar vlastite institucije, a razlika NKS-a i HSVP-a su prije svega u načinu obrade naloga za plaćanje, njihovoj svrsi i obračunu troškova platnih transakcija, odnosno njihovoj cijeni. Sva međunarodna plaćanja, odnosno plaćanja u stranim valutama izvode su korištenjem mreže **SWIFT** (*Society for Worldwide Interbank Financial Telecommunication*).

Sustavom **NKS** od početka rada 2001. godine operativno upravlja FINA, a funkcionalno Hrvatska narodna banka (HNB). Sustav NKS je platni sustav za obračun naloga za prijenos između njegovih sudionika na neto multilateralnom načelu, što znači kontinuirani obračun naloga za prijenos podnesenih na obračun od strane sudionika tijekom obračunskog dana sustava NKS. Izravni sudionici sustava NKS su banke, HNB i treće strane, a posredni sudionici su i druge financijske institucije koje izvršavaju namiru međubankovnih platnih transakcija. Sustav NKS služi kao sustav za obračun međubankovnih platnih transakcija (tzv. sustav malih plaćanja), namira velikog broja plaćanja koja glase na relativno male iznose u trima obračunskim ciklusima tijekom radnog dana. Sučelje sustava NKS prema bankama izraženo je kroz sigurnu komunikaciju, a svi podaci koji se razmjenjuju potpisani su elektroničkim potpisom.

Sustav **HSVP** je platni sustav za namiru naloga za prijenos novčanih sredstava u realnom vremenu na bruto načelu, pri čemu se namira na računima sudionika tijekom obračunskog dana u sustavu HSVP provodi pojedinačno, nalog po nalog, u trenutku kada je pojedinačni nalog zaprimljen. To znači da HSVP funkcionira kao sustav bruto namira u stvarnom vremenu RTGS (*Real Time Gross Settlement*). Vlasnik i upravitelj sustava HSVP je HNB, a započeo je s radom 1999. godine kao sustav za kliring i namiru – središnji računovodstveni sustav, proizvod tvrtke LogicaCMG (*Logica Clearing and Settlement System – Central Accounting System – LCSS CAS*). Sudionici u sustavu HSVP su banke i HNB te Središnje klirinško depozitarno društvo (SKDD). Platne transakcije provode se putem platnih poruka i razmjenjuju se mrežom SWIFT.

2.6.5 Poslovni procesi elektroničkog poslovanja

Svi poslovni procesi elektroničkog poslovanja nisu normirani, već postoje u različitim oblicima. Najčešće su zapisani u obliku tekstualnog opisa, a svakako je poželjno modeliranje procesa i grafički prikaz pomoću nekog oblika zapisa modela i grafičke prezentacije, kao što je jezik **UML** (*Unified Modeling Language*). Jezik UML koristi se za opis aktivnosti, događaja, dijelova, procesa, uloga i odnosa na različitim razinama. Za to postoje različite vrste dijagrama, kao što su slučaj korištenja (engl. *use case*), dijagram aktivnosti (engl. *activity diagram*), dijagram klasa (engl. *class diagram*), dijagram komponenata (engl. *component diagram*), dijagram stanja

(engl. *state machine diagram*), dijagram sekvencija (engl. *sequence diagram*), komunikacijski dijagram (engl. *communication diagram*) i drugi.

Primjerice, kao osnovni poslovni procesi dobavnog lanca (engl. *supply-chain*) identificirani su:

- **Natjecanje** ili **nadmetanje** (engl. *tendering*) – proces natječaja i natjecanja za naručivanje radova, roba i usluga. Najčešće se primjenjuje u javnoj nabavi. U Hrvatskoj se trenutno natjecanja većinom odnose na javni sektor gdje postoji Zakon o javnoj nabavi, dok se u privatnom sektoru tek očekuje šira primjena. U procesima javne nabave koristi se jedinstveni rječnik javne nabave CPV.
 - Osnovni poslovni procesi vezani uz natjecanje su: registracija, poziv na natjecanje, slanje informacija o natjecanju, slanje natječajne dokumentacije, otvaranje natječajne dokumentacije, odabir i objava pobjednika.
- **Nuđenje** (engl. *bidding*) – proces nuđenja određene robe i usluge za određenu cijenu
 - Ovom procesu osim definiranih kataloga (opisanih u sljedećoj natuknici) pripadaju i definirani cjenici u pripadajućim mjernim količinama (npr. komadi, paketi, težina, obujam i sl.). Alternativno se nuđenje može izvoditi i korištenjem aukcije i drugih metoda bez unaprijed poznate konačne cijene, već se cijena prilagođava tijekom procesa nuđenja.
- **Katalog** (engl. *catalog*) – popis proizvoda, roba i usluga dostupnih za narudžbu, uz klasifikaciju i s automatskim korištenjem jedinstvenih identifikatora. Katalogi bi trebali biti višjezični i semantički jasni, kako na lokalnoj i nacionalnoj tako i na međunarodnoj razini. Neki od primjera međunarodnih kataloga su otvoreni elektronički katalog opće namjene **UNSPSC** (engl. *United Nations Standard Product and Services Code*), europski komercijalni katalog proizvoda, materijala i usluga **eCl@ss**, katalog proizvoda i roba unutar GS1 sustava **GS1 GPC** (engl. *Global Product Classification*), koji koristi GPC „Brick“ kodove i **GTIN** (engl. *Global Trade Identification Number*) kodove najčešće prikazane korisnicima u obliku crtičnog koda ili prezentirane tehnologijom RFID (engl. *Radio Frequency ID*), te mnogi drugi. U Hrvatskoj se najviše koriste katalog **eCROKAT**, koji je sukladan GS1 GPC specifikaciji, ali je dostupan samo članovima GS1 organizacije, te klasifikacija roba i usluga u obliku **Zajedničkog rječnika nabave CPV** (engl. *Common Procurement Vocabulary*), koji nije pravi katalog, već samo rječnik javne nabave. U budućnosti se preporučuje šira uporaba eCROKAT-a, eCl@ss-a i drugih kataloga u skladu s normom ISO 13584.
 - Osnovni poslovni procesi vezani uz katalog su: zahtjev za katalogom, objava, pretplata, ažuriranje i razmjena
- **Ugovor** (engl. *contract*) – sporazumi dviju strana o izvršenju dobave proizvoda, robe ili usluge. U Hrvatskoj postoji razrađena zakonska regulativa koja omogućuje uporabu elektroničkog ugovaranja, kao što je navedeno u potpoglavlju 0. Svaki ugovor u skladu sa Zakonom o elektroničkoj ispravi mora obvezno sadržavati napredni elektronički potpis i podatke o vremenu nastanka ugovora te se onda može smatrati izvornikom. Na razini EU-a ugovor nije formalno oblikovan, već samo definira dogovor i razumijevanje strana te postupak nuđenja i prihvaćanja ponude.
- **Naručivanje** (engl. *order*) – dogovor oko nabave, dostave ili izvršenja i oblika plaćanja određene robe ili usluge. Uobičajena pretpostavka u procesu naručivanja je proces ugovaranja, odnosno sklapanja ugovora dviju strana koji se onda detaljizira kroz narudžbu. U Hrvatskoj se još uvijek u načelu koristi oblik poruke narudžbe prema normi **GS1 EANCOM** temeljenoj na normi **EDIFACT**. U budućnosti se preporučuje prijelaz na napredniju normu **GS1 XML**, a smatra se, s obzirom na to da su e-Narudžba i e-Račun

usko podatkovno povezani procesi, kako bi norma za e-Narudžbu zapravo bila potpuno sukladna normi e-Računa.

- Osnovni poslovni procesi vezani uz naručivanje su: stvaranje prijedloga narudžbe, slanje prijedloga narudžbe dobavljaču, potencijalne promjene narudžbe, prihvata (ili odbijanje) narudžbe.
- **Izdavanje računa ili fakturiranje** (engl. *billing*) – proces izdavanja računa
- **Plaćanje** (engl. *payment*) – transfer novčanih sredstava u skladu s računom
- **Obavješćavanje o plaćanju** – komunikacija o plaćanju kupca i dobavljača generiranjem i prijenosom obavijesti
- **Vremensko terminiranje isporuke** (engl. *scheduling*) – proces određivanja i praćenja točnog plana isporuka pojedinih proizvoda ili usluga kako bi i dobavljač i kupac mogli optimalno planirati potrebe angažiranja resursa, bilo ljudskih, bilo robnih, zalihe, skladištenje i slično. Ovdje se često isprepliću interni procesi obiju strana, podržani najčešće od sustava ERP (engl. *Enterprise Resource Planning*).
 - Osnovni poslovni procesi vezani uz vremensko terminiranje isporuke su: prognoziranje potražnje, prognoziranje potrošnje, upute za isporuku i obavijest o isporuci
- **Otpremanje** (engl. *dispatching, shipping*), **prijevoz** (engl. *transport*) i **primanje** (engl. *receiving*) – procesi vezani uz isporuku i primopredaju roba i usluga kupcu ili krajnjem korisniku. Neće biti pitanje obrađivani s obzirom na to da prate druge navedene procese i njihove elektroničke izvedbe, no podrazumijeva se da mora postojati i prikladna podrška u smislu informacijskog sustava za ove procese i primopredaju roba i usluga.

2.7 Norme za zapis znakova i hrvatski grafemi

U ovom poglavlju objasniti ćemo kodiranje znakova i norme za zapis znakova. No, krenimo od najjednostavnijeg pojma. **Grafem** je simbol ili znak određenog glasa nekog jezika. U pismu, ali i na računalnom zaslonu, grafemi se prikazuju znakovima ili glifovima, pa možemo reći da je **glif** na određeni način oblikovani prikaz grafema. Određeni skup glifova koji su izvedeni u istom stilu čini **font** ili **pismo**. U računarstvu se češće ipak koristi naziv **znak** (engl. *character*), koji također predstavlja grafem, odnosno njegov glif.

Kodiranje znakova (engl. *character encoding*), često nazvano i **kodnom stranicom** (engl. *code page*), **skupom znakova** (engl. *character set*) ili **mapom znakova** (engl. *character map*), označava tablicu skupa znakova koji se koriste u jezicima i pripadajuće numeričke oznake njihovog smještaja u skupu te poredak.



Iako se osim numeričkih oznaka za zapis znakova mogu koristiti i razni drugi oblici, poput električnih impulsa i različiti oblici zapisa, primjerice Morseov kôd, ovdje se ograničavamo na uobičajene zapise znakova za pohranu i prijenos u računalima.

2.7.1 Povijest normi za zapis znakova

U ovom kratkom povijesnom pregledu krenut ćemo od samih početaka kodiranja znakova u računarstvu. Nekadašnja organizacija ASA, odnosno njegova komisija, koja je kasnije nazvana USASI te konačno ANSI, razvila je još davne 1963. godine za potrebe telegrafa skup znakova

nazvan **American Standard Code for Information Interchange** (akronim **ASCII**). IANA ga još naziva i **US-ASCII**, a zadnja revizija izašla je 1986. godine. Sljedeća tablica prikazuje skup znakova ili kôd ASCII koji koristi svega 7 bitova da definira 128 znakova, od čega 33 neispisiva kontrolna znaka, 93 ispisiva znaka i prazninu (engl. *space*).



Originalni ASCII koristi samo 7 bitova za zapis znaka, a iako je bilo razmišljanja o uporabi 8 bitova, u ono je doba prevladao razlog štednje na memoriji i prijenosu. No, osmi se bit katkad koristio kao paritetni bit, a u kasnijim je izvedbama često bio jednostavno postavljen na nulu.

	-0	-1	-2	-3	-4	-5	-6	-7	-8	-9	-A	-B	-C	-D	-E	-F
0-	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
1-	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2-		!	"	#	\$	%	&	'	(	)	*	+	,	-	.	/
3-	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4-	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5-	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6-	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7-	p	q	r	s	t	u	v	w	x	y	z	{		}	~	DEL

Tablica 2.4: Skup znakova ASCII

Kontrolni znakovi uključuju prelazak u novi red (engl. *line feed*), pomak na početak reda (engl. *carriage return*), tabulator, zvono, brisanje (engl. *backspace*), znak „*escape*“ i slične.



Zanimljivo je da različiti sustavi, platforme i proizvođači i dan danas različito tumače „novi red“, pa tako sustavi temeljeni na Unixu, Linuxu i slično i dalje koriste samo znak LF (engl. *line feed*), dok sustavi temeljeni na Microsoftovim operacijskim sustavima MS-DOS i Windows koriste oba znaka, LF i CR (engl. *carriage return*), a donedavno je i Appleov macOS koristio samo znak CR. Na Internetu je ipak najčešće korišten oblik kombinacije obaju znakova.



Zanimljivo je da se razvio i jedan stil u umjetnosti, odnos tehnika grafičkog dizajna nazvana „ASCII art“, koja koristi ispisive znakove iz skupa ASCII za prikaz različitih slika i grafika.

Istovremeno s razvojem skupa znakova ASCII, IBM za potrebe svojih *mainframe* (centralnih) računala razvija **Extended Binary Coded Decimal Interchange Code (EBCDIC)**, koji koristi 8 bitova za zapis znaka. Tijekom povijesti EBCDIC je imao dosta inačica pa mu je jedna od zamjerki kompatibilnost između tih verzija. Jedna od inačica koja je bila rasprostranjena na

mainframe računalima u nama bliskom okruženju je inačica **EBCDIC Latin 2 Multilingual**, poznatija i pod nazivom **IBM Latin-2**, odnosno kodna stranica **IBM CP870**. Ovaj se način zapisa danas više jako rijetko koristi.



Iako su imali neke vrlo dobre ideje, poput znakova za slova engleske abecede u abecednom poretku i razliku između velikih i malih slova u samo jednom bitu, što je znatno pojednostavnilo operacije sortiranja i usporedbe, ASCII i slični kodovi uskoro su se susreli s velikim problemom. Naime, 127 znakova je jednostavno bilo premalo, čak i za sva latinična pisma, a kamoli za neka druga, kao što su ćirilica, grčko, arapsko ili kinesko pismo. Iako je još prije desetak godina zastarjeli US-ASCII je široko raširen znakovni skup na stranicama weba prvenstveno engleskoga govornog područja, danas ga je zamijenio UTF-8.

2.7.2 Internacionalizacija normi za zapis znakova

Potaknuti razvoje Interneta i općom globalizacijom, na međunarodnoj se razini još u prošlom stoljeću pokušavalo doći do norme za zapis znakova, pa uvažavajući postojeće *de facto* telekomunikacijske norme ISO i IEC definiraju normu **ISO/IEC 646:1991**, *Information technology - ISO 7-bit coded character set for information interchange*, prvi put predstavljenu 1972. godine. Norma ISO/IEC 646 zapravo potpuno prati normu i znakovni skup ASCII, osim što se iz nje izvode nacionalne inačice norme umetanjem pripadajućih nacionalnih grafema na određenim mjestima. Tako nastaje niz nacionalnih inačica za većinu europskih zemalja, neke azijske inačice, kao što su Japan, Kina i Koreja, inačice za američki kontinent, odnosno inačica za Kanadu i inačica za SAD, koja je zapravo i dalje norma ASCII, te osnovna globalna inačica, odnosno međunarodna referentna inačica IRV (engl. *International Reference Variant*).

Na području bivše države koristili su se jezici koji koriste znakove s dijakriticima (č, ć, đ, š i ž), pa se također nakon početnog korištenja norme ASCII zaključilo kako računala moraju moći koristiti i nacionalne grafeme. Izvedba je iskoristila 10 manje korištenih znakova iz skupa ASCII i zamijenila ih znakovima: č, Č, ć, Ć, đ, Đ, š, Š, ž i Ž. Tako je zapravo nastala prvo norma **JUS I.B1.002:1982** sa znakovima zapisanim u 7 bitova, popularno nazvana **YUSCII**, koja je kasnije usklađena i s normom ISO-646:1973 tako da je izvedena inačica **ISO-646-YU**. Ta je norma kasnije u Hrvatskoj postala i hrvatska norma **HRN I.B1.002**. No, jedan od problema prilikom korištenja ove norme je nemogućnost ispravnog sortiranja korištenjem uzlaznog poretka kodova.

	-0	-1	-2	-3	-4	-5	-6	-7	-8	-9	-A	-B	-C	-D	-E	-F
0-	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
1-	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2-		!	"	#	\$	%	&	'	(	)	*	+	,	-	.	/
3-	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4-	Ž	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5-	P	Q	R	S	T	U	V	W	X	Y	Z	Š	Đ	Ć	Č	—
6-	ž	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7-	p	q	r	s	t	u	v	w	x	y	z	š	đ	ć	č	DEL

Tablica 2.5: Norma HRN I.B1.002 (bivša ISO-646-YU)

Dolaskom osobnih računala predvođenih IBM PC-om koji je imao operacijski sustav MS-DOS došlo je do potrebe razvoja novog skupa znakova te je tako u Microsoftu i IBM-u razvijen niz kodnih stranica za operacijski sustav DOS, odnosno tzv. **DOS code page**, od kojih je najpoznatija kodna stranica s brojem 437 za engleski jezik, skraćeno **CP437**, još nazvana DOS-US ili OEM-US (Slika 2.7).



Slika 2.7: Kodna stranica 437 za MS-DOS



Osnovne razlike CP437 u odnosu na ASCII su što se kontrolni znakovi od heksadekadskih kodova 00 do 1F, kao i 7F (DEL) mogu i grafički prikazati, a kodovi od 128 do 255 koriste se za različite simbole, neke zapadnoeuropske grafeme, matematičke simbole, znakove za crtanje okvira i poneka grčka slova. U CP437 opet nedostaju neki znakovi, odnosno slova iz španjolskog, francuskog, portugalskog i skandinavskih jezika te preglasi u njemačkom, a kasnije su razvijeni i drugi skupovi znakova.

S obzirom na to da kodna stranica CP437 definira nepotpunu međunarodnu inačicu koja ne sadrži određene znakove nacionalnih abeceda, kasnije se izvode i druge inačice kodne stranice, konkretno inačice koje započinju sa znamenkom 7 i znamenkom 8, od kojih su nama najpoznatije CP850 za zapadnoeuropske jezike te **CP852**, nazvana još i **Latin II**, **IBM PC Latin-2** ili **Microsoft C-852**, za jezike središnje Europe, kojima je pripao i hrvatski.



Glavna razlika kodnih stranica CP850 i CP852 od CP437 je što su određeni znakovi za crtanje okvira, grčka slova i još neki drugi znakovi zamijenjeni nacionalnim grafemima.

Kodna stranica CP852 se dosta raširila u upotrebi na našem području, ali ona više ne slijedi ideju razlike malih i velikih slova hrvatskih grafema u jednom bitu. Združivanjem kodne stranice CP437 i norme JUS I.B1.002 nastala je i tablica koju nazivamo **CP- 999**, koja zapravo odgovara normi ISO/IEC 646 YU. Sve ove norme i dalje prate određeni problemi s jednostavnim sortiranjem.

Međunarodna inicijativa ponovno počinje izrađivati norme s prikazom u 8 bitova temeljene na ASCII-u, koje obuhvaćaju sve više grafema te je s njima manje problema, pa tako nastaje norma ISO/IEC 8859 podijeljena na 16 dijelova, od kojih su neki kasnije i napušteni. Za naše su područje najpoznatije su dvije norme.

Norma ISO/IEC 8859-1 za grafeme zapadnoeuropskih jezika, u zadnjoj inačici punim nazivom nazvana **ISO/IEC 8859-1:1998**, *Information technology - 8-bit single-byte coded graphic character sets - Part 1: Latin alphabet No. 1*, a često označena i kao **Latin-1**, odnosno po

nomenklaturi IANA **ISO-8859-1**, koja rješava većinu europskih grafema, ali ne i sve hrvatske. Zato je naknadno nastala i norma ISO/IEC 8859-2, odnosno **ISO/IEC 8859-2:1999**, *Information technology — 8-bit single-byte coded graphic character sets — Part 2: Latin alphabet No. 2*, zvana još i **Latin-2** ili po nomenklaturi IANA **ISO-8859-2**, koja uključuje sve hrvatske dijakritičke znakove.



Zanimljivo je da postoji i norma ISO/IEC 8859-16 ili Latin-10 koja još jednom definira hrvatske grafeme, ali koja nikad nije zaživjela u široj upotrebi. Postoji i IBM-ova inačica ove norme nazvana IBM CP-912, koja je konsolidirana verzija kodne stranice, a primjenjuje osnovne postavke iz norme ISO/IEC 8859-2. Iako međunarodno propisana, ova se norma polako probijala na našem području. Na području bivše države uglavnom se prihvaća norma ISO/IEC 8859-2 kao norma JUS I.B1.013.

Dodatne probleme u već mnoštvo normi unosi Microsoft uvođenjem novog skupa kodnih stranica u svoje operacijske sustave Windows. To se odnosi na kodne stranice koje započinju znamenkom 9 za dalekoistočne zemlje te niza kodnih stranica koje započinju znamenkama 12, od kojih su nama najpoznatije 1250 i 1252. Ove kodne stranice nisu nikad postale međunarodne norme, iako se Microsoft za to zalagao, ali su postali *de facto*, odnosno tržišne norme. Kodna stranica **Windows- 1252** ili **CP-1252** predviđena je za engleski i neke zapadnoeuropske jezike i dijelom je kompatibilna s normom ISO/IEC 8859-1, ali nažalost ne u potpunosti, jer su neki znakovi zamijenjeni. Na našem je području poznatija kodna stranica **Windows-1250** ili **CP-1250** za jezike središnje Europe, što uključuju hrvatski jezik, koja dijeli dio znakova s Windows 1252, a djelomično je kompatibilna i s normom ISO/IEC 8859-2. No kodna stranica Windows-1250 unosi dodatnu pomutnju jer neki znakovi nisu samo zamijenjeni drugima u odnosu na normu ISO, već su im zamijenjena i mjesta. Popularizacijom operacijskog sustava Windows kodna se stranica Windows-1252 u Hrvatskoj počinje široko koristiti te i u našoj zemlji postaje *de facto* norma, što unosi dodatne probleme oko zapisivanja ili prihvata dokumenata s drugih operacijskih sustava.

2.7.3 Univerzalni skupovi znakova – UTF i UCS

Uvidom da podjela na pojedine kodne stranice, pogotovo vezane uz određene sustave i proizvođače, unosi samo dodatne probleme, došlo se do ideje izvedbe univerzalnog skupa znakova koji će se moći koristiti na globalnoj, svjetskoj razini. Pitanja poput kako zapisati jedan dokument s više pisama, odnosno na više jezika te kako podržati neke jezike s više tisuća znakova dodatno su povećala pritisak tržišta u zbog globalizacije i potrebe za globalnom interoperabilnosti.

Konzorcij Unicode, neprofitna organizacija koja koordinira razvoj **norme Unicode** za razmjenu podataka, došla je do ideje razvoja norme temeljene na zapisu u 16 bitova i sa 65.536 znakova. Istovremeno je ISO došao do slične ideje izdavanja univerzalne norme koju je znakovito nazvao **ISO 10646**, kako bi se vidjela povezanost s normom ISO/IEC 646. Inicijalna ideja bila je zasnovana na zapisu od 128 grupa od 256 tablica s 256 redaka i 256 stupaca, koja bi u praksi omogućavala zapis 679.477.248 znakova. S obzirom na to da industrija nije željela prihvatiti tako složen i memorijski zahtjevan zapis znakova, ISO je prihvatio osnovnu ideju norme Unicode i zapis od 16 bitova, koja se ipak kroz vrijeme pokazala nedovoljnom pa su osmišljene nove inačice norme ISO 10646 i norme Unicode. Zapis znakova u normama ISO i

Unicode zasniva se na univerzalnim znakovnim skupovima **UCS** (*Universal Character Set*) i **UTF** (*Unicode Transformation Format*). Tako su određeni dijelovi norme Unicode 2.0 potpuno odgovarali normi ISO/IEC 10646-1:1993. Konzorcij Unicode i ISO/IEC i danas tijesno surađuju u razvoju novih inačica normi ISO/IEC 10646 i Unicode.

Najraširenija inačica norme Unicode za zapis znakova koristi 17 razina (engl. *plane*) ili tablica s 2^{16} znakova ili tzv. kodnih točaka (engl. *code point*), što ukupno omogućuje u kodovima od 0_{16} do $10FFFF_{16}$ zapis 1.114.112 različitih znakova, a trenutno se koristi tek oko 10%. Najčešće se koristi samo nulta razina (engl. *plane 0*) koja za zapis znakova koristi 16 bitova od 0_{16} do $FFFF_{16}$, a koja se još naziv i osnovna višjezična razina **BMP** (engl. *Basic Multilingual Plane*). U razini BMP nalazi se većina znakova modernih jezika i velik broj specijalnih znakova. Zanimljivo je da je prvih 256 znakova potpuno sukladno normi ISO/IEC 8859-2.



Standardni prikaz znakova je korištenjem slova „U“ i znaka „+“ te heksadekadskim kodom znaka, pa se tako znakove iz razine BMP može prikazati u obliku „**U+nnnn**“, gdje je nnnn kôd znaka. Standardna engleska abeceda u razini BMP započinje slovom „A“ u kodnoj točki U+0041, a završava slovom „z“ u kodnoj točki U+007A.

Iako su postojali razni oblici zapisa, danas imamo tri najpoznatije vrste zapisa znakova:

- UTF-8,
- UTF-16 i UCS-2,
- UTF-32 i UCS-4.

UTF-8 je vrsta kodiranja znakova varijabilne dužine od 1, 2, 3 ili 4 bajta. Ako se koristi jedan bajt, odnosno zapis od 00_{16} do $7F_{16}$, onda se zapisuje prvih 128 znakova koji su sukladni normi ASCII, što omogućuje unatražnu kompatibilnost. Za zapis hrvatskih grafema potrebno je ipak 2 bajta, jer su hrvatski dijakritički znakovi u području vrijednosti kodnih točaka od 0100_{16} do $7FFF_{16}$. Pomoću zapisa od 3 i 4 bajta mogu se zapisati praktički svi znakovi svih razina. Zapis u kodiranju UTF-8 je danas vrlo raširen jer se koristi kao pretpostavljeni oblik u jeziku XML, za zapisivanje dokumenata, kod elektroničke pošte, za zapis stranica Weba i u mnoge druge svrhe. Prema IETF-u svi protokoli Interneta moraju podržavati UTF-8¹⁴², a prema IMC-u (engl. *Internet Mail Consortium*) svi programi za čitanje, slanje i upravljanje elektroničkom poštom također bi trebali podržavati UTF-8. UTF-8 je uspio biti toliko široko prihvaćen zbog unatražne kompatibilnosti s ASCII-om kod kodiranja jednim bajtom, a i lako ga se može prepoznati.

Sljedeća tablica 2.6 prikazuje način zapisa i **kodiranje pomoću UTF-8**, koje ćemo i objasniti. U slučaju kodiranja jednim bajtom za vrijednosti od 00_{16} do $7F_{16}$, najznačajniji bit se postavlja u 0, a preostalih 7 bitova označava vrijednost kodne točke. Kod kodiranja s dva bajta za vrijednosti od 80_{16} do $7FF_{16}$, donji bajt započinje bitovima 10, a slijedi 6 donjih bitova vrijednosti kodne točke, dok gornji bajt započinje sa 110 i slijedi 5 gornjih bitova vrijednosti kodne točke. Kodiranje s tri bajta za vrijednosti od 800_{16} do $FFFF_{16}$ slijedi istu logiku, pa donji bajt započinje s bitovima 10 i slijedi 6 donjih bitova vrijednosti kodne točke, srednji bajt isto započinje s bitovima 10 i slijedi 6 srednjih bitova vrijednosti kodne točke, a gornji bajt započinje bitovima

¹⁴² Alvestrand, H., IETF Policy on Character Sets and Languages, RFC2277, Internet Engineering Task Force, <http://tools.ietf.org/html/rfc2277>

1110 i slijede 4 gornja bita vrijednosti kodne točke. Analogno, kod kodiranja s četiri bajta, za vrijednosti od 10000_{16} do $10FFFF_{16}$, sva tri niža bajta započinju s bitovima 10 i slijedi 6 bitova iz vrijednosti kodne točke, dok jedino gornji bajt započinje s bitovima 11110 i slijede 3 gornja bita vrijednosti kodne točke. Može se zaključiti da kod kodiranja s 2, 3 i 4 bajta gornji bajt započinje s 2, 3 ili 4 bita u 1, nakon čega slijedi bit u 0, a zatim preostali bitovi vrijednosti kodne točke, dok niži bajtovi uvijek započinju bitovima 10 koje slijede 6 bitova vrijednosti kodne točke, prema raspodjeli 5+6, 4+6+6 ili 3+6+6+6. Tablica 2.6 sadrži potanji prikaz sva četiri oblika zapisa. Tako je lako provjeriti i dužinu koda jer kodiranje s 1 bajtom nosi vrijednosti od 00_{16} do $7F_{16}$, kodiranje s 2 bajta u gornjem bajtu ima vrijednosti od $C2_{16}$ do DF_{16} , kodiranje s 3 bajta u gornjem bajtu ima vrijednosti od $E0_{16}$ do EF_{16} , a kodiranje s 4 bajta u gornjem bajtu ima vrijednosti od $F0_{16}$ do $F4_{16}$, dok se u nižim bajtovima uvijek nalaze vrijednosti od 80_{16} do BF_{16} .

Opseg kodne točke	Shema bitova	Zapis u UTF-8 (binarno)
000000 - 00007F	00000000 00000000 0AAAAA	0AAAAAAA
000080 - 0007FF	00000000 0000BBBB BBAAAAAA	110BBBBB 10AAAAAA
000800 - 00FFFF	00000000 CCCCBBBB BBAAAAAA	1110CCCC 10BBBBBB 10AAAAAA
010000 - 10FFFF	000DDDDC CCCCBBBB BBAAAAAA	11110DDD 10CCCCC 10BBBBBB 10AAAAAA

Tablica 2.6: Kodiranje pomoću UTF-8

UTF-16 je vrsta kodiranja znakova varijabilne dužine od 2 ili 4 bajta, pri čemu se praktički većina korištenih znakova može prikazati korištenjem samo dva bajta. Znakovi razine BMP mogu se prikazati uporabom samo dva bajta, a tek za znakove iz drugih razina potrebno je koristiti četiri bajta. UTF-16 je definiran normom ISO/IEC 10646-1, Unicode 2.0 i višima te prema IETF-u prepoznat na Internetu¹⁴³. **UCS-2** ili dvobajtni UCS je prethodnik UTF-16, ali postoji samo u obliku dužine 2 bajta, što znači da može samo prikazati znakove razine BMP. UTF-16 je nativna interna reprezentacija tekstualnih dokumenata na platformi Microsoft Windows od Windowsa 2000 nadalje i Mac OS X, te u programskim jezicima, okruženjima i platformama Java i .NET.

Kod **kodiranja pomoću UTF-16** postupak je još jednostavniji nego kod UTF-8. Svi znakovi s vrijednostima kodne točke od 0000_{16} do $FFFF_{16}$, što odgovara razini BMP, prikazuju se identično kakvi jesu, odnosno vrijednost kodne točke samo se preslikava u 16 bita. Za vrijednosti kodnih točaka drugih razina iznad BMP, odnosno za vrijednosti od 10000_{16} do $10FFFF_{16}$ koristi se kodiranje s 4 bajta, odnosno tzv. parovima surogata (engl. *surrogate pairs*). Prvo se od vrijednosti kodne točke oduzme 10000_{16} kako bi se dobila vrijednost u 20 bitova, koja se zatim podijeli na dva dijela od 10 bitova. Uz gornjih 10 bitova pribroji se vrijednost $D800_{16}$ i to čini prva dva bajta zapisa, a uz donjih 10 bitova pribroji se vrijednost $DC00_{16}$ i to čini druga dva bajta zapisa, što sve zajedno čini 4 bajta. Provjera je opet jednostavna jer gornja dva bajta uvijek imaju vrijednosti od $D800_{16}$ do $DBFF_{16}$, a donja dva bajta imaju vrijednosti od $DC00_{16}$ do $DFFF_{16}$. Uzevši u obzir sve kombinacije, zaključuje se da postoji 1024 vrijednosti za gornja dva bajta i 1024 vrijednosti za donja dva bajta, što sve zajedno čini 1.048.576 jedinstvenih parova. Kako se vrijednosti kodnih točaka od $D800_{16}$ do $DFFF_{16}$ u normama

¹⁴³ Hoffman, P., UTF-16, an encoding of ISO 10646, RFC2781, Internet Engineering Task Force, <http://tools.ietf.org/html/rfc2781>

Unicode i ISO/IEC 10646 nikad neće koristiti za konkretne znakove, onda pojedini kod iz ovog para od po 2 bajta nikad samostalno ne predstavlja znak, pa se zna da je riječ o kodiranju pomoću UTF-16.

Jedan od problema koji uvode zapisi s više od jednog bajta je **poredak bajtova** (engl. *endianness*). Kao što već znate, može postojati oblik zapisa *Little Endian*, kada se prvo pohranjuje najmanje značajan bajt, i oblik zapisa *Big Endian*, kada se prvo pohranjuje najznačajniji bajt. Kako se u različitim sustavima bajtovi zapisuju na različite načine, rješenje se pronašlo uporabom oznake poretka bajtova **BOM** (engl. *Byte Order Mark*). BOM je zapravo znak U+0FEFF u normi Unicode koji signalizira poredak bajtova, a pojavljuje se na početku dokumenta ili zapisa. Ako se pojavi u poretku FEFF₁₆, onda se radi je riječ o zapisu s poretkom prvi najznačajniji bajt (engl. *Big Endian*), a ako se pojavi u poretku FFFE₁₆, onda je riječ o zapisu s poretkom prvi najmanje značajan bajt (engl. *Little Endian*). S obzirom na to da u načelu UTF-8 ne treba BOM, on dolazi do izražaja kod UTF-16 kada se prepoznaje znak U+FEFF, dok znak U+FFFE ni ne postoji. No, ako se koristi oznaka skupa znakova UTF-16BE ili UTF-16LE koje objašnjavaju poredak bajtova, onda se BOM ne smije koristiti. Inače, u tekstu se znak U+FEFF može interpretirati kao praznina bez prijeloma nulte duljine (engl. *zero width no-break space*).

UTF-32 je vrsta kodiranja znakova fiksne dužine od 4 bajta, odnosno 32 bita. S obzirom na svoju dužinu vrlo je prostorno, odnosno memorijski zahtjevan te zbog toga nije baš često korišten ni rasprostranjen. **UCS-4** je vrsta kodiranja korištenjem 31 bita sukladna UTF-32.

2.7.4 Norme za zapis znakova korištene u Hrvatskoj

Kao što smo već spomenuli, u hrvatskom se jeziku koriste grafemi, odnosno znakovi za određene glasove koji su specifični za hrvatski jezik. Tako hrvatska abeceda ima jednoznakovne grafeme s dijakritičkim znakom (jednoslovne) **č, ć, đ, š i ž** te grafeme koji se sastoje od dvaju znakova iako predstavljaju jedan glas, odnosno dvoslove ili digrafe **dž, nj i lj**. Prilikom stvaranja hrvatskih nacionalnih norma u Hrvatskoj su se preuzele postojeće norme ISO i JUS, pa su tako donesene norme **HRN I.B1.002:1982 Obrada podataka: Skup znakova za razmjenu podataka kodiranih sa 7 bitova** koja nasljeđuje zatečenu normu ISO/IEC 646 u inačici YU, odnosno JUS I.B1.002, često kolokvijalno nazvanu „**CROSCII**“, te **HRN I.B1.013:1988 Obrada podataka: Skup grafičkih znakova za razmjenu podataka kodiranih jednim bajtom za latinična pisma** koja nasljeđuje zatečenu normu ISO/IEC 8859-2, odnosno JUS I.B1.013.

No, povijesno i u praksi u Hrvatskoj su se tijekom godina koristile različite norme. Prvo su u upotrebi bile norma **ASCII i EBCDIC**, ali **bez hrvatskih grafema**, gdje su oni onda prikazivani slovima c, dj, s i z. Zatim je slijedila norma **HRN I.B1.002** (zvana i **CROSCII**, odnosno norma ISO/IEC 646 YU ili JUS I.B1.002). Nakon toga se s pojavom osobnih računala prvo koristila kodna stranica **CP437**, ali opet **bez hrvatskih grafema**, koju slijedi kodna stranica **CP852** koja ima hrvatske grafeme. Iako se u međuvremenu očekivao širi prihvrat norme **HRN I.B1.013** (zване i Latin 2, odnosno norma ISO/IEC 8859-2 ili JUS I.B1.013), Microsoftove kodne stranice Windows-1252 i inačica s hrvatskim grafemima **Windows-1250** počinju dominirati dokumentima na osobnim računalima. Uz navedene norme i kodne stranice postoji još niz drugih koji su u uporabi tako da je još prije nekoliko godina vladala velika zbrka oko načina zapisa podataka, koja se i danas nije u potpunosti riješila.

Sljedeća tablica prikazuje najčešće korištene skupove znakova, kodne stranice i norme koje su se koristile u Hrvatskoj, a katkad se, pogotovo kod starijih sustava, još danas mogu pronaći.

Znak	HRN I.B1.002 „CROSCII“ ISO-646-YU	EBCDIC Latin- 2 Multilingual	IBM CP852 „Latin II“	HRN I.B1.013 „Latin-2“ ISO-8859- 2	MS CP-1250 Windows-1250
č	7E	49	9F	E8	E8
Č	5E	69	AC	C8	C8
ć	7D	47	86	E6	E6
Ć	5D	67	8F	C6	C6
đ	7C	8C	D0	F0	F0
Đ	5C	AC	D1	D0	D0
š	7B	9C	E7	B9	9A
Š	5B	BC	E6	A9	8A
ž	60	B6	A7	BE	9E
Ž	40	B8	A6	AE	8E

Tablica 2.7: Kodovi za hrvatske grafeme (dijakritičke znakove) u skupovima znakova

Kao što se može primijetiti, različiti oblici zapisa hrvatskih grafema uzrokovali su mnogo problema, a uzrokuju ih i danas. Dokumenti su zapisivani u različitim kodnim stranicama, uporabom različitih normi, a isto tako su pohranjivani i u baze podataka. S obzirom na to da su različiti operacijski sustavi, baze podataka, platforme i aplikacije podržavale različite kodne stranice, **problemi kod interoperabilnosti na razini zapisa znakova** bili su značajni, pa se jedan dobar dio razmjene podataka zapravo odnosio na učestale konverzije znakova. No, često se događalo da su konverzije neprimjerene ili se čak jednostavno ne primjenjuju, pa su na zaslonu prikazivani različiti specijalni znakovi umjesto hrvatskih grafema.

Vjerojatno najizraženiji problem, osim konverzija između različitih normi, je **sortiranje** korištenjem uzlaznog poretka kodova. Ovo kod većine normi korištenih u Hrvatskoj očigledno nije moguće jednostavno implementirati jer poredak hrvatskih dijakritika u kodnoj stranici ne prati hrvatsku abecedu, a time ni poredak slova kod sortiranja. Da stvar bude još zanimljivija, nisu problematični samo hrvatski grafemi koji se sastoje od jednog znaka s dijakritikom č, ć, đ, š i ž, već i digrafi dž, lj i nj. Inače, nisu nužno dva znaka koja mogu biti digraf uvijek digraf, pa tako primjerice postoje riječi koje sadržavaju slova „n“ i „j“ slijedno i ona koja sadržavaju digraf „nj“, te se ona drukčije sortiraju. Digrafi se u većini kodnih stranica i normi zapisuju kao dva znaka te se kod sortiranja isto tako trebaju i koristiti. Dodatni je problem kod zapisivanja velikih i malih slova, pa tako računalno razlikujemo dž, Dž (veliko početno slovo) i DŽ (kod pisanja svim velikim slovima) jer su zapisani od dvaju znakova s dvama različitim kodovima. To kod sortiranja malih slova konkretno znači da:

- digraf „dž“ dolazi iza slova „d“, a ispred slova „đ“, odnosno između znakova „dz“ i „da“,
- digraf „lj“ dolazi iza slova „l“, a ispred slova „m“, odnosno između znakova „lz“ i „ma“,
- digraf „nj“ dolazi iza slova „n“, a ispred slova „o“, odnosno između znakova „nz“ i „oa“.



Jedna od najvećih nelogičnosti, a koja se mogla izbjeći, je bila djelomična nesukladnost raširene Microsoftove kodne stranice CP-1250 i norme HRN I.B1.013, odnosno kodne stranice ISO-8859-2. U kontekstu interoperabilnosti ovo je uzrokovalo niz problema prilikom razmjene podataka, pogotovo između informacijskih sustava javne uprave i raznih tvrtki, jer se dogodio stvarni sraz pravne *de iure* i tržišne *de facto* norme. Drugi problem je bio prikaz internetskih stranica koje su zapisivane u raznim kodiranjima, a preglednici su nepravilno tumačili oblike zapisa, što zbog pogrešaka programera i dizajnera stranica Weba, što zbog problema samih preglednika u automatskom prepoznavanju i interpretaciji stranica Weba zapisanih u određenim kodnim stranicama.

Stvar se ipak donekle počela razjašnjavati učestalim korištenjem norme **ISO/IEC 10646** i **Unicode**, odnosno oblika zapisa **UTF-8**, te povremeno i **UTF-16**. Ovdje su hrvatski grafemi jasno i jednoznačno zapisani, nema problema s nejasnoćama, UTF-8 i UTF-16 su široko prihvaćeni kod svih oblika interoperabilnosti te se preporučuje njihova uporaba.

Sljedeća tablica prikazuje kodove hrvatskih grafema, odnosno vrijednosti kodnih točaka i pripadajuće zapise u UTF-8 i UTF-16, te nazive entiteta korištenih u jezicima XML i HTML.

Znak	Vrijednost kodne točke	Zapis u UTF-8 (2 bajta)	Zapis u UTF-16 (2 bajta)	Engl. naziv	Kod entiteta	Znak entiteta
č	U+010D = 0100001101 ₂	11000100 10001101 ₂ = C48D ₁₆	00000001 00001101 ₂ = 010D ₁₆	Small C with caron	č	č
Č	U+010C = 0100001100 ₂	11000100 10001100 ₂ = C48C ₁₆	00000001 00001100 ₂ = 010C ₁₆	Capital C with caron	Č	Č
ć	U+0107 = 0100000111 ₂	11000100 10000111 ₂ = C487 ₁₆	00000001 00000111 ₂ = 0107 ₁₆	Small C with acule	ć	&caacute;
Ć	U+0106 = 0100000110 ₂	11000100 10000110 ₂ = C486 ₁₆	00000001 00000110 ₂ = 0106 ₁₆	Capital C with acule	Ć	&Caacute;
dž	U+01C6 = 0111000110 ₂	11000111 10000110 ₂ = C786 ₁₆	00000001 11000110 ₂ = 01C6 ₁₆	Small DZ with caron	ǆ	
Dž	U+01C5 = 0111000101 ₂	11000111 10000101 ₂ = C785 ₁₆	00000001 11000101 ₂ = 01C5 ₁₆	Capital D with small Z with caron	ǅ	
DŽ	U+01C4 = 0111000100 ₂	11000111 10000100 ₂ = C784 ₁₆	00000001 11000100 ₂ = 01C4 ₁₆	Capital DZ with caron	Ǆ	
d	U+0111 = 0100010001 ₂	11000100 10010001 ₂ = C491 ₁₆	00000001 00010001 ₂ = 0111 ₁₆	Small D with stroke	đ	đ
Đ	U+0110 = 0100010000 ₂	11000100 10010000 ₂ = C490 ₁₆	00000001 00010000 ₂ = 0110 ₁₆	Capital D with stroke	Đ	Đ
lj	U+01C9 = 0111001001 ₂	11000111 10001001 ₂ = C789 ₁₆	00000001 11001001 ₂ = 01C9 ₁₆	Small LJ with caron	ǉ	
Lj	U+01C8 = 0111001000 ₂	11000111 10001000 ₂ = C788 ₁₆	00000001 11001000 ₂ = 01C8 ₁₆	Capital L with small J with caron	ǈ	
LJ	U+01C7 = 0111001011 ₂	11000111 10001011 ₂ = C787 ₁₆	00000001 11001011 ₂ = 01C7 ₁₆	Capital LJ with caron	Ǉ	
nj	U+01CC = 0111001100 ₂	11000111 10001100 ₂ = C78C ₁₆	00000001 11001100 ₂ = 01CC ₁₆	Small NJ with caron	ǌ	
Nj	U+01CB = 0111001011 ₂	11000111 10001011 ₂ = C78B ₁₆	00000001 11001011 ₂ = 01CB ₁₆	Capital N with small J with caron	ǋ	
NJ	U+01CA = 0111001010 ₂	11000111 10001010 ₂ = C78A ₁₆	00000001 11001010 ₂ = 01CA ₁₆	Capital NJ with caron	Ǌ	
š	U+0161 = 0101100001 ₂	11000101 10100001 ₂ = C5A1 ₁₆	00000001 01100001 ₂ = 0161 ₁₆	Small S with caron	š	š
Š	U+0160 = 0101100000 ₂	11000101 10100000 ₂ = C5A0 ₁₆	00000001 01100000 ₂ = 0160 ₁₆	Capital S with caron	Š	Š
ž	U+017E = 0101111110 ₂	11000101 10111110 ₂ = C5BE ₁₆	00000001 01111110 ₂ = 017E ₁₆	Small Z with caron	ž	ž
Ž	U+017D = 0101111101 ₂	11000101 10111101 ₂ = C5BD ₁₆	00000001 01111101 ₂ = 017D ₁₆	Capital Z with caron	Ž	Ž

Tablica 2.8: Kodovi za hrvatske grafeme (dijakritičke znakove) u UTF-8, UTF-16 i kao entiteti

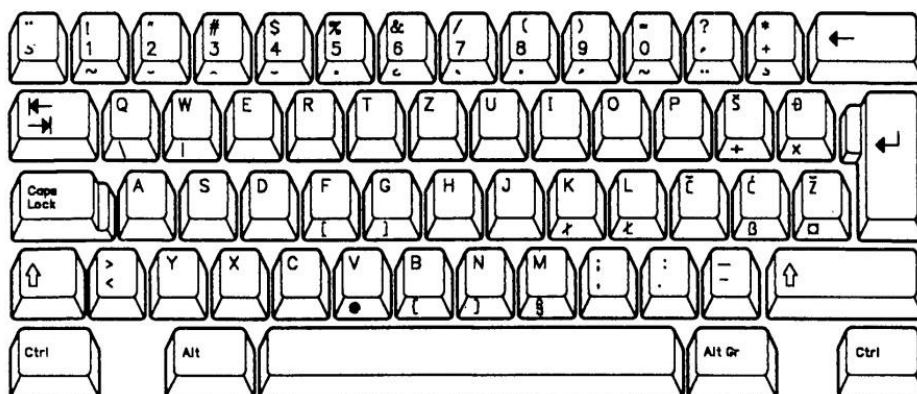
Inače, UTF-8 i UTF-16 dopuštaju i uporabu **digrafa** pomoću postupka sažimanja (engl. *contraction*), ali to nažalost nije često jer je otežan unos digrafa zato što zapravo ne postoji takva tipka na tipkovnici. Prepoznavanje na računalu bez uporabe automatskog rječnika je otežano jer, kao što je navedeno, nisu svi susjedni znakovi digrafi pa se ne zna kad je potrebno zapisati digraf, a kada dva odvojena znaka.

Prilikom prikaza na zaslonu prepoznavanje kodne stranice i prilagodba prikaza događa se automatski ili ručno, bez obzira na to je li riječ o stranici weba, poruci elektroničke pošte ili tekstualnoj datoteci pohranjenoj na disku. Ako postoji mogućnost, preporučuje se da se uvijek prikaz prilagodi automatski, no ako to nije moguće, u programu katkad još uvijek treba ručno odabrati koju kodnu stranicu koristiti.



Sa stajališta interoperabilnosti preporuka za zapis znakova je korištenje norme **ISO/IEC 10646**, odnosno Unicode i zapisa u obliku **UTF-8** ili **UTF-16**.

Inače, slična stvar, iako u dosta jednostavnijim razmjerima, događala se i u vezi s rasporedom tipaka s hrvatskim grafemima na tipkovnici. Tzv. „hrvatska tipkovnica“ zapravo nije nikad definirana, pa ni normirana, no načelno se prihvaća stari IBM-ov raspored prilagođen hrvatskom jeziku prema preporuci Hrvatske informatičke zajednice¹⁴⁴, kao na sljedećoj slici. S obzirom na to da na različitim tipkovnicama postoji različit razmještaj tipki, pogotovo onih koje imaju neko posebno značenje izvan standardne engleske abecede, tako i hrvatski grafemi još uvijek katkad završe na potpuno neočekivanim mjestima.



Slika 2.8: Preporuka rasporeda tipaka na tipkovnici za primjenu u Hrvatskoj

¹⁴⁴ Normirani raspored za hrvatsku tipkovnicu, Prijedlog norme za kodne sustave u Hrvatskoj, HIZ_norme_02.Doc, verzija 03.01., 1997-09-29, http://www.open.hr/hiz/kodsus/hiz_norme_301_42.pdf

3. Poglavlje: Označni jezici

U ovom poglavlju naučit ćete:

- ✓ osnove označnih jezika
- ✓ o označnom jeziku SGML
- ✓ o označnom jeziku XML u kontekstu interoperabilnosti
- ✓ osnove o jeziku YAML i JSON
- ✓ validaciju XML dokumenata korištenjem jezika RELAX NG

3.1 Uvod u označne jezike

Označavanje (engl. *markup*) je znakovni (ili binarni) niz koji se na određenim mjestima umeće u tekst i tako označava određeni dio teksta kako bismo ga razlikovali od ostatka. Iako termin *markup* u engleskom jeziku ima više različitih značenja, npr. u ekonomskom smislu znači iznos ili postotak prilikom prodaje ili povećanja cijene, ovdje se govori o značenju koje je u vezi s pisanjem tekstova. Povijesno je postojao dulji oblik „*marking up*“, što se odnosilo na označavanje rukopisa i tekstova dodatnim oznakama prilikom recenzija i revizija od strane lektora, urednika ili recenzenata, kako bi se u nekom tekstu označili određeni dijelovi, što se izvodilo dodavanjem pisanih oznaka u samom tekstu ili na margini pored teksta, često crvenom ili plavom tintom. Razlog označavanja je u lektorskom smislu najčešće bio u vezi s isticanjem pogrešaka, potrebom promjene nerazumljivih ili stilski neprilagođenih dijelova te uputa autoru u vezi s poboljšanjem, smanjenjem ili povećanjem pojedinog dijela teksta. U prezentacijskom smislu oznake su prilikom tiska objašnjavale grafičkom uredniku kako će se koji dio teksta otisnuti, u pogledu vrste slova, veličine slova, kurziva, masno otisnutog teksta, proreda, praznina i slično. U svim je slučajevima bila riječ o dodatnim uputama nad postojećim tekstom, kojima se osobi koja je provodila određeni postupak nad tekstom davalo upute da nešto treba preraditi, promijeniti, popraviti, doraditi, istaknuti ili prikazati. Isti se smisao zadržao i u računalnim označnim jezicima.

Jezik za označavanje ili **označni jezik** (engl. *markup language*) predstavlja sustav oznaka ili anotacija nad tekstom kojime se određene cjeline sintaktički odjeljuju od drugih. Označni jezik sadrži skup konvencija označavanja kojih se treba pridržavati kod označavanja teksta. Označni jezik zapravo definira koje su oznake dopuštene za korištenje, koje se oznake koriste za određene potrebe, kako se oznake razlikuju od ostatka teksta te mogu, ali i ne moraju, opisivati semantičko značenje oznaka, odnosno što svaka oznaka predstavlja.

U kontekstu interoperabilnosti označni su jezici vrlo važni jer omogućuju **razmjenu podataka između dviju strana u točno definiranom obliku koji obje strane razumiju**. Na taj je način omogućena komunikacija dviju strana, na razini prijenosa, ali i, što je važnije, na razini razumijevanja smisla prenesenih podataka, odnosno semantike.

Danas postoji niz različitih označnih jezika koji se koriste u različite svrhe. Ovdje su navedene neke najvažnije skupine označnih jezika, čije su konkretne izvedbe objašnjene kasnije u ovom poglavlju:

- **Označni jezici opće namjene** (engl. *general purpose markup languages*) su jezici koji služe za općeniti opis oznaka i kao osnova izgradnje drugih jezika. Najpoznatiji predstavnici su metaoznačni jezici GML, SGML, XML i YAML, ali postoji i niz manje poznatih poput jezika EBML i ASN.1.
- **Označni jezici dokumenata** (engl. *document markup languages*) su danas najrasprostranjeniji i najšire prihvaćeni označni jezici u svakodnevnom korištenju. Ovoj skupini pripadaju najpoznatiji označni jezici HTML i XHTML za prikaz stranica Weba, KML i KMZ, koji predstavljaju označni jezik za razmjenu geografskih informacija za korištenje s Google Earth programom, ali i drugi specifičniji predstavnici, kao što su označni jezici tekstualnih i drugih dokumenata ODF, OOXML, RTF, TeX, LaTeX, DocBook, troff, Wikitext, AsciiDoctor i niz drugih.

- **Označni jezici za sindikaciju sadržaja** (engl. *content syndication markup languages*) služe za objavljivanje dijelova internetskih stranica na drugim stranicama, a najčešće su zasnovani na označnom jeziku opće namjene XML.
- **Označni jezici za opis korisničkog sučelja** (engl. *user interface markup languages*) služe za opis dijelova korisničkog sučelja kao i razmještaja pojedinih komponenata, a najčešće se dijele prema platformama ili proizvođačima. Od najpoznatijih predstavnika mogu se navesti Adobeov označni jezik MXML, označni jezici FXML (koji se koristi u JavaFX-u) iz područja tehnologija Java, Microsoftov označni jezik XAML, Mozillin označni jezik XUL, označni jezik LZX platforme OpenLaszlo i mnogi drugi. Na neki način i označni jezici HTML i XHTML mogu pripadati i ovoj skupini jer također opisuju korisničko sučelje stranica Web.
- **Označni jezici za vektorsku grafiku** (engl. *vector graphics markup languages*) omogućavaju zapis vektorske grafike u dvama ili trima dimenzijama, a najpoznatiji predstavnici su označni jezici SVG i VRML, no čak i neki označni jezici za opis korisničkog sučelja, kao što je označni jezik XAML, mogli bi pripadati i ovoj kategoriji.
- **Označni jezici usluga Web i drugih udaljenih usluga** (engl. *Web service markup languages*) služe za opis različitih usluga dostupnih putem Interneta i na druge načine. Ovoj skupini pripadaju označni jezici WSDL, SOAP, WSCL i BPEL, ali i jezik XML-RPC.
- **Označni jezici ostalih specifičnih namjena** rješavaju probleme zapisa za specifične potrebe, kao što su označni jezik za zapis matematičkih jednažbi MathML, označni jezik za zapis matematičkih modela CellML, već opisani označni jezik za zapis informacija elektroničkog poslovanja ebXML, označni jezik za zapis geografskih značajki GML, označni jezik za zapis muzičke notacije MusicML, označni jezik za zapis objekata e-učenja SCORM, označni jezik za zapis multimedijских prezentacija SMIL, označni jezik za zapis glasovnih podataka VoiceXML i mnogi drugi.

Kao što se može vidjeti iz prethodne podjele, razlikuju se **konkretni označni jezici** koji se koriste za određene potrebe, kao što je npr. jezik HTML korišten za internetske stranice, od **metaoznačnih jezika opće namjene**, koji definiraju pravila, ali ne i značenje pojedinih oznaka, a čiji su najpoznatiji predstavnici jezici SGML i XML predstavljeni kasnije u ovom poglavlju.

Moguće je označne jezike podijeliti na one koji su **izvedeni iz jezika SGML**, one koji su **izvedeni iz jezika XML** i one koji **nisu izvedeni ni iz jednoga drugog jezika opće namjene**. Neki jezici po svojoj definiciji mogu pripadati i u više skupina pa ova podjela nije jednoznačna. Pretpostavlja se da se danas u svijetu koristi nekoliko stotina označnih jezika različite namjene te je ovo jedno od vrlo važnih oblika interoperabilnosti na globalnoj razini.

U općem smislu postoje **tri glavne vrste označnih jezika**:

- **Prezentacijski označni jezici** (engl. *presentational markup languages*) – predstavljaju označne jezike koji se najčešće koriste kod uređivača teksta, na način da se u dokument s tekстом umeću oznake, koje se kasnije koriste prilikom prikaza ili ispisa dokumenta. Takve se oznake prilikom pregleda teksta najčešće sakrivaju od pogleda korisnika, ali u pozadini oblikuju konačni izgled dokumenta kada se on prikazuje za

čitanje ili šalje pisaču na ispis, a autor teksta može ih uređivati. Uređivači teksta često koriste načelo WYSIWYG (engl. *What You See Is What You Get*), koje sakriva oznake i omogućava prikaz prilikom uređivanja (najčešće na zaslonu) koji načelno odgovara konačnom izgledu, što se može odnositi na tekstualne dokumente za ispis, stranice Weba, prezentacije i slično.

- **Proceduralni označni jezici** (engl. *procedural markup languages*) – jezici koji umetanjem oznaka upravljaju obradom tekstualnih podataka. Program koji obrađuje zadani tekst uočavanjem umetnutih oznaka pokreće određene akcije nad tekстом pa se ovu vrstu oznaka može prozvati i naredbama obrade (engl. *process instructions*). Oznake su najčešće vidljive, odnosno nisu sakrivene ni autoru prilikom izrade i promjene teksta niti drugim korisnicima koji pregledavaju dokument.
- **Opisni označni jezici** (engl. *descriptive markup languages*) – služe za označavanje dijelova teksta, odnosno dokumenta u strukturnom smislu i nisu izravno povezani ni s prezentacijom teksta korisniku niti s uputama za procesiranje. Određeni dijelovi teksta, od najniže razine slova ili riječi pa sve do rečenica, odlomaka ili poglavlja, mogu se označiti u semantičkom smislu, kako bi se objasnilo što oni predstavljaju, odnosno kako bi se umetnula dodatna informacija o tom dijelu teksta.

3.1.1 Razvoj označnih jezika

William W. Tunnicliff prvi spominje korištenje označnih jezika još 1967. godine pod nazivom „generičko kodiranje“¹⁴⁵, u obliku **razdvajanja definicije oblikovanja od strukture sadržaja** elektroničkih dokumenata. Tunnicliff je kasnije postao i prvi predsjednik Međunarodne organizacije za normizaciju ISO, a poznat je i kao voditelj razvoja norme GenCode. „Ocem označnih jezika“ smatra se ipak Charles F. Goldfarb, IBM-ov istraživač koji je 1969. godine vodio razvoj prvoga **općeg označnog jezika GML** (engl. *Generalized Markup Language*). Termin opći označni jezik GML stvorili su Charles Goldfarb, Edward Mosher i Raymond Lorie radeći zajedno na projektu sustava za upravljanjem i dijeljenjem dokumenata pravnih tvrtki, a njegov naziv je i akronim njihovih prezimena¹⁴⁶. Jezik GML objavljen je 1973. godine u IBM-ovom tehničkom izvještaju o izgradnji sustava za integrirano procesiranje teksta¹⁴⁷, gdje se spominje da „oznake trebaju biti korisne za više različitih aplikacija ili računalnih sustava“ te da „kod procesiranja teksta treba ignorirati uobičajene razlike izdavačkih aktivnosti i aktivnosti dohvata informacija“. Ovo je vjerojatno prva artikulacija ideja interoperabilnosti zasnovanih na zajedničkom označnom jeziku. Paralelno s jezikom GML razvijao se i jedan od prvih jezika koji imaju izraženu podjelu na strukturu i prezentaciju, a kojeg je 1980. godine predstavio Brian Reid u sklopu sustava za uređivanje dokumenata **Scribe**. Na osnovama označnog jezika GML i projekta GenCode kasnije je nastao označni jezik SGML, potanje opisan u potpoglavlju 3.2.

U razvoju označnih jezika značajnu ulogu ima i sustav za procesiranje dokumenata **troff** i njegov prethodnik, program za obradu teksta **nroff**, izvedeni na operacijskom sustavu Unix. Oba su nastala iz programa za obradu teksta **RUNOFF**, koji je poslije evoluirao u program nazvan **roff**. Sustav **troff** sadržava naredbe za definiranje znakovnih pisama, odlomaka,

¹⁴⁵ Goldfarb, Charles F., *The SGML Handbook*, Oxford university Press, USA, 1991

¹⁴⁶ Goldfarb, Charles F., *The Roots of SGML -- A Personal Recollection*, 1996., <http://www.sgmlsource.com/history/roots.htm>

¹⁴⁷ Goldfarb, Charles F., *Design Considerations for Integrated Text Processing Systems*, IBM Cambridge Scientific Center Technical Report G320-2094, 1973., <http://www.sgmlsource.com/history/G320-2094/G320-2094.htm>

marginama i drugih obilježja tekstualnog dokumenta, a najpoznatija primjena je u sustavu pomoći *man*.

Sustav za uređivanje teksta **TeX**, koji je 1978. godine razvio Donald Knuth, te program i prateći označni jezik dokumenata **LaTeX**, koji je 1980. godine razvio Leslie Lamport, imaju značajnu ulogu u razvoju drugih označnih jezika za tekstualne dokumente. Cilj stvaranja sustava TeX bio je da se omogući izvedba kvalitetnih tekstova i knjiga širim krugovima uz razumnu mjeru truda te da se proizvedeni dokument uvijek na isti način prikazuje na različitim računalima i ispisuje na različitim pisačima. Osnovna ideja označnog jezika LaTeX, koji je dodatno popularizirao sustav TeX, je da se autori teksta mogu koncentrirati na sadržaj koji pišu bez da ih ometa promišljanje o vizualnoj prezentaciji dokumenta. Tako se definiraju poglavlja, odlomci, tablice, slike i ostali elementi logičke strukture dokumenta koji se samo koriste tijekom pisanja, a kasnije se definira točan izgled i položaj pojedinog elementa. Ovo je usporedivo s načelom globalnih stilova kod uređivača teksta ili jezika CSS za stranice Weba. Danas je uporaba sustava i jezika TeX i LaTeX široko prihvaćena u akademskoj zajednici, osobito u matematici, fizici, elektrotehnici, računalnoj znanosti i računalnom inženjerstvu, čemu je pridonijela i korištenje besplatno.

Najpoznatiji primjer prezentacijskog označnog jezika je jezik **HTML** (engl. *Hypertext Markup Language*), jednostavni označni jezik dominantan u ulozi jezika za izradu stranica Weba, a koji ste već upoznali na drugim kolegijima. Jezik HTML također je bitan u kontekstu interoperabilnosti jer omogućava prikaz na raznorodnim uređajima, platformama, operacijskim sustavima i programskoj podršci, odnosno preglednicima. Bitno je uočiti kako je i jezik HTML izvorno nastao kao primjena jezika SGML.

3.2 Označni jezik SGML

Kako se potreba za standardiziranim generičkim označnim jezikom pojavljuje već polovicom prošlog stoljeća, u 60-im se godinama u računalnim sustavima spominju prva korištenja oznaka, kao što je npr. oznaka „naslov“ umjesto samo strojno razumljivih oznaka poput „format-17“. Kao što je već spomenuti navedeno u potpoglavlju 3.1.1, u IBM-u je početkom 70-ih godina razvijen jezik GML, koji je kasnije široko prihvaćen u IBM-ovim izdavačkim aktivnostima. Daljnja istraživanja struktura dokumenata, povezivanja procesa i tipova dokumenata dovode do novoga označnog jezika kasnije nazvanog **SGML** (engl. *Standard Generalized Markup Language*).

Jezik SGML razvio je Američki nacionalni institut za norme ANSI (poglavlje 2.4.8.) kako bi se normirao opis tekstualnih dokumenata zasnovan na jeziku GML. Iako je razvoj započeo 1978. godine, norma je dovršena tek 1983. godine, kada su je prihvatile i državne institucije u SAD-u. Međunarodna organizacija za normizaciju ISO prepoznaje 1986. godine jezik SGML kao normu pod oznakom **ISO 8879:1986**. Prvi se je put šire primjenjivala za projekt elektroničkog rukopisa (engl. *Electronic Manuscript Project*) Udruge američkih izdavača (engl. *Association of American Publishers*) i dokumentacijski dio projekta računalno potpomognute podrške akviziciji i logistici (engl. *Computer-aided Acquisition and Logistic Support*) Ministarstva obrane SAD-a.

Pojednostavljeno, jezik SGML može se definirati kao međunarodna norma za predstavljanje tekstualnih informacija u elektroničkom obliku, neovisno o uređaju ili sustavu. Jezik SGML nije konkretna inačica označnog jezika za opis podataka iz neke domene, već služi kao **metajezik**, odnosno **način formalnog opisa označnog jezika**. Osnovni cilj jezika SGML je definicija oznaka i shema pisanja, koji se jedinstveno nazivaju označavanjem (engl. *markup*). U engleskom se jeziku često koristi i termin *encoding*, koji označava načine eksplicitne interpretacije teksta.

SGML kao norma ISO nosi puni naziv „ISO 8879:1986 Information processing -- Text and office systems -- Standard Generalized Markup Language (SGML)“ i ima tri inačice:

- izvorni SGML iz 1986. godine s manjim tehničkim ispravcima,
- SGML (ENR) iz 1996. godine s proširenim pravilima imenovanja (engl. *Extended Naming Rules*) koji dopuštaju oznake proizvoljnih jezika,
- SGML (ENR+WWW ili WebSGML) iz 1998. godine s boljom podrškom za jezik XML i potrebe Web-a.

Svaki SGML dokument može imati tri dijela:

- deklaraciju SGML,
- **prolog** koji sadržava deklaraciju DOCTYPE s različitim deklaracijama oznaka zajedno s definicijom tipa dokumenta DTD (*Document Type Definition*),
- samu **instanciju** koja sadržava vršni element i njegov sadržaj.

Primjene ili tzv. **aplikacije** (engl. *applications*) jezika SGML su konkretni označni jezici razvijeni za određene potrebe. Neke primjene prepoznate su mnogo prije nego što je došlo do razvoja Web-a i najpoznatijih jezika HTML i XML, a neke su se razvile baš iz najpoznatijeg označnog jezika XML. Primjene koje su se razvile iz jezika XML kao podskupa jezika SGML su neizravne primjene jezika SGML. Mnogi jezici koji su se prvotno razvili kao primjena jezika SGML danas postoje u „moderniziranoj“ izvedbi kao primjene jezika XML, jer su tijekom godina evoluirali i koriste mogućnosti interoperabilnosti jezika XML, koji je potanje opisan u sljedećem potpoglavlju. Lista postojećih označnih jezika¹⁴⁸ je prevelika kako bi je ovdje prikazali, a i novi se jezici svakodnevno stvaraju.

3.3 Označni jezik XML

O osnovama jezika XML već ste čuli u priručniku „Standardi u primjeni internetske tehnologije“, a ovdje ćemo ponoviti neke osnove i proširiti ih s novim potankostima. Podsjetimo se, čuli ste da XML:

- odvaja podatke od prezentacije,
- omogućuje dijeljenje podataka,
- pojednostavljuje prijenos podataka,
- pojednostavljuje izmjene platformi,
- čini podatke dostupnima.

¹⁴⁸ List of Markup Languages, Wikipedia, http://en.wikipedia.org/wiki/List_of_markup_languages

Ako potanje promotrimo ove uloge jezika XML, uočiti ćemo pojmove „dijeljenje“, „prijenos“, „izmjene“ i „dostupnost“. U kontekstu interoperabilnosti ove značajke jezika XML, uz odgovarajući komunikacijski, odnosno transportni mehanizam, čine ga idealnim kandidatom za zapis podataka i izvedbu tehničke i sintaksne interoperabilnosti. No s obzirom na to da jezik XML sadržava i metapodatke koji opisuju sadržaj, može se iskoristiti i na višoj razini semantičke interoperabilnosti, gdje će dvije strane koje imaju istu primjenu jezika XML moći ostvariti međusobnu razmjenu i korištenje podataka. No krenimo redom, od normi.

Extensible Markup Language (XML) inačice 1.0 opisan je u preporuci W3C-a¹⁴⁹ kao podskup jezika SGML koji se koristi na Webu („*lakoćom jezika HTML*“), no dizajniran s velikom jednostavnošću izvedbe i interoperabilnosti s jezicima SGML i HTML. Osim inačice 1.0 jezika XML postoji i preporuka W3C-a¹⁵⁰ za inačicu XML 1.1, koja donosi neke novosti, kao što su: korištenje znakova definiranih normom Unicode koji dosad nisu bili dopušteni,

- manje strogu definiciju naziva („*sve što nije zabranjeno je dopušteno*“),
- dodavanje znakova koji omogućavaju interoperabilnost *mainframe* i drugih sustava,
- korištenje kontrolnih znakova (npr. s početka tablice znakova),
- skup ograničenja nazvan „potpuna normalizacija“ (engl. *full normalization*), kojih se treba pridržavati prilikom izrade dokumenata i koje trebaju provjeravati procesori dokumenata,
- oznaku nove inačice u zaglavlju, odnosno deklaraciji XML dokumenta kako bi procesori mogli razlikovati koja pravila koriste pri provjeri.

Ciljevi koje jezik XML treba zadovoljiti jasno su definirani već kod njegovog nastajanja, a to su:

- Jezik XML treba se moći jasno i neposredno koristiti na internetu.
- Jezik XML treba podržavati mnoštvo različitih aplikacija (primjena).
- Jezik XML treba biti kompatibilan s jezikom SGML.
- Pisanje programa koji procesiraju XML dokumente treba biti jednostavno.
- Broj opcionalnih značajki u jeziku XML treba biti što manji, idealno bi bilo da je bez njih.
- XML dokumenti trebaju ljudima biti čitljivi i jasni.
- Oblikovanje XML-a treba biti brzo izvedeno.
- Oblikovanje XML-a treba biti formalno i sažeto.
- XML dokumenti trebaju se moći jednostavno izraditi.
- Sažeto i jasno u XML oznakama je od minimalnog značenja.

Već ste upoznati i sa stablastom strukturom XML dokumenta, sintaksnim pravilima i pravilima imenovanja te elementima i atributima. Osim toga, upoznati ste i s objektnim modelom DOM i metodama nad njima, objektom XMLHttpRequest, DTD-om i prostorima imena.

¹⁴⁹ Extensible Markup Language (XML) 1.0 (Fifth Edition), W3C Recommendation 26 November 2008, <http://www.w3.org/TR/xml/>

¹⁵⁰ Extensible Markup Language (XML) 1.1 (Second Edition), W3C Recommendation 16 August 2006, edited in place 29 September 2006, <http://www.w3.org/TR/xml11/>

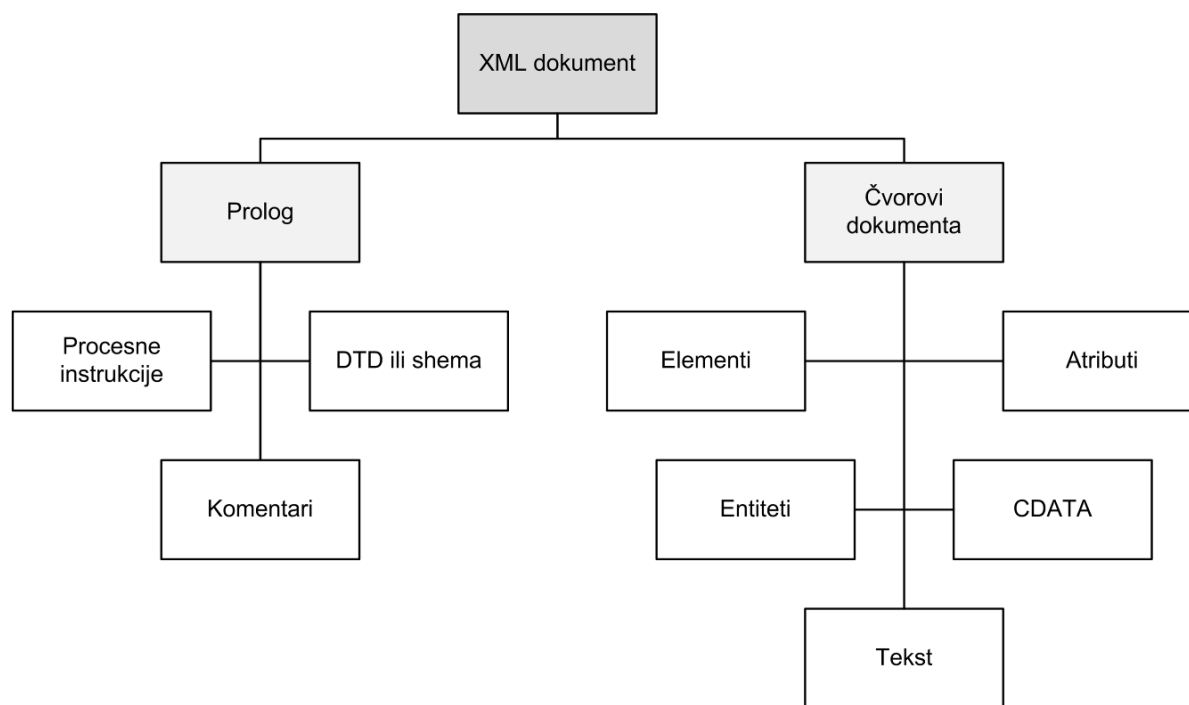
Stoga ćemo ukratko ponoviti osnovne značajke i pojmove:

- **znak** (engl. *character*) – osnovni građevni element, jer se XML dokument sastoji od skupa znakova, a smije se koristiti većina znakova iz norme Unicode, konkretno znakovi #x9, #xA, #xD, #x20 do #xD7FF, #xE000 do #xFFFD i #x10000 do #x10FFFF,
- **oznaka** (engl. *tag*) – svi znakovni nizovi (engl. *markup*) koji započinju znakom „<“ i završavaju znakom „>“, postoje u tri oblika:
 - **početna oznaka** (engl. *start-tag*) – označava početak svake oznake u obliku <oznaka>,
 - **završna oznaka** (engl. *end-tag*) – označava završetak svake oznake u obliku </oznaka>,
 - **prazni element** (engl. *empty-element*) – označava prazni element u obliku <oznaka/>,
- **referenca entiteta** (engl. *entity reference*) – upućuje na imenovani entitet, predefiniran ili eksplicitno deklariran
 - **referenca znakovnog entiteta** (engl. *character entity reference*) – upućuje na znakovni entitet, odnosno znak iz korištenog skupa znakova UCS,
 - **referenca predefiniranog entiteta** (engl. *predefined entity reference*) – upućuje na predefinirani entitet, a to su specijalni znakovi < (<), > (>), & (&), ' (') i " ("),
 - **referenca imenovanog entiteta** (engl. *named entity reference*) – upućuje na entitet deklariran u DTD ili XML shemi,
- **sadržaj** (engl. *content*) – sadržaj je zapravo sve što nije oznaka ili ne započinje znakom „&“ a završava znakom „>“, vrijednost podatka u tekstualnom obliku
- **element i atribut** – bit će objašnjeni u potpoglavlju 3.3.2.

XML dokumenti su datoteke koje imaju određeni sadržaj u jeziku XML, zapisan prema pravilima jezika XML. No XML dokument ne mora nužno označavati fizički jedinstveni dokument zapisan u jednoj datoteci, već se pod XML dokumentom može smatrati i tok podataka koji prati pravila zapisa u jeziku XML.

3.3.1 Struktura XML dokumenta

XML dokument sastoji se od dvaju dijelova: prologa i čvorova dokumenata (Slika 3.1).



Slika 3.1: Struktura XML dokumenta

Prolog XML dokumenta sadržava metainformacije o dokumentu, uključujući XML deklaraciju, procesne instrukcije, komentare i ugrađeni DTD ili shemu, te nije obavezan.

Većina XML dokumenata započinje s **XML deklaracijom**, koja govori o kojoj je inačici norme jezika XML u dokumentu riječ, vrsti kodiranja znakova odnosno teksta te je li dokument samostalan, kako bi aplikacije koje obrađuju XML dokument znale osnovne informacije o njemu. XML deklaracija započinje znakovima `<?xml`, a završava znakovima `?>`.

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

XML deklaracija je neobavezna, ali ako postoji, mora se nalaziti odmah u prologu, na prvom mjestu u dokumentu, bez ikakvih znakova prije nje, uključujući razmake i prazne retke (što može stvoriti probleme procesorima XML-a).

Napomena: Iako je neobavezna, preporučuje se korištenje XML deklaracije jer definira vrstu kodiranja znakova i druge značajke dokumenta.

Sve informacije u XML deklaraciji pružaju se u obliku atributa u točno definiranom poretku:

- atribut `version` definira inačicu XML-a, moguće vrijednosti su 1.0 ili 1.1,
- opcionalni atribut `encoding` definira vrstu kodiranja znakova, vrijednost se određuje prema nazivima kodiranja znakova IANA-CHARSET¹⁵¹, a najčešće korištene vrijednosti su UTF-8, UTF-16, windows-1250, windows-1252, ISO-8859-1 i ISO-8859-2,

¹⁵¹ Internet Assigned Numbers Authority (IANA), Official Names for Character Sets, ur. Keld Simonsen i drugi, <http://www.iana.org/assignments/character-sets>

- opcionalni atribut **standalone** označava ovisnost dokumenta o vanjskim entitetima, moguće vrijednosti su **yes** i **no**, a ako se atribut ne navede, pretpostavljena vrijednost je **no**.

Napomena: Iako je trenutačna aktualna inačica XML-a 1.1, preporučuje se korištenje inačice 1.0, koju sigurno razumiju i starije aplikacije.

Procesne instrukcije ili **naredbe obrade** proslijeđuju informaciju o XML dokumentu aplikacijama koje ga obrađuju. U kontekstu interoperabilnosti ovo je važno svojstvo jer aplikacijama daje dodatne upute kako obraditi XML dokument. Procesne instrukcije također započinju znakovima `<?xml`, a završavaju znakovima `?>`. Procesne se instrukcije najčešće pojavljuju u prologu zato što imaju smisao uputa koje se daju prije obrade dokumenta, ali moguće je, iako ne i previše vjerojatno, njihovo pojavljivanje dublje u dokumentu. Primjer procesne instrukcije je referenca na XSL listu stilova (engl. *XSL stylesheet*).

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

Komentari se mogu koristiti bilo gdje u dokumentu. Komentari započinju znakovima `<!--`, a završavaju znakovima `-->`. Aplikacije i procesori koji obrađuju XML dokument zanemaruju komentare. Tipičan komentar izgleda ovako:

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

No postoji i nekoliko pravila za komentare, pa tako komentari ne mogu sadržavati tekst `„-->“`, ne smiju biti ubačeni usred naziva oznake i ne smiju sadržavati znakove `„<“` ili `„>“`, inače ih primatelj ili aplikacija za obradu neće pravilno protumačiti.

Zadnji dio prologa čini **DTD** ili **XML shema**, koji opisuju pravila kojih se prilikom pisanja elemenata i atributa u XML dokumentu treba pridržavati.

Nakon prologa slijedi ostatak dokumenta ili **sadržaj dokumenta**, koji se sastoji od barem jednog elementa. Taj element nazivamo **korijenskim elementom** ili **elementom dokumenta** i on može sadržavati druge elemente, attribute i ostali sadržaj (komentare, sadržaj, procesne instrukcije ...), ali mora postojati isključivo jedan. Osim po tom pravilu, korijenski se element ne razlikuje od bilo kojeg drugog elementa.

3.3.2 Elementi i atributi

Unutar XML dokumenta **elementi** služe za razne namjene, od kojih su najvažnije označavanje sadržaja, opis sadržaja koji označavaju, informacije o poretku sadržaja i odnosi između podataka. Elementi mogu sadržavati tekst, druge elemente i komentare, a početna oznaka elementa može sadržavati i attribute. Sadržajem elementa definira se sve između početne i završne oznake elementa, uključujući attribute navedene u početnoj oznaci.

Postoji nekoliko oblika elementa:

- **prazni elementi** (engl. *empty element*) ili **elementi bez sadržaja** – ne sadržavaju ni tekst ni druge elemente, služe samo kao strukturne oznake, mogu se prikazati kao

<oznaka></oznaka> ili <oznaka/>, pri čemu drugi način štedi memoriju i povećava čitljivost,

- **elementi jednostavnog sadržaja** ili elementi koji sadržavaju samo tekst – sadržavaju tekst bez specijalnih znakova, prikazani kao <oznaka>Neki sadržaj</oznaka>,
- **elementi koji sadržavaju druge elemente** – sadržavaju samo druge elemente, ali ne i sadržaj (iako drugi elementi sadržani u njima mogu imati sadržaj),
- **elementi koji sadržavaju miješani sadržaj i elemente** – sadržavaju kombinaciju sadržaja i drugih elemenata.

Imenovanje elemenata prate određena pravila, pa tako nazivi elemenata moraju započeti slovom ili znakom „_“, iza kojeg mogu slijediti druga slova, znamenke ili neki specijalni znakovi. No nazivi ne smiju započeti slovima „xml“ niti sadržavati praznine. Nazivi se odabiru na temelju značenja elemenata, a ako je riječ o zapisu koji će se potencijalno koristiti u međunarodnoj komunikaciji, korisno je koristiti engleske nazive ili nazive u jeziku za koji se pretpostavlja da će sve strane moći jednostavno interpretirati. Preporučuje se korištenje kraćih naziva, ali opet dovoljno dugih da se razumije značenje. Također je preporučuje uporaba kombinacija velikih i malih slova ili znaka „_“ za odvajanje riječi u nazivu, dok se uporaba lokalnih grafema izvan standardne engleske abecede, znakova „-“ i „.“ ne preporučuje zbog potencijalne zamjene sa znakovima minus i operatorom između razreda i atributa.

Napomena: Nazivi elemenata nedvosmisleno predstavljaju značenje. U međunarodnoj komunikaciji preporučuje se korištenje engleskih naziva. Nije preporučljivo znatno kratiti, ali ni produljivati nazive. Za odvajanje riječi u nazivu preporučuje se koristiti velika početna slova riječi ili znak „_“. Ne preporučuje se korištenje znakova izvan engleske abecede niti znakova „-“ i „.“.

Atributi se uvijek navode u početnoj oznaci elementa. U načelu atributi pružaju dodatnu informaciju o elementu unutar kojeg se navode. Ne postoji ograničenje o brojnosti atributa unutar jednog elementa, ali se u istom elementu ne mogu pojaviti dva atributa istog naziva. Atributi se prikazuju kao par naziva atributa i sadržaja, odnosno vrijednosti atributa, pri čemu se između stavlja znak „=“, a vrijednost atributa mora se navesti unutar znakova navodnika, jednostrukih ili dvostrukih.

Najčešća uporaba atributa je za prikaz određenog oblikovanja podatka ili kao oznaka korištenja nekog specifičnog formata ili kodiranja, pa tako primjerice možemo oblikovati datum:

```
<date format="dd.mm.yyyy.">20.09.2010.</datum>
<date format="ISO8601">2010-09-20</datum>
```

Imenovanje atributa u načelu slijedi ista pravila kao i imenovanje elemenata. Postoji i prazni atribut, kada se između navodnika ne navodi nikakav sadržaj.

Napomena: Kod umetanja koda iz raznih izvora, poput dokumenata izrađenih uređivačem teksta ili izreži-umetni (engl. *copy-paste*) metodom, moguće je da ćete umjesto dvostrukih navodnika (") umetnuti tzv. „pametne navodnike“ (engl. *smart quotes*), odnosno znakove „“ i „“, koji u XML dokumentu predstavljaju pogrešku.

Već znate da ne postoji opće pravilo kada koristiti atribute, a kada elemente. No postoje neka ograničenja atributa, koja se svode na to da su atributi vezani uz pojedini element, uvijek sadržavaju samo tekst, nemaju podstrukturu, ne mogu se proširivati i ne mogu sadržavati višestruke vrijednosti.

Tekst u XML dokumentu je zapravo znakovni niz, odnosno sadržaj između dviju oznaka. Ako se ne označi kao vrsta sadržaja CDATA, XML parser procesirat će ga kao i sav drugi XML kod, stoga treba paziti na ograničenja uporabe posebnih znakova, poput oznake otvaranja „<“. Sve posebne znakove treba zapisati kao entitete.

Posebni oblik teksta je **tekst koji se ne parsira**, a označava se oznakom **CDATA**. Ovo se može koristiti ako se u XML dokument umeće neki programski kod ili skripta koju se ne smije parsirati ili jednostavno podaci koje ne želimo parsirati.

```
<tekst><!CDATA[ Ako je 2009 < 2010 onda je to dobro. ]></tekst>
```

Znakovni entiteti su simboli od jednog znaka koje možemo koristiti u XML dokumentu. Znakovni se entiteti zapisuju kao znak „&“, kojeg slijedi naziv entiteta i završava s „““. Ako se želi predstaviti znak koji nije moguće utipkati tipkovnicom, ali postoji u stranici kodova, onda se umjesto naziva može koristiti znak „#“, koji slijedi Unicode kod ili znak „x“ i heksadekadski kod znaka iz stranice kodova. Najočitiiji primjer su znakovi < (<), > (>), & (&), ' (') i “ ("), a postoji i niz drugih primjera, kao što su drugi specijalni znakovi, npr. znak za zaštitu prava autorstva © (© ili ©)

```
<tekst>Ovo će prikazati znak manje &lt; a ovo copyright &#169; </tekst>
```

3.3.3 Pravila pisanja u jeziku XML

Netočno je kada netko kaže da XML dokument nije dobro oblikovan. Svi XML dokumenti su **dobro oblikovani** (engl. *well-formed*), što znači da udovoljavaju određenim sintaksnim pravilima. Ako dokument nije dobro oblikovan, onda on zapravo ni nije XML dokument (iako bi to htio postati).

Ima mnogo pravila dobre oblikovanosti, a ovdje navodimo samo najvažnija:

- samo jedan korijenski element,
- elementi moraju biti pravilno zatvoreni,
- elementi moraju biti pravilno gniježđeni,
- smiju se koristiti samo dopušteni znakovi,
- nazivi elemenata osjetljivi su na mala i velika slova,
- vrijednosti atributa moraju se postaviti u navodnike,

3.3.4 Domene i prostori imena

Iako ste se s **prostorima imena u jeziku XML** (engl. *namespace in XML*) već susreli u kolegiju „Standardi u primjeni internetske tehnologije“, ponovit ćemo ukratko njihovu svrhu. Mogućnost korištenja istih naziva u jednom XML dokumentu ili zajedničkoj XML strukturi i izbjegavanje konflikata imena glavni je razlog uvođenja prefiksa prostora imena u nazive elemenata. No

svrha korištenja prostora imena zasniva se na **korištenju domena**, odnosno **jedinstveno imenovanih elemenata i atributa iz nekog određenog skupa vrijednosti**, najčešće **definiranog konkretnom primjenom**. Prostori imena u jeziku XML su i preporuka W3C-a¹⁵². Iako informacija o prostorima imena u XML dokumentu dodatno povećava veličinu dokumenta te opterećuje obradu u smislu potrebnih provjera prostora imena, ona je nužna ako se žele koristiti prednosti prostora imena.

Prostori imena mogu se koristiti u sljedećim slučajevima i za navedene svrhe:

- ponovno korištenje postojećih shema – mnoge već izvedene sheme za razne oblike mogu se ponovno koristiti u drugim zapisima; primjeri su grafičke, matematičke, tekstualne, metapodatkovne i poslovne sheme,
- proširenja postojećih shema – dio izvedenih shema oblikovan je da koristi proširenja koje korisnik dodaje ovisno o svojim potrebama,
- ponovno korištenje programskog koda za obradu postojećih shema – ako je razvijen određen programski kod koji obrađuje strukture, odnosno elemente i attribute prema nekoj shemi, on se može koristiti i u drugim aplikacijama,
- kombiniranje proizvoljnih elemenata i elemenata procesiranja – postojanje sadržajnih i procesnih struktura u istom XML dokumentu omogućuje istovremeno korištenje obrade i sadržajnim mogućnosti; primjeri su SOAP i XSLT,
- podrška verzijama – korištenje novog prostora imena kod pojave nove verzije sheme;
- pretraživanja temeljena na odabiru shema – jezici kao što je XQuery mogu koristiti pretraživanje koje je svjesno sheme,
- validacija (provjera) temeljena na višestrukim shemama – provjere valjanosti mogu se izvoditi u skladu sa shemama koje prate određene podstrukture u XML dokumentu.

Kao što je navedeno, korištenje prostora imena ima svoju svrhu i kod validacije XML dokumenata kod korištenja višestrukih shema. Ako se validacija izvodi metodom koja ne podržava prostore imena, onda je i njihovo korištenje, barem u validacijskom smislu, nepotrebno. No ako se za postupke validacije koriste metode zasnovane na XML shemama koje su svjesne prostora imena, onda se validacija može izvoditi u skladu s kvalificiranim nazivima u XML strukturi. Za validacije kod kojih se koriste višestruke sheme moguće je dijeljenje osnovne strukture na podstrukture (podstabla), koje sadržavaju samo elemente pojedinog prostora imena. Tada se zasebno mogu validirati elementi u svakom pojedinom prostoru imena. Ovakav proces opisan je u specifikaciji **jezika validacija temeljenih na prostorima imena NVDL** (engl. *Namespace-based Validation Dispatching Language*), koji je i dio jezika definicija shema dokumenata DSDL, opisanog u potpoglavlju 3.4.2. Jezik NVDL omogućava validacije prema višestrukim shemama jer ima funkcionalnosti odabira elemenata u određenom prostoru imena i pokretanja validacija prema pripadajućim shemama. No o validacijama ćemo potanje u sljedećem potpoglavlju.

¹⁵² Namespaces in XML 1.0 (Third Edition), W3C Recommendation, 2009-12-08, <http://www.w3.org/TR/xml-names/>

3.4 YAML

YAML je jezik koji se često smatra označnim jezikom, međutim on to nije. Sam akronim koji predstavlja naziv je nastao od isticanja da YAML nije označni jezik (engl. *YAML Ain't Markup Language*).¹⁵³ U trenutku pisanja priručnika je aktualna inačica bila 1.2 objavljena 2009. godine.

Vrlo često se koristi u sprezi s drugim programskim jezicima i koristi principe uvedene u druge tehnologije. Na primjer, skalarne nizove, liste i asocijativna polja je preuzeo iz Perla, odvojnika (engl. *separator*) je temeljen na MIME proširenju (oznaka s tri crtice, "---"), „escape“ sekvence su temeljene na programskom jeziku C, a omotavanje (engl. *wrapping*) praznina se temelji na HTML jeziku.¹⁵⁴ Najprepoznatljivija značajka YAML jezika je u tome što se strukturiranje postiže umetanjem praznina (engl. *indentation*) umjesto korištenja raznih „delimitera“. Po pitanju sigurnosnih značajki, YAML se može sadržavati izvršne naredbe, čime se spriječeno nenadzirano pokretanje raznih malicioznih programa.

3.4.1 Primjeri korištenja YAML-a

Za označavanje skalarnih vrijednosti ili varijabli koristi se sljedeći format s dvotočjem i prazninom:

```
integer: 25
string: "25"
float: 25.0
boolean: Yes
```

Asocijativna polja i liste mogu biti definirane korištenjem blokova definicija ili „*inline*“ format, slično kao što to definira i JSON format:

```
--- # Lista programskih jezika u formatu bloka definicija
- C
- C++
- C#
- Java
--- # Lista programskih jezika u „inline“ formatu
[C, C++, C#, Java]
```

Nizove znakova (stringove) je moguće označiti s znakovima „|“ i „>“. Znak „|“ čuva mjesta gdje su kreirane nove linije, a znak „>“ ih ignorira:

```
podaci: |
Svaka od ovih
linija
Biti će odvojena

podaci: >
Ovaj tekst je
Razvojen po linijama
I biti će spojen u jedan odlomak
```

¹⁵³ Službena YAML Web stranica, <https://yaml.org/>

¹⁵⁴ Informacije o jeziku YAML, <https://blog.stackpath.com/yaml/>

Vrlo popularna namjena YAML jezika je kod korištenja alata Liquibase i kreiranja „migracija“ nad bazama podataka kako bi se definirane promjene koje se moraju dinamički izvršiti prilikom pokretanja aplikacija na web poslužiteljima.¹⁵⁵ Velika prednost korištenja Liquibase biblioteke je u tome što programeri ne moraju „ručno“ održavati svoju instancu baze podataka i izvršavati SQL upite na njima, već se sve promjene automatski izvršavaju temeljene na „migracijskim“ datotekama koje opisuju te promjene. Primjer jedne takve „migracije“ u YAML formatu je prikazana u nastavku:

```
databaseChangeLog:
  - preConditions:
    - runningAs:
        username: liquibase

  - changeSet:
      id: 1
      author: nvoxland
      changes:
        - createTable:
            tableName: person
            columns:
              - column:
                  name: id
                  type: int
                  autoIncrement: true
                  constraints:
                    primaryKey: true
                    nullable: false
              - column:
                  name: firstname
                  type: varchar(50)
              - column:
                  name: lastname
                  type: varchar(50)
                  constraints:
                    nullable: false
              - column:
                  name: state
                  type: char(2)

  - changeSet:
      id: 2
      author: nvoxland
      changes:
        - addColumn:
            tableName: person
            columns:
              - column:
                  name: username
                  type: varchar(8)

  - changeSet:
      id: 3
      author: nvoxland
```

¹⁵⁵ Liquibase migracija u YAML formatu, https://www.liquibase.org/documentation/yaml_format.html

```
changes:
  - addLookupTable:
      existingTableName: person
      existingColumnName: state
      newTableName: state
      newColumnName: id
      newColumnDataType: char(2)
```

Navedena migracija se prijavljuje na bazu podataka korištenjem korisničkog imena „liquibase“, kreira tablicu „person“ koja ima kolone „id“, „firstname“, „lastname“ i „state“, dodaje kolonu „username“ u tablicu „person“ te dodaje tablicu „state“ koja predstavlja šifarnik statusa koji su opisani s dva slova.

3.5 JSON

Iako se ne radi o označnom jeziku, zbog važnosti i utjecaja JSON-a kao formata podataka koji se koristi kod različitih integracija i komunikacija sustava nije mogao biti izostavljen. Popularizacijom REST API sučelja, koja prvenstveno omogućavaju interoperabilnost poslužiteljskog dijela aplikacije s klijentskim dijelovima, naglo je porasla popularnost podatkovnog formata koji je bio lakši za čitanje od XML-a i sadržavao manje nepotrebno ponavljajućih dijelova koji su samo definirali strukturu elementa. Taj novi podatkovni format za razmjenu zove se JSON (engl. *JavaScript Object Notation*). Vrlo je pogodan za korištenje kod pretvorbe objekata iz objektno orijentiranih programskih jezika i ima vrlo široku podršku u većini relevantnih tehnologija na tržištu. Temelji se na podskupu JavaScript standarda ECMA-262, 3rd edition.¹⁵⁶

JSON je tekstualni format neovisan tehnologiji pa je moguće kombinirati npr. poslužiteljski dio koji je napisan u C# programskom jeziku s klijentskim dijelovima sustava koji mogu biti Angular aplikacija, iOS mobilna aplikacija ili JavaFX *desktop* aplikacija.

JSON je izgrađen korištenjem dviju struktura koje su univerzalne i korištene u raznim programskim jezicima i tehnologijama:

- zbirka parova „ključ-vrijednost“ (slično kao što je organizirana u zbirka iz skupine mapa u Javi ili rječnik u C#-u).
- poredana lista vrijednosti (slično strukturi liste ili polja u raznim programskim jezicima).

3.5.1 Primjeri korištenja JSON-a

U nastavku je prikazan primjer JSON strukture koja predstavlja rječnik termina, s jednim terminom „SGML“:

```
{
  "rjecnik" : {
    "naslov" : "primjer rjecnika",
    "RjecnikDiv" : {
      "naslov" : "S",
      "RjecnikList" : {
```

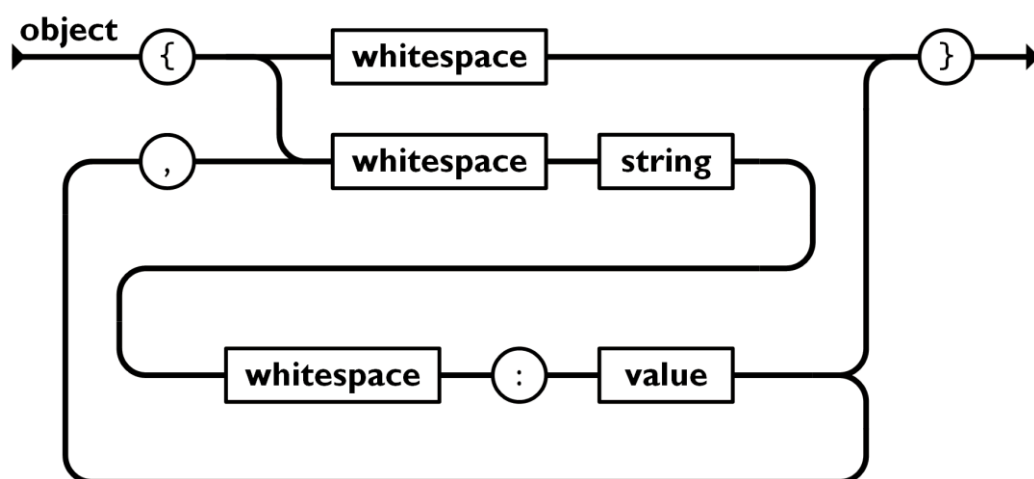
¹⁵⁶ Uvod u JSON, <https://www.json.org/json-en.html>

```

    "RjecnikElement" : {
      "ID" : "SGML",
      "Sortiranje" : "SGML",
      "RjecnikTermin" : "Standard Generalized Markup Language",
      "Akronim" : "SGML",
      "Standard" : "ISO 8879:1986",
      "RjecnikDefinicija" : {
        "definicija" : "Meta označni jezik koji služi za označavanje
                        drugih označnih jezika kao što je DocBook.",
        "Reference" : ["GML", "XML"]
      },
      "DodatniTermini" : "markup"
    }
  }
}

```

Prikazana JSON struktura predstavlja *object* strukturu, neporedani skup parova „ime-vrijednost“ koja počinje s vitičastim zagradama, nakon svakog naziva je dvotočka, a parovi „ime-vrijednost“ su odvojeni zarezom. *Whitespace* element označava prazninu, znak za novi red ili tabulator, što će biti opisano kasnije, kao i *value* i *string* strukture. Na sljedećem dijagramu opisan je način na koji se sastavlja *object* struktura u JSON-u:



157

Slika 1. Object struktura u JSON-u

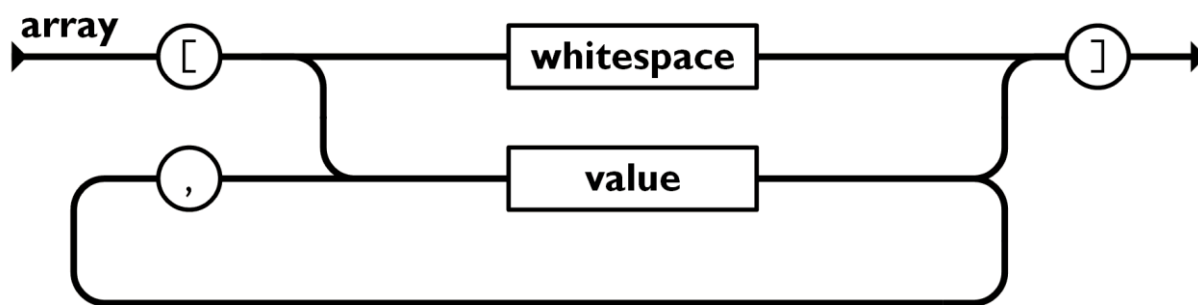
Array struktura u JSON-u predstavlja zbirku vrijednosti. Počinje s uglatim zagradama, a vrijednosti su odvojene zarezima. Na primjer, u sljedećem primjeru je prikazano polje naziva programskih jezika:

```

{
  "ime" : "Petar",
  "starost" : 25,
  "programskiJezici" : [ "Java", "C#", "JavaScript" ]
}

```

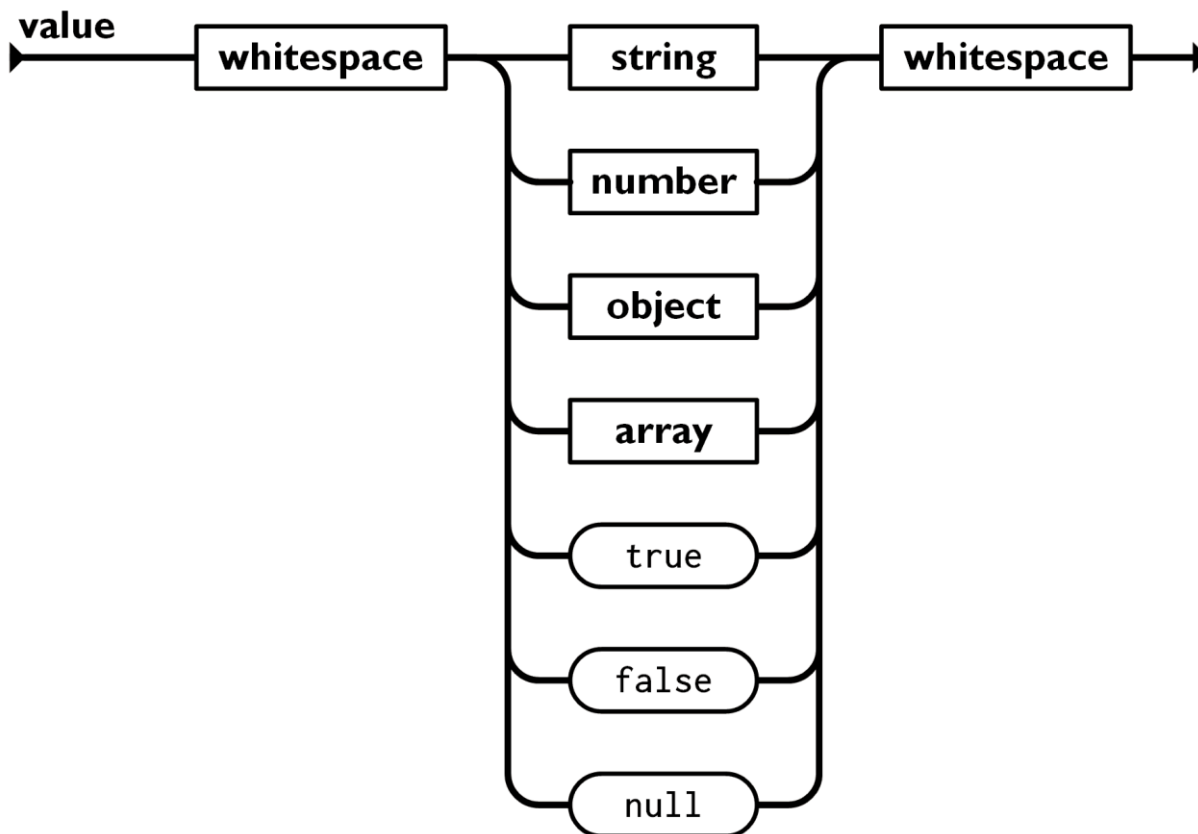
Sljedeći dijagram opisuje način na koji se može kreirati *array* struktura:



158

Slika 2. Array struktura u JSON-u

Value struktura u JSON-u predstavlja bilo kakav niz znakova, brojeva, **true** ili **false** vrijednost, kao i **null** unutar dvostrukih navodnika. Strukture mogu biti ugnježdjivane pa je moguće dobiti hijerarhijsku strukturu proizvoljne dubine. Na sljedećoj slici je prikazan dijagram koji prikazuje mogućnosti kreiranja *value* elemenata:



Slika 3. Value struktura u JSON-u¹⁵⁹

¹⁵⁸ Preuzeto iz <https://www.json.org/json-en.html>

¹⁵⁹ Preuzeto iz <https://www.json.org/json-en.html>

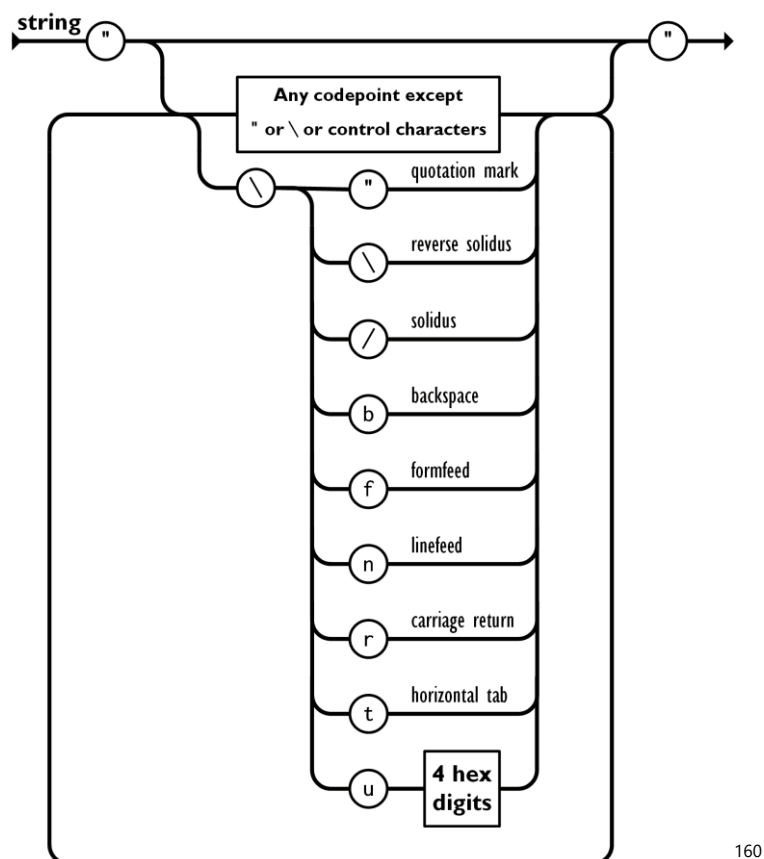
Primjer *value* strukture s tri različite vrijednosti izgleda ovako:

```
{ "ime": "Ivan", "starost": 14, "programskiJezici": null }
```

String struktura predstavlja sekvencu nula ili više Unicode znaka označenih s *backslash escape* znakom koje se nalaze unutar dvostrukih navodnika. Primjer takve *string* strukture prikazan je u nastavku:

```
{ "programeri": [ { "ime": "\u017Darko", "starost": 27, "programskiJezici": "Python" } ] }
```

Vrijednost ime je postavljeno na „Žarko“, međutim zbog potrebe korištenja Unicode notacije, umjesto „Ž“ nalazi se znak „\u017D“. Može sadržavati bilo koji znak koji predstavlja početak oznake (engl. *Any codepoint except " or \ or control character*) kao što je Unicode, ali bez dvostrukih navodnika (engl. *quotation mark*), *backslash* znaka (engl. *reverse solidus*) ili kontrolnih znakova. Osim toga je moguće koristiti znakove *slash* (engl. *solidus*), *backspace*, *formfeed*, *linefeed* i *carriage return* koji se koriste za označavanje novog prelaska u novi redak, horizontalni tabulator (engl. *tab*) te na kraju četveroznamenkaste vrijednosti u heksadecimalnom obliku (engl. *4 hex digits*). Općenita struktura *string* elementa prikazana je u nastavku:

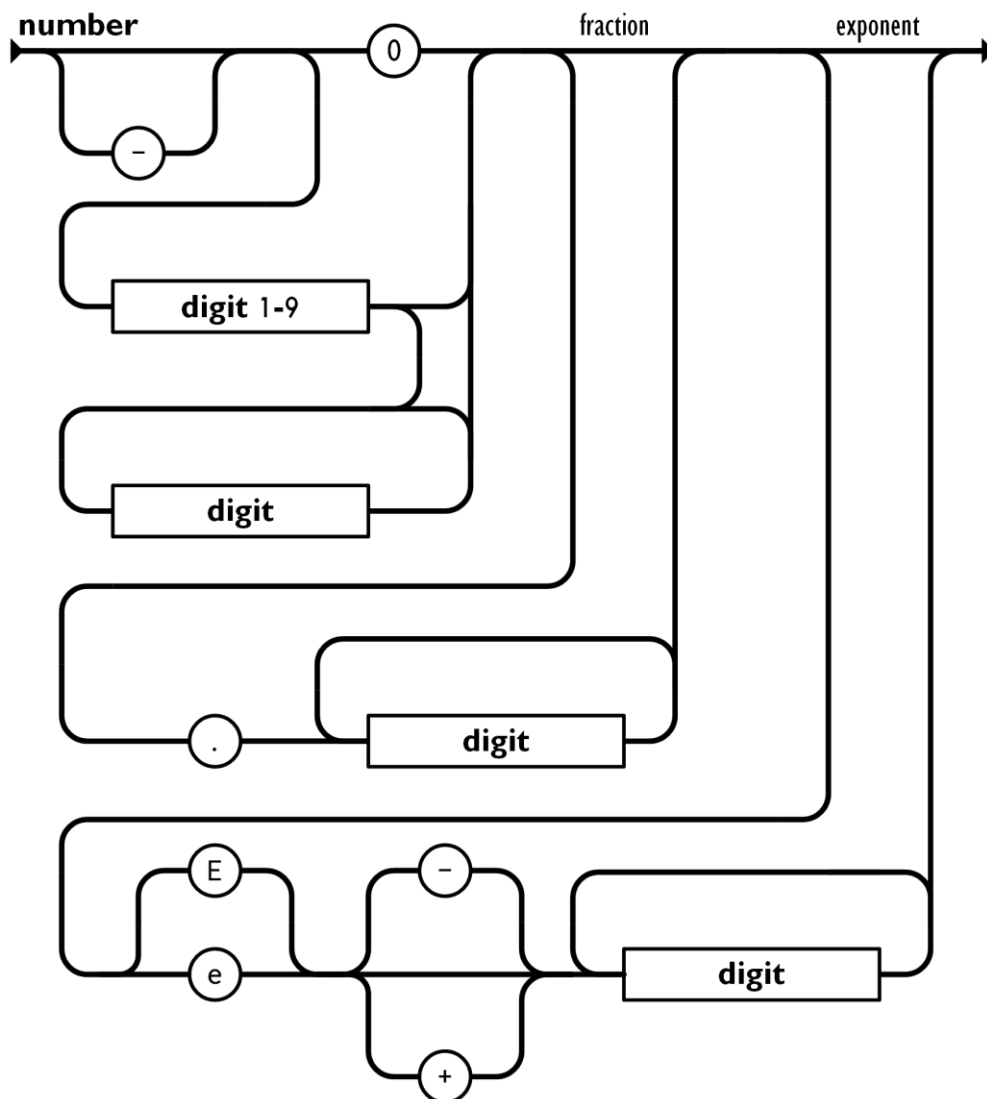


160

Slika 4. String struktura u JSON-u

¹⁶⁰ Preuzeto iz <https://www.json.org/json-en.html>

Brojevne strukture su u JSON-u vrlo slične strukturama koje označavaju brojčane vrijednosti u Java ili C programskom jeziku, samo što nije moguće koristiti vrijednosti prikazane u oktalnom ili heksadecimalnom brojevnom sustavu. Primjer postavljanja jednostavne brojčane vrijednosti prikazan je u prethodnim primjerima u kojima se postavlja parametar „starost“. Osim jednostavnih cijelih brojeva sa jednom ili više znamenki (engl. *digit*), moguće je definirati negativne brojeve, brojeve s eksponentom (engl. *exponent*) i brojeve s decimalnim mjestima (engl. *fraction*), kao što je prikazano na sljedećem dijagramu:

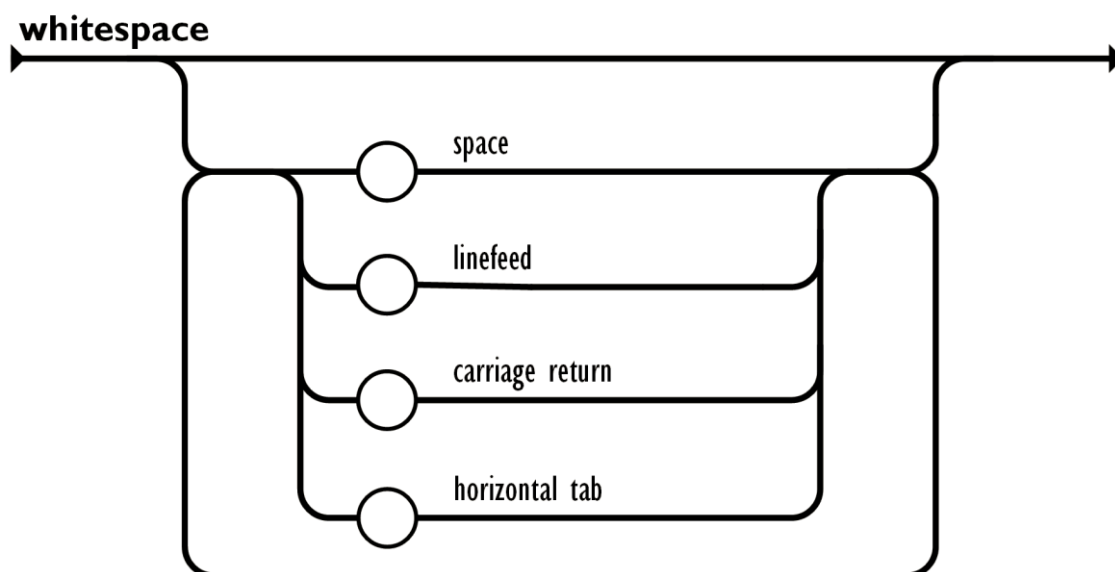


161

Slika 5. Number struktura u JSON-u

Whitespace znak predstavlja znak koji se koristi kako bi se strukturirao JSON zapis i može se umetnuti između bilo kojeg para tokena u samom zapisu. Može biti praznina (engl. *space*), oznake koje označavaju novi red (engl. *linefeed and carriage return*) i horizontalni tabulator (engl. *tab*). U nastavku je prikazan dijagram s pravilima korištenja *Whitespace* elementa:

¹⁶¹ Preuzeto iz <https://www.json.org/json-en.html>



162

Slika 6. Whitespace struktura u JSON-u

3.6 Validacije dokumenata

3.6.1 XML dokumenata

Validacija XML dokumenata i općenito struktura podataka u obliku XML (engl. *XML validation*) je proces provjere je li dokument „dobro oblikovan“ (engl. *well-formed*), odnosno je li uopće XML dokument, a zatim je li „valjan“ ili „validan“ (engl. *valid*). S obzirom na to da se „dobra oblikovanost“ može lako provjeriti, u ovom potpoglavlju bavit ćemo se provjerom valjanosti, pa stoga pod validacijom možemo podrazumijevati provjeru u odnosu na neki opis koji nam specificira uvjete valjanosti dokumenta. Takva provjera ili validacija se temelji na DTD-u i XSD-u. No postoji niz različitih načina i metoda provjere valjanosti koji se zasnivaju na gramatikama, pravilima, izrazima i tvrdnjama, a koje ćemo opisati u nastavku ovog poglavlja.

3.6.2 Document Type Definition (DTD) i XML Schema Definition (XSD)

DTD (engl. *Document Type Definition*) sintaksa, koja služi kao skup deklaracija oznaka koje definiraju tipove dokumenata za označne jezike kao što je XML. Tako se može definirati struktura dokumenta, tipovi podataka i obveznost elemenata, a u ovom poglavlju nećemo potanje ulaziti u problematiku izrade i sintaksu DTD-a, uz pretpostavku da je to obrađeno na navedenom kolegiju. Danas se napušta korištenje DTD-a u korist jezika shema koji su svjesni prostora imena, kao što su XSD i RELAX NG. No, razvoj DTD-a nije stao, već se izrađuje nova

¹⁶² Preuzeto iz <https://www.json.org/json-en.html>

verzija koja će moći koristiti prostore imena kao 9. dio norme DSDL¹⁶³, opisane u sljedećem potpoglavlju.

Napomena: Umjesto korištenja DTD-a preporučuje se korištenje jezika svjesnih prostora imena, XML Schema (XSD) i RELAX NG.

U navedenom je priručniku opisana i uporaba XML sheme **XSD** (engl. *XML Schema Definition*). Odmah na početku nailazimo na nedosljednost u nazivu jer W3C-a preporučuje naziv „XML Schema“, koji se koristi kao općeniti naziv za sve XML sheme, uključujući i XSD, dok se radi razumijevanja većinom koristi kratica XSD izvedena iz naziva XML Schema Definition. Dio zajednice koji želi naglasiti da je riječ o W3C-ovoj XML Schemi za XSD koristi i kraticu WXS.

Napomena: Preporučuje se korištenje naziva XSD (XML Schema Definition) ili alternativno WXS umjesto naziva XML Schema, koji može označavati opći naziv skupa različitih XML shema.

Na primjer, ako je potrebno kreirati XSD dokument za koje vrijede sljedeći kriteriji:

- I. ugraditi u **validaciju** prema XSD uz attribute
`<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">`
- II. element **igrac** je kompleksni tip
- III. element **osobni_podaci** je kompleksni tip sa točno jednim elementom **ime**, jednim elementom **prezime**, jednim elementom **visina**, jednim elementom **tezina** i jednim elementom **datumrod**
- IV. element **klub** je jednostavni tip s nazivom kluba za koji igrač igra
- V. element **drzava** je jednostavni tip s nazivom države za koju igrač nastupa
- VI. element **datumrod** je tipa date
- VII. element **visina** je tipa integer koji je između 140 i 220 sa atributom **jedinica**
- VIII. element **tezina** je tipa integer vrijednosti koji je između 60 i 90 sa atributom **jedinica**
- IX. element **pozicija** je tip koji može sadržavati samo sljedeće enumerirane vrijednosti: **vratar**, **obrambeni**, **vezni** i **napadac**
- X. element **igrac** može sadržavati samo točno jedan element **osobni_podaci**, samo jedan element **klub**, jedan ili više elemenata **drzava** i jedan ili više elemenata **pozicija**

on bi izgledao ovako:

¹⁶³ Namespace and datatype declaration in Document Type Definitions, Datatype- and namespace-aware DTDs, <http://www.itscj.ipsj.or.jp/sc34/open/0898.pdf>

```

<?xml version="1.0" encoding="utf-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

  <xs:simpleType name="tezinaRes">
    <xs:restriction base="xs:integer">
      <xs:minExclusive value="60"/>
      <xs:maxExclusive value="90"/>
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="visinaRes">
    <xs:restriction base="xs:integer">
      <xs:minExclusive value="140"/>
      <xs:maxExclusive value="220"/>
    </xs:restriction>
  </xs:simpleType>

  <xs:element name="igrac">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="osobni_podaci" maxOccurs="1" >
          <xs:complexType>
            <xs:sequence>
              <xs:element name="ime" type="xs:string" maxOccurs="1"/>
              <xs:element name="prezime" type="xs:string" maxOccurs="1"/>
              <xs:element name="visina" maxOccurs="1">
                <xs:complexType>
                  <xs:simpleContent>
                    <xs:extension base="visinaRes">
                      <xs:attribute name="jedinica" type="xs:string" />
                    </xs:extension>
                  </xs:simpleContent>
                </xs:complexType>
              </xs:element>
              <xs:element name="tezina" maxOccurs="1">
                <xs:complexType>
                  <xs:simpleContent>
                    <xs:extension base="tezinaRes">
                      <xs:attribute name="jedinica" type="xs:string"/>
                    </xs:extension>
                  </xs:simpleContent>
                </xs:complexType>
              </xs:element>
              <xs:element name="datumrod" type="xs:date" maxOccurs="1"/>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
        <xs:element name="klub" type="xs:string" maxOccurs="1"/>
        <xs:element name="drzava" type="xs:string" minOccurs="1"
          maxOccurs="unbounded" />
        <xs:element name="pozicija" minOccurs="1" maxOccurs="unbounded" >
          <xs:simpleType>
            <xs:restriction base="xs:string">
              <xs:enumeration value="vratar"/>
              <xs:enumeration value="obrambeni"/>
              <xs:enumeration value="vezni"/>
              <xs:enumeration value="napadac"/>
            </xs:restriction>
          </xs:simpleType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

```

```

    </xs:element>
  </xs:sequence>
</xs:complexType>
</xs:element>
</xs:schema>

```

Odgovarajući XML dokument koji sadržava podatke u skladu s navedenim validacijama bi izgledao ovako:

```

<?xml version="1.0" encoding="utf-8"?>
<igrac>
  <osobni_podaci>
    <ime>Ante</ime>
    <prezime>Rebic</prezime>
    <visina_jedinica="cm">185</visina>
    <tezina_jedinica="kg">77</tezina>
    <datumrod>1993-09-21</datumrod>
  </osobni_podaci>
  <klub>Milan</klub>
  <drzava>Italija</drzava>
  <pozicija>napadac</pozicija>
</igrac>

```

3.6.3 Jezik definicije shema dokumenata DSDL

Osnovna uloga **jezika definicije shema dokumenata DSDL**¹⁶⁴ (engl. *Document Schema Definition Language*) je prikupljanje različitih zadataka i izraza vezanih uz validacije u jedinstvenu cjelinu skupa specifikacija i opisa nadogradivog radnog okvira koji različitim tehnologijama omogućava da rade zajedno, paralelno ili serijski, kako bi proizvele jedinstveni validacijski rezultat. DSDL je zapravo radni okvir koji omogućava višestruke validacijske zadatke različitih tipova, koji se istovremeno mogu primijeniti na XML dokument kako bi ukupna validacija bila potpunija od pojedinačnih izvedenih uporabom samo jedne metode ili tehnologije. Primjer takve kombinirane validacije u paralelnom smislu bio bi istovremena primjena dviju ili više različitih validacija, npr. istovremenim korištenjem XSD-a i metoda opisanih u sljedećim potpoglavljima, kao što su RELAX NG i Schematron, a čijim se ukupnim validacijskim rezultatima na kraju postupka može upravljati. U serijskom smislu, validacije se mogu izvoditi slijedno u različitim metodama, kada jedna validacija slijedi drugu, a rezultati prethodne mogu se koristiti i kao ulazni parametri u sljedeću metodu.

Jezik DSDL je međunarodna norma **ISO/IEC 19757**¹⁶⁵, a izvedena je od više dijelova, odnosno skupova specifikacija koje opisuju strukture dokumenata, tipove podataka, odnos među podacima, validacijska pravila i skup sintaksi za deklaraciju pravila. Sastoji se od sljedećih dijelova:

¹⁶⁴ Document Schema Definition Language (DSDL), <http://www.dsdl.org>

¹⁶⁵ 174 ISO/IEC 19757, <http://www.iso.org/iso/search.htm?qt=19757&searchSubmit=Search&sort=rel&type=simple&published=on>

- 1. dio – **pregled** (engl. *Overview*) – sadržava pregled uloga preostalih dijelova norme I opće informacije,
- 2. dio – **validacije temeljene na regularnim gramatikama** (engl. *Regular-grammar-based Validation*) – sadržava opis jezika shema koji validiraju strukturu i sadržaj podataka u stablastom modelu korištenjem gramatika; trenutno temeljen na specifikaciji **RELAX NG**,
- 3. dio – **validacije temeljene na pravilima** (engl. *Rule-based Validation*) – sadržava opis jezika validacije ispitivanjem pravila; trenutno temeljen na specifikaciji **Schematron**,
- 4. dio – **validacije temeljene na prostorima imena NVDL** (engl. *Namespace-based Dispatching Language Validation*) – sadržava jezike za odabir elemenata u određenim prostorima imena i validaciju prema pripadajućim shemama, kao i skupove validacijskih pravila koje se dijele među aplikacijama,
- 5. dio – **proširivi podatkovni tipovi** (engl. *Extensible Datatypes*) – sadržava sintaksu za izvedbu korisnički definiranih tipova podataka, parsiranje i validaciju, definiciju skupova dopuštenih vrijednosti, ponavljanja, odabira, ograničenja, uvjeta, međuodnosa i poretka podatkovnih tipova; trenutno temeljen na **jeziku biblioteke podatkovnih tipova DTLL** (engl. *Datatype Library Language*)
- 6. dio – **ograničenja integriteta temeljena na putu** (engl. *Path-based integrity constraints*) – sadržava pravila identifikacije čvorova i odnosa elemenata; trenutno temeljen na **XPath 2.0**,
- 7. dio – **jezik deklaracije repertoara znakova – CRDL ili CREPDL** (engl. *Character Repertoire Description Language*) – sadržava sintaksu definiranja podskupova i imenovanja znakovnih skupova koji se koriste kod validacije; trenutno temeljen na ISO/IEC 10646 **UCS** znakovnom skupu,
- 8. dio – **jezik promjene naziva sheme ili semantike dokumenta DSRL** (engl. *Document Schema Renaming Language* ili *Document Semantics Renaming Language*) – sadržava mehanizme gdje se nazivi elemenata istih značenja mogu zamijeniti kako bi bolje odgovarali lokalno korištenim nomenklaturama,
- 9. dio – **deklaracija prostora imena i podatkovnih tipova u DTD (svjesnost podatkovnih tipova i prostora imena u DTD)** (engl. *Namespace and datatype declaration in Document Type Definitions (Datatype- and namespace-aware DTDs)*) – sadržava proširivu sintaksu za korištenje prostora imena i podatkovnih tipova; trenutno temeljen na jeziku SGML i DTD-u primijenjenim na jezik XML,
- 10. dio – **upravljanje validacijom** (engl. *Validation Management*) – sadržava jezike za orkestriranje validacijom i pratećim procesima, razvoj je prekinut,
- 11. dio – **pripajanje shema** (engl. *Schema Association*) – sadržava mogućnosti pripajanja različitih shema dokumentu; trenutno temeljen na *Associating Schemas with XML documents 1.0*.

Nadalje ćemo objasniti validacije uporabom regularnih gramatika predstavljene specifikacijom RELAX NG i validacije temeljene na pravilima predstavljene specifikacijom Schematron.

3.6.4 Jezik regularnih izraza RELAX NG

Kratika **RELAX NG**¹⁶⁶ označava jezik regularnih izraza za XML nove generacije (engl. *REgular LAnguage for XML Next Generation*) koji pripada skupini jezika XML shema. Izveden je objedinjavanjem jezika RELAX (engl. *Regular Language description for XML*) i TREX (engl. *Tree Regular Expressions for XML*). Osnovne značajke jezika RELAX NG uključuju jednostavnost, dva oblika sintakse, podršku za prostore imena, uniformnost postupanja s atributima i elementima, podršku za nepobrojani i miješani sadržaj te mogućnost kombiniranja s drugim jezicima. **Specifikacija jezika RELAX NG**¹⁶⁷ izrađena je od organizacije OASIS još 2001. godine, a dvije godine kasnije postao je i međunarodna norma **ISO/IEC 19757-2:2003**¹⁶⁸.

Schema RELAX NG koristi uzorke strukture XML dokumenata i smatra se prilično jednostavnijom u odnosu na druge jezike XML shema. Zapis sheme RELAX NG postoji u dvama oblicima:

- u osnovnom obliku u zapisu jezika XML, uobičajene ekstenzije datoteka „rng“,
- u kompaktnom obliku, koji nije u obliku XML, nazvan **RELAX NG Compact Syntax**¹⁶⁹, uobičajene ekstenzije datoteka „rnc“.

Korištenje jezika RELAX NG u osnovnom obliku zapisom u jeziku XML te u kompaktnoj sintaksi prikazat ćemo na sljedećem primjeru. Pretpostavimo da imamo sljedeći XML dokument:

```
<imenik>
  <osoba oib="12345678901">
    <ime>Ana</ime>
    <prezime>Anić</prezime>
    <email>ana.anic@racunarstvo.hr</email>
  </osoba>
  <osoba oib="23456789012">
    <cijeloIme>Ivo I. Ivić mlađi</cijeloIme>
  </osoba>
</imenik>
```

DTD za takav dokument s barem jednom osobom u imeniku, razdvojenim ili spojenim imenom i prezimenom te neobaveznom adresom elektroničke pošte izgledao bi ovako:

```
<!DOCTYPE imenik [
<!ELEMENT imenik (osoba+)>
<!ELEMENT osoba (((ime, prezime) | cijeloIme), email?)>
<!ATTLIST osoba oib CDATA #REQUIRED >
<!ELEMENT ime (#PCDATA)>
<!ELEMENT prezime (#PCDATA)>
<!ELEMENT cijeloIme (#PCDATA)>
<!ELEMENT email (#PCDATA)>
]>
```

¹⁶⁶ RELAX NG, <http://www.relaxng.org>

¹⁶⁷ RELAX NG Specification, OASIS Committee Specification, <http://www.relaxng.org/spec-20011203.html>

¹⁶⁸ ISO/IEC 19757-2:2003 Document Schema Definition Language (DSDL) - Part 2: Regular-grammar-based validation - RELAX NG, [http://standards.iso.org/ittf/PubliclyAvailableStandards/c037605_ISO_IEC_19757-2_2003\(E\).zip](http://standards.iso.org/ittf/PubliclyAvailableStandards/c037605_ISO_IEC_19757-2_2003(E).zip)

¹⁶⁹ RELAX NG Compact Syntax, OASIS Committee Specification, <http://www.relaxng.org/compact-20021121.html>

Uzorak iste gramatike zapisan u jeziku RELAX NG izveo bi se na sljedeći način:

```
<?xml version="1.0" encoding="UTF-8"?>
<element name="imenik" xmlns="http://relaxng.org/ns/structure/1.0">
  <oneOrMore>
    <element name="osoba">
      <attribute name="oib">
        <text/>
      </attribute>
      <choice>
        <group>
          <element name="ime">
            <text/>
          </element>
          <element name="prezime">
            <text/>
          </element>
        </group>
        <element name="cijeloIme">
          <text/>
        </element>
      </choice>
    </optional>
    <optional>
      <element name="email">
        <text/>
      </element>
    </optional>
  </element>
</oneOrMore>
</element>
```

Isti zapis u obliku RELAX NG izveo bi se na sljedeći način:

```
element imenik {
  element osoba {
    attribute oib { text },
    ( ( element ime { text },
      element prezime { text } ) | element cijeloIme
      { text } ), element email { text }?
  }+
}
```

Razvoj jezika RELAX NG događao se vremenski paralelno s razvojem XSD-a. Danas je XSD nešto rasprostranjeniji te je podržan od većine parsera i uređivača XML dokumenata, no i RELAX NG je rasprostranjen, pogotovo kod softvera prezentacijskih označnih jezika, kao što su OpenDocument Format i DocBook.

Napomena: Korištenje jezika RELAX NG preporučuje se u jednakoj mjeri kao i XSD.

3.6.5 Jezik tvrdnji Schematron

Schematron¹⁷⁰ je jezik za umetanje tvrdnji o prisustvu ili odsustvu uzorka u XML dokumentima. U svojoj se osnovi Schematron ne temelji na gramatici, već na pronalaženju stablastih uzoraka (engl. *tree patterns*) tijekom parsiranja. Ovakav pristup temeljen na pravilima omogućuje prikaz različitih struktura koje je problematično prikazati u jezicima shema temeljenim na gramatici DTD, XSD i RELAX NG. Kod jezika temeljenih na gramatici sve što nije eksplicitno dopušteno nije valjano (engl. *invalid*), dok kod jezika temeljenih na pravilima sve što nije eksplicitno zabranjeno je valjano (engl. *valid*). Razvoj jezika tvrdnji kod validacije koristi se dijelom i zbog problematike međusobnih validacija, kao što je trivijalni primjer da ako u elementu <spol> stoji vrijednost „muški“, onda u istoj strukturi čvorova pod elementom <diagnoza> ne bi smjelo pisati „trudan“.

Schematron koristi XPath, a izveden je povrh tehnologije XSLT. Postoje i zasebne implementacije u jezicima Python i Perl. Schematron se može kombinirati s drugim jezicima za validaciju zasnovanim na gramatikama (DTD, XSD i RELAX NG). Tako se i dalje u prvom koraku mogu koristiti metode temeljene na gramatici za provjeru strukture i ograničenja vrijednosti, dok se za ograničenja između višestrukih elemenata i atributa te za validacije između više dokumenata mogu u drugom koraku koristiti pravila Schematrona. Sintaksa Schematrona je i sama opisana u jeziku XML te se može provjeriti zbijenom shemom (engl. *Compact Schema*) u obliku RELAX NG¹⁷¹. Pravila koriste izraze XPath, mogu se pohraniti u odvojenu datoteku (ekstenzija datoteke .sch) i mogu se ugraditi u druge datoteke sa shemama.

Schematron je postao i međunarodna norma ISO/IEC 19757-3:2006 *Information technology - Document Schema Definition Language (DSDL) - Part 3: Rule-based validation - Schematron*¹⁷².

Osnovni elementi Schematrona su:

- <schema> – glavni element dokumenta koji sadržava sve druge elemente,
- <title> – opcionalni element naslov sheme,
- <ns> – element prostora imena i prefiksa korištenih u shemi, ne mora postojati, a može ih biti više,
- <pattern> – element uzorka koji sadržava pravila,
- <rule> – element pravila s atributom context koji sadržava tvrdnje,
- <assert> – element tvrdnje s atributom test koji sadržava izraz XPath s opisom tvrdnje,
- <report> – element izvještaja s atributom test koji sadržava XPath izraz i opis izvještaja.

Osnovni oblik strukture Schematron dokumenta izgleda ovako:

```
<schema xmlns="http://purl.oclc.org/dsdl/schematron">
  <title>Schematron shema</title>
  <ns prefix="sch" uri="http://purl.oclc.org/dsdl/schematron">
```

¹⁷⁰ Schematron, <http://www.schematron.com>

¹⁷¹ RELAX NG Compact schema, <http://www.schematron.com/iso/iso-schematron.rnc.txt>

¹⁷² ISO/IEC 19757-3:2006 Information technology -- Document Schema Definition Language (DSDL) -- Part 3: Rule-based validation -- Schematron, ISO [http://standards.iso.org/ittf/PubliclyAvailableStandards/c040833_ISO_IEC_19757-3_2006\(E\).zip](http://standards.iso.org/ittf/PubliclyAvailableStandards/c040833_ISO_IEC_19757-3_2006(E).zip)

```

    <pattern>
      <rule context="sch:schema">
        <assert test="sch:pattern">
          ...
        </assert>
        <assert test="sch:pattern/sch:rule[@context]">
          ...
        </assert>
        <assert test="sch:pattern/sch:rule/sch:assert[@test] or
          sch:pattern/sch:rule/sch:report[@test]">
          ...
        </assert>
      </rule>
    </pattern>
  </schema>

```

Osnovni koraci rada mogu se opisati na sljedeći način:

```

Za svaki kontekstni čvor u dokumentu koji se validira
  Za svaku fazu koja se evaluira (ako faze postoje, inače uzorak)
    Za svaki uzorak (u fazi)
      Pronađi prvo pravilo koje odgovara kontekstnom čvoru
      Za svaku tvrdnju i izvještaj u pravilu
        Izvedi testiranje
          Ako test tvrdnje nije uspio ili je uspio test
            izvještaja
            Pošalji poruku
            (Ako postoji dijagnostika i uključena je
              Pošalji dijagnostičku poruku)

```

Bitan je samo poredak elemenata pravila, dok poredak ostalih elemenata ovisi o konkretnoj izvedbi. Uvjeti koji se ispituju odnose se na postojanje i vrijednosti elemenata i atributa. Pozitivne tvrdnje izražavaju ispitivanje korištenjem atributa test u kojem se nalaze XPath izrazi istinosnih rezultata (engl. *boolean XPath expression*). Ako kod ispitivanja uvjet nije ispunjen, onda se šalje poruka. Kod izvještaja je riječ o negativnim tvrdnjama pa će poruka biti poslana ako je uvjet ispitivanja ispunjen. Pravila čine skupine tvrdnji koje odabiru kontekstne čvorove koji se evaluiraju. Uzorci čine skupine pravila i prvo pravilo je ono koje se koristi.

Ponovno ćemo uzeti primjer XML dokumenta imenika s dvama osobama, sličan onom iz prethodnog poglavlja, i izvesti shemu za pravila.

```

<imenik>
  <osoba oib="12345678901">
    <ime>Ana</ime>
    <prezime>Anić</prezime>
    <email>ana.anic@racunarstvo.hr</email>
  </osoba>
  <osoba oib="23456789012">
    <ime>Ivo</ime>
    <prezime>Ivić</prezime>
    <email>ivo.ivic@racunarstvo.hr</email>
  </osoba>
</imenik>

```

Kao što se može primijetiti, ne koriste se faze, već samo jedan uzorak u kojem se ispituju različita pravila i tvrdnje.

```
<?xml version="1.0" encoding="UTF-8"?>
  <schema
xmlns="http://purl.oclc.org/dsdl/schematron">
  <title>Imenik<
    /title>
  <ns prefix="im" uri="http://www.racunarstvo.hr/imenik">
    <pattern
      name="all">
      <rule context="/">
        <assert test="im:imenik">
          Korijenski element mora biti imenik.
        </assert>
      </rule>
      <rule context="im:imenik">
        <assert test="count(*) = count(im:osoba)">
          Dozvoljena djeca elementa imenik mora biti element
          osoba.
        </assert>
      </rule>
      <rule context="im:osoba">
        <assert test="parent::im:imenik">
          Roditelj elementa osoba mora biti element imenik.
        </assert>
        <assert test="count(*) =
          count(im:ime|im:prezime|im:email)"> Djeca elementa
          osoba moraju biti element ime, prezime ili email.
        </assert>
        <assert test="count(im:ime) = 1">
          Element ime može biti samo jedan.
        </assert>
        <assert test="count(im:prezime) =
          1"> Element prezime može biti
          samo jedan.
        </assert>
        <assert test="count(im:email) =
          1"> Element email može biti
          samo jedan.
        </assert>
        <assert test="@oib">
          Element osoba mora imati atribut oib.
        </assert>
        <assert test="count(@*) = count(@oib)">
          Element osoba smije imati samo atribut oib.
        </assert>
      </rule>
      <rule context="im:ime">
        <assert test="parent::im:osoba">
          Roditelj elementa ime mora biti osoba.
        </assert>
        <assert test="count(*) = 0">
          Element ime ne može imati djecu.
        </assert>
      </rule>
      <rule context="im:prezime">
        <assert test="parent::im:osoba">
```

```

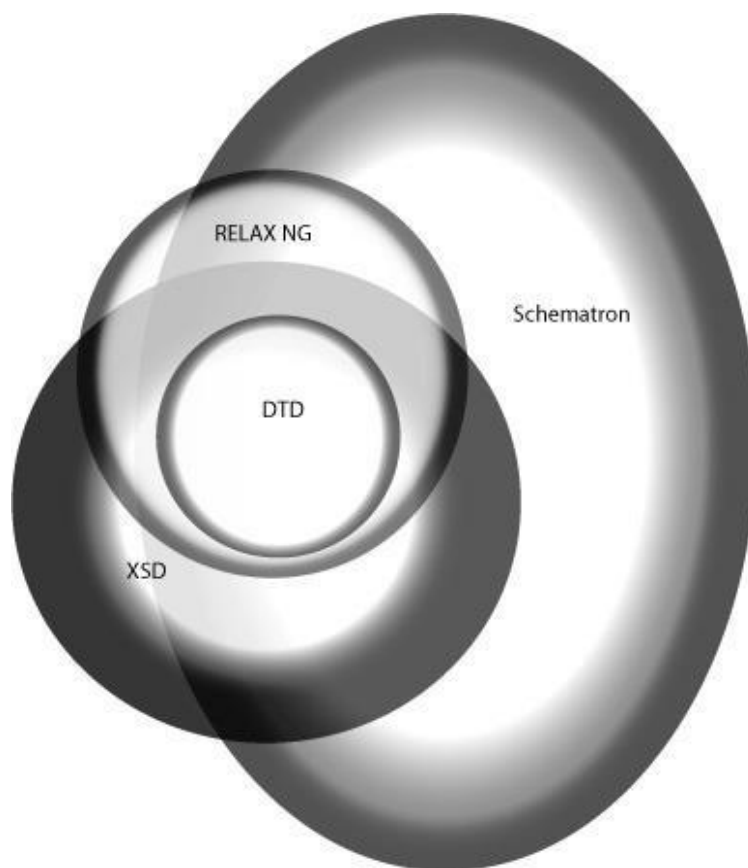
        Reditelj elementa prezime mora biti osoba.
    </assert>
    <assert test="count(*) = 0">
        Element prezime ne može imati djecu.
    </assert>
</rule>
<rule context="im:ime">
    <assert test="parent::im:osoba">
        Reditelj elementa email mora biti osoba.
    </assert>
    <assert test="count(*) = 0">
        Element email ne može imati djecu.
    </assert>
</rule>
<rule context="*">
    <assert test="true()">
        Element nije valjan u ovom dokumentu.
    </assert>
</rule>
...
</pattern>
</schema>

```

Kao što se može primijetiti, sam zapis je složeniji nego u jeziku RELAX NG, ali i s mnogo većim mogućnostima. Treba razmisliti kako se u jeziku Schematron mogu izvesti pravila podrške za element cijelolme iz prethodnog primjera kao alternativu elementima ime i prezime te kako bi provjerili duljinu znakova u vrijednosti atributa oib.

3.6.6 Grafička usporedba jezika

Ako želimo usporediti jezike DTD, XSD, RELAX NG i Schematron predstavljene u ovom poglavlju, bez potanjih analiza i dugih opisa može se izvesti dijagram skupova koji opisuje logičke odnose skupova (Slika 3.2). Nijanse sive boje na dijagramu imaju dodatno značenje, pa tako svjetlije nijanse označavaju mogućnosti koje su izvedive, ali su nezgodne u praksi, dok tamnije nijanse označavaju mogućnosti koje su izvedive samo ako se sheme prilagode na određeni način. Iako ovakav grafički prikaz može dovesti do raznolikih, pa čak i pogrešnih zaključaka, u općem smislu opisuje relativne odnose jezika validacije. Tako se može zaključiti da su gramatike zasnovane na XSD-u moćnije od gramatika zasnovanih na DTD-u, što ste već i sami uvidjeli na drugim kolegijima. Očito je da RELAX NG sadržava moćne konstrukcije, ali nisu izravno usporedive s XSD-om, dok Schematron pruža još veće mogućnosti.



Slika 3.2: Dijagram skupova jezika DTD, XSD, RELAX NG i Schematron¹⁷³

¹⁷³ Preuzeto iz http://www.oreillynet.com/xml/blog/2007/01/diagram_comparing_schema_langu.html

4. Poglavlje: Rukovanje i upravljanje podacima u obliku XML

U ovom poglavlju naučit ćete:

- ✓ o rukovanju podacima u obliku XML
- ✓ neke od najpoznatijih primjena označnih jezika

4.1 Rukovanje podacima u obliku XML

Podaci pohranjeni u obliku XML imaju namjenu sukladnu drugim strukturiranim podacima pohranjenim na bilo koji drugi način, primjerice u bazama podataka, datotekama i slično. Podaci pohranjeni u nekom obliku mogu se primati i slati, odnosno razmjenjivati, a strana koja ih u takvoj komunikaciji uspije ispravno prihvatiti očigledno zadovoljava osnovne zahtjeve sintaksne i tehničke interoperabilnosti. No osim što ih se je uspjelo prihvatiti, treba ih se moći i razumjeti kako bi ih se moglo upotrijebiti, što odgovara razini semantičke interoperabilnosti. Nakon što se podaci ispravno razumiju, postoji mogućnost njihovog korištenja u procesima poslovne logike i prikaza korisniku. Podatke treba preoblikovati u oblik koji udovoljava aktivnostima koje su zahtijevane od određene funkcionalnosti sustava ili aplikacije. Tada možemo govoriti o primjeni podataka za neku svrhu, a najčešće akcije su daljnja računalna obrada ili prikaz korisniku.

Kada se želi koristiti sadržaj XML dokumenata, dva su osnovna načina rada:

- **parsiranje temeljeno na stablu** (engl. *tree-based parsing*),
- **parsiranje temeljeno na događajima** (engl. *event-based parsing*).

Parsiranje temeljeno na stablu parsira cijeli XML dokument i prebacuje ga u objektu, odnosno stablastu strukturu nazvanu DOM, u kojoj se posljedično može pristupiti bilo kojem elementu, dok parsiranje temeljeno na događajima jednim prolazom sekvencijalno parsira XML dokument i generira događaje kako nailazi na pojedine dijelove, odnosno elemente.

4.1.1 Objektni model dokumenta DOM i sučelje SAX

U priručniku „Standardi u primjeni internetske tehnologije“ upoznali ste se s **objektnim modelom dokumenta** pod nazivom **DOM** (engl. *Document Object Model*). Ponovit ćemo ukratko da je DOM konvencija, neovisna o platformi i jeziku izvedbe, za interakciju s objektima označnih jezika, pri čemu se ponajprije misli na jezike HTML, XHTML i XML. Ta se interakcija odvija korištenjem programskih sučelja kojima se adresira i upravlja elementima, odnosno čvorovima.

S obzirom na to da ste se već upoznali s rukovanjem XML dokumentom korištenjem parsera koji koriste objektni model, nećemo se potanje baviti parsiranjem, svojstvima čvorova i metodama nad čvorovima, već ćemo objasniti korištenje modela DOM s aspekta interoperabilnosti. Budući da su izvedbe parsiranja i rezultatnog objektnog modela usko povezane s razvojem preglednika i skriptnih jezika, usporedit ćemo njihovu međuovisnost. Sve je započelo razvojem prvoga objektnog modela, djelomično opisanog u specifikaciji HTML 4, kojeg implementiraju preglednici Netscape Navigator 2 i Internet Explorer 3. Ova se verzija objektnog modela naziva i „Legacy DOM“ ili „DOM Level 0“. Razvoj skupa tehnologija DHTML (engl. *Dynamic HTML*), podržanih u preglednicima Netscape Navigator 4 i Internet Explorer 4, uveo je nove funkcionalnosti, kao što su poboljšano upravljanje objektnim oblikom HTML dokumenta, korištenje tehnologije CSS (engl. *Cascading Style Sheets*) i slojevitost (engl. *layeriness*) u prikazu. Prateće izvedbe objektnog modela nazvanog „Intermediate DOM“ razlikuju se između različitih preglednika i dolazi do prvih većih nekompatibilnosti.

Konzorcij W3C u cilju promicanja otvorenih normi i rješavanja nekompatibilnosti uvodi 1997. godine normu za skriptne jezike preglednika pod nazivom „ECMAScript“ te nastavlja s razvojem objektnog modela, kojeg normira 1998. godine pod nazivom „**DOM Level 1**“¹⁷⁴. Preglednici djelomično prihvaćaju ovu normu, a konzorcij W3C nastavlja razvoj novih verzija objektnog modela. Tako već 2000. godine izdaje „**DOM Level 2**“¹⁷⁵, koji uvodi model događaja, podršku za prostore imena i CSS, a 2004. godine izdaje „**DOM Level 3**“¹⁷⁶, koji dodaje podršku za XPath, upravljanje događajima tipkovnice i sučelje za serijalizaciju dokumenata u obliku XML, a 2015. godine objavljuje se „**DOM Level 4**“¹⁷⁷ koji konsolidira DOM Level 3 s dodatnim neovisnim komponentama. Većina preglednika izdanih nakon 2005. godine djelomično podržava navedene W3C-ove norme objektnog modela DOM, uključujući novije inačice Internet Explorera, preglednike zasnovane na platformi Gecko, kao što su Firefox i Mozilla, te druge preglednike kao što su Opera i Safari. Preglednici ne moraju nužno koristiti objektni model kod renderiranja prikaza dokumenta, ali skriptni jezik **JavaScript** koristi DOM kao osnovni način komuniciranja i upravljanja objektima u HTML dokumentu, odnosno dijelovima dokumenta.

Model DOM je najbolje iskorišten kod aplikacija koje učestalo pristupaju cjelovitom dokumentu u memoriji te omogućuje izravan pristup bilo kojem elementu na zahtjev, no ako je tijekom obrade dovoljan jedan prolaz kroz dokument prilikom parsiranja i cijeli dokument ne treba biti prisutan u memoriji u objektnom obliku, prikladniji su jednoprolazni sekvencijalni parseri s parsiranjem temeljenim na događajima.

Programsko sučelje **SAX**¹⁷⁸ (engl. *Simple API for XML*) sadržava sekvencijalni parser za XML dokumente koji radi na načelu jednosmjernog prolaza kroz dokument i vraća rezultatne podatke obrade u jednom toku (engl. *stream*), bez povratka na obrađeno mjesto, te nema navigaciju unatrag. Iako SAX ne postoji u obliku striktno specifikacije, svejedno je postao *de facto* norma. Iako je inicijalno pisan u jeziku, Java danas postoje implementacije u različitim programskim jezicima, kao što su Python, Perl, Pascal i C++.

SAX je zasnovan na pozivima metoda prilikom određenih događaja tijekom parsiranja. Događaji mogu biti temeljeni na pronalasku tekstualnih čvorova, čvorova elemenata, naredbi obrade ili komentara u XML dokumentu, dok se atributi obrađuju kao dijelovi čvora elementa. Ovo ćemo najbolje objasniti na sljedećem primjeru XML dokumenta.

```
<?xml version="1.0" encoding="UTF-8"?>
<osoba oib="12345678901">
  <ime>
    Ana
  </ime>
  <?naredba_obrade obradi_djevojacko_prezime="da"?>
  < prezime postojiDjevojackoPrezime="da">
```

¹⁷⁴ Document Object Model (DOM) Level 1 Specification, ver. 1.0, W3C Recommendation, 1998., <http://www.w3.org/TR/REC-DOM-Level-1>

¹⁷⁵ Document Object Model (DOM) Level 2 Core Specification, ver. 1.0, W3C Recommendation, 2000., <http://www.w3.org/TR/DOM-Level-2-Core>

¹⁷⁶ Document Object Model (DOM) Level 3 Core Specification, ver. 1.0, W3C Recommendation, 2004., <http://www.w3.org/TR/DOM-Level-3-Core>

¹⁷⁷ Document Object Model (DOM) Level 4 Core Specification, ver. 1.0, W3C Recommendation, 2015., <https://www.w3.org/TR/domcore/>

¹⁷⁸ SAX, <http://www.saxproject.org/>

```
Anić  
  <djevojackoPrezime>Ivić</djevojackoPrezime>  
</prezime>  
</osoba>
```

SAX parser će pri obradi navedenog dokumenta generirati sljedeći niz događaja:

- početak čvora elementa osoba s atributom oib vrijednosti „12345678901“,
- početak čvora elementa ime,
- tekstualni čvor sadržaja „Ana“,
- završetak čvora elementa ime,
- naredba obrade „naredba_obrade“ s atributom „obradi_djevojacko_prezime“ vrijednosti „da“,
- početak čvora elementa Prezime s atributom „postojiDjevojackoPrezime“ vrijednosti „da“,
- tekstualni čvor sadržaja „Anić“,
- početak čvora elementa djevojackoPrezime,
- tekstualni čvor sadržaja „Ivić“,
- završetak čvora elementa djevojackoPrezime,
- završetak čvora elementa prezime,
- završetak čvora elementa osoba.-

Danas SAX postoji u dvije glavne inačice, od kojih se zadnja jednostavno naziva SAX2. SAX2 je zadržao osnovnu ideju parsiranja temeljenog na događajima, no programiran je praktički ispočetka kako bi podržao neke nove funkcionalnosti, kao što su prostori imena, filteri, podrška za leksičke događaje, podrška za DTD i mnoge druge. U međuvremenu se razvija i nova inačica SAX 2.1, koja donosi još neke novije, dosta specifične funkcionalnosti, koje ipak u većini standardnih korištenja nisu naročito važne.

Napomena: Inačica SAX2 je danas praktički potpuno podržana u svim parserima te se preporučuje kao tržišna norma za sekvencijalni parser temeljen na događajima.

Prednosti pristupa jednoprolaznog parsera su u manjem korištenju memorijskih resursa, s obzirom na to da za razliku od modela DOM ne pohranjuju cijelo stablo dokumenta u memoriji. U općem slučaju može se reći da su memorijski zahtjevi parsera SAX usporedivi s najvećom dubinom stabla u XML dokumentu i najvećim podatkom pohranjenim u jedan XML element, što je u svakom slučaju manje od cijelog dokumenta. Ovo je značajno kod vrlo velikih XML struktura na uređajima s malom količinom memorije, kada se može dogoditi da je objektni model veći od raspoložive radne memorije pa parsiranje DOM-om ne može funkcionirati, dok se parsiranje SAX-om može izvoditi. Kod DOM-a se u takvom slučaju često pribjegava pohrani objektnog modela na veću, ali sporiju memoriju, primjerice datotečni sustav diska. Osim toga, brzina parsiranja SAX-om je u načelu uvijek brža nego kod parsera modela DOM. SAX je bolji i ako dolazi do promjena strukture, jer se kod DOM-a onda praktički treba rukovati s dvama velikim strukturama u memoriji, starom i novom.

No ako se određene validacije izvode nad cijelom XML strukturom, odnosno korištenjem različitih dijelova istovremeno, to nije moguće izvesti u jednom prolazu parsera pa SAX nije dobar odabir. Kod DOM-a se jednom učitani XML dokument više ne dohvaća, već se može

raditi samo s objektnom strukturom u memoriji. Jednako tako, ako određene tehnologije, kao što su XPath i XSLT, trebaju imati pristup bilo kojem čvoru dokumenta tijekom obrade, onda SAX ne može odgovoriti na takve zahtjeve.

Napomena: SAX je u načelu brži i koristi manje memorijskih resursa te se koristi kod pronalaženja pojedinih elemenata u jednom prolazu i kod promjena struktura. DOM se koristi kod višestrukih ponovnih korištenja elemenata, kod paralelnih pronalaženja i kod transformacija koje zahtijevaju cijeli dokument u objektnoj strukturi.

Postoji još i sučelje **StAX**¹⁷⁹ (engl. *Streaming API for XML*), koje se razvilo kao rješenje koje se postavlja između stablaste strukture, odnosno objektnog modela, s jedne strane i sekvencijalnog jednoprolaznog parsiranja temeljenog na događajima s druge strane. Sučelje StAX koristi programatsku ulaznu točku kao pokazivač na određenu lokaciju u dokumentu. Aplikacija pokreće pokazivač naprijed, dohvaćajući pritom parsirane podatke.

Najpoznatije implementacije objektnog modela DOM i sučelja SAX su u skupu usluga **MSXML** (engl. *Microsoft XML Core Services*), biblioteci **Xerces** izvorno napisanoj u jeziku C++, koja je danas izvedena i u jezicima Java i Perl, te biblioteci **libxml2** napisanoj u jeziku C. Osim toga postoje i programska sučelja uključena u osnovne biblioteke jezika Java, primjerice **JAXP**¹⁸⁰ (engl. *Java API for XML Processing*), koji objedinjuje parsiranja temeljena na sučeljima DOM, SAX i StAX. Postoje i alternativni objektni modeli otvorenog koda za XML pisani u jeziku Java, kao što su JDOM i dom4j.

4.1.2 Transformacije i prezentacije korištenjem jezika XSL (XPath, XSLT i XSL-FO)

U ovom ćemo se poglavlju nakratko podsjetiti na temu jezika **XSL** (engl. *eXtensible Stylesheet Language*) opisanog u priručniku „Standardi u primjeni internetske tehnologije“. XSL je zapravo porodica preporuka za definiranje transformacije i prezentacije XML dokumenata, a sastoji se od triju dijelova:

- jezika **XPath** ili XML Path Language,
- jezika **XSLT** (engl. *eXtensible Stylesheet Language Transformations*),
- jezika **XSL-FO** (engl. *eXtensible Stylesheet Language – Formatting Objects*).

S obzirom na to da ste već upoznali s preporukama XSL, obratit ćemo više pozornosti na način korištenja ove preporuke u kontekstu obrade XML dokumenata i interoperabilnosti.

XPath je jezik izraza (engl. *expression language*) koji se koristi za pristup i lociranje, odnosno adresiranje dijelova XML dokumenata. Objavljen je u verziji 1.0 kao preporuka W3C-a¹⁸¹ još davne 1999. godine. Široko je rasprostranjen i koristi se u mnogim programskim jezicima,

¹⁷⁹ Streaming API for XML, <http://www.jcp.org/en/jsr/detail?id=173>

¹⁸⁰ Java API for XML Processing, <https://jaxp.dev.java.net/>

¹⁸¹ XML Path Language, Version 1.0, W3C Recommendation, 1999-11-16, <http://www.w3.org/TR/xpath/>

uključujući jezike Java, C# i JavaScript, odnosno ugrađen je u jezik XSLT i XForms¹⁸². Godine 2007. izašla je nova preporuka W3C-a za jezik XPath 2.0¹⁸³, koji koristi puno bogatiji sustav tipova, pa se čvorovi promatraju kao članovi sekvencije u određenom poretku. Osim toga proširen je skup funkcija i operatora, a osim inicijalna četiri tipa izraza mogu se koristiti i atomarni tipovi definirani kao ugrađeni tipovi XML sheme te korisnički definirani tipovi, uključujući tipove za datume, vrijeme, URI-je i druge. I dalje postoji sedam vrsta čvorova, osim što se korijenski čvor naziva čvorom dokumenta. XPath 2.0 je zapravo podskup jezika XQuery 1.0 i nudi niz izraza koji se koriste u jeziku XQuery. Sastavljen je i prijedlog XPath 3.0¹⁸⁴, koji sadrži funkcije visoke razine i još bolju podršku za XSLT i XQuery.

Napomena: Iako je XPath 2.0 već neko vrijeme prisutan, još uvijek je jedan dio primjena zasnovan na starijem XPath-u 1.0 pa pri upotrebi treba biti oprezan.

XSLT je deklarativni jezik za transformacije XML dokumenata koji omogućava prikaz i dodavanje elemenata i atributa, brisanje i sakrivanje dijelova dokumenta, uređivanje, oblikovanje i sortiranje elemenata te uvjetno izvršavanje pravila i testiranje. Podsjetimo se da je očekivani rezultat postupka transformacije novi XML dokument, no moguće je proizvesti i drugi dokument koji nije XML dokument, već običan tekst (engl. *plain text*). Kada govorimo o transformaciji u oblik za prezentaciju putem stranica Weba očigledno je da je očekivani izlazni oblik HTML ili XHTML dokument. Ako transformacija prati pravila XML-a, onda je rezultat XHTML dokument, a ako ne prati, onda je rezultat dokument s oznakama HTML koji ne odgovara pravilima XML-a pa nije XHTML dokument, već HTML dokument u obliku običnog teksta.

Prva verzija XSLT-a¹⁸⁵ u obliku preporuke W3C-a izašla je istovremeno kad i XPath 1.0 te je danas široko prihvaćena. U međuvremenu se pokušalo izdati verziju 1.1, od koje se na kraju odustalo. Verzija XSLT-a je 2.0¹⁸⁶, izašla 2007. godine kao preporuka W3C-a, koja je donijela novosti u obliku konverzija fragmenata rezultatnog stabla u skupove čvorova, višestruke izlazne dokumente, ugrađenu podršku za grupiranja i korisnički definirane funkcije. Aktualna verzija 3.0¹⁸⁷ objavljena je 2017. godine i uključuje *streaming* način rada pa izvorni dokument je treba biti spremljen u memoriji u potpunosti te je dizajniran za rad s verzijama XPath-a 3.0 i 3.1. XSLT 3.0 također uključuje mogućnost serijalizacije rezultata transformacije..

Osnovni XSLT model sadržava XML izvorni dokument, predložak stila XSLT i XSLT procesor koji izvodi transformaciju. Posljedično sadržava i odredišni dokument kao rezultat transformacije. Predložak stila XSLT je također XML dokument koji sadržava niz pravila i naredbi XSLT procesora kako izvesti transformacije. Izvedbe XSLT procesora postoje u obliku samostalnih aplikacija ili kao komponenta ugrađena u preglednik, aplikacijski poslužitelj, programski radni okvir (engl. *framework*) ili sam operacijski sustav te se očigledno mogu

¹⁸² XForms, Version 1.1, W3C Recommendation, 2009-10-20, <http://www.w3.org/TR/xforms/>

¹⁸³ XML Path Language (XPath) 2.0, W3C Recommendation, 2007-01-23, <http://www.w3.org/TR/xpath20/>

¹⁸⁴ XPath 3.0, <http://www.w3.org/TR/xpath-30/>

¹⁸⁵ XSL Transformations (XSLT) Version 1.0, W3C Recommendation, 1999-11-16, <http://www.w3.org/TR/xslt>

¹⁸⁶ XSL Transformations (XSLT) Version 2.0, W3C Recommendation, 2007-01-23, <http://www.w3.org/TR/xslt20/>

¹⁸⁷ XSL Transformations (XSLT) Version 3.0, W3C Recommendation, 2019-06-08, <https://www.w3.org/TR/xslt-30/>

izvršavati na klijentu ili poslužitelju. Poznatiji XSLT procesori uključuju Saxon, Xalan, xsltproc, XT, MSXML, Sablotron i 4XSLT.

Zadnji dio XSL-a je jezik **XSL-FO**, koji se koristi kod oblikovanja XML dokumenata za prikaz. Iako oblikovanje podataka za prikaz nije nužno vezano uz interoperabilnost, ovdje ćemo zbog cjelovitosti pregleda navesti osnovne izvedbe. O metodama **prikaza podataka u obliku XML** već ste čitali u priručniku „Standardi u primjeni internetskih tehnologija“, gdje je opisano kako se podaci iz XML datoteke mogu jednostavno prikazati korisniku. Mogućnosti prikaza XML dokumenata su sljedeće:

- **Primjena predloška CSS** (engl. *Cascading Style Sheets*) na XML dokument kako bi se podaci prikazali u pregledniku. U CSS datoteci se za selektore navode elementi iz XML dokumenta uz koje se vežu deklaracije stilova oblikovanja.
- **Korištenje XSLT-a za transformaciju XML datoteke u HTML ili XHTML datoteku**, na koju se opet naknadno može primijeniti predložak CSS, te razni drugi grafički elementi, kao što su slike, fontovi i slično.
- **Korištenje XSL-FO-a** za formatiranje prikaza i posljedičnu konverziju u PDF (engl. *Portable Document Format*), PS (engl. *PostScript*), RTF (engl. *Rich Text Format*) i druge oblike
- **Obrada XML dokumenta u aplikaciji** i prebacivanje oblikovanih podataka u neki oblik grafičkog sučelja, bilo da je riječ o tankom Web klijentu, grafičkom sučelju debelog klijenta, izvještaju iz izvještajnog podsustava (engl. *reporting engine*) ili nekom drugom. Ovo se najčešće izvodi korištenjem programiranja za specifičnu namjenu prikaza ili uporabom gotovih predložaka u određenim tehnologijama.

Napomena: Svi navedeni načini prikaza podataka u obliku XML postoje u praksi i općenito se ne može govoriti o preporukama, već se ovisno o potrebama i zahtjevima određene primjene procjenjuje najprimjereniji način prikaza podataka u obliku XML.

4.1.3 Jezik upita XQuery

Označni jezik upita i funkcionalnog programiranja XQuery¹⁸⁸ ili **XML Query** je izveden za upite nad kolekcijama XML podataka, ali može pronalaziti sadržaj i u raznim drugim oblicima podatkovnih izvora, uključujući strukturirane i polustrukturirane dokumente, relacijske baze podataka i repozitorije objekata. Jezik XQuery, korištenjem strukture u obliku XML, može izvoditi upite nad različitim oblicima zapisa podataka dok su pohranjeni u obliku XML ili se od transformacijskih mehanizama i međuopreme (engl. *middleware*) mogu prikazati u obliku XML.

Jezik XQuery, kao preporuka W3C-a, razvijao se paralelno s jezikom XSLT 3.0 te je intenzivna suradnja dviju razvojnih skupina uz podijeljenu odgovornost izražena u zajedničkom razvoju već navedenog jezika XPath u verziji 3.0. Kao što je već navedeno, XSLT 3.0 intenzivno koristi

¹⁸⁸ XQuery 1.0: An XML Query Language, W3C Recommendation, 2007., <http://www.w3.org/TR/xquery>

XPath 3.0. i 3.1, koji je zapravo podskup jezika XQuery, pa oba jezika dijele iste funkcije i operatore te se izvode nad istim podatkovnim modelom sa sedam tipova čvorova.

Poznata je izjava „XQuery je za XML što je SQL za baze podataka“. Misija projekta izrade jezika XQuery je izgradnja prilagodljivih upita koji pronalaze podatke na različitim dokumentima Weba, što omogućuje interakciju svijeta Weba i baza podataka. XQuery se može koristiti za pretraživanje dokumenata Weba, izradu izvještaja, transformacije XML podataka u druge oblike, kao što je XHTML, te izdvajanje informacija, što može služiti za razne svrhe, uključujući korištenje u tehnologijama Web Services.

XQuery koristi sintaksu XPath izraza za adresiranje dijelova XML dokumenta. No, za razliku od jezika XPath, to može izvoditi i korištenjem izraza FLWOR, koji se gradi iz pet osnovnih izjava: FOR, LET, WHERE, ORDER BY i RETURN. Osim toga sintaksa za izgradnju novih XML dokumenata može se koristiti na dva načina. Ako su nazivi elemenata i atributa struktura u obliku XML unaprijed poznati, može se koristiti sintaksa zasnovana na XML-u, a ako nisu, koriste se izrazi za dinamički izgradnju čvorova, te se oba načina mogu kombinirati.

Sustav tipova podataka zasnovan je na sljedovima ili sekvencijama (engl. *sequence*) čvorova ili atomarnih vrijednosti u skladu s tipovima XML sheme, kao što su cijeli brojevi, znakovni nizovi, istinosne vrijednosti i slično, a jednostruka se vrijednost smatra sekvencijom duljine jedan. Sintaksa je usporediva s pravilima koja općenito vrijede za XML, pa se razlikuju velika i mala slova, a nazivi moraju udovoljavati pravilima XML-a. Varijable se definiraju znakom „\$“, dok se komentari odvajaju znakovima „(:“ i „:)“ koji podsjećaju na „smješka“ (engl. *smiley*). Uvjetni izrazi ostvaruju se korištenjem sintakse „if-then-else“, a usporedbe se izvode korištenjem standardnih operatora =, !=, <, <=, > i >=, odnosno eq, ne, lt, le, gt i ge kod jednostrukih povratnih vrijednosti.

Kako bismo promotрили rada XQueryja, uzet ćemo primjer XML dokumenta podaci.xml s imenikom osoba:

```
<imenik>
  <osoba oib="12345678901">
    <ime>Ana</ime>
    <prezime>Anić</prezime>
    <email>ana.anic@racunarstvo.hr</email>
    <godinaRodjenja>1985</godinaRodjenja>
  </osoba>
  <osoba oib="23456789012">
    <ime>Ivo</ime>
    <prezime>Ivić</prezime>
    <email>ivo.ivic@racunarstvo.hr</email>
    <godinaRodjenja>1991</godinaRodjenja>
  </osoba>
</imenik>
```

Za primjer ćemo koristiti sljedeći XQuery upit:

```
doc("podaci.xml")/imenik/osoba[godinaRodjenja<1990]
```

Prvo se koristi funkcija doc() koja otvara dokument podaci.xml, zatim XPath izraz koji locira i adresira čvorove osoba u čvoru dokumenta imenik, te predikat godinaRodjenja<1990 koji

postavlja uvjet dohvata podataka na osobe rođene prije 1990. godine. Rezultat izvođenja bit će sljedeći:

```
<osoba oib="12345678901">
  <ime>Ana</ime>
  <prezime>Anić</prezime>
  <email>ana.anic@racunarstvo.hr</email>
  <godinaRodjenja>1985</godinaRodjenja>
</osoba>
```

Sad ćemo za primjer dodati i nastavak XPath izraza koji locira čvor ime, kako slijedi:

```
doc("podaci.xml")/imenik/osoba[godinaRodjenja<1990]/ime
```

Dobiveni rezultat izvođenja bit će:

```
<ime>Ana</ime>
```

Isti upit zapisan kao izraz FLWOR korištenjem izjava for, where i return izgledao bi ovako:

```
for $a in doc("podaci.xml")/imenik/osoba
  where $a/godinaRodjenja<1990
  return $a/ime
```

Kao što smo već spomenuli, izrazi se mogu kombinirati s drugim oblicima pa se tako može oblikovati izlaz u obliku HTML tablice za ispis svih imena i prezimena kao u sljedećem primjeru:

```
<table>
  <tr><th>IME</th><th>PREZIME</th></tr>
  <tr>
  {
    for $a in doc("podaci.xml")/imenik/osoba
    return <td>{data($a/ime)}</td> <td>{data($a/prezime)}</td>
  }
  </tr>
</table>
```

4.1.4 JAXB

Java Architecture for XML Binding – JAXB¹⁸⁹, poznatiji i kao JSR 222, radni je okvir koji omogućuje jednostavnu manipulaciju XML dokumentima. Glave prednosti se očituju u mnogo jednostavnijem korištenju u odnosu na klasične metode koje slijede DOM metodologiju.

Kod modela DOM (poglavlje 4.1.1) kreira se stablo objekata koje prikazuje sadržaj i organizaciju podataka koji se nalaze u promatranom XML dokumentu. Ograničenja u ovom pristupu uključuju vrijednosti koje su dostupne isključivo kao podaci tipa String i slično.

Kada se XML dokumentu pristupa korištenjem JAXB metoda, također ćemo dobiti stablo objekata, no sa značajnom razlikom što će čvorovi u tome stablu u potpunosti odgovarati XML

¹⁸⁹ JSR 222: Java Architecture for XML Binding (JAXB) 2.0, <http://jcp.org/en/jsr/detail?id=222>

elementima, sadržavati atribute i sadržaj u obliku varijabli instance, te će zadržavati veze a objekte djecu putem objektnog referenciranja.

JAXB definira standardan način vezivanja (engl. *binding*) iz objekata (tipova podataka) napisanih u programskom jeziku Java u komponente definirane XML Schemom i obrnuto. Korištenjem ovog radnog okvira, ponešto je moguće pojednostaviti proces u kojem je potrebno koristiti Java i XML Schema. No kako bi se ostvario taj cilj, sama je specifikacija morala definirati veliku količinu različitih veza, tako da je posljednja verzija ove specifikacije izdana 11.07.2017. godine (inačica 2.3) ima 344 stranica.

Osnovna ideja temeljem koje je nastao radni okvir JAXB je da se olakša programerima u Javi da učitavaju i obrađuju podatke zapisane u XML-u izravno iz programskog koda. Iz tog je razloga za početak korištenja ovog radnog okvira dovoljno osnovno poznavanje XML-a. Ukoliko se u aplikaciji koristi JAXB, neće biti potrebno stvaranje vlastitog parsera XML-a koda, već korištenjem pripadajućih alata razvijenih na temelju ove specifikacije osigurava automatsko kreiranje primjerice Javinih klasa na temelju dobivene XML datoteke.

Temelj za proces vezivanja Javinih objekata i XML komponenti jesu mapiranja istih objekata jednih prema drugima. Mapiranje elemenata definiramo kao njihov međusobni odnos. Sam proces mapiranja provodi se kroz proces serijalizacije (kada se instancija klase u programskom jeziku Java preoblikuje u instancu koja odgovara XML Schemi) u jednom smjeru, odnosno deserijalizacije (obrnut proces, iz XML komponente u instancu klase u Javi) u drugome. Proces mapiranja ne mora nužno imati obilježja neke funkcije, tj. nije nužno da se klase i komponente mapiraju u odnosu jedan za jedan, već je moguće da neka klasa bude mapirana u dvije komponente XML Scheme.

U specifikaciji JAXB 2.0 postoje dvije izvedbe procesa vezivanja Javinih klasa i XML Scheme. Prvi proces vezivanja kreće od Javine klase te se pomoću nje i JAXB alata kreiraju odgovarajuće komponente u XML-u. Pritom Javin kod ne mora biti ni označen, jer će se u tom slučaju koristiti podrazumijevane (engl. *default*) oznake definirane u JAXB specifikaciji. U drugoj je inačici procesa vezivanja Jave i XML-a na početku dostupna XML Schema (ili više njih) te se Javina klasa generira korištenjem JAXB alata.

Kada govorimo općenito o sustavima zasnovanim na uslugama i uslugama Weba, možemo uočiti jednu od razlika. Naime, kod njih ne postoji ovakav slučaj, već se obično pojavljuje problem jer istovremeno postoje definirana XML Schema i već napisan programski kod uobičajen u nekoliko klasa u Javi. Tu dolazi do problema kako vezati već postojeće objekte na već postojeće komponente.

Općenito govoreći, proces vezivanja u alatima koji su nastali kao implementacija specifikacije JAXB 2.0 izveden je kao prevoditelj (engl. *compiler*) odnosno generator sheme. Pritom sama specifikacija omogućava neke dodatne mehanizme, poput mijenjanja naziva koji se koriste u klasi (za što se koristi *JAXB Binding Language*). Ipak, osnovna struktura klase u pravilu je određena načinom izrade komponente u shemi koja se prevodi u klasu. Primjerice, neki element koji se prema shemi može pojaviti više puta definiran je sljedećim odsječkom:

```
<xs:element name="primjer" type="Bar" maxOccurs="unbounded" />
```

Ako nad ovim elementom XML Scheme provedemo njegovo prevođenje u sukladan tip podatka u programskom jeziku Java, tada ga, ovisno o potrebama naše usluge, možemo prikazati bilo kao `List<Bar>` ili kao `Bar[]`, no njihova će struktura biti praktički ista.

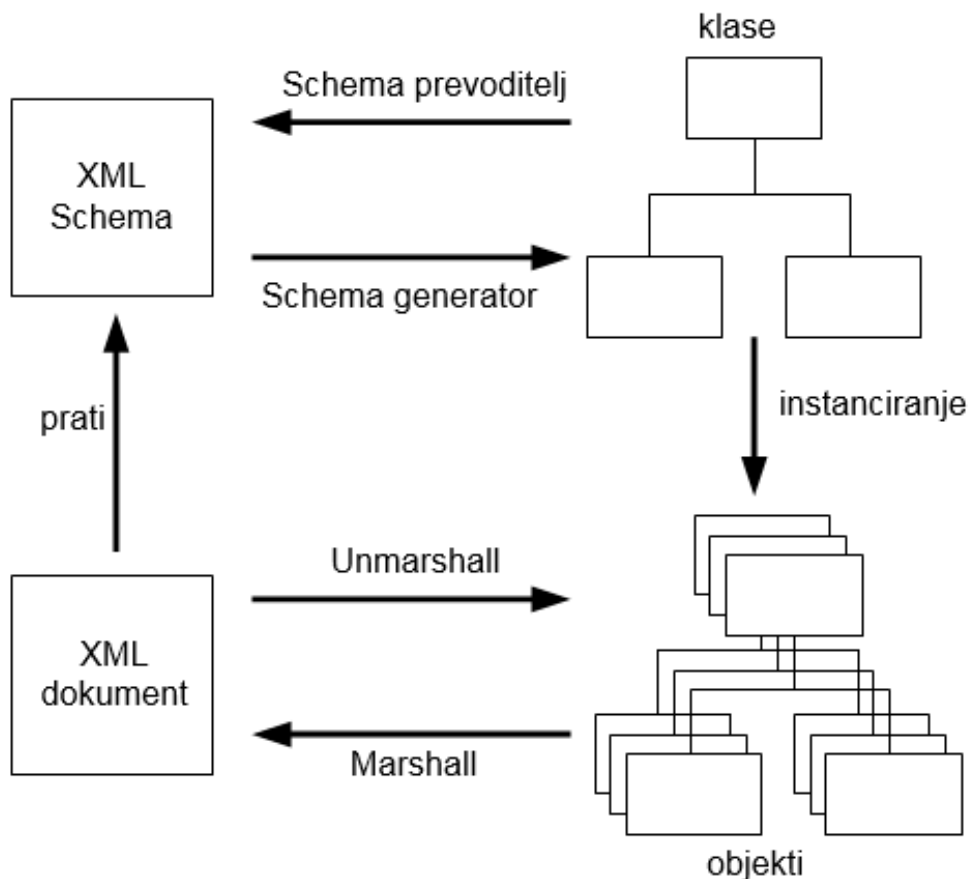
Sljedeća tablica prikazuje kako će se pojedini elementi XML Scheme mapirati u odgovarajuće tipove podataka u Javi.

XML Schema Tip	Javin tip podatka
xsd:string	<code>java.lang.String</code>
xsd:positiveInteger	<code>java.math.BigInteger</code>
xsd:int	<code>int</code>
xsd:long	<code>long</code>
xsd:short	<code>short</code>
xsd:decimal	<code>java.math.BigDecimal</code>
xsd:float	<code>float</code>
xsd:double	<code>double</code>
xsd:boolean	<code>Boolean</code>
xsd:byte	<code>byte</code>
xsd:QName	<code>javax.xml.namespace.QName</code>
xsd:dateTime	<code>javax.xml.datatype.XMLGregorianCalendar</code>
xsd:base64Binary	<code>byte[]</code>
xsd:hexBinary	<code>byte[]</code>
xsd:unsignedInt	<code>long</code>
xsd:unsignedShort	<code>int</code>
xsd:unsignedByte	<code>short</code>
xsd:unsignedLong	<code>java.math.BigDecimal</code>
xsd:time	<code>javax.xml.datatype.XMLGregorianCalendar</code>
xsd:date	<code>javax.xml.datatype.XMLGregorianCalendar</code>
xsd:g	<code>javax.xml.datatype.XMLGregorianCalendar</code>
xsd:anySimpleType	<code>java.lang.Object</code>
xsd:anySimpleType	<code>java.lang.String</code>
xsd:duration	<code>javax.xml.datatype.Duration</code>
xsd:NOTATION	<code>javax.xml.namespace.QName</code>

Sljedeća tablica daje pregled vezivanja klasa u Javi u XML podatkovne tipove:

Javina klasa	XML podatkovni tip
<code>java.lang.String</code>	<code>xs:string</code>
<code>java.math.BigInteger</code>	<code>xs:integer</code>
<code>java.math.BigDecimal</code>	<code>xs:decimal</code>
<code>java.util.Calendar</code>	<code>xs:dateTime</code>
<code>java.util.Date</code>	<code>xs:dateTime</code>
<code>javax.xml.namespace.QName</code>	<code>xs:QName</code>
<code>java.net.URI</code>	<code>xs:string</code>
<code>javax.xml.datatype.XMLGregorianCalendar</code>	<code>xs:anySimpleType</code>
<code>javax.xml.datatype.Duration</code>	<code>xs:duration</code>
<code>java.lang.Object</code>	<code>xs:anyType</code>
<code>java.awt.Image</code>	<code>xs:base64Binary</code>
<code>javax.activation.DataHandler</code>	<code>xs:base64Binary</code>
<code>javax.xml.transform.Source</code>	<code>xs:base64Binary</code>
<code>java.util.UUID</code>	<code>xs:string</code>

Pažljivijim iščitavanjem specifikacije te proučavanjem dijelova koda, uočiti ćete da unutar samog JAXB API-ja (koji se nalazi u `javax.xml.bind` paketu) središnje mjesto zauzima klasa `JAXBContext`. Tu su i dva sučelja: `Unmarshaller` – koji deserijalizira XML u Javu te po potrebi provodi validaciju dobivenog XML-a, i `Marshaller` – koji izvodi proces serijalizacije. Slika 4.1 prikazuje korištenje specifikacije JAXB.



Slika 4.1: JAXB arhitektura i njeno korištenje

Slično kao i već opisana specifikacija JAX-WS, i ova specifikacija intenzivno koristi oznake. Među ostalima, koriste se sljedeće oznake¹⁹⁰:

- `XmlValue` – mapira klasu u složeni tip s jednostavnim sadržajem prema XML Schemi, ili u jednostavni tip prema XML Schemi.
- `XmlType` – mapira klasu u tip (jednostavni ili složeni) prema XML Schemi. Sadržava elemente `name`, `namespace` i `propOrder`.
- `XmlSchema` – mapira ime paketa u XML imenički prostor. Moguće je definirati elemente `attributeFormDefault`, `elementFormDefault`, `namespace` i `xmlns`.
- `XmlRootElement` – mapira klasu kao korijenski element XML datoteke. Sadržava elemente `name` i `namespace`.
- `XmlList` – mapira svojstvo u jednostavni oblik liste.

¹⁹⁰ Vohra, D., Improved XML Binding with JAXB 2.0

- `XmlEnum` – mapira Javin tip enum u jednostavni tip s enumeracijom.
- `XmlElement` – mapira svojstvo JavaBeana kao element. Sadržava elemente `defaultValue`, `name`, `namespace`, `nillable`, `Type`.
- `XmlAttribute` – mapira svojstvo JavaBeana kao atribut. Sadržava elemente `name`, `namespace` i `required`.

Poredak navođenja ovih oznaka u vašem kodu nije bitan jer će se prevoditelj pobrinuti da odgovarajuće elemente u XML-u smjesti na pravo mjesto. Stoga je dopušteno da se primjerice `@XmlRootElement` smjesti nakon oznake `@XmlType` premda predstavlja korijenski element XML dokumenta koji nastaje JAXB vezivanjem.

Da bi ilustrirali prednosti uporabe JAXB radnog okvira nad tradicionalnim metodama rada sa XML dokumentima, izvesti ćemo za početak jednostavan „Hello World“ primjer¹⁹¹.

Za početak, napišimo XML Schemu koja definira strukturu našeg dokumenta:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:jxb="http://java.sun.com/xml/ns/jaxb"
  jxb:version="2.0">

  <xsd:element name="Greetings" type="GreetingListType"/>

  <xsd:complexType name="GreetingListType">
    <xsd:sequence>
      <xsd:element name="Greeting" type="GreetingType"
        maxOccurs="unbounded"/>
    </xsd:sequence>
  </xsd:complexType>

  <xsd:complexType name="GreetingType">
    <xsd:sequence>
      <xsd:element name="Text" type="xsd:string"/>
    </xsd:sequence>
    <xsd:attribute name="language" type="xsd:language"/>
  </xsd:complexType>

</xsd:schema>
```

Kao što možemo uočiti, dana shema definira da će naš dokument sadržavati listu pozdrava, a svaki pozdrav će, uz samu izrijeku pozdrava sadržavati i atribut koji će označavati jezik na kojem je on izrečen.

Ukoliko na vaše računalo preuzmete referentnu implementaciju JAXB-a¹⁹², tada ćete s njom dobiti i nekoliko korisnih alata, među kojima treba izdvojiti:

- XML Schema Generator (schemagen) koji će napraviti schemu na temelju skupa klasa napisanih u programskom jeziku Java

¹⁹¹ JAXB Tutorial, <http://jaxb.java.net/tutorial>

¹⁹² JAXB RI 2.2.7, <http://jaxb.java.net/2.2.7/>

- XML Schema Compiler (xjc) kojim ćete moći iz scheme napraviti Java klase. Ovaj se alat može proširivati dodacima (engl. Plugins)

Ukoliko gore prikazanu schemu spremimo na tvrdi disk pod imenom hello.xsd, možemo iskoristiti alat xjc kako bi generirali odgovarajuće klase u programskom jeziku Java:

```
xjc -p hello hello.xsd
```

Pokretanjem ove naredbe kreirati će se nekoliko klasa u poddirektoriju hello. Klasa Hello pokazuje nam kako ih trebamo koristiti.

```
import java.util.*;
import javax.xml.bind.*;
import hello.*;

public class Hello {

    private ObjectFactory of;
    private GreetingListType grList;

    public Hello(){
        of = new ObjectFactory();
        grList = of.createGreetingListType();
    }

    public void make( String t, String l ){
        GreetingType g = of.createGreetingType();
        g.setText( t );
        g.setLanguage( l );
        grList.getGreeting().add( g );
    }

    public void marshal() {
        try {
            JAXBElement<GreetingListType> gl =
                of.createGreetings( grList );
            JAXBContext jc = JAXBContext.newInstance( "hello" );
            Marshaller m = jc.createMarshaller();
            m.marshal( gl, System.out );
        } catch( JAXBException jbe ){
            // ...
        }
    }
}
```

Pažljivijim čitanjem programskog koda možemo uočiti da konstruktor koristi metodu klase ObjectFactory kako bi stvorio korijenski element u našem XML dokumentu (u ovome slučaju, radi se o GreetingListType). Metoda make() dodaje još jedan pozdrav u listu, sa atributima u obliku teksta i jezika na kojem je sam pozdrav. Naposljetku, pozivanjem metode marshal() lista se zamata u obliku XML dokumenta, koji se potom ispisuje na standardnom izlazu. Pogledajmo kako izgleda lista poziva odgovarajućih metoda:

```
Hello h = new Hello();
h.make( "Bonjour, madame", "fr" );
h.make( "Hey, you", "en" );
h.marshal();
```

U konačnici, ovako kreiran XML dokument će izgledati ovako:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<Greetings>
  <Greeting language="fr">
    <Text>Bonjour, madame</Text>
  </Greeting>
  <Greeting language="en">
    <Text>Hey, you</Text>
  </Greeting>
</Greetings>
```

Kako je već rečeno, sama specifikacija, a time i njena referentna implementacija je dosta složena, ovdje ćemo navesti samo nekoliko primjera kako se ona koristi. Zainteresirani se čitatelj svakako upućuje da detaljnije prouči literaturu navedenu u popisu referenci ovog poglavlja.

Sljedeći djelić scheme definira korištenje tipa podataka integer, no s ograničenjem da mu vrijednost može biti jedino iz skupa od 1 do 255.

```
<xsd:simpleType name="GroupType">
  <xsd:restriction base="xsd:int">
    <xsd:minInclusive value="1"/>
    <xsd:maxInclusive value="255"/>
  </xsd:restriction>
</xsd:simpleType>
```

Napomenimo da korištenje jednostavnog ograničavanja putem `xsd:int` neće rezultirati generiranjem zasebne Java klase. JAXB alat `xjc` će u tom slučaju jednostavno koristiti `int` (ili `Integer`) na svim mjestima gdje se koristi oznaka `GroupType`.

Ako sada pogledamo generirani kod, uočiti ćemo nešto zanimljivo:

```
public class ElementType {
    // ...
    @XmlAttribute(name = "Group", required = true)
    protected int group;
    // ...
    public int getGroup() {
        return group;
    }
    public void setGroup(int value) {
        this.group = value;
    }
    // ...
}
```

Vidimo da generirani kod sadrži samo jednostavne `get` i `set` metode. Ukoliko se pak pitate gdje je kod koji osigurava da je vrijednost cijelog broja unutar zadanih ograničenja (1-255), odgovor je: takav kod ne postoji!. Razlog tome zapravo je vrlo jednostavan, jer JAXB očekuje da napravite detaljnu validaciju vašeg dokumenta korištenjem `javax.xml.validation.Schema` objekta, koji nastaje iz vaše XML scheme, te ga takvog proslijedite metodama `marshal` ili `unmarshal`.

JAXB kombinira korištenje triju tipova schema za datum i vrijeme (`xsd:date`, `xsd:time`, `xsd:dateTime`) u klasi `XMLGregorianCalendar`. Ta se klasa nalazi u paketu `javax.xml.datatype`. No, s obzirom da je ta klasa apstraktna, stvaranje vrijednosti poput godine, dana ili minute nije jednostavno.

Isječak iz scheme definira elemente koje sadrže podelemente tipa `xsd:date` i `xsd:time`.

```
<xsd:complexType name="DateTimeType">
  <xsd:sequence>
    <xsd:element name="Date" type="xsd:date"/>
    <xsd:element name="Time" type="xsd:time"/>
  </xsd:sequence>
</xsd:complexType>
```

Generirana klasa sadržavati će uobičajene `get` i `set` metode:

```
public class DateTimeType {

    protected XMLGregorianCalendar date;
    protected XMLGregorianCalendar time;

    public XMLGregorianCalendar getDate() {
        return date;
    }

    public void setDate(XMLGregorianCalendar value) {
        this.date = value;
    }

    public XMLGregorianCalendar getTime() {
        return time;
    }

    public void setTime(XMLGregorianCalendar value) {
        this.time = value;
    }
}
```

Međutim, kako bi pozvali bilo koju `set` metodu, potrebno je dodati dio logike za pretvaranje. To možemo učiniti u klasi `DatatypeFactory` koja ima metode s kojima možemo stvoriti objekte tipa `XMLGregorianCalendar`.

```
// Napravi DateTimeType element za trenutno vrijeme i
datum. ObjectFactory of = new ObjectFactory();
DateTimeType meta =
of.createDateTimeType();
GregorianCalendar now = new
GregorianCalendar();

// stvori instancu DatatypeFactory.
DatatypeFactory df = DatatypeFactory.newInstance();

// Napravi XMLGregorianCalendar sa trenutnim
datumom. XMLGregorianCalendar gcDate =
    df.newXMLGregorianCalendarDate(
        now.get( Calendar.YEAR ),
```

```
now.get( Calendar.MONTH ),
now.get( Calendar.DAY_OF_MONTH ),
DatatypeConstants.FIELD_UNDEFINED );

// Napravi XMLGregorianCalendar sa trenutnim
vremenom. XMLGregorianCalendar gcTime =
    df.newXMLGregorianCalendarTime(
        now.get( Calendar.HOUR_OF_DAY ),
        now.get( Calendar.MINUTE ),
        now.get( Calendar.SECOND ),
        null,                                //nema dijelova sekunde
        DatatypeConstants.FIELD_UNDEFINED );

// Ubaci podelemente u element DateTimeType.
meta.setDate( gcDate );
meta.setTime( gcTime );
```

Upotrebom argumenta `null` prilikom konstruiranja `XMLGregorianCalendar` zapravo smo označili da nas djelići sekunde ne zanimaju. Ipak, nije moguće postaviti vrijeme na način da se sekunde u potpunosti izbjegnju.

XML element koji će biti kreiran uporabom ovog programskog koda izgleda ovako:

```
<DateTime>
  <Date>2008-07-23</Date>
  <Time>18:42:24</Time>
</DateTime>
```

Uočite da su datum i vrijeme zapisani sukladno normi ISO 8601.

Već smo spomenuli da ukoliko je za vašu primjenu potrebno validirati dokument, taj dio morate napraviti sami prije nego ga predate `unmarshal` metodi. To je moguće učiniti na način da predate objekt klase `javax.xml.validation.Schema` objektu klase `Unmarshaller`. Prvo kreirajmo taj objekt `schema` postavljajući `schema factory` na jezik `schema` po našem izboru. Tada je moguće stvoriti `Schema` objekt pozivom metode `newSchema`.

```
Schema mySchema;
SchemaFactory sf =
    SchemaFactory.newInstance( XMLConstants.W3C_XML_SCHEMA_NS_URI );
try {
    mySchema = sf.newSchema( file );
} catch( SAXException saxException ){
    // ...(error handling)
    mySchema = null;
}
```

Tako kreiranu `schema` predajemo objektu `Unmarshaller`:

```
JAXBContext jc = JAXBContext.newInstance( packagePath );
Unmarshaller u = jc.createUnmarshaller();
u.setSchema( mySchema );
```

Važno je napomenuti da u slučaju da validacija ne bude uspješna, dogoditi će se `UnmarshallerException`. Sljedeći programski odsječak omogućava rad s tom pogreškom:

```
//prije unmarshall metode
ValidationEventCollector vec = new ValidationEventCollector();
u.setEventHandler( vec );

//unutar finally faze
if( vec != null && vec.hasEvents() ){
    for( ValidationEvent ve: vec.getEvents() ){
        String msg = ve.getMessage();
        ValidationEventLocator vel = ve.getLocator();
        int line = vel.getLineNumber();
        int column = vel.getColumnNumber();
        System.err.println( origin + ": " + line + "." +
                           column + ": " + msg );
    }
}
```

Nakon što napravite vlastitu schemu, i iz nje stvorite sve klase koje su potrebne, među njima će se nalaziti i jedna pod nazivom `ObjectFactory`. Njene metode omogućuju nam jednostavno stvaranje elemenata koji moraju biti omotan objektom `JAXBElement<?>`. Za primjer uzmimo da je korijenski element dokumenta omotan sa `JAXBElement<RulebaseType>` i oznakom „rulebase“ kreiran sljedećim programskim odsječkom:

```
ObjectFactory objFact = new ObjectFactory(); RulebaseType rulebase
= objFact.createRulebaseType(); JAXBElement<RulebaseType> doc =
objFact.createRulebase( rulebase );
```

Jednostavni element koji ne zahtijeva omotač `JAXBElement<?>` može se kreirati korištenjem jednostavnog poziva odgovarajuće metode:

```
ModuleType module = objFact.createModuleType();
```

Napomenimo da je omotač potreban i kada želite koristite elemente istog tipa, ali sa različitim oznakama, kao u sljedećem odsječku scheme:

```
<xsd:complexType name="FooBarListType">
  <xsd:sequence>
    <xsd:choice minOccurs="0" maxOccurs="unbounded">
      <xsd:element name="foo" type="FooBarType"/>
      <xsd:element name="bar" type="FooBarType"/>
    </xsd:choice>
  </xsd:sequence>
</xsd:complexType>
```

U ovom slučaju, `ObjectFactory` bi sadržavao nekoliko metoda pomoću kojih bi bilo moguće stvarati potrebne elemente:

```
FooBarListType fblElem = objFact.createFooBarListType();
List<JAXBElement<FooBarType>> fbList = fblElem.getFooOrBar();

// stvori "foo" element
FooBarType foo = objFact.createFooBarType();
// ...(dodaj atribut i komponente za foo)
// kreiraj element <foo>...</foo>
JAXBElement<FooBarType> fooElem = objFact.createFooBarTypeFoo( foo );
```

```
// dodaj ih u listu roditelja.  
fbList.add( fooElem );  
  
// stvori "bar" element  
FooBarType bar = objFact.createFooBarType();  
// ...( dodaj attribute i komponente za bar)  
// kreiraj element <bar>...</bar>  
JAXBElement<FooBarType> barElem = objFact.createFooBarTypeBar( bar );  
// dodaj ih u listu roditelja.  
fbList.add( barElem );
```

Pogledajmo na kraju i uporabu oznaka na konkretnom primjeru. Definirali smo klasu Person, koja predstavlja objektni model jedne osobe. Osoba je predstavljena atributima vlastitog imena (`firstName`) i prezimena (`lastName`). Ovako to izgleda u programskom kodu:

```
package pog8;  
  
import javax.xml.bind.annotation.XmlRootElement;  
  
@XmlRootElement( )  
public class Person {  
    private String firstName;  
    private String lastName;  
  
    public String getFirstName( ) { return firstName; }  
    public String getLastName( ) { return lastName; }  
    public void setFirstName(String s) { firstName = s; }  
    public void setLastName(String s) { lastName = s; }  
}
```

Vidimo da smo u kodu definirali samo jednu oznaku, `@XmlRootElement`. Dakle, po provedenom postupku JAXB vezivanja, u rezultirajući će se XML dokument ubaciti, odmah ispod njegovoga korijenskog elementa, svaki od atributa ili svojstava koje je posjedovala izvorišna klasa (a ona može biti i JavaBean). To znači da će na kraju XML dokument izgledati otprilike ovako:

```
<?xml version="1.0" encoding="UTF-8"?>  
<person>  
    <firstName>Ivan</firstName>  
    <lastName>Horvat</lastName>  
</person>
```

Naravno, kreiranje ovog dokumenta pokrećemo iz Jave, na sljedeći način:

```
package pog8;  
  
import javax.xml.bind.JAXBContext;  
import javax.xml.bind.JAXBException;  
import javax.xml.bind.Marshaller;  
  
import com.example.person.ObjectFactory;  
import com.example.person.Person;
```

```

public class PersonMarshaller {

    public static void main(String[] args) throws JAXBException {
        ObjectFactory factory = new ObjectFactory( );
        Person person = factory.createPerson( );
        person.setFirstName("Ivan");
        person.setLastName("Horvat");

        JAXBContext context = JAXBContext.newInstance("pog8.person");
        Marshaller marshaller = context.createMarshaller( );
        marshaller.marshal(person, System.out);
    }
}

```

Vidimo da ovim odlomkom koda stvaramo novi `JAXBContext` objekt, korištenjem metode `newInstance`, kojoj kao parametar predajemo referencu na klasu na temelju koje želimo kreirati XML dokument. Nakon toga instanciramo objekt `Marshaller`, koji ima ugrađenu metodu kojom se generira XML (metoda `marshal()`). Mogli smo se odlučiti za neku klasu koja obavlja ispis (iz grupe `Writer`¹⁹³ klasa), no ovaj jednostavni primjer rezultat će ispisati na standardnom izlazu.

4.2 Poznatije primjene označnih jezika

U ovom ćemo potpoglavlju navesti neke rasprostranjene primjene označnih jezika kako biste stekli uvid o jednostavnosti korištenja i mogućnostima označnih jezika u kontekstu interoperabilnosti. Napominjemo da prikazani označni jezici nisu najznačajniji i najrasprostranjeniji primjeri korištenja, prije svega zato što takva usporedba ni ne postoji, već su izabrani kao predstavnici nekih poznatijih usluga, kao što su sindikacija sadržaja i elektroničko poslovanje.

4.2.1 Univerzalni poslovni jezik UBL

Univerzalni poslovni jezik UBL (engl. *Universal Business Language*) opisuje biblioteku poslovnih dokumenata elektroničkog poslovanja u obliku XML, kao što su primjerice narudžbe i računi. UBL zapravo donosi pravila strukture i zapisa poslovnih elektroničkih dokumenata u obliku XML shema koji se posljedično primjenjuju za ispitivanje valjanosti dokumenata. UBL je razvila tehnička komisija OASIS-a¹⁹⁴ kao prvu implementaciju specifikacije ebXML CCTS (engl. *Core Components Technical Specification*) zasnovanu na konceptualnom modelu informacijskih komponenti (engl. *Business Information Entities*). Kao što je objašnjeno u poglavlju 2.5.2, norma CCTS je postala i norma ISO/TS 15000-5:2005¹⁹⁵, koja osim poslovnih dokumenata opisuje i tehničke specifikacije osnovnih komponenti. Prva inačica UBL 1.0

¹⁹³ <http://download-l1nw.oracle.com/javase/1.5.0/docs/api/java/io/Writer.html>

¹⁹⁴ OASIS Universal Business Language (UBL) TC, <https://www.oasis-open.org/committees/ubl>

¹⁹⁵ ISO/TS 15000-5:2005 Electronic Business Extensible Markup Language (ebXML) -- Part 5: ebXML Core Components Technical Specification, Version 2.01 (ebCCTS)

donesena je 2004. godine, trenutno je još uvijek aktualna inačica UBL 2.0¹⁹⁶ iz 2006. godine, Godine 2018. objavljena je aktualna verzija UBL 2.2.¹⁹⁷

Komponente koje odgovaraju poslovnim entitetima, kao što je primjerice račun, pretvaraju se prema odgovarajućim XML shemama i pravilima za imenovanje i oblikovanje UBL NDR¹⁹⁸ (engl. *UBL Naming and Design Rules*) u odgovarajuće XML dokumente. UBL zapravo sadržava biblioteku XML shema osnovnih i agregiranih podatkovnih komponenti od kojih se grade poslovni dokumenti, odnosno njihove XML sheme. Tako se primjerice poslovni dokument računa gradi od niza komponenti, a za svaki poslovnih proces definiraju se tipovi dokumenata koji omogućavaju izvođenje tih procesa u obliku elektroničkog poslovanja.

Još je UBL 1.0 donio 8 osnovnih tipova dokumenata, koji su kasnije osvježeni, a u inačici UBL 2.0 svjedočili smo pojavi 23 nova tipa dokumenta, koja pokrivaju područje procesa nabave, narudžbe, fakturiranja, isporuke, plaćanja, prijevoza te dodatnih izvještaja. Iako tehnički ne donosi neke novosti, prijedlog norme UBL 2.1 donosi čak 34 nova tipa dokumenta, što sveukupno znači 65 tipova poslovnih dokumenata.

Napomena: Specifikacije tipova dokumenata norme UBL 2.0 i prijedloga norme UBL 2.1 nisu unutražno kompatibilni s izvornim dokumentima norme UBL 1.0, a svi tipovi dokumenata norme UBL 1.0 su osvježeni u novim inačicama.

Popis **tipova dokumenata** koje obrađuje aktualna norma UBL 2.0 su sljedeći:

- tipovi dokumenata naslijeđeni iz UBL 1.0:
 - Order – narudžba
 - Order Response – odgovor na narudžbu
 - Order Response Simple – jednostavan odgovor na narudžbu
 - Order Change – izmjena narudžbe
 - Order Cancellation – otkaz narudžbe
 - Despatch Advice – otpremnica
 - Receipt Advice – primka
 - Invoice – račun
- katalogiziranje i navođenje cijena (engl. *Sourcing*):
 - Catalogue – katalog
 - Catalogue Deletion – brisanje kataloga
 - Catalog Item Specification Update – osvježavanje
 - Catalog Pricing Update – osvježavanje cijena kataloga
 - Catalog Request – zahtjev za katalogom
 - Quotation – ponuda
 - Request for Quotation – zahtjev za ponudu
- **isporuka** (engl. *Fulfillment*):
 - Bill of Lading – teretnica

¹⁹⁶ Universal Business Language v2.0, OASIS Standard, 2006-12-12, <http://docs.oasis-open.org/ubl/os-UBL-2.0/>

¹⁹⁷ Universal Business Language v2.2, OASIS Standard, 2018-07-09, <http://docs.oasis-open.org/ubl/os-UBL-2.2/UBL-2.2.html>

¹⁹⁸ Universal Business Language Naming & Design Rules v1.0 (UBL NDR), 2004-11-15, OASIS <http://www.oasis-open.org/committees/download.php/10323/cd-UBL-NDR-1.0Rev1c.pdf>

- Certificate of Origin – certifikat o porijeklu robe
- Forwarding Instructions – upute za otpremu
- Packing List – spisak pakiranja
- Transportation Status – prijevozni status
- Waybill – teretni/tovarni list
- izdavanje računa (engl. *Billing*)
 - Credit Note – odobrenje
 - Debit Note – zaduženje
 - Freight Invoice – račun za prijevoz
 - Reminder – opomena
 - Self Billed Credit Note – samofakturirano odobrenje
 - Self Billed Invoice – samofakturirani račun
 - Remittance Advice – obavijest o plaćanju
 - Statement – stanje računa
- **dodatni tipovi** (engl. *Supplementary types*)
 - Application Response – odgovor aplikacije
 - Attached Document – dokument u privitku

S obzirom da će prijedlog norme UBL 2.1 vjerojatno uskoro biti prihvaćen, donosimo popis i 34 dokumenta koja donosi:

- **e-Nadmetanje** (engl. *eTendering*)
 - Awarded Notification
 - Call for Tenders
 - Contract Award Notice
 - Contract Notice
 - Guarantee Certificate
 - Tender
 - Tender Receipt
 - Tenderer Qualification
 - Tenderer Qualification Response
 - Unawarded Notification
- **kolaborativno planiranje prognoziranje i nadopunjavanje** (engl. *Collaborative planning, forecasting and replenishment*)
 - Exception Criteria
 - Exception Notification
 - Forecast
 - Forecast Revision
 - Item Information Request
 - Prior Information Notice
 - Trade Item Location Profile
- **roba upravljana od dobavljača** (engl. *Vendor Managed Inventory*)
 - Instruction for Returns
 - Inventory Report
 - Product Activity
 - Retail Event
 - Stock Availability Report
- **intermodalno upravljanje prijevozom** (engl. *Intermodal Freight Management*)

- Goods Item Itinerary
- Transport Execution Plan
- Transport Execution Plan Request
- Transport Progress Status
- Transport Progress Status Request
- Transport Service Description
- Transport Service Description Request
- Transportation Status
- Transportation Status Request
- izdavanje računa za komunalije (engl. *Utility billing*)
 - Utility Statement
- **dodatni tipovi** (engl. *Supplementary types*)
 - Document Status
 - Document Status Request

Kako bismo bolje razumjeli sadržaj poruka prikazat ćemo primjer jednu UBL poruke narudžbe, u kojem se mogu vidjeti svi važni podaci narudžbe:

```
<po:Order xmlns:po="urn:oasis:names:tc:ubl:Order:1.0:0.70"
  xmlns="urn:oasis:names:tc:ubl:CommonAggregateTypes:1.0:0.70">
  <ID>1000000001</ID>
  <IssueDate>2010-09-01</IssueDate>
  ...
  <BuyerParty>
    <ID>A001</ID>
    <PartyName>
      <Name>Moja tvrtka d.o.o.</Name>
    </PartyName>
    <Address>
      <ID>X003</ID>
      <Street>Ilica 1111</Street>
      <CityName>Zagreb</CityName>
      <CountrySub-EntityCode listID="3166-2" listAgencyID="ISO">HR-21
      </CountrySub-EntityCode>
      <Country>
        <Code listID="3166-1" listAgencyID="ISO">HR</Code>
      </Country>
    </Address>
    <BuyerContact>
      <ID></ID>
      <Name>Ana Anić</Name>
    </BuyerContact>
  </BuyerParty>
  ...
  <OrderLine>
    <BuyersID>B001</BuyersID>
    <Quantity unitCode="unit">10</Quantity>
    <Item>
      <ID>C100001</ID>
      <Description>Priručnik IIS</Description>
    </Item>
    <BasePrice>
      <PriceAmount currencyID="HRK">10</PriceAmount>
    </BasePrice>
  </OrderLine>
</po:Order>
```

U prethodnoj poruci, osim identifikatora narudžbe <ID> i datuma <IssueDate>, postoji dio o kupcu <BuyerParty> i dio o sadržaju narudžbe <OrderLine>. Podaci o kupcu sadrže identifikator kupca <ID>, naziv <PartyName> i <Name> te adresu <Address> sa svima podacima. Sadržaj narudžbe sadrži identifikator <BuyersID>, podatke o količini <Quantity> i o samom predmetu narudžbe <Item> sa svim podacima..

Od inačice UBL 2.0 omogućena je izvedba podskupova, odnosno shema, pa su tako izvedeni nacionalni podskup sjevernoeuropskih zemalja NESUBL (engl. *Northern European Subset UBL*), koji razrađuje semantičku uporabu pojedinih poslovnih procesa, španjolska inačica vezana uz procese elektroničkog računa, danska inačica OIOUBL, također vezana uz elektronički račun, turska lokalizacija UBLTR i slovenska inačica E-SLOG. Zbog kašnjenja u isporuci dijela normi i shema koje su povezane s UBL-om, UN/CEFACT je prihvatio postojeću normu UBL 2.0 na kojoj će zasnivati nove XML sheme za elektroničko poslovanje.

Danas postoji implementacija UBL 2.0 biblioteke za .NET Framework, C++ i srodne tehnologije nazvana SimpleUBL¹⁹⁹, a UBL-ove sheme mogu se koristiti i u okruženju platforme Java uporabom tehnologije JAXB, odnosno prevođenjem sheme u klase jezika Java.

4.2.2 Norma GS1 XML

Normu **GS1 XML** razvila je organizacija GS1, a propisuje strukturu poslovnih poruka u jeziku XML, definira XML sheme i scenarije razmjene poruka. Ovo je novija norma koju nudi organizacija GS1, a postoji i starija norma EANCOM, temeljena na normi EDIFACT. Danas postoji puno više korisnika starije norme na globalnoj razini, pa ih se potiče da prijeđu s norme EANCOM na GS1 XML, a o tome je već bilo riječi u potpoglavlju 2.4.15.

No, iako je norma GS1 XML dostupna i javno objavljena, njeno se korištenje uvjetuje članstvom u organizaciji GS1. Po učlanjenju mogu se početi koristiti već spomenuti ključevi za **globalni broj lokacije GLN** (engl. *Global Location Number*) i ključevi pojedinih artikala u obliku **globalnog broja trgovinskog artikla GTIN** (engl. *Global Trade Item Number*), koje dodjeljuje organizacija GS1. To znači da ipak samo organizacija GS1 može definirati te brojeve, a ne sami korisnici, pa je otvorenost norme GS1 XML upitna. S druge strane, ovo je omogućilo da GS1 održava globalne kataloge brojeva GLN i GTIN, pa se podaci o pojedinim subjektima i artiklima ne prenose u svakoj poruci, već samo njihovi identifikacijski brojevi GLN ili GTIN. Elektronička poruka koja sadržava račun tako je mnogo jednostavnija, ali pretpostavlja uporabu globalnih kataloga proizvoda od GS1.

U smislu prihvaćenosti, norma GS1 XML te brojevi GLN i GTIN najrašireniji su u trgovini, dok se u drugim granama gospodarstva još uvijek nadopunjuju brojevi koji označavaju proizvode, robu i usluge koji nisu toliko prisutni u elektroničkom poslovanju. Većina proizvođača softvera danas podržava sintaksu jezika XML te nije potrebna dodatna konverzija poruka. Osim toga podrška za jezik XML dio je najpoznatijih programskih jezika i platformi, kao što su Java i .NET, a i sustavi informacijske sigurnosti prilagođeni su porukama u obliku XML.

¹⁹⁹ SimpleUBL, <http://www.simpleubl.com/products/universal-business-language-implementation-library/>

U Hrvatskoj postoje implementacije eCROKAT i e-CRODOK. eCROKAT je hrvatski elektronički katalog trgovačkih jedinica i partnera. eCROKAT predstavlja čvor u globalnoj mreži elektroničkih kataloga GDSN (engl. *Global Data Synchronisation Network*) za automatsko sinkroniziranje matičnih podataka. Korištenjem eCROKAT-a omogućeno je izravno razmjenjivanje (engl. *machine to machine*) matičnih podataka o trgovačkim jedinicama i partnerima uporabom globalne tehnološke infrastrukture GDSN, koja je *de facto* norma za razmjenu matičnih podataka. Podržava različite tipove poslovnih dokumenata, kao što su narudžba, potvrda narudžbe, otpremnica, potvrda otpremnice, povratnica i račun. U Hrvatskoj veliki trgovački lanci najviše koriste staru normu EANCOM i ne može se očekivati da će tako brzo prijeći na novu normu GS1 XML. No zbog zastarjelosti starije norme EANCOM, za razmjenu poruka elektroničkog poslovanja ipak se preporučuje norma GS1 XML. No zasad treba podržati obje norme, potencijalno uz usluge informacijskih posrednika koji su u mogućnosti ponuditi konverzije iz EDIFACT poruka u poruke u jeziku XML i obrnuto.

4.2.3 OpenDocument Format (ODF) i Office Open XML (OOXML)

Kada govorimo o oblicima zapisa poslovnih dokumenata, prvo pomislimo na najčešće poslovne dokumente u uredskom poslovanju. To su ponajprije tekstualni dokumenti koji se izrađuju uređivačima ili alatima za obradu teksta (engl. *word processor*), zatim tablični izračuni koje izrađuju tablični kalkulatori (engl. *spreadsheet*) i prezentacije koje izrađuju alati za izradu prezentacija. Glavna asocijacija kod skupa programa za podršku uredskom poslovanju je poznati i rašireni Microsoftov komercijalni paket Office, ali on nikako nije jedini koji postoji na tržištu, već postoji niz drugih komercijalnih i besplatnih aplikacija i uredskih paketa. Osim navedene tri uredske aplikacije, postoji i niz drugih pomoćnih programa koji služe za izradu različitih grafičkih elemenata i pohranu podataka u drugim oblicima koji se isporučuju u takvim skupovima ili paketima uredskih aplikacija.

Kod paketa Microsoft Office, od inačice 2007 uveden je oblik zapisa zasnovan na jeziku XML, koji je nazvan **Office Open XML**, odnosno skraćeno **OOXML** ili **OpenXML**. Prvo ga je normirala industrijska udruga ECMA²⁰⁰, da bi kasnije bio normiran i kao međunarodna norma ISO/IEC 29500:2008. I prije oblika zapisa OOXML, Microsoft je pokušavao uvesti druge oblike zapisa zasnovane na jeziku XML, kao što je zapis tabličnog izračuna za aplikaciju Excel u paketu Office XP ili zapis tekstualnog dokumenta za aplikaciju Word u paketu Office 2003, koji su bili zasnovani na XML shemama, ali se od njih odustalo u inačici paketa Office 2007. Prvu verziju norme OOXML odobrila je udruga ECMA još 2006. godine, ali sam proces normiranja kao norma ISO/IEC bio je dosta kontroverzan jer je prijedlog norme često kritiziran^{201, 202} kao nepotreban s obzirom na to da je u međuvremenu nastao manje složen oblik zapisa koji je prihvaćen kao međunarodna otvorena norma, nazvan OpenDocument Format.

²⁰⁰ Standard ECMA-376, Office Open XML File Formats, ECMA, Second ed., 2008, <http://www.ecma-international.org/publications/standards/Ecma-376.htm>

²⁰¹ Kirk, Jeremy, ISO publishes Office Open XML specification, InfoWorld, 2008, <http://www.infoworld.com/t/applications/iso-publishes-office-open-xml-specification-918>

²⁰² Stallman, Richard, We Can Put an End to Word Attachments, 2009, <http://www.gnu.org/philosophy/no-word-attachments.html>

Norma **ISO/IEC 29500:2008 Information technology – Document description and processing languages – Office Open XML File Formats** sastoji se od četiriju dijelova, od kojih su prva tri neovisne norme, dok je četvrti modifikacija prvog dijela, kako slijedi:

- **1. dio – Osnove i reference označnog jezika** (engl. *Fundamentals and Markup Language Reference*) – sadržava definicije usklađivanja, XML sheme u XSD i RELAX NG obliku i druge osnovne informacije na 5560 stranica,
- **2. dio – Konvencije otvorenog pakiranja** (engl. *Open Packaging Conventions*) – sadržava opis otvorenog pakiranja, osnovne značajke, elektroničke potpise i XML sheme za pakiranje na 129 stranica,
- **3. dio – Kompatibilnost i nadogradnja oznaka** (engl. *Markup Compatibility and Extensibility*) – sadržava opise nadogradnji na 40 stranica,
- **4. dio – Značajke prijelazne migracije** (engl. *Transitional Migration Features*) – sadržava upute za kompatibilnost i razlike normi na 1464 stranice.

Danas je norma ISO/IEC 29500 nacionalno prihvaćena kod određenog broja zemalja, a kod nekih se tek razmatra kao nacionalna norma oblika zapisa dokumenata u javnoj upravi. Oblik zapisa OOXML podržava²⁰³ i niz drugih uredskih paketa, kao što su OpenOffice, KOffice, Joffice, SoftMaker, Google Docs, IBM Lotus Notes i mnoge druge aplikacije, te informacijski sustavi usko vezani uz uredsko poslovanje i razmjenu podataka. Dokumenti u zapisu OOXML prepoznatljivi su po sufiksima .docx i .docm za tekstualne dokumente, .pptx i .pptm za prezentacije te .xlsx i .xslm za tablične izračune. Svi oblici koriste kompresiju ZIP.

Drugi važan oblik zapisa uredskih dokumenata je **OpenDocument Format** ili skraćeno **ODF**. To je također oblik zapisa za tekstualne dokumente, tablične izračune, prezentacije i grafičke elemente temeljen na jeziku XML. Inicijalno ga je razvila tvrtka Sun Microsystems, uz podršku drugih velikih tvrtki, kao što je IBM, da bi kasnije razvojem upravljala tehnička komisija OASIS-a²⁰⁴ temeljem čega je i prihvaćen kao otvorena norma. Oblik zapisa ODF je 2006. godine postao i međunarodna norma **ISO/IEC 26300:2006 Information technology -- Open Document Format for Office Applications (OpenDocument)**²⁰⁵ u inačici 1.0. U 2007. godini izvedena je i inačica 1.1, koja nije predložena kao međunarodne norme, a upravo se razvija inačica 1.2 koja bi trebala dodati neke nove značajke, kao što su metapodaci u RDF-u, zapis OpenFormula i podršku za elektronički potpis.

Danas je norma ISO/IEC 26300 prihvaćena kod velikog broja zemalja kao nacionalna norma oblika zapisa dokumenata²⁰⁶ u javnoj upravi, pa tako i u velikom dijelu EU-a, a prihvatio ga je i NATO. U Hrvatskoj je 2008. godine ODF prihvaćen kao hrvatska norma **HRN ISO/IEC 26300:2008²⁰⁷ Informacijska tehnologija - Otvoreni format dokumenta za uredsku primjenu (OpenDocument) v1.0.**, što je u skladu s Odrednicama razvitka i uporabe

²⁰³ List of software that supports Office Open XML, Wikipedia,

http://en.wikipedia.org/wiki/List_of_software_that_supports_Office_Open_XML

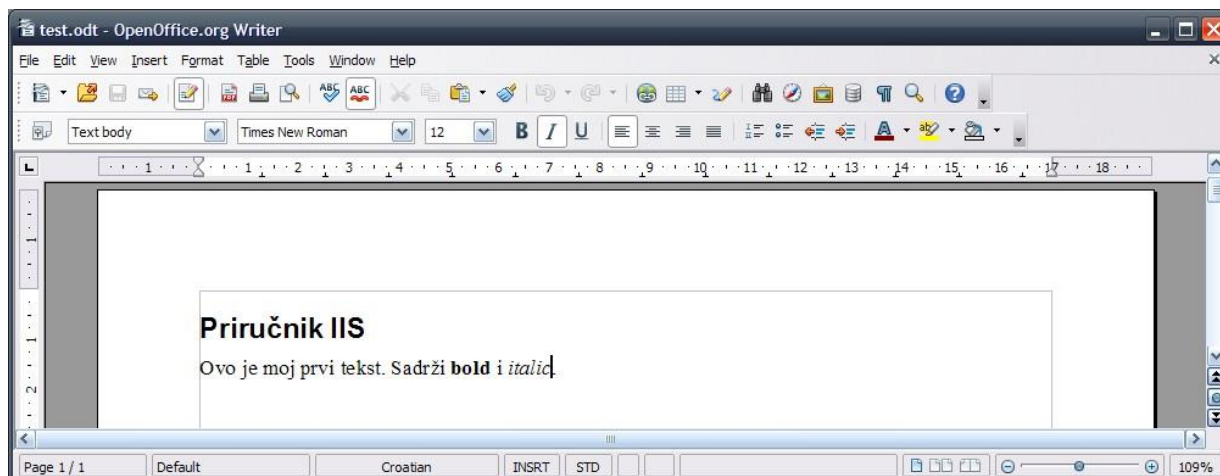
²⁰⁴ OASIS Open Document Format for Office Applications (OpenDocument) TC, OASIS, http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=office

²⁰⁵ ISO/IEC 26300:2006 Information technology - Open Document Format for Office Applications (OpenDocument) v1.0, http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=43485

²⁰⁶ OpenDocument adoption, Wikipedia, http://en.wikipedia.org/wiki/OpenDocument_adoption

²⁰⁷ HRN ISO/IEC 26300:2008 Informacijska tehnologija -- Otvoreni format dokumenta za uredsku primjenu(OpenDocument) v1.0

računalnih programa s otvorenim kodom u tijelima državne uprave²⁰⁸. Aplikacije i uredski paketi koji podržavaju oblik zapisa ODF uključuju i OpenOffice.org, Corel WordPerfect Office, WordPad, SoftMaker Office, Sun StarOffice, NeoOffice, Microsoft Office 2007 SP 2 i 2010, KOffice, IBM Lotus Symphony, Google Docs i mnogi drugi. OpenDocument Format Alliance je organizacija koju su osnovali IBM, Sun i druge velike tvrtke, a promiče uporabu zapisa ODF. Dokumenti u zapisu ODF prepoznatljivi su po sufiksima .odt za tekstualne dokumente, .ods za tablične izračune, .odp za prezentacije, .odg za grafiku i .odf za matematičke i druge formule. Svi oblici zapisa mogu koristiti kompresiju ZIP. Pokušajmo sad izraditi jedan jednostavni tekstualni dokument kao na sljedećem prikazu (Slika 4.4) i pogledati kako se interno zapisuje.



Slika 4.4: Primjer tekstualnog dokumenta u normi ODF

Stvorena datoteka pohranjena na disk sadržavat će sljedeće mape i datoteke sažete ZIP kompresijom:

- Configurations2 – mapa koja sadržava niz podmapa i datoteka s konfiguracijama,
- META-INF – mapa s datotekom manifest.xml koja sadržava popis strukture ostalih mapa,
- Thumbnails – mapa s malim sličicama izgleda za brzi pregled dokumenta,
- content.xml – sadržaj tekstualnog dokumenta,
- meta.xml – značajke tekstualnog dokumenta,
- mimetype – MIME tip dokumenta (application/vnd.oasis.opendocument.text),
- settings.xml – postavke tekstualnog dokumenta,
- styles.xml – stilovi tekstualnog dokumenta.

Ako otvorimo sadržaj tekstualnog dokumenta u datoteci content.xml, učit ćemo sadržaj sličan sljedećem:

```
<?xml version="1.0" encoding="UTF-8"?>
<office:document-content
xmlns:office="urn:oasis:names:tc:opendocument:xmlns:office:1.0"
xmlns:style="urn:oasis:names:tc:opendocument:xmlns:style:1.0"
```

²⁰⁸ Odrednice razvitka i uporabe računalnih programa s otvorenim kodom u tijelima državne uprave, e-Hrvatska, Vlada RH, 2006., http://e-hrvatska.hr/sdu/hr/ProgramEHrvatska/Provedba/StrategijeIProgrami/categoryParagraph/01110/document/OSSpolicy_Odrednice.pdf

```

xmlns:text="urn:oasis:names:tc:opendocument:xmlns:text:1.0"
xmlns:table="urn:oasis:names:tc:opendocument:xmlns:table:1.0"
xmlns:draw="urn:oasis:names:tc:opendocument:xmlns:drawing:1.0"
...
office:version="1.2">
  <office:scripts/>
  <office:font-face-decls>
    ...
    <style:font-face style:name="Arial" svg:font-family="Arial"
      style:font-family-generic="swiss" style:font-pitch="variable"/>
    ...
  </office:font-face-decls>
  <office:automatic-styles>
    <style:style style:name="T1" style:family="text">
      <style:text-properties fo:font-weight="bold"
        style:font-weight-asian="bold" style:font-weight-complex="bold"/>
    </style:style>
    <style:style style:name="T2" style:family="text">
      <style:text-properties fo:font-style="italic"
        style:font-style-asian="italic" style:font-style-complex="italic"/>
    </style:style>
  </office:automatic-styles>
  <office:body>
    <office:text>
      <text:sequence-decls>
        ...
        <text:sequence-decl text:display-outline-level="0"
          text:name="Text"/>
        ...
      </text:sequence-decls>
      <text:h text:style-name="Heading_20_1" text:outline-level="1">
        Priručnik IIS
      </text:h>
      <text:p text:style-name="Text_20_body">
        Ovo je moj prvi tekst. Sadrži
        <text:span text:style-name="T1">bold</text:span>
        i <text:span text:style-name="T2">italic</text:span>
        .</text:p>
      </office:text>
    </office:body>
  </office:document-content>

```

Ako promotrimo sadržaj u datoteci content.xml, učit ćemo upisani tekst, naslov, prvi odlomak i dijelove teksta označene *bold* i *italic*. Jednako se tako zapisuju i vrlo složeni dokumenti.

4.2.4 Elektronički obrazac platnog naloga (e-HUB)

Još 2003. godine Odbor za platni promet Hrvatske udruge banaka²⁰⁹ (HUB) je, sukladno Odluci o nalogima za plaćanje²¹⁰ HNB-a, standardizirao novi obrazac naloga za plaćanje, koji se koristi za uplate, isplate i prijenose. Izdana je Uputa o obliku, sadržaju i upotrebi obrazaca²¹¹,

²⁰⁹ Hrvatska udruga banaka (HUB), <http://www.hub.hr>

²¹⁰ Odluka o nalogima za plaćanje Hrvatske narodne banke, Narodne novine, br. 14/2002

²¹¹ Uputa o obliku, sadržaju i upotrebi naloga za plaćanje u domaćem platnom prometu, Narodne novine, br.77/03.

koja opisuje obrazac HUB 1 namijenjen ručnom ili strojnom ispunjavanju i obrazac HUB 1-1 namijenjen isključivo strojnom ispunjavanju, te su predstavljena pravila ispravnog ispunjavanja obrazaca. Obličje obrazaca HUB je zaštitio pri Zavodu za intelektualno vlasništvo RH, tako da svim korisnicima, npr. bankama i drugima koji žele tiskati naloge, HUB izdaje odobrenje za korištenje obličja obrazaca HUB 1 i HUB 1-1 ako obrasci zadovoljavaju sva tehnička svojstva iz Upute.

Nakon početka primjene novog Zakona o platnom prometu²¹² usklađenog s europskom direktivom PSD (engl. *Payments Services Directive*) donesena je i Odluka o nalogima za plaćanje²¹³ koja uvodi **standardizirani oblik broja računa IBAN**, obavezan od lipnja 2013. godine. Pristupanja Hrvatske EU izjednačuje se i način obavljanja transakcija u domaćoj valuti i eurima u Hrvatskoj (nacionalne platne transakcije) s načinom obavljanja prekograničnih transakcija (unutar EU), dok su za manji dio međunarodnih transakcija (izvan EU) i dalje biti potrebni dodatni podaci, iako u manjem obujmu nego danas.

Istovremeno je predstavljen i novi obrazac platnog prometa u verziji **HUB 3** (Slika 4.5) i **HUB 3^a** (Slika 4.6), koji sadrži i prikladan prostor za dvodimenzionalni crtični kôd (engl. *2D Bar code*).

Forma UNIVERZALNI NALOG ZA PLAĆANJE (HUB 3) sadrži sljedeće polja:

- PLATITELJ (naziv/ime i adresa):** Polje za unos podataka platitelja.
- IBAN ili broj računa primatelja:** Polje za unos IBAN broja računa primatelja.
- PRIMATELJ (naziv/ime i adresa):** Polje za unos podataka primatelja.
- BIC i/ili naziv banke primatelja:** Polje za unos BIC i/ili naziva banke primatelja.
- Primatelj (osoba):** Polje za unos podataka o primatelju (osoba).
- Fizička / Pravna:** Polja za označavanje vrste primatelja (fizička ili pravna).
- Pečat korisnika PU:** Polje za unos pečata korisnika PU.
- Potpis korisnika PU:** Polje za unos potpisa korisnika PU.
- Valuta pokriva:** Polje za unos valute pokriva.
- Troškovna opcija:** Polje za unos troškovne opcije (BEN, SHA, OUR).
- Hitno:** Polje za označavanje hitnosti.
- Valuta plaćanja:** Polje za unos valute plaćanja.
- Iznos:** Polje za unos iznosa.
- IBAN ili broj računa platitelja:** Polje za unos IBAN broja računa platitelja.
- Model:** Polje za unos modela.
- Poziv na broj platitelja:** Polje za unos poziva na broj platitelja.
- Model:** Polje za unos modela.
- Poziv na broj primatelja:** Polje za unos poziva na broj primatelja.
- Šifra namjene:** Polje za unos šifre namjene.
- Opis plaćanja:** Polje za unos opisa plaćanja.
- Datum izvršenja:** Polje za unos datuma izvršenja.

Obr. HUB 3 -

Slika 4.5: Obrazac HUB 3 u prirodnoj veličini

²¹² Zakon o platnom prometu, Narodne novine, br. 133/2009 i 136/2012

²¹³ Odluku o nalogima za plaćanje, Narodne novine, br. 3/2011

NALOG ZA NACIONALNA PLAĆANJA			
PLATITELJ (naziv/ime i adresa):	Hibrid: ☐	Valuta plaćanja:	Iznos:
	IBAN ili broj računa platitelja:		
	Model:	Poziv na broj platitelja:	
	IBAN ili broj računa primatelja:		
PRIMATELJ (naziv/ime i adresa):	Model:	Poziv na broj primatelja:	
	Šifra namjene:	Opis plaćanja:	
	Datum izvršenja:		
	Pečat korisnika PU:		
Potpis korisnika PU:		Valuta i iznos:	
		IBAN (račun) platitelja ili Platielja:	
		Model i poziv na broj platitelja:	
		IBAN (račun) primatelja:	
		Model i poziv na broj primatelja:	
		Opis plaćanja:	
		Ovjera:	

Slika 4.6: Obrazac HUB 3A

Ponovno je drugi oblik obrasca HUB 3A namijenjen za strojni ispis i samo za nacionalna plaćanja, a napravljen je i **interaktivni elektronički oblik obrasca u PDF obliku** koji se može ispuniti u čitaču PDF dokumenata. Osim novih obrazaca uvedeno je i korištenje tzv. šifara namjene, odnosno svrha plaćanja u obliku šifara od 4 znaka, u skladu s normom ISO 20022²¹⁴. Također je krajem 2012. godine uveden normirani 2D crtični kod u obliku zapisa PDF417, u skladu s normom ISO/IEC 15438²¹⁵ i usklađen sa standardom HUB 3, a koji se zapisuje u donjem lijevom kutu obrasca.



Slika 4.7: 2D crtični kod PDF417

Još je 2008. godine, u sklopu projekta e-račun Radna skupina HUB-a prema obrascu HUB1 izradila **elektronički obrazac e-HUB** u jeziku XML, koji je namijenjen svim sudionicima platnog prometa koji koriste e-poslovanje i e-račun. Standard formata obrasca e-HUB²¹⁶ propisuje format u jeziku XML, opis svakog pojedinog atributa aplikacijske sheme (Tablica 4.2) te izgled obrasca nakon prikaza na zaslonu ili ispisa na pisaču (ako želimo vidjeti strukturu). Standard je otvoren i dostupan za slobodnu upotrebu svim zainteresiranim korisnicima, a po potrebi će se razvijati, mijenjati i nadopunjavati u skladu sa zakonskim promjenama i zahtjevima tržišta.

²¹⁴ ISO 20022 Financial Services - Universal financial industry message scheme, <http://www.iso20022.org/>

²¹⁵ ISO/IEC 15438:2006 Information technology -- Automatic identification and data capture techniques -- PDF417 bar code symbology specification

²¹⁶ Standard formata e-HUB obrasca, HUB, 2008., <http://www.hub.hr/fqs.axd?id=3172>

Naziv atributa	Duljina	Vrsta	Nužno za izdavatelja	Nužno za primatelja	Napomena	
Hitnost	2	n			Prioritet izvršenja	
Kanal izvršenja	1	a			N = normalno (NKS), H = hitno (HSVP)	
Oznaka valute	3	a	da	da	Inicijalno	
Iznos	13+2		da	da		
Datum			da	da	Norma ISO	
Podaci o platitelju						
Naziv	50	an	da	da		
Ulica	25	an				
Kućni broj	5	an				
Mjesto	25	an				
Poštanski broj	5	n				
Država	2	an			Inicijalno HR	
Broj računa IBAN	34	an		da	2+2+7+10 (HR, kont.br, VBDI, br.	
Model	2	n				
PNBR zaduženja	22	an				
Podaci o primatelju						
Naziv	50	an	da	da		
Ulica	25	an				
Kućni broj	5	an				
Naziv atributa	Duljina	Vrsta	Nužno za izdavatelja	Nužno za primatelja	Napome	
Mjesto	25	an				
Poštanski broj	5	n				
Država	2	an				
Broj računa IBAN	34	an	da	da		
Model	2	n				
PNBR zsaduženja	22	an				
Opis plaćanja (svrha)	105	an			Nije obavezno	

Tablica 4.2: Opis atributa aplikacijske sheme obrasca e-HUB

Elektronički obrazac e-HUB može se prikazati na zaslonu i ispisati, a osim preporučenog izgleda (sljedeća slika) prema standardu²¹⁷, neke ga banke prikazuju vrlo slično obrascu HUB 3, i to najčešće u crvenoj boji, najviše zbog klijenata naviklih na papirnate obrasce HUB 3 te zbog uniformnosti izgleda.

²¹⁷ Standard formata e-HUB obrasca, HUB, 2008., <http://www.hub.hr/fqs.axd?id=3172>

Obrazac e-HUB

Transakcija hitnost ☐ kanal izvršenja ☐ oznaka valute HRK
 iznos datum valute

Platitelj naziv tvrtke ili ime i prezime
 adresa ulica kućni broj
 mjesto poštanski broj
 država HR Republika Hrvatska
 broj računa (IBAN) HR
 model i poziv na broj zaduženja model poziv na broj

Ovjera platitelja

Primatelj naziv tvrtke ili ime i prezime
 adresa ulica kućni broj
 mjesto poštanski broj
 država HR Republika Hrvatska
 broj računa (IBAN) HR
 model i poziv na broj odobrenja model poziv na broj
 opis plaćanja

Slika 4.8: Izgled obrasca e-HUB nakon prikaza na zaslonu ili ispisa na pisaču

Ako želimo vidjeti strukturu jednog dokumenta e-HUB možemo pogledati sljedeći XML:

```
<?xml version="1.0" encoding="utf-8"?>
<e-HUB>
  <podaci>
    <zaglavlje>
      <ISOkodDrzave>HR</ISOkodDrzave>
      <nazivObrasca>e-HUB</nazivObrasca>
    </zaglavlje>
    <transakcija>
      <hitnost />
      <kanalIzvršenja />
      <oznakaValute>HRK</oznakaValute>
      <iznos />
      <datumValute />
    </transakcija>
    <platitelj>
      <naziv />
      <adresa>
        <ulica />
        <kucniBroj />
        <mjesto />
        <postanskiBroj />
        <ISOkodDrzave>HR</ISOkodDrzave>
        <nazivDrzave>Republika Hrvatska</nazivDrzave>
      </adresa>
      <IBAN>
        <ISOkodDrzave>HR</ISOkodDrzave>
        <kontrolniBroj />
        <VBDI />
        <brojRacuna />
      </IBAN>
      <PNB_Zaduzenje>
```

```

        <model />
        <pozivNaBroj />
    </PNB_Zaduzenje>
</platitelj>
<ovjeraPlatitelja
    xmlns:xfa="http://www.xfa.org/schema/xfadata/1.0/"
    xfa:dataNode="dataGroup" />
<primatelj>
    <naziv />
    <adresa>
        <ulica />
        <kucniBroj />
        <mjesto />
        <postanskiBroj />
        <ISOkodDrzave>HR</ISOkodDrzave>
        <nazivDrzave>Republika Hrvatska</nazivDrzave>
    </adresa>
    <IBAN>
        <ISOkodDrzave>HR</ISOkodDrzave>
        <kontrolniBroj />
        <VBDI />
        <brojRacuna />
    </IBAN>
    <PNB_Odobrenje>
        <model />
        <pozivNaBroj />
    </PNB_Odobrenje>
    <opisPlacanja />
</primatelj>
</podaci>
</e-HUB>

```

Dosad nije zamijećen razvoj otvorenog validatora za e-HUB obrazac, pretpostavljeno u obliku XML sheme, ali poznato je da mnoge veće banke u Hrvatskoj imaju za svoje potrebe razvijene validacije zapisa e-HUB.

Obrazac platnog naloga e-HUB je u sve većoj primjeni u Hrvatskoj te mnoge financijske institucije i poduzeća prelaze na korištenje istog.

4.2.5 Elektronički račun (e-Račun)

Kao što je navedeno u potpoglavlju 2.6.4, e-Račun je najrašireniji oblik elektroničke isprave u elektroničkom poslovanju na svijetu koji služi kao izdavanje računa za obavljenу uslugu ili isporučenu robu. U Hrvatskoj se još 2009. godine počela razvijati hrvatska norma za e-Račun definiranjem informacijskog modela, i to od strane tadašnjeg Povjerenstva za e-Račun pri Ministarstvu gospodarstva, rada i poduzetništva, ali je to završilo donošenjem prijedloga nacрта Specifikacije elektroničkog računa u verziji 0.1²¹⁸. No, navedena specifikacija je ipak zanimljiva polazišna točka za sve buduće izvedbe e-Računa u Hrvatskoj.

²¹⁸ Specifikacija elektroničkog računa, nacrt, verzija 0.1, Povjerenstvo za e-Račun, MINGORP, 2009-12-23, <http://www.mingo.hr/userdocsimages/trgovina/Specifikacija%20e-Ra%C4%8Duna%202009-12-23.pdf>

Osnovna struktura elektroničkog računa sadržava zaglavlje i tijelo računa. **Zaglavlje** se sastoji od osnovnih elemenata, proširenja specifičnih za pojedinu industrijsku granu i proširenja definiranih bilateralnim ugovorima. **Tijelo** računa sastoji se od pojedinih stavaka računa te tako sadržava osnovne elemente, proširenja specifična za pojedinu industrijsku granu i proširenja definirana bilateralnim ugovorima.

Elementi računa definirani su na različite načine, primjerice:

- direktivom EU 2001/115/EC prema dokumentu CWA 15575, 15576 i 15577,
- Pravilnikom o PDV-u te obrascima računa R1 i R2,
- obrascem e-HUB,
- minimalnim skupom podataka koji je definirala ekspertna skupina za e-Račun EU-a.

Informacijski model e-Računa prikazan sažeto sadržava:

- normu e-Računa,
- verziju računa, poslovni proces, jedinstveni broj računa, pokazatelj preslike (kopije),
- datum izdavanja i datum na koji porez postane važeći,
- vrstu računa,
- valutu,
- broj stavaka,
- napomenu,
- podatke o periodu konsolidacije,
- reference na narudžbu, otpremnicu, primku, ugovor, cjenik, predračun i ponudu,
- podatke o dobavljaču (identifikatore, OIB, GLN, MB, šifre, naziv, adresu, mjesto izdavanja, odgovornu osobu, kontakt otpreme, kontakt računovodstva, kontakt prodaje),
- podatke o kupcu (identifikatore, OIB, GLN, MB, šifre, naziv, adresu, kontakt računovodstva, kontakt kupca, kontakt isporuke),
- podatke o poreznom posredniku (OIB, GLN, MB, šifre, naziv, adresu),
- podatke o isporuci:
 - podatke o isporučitelju (OIB, GLN, MB, šifre, naziv, adresu),
 - podatke o preuzimatelju (OIB, GLN, MB, šifre, naziv, adresu),
 - datum i mjesto isporuke,
 - datum i lokaciju otpreme,
 - uvjete isporuke,
- podatke o načinu plaćanja (šifru, datum, kanal, model, poziv na broj, račun primatelja i račun platitelja),
- podatke o uvjetima plaćanja (datum dospijeca, popust, kaznu, iznos i napomenu),
- podatke o troškovima i popustima (iznos, osnovicu, šifru i opis, postotak, izračun i porez),
- podatke o ukupnom porezu (iznos i razradu),
- ukupne iznose (neto, ukupno bez poreza, s porezom, popuste, troškove, avansni iznos, iznos za platiti),
- stavke računa.

Pojedina stavka računa sadržava:

- redni broj,
- količinu,

- ukupan iznos s popustom bez poreza,
- podatke o referenci na stavku narudžbe,
- podatke o referenci na stavku otpremnice,
- podatke o referenci na stavku primke,
- datum i mjesto isporuke,
- datum i lokaciju otpreme,
- podatke o troškovima i popustima,
- podatke o ukupnom porezu,
- podatke o artiklu (opis, naziv, broj paketa, broj stvari u paketu, šifre, klasifikacije i državu podrijetla),
- cijenu (jedinični iznos i jedinicu mjere).

Svi navedeni elementi informacijskog modela e-Računa usklađeni su (engl. *conformance*) s normom UBL 2.0 (potpoglavlje 4.2.1) tako da su izbačeni nepotrebni elementi norme UBL (kako bi se stvorio podskup norme UBL) te dodani elementi korištenjem elementa UBLExtension. Konkretna struktura e-Računa prema normi UBL dana je u dokumentu spomenutog nacrtu Specifikacije elektroničkog računa²¹⁹, a tamo je prikazana i struktura složenih tipova XML shema.

Struktura korištenih XML shema e-Računa je sljedeća:

- xsd/main/UBL-Invoice-2.0.xsd – za jednostavni račun, glavna shema,
- xsd/common/UBL-CommonAggregateComponents-2.0.xsd – za složene komponente,
- xsd/common/UBL-CommonBasicComponents-2.0.xsd – za osnovne tipove,
- xsd/common/UBL-ExtensionContentDatatype-2.0.xsd – za složenu komponentu,
- xsd/common/UBL-QualifiedDatatypes-2.0.xsd – za kvalificirane tipove podataka,
- xsd/common/ UnqualifiedDataTypeSchemaModule-2.0.xsd – za nekvalificirane tipove podataka,
- xsd/common/CodeList_CurrencyCode_ISO_7_04.xsd – za listu valuta,
- xsd/common/CodeList_LanguageCode_ISO_7_04.xsd – za listu jezika,
- xsd/common/CodeList_MIMEMediaTypeCode_IANA_7_04.xsd – za listu standardnih tipova dokumenata,
- xsd/common/CodeList_UnitCode_UNECE_7_04.xsd – za listu mjernih jedinica.

Provjera ispravnosti dokumenta e-Računa preporučuje se u dvije faze. U prvoj se fazi provjerava **struktura dokumenta** u odnosu na navedene XML sheme. Ova validacije ne može provjeriti kardinalnost²²⁰ u svim slučajevima pa se preporučuje **dodatna validacija** transformacijom XSLT-a uporabom Schematrona, koja provjerava složena pravila postojanja i vrijednosti elementa u odnosu na vrijednost nekoga drugog elementa, atributa, kodne liste i slično.

²¹⁹ Specifikacija elektroničkog računa, nacrt, verzija 0.1, Povjerenstvo za e-Račun, MINGORP, 2009-12-23, <http://www.mingo.hr/userdocsimages/trgovina/Specifikacija%20e-Ra%C4%8Duna%202009-12-23.pdf>

²²⁰ Kardinalnost označava broj pojavljivanja, osnovne postavke su nijednom, jednom ili više

Kodne liste kod e-Računa koriste se za:

- valute – prema normi ISO 4217,
- oblik adrese – prema obliku adrese UN/CEFACT 3477 Address format code,
- šifre država – prema normi ISO 3166-1 country codes,
- poštanske brojeve u Hrvatskoj – popis mjesta u RH s poštanskim brojem²²¹,
- šifre načina plaćanja – prema UN/CEFACT 4461 Payment means code,
- šifre kanala plaćanja – prema UN/CEFACT 4435 Payment channel code,
- šifre kodnih stranica – prema IANA CharacterSetCode,
- šifre tipa elektroničkog sadržaja – prema IANA MIMEMediaType,
- šifre tipova računa – bit će naknadno izrađena.

Iako je razvijen informacijski model usklađen s normom UBL 2.0, trebat će ga u budućnosti uskladiti i s normom UBL 2.1 kada bude donesena, a razmatra se i mogućnost uporabe drugih normi kao što su norma UN/CEFACT CII 2.x (engl. *Cross Industry Invoicing*) i norma ISO 20022²²². U budućnosti se, ovisno o preporukama, prvenstveno međunarodne zajednice, kao što je primjerice ova²²³, može odlučiti i o usklađivanju s tim normama, pri čemu bi se dodatno trebali preslikati elementi modela e-Računa na elemente tih normi.

²²¹ Mjesta u RH i poštanski uredi, Hrvatska pošta, <http://www.posta.hr/pomoc-i-informacije/preuzimanje-podataka-o-postanskim-uredima>

²²² ISO 20022 Financial Services - Universal financial industry message scheme, <http://www.iso20022.org/>

²²³ e-Invoicing Standardisation Overview, issues and conclusions for future actions, 2012., http://ec.europa.eu/enterprise/sectors/ict/files/invoicing/e-invoicing-standardisation-overview-issues-and-conclusions-for-future-actions_en.pdf

5. Poglavlje: Softverske arhitekture, tehnologije i protokoli

U ovom poglavlju naučit ćete:

- ✓ općenito o arhitekturama, tehnologijama, protokolima i uslugama
- ✓ arhitekture raspodijeljenih sustava
- ✓ mogućnosti interoperabilnosti kod višeslojnih arhitektura
- ✓ tehnologije izvedbe interoperabilnosti
- ✓ o međuopremi i komunikaciji porukama

5.1 Općenito o uslugama, definicije i stanja usluga

Pojam **usluga** je u širokoj primjeni u modernom svijetu. U gospodarskoj djelatnosti usluga predstavlja nematerijalni oblik nekog dobra, neopipljiva je i ne može rezultirati vlasništvom. Pojednostavljeno govoreći, usluga se izvršava kada jedna strana izvodi posao (ili neku funkciju) za drugu stranu. Slično je i u svijetu računalnih usluga. Ovo će poglavlje dati pregled najvažnijih definicija, arhitektura i mehanizama bitnih kako bismo pojasnili što podrazumijevamo pod uslugom.

Zbog činjenice da se pojam usluge vrlo često koristi, i to u različitim poljima ljudske djelatnosti, ne postoji jednostavna i jednoznačna definicija što je to usluga. Kako se ovaj priručnik bavi uslugama u svijetu računala, pokušajmo definirati na što mislimo kada govorimo o računalnoj usluzi.

5.1.1 Definicija usluga

Organizacija OASIS (poglavlje 2.4.14) definira uslugu kao „**mehanizam koji omogućava pristup do jedne ili više mogućnosti, pri čemu je pristup osiguran putem opisanog sučelja i dosljedno primijenjen s ograničenjima i politikama koje su opisane u opisu same usluge**“²²⁴. Ovakva, premda općenita, definicija vrlo dobro izražava nekoliko bitnih svojstava za svaku računalnu uslugu. Kako bi je korisnici mogli nesmetano koristiti, njihov pristup samoj usluzi mora se odvijati putem dobro definiranih sučelja. Korisniku usluge mora u svakom trenutku biti jasno koji podatak, u kojem obliku i vrsti treba predati usluzi kako bi od nje, nakon obrade, dobio zadovoljavajući rezultat. Naravno, treba poštovati i neka ograničenja koja su možda rezultat dizajna ili implementacije same usluge. Takva ograničenja u obliku politika nalaze se u opisu same usluge (o opisu usluge više ćemo govoriti u sljedećem poglavlju).

Vratimo se na trenutak u domenu gospodarstva. Uzmimo za primjer izmišljenu tvrtku Usluga d.o.o., čija je temeljna djelatnost dostava paketa od vrata do vrata. Naravno, kako bi uspješno obavila posao, tvrtka je zaposlila određen broj djelatnika, uključujući telefonskog operatera koji prima zahtjeve za uslugom, djelatnika koji dostavlja (odlazi kod pošiljatelja, preuzima paket te ga odvozi do krajnjeg odredišta), kao i računovođu koji se brine kako bi se usluga naplatila.

Svaki od djelatnika zapravo, obavljajući svoj posao, pruža uslugu tvrtki Usluga d.o.o. Tako oni čine male građevne blokove od kojih se sastoji cjelovita usluga dostave koju pruža tvrtka Usluga d.o.o. Premda svaki od zaposlenika može vidjeti samo neki ograničeni dio poslovanja tvrtke, sama tvrtka ipak se mora pobrinuti za neke zajedničke elemente poslovanja, poput dostupnosti, pouzdanosti i jasne nedvosmislene komunikacije kako bi učinkovito funkcionirala i pružala uslugu svojim korisnicima. Ti su ciljevi formulirani u obliku politike i smjernica ponašanja djelatnika tvrtke.

²²⁴ OASIS Reference Architecture Foundation for Service Oriented Architecture 1.0, http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=soa-rm

Ovakav način slaganja usluga iz malih dijelova (možemo ih zvati i komponentama) prisutan je i u računarstvu, a ta se grana programskog inženjerstva naziva **programsko inženjerstvo usmjereno komponentama** (engl. *Component-based Software Engineering*)²²⁵.

Potrebno je naglasiti razliku između komponente i usluge koja je zapravo rezultat točke gledišta. Kada govorimo o komponentama zapravo mislimo na komadić programskog koda koji će se koristiti u točno tom obliku (bez modifikacije ili nekog drugog utjecanja onoga tko je tu komponentu programski izveo). S druge strane, usluga je dostupna izvana, putem nekog od načina udaljenog sinkronog ili asinkronog pristupa²²⁶.

5.1.2 Stanja usluga

Radi lakšeg razumijevanja ostalih obilježja usluga, o čemu ćemo više raspravljati u sljedećem poglavlju, prvo ćemo definirati pojam stanja usluge.

Riječ **stanje** opisuje situaciju u kojoj se nalazi promatrani objekt. Računalni se programi najčešće nalaze u dvama stanjima, aktivnom i pasivnom. **Aktivno** je stanje ono u kojem program nešto obrađuje, pri čemu bivaju uključene njegove funkcije ili metode te se dobiva neki rezultat. U **pasivnom** se stanju program ne koristi nego je ili isključen te ne može primiti nove zadatke ili čeka nove zahtjeve te se nalazi u stanju pripravnosti. U praksi se aktivno stanje često detaljnije raščlanjuje kako bi se pobliže opisalo njegovo ponašanje u određenim slučajevima. Takav detaljniji opis moguće je formalno prikazati putem dijagrama stanja (engl. *state diagram*) u jeziku UML.

Sami programi mogu, ali i ne moraju, biti **svjesni stanja**²²⁷ u kojem se nalaze. Ako programi **nisu svjesni stanja** u kojem se nalaze, govorimo o tzv. **stateless** programima, a ako oni znaju u kojem se stanju nalaze, nazivamo ih programima **svjesnim stanja** ili tzv. **statefull** programima²²⁸.

Kada promatramo računalne programe koje zovemo usluge, tada iz razloga njihove dostupnosti i proširivosti najčešće ne želimo da se zbog mnogobrojnih provjera koje podrazumijeva očuvanje informacija o stanju neke usluge troše dodatni resursi (npr. memorija) koji bi usporili izvođenje usluge. Stoga se po definiciji **projektiranje usluga** izvodi na **načelu nesvjesnosti stanja**, odnosno *stateless*. Iznimno, usluge mogu biti svjesne informacije o svome stanju, što se može iskoristiti ako usluga treba čekati na rezultat neke druge usluge kako bi mogla nastaviti posao.

Razmotrimo primjer složene usluge kupnje proizvoda na Webu koja se sastoji od (pojednostavljeno gledano) triju koraka. Prvi korak izvodi proces autentikacije kupca, u drugom se koraku odabire proizvod te se u trećem koraku plaća. Kada bi ovakva usluga bila izvedena tako da prati stanje u kojem se proces nalazi u svakoj pojedinoj fazi, tada bi se ta informacija

²²⁵ Crnković, M. Larsson, Building Reliable Component-Based Software Systems, Artech House, 2002.

²²⁶ Fowler, M., Inversion of Control Containers and the Dependency Injection pattern, <http://martinfowler.com/articles/injection.html>

²²⁷ Brown, S., Vranesic, Z., Fundamentals of Digital Logic with Verilog Design, McGraw-Hill, 2nd ed., 2007.

²²⁸ Prasad, K., Statefull vs. Stateless, in A Practical Introduction to Enterprise Java Beans, A Component Technology, <http://www.comptechdoc.org/docs/kanti/ejb/index.html>

zapisivala u memoriju, kako bi bila spremna za nastavak procesa. U takvom slučaju korisnik bi morao za svaki korak u procesu ponovno koristiti točno istu uslugu koja je obavila prijašnji korak. U slučaju velikog broja korisnika takve usluge te njene nedostupnosti ili pojave kvara, može doći do kašnjenja u pružanju usluge ili potpune nemogućnosti korištenja usluge.

S druge strane, ako je usluga realizirana tako da bude nesvjesna stanja, odnosno *stateless*, tada će nakon izvršavanja svakog pojedinog koraka korisnik dobiti (uz rezultat) dovoljnu informaciju o stanju kako bi proces mogao nastaviti putem bilo koje usluge koja može izvesti sljedeći korak. Na taj je način omogućeno korištenje mehanizma raspodijeljenosti u ovom procesu, čime je povećana raspoloživost i smanjena osjetljivost na pogreške.

Kako se u vrlo malom broju aplikacija može govoriti o tome da stanje uopće nije potrebno pratiti, kada govorimo o pohrani informacije o stanju neke usluge zapravo nam je bitno gdje će ta informacija biti pohranjena. U opisanom će primjeru većina informacija biti zapisana u bazi podataka koja će čuvati stanja, dok će se među korisnikom i uslugom prenositi primjerice samo identifikator kupca.

5.2 Arhitekture raspodijeljenih sustava

Prije nastavka razmatranja usluga i njihovog prikaza u svijetu raspodijeljenih računalnih arhitektura, podsjetimo se nekih tehnologija na kojima se zasniva dostupnost takvih sustava.

5.2.1 Protokol HTTP

Protokol HTTP ili **HyperText Transfer Protocol** je vjerojatno je najpoznatiji i najčešće korišten protokol aplikacijskog sloja namijenjen za prijenos hipertekstualnih datoteka između raspodijeljenih kolaborativnih informacijskih sustava. Davne 1991. godine izdana je prva verzija protokola HTTP 0.9 koja više nije u upotrebi, a 1996. godine izdana je verzija **HTTP/1.0** opisana u RFC 1945, verzija **HTTP/1.1** 1997. godine u dokumentu RFC 2616, a 2015. godine verzija **HTTP/2** koja je opisana u RFC 7540.

Verzija HTTP/2 unapređuje verziju HTTP/1.1 na način da reducira latenciju, uz efikasno komprimiranje optimizira količinu podataka u zaglavlju zahtjeva, uvodi podršku za određivanje prioriteta zahtjeva i tehniku *server push*. Iako uvodi mnoge promjene kako bi se promet podataka optimizirao po pitanju *streaminga* i prijenosa video formata, promjenu formata podataka iz tekstualnog u binarni, HTTP/2 ne uvodi semantičke promjene u protokol HTTP. Svi ključni koncepti, kao što su HTTP metode, statusni kodovi, URI-jevi i polja u zaglavlju, ostaju gdje su bili i prije. HTTP/2 uvodi promjene u način formatiranja podataka i prijenosa od klijenta do poslužitelja, ali ujedno i sakriva kompleksnost uvođenjem novog sloja okvira (engl. *framing layer*). Radi toga nije potrebno mijenjati i ažurirati aplikacije, jer je zadržana kompatibilnost.²²⁹

²²⁹ Uvod u HTTP/2 protokol - <https://developers.google.com/web/fundamentals/performance/http2>

5.2.1.1 HTTP/1.1

Protokol HTTP spada u skupinu protokola **zahtjev / odgovor** (engl. *request / response*). Podrazumijeva postojanje klijenta i poslužitelja s priključcima (engl. *ports*), definira da poslužitelj stalno osluškuje zahtjeve klijenata na nekom određenom priključku (najčešće je riječ o standardnim priključcima 80 ili 8080), a za prijenos podataka zahtijeva pouzdanu vezu korištenje protokola prijenosnog sloja (engl. *transport layer protocol*), najčešće uspostavom veze protokolom TCP između klijenta i poslužitelja.

Klijent šalje poslužitelju svoj **zahtjev** koji sadržava pristupnu metodu, identifikator traženog objekta, verziju protokola koju koristi, a može dodati i eventualno potrebne dodatne informacije. Kada poslužitelj obradi zahtjev, vraća klijentu **odgovor** u kojem je sadržana informacija o rezultatu (u obliku poznatog koda), metapodaci o odgovoru te sama stranica koja se željela dohvatiti.

Sve tekstualne poruke protokola HTTP/1.1 sadrže dva dijela:

- **upravljački dio poruke** – sadrži početni redak i zaglavlje, načelno čitljivo ljudima,
- **tijelo poruke** – sam sadržaj poruke, može, ali i ne mora biti čitljiv ljudima

Struktura poruke protokola HTTP sadrži:

- **početni redak** (engl. *start-line*) te:
 - kod zahtjeva sadrži pristupnu metodu, URI i verziju protokola HTTP, oblika: metoda URI verzija-HTTP CRLF
 - kod odgovora sadrži verziju protokola HTTP, statusni kod i objašnjenje, oblika: verzija-HTTP statusni-kod razlog CRLF
- **zaglavlja** (engl. *message-headers*) – mogu biti različita, ovise o poruci
 - oblik: ime:vrijednost
- **prazni redak** – odjeljuje zaglavlje od tijela poruke
- **tijelo poruke** (engl. *message-body*) – opcionalno, sadrži sadržaj koji može biti kodiran

Tijelo poruke sadrži jedan ili više resursa, pa sam sadržaj može biti cjelokupni resurs, odsječak tzv. *chunk* ili više resursa tzv. *multipart*. Vrsta tijela označava se MIME tipom (poglavlje 5.2.3), a veličina tijela izražava se u zaglavlju poruke brojem okteta (engl. *octet counting*), odnosno bajtova.

U smislu najkraće **konverzacije** podrazumijeva se jedan zahtjev i jedan odgovor. Pri tome poruka zahtjeva treba sadržavati sve informacije potrebne za ispunjenje tog zahtjeva, a konverzacija završava odgovorom u smislu ispunjena zahtjeva. HTTP je protokol bez očuvanja stanja (engl. *stateless*), pa poslužitelj ne čuva stanje između dvije konverzacija, osim ako to nije na neki drugi način izvedeno.

Protokol HTTP definira i **metode** (katkad ih zovu i „glagolima“) koji definiraju određenu akciju koja se želi izvesti nad imenovanim resursom. Specifikacija protokola HTTP/1.0 definira GET, POST i HEAD metode, dok se u specifikacija HTTP/1.1 definira dodatnih pet metoda, to CONNECT, DELETE, OPTIONS, PUT i TRACE. Prema potrebi mogu se definirati i dodatne metode bez narušavanja postojeće infrastrukture, pa tako primjerice WebDAV definira još 7 metoda, a RFC5789 definira metodu PATCH.

Ukratko ćemo objasniti smisao najvažnijih metoda protokola HTTP:

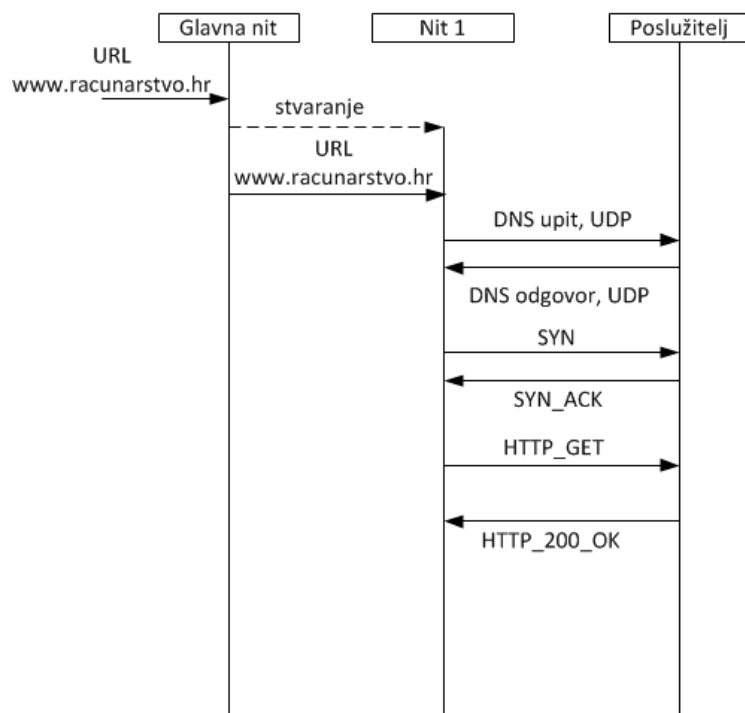
- metoda **GET** zahtijeva dohvat određenog resursa, odnosno njegovog sadržaja te se koristi isključivo za dohvaćanje određenog sadržaja bez mogućnosti promjene sadržaja na poslužitelju ili bilo kakvih drugih akcija,
- metoda **POST** zahtijeva da poslužitelj prihvati entitet pridružen zahtjevu kako novi sadržaj ili nadopunu resursa identificiranog putem URI. Metoda POST zapravo šalje neke dodatne podatke na poslužitelj, bilo da je riječi o objavi neke poruka ili o nekom bloku ili skupu podataka, primjerice sadržaja obrasca na internetskim stranicama ili sadržaja objekta koji se želi pohraniti u bazu podataka,
- metoda **HEAD** zahtijeva dohvat podataka o određenom resursu, slično metodi GET, ali samo podatke iz zaglavlja bez tijela odgovora. Ovo je korisno kod potreba za dohvaćanjem meta podataka sadržanih u zaglavlju odgovora, bez dohvaćanja cjelokupnog resursa u odgovoru,
- metoda **PUT** zahtijeva pohranu sadržanog entiteta pod zahtijevanim URI, uz posljedicu stvaranja novog resursa ako takav ne postoji.
- metoda **DELETE** zahtijeva brisanje entiteta s poslužitelja.

Neke metode su definirane kao **sigurne** (engl. *safe methods*), odnosno kao metode koje ne mijenjaju stanje poslužitelja već služe isključivo za dohvat podataka. To se prvenstveno odnosi na metode GET, HEAD, OPTIONS i TRACE, a te metode nemaju neželjenih posljedica u obliku izvođenja akcija na poslužitelju, osim relativno bezopasnih poput logiranja, predmemoriranja i sličnih. S druge strane postoje **nesigurne** metode (engl. *unsafe methods*), kao što su POST, PUT i DELETE, koje su namijenjene za izvođenje određenih akcija na poslužitelju u smislu promjene stanja poslužitelja, najčešće promjenom podataka u nekom obliku transakcije. Iako se metoda GET smatra sigurnom u praksi je moguće da se dohvat pojedinog resursa pokreću i neke akcije na poslužitelju, što se pokušava umanjiti propisivanjem dobrih praksi izvedbe odgovora na zahtjev koje ne uključuju takve akcije. Ovo je dodatno bitno zato što bi automatski agenti koji provode predmemoriranje ili pretraživanje stranica mogli stalno mijenjati stanje na poslužitelju, što je loša praksa.

Osnovna konverzacija protokolom HTTP uobičajeno sadrži sljedeće korake:

13. korisnik (ili agent) unosi URL
14. stvara se nit za obradu
15. zahtijeva se dohvat određenog resursa (npr. otvaranje internetske stranice)
16. traži se IP adresa poslužitelja od poslužitelja DNS
17. otvara se TCP konekcija prema poslužitelju
18. nakon trostrukog rukovanja (engl. *handshake*) uspostavlja se veza
19. šalje se zahtjev metodom GET protokola HTTP
20. dobiva se odgovor od poslužitelja, u slučaju uspjeha kod 200 i sadržaj tražene stranice

Ovi osnovni koraci jednostavne konverzacije protokolom HTTP mogu se i prikazati (Slika 5.1).



Slika 5.1: Osnovni koraci jednostavne konverzacija protokolom HTTP

U nastavku možemo vidjeti primjer zahtjeva i odgovora korištenjem protokola HTTP. Tablica 5.1 prikazuje nepotpun zahtjev metodom GET protokola HTTP/1.1 za dohvaćanjem stranice „index.html“ u korijenskom direktoriju poslužitelja imena „www.usluga.org“ od strane Mozilla/4.0 kompatibilnog agenta.

HTTP zahtjev	Strukturni dio	Objašnjenje
GET /index.html HTTP/1.1	Početni redak	Pristupna metoda, put i verzija protokola kojim se želi pristupiti
User Agent: Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.0; WOW64; Trident/4.0; SLCC1; .NET CLR 2.0.50727; Media Center PC 5.0; InfoPath.2; .NET CLR 3.5.30729; .NET CLR 1.1.4322; .NET CLR 3.0.30729;	Zaglavlje	Identifikator pretraživača koji se koristi
Host: www.usluga.org		Adresa poslužitelja kojem je zahtjev namijenjen
Accept: image/gif, image/x-bitmap, image/jpeg, image/pjpeg, image/png, */* Accept-Encoding: gzip Accept-Language: en Accept-Charset: iso-8859-1, *, utf-8		Informacija o tipu podataka koje klijent može primiti
	Prazni redak	

Tablica 5.1: Zahtjev protokolom HTTP

Tablica 5.2 prikazuje primjer nepotpunog sadržaja odgovora koji vraća internetsku stranicu.

HTTP zahtjev	Strukturni dio	Objašnjenje
HTTP/1.1 200 OK	Početni redak	Verzija protokola, statusni kod i objašnjenje
Date: Thu, 09 Aug 2013 12:23:29 GMT Server: Apache/2.2	Zaglavlje	Datum obrade zahtjeva i verzija poslužiteljskog softvera
Last-Modified: Mon, 04 Aug 2013 09:33:15 GMT		Datum zadnje promjene traženog dokumenta
Accept-Ranges: bytes Content-Length: 884 Content-Type: text/html		Metapodaci o samom odgovoru
	Prazni redak	
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN"> [...]	Tijelo poruke	Sadržaj odgovora je primjerice HTML dokument

Tablica 5.2: Odgovor protokolom HTTP

Zaglavlje poruke protokola HTTP najčešće sadrži sljedeće dijelove:

- **Connection** – služi za kontrolu veze,
- **Host** – obavezno ime poslužitelja, podrška za *proxy* i virtualne poslužitelje,
- **Content-Type** – oznaka MIME tipa poruke,
- **Content-Length** – oznaka duljine sadržaja poruke u oktetima,
- **Content-Range** – oznaka dijela resursa,
- **Accept-Language** – popis jezika koje klijent prihvaća,
- **User-Agent** – vrsta klijenta,
- **Date** – datum.

Vrijednosti s prefiksom „X-„ su nestandardna zaglavlja vezana uz određene tehnologije, klijente i poslužitelje, a nepoznata zaglavlja se zanemaruju.

Jedan od zanimljivijih dijelova HTTP odgovora je rezultat obrade zahtjeva. Specifikacija protokola definira brojčane vrijednosti, tzv. statusne **kodove odgovora protokolom HTTP** (engl. *HTTP response status codes*), u rasponu **od 100 do 599**, koji kodiraju razne moguće odgovore poslužitelja na zahtjev za prikazom neke stranice.

Statusne kodove odgovora protokolom HTTP možemo prema prvoj znamenici podijeliti u pet grupa:

- **1xx: informativni** (engl. *Informational*) – zahtjev primljen, nastavak rada,
- **2xx: uspjeh** (engl. *Success*) – zahtjev je primljen, protumačen i prihvaćen,

- **3xx: preusmjerenje** (engl. *Redirect*) – akcija preusmjerenja,
- **4xx: greška klijenta** (engl. *Client Error*) – zahtjev nije ispravan ili neispunljiv zbog klijenta,
- **5xx: greška poslužitelja** (engl. *Server Error*) – poslužitelj nije uspio ispuniti zahtjev.

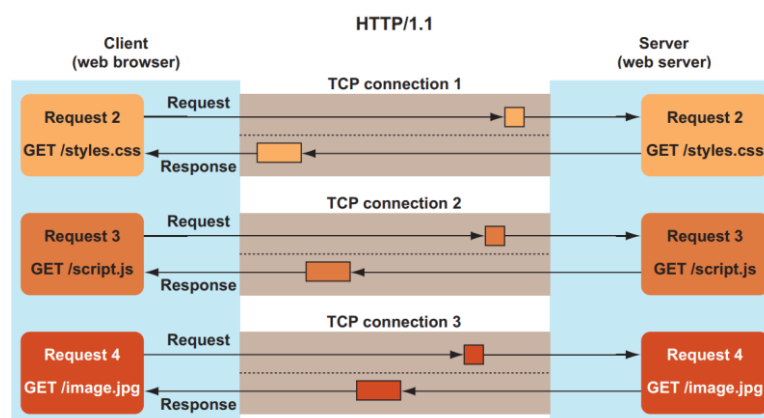
Među najpoznatije među njima pripadaju:

- **200 OK** – uobičajeni kôd uspjeha, označava da je zahtjev uspio i preglednik će dobiti traženu stranicu u nastavku odgovora, stvarni odgovor ovisi o korištenoj metodi,
- **400 Bad Request** – zahtjev ne može biti ispunjen zbog pogrešne sintakse,
- **401 Unauthorized** – neautorizirani pristup, odgovor mora uključiti zaglavlje autentikacije,
- **403 Forbidden** – iako je zahtjev valjan, poslužitelj odbija odgovoriti, problem autorizacije,
- **404 Not Found** – traženi resurs (npr. stranicu) poslužitelj nije pronašao,
- **500 Internal Server Error** – generička poruka greške na poslužitelju, neočekivano stanje poslužitelja, često kod dinamički stranica, odnosno aplikacija Web.

5.2.1.2 Usporedba protokola HTTP/1.1 i HTTP/2

HTTP/2 je u potpunosti binarni protokol, za razliku od HTTP/1.1 koji je tekstualni protokol kako bi bio što prilagođeniji za *streaming* i prenošenje video podataka. Osim toga, HTTP/1.1 je sinkroni protokol kod kojeg je u slučaju slanja velike količine podataka potrebno otvoriti više konekcija kako bi bilo moguće istovremeno slati nekoliko većih zahtjeva umjesto puno malih, kako to izvršava protokol HTTP/2 korištenjem jedne konekcije i više *streamova* za svaki zahtjev i odgovor.

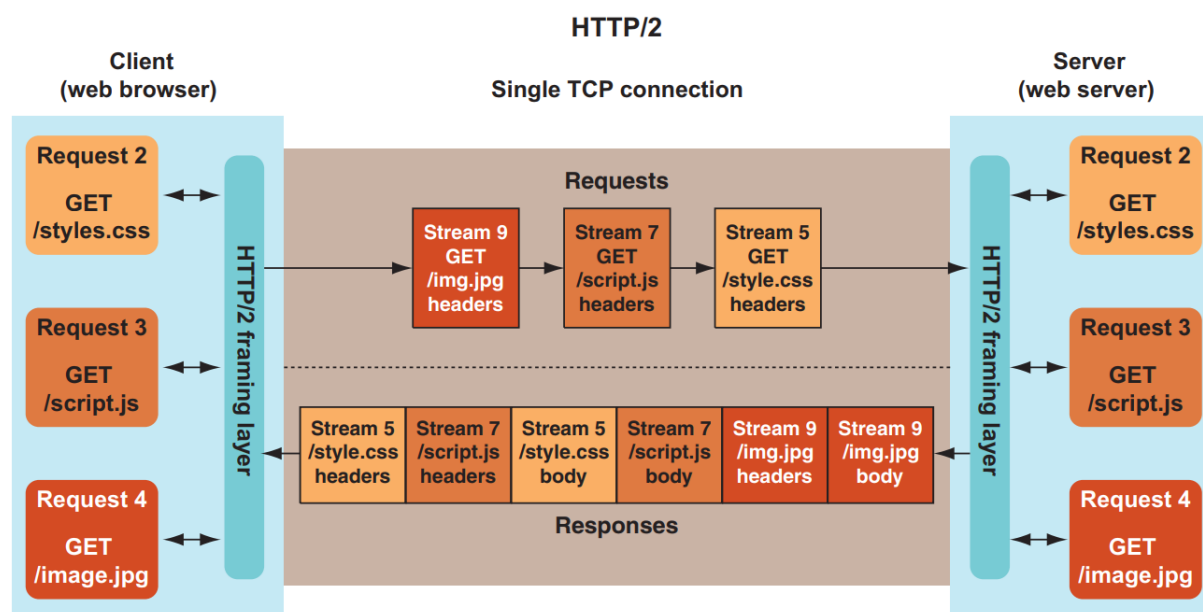
Okviri (engl. *frames*) su temelj za implementaciju mogućnosti slanja više poruka u isto vrijeme. Svaki okvir je označen, kako da bi mogao biti povezan s tokom (engl. *stream*) koji omogućava više poruka koje se mogu poslati u potpunosti paralelno. Na sljedećoj slici je prikazan način na koji se prenose tri datoteke („styles.css“, „script.js“ i „image.jpg“) u slučaju protokola HTTP/1.1:



Slika 7. Slanje više istovremenih zahtjeva kod HTTP/1.1 protokola²³⁰

²³⁰ Barry Pollard: HTTP/2 In Action, Manning, 97. stranica

Za svaku od datoteka je potrebno kreirati novu konekciju, ako se radi o protokolu HTTP/1.1. U slučaju HTTP/2 verzije protokola omogućeno je slanje više zahtjeva istovremeno korištenjem samo jedne TCP konekcije (engl. *connection*), ali različitih tokova. To se omogućava korištenjem binarnih tipova podataka organiziranih u okvire, pri čemu svaki okvir ima identifikator toka. Primatelj u tom slučaju može rekonstruirati cijelu poruku kad budu primljeni svi okviri toka.



Slika 8. Zahtijevanje tri resursa korištenjem multipleksirane HTTP/2 konekcije²³¹

U slučaju protokola HTTP/2 se koristi multipleksirana veza koja ne blokira komunikaciju nakon slanja zahtjeva kao u slučaju protokola HTTP/1.1.

HTTP/1.1 je protokol koji se temelji na jednostrukom slanju zahtjeva i primanju odgovora, pa nema ni potrebe za određivanjem prioriteta. Internetski preglednik odlučuje o prioritetu izvan HTTP protokola na način da se najprije zahtijevaju resursi koji blokiraju konekciju (HTML, CSS i JavaScript datoteke), a kasnije ostali resursi. Resursi se dohvaćaju jedan po jedan, čekajući na oslobađanje konekcije koja se nakon toga koristi.

Protokol HTTP/2 ima puno veći limit na broj zahtjeva koje može izvršavati (obično stotinjak, što je podrazumijevana količina) pa većina datoteka može odmah biti poslana, bez potrebe za čekanjem (kao što je to slučaj u protokolu HTTP/1.1). Kako bi se brže dohvatili resursi od ključne važnosti, moraju se **odrediti pripriteti** kako bi se optimizirao prikaz sadržaja u internetskom pregledniku, što se implementira na poslužitelju i omogućava bolje performanse.

Još jedna prednost protokola HTTP/2 je upravljanje tokom podataka (engl. *flow control*) što se koristi u slučaju kad primatelj ne može obrađivati sve podatke koje mu poslužitelj šalje. U tom slučaju se zaustavlja slanje podataka sve do trenutka dok primatelj opet može primiti podatke.

²³¹ Barry Pollard: HTTP/2 In Action, Manning, 97. stranica

HTTP zaglavlja prilikom slanja zahtjeva i primanja odgovora sadrže važne podatke kao što su kolačići (engl. *cookies*), informacije o računalu i internetskom pregledniku te formate podataka koji se izmjenjuju. Ti podaci se vrlo često ponavljaju, iako se nisu promijenili. Iako protokol HTTP/1.1 omogućava komprimiranje tijela poruke, ne obavlja komprimiranje zaglavlja poruke, što je implementirano u protokolu HTTP/2, čime se dodatno optimizira količina poslanih i primljenih podataka.

Još jedna bitna razlika između protokola HTTP/1.1 i HTTP/2 je korištenje koncepta *server push*. Time je omogućeno da na jedan zahtjev klijenta, poslužitelj odgovara s više odgovora. Na primjer, kod dohvaćanja datoteka potrebnih za prikazivanje stranica u internetskom pregledniku je potrebno poslati samo jedan zahtjev da poslužitelj vrati sve resurse koji su potrebni za prikazivanje sadržaja, a ne da za svaki od njih mora poslati zaseban zahtjev (kao što je to slučaj s protokolom HTTP/1.1).

5.2.2 Usklađeni identifikator resursa URI

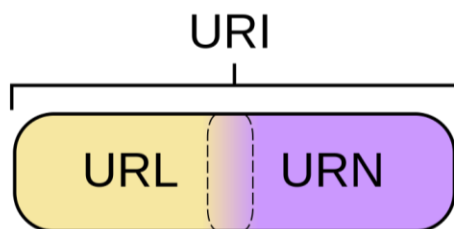
Kako bismo pronašli neki resurs, moramo točno znati gdje se nalazi. Na lokalnoj razini našeg računala najčešće se radi o lokacijskom sustavu pohrane resursa, poput datotečnog sustava na tvrdom disku. Na globalnoj razini koju predstavlja Internet, imamo, iako konceptualno sličan, zapravo znatno složeniji problem **jedinstvene identifikacije resursa**. Nad različitim vrstama resursa se mogu izvoditi različite akcije, a ovisno o njihovim lokacijama mogu se koristiti i različiti načini pristupa. Preduvjet globalnog pronalaska resursa, primjerice dokumenta, je prikladna identifikacija resursa zasnovana na njegovom nazivanju (imenovanju) i poznavanju točne lokacije.

Radi ujednačavanja tada već pomalo različitih pogleda na resurse na Internetu, Tim Berners-Lee je 1994. godine objavio RFC 1630²³², u kojem je, uz detaljniji opis **usklađenog naziva resursa URN** (engl. *Uniform Resource Name*) i **usklađenog lokatora resursa URL** (engl. *Uniform Resource Locator*), definiran i **usklađeni identifikator resursa URI** (engl. *Uniform Resource Identifier*) i formalizirana njegova sintaksa.

URI je definiran kao „*kompaktan niz znakova koji identificiraju apstraktni ili fizički resurs*“, a sadrži niz znakova iz ograničenog skupa znakova i može se pojaviti u različitim oblicima. Naglasak je na jedinstvenom razlikovanju resursa te je važna identifikacija i lokacija, a ne dostupnost (mogućnost dohvata).

URI pretpostavlja da svaki resurs može biti identificiran putem **lokacije** koristeći **URL**, putem **naziva** koristeći **URN**, ili koristeći **oba** kriterija. Slika 5.2 prikazuje ovu problematiku:

²³² Berners-Lee, T., Universal Resource Identifiers in WWW, IETF; 1994, <http://tools.ietf.org/html/rfc1630>



Slika 5.2: Odnos URI, URL i URN

URI je definiran kao proširenje URL-a, a njegova sintaksa nalikuje onoj URL-a, s time da se točan opis definira korištenjem pojedine sheme. Upravo je ovako široka mogućnost identifikacije putem lokacije nekog resursa na Internetu razlog zašto se često greškom govori o URL-u iako se zapravo koristi URI. Prikazat ćemo nekoliko primjera URI-ja:

```
file://D:/knjiga/racunarstvo/IIS.txt
ftp://ftp.racunarstvo.hr/iis/IIS.txt
http://www.racunarstvo.hr/iis/iis.html
mailto:branko.mihaljevic@racunarstvo.hr
news:hr.racunarstvo.iis
tel:+385-1-555-5555
telnet://192.168.0.1:80/
```

Kao što se može vidjeti u primjerima URI se koristi kod različitih vrsta identifikatora resursa, a sam URI definira okvir oblikovanja i korištenja u obliku sintaksnih pravila, strukture i mehanizma obrade, dok je način ostvarenja identifikacije prepušten određenoj shemi (npr. http, ftp, ssh, svn, mailto i sl.). Za ostvarenje identifikacije potrebno je prvo obraditi sadržaj URI-ja u skladu sa shemom. Za rastavljanje URI-ja na osnovne komponente zadužen je tzv. **URI parser**.

Na primjeru URI-ja <http://www.racunarstvo.hr:80/iis/nekiput?resurs=iis#url> možemo proučiti pet **osnovnih komponenta** URI-ja:

- **shema** (engl. *schema*) – npr. http,
- **autoritet** (engl. *authority*) – npr. www.racunarstvo.hr:80,
- **put** (engl. *path*) – npr. iis/nekiput,
- **upit** (engl. *query*) – npr. resurs=iis,
- **fragment** (engl. *fragment*) – npr. url.

Svaki URI se mora sastojati barem od sheme i (potencijalno praznog) puta.

URN, kao podskup URI, identificira neki resurs preko naziva (imena) u nekoj **domeni**, odnosno njegovom **prostoru naziva** (engl. *namespace*), ali ne definira kako se taj resurs može dohvatiti. URN garantira jedinstvenost i trajnost identifikacije, a oblik zapisa URN-a je:

```
<URN> ::= "urn:" <NID> ":" <NSS>
```

pri čemu NID označava prostor imena (engl. *namespace identifier*), a NSS označava specifičan znakovni niz (engl. *namespace specific string*). Prikazat ćemo nekoliko primjera URN-a:

```
urn:isbn:0-395-36341-1
urn:ietf:rfc:3187
urn:isan:0000-0000-9E59-0000-0-0000-0000-2
```

URN najčešće služi samo kao pokazivač gdje se nalaze definicije ili sheme koje se koriste unutar nekog dokumenta koji se želi dohvatiti. Tipičan primjer je definicija XSD-a ili XML sheme (poglavlje 3.4.1), a ovako izgleda primjer URN-a:

```
<xsd:schemamlns="http://www.w3.org/2001/XMLSchema"
  xmlns:xsd=http://www.w3.org/2001/XMLSchema
  targetNamespace="urn:example">
```

URL je najčešće korišteni način pronalaženja nekog resursa. On definira točnu **lokaciju** resursa, ali definira i **način pristupa** resursu, koji je opisan shemom, odnosno pristupnim protokolom zapisanim u prefiksu URL-a. Stoga URL podržava različite pristupne protokole, iako je u praksi najčešće korišten protokol HTTP. No, URL ne garantira jedinstvenost ni trajnost resursa, pa resurs može biti prisutan na više različitih lokacija ili premješten. Prikazat ćemo nekoliko primjera URL-a:

```
http://www.racunarstvo.hr
ftp://ftp.racunarstvo.hr
```

Tijekom vremena razvio se još jedan oblik nazvan **URI Reference**. On može sadržavati cijeli URI ili samo neki, shemom definirani dio, a moguće je čak koristiti i prazan znakovni niz. Koristi se poseban identifikator (znak #), koji se nalazi na kraju URI Reference. Ovaj se oblik često koristi kod opisnih jezika kako bi se pokazalo na druge resurse, primjerice kod korištenja sidra (engl. *anchor*) u jeziku HTML koji pokazuje na dio dokumenta označen URI Referencom, kao što je:

```
<a href="#URI">
```

Na kraju napomenimo da URI, URL i URN mogu koristiti ograničeni skup znakova koji uključuje mala i velika slova, brojeke i neke posebne znakove, dok je upotreba određenih rezerviranih znakova zabranjena (npr. : / ? # [] @ ! \$ & ' () * + , ; =). Ipak, zabranjene znakove je moguće je koristiti u obliku zapisa heksadekadskim znamenkama (%HH), pa se tako primjerice znak praznine (engl. *space*) može upisati %20 (ASCII kod 0x20), kao u primjeru <http://www.racunarstvo.hr/interoperabilnost%20informacijskih%20sustava.html>.

5.2.3 Višenamjensko proširenje elektroničkih poruka na Internetu MIME

Protokol **Multipurpose Internet Mail Extension (MIME)** inicijalno je zamišljen kao norma za proširenje elektroničke pošte, pa se u kontekstu elektroničke pošte koristi za:

- **proširenje poruka** elektroničke pošte drugim kodiranjima teksta (kodnim stranicama) osim 7-bitnog ASCII koda,
- **privitke poruka** elektroničke pošte u drugim oblicima osim teksta, što se može koristiti za slike, različite dokumente i slično,
- **višedijelne** (engl. *multi-part*) **poruke** elektroničke pošte,
- **podatke u zaglavlju** elektroničke pošte (npr. polje naslova i sl.) **u drugim kodiranjima teksta** (kodnim stranicama).

No, protokol MIME je evoluirao izvan tih okvira upotrebe za potrebe elektroničke pošte te je prihvaćen iza potrebe **općenitog opisivanja tipa sadržaja** (engl. *content type*) na Internetu (i Webu) korištenjem u drugim protokolima, poput protokola HTTP, te kao **pohrana za različite tipove sadržaja** u obliku **MIME objekata**, označavajući resurse dodanim oznakama tipa i načina kodiranja podataka.

Stoga se danas **norma MIME**, definirana u šest povezanih dokumenata RFC (RFC 2045, RFC 2046, RFC 2047, RFC 4288, RFC 4289 i RFC 2049), koristi u tri svrhe:

- **za označavanje tipa podataka,**
- **za označavanje načina kodiranja podataka,**
- **za strukturiranje poruka.**

Normirane strukture **tipova podataka**, tzv. **MIME tipovi** (engl. *MIME types*), označavaju se oznakom u obliku **tip/podtip** (engl. *type/subtype*), a postoje četiri glavna tipa:

- **application** – označava razne aplikacijske podatke, kao što su /javascript, /octet-stream, /xhtml+xml, /zip itd.,
- **audio** – označava zvučne datoteke, kao što je npr. /mpeg,
- **image** – označava slikovne datoteke, kao što su /gif, /jpeg, /png i slični,
- **text** – označava tekstualne podatke, kao što su /html i /plain.

Dodatno, postoji podrška i nenormiranim tipovima podataka koja počinje s prefiksom „X-“ te tipovima podataka pod kontrolom različitih proizvođača koja započinje prefiksom „vnd“.

Kako bi naznačili da je poruka oblikovana korištenjem MIME, postavlja se sljedeće zaglavlje:

```
MIME-Version: 1.0
```

Oznaka tipa sadržaja u obliku tip/podtip također se definira u zaglavlju poruke, npr. kao:

```
Content-Type: text/plain
```

Višedijelne poruke (engl. *multipart*) izvode se u stablastoj strukturi dijelova te se cijela poruka označava tipom multipart/mixed, dok se tip svakog dijela poruke dodatno označava u zaglavlju tog dijela.

Tekst zaglavlja poruke u kodiranjima različitim od ASCII zapisuje se u skladu sa **sintaksom MIME kodiranih riječi** (engl. *MIME encoded-word syntax*) definiranom u RFC 2047, u obliku `=?kodna_stranica?kodiranje?kodirani_tekst?=`, pa se tako primjerice riječ „Računarstvo“ u kodiranju UTF-8 zapisuje kao:

```
=?utf-8?Q?Ra=C4=8Dunarstvo?=
```

Kodiranje teksta poruke može se izvesti na niz načina, a najčešći su:

- **quoted printable** – svaki znak koji nije ispisivi ASCII znak kodira se s =HH
- **base64** – znakovi u grupama s 6 binarnih znamenaka, svaka grupa zamjenjuje se znakom A do Z, a do z, 0 do 9, + i /, a dulje linije (76 znakova) prekinute su znakom „=“

5.3 Arhitekture raspodijeljenih sustava

Raspodijeljeni sustav^{233, 234, 235} (engl. *distributed system*) možemo definirati kao sustav čiji se hardverski ili softverski dijelovi nalaze na različitim čvorovima (pod čvorom najčešće podrazumijevamo računalo – poslužitelj) u mreži, a međusobno razmjenjuju poruke kako bi obavili povjereni im posao.

Iz ovakve bi se definicije moglo iščitati da je za postojanje raspodijeljenog sustava dovoljno da je računalo spojeno na mrežu, no osim mreže potrebne su još neka obilježja, poput koordinacije vremena, smanjivanja osjetljivosti na pogreške i sigurnosti.

Možemo reći da vrijedi tvrdnja da bismo neki sustav zvali raspodijeljenim, on mora svojim korisnicima izgledati jedinstven. Drugim riječima, raspodijeljenost je obilježje samo na razini sustava, dok se ona mora sakriti na prezentacijskoj razini.

Ovo je najlakše objasniti kroz raspodijeljeni sustav Weba, gdje velike kompanije, primjerice Google, posjeduju velik broj poslužitelja zemljopisno raspodijeljenih po čitavom svijetu, dok korisnik putem jedinstvene adrese sjedišta Weba pristupa zapravo onom najbližem.

Kako se raspodijeljenost koristi dulje vremensko razdoblje, razvilo se nekoliko arhitektura od kojih je svaka pogodna za neku specifičnu namjenu.

5.3.1 Arhitektura klijent – poslužitelj

Arhitektura sustava u kojem je jedan od sudionika poslužitelj dok drugi predstavlja njegovog klijenta (arhitektura klijent – poslužitelj)²³⁶ jedan je od prvih načina korištenih u svijetu raspodijeljenih sustava. Prvotno je zamišljen kao udaljeni pristup, koji na primitivan način omogućava korisnicima da se preko terminala putem telefonske linije (kasnije i putem lokalne mreže) spoje na neko veliko računalo (engl. *mainframe*) te na njemu obave posao. S razvojem sustava i povećanjem broja korisnika, razvili su se sustavi koji su imali raspodijeljene podatke i memoriju.

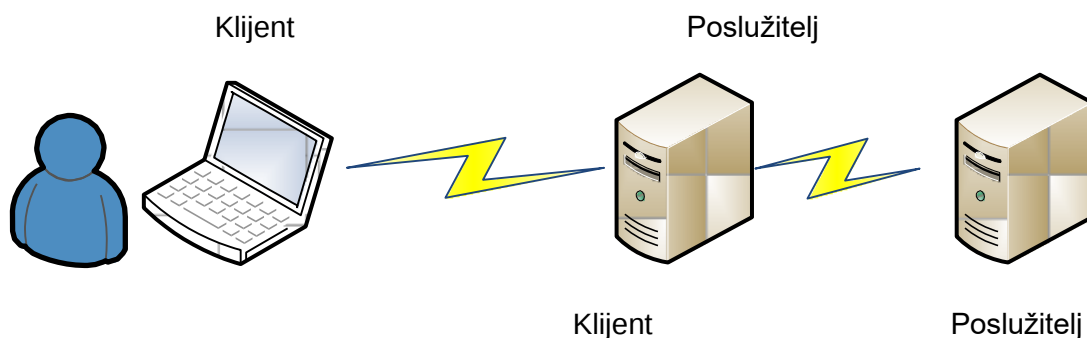
Klijenti u ovakvom sustavu raspolažu aplikacijom kojom se spajaju na poslužitelja i koja je u mogućnosti interpretirati podatke koje dobiva od poslužitelja. S druge strane, na poslužitelju se odvija proces koji stalno osluškuje i prihvaća zahtjeve klijenata, obrađuje ih i vraća im rezultate. Jednako tako, pojedini programi na poslužiteljima mogu biti podešeni tako da dohvaćaju neke podatke s drugih poslužitelja, postajući tako njihovi klijenti. Sljedeća slika pokazuje osnovnu klijent – poslužitelj arhitekturu.

²³³ Coulouris, G. F., Dollimore, J., Kindberg, T., Distributed systems: concepts and design, Pearson Education, 2005.

²³⁴ Veríssimo, P., Rodrigues, L., Distributed systems for system architects, Springer, 2001

²³⁵ Tanenbaum, Andrew S., Van Steen, M., Distributed systems: principles and paradigms, Second Edition, Pearson Prentice Hall, 2007

²³⁶ Tanenbaum, A. S., Modern Operating Systems, 3rd edition, 2007.



Slika 5.3: Arhitektura klijent - poslužitelj

Većina današnjih poslužitelja Weba koristi upravo arhitekturu koja je prikazana na slici 1. Najpoznatije besplatno rješenje za poslužitelj Weba zasnovano je na tzv. LAMP platformi^{237 238} (Linux²³⁹, Apache²⁴⁰, MySQL²⁴¹, Perl²⁴²/PHP²⁴³/Python²⁴⁴). Na operacijskom sustav Linux instaliran je poslužitelj Weba (Apache), kojem korisnik takvog sjedišta Weba (koje je u cijelosti ili djelomično napisano u jednom od navedenih programskih jezika) pristupa na uobičajen način putem Web preglednika. Većina modernih sjedišta Weba (danas je uobičajeno raditi s nekom inačicom sustava za upravljanje sadržajem CMD (engl. *Content Management System*)) za svoj rad treba bazu podataka. Ako je riječ o većem sjedištu Weba, tada će, zbog optimalnog rada sustava, baza podataka biti instalirana na zasebnom poslužitelju, kako ne bi opterećivala rad samog poslužitelja Weba. Klijent pritom ne mora biti svjestan da se u pozadini zapravo krije nekoliko poslužitelja koji osiguravaju ispravan i nesmetan prikaz informacija koje je zatražio.

S vremenom se tendencija smanjivanja procesa našla na strani klijenta. U takvom se slučaju klijentske mogućnosti reduciraju u tzv. tankog klijenta, koji većinu procesiranja prepuštaju poslužiteljskoj strani te se sami brinu samo za grafičku prezentaciju prema korisniku. To dovodi do smanjenja troškova za opremu na strani klijenta, ali i povećane ovisnosti o mrežnoj komunikaciji. S druge strane, poslužitelji se time dodatno opterećuju, pa se povećava robusnost sustava i troškovi na poslužiteljskoj strani sustava.

5.3.2 Slojevitost

Vratimo se na trenutak korak unatrag te promotrimo kako sličan model možemo primijeniti ako govorimo o aplikaciji. Tradicionalne su aplikacije izvedene obično unutar jednoga procesnog bloka, koji se izvodio na računalu, i iako je unutar same programske izvedbe aplikacije moglo biti određenog raslojavanja (jedan primjer za to je korištenje različitih gotovih biblioteka kako bi se postigla neka funkcionalnost), sve je skupa bilo uklopljeno u zajednički izvršni kod

²³⁷ Lee, J., Brent, W., Open Source Web Development with LAMP: Using Linux, Apache, MySQL, Perl, and PHP. Addison Wesley, 2002.

²³⁸ Rosebrock, E., Filson, E., Setting up LAMP: getting Linux, Apache, MySQL, and PHP working together, John Wiley and Sons, 2004

²³⁹ Siever, E., Figgins, S., Love, R., Linux in a Nutshell, O'Reilly, 2009

²⁴⁰ Coar, Ken A. L., Bowen, R., Apache Cookbook, O'Reilly, 2008.

²⁴¹ Ullman, L., MySQL, Peachpit Press, 2006

²⁴² Schwartz, R. L., Phoenix, T., Foy, B. D., Learning Perl, O'Reilly, 2008.

²⁴³ Vaswani, V., PHP: A Beginner's Guide, McGraw-Hill Professional, 2008.

²⁴⁴ Lutz, M., Learning Phyton, O'Reilly, 2009.

aplikacije. Više takvih aplikacija moglo se, u višeprocennoj izvedbi, izvoditi na jednom ili više računala te se tako stvorila prva raspodijeljena aplikacija. Procesi su u takvom sustavu morali na neki način međusobno komunicirati, za što su se koristili komunikacijski protokoli.

Moderne se aplikacije danas projektiraju kao višeslojne²⁴⁵. Pritom se u svaki sloj odvajaju zasebni, logički izdvojeni dijelovi. Najjednostavniji primjer ovakve arhitekture je tzv. troslojna aplikacija, čiji su sastavni dijelovi prezentacijski, aplikacijski i podatkovni sloj.

Prezentacijski sloj (engl. *Presentation layer*) nalazi se na vrhu ove arhitekture, te se u njemu grafički predočuju podaci prema korisniku aplikacije (najčešće se to izvodi putem tzv. grafičkoga korisničkog sučelja). U ovom se sloju obavlja prihvatanje korisnikovih zahtjeva, transformacija tog zahtjeva u oblik razumljiv nižim slojevima aplikacije te obrnut postupak, kada se rezultati aplikacije prevode u oblik razumljiv korisniku.

Aplikacijski sloj (engl. *Application layer*) često se naziva slojem poslovne logike i predstavlja samu srž aplikacije, jer se u njemu izvode poslovni procesi koje je definirao programer. Ovaj sloj prima podatke s prezentacijskog sloja, obrađuje ih, traži dodatne podatke koji su dostupni u podatkovnom sloju te konačan rezultat ponovno vraća u prezentacijski sloj kako bi ih on mogao prezentirati prema korisniku.

Podatkovni sloj (engl. *Data layer*) izdvojen je dio u ovakvoj arhitekturi jer se na taj način omogućava da više različitih aplikacija ili sustava koristi iste podatke, koji su pohranjeni npr. u poslužiteljima baza podataka. Time se postiže povećanje performansi i veća skalabilnost sustava.

Pored troslojnih aplikacija, pojavljuju se i tzv. višeslojne aplikacije, koje se već približavaju konceptima sustava zasnovanih na uslugama (o čemu ćemo govoriti u sljedećim poglavljima). Primjerice, tvrtke koje u svom poslovnom sustavu koriste različite aplikacije koje pružaju računalnu podršku različitim dijelovima poslovnog procesa žele, što je moguće više, ponovno iskoristiti jednom napisan programski kod. Zbog toga se često funkcije autorizacije korisnika ili očuvanja korisnikove sjednice izdvajaju u zaseban sloj i na jednak se način koriste u svim aplikacijama unutar jedne tvrtke.

5.4 Interoperabilnost kod višeslojnih arhitektura

U informacijskim sustavima današnjice često je primjena sustava s višeslojnim arhitekturama. Kao što je već rečeno, većina aplikacija izvedena nad arhitekturom klijent–poslužitelj ima tri osnovna sloja: prezentacijski sloj, sloj poslovne logike i podatkovni sloj, koji u sebi sadržava i dio za pristupanje podacima. No aplikacije zasnovane na višeslojnim arhitekturama mogu se zasnivati na još većem broju slojeva, odnosno mogu se još detaljnije raslojavati, a najpoznatiji primjeri takvih arhitektura dolaze iz tzv. *enterprise* okruženja, koja uključuju tehnološke platforme kao što su Spring MVC, Java EE i .NET.

²⁴⁵ Alonso, G. et al. *Web Services: concepts, architectures and applications*, Springer, 2004.

U ovom pregledu predstavljeni su slojevi takvih višeslojnih arhitektura koje sadržavaju dodatne slojeve²⁴⁶, prikazani u kontekstu interoperabilnosti, odnosno postojeći slojevi koji su dodatno podijeljeni (Slika 5.4):

- **klijentski sloj** (engl. *client tier*) ili **prezentacijski sloj na klijentu** – grafički dio aplikacije na klijentu, moguće izvedbe uključuju:
 - **tanki klijent** (engl. *thin application client*) – nekad najčešće temeljen na terminalu, a danas na pregledniku (engl. *browser*) koji se izvodi na klijentskom računalu manje snage; služi za iscrtavanje grafičkoga korisničkog sučelja i izvođenje skriptnih jezika, kao i za izvedbu prezentacijske logike, danas već često s vrlo složenim grafičkim elementima,
 - **bogati aplikacijski klijent** (engl. *rich application client*) ili **debeli klijent** (engl. *fat client* ili *thick client*) – samostalni aplikacijski klijent koji se izvodi na klijentskim računalima veće snage, a koji omogućava funkcionalnosti bogatoga grafičkog korisničkog sučelja, često uz standardne elemente poput različitih oblika elemenata, izbornika, alatnih traka i stablastih prikaza; sadržava i napredne mogućnosti kao što su povuci-i-spusti (engl. *drag-and-drop*), kontekstno osjetljivi dijelovi prikaza, različite mogućnosti manipulacije grafičkim elementima i slikama, iscrtavanja vektorske 2D i 3D grafike i slično,
 - **pametni klijent** (engl. *smart client*) – kombinacija evoluiranoga bogatog aplikacijskog klijenta i tankog klijenta, koji omogućuje prikaz korisničkog sučelja i prezentacijske logike bez uporabe preglednika,
- **prezentacijski sloj na poslužitelju** (engl. *server presentation tier*) – omogućuje prikaz korisničkog sučelja, prihvaća sve interakcije od korisnika i upravlja grafičkim prezentacijama, pristupu elementima sučelja, osnovnim transformacijama prikazanih i primljenih podataka (jezik, oblici datuma, vremena i sl.), osnovnim validacijama unesenih podataka, ispravcima korisnikovih akcija, stanjima interakcije te pozivima metoda poslovne logike; češći kod izvedbe tankih klijenata,
- **sloj poslovne logike** (engl. *business logic tier*) – služi za izvođenje poslovnih procesa, često se može podijeliti i na nekoliko podslojeva, od onih koji su više vezani uz primanje i pružanje podataka prezentacijskom sloju, preko same poslovne logike koja radi neke izračune i čini „pamet“ aplikacije, do onih kojih vezani uz primanje i pružanje podataka sloju koji upravlja podacima,
- **sloj pristupa podacima** (engl. *data access tier*) – koristi sve razine apstrakcije prikaza podataka koje prima ili šalje podatkovnom sloju, najčešće se u modernim programskim objektnim jezicima koriste objektni prikazi pomoću podatkovnih objekata, a mogući su i drukčiji oblici; u slučaju objektnog prikaza i podatkovnog sloja u obliku relacijske baze podataka najčešće se koristi neki oblik objektno-relacijskog mapiranja (ORM, engl. *object-relational mapping*),
- **podatkovni sloj ili sloj podataka** (engl. *data tier*) – u ovoj se podjeli označava sloj koji pohranjuje i upravlja podacima na fizičkoj i logičkoj razini, a najčešće je predstavljen relacijskim bazama podataka, odnosno sustavima za upravljanje podacima baze podataka (DBMS, engl. *database management system*), iako su mogući bilo kakvi sustavi za pohranu podataka ili izvori podataka, kao što su sustavi za upravljanje

²⁴⁶ Application interoperability overview u *IBM WebSphere and Microsoft .NET Interoperability*, IBM Redbook, IBM, 2006., <http://www.redbooks.ibm.com/redbooks/pdfs/sq246799.pdf>

sadržajem (CMS, engl. *content management system*), sustavi za upravljanje dokumentima (DMS, engl. *document management system*), skladišta podataka (engl. *data-warehouse*), objektne baze podataka, XML baze podataka ili općenito datoteke u nekom obliku.

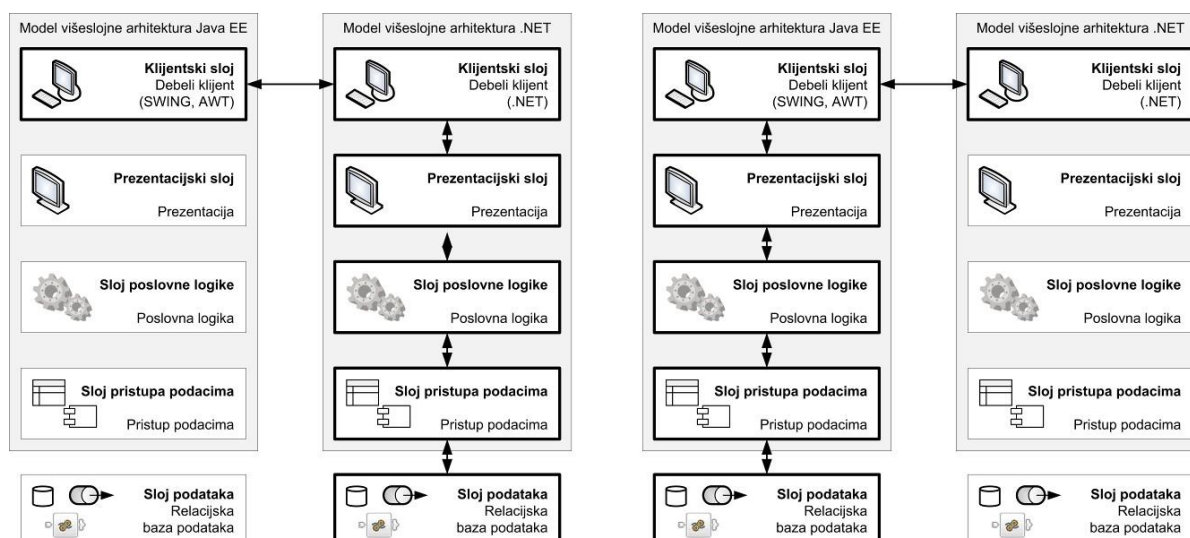


Slika 5.4: Slojevi modela višeslojne arhitekture

S obzirom na raslojenost ovakvog oblika arhitekture možemo zaključiti da je moguće ostvariti povezivanje slojeva izvedenih u različitim tehnologijama, a time i interoperabilnost, pa ćemo ovdje navesti primjere povezivanja dvaju slojeva izvedenih u različitim tehnologijama, kao što su Spring MVC, Java EE i .NET. Krenut ćemo od viših slojeva, s korisnikova gledišta, prema nižima.

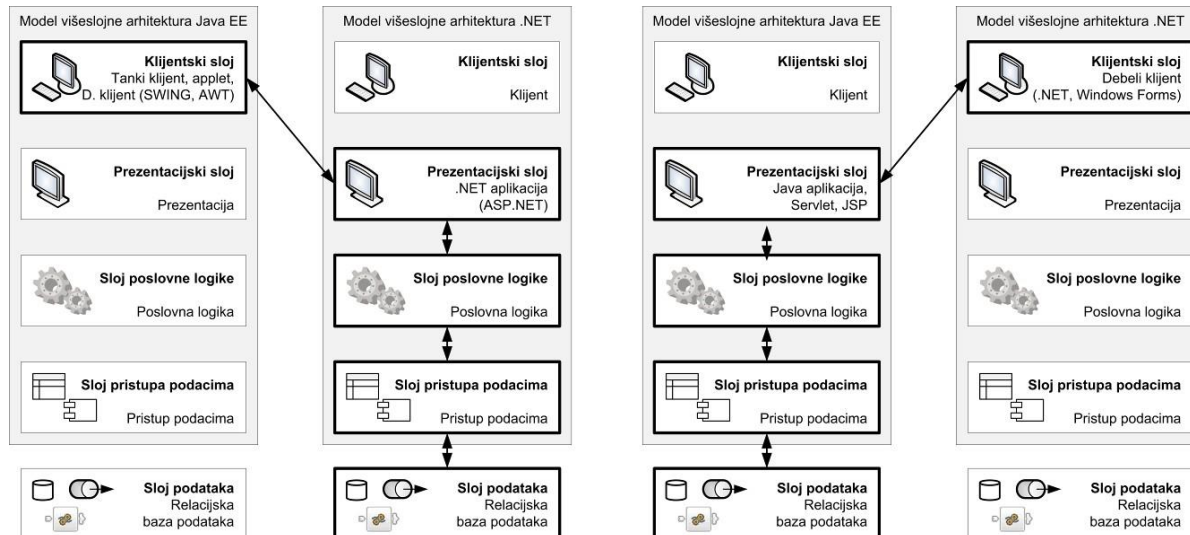
5.4.1 Interoperabilnost klijentskog sloja

Na razini klijentskog sloja interoperabilnost ovisi o vrsti klijenta. Prvo ćemo razmotriti **interoperabilnost dvaju klijenata**, koja je moguća, ali u praksi nije često izvedena. Iako je moguća interoperabilnost dvaju tankih klijenata izvedena u različitim tehnologijama, ovakav odnos nije uobičajen. U slučaju dvaju debelih klijenata ili bogatih aplikacijskih klijenata odnos je nešto vjerojatniji, jer debeli klijent može eksponirati neke svoje funkcionalnosti kako bi ih drugi mogli koristiti (Slika 5.5). U slučaju arhitekture Spring MVC ili Java EE može biti riječ o klijentu izvedenom u JavaFX, Swing ili AWT tehnologiji, Angular, ReactJS ili Vue.js, a u slučaju arhitekture .NET o debelom klijentu za operacijski sustav Windows. Postoji i mogućnost odnosa tankog klijenta i debelog klijenta, gdje uporabom protokola HTTP tanki klijent koristi neke funkcionalnosti debelog klijenta, kada debeli klijent na neki način zamjenjuje i prezentacijski sloj.



Slika 5.5: Interoperabilnost klijentskog sloja – interakcija dvaju debelih klijenata

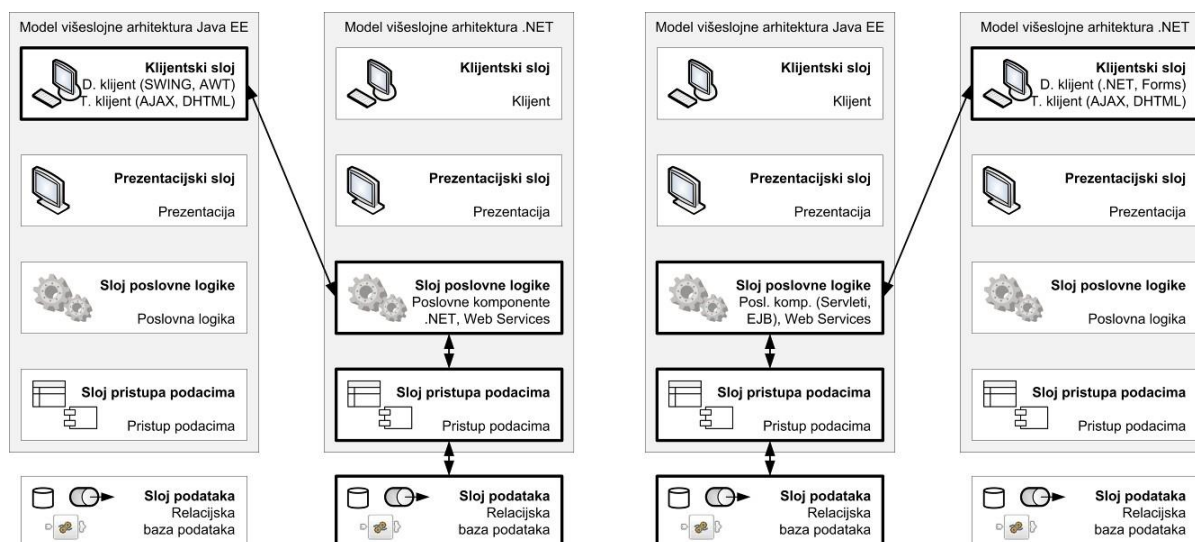
Interoperabilnost klijentskog sloja prema prezentacijskom sloju je vjerojatnija mogućnost (Slika 5.6). Primjer takvog klijenta je aplikacija u jeziku Java izvedena kao tanki klijent, *applet* u jeziku Java ili debeli klijent izveden u JavaFX, Swing ili AWT, Angular, ReactJS ili Vue.js tehnologiji koji komunicira s .NET aplikacijom, primjerice ASP.NET artefaktom. Primjer u suprotnom smjeru bio bi debeli klijent, odnosno aplikacija izvedena u .NET tehnologiji, primjerice Windows Forms aplikacija koja komunicira s Java aplikacijom, Servletom, JSP stranicom, Angular, ReactJS ili Vue.js klijentskim aplikacijama.



Slika 5.6: Interoperabilnost klijentskog i prezentacijskog sloja

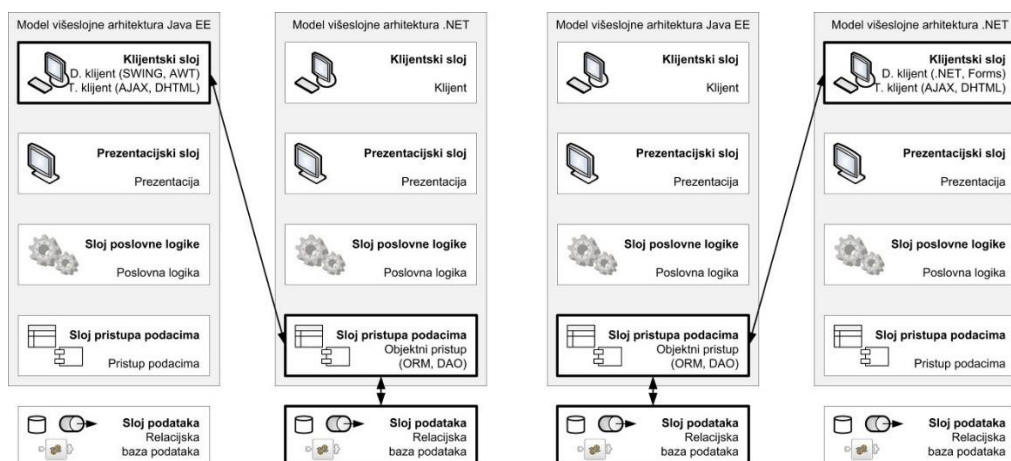
Klijentski se sloj može spojiti, odnosno komunicirati i izravno sa slojem poslovne logike (Slika 5.7) kada govorimo o **interoperabilnosti klijentskog sloja prema sloju poslovne logike**. U tom se slučaju praktički odustaje od prezentacijskog sloja, odnosno on je integriran u klijentskom sloju. Osnovni primjer takve interoperabilnosti bio bi izravan spoj klijentske Java aplikacije ili debelog klijenta izvedenog u JavaFX, Swing ili AWT tehnologiji koji komunicira s komponentama poslovnog sloja razvijenih u .NET tehnologiji. Obrnuti pristup bio bi kad bi .NET klijentska aplikacija ili Windows Forms aplikacija komunicirala s komponentama poslovnog

sloja arhitekture Spring MVC, Java EE, kao što su Servleti ili komponente Enterprise JavaBeans (EJB). Druga mogućnost je izvedba s tankim klijentom, odnosno uporabom preglednika i klijentskog Web sučelja koje koristi objekt XMLHttpRequest za komunikaciju s poslovnom logikom. Korištenje serijalizacije i deserijalizacije XML objekata na razini preglednika omogućava načelo uporabe asinkronih poziva korištenjem jezika JavaScript i jezika XML (AJAX, engl. *Asynchronous JavaScript and XML*), a na strani poslovne logike kao sučelje se mogu ostvariti usluge Web-a temeljene na protokolu SOAP, REST i drugim tehnologijama Web Services. Na klijentskom se sloju može koristiti skup tehnologija DHTML koje izravno pozivaju usluge Web-a.



Slika 5.7: Interoperabilnost klijentskog sloja i sloja poslovne logike

Ako se svi viši slojevi integriraju u jednoj cjelini klijentske aplikacije, dobit ćemo klasičnu dvoslojnu arhitekturu. U tom je slučaju moguće da klijentski sloj pristupa izravno sloju pristupa podacima pa tada možemo govoriti o **interoperabilnosti klijentskog sloja i sloja poslovne logike**, no taj se način komunikacije ne preporučuje. Ponovno je moguć scenarij korištenja debelog klijenta ili tankog klijenta koji korištenjem načela AJAX ili DHTML-a pristupaju objektnoj reprezentaciji podataka, što je moguće ostvariti između arhitektura .NET, Spring MVC i Java EE u oba smjera (Slika 5.8 – interoperabilnost).



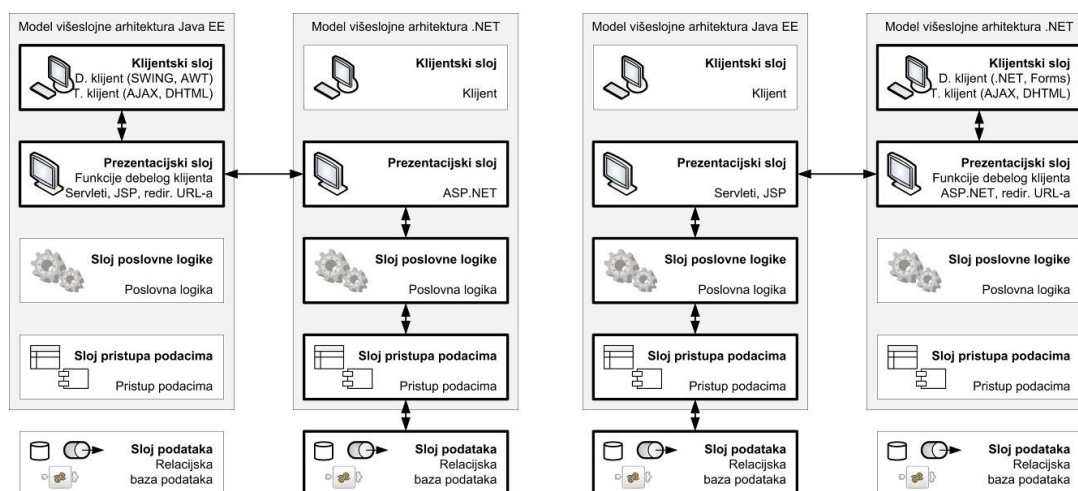
Slika 5.8: Interoperabilnost klijentskog sloja i sloja pristupa podacima

Kada se govori o interoperabilnosti klijentskog sloja, najčešće se govori o komunikaciji prema prezentacijskom sloju ili prema sloju poslovne logike, dok je komunikacija prema drugom klijentskom sloju ili prema sloju pristupa podacima vrlo rijedak slučaj, koji se ne preporučuje iz niza razloga. Bez obzira na to kako je izveden klijentski sloj, u obliku debelog ili tankog klijenta, moguće ga je povezati s komponentama nižih slojeva izvedenih u arhitekturama .NET, Spring MVC i Java EE. S jedne strane to su najčešće aplikacije Weba, Java aplikacije, *applet*i, Swing ili AWT aplikacije, odnosno .NET aplikacije, Windows Forms, ASP.NET i slični oblici, a s druge strane razni oblici prezentacijskih komponenata, kao što su Servleti, JSP stranice, Angular, ReactJS ili Vue.js, komponente EJB, odnosno ASP.NET artefakti i druge komponente .NET platforme. Za komunikacije se mogu koristiti različiti oblici protokola i usluga, uključujući Web Services, a kod tankih su klijenata sve popularniji načelo AJAX i skup tehnologija DHTML.

5.4.2 Interoperabilnost prezentacijskog sloja

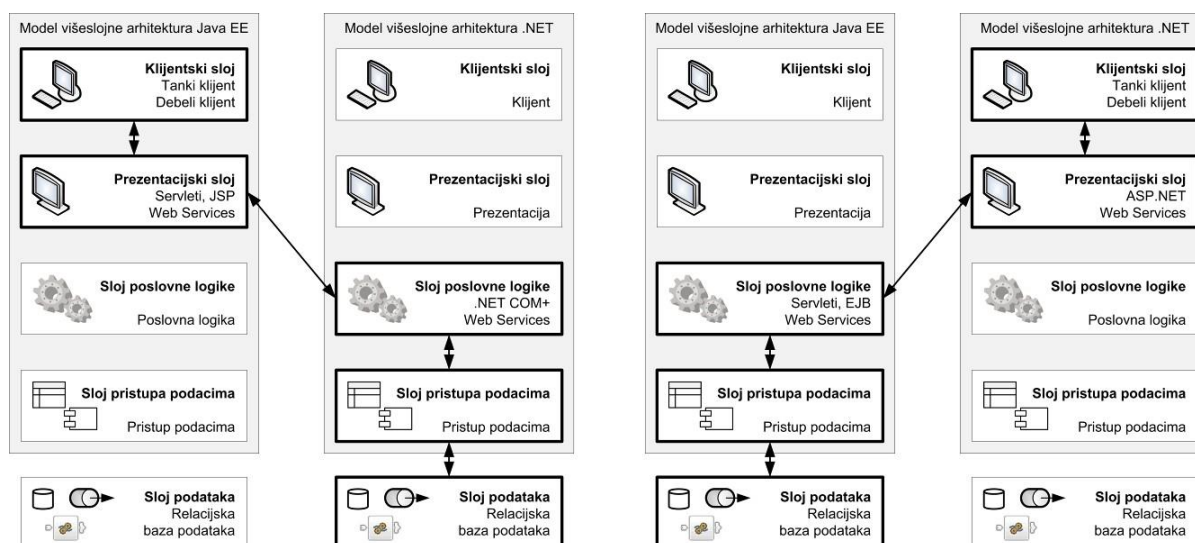
S obzirom na to da prezentacijski sloj osim prikaza korisničkog sučelja omogućuje i upravljanje, koordinaciju i orkestraciju elemenata i komponenata korisničkog sučelja, logično je da se i s ovog sloja može ostvariti komunikacija s drugim slojevima različitih tehnologija ili arhitektura. Prezentacijski je sloj zadužen i za interpretaciju svih korisničkih akcija, grafički prikaz svih elemenata, ali i održavanje stanja usluge tijekom korisničkih sjednica, toka procesa poslovne logike i poziva udaljenih metoda ili usluga. Kod dvoslojnih aplikacija prezentacijski je sloj zapravo ugrađen u klijentski sloj, pa za njega vrijedi sve što i za interoperabilnost klijentskog sloja.

Prezentacijski sloj može komunicirati s drugim prezentacijskim slojem (Slika 5.9), pa tada govorimo o **interoperabilnosti dvaju prezentacijskih slojeva**. U primjeru s arhitekturama Spring MVC, Java EE i .NET to znači da ako je riječ o tankom klijentu komponente, kao što su Servleti ili JSP stranice, Angular, ReactJS ili Vue.js, pristupaju komponentama ASP.NET i obrnuto. S obzirom na probleme održavanja korisničke sjednice (engl. *session*) i stanja aplikacije, ovo nije često ni popularno rješenje. No, jedna od mogućih implementacije je korištenje redirekcije URL-a. Nešto je češća interoperabilnost komunikacijom dvaju debelih klijenata, primjerice Java aplikacije i .NET aplikacije, gdje jedna strana poziva određene funkcionalnosti druge strane.



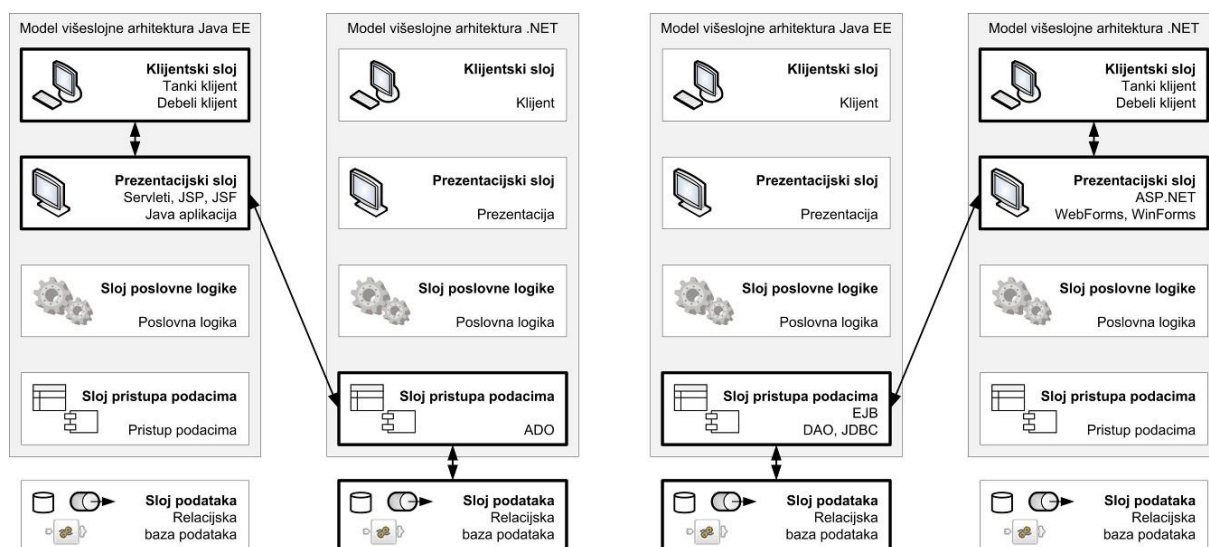
Slika 5.9: Interoperabilnost dvaju prezentacijskih slojeva

Komunikacija prezentacijskog sloja jedne tehnologije i poslovnog sloja druge tehnologije najčešći je scenarij interoperabilnosti između tehnologija Spring, Java EE i .NET, pogotovo kod aplikacijskih poslužitelja. U tom slučaju prezentacijski sloj izveden pomoću JSP stranica, Servleta i komponenti JavaBean, ili Angular, ReactJS ili Vue.js, izravno komunicira s komponentama .NET tehnologija, kao što su COM+. U obrnutom slučaju prezentacijski sloj izveden u ASP.NET tehnologiji izravno komunicira sa Servletima i komponentama Enterprise JavaBean (EJB). Druga mogućnost izvedbe **interoperabilnosti prezentacijskog i poslovnog sloja** je uporabom usluga Weba kao što je skup tehnologija Web Services, što je i preporučeni oblik komunikacije u arhitekturi orijentiranoj prema uslugama. Slika 5.10 prikazuje oba slučaja, a o tehnologiji Web Services čut ćete više u sljedećem poglavlju.



Slika 5.10: Interoperabilnost prezentacijskog i poslovnog sloja

Interoperabilnost prezentacijskog i sloja pristupa podacima moguća je kada je uloga sloja poslovne logike minimalna. Primjer takve aplikacije je izravna veza prezentacije, recimo kroz korisničko grafičko sučelje na Webu, i podataka kojima se manipulira, predstavljenih od podatkovnog sloja. Najčešće primjene su kod upravljanja podacima, administracije matičnih podataka pohranjenih u bazama podataka i drugih oblika kada je složenija poslovna logika nepostojeća (Slika 5.11). U slučaju prezentacijskog sloja izvedenog u tehnologijama JSP ili JSF moguć je izravan pristup podatkovnim objektima u tehnologiji ADO.NET (engl. *ActiveX Data Objects*). U slučaju prezentacijskog sloja izvedenog u tehnologijama ASP.NET, WebForms ili WinForms pristupalo bi se podatkovnim objektima u tehnologijama Spring MVC ili Java EE, kao što su objekti EJB (engl. *Enterprise JavaBeans*) i DAO (engl. *Data Access Objects*), ili izravno korištenjem konekcije JDBC.



Slika 5.11: Interoperabilnost prezentacijskog sloja i sloja pristupa podacima

Kombinacija u kojoj prezentacijski sloj može izravno pristupiti sloju podataka je u praksi vrlo rijetka i neuobičajena pa je ovdje nećemo ni obrađivati.

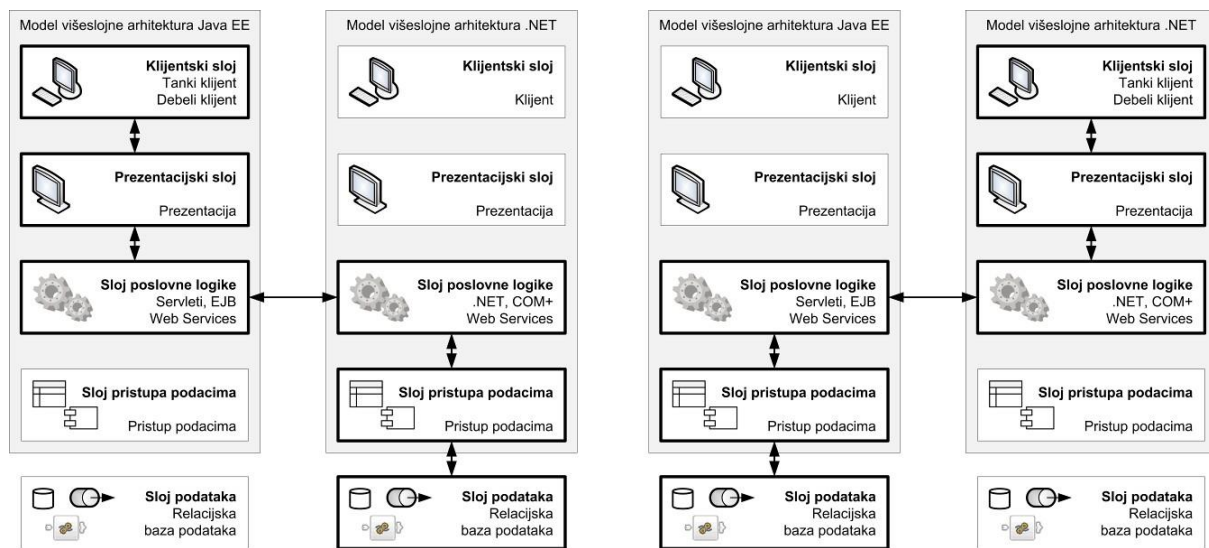
5.4.3 Interoperabilnost poslovnog sloja

Poslovni sloj, koji odrađuje većinu poslovne logike, vrlo često komunicira izravno s drugim poslovnim slojem. Kada kažemo da ovaj sloj odrađuje većinu poslovne logike, onda se misli na to da ipak određeni manji dio poslovne logike može odraditi i prezentacijski sloj u kontekstu validacija, kontrola i povezivanja podataka te oblikovanja struktura podataka koje se dalje prenose u poslovni sloj, kao i sloj pristupa podacima koji također validira, kontrolira i preoblikuje podatke koji se pohranjuju u sloju podataka ili dohvaćaju iz njega. U određenim slučajevima sama relacijska baza podataka sadržava različite procedure i mehanizme, kao što su okidači (engl. *triggers*) i ugrađene ili pohranjene procedure (engl. *stored procedures*), koji također odrađuju dio poslovne logike, ali takvim se slučajevima nećemo potanje baviti u ovom pregledu. Slučajevi interoperabilnosti poslovnog sloja s klijentskim slojem i prezentacijskim slojem obrađeni su već u prethodnim potpoglavljima, pa se ovdje usredotočujemo na interoperabilnost dvaju poslovnih slojeva i poslovnog sloja sa slojem pristupa podacima.

Komunikacija dvaju poslovnih slojeva kao osnovni razlog ima redukciju ponavljanja istih ili sličnih funkcionalnosti i ponovno iskorištavanje dijelova poslovne logike radi smanjenja troškova i drugih koristi. **Interoperabilnost poslovnih slojeva** je česta na razini B2B, kada sustavi različitih poslovnih subjekata ili sustavi istoga poslovnog subjekta surađuju kao bi ostvarili određenu cjelovitu uslugu. Primjeri unutar jednog poslovnog subjekta obično uključuju suradnju sustava ERP (engl. *Enterprise Resource Planning*), CRM (engl. *Customer Relationship Management*), HR (engl. *Human Resources*) i drugih sustava, a najčešći primjeri između različitih subjekata su sustavi naručivanja, fakturiranja i plaćanja, koji omogućavaju elektroničku trgovinu i elektroničko poslovanje u cjelini.

Kod suradnje komponenti temeljenih na Spring MVC, Java EE i .NET tehnologijama riječ je o suradnji komponenti Servleta, EJB i COM+ koje mogu uspješno surađivati. Slika 5.12 prikazuje

takav slučaj, ali i drugi slučaj u kojem su poslovne komponente eksponirane u obliku usluga korištenjem tehnologije Web Services.

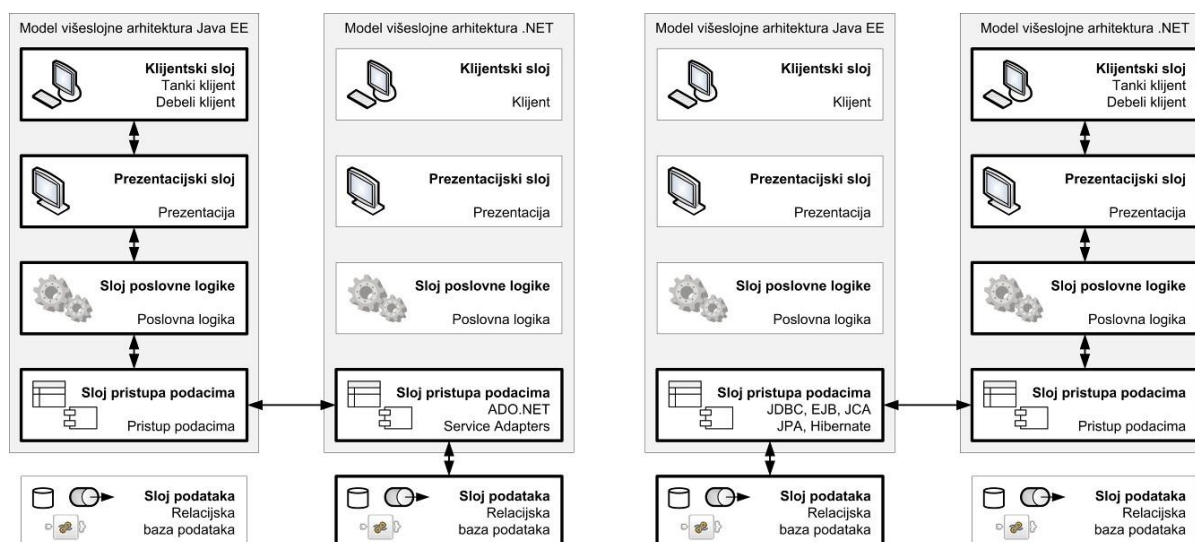


Slika 5.12: Interoperabilnost poslovnih slojeva

Iako je češća interoperabilnost na razini poslovnih slojeva, moguće je da poslovne komponente jedne tehnologije izravno pristupaju sloju podataka izvedenom u drugoj tehnologiji. Tada govorimo o **interoperabilnosti poslovnog sloja i sloja pristupa podacima**, koji se kod poslovnih komponenti u tehnologijama Spring MVC ili Java EE predstavljaju komponentama Servleta, EJB ili nekim drugim oblikom poslovnih objekata koji pristupaju komponentama ADO.NET ili agentima usluga (engl. *Service Agents*). Ako je riječ o komponentama COM+, onda se spoj na Java EE tehnologije ostvaruje korištenjem arhitekture za spajanje JCA (engl. *Java Connector Architecture*).

5.4.4 Interoperabilnost sloja pristupa podacima i sloja podataka

S obzirom na to da smo u prethodnim potpoglavljima obradili sve oblike interoperabilnosti sloja pristupa podacima iz drugih slojeva, osim između dvaju takvih slojeva, ovdje ćemo razmotriti i tu mogućnost (Slika 5.13). **Interoperabilnost slojeva pristupa podacima** kod tehnologije .NET za pristup i pohranu podataka koristi ADO.NET, a za pristup uslugama koriste se adapteri (engl. *Service Adapters*). Kod tehnologije Spring MVC ili Java EE za perzistenciju podataka koriste se komponente EJB, JDO (engl. *Java Data Objects*), JDBC i razna objektno-relacijska mapiranja, uključujući *Java Persistence API* s POJO (engl. *Plain Old Java Object*) i *Hibernate*.



Slika 5.13: Interoperabilnost sloja pristupa podacima

Postoji još jedna mogućnost, a to je da se **interoperabilnost** događa na samom **sloju podataka**. Ovo bi značilo da ako je riječ o relacijskim bazama podataka, jedna baza zove podatke iz druge. Iako je ovo tehnički izvedivo, upitno je kako izvesti ovaj oblik interoperabilnosti. Najčešće se koriste dijeljene baze podataka (engl. *shared databases*) koje se teško ostvaruju u heterogenim, raspodijeljenim i fizičkim dislociranim sustavima. Moguće je i da određena logika u pohranjenim procedurama koristi razne izvore podataka i tako omogućava interoperabilnost. Jednako tako, u transakcijskoj je logici često da jedna poslovna komponenta koristi niz različitih komponenti pristupa podacima koji tada izvode dohvat, izmjenu ili pohranu podataka na više različitih izvora podataka.

5.5 Tehnologije izvedbe interoperabilnosti

Ako sagledamo sve postojeće tehnologije interoperabilnosti, uočiti ćemo nekoliko glavnih smjerova²⁴⁷, odnosno pristupa kod izvedbe interoperabilnosti:

- uporabom **usluga** (engl. *service-oriented approach*) – najčešće izvedeno kao usluge Web tehnologijama Web Services,
- na razini **klasa** (engl. *class level approach*) – najčešće izvedeno:
 - **križnim prevođenjem** (engl. *cross compilation*) – uporabom međukoda, kao što je Java bytecode ili MSIL (*Microsoft Intermediate Language*),
 - **pozivima udaljenih procedura** (engl. *Remote Procedure Calls*) – uporabom metoda, kao što su RMI (*Remote Method Invocation*) ili .NET remoting,
 - **premošćivanjem** (engl. *bridging*) – međusobnim pozivima klasa u tehnologijama Java i .NET koje se izvode u JVM-u (*Java Virtual Machine*) i CLR-u (*Common Language Runtime*) putem mosta (engl. *bridge*),

²⁴⁷ Application interoperability overview u *IBM WebSphere and Microsoft .NET Interoperability*, IBM Redbook, IBM, 2006., <http://www.redbooks.ibm.com/redbooks/pdfs/sg246799.pdf>

- **porukama** (engl. *messaging*) i **redovima poruka** (engl. *message queuing*) – razmjena poruka na siguran i pouzdan način korištenjem mehanizama redova poruka, kao što su produkti IBM WebSphere MQ, ActiveMQ, RabbitMQ ili MSMQ,
- na razini **podataka** – razmjenom podataka na razini izvora podataka uporabom različitih posrednika, međuopreme, agregatora i integracijskih tehnika te sinkronizacija i replikacija baza podataka, kao i uporabom dijeljenih baza podataka.

Na temelju prethodne podjele i najčešće prakse može se izvesti popis najčešće korištenih tehnologija za postizanje interoperabilnosti²⁴⁸:

- **Tehnologije (XML) Web Services** – omogućavaju interoperabilnost na razini usluga, gdje aplikacije ili sustavi mogu razmjenjivati podatke korištenjem eksponiranih sučelja u obliku tehnologija Web Services. Razmjena podataka događa se porukama u obliku XML, a za izvedbu se koriste postojeći protokoli, poput HTTP, SOAP i REST, koji su široko prihvaćeni i relativno jednostavni za implementaciju kod svih sudionika u komunikaciji.
- **Tehnike eksponiranja razreda i metoda drugim tehnologijama** – omogućavaju interoperabilnost na razini prosljeđivanja poziva metoda jedne platforme u drugu, odnosno iz jednoga programskog jezika u drugi. Osnovni preduvjet za ovakav oblik interoperabilnost je postojanje mosta između tehnologija (engl. *runtime bridge*), najčešće ostvarenog kao zasebno rješenje, ili uporaba tehnika poziva udaljenih procedura (engl. *remote procedure call*).
- **Međuoprema orijentirana prema porukama** (engl. *message-oriented middleware*) – omogućava interoperabilnost na razini asinkrone razmjene poruka, potpomognute drugim funkcionalnostima, kao što su transakcijska okolina, sigurnost u smislu kriptiranja podataka i autentikacije korisnika, tolerancije na mrežne nedostupnosti i siguran te pouzdan prijenos podataka. Nedostaci ovakvih tehnologija su teže ostvariva sinkrona komunikacija i potencijalni problemi s mrežnom sigurnošću (vatrozid).
- **Dijeljene baze podataka** (engl. *shared databases*) – omogućavaju interoperabilnost na razini razmjene pohranjenih zapisa s podacima. Za ovu se metodu koriste razni oblici programskih sučelja za pristup bazama podataka, kao što su ODBC (*Open Database Connectivity*) ili JDBC (*Java Database Connectivity*) i mehanizmi sinkronizacije baza podataka. Preduvjet je da postoje zajednička dijeljenja jedinstvene baze podataka koja koriste istu shemu, odnosno model podataka koji razumiju svi sudionici u komunikaciji.
- **Integracijski brokeri** (engl. *integration brokers*) – omogućavaju interoperabilnost kroz razmjenu podataka na aplikacijskoj razini. Brokeri povezuju heterogene tehnologije korištenjem izgrađenih sučelja i adaptera te služe kao međusloj između različitih aplikacija ili sustava.

5.5.1 Međuoprema

Pojam **međuopreme** (engl. *middleware*) ponovno je jedan od onih koji nemaju jedinstvenu definiciju pa ćemo ga objasniti kroz različite poglede. Neka će od najjednostavnijih objašnjenja slikovito opisati međuopremu kao „/“ (engl. *slash*) u pojmu „klijent/poslužitelj“, „ono

²⁴⁸ Application Interoperability: MS.NET and J2EE, Microsoft Press, <http://msdn.microsoft.com/en-us/library/ff650279.aspx>

nešto između“ ili čak „softversko ljepilo“ (engl. *software glue*)²⁴⁹. No to nam ne pomaže shvatiti što je to, već samo čemu služi. Stoga ćemo navesti neke najpoznatije definicije:

„Međuoprema je softver koji **posreduje između aplikacijskog programa i mreže. Upravlja interakcijom nesukladnih aplikacija kroz heterogene računalne platforme.**“²⁵⁰

„Međuoprema je računalni softver koji **spaja softverske komponente i njihove aplikacije. Sastoji se od skupa usluga koje omogućavaju izvođenje višestrukih procesa na jednom ili više računala koje međusobno surađuju. Omogućava interoperabilnost kao podršku prelasku na usklađene raspodijeljene arhitekture koje izvode kompleksne raspodijeljene aplikacije.**“²⁵¹

U prethodnim su definicijama podebljanim slovima označene važne riječi, pa tako uočavamo pojmove interoperabilnost te procese posredovanja, spajanja i interakcije, a kao objekte ili subjekte uočavamo nesukladne aplikacije, softverske komponente i heterogene platforme. Konačna definicija koju bismo mogli izvesti je sljedeća:

Međuoprema je programska podrška (softver) koja posreduje između dviju strana i tako omogućuje da pojedine softverske komponente ili aplikacije surađuju u jedinstvenom funkcionalnom smislu, bez obzira na to što je riječ o nesukladnim dijelovima koji se izvršavaju na tehnološki nespojivim platformama. Osim naziva međuoprema često se kao sinonimi koriste **međuprogram, međuprogramska podrška, posrednička programska podrška i posrednički međusloj.**

Uloga međuopreme je omogućavanje aplikacijama ili informacijskim sustavima da transparentno komuniciraju bez obzira na fizičku udaljenost i nesukladnost komunikacije i platformi na kojima su izvedeni. Neovisnost o infrastrukturi mreže i pojednostavljena komunikacija uz zadržavanje semantičke interoperabilnosti neke su od najčešćih koristi međuopreme.

Osnovni tipovi međuopreme mogu se podijeliti na:

- **podatkovnu međuopremu** (engl. *database middleware*) – spoj između podatkovnih izvora i isporuka podataka, najčešće povezuju baze podataka i aplikacije,
- **objektno orijentiranu međuopremu** (engl. *object-oriented middleware*) – primanje i slanje objekata te zahtijevanja usluga u sustavima objektno orijentiranog okruženja,
- **međuoprema zasnovana na pozivima udaljenih procedura** (engl. *remote procedure call middleware*) – pozivi procedura koji se nalaze na udaljenim računalima, asinkronim i sinkronim načinom,
- **transakcijska međuoprema** (engl. *transaction middleware*) – razmjena podataka zasnovana na transakcijama, što omogućuje objedinjavanje skupina zahtjeva i poruka, a time i posljedičnih akcija,

²⁴⁹ What is middleware?, MRC, Defining Technology, <http://www.middleware.org/whatis.html>

²⁵⁰ Middleware, Free On-Line Dictionary of Computing, <http://foldoc.org/middleware>

²⁵¹ Middleware, Wikipedia, <http://en.wikipedia.org/wiki/Middleware>

- **međuprema orijentirana prema porukama, MOM** (engl. *message-oriented middleware*) – omogućuje razmjenu poruka opće namjene pohranom poruka u međuspremniku dok ih druga strana ne preuzme,
- **portalski** (engl. *portals*) i **aplikacijski poslužitelji** (engl. *application servers*) – poslužiteljski softver koji omogućuju izvršavanje drugih aplikacija, najčešće izvedenih s Web korisničkim sučeljima.

Kod međupreme orijentirane prema porukama podaci se razmjenjuju korištenjem poruka različitim komunikacijskim protokolima i izvedbama redova poruka. Većina takvih softvera koristi mehanizme sigurne i pouzdane isporuke poruka. Podržane su sinkrona i asinkrona komunikacija poruka, ali se ipak najčešće koristi asinkrona komunikacija, pogotovo kod redova poruka.

5.5.2 Komunikacija zasnovana na porukama

Jedna od najčešćih izvedbi međupreme zasnovane na porukama je korištenjem arhitekture **pohrane i prosljeđivanja** (engl. *store-and-forward*) i **redova poruka** (engl. *message queues*), koja se tada naziva **međupremom zasnovanom na redovima poruka** ili **MQM** (engl. *Message Queuing Middleware*). Osim međupreme zasnovane na redovima poruka, postoji i manji broj izvedbi međupreme koja poruke šalje prema većem broju primatelja u sveodredišnoj (engl. *broadcast*) ili višeodredišnoj (engl. *multicast*) komunikaciji.

U arhitekturi MQM postoje softverske komponente na svakoj strani koja sudjeluje u komunikaciji. Redovi poruka su međuspremници koji stoje između strana u komunikaciji i prihvaćaju, odnosno isporučuju poruke. Dva su osnovna tipa sudionika u pojedinoj jednosmjernoj komunikaciji, **pošiljatelji** (engl. *sender*) ili **proizvođači** (engl. *producer*), koji pohranjuju poruke u red poruka, i **primatelji** (engl. *receiver*) ili **potrošači** (engl. *consumer*), koji dohvaćaju poruke iz tog reda poruka. Komunikacija je u načelu asinkrona, a komunikacijski protokol nije nužno određen, već je komunikaciju moguće izvesti korištenjem nekih od poznatih komunikacijskih protokola.

Svaki red poruka služi kao međuspremnik za pristigle, a nepreuzete poruke. Pošiljatelji pohranjuju poruke u red poruke, gdje se one nalaze dok ih netko ne preuzme. U većini implementacija ne mora postojati istovremena dostupnost obje strane već se poruka pohranjuje na neodređeno vremensko razdoblje prije nego što se dohvati. Poruka se iz reda poruka dohvaća u onom trenutku kad je primatelj postao dostupan. No iako red poruka prihvaća poruke u nekom poretku, poruke se iz reda ne moraju nužno dohvaćati u istom poretku. Primatelj može dohvaćati poruke iz reda poruka u poretku koji njemu odgovara. Poruke najčešće sadržavaju informaciju o prioritetu koji može postaviti pošiljatelj. No, posrednik može upravljati porukama koje su pristigle, pa tako može i naknadno promijeniti taj prioritet.

Osnova arhitekture MQM je da je prijenos poruka pouzdan, odnosno da se posrednik oporavlja od pogrešaka koje bi mogle rezultirati promjenom ili gubitkom poruke, odnosno njenom pogrešnom isporukom. U većini slučajeva koriste se **postojani redovi** koji kao tehnološku podlogu za pohranu poruka koriste bazu podataka ili datotečni sustav.

Prednosti korištenja međuopreme zasnovane na porukama su svakako vezane uz izbjegavanje problema s povremenom nedostupnošću sudionika komunikacije i prekida u mrežnoj komunikaciji. Mehanizmi koji se koriste osiguravaju pouzdanu isporuku poruka iako je komunikacijski sustav koji se koristi zasnovan na uobičajenim (nepouzdanim) komunikacijskim protokolima iskorištenim za prijenos poruka. Moguće je koristiti dodatne mogućnosti privremene pohrane, usmjeravanja, pretvorbe i upravljanja prioritetom isporuke poruka. Za to se koristi manja količina komunikacije na nadzornoj razini nego kod nekih drugih načina komunikacije i pouzdanog prijenosa podatkovnih objekata. No postoje i određeni nedostaci ovakve komunikacije, koji se prije svega očituju u relativno visokoj cijeni kvalitetnih komercijalnih rješenja. Danas postoji i niz implementacija izvedenih u otvorenom kodu, no one često ne sadržavaju napredne mogućnosti komercijalnih rješenja. Za izvedbu kvalitetnog rješenja često su potrebna skupa i specifična klijentska sučelja i još skuplji namjenski poslužitelji. Takvo stanje je rezultat i relativno slabe normiranosti u ovom području, pa postoji niz vlasničkih (engl. *proprietary*) rješenja.

Jedna od poznatijih oblika međuopreme zasnovane na porukama izvedene u programskom jeziku Java je programsko sučelje **JMS**²⁵² (engl. **Java Message Service**). JMS je dio platforme Java EE. Omogućava slanje poruka između više klijenata prema modelu od točke do točke i objava-pretplata. Prema modelu **od-točke-do-točke** (engl. *point-to-point*) pošiljatelj šalje poruku u određeni red poruka, a primatelj je dohvaća iz tog reda. Pretpostavka je da pošiljatelj zna u koji red poruka mora poslati poruku, a primatelj poruku dohvaća asinkrono kad želi. Kod modela **objava-pretplata** (engl. *publish-subscribe*) poruke se ne šalju određenom primatelju već se objavljuju pod određenom temom (engl. *topic*). Primatelji su zapravo pretplatnici na određene teme i općenito primatelj i pošiljatelj ne znaju jedan za drugoga. Primatelji mogu biti višestruki, odnosno svi koji su pretplaćeni na određenu temu. Najpoznatije implementacije sučelja JMS uključuju Apache MQ, RabbitMQ, OpenJMS, IBM WebSphere MQ, Oracle AQ i druge.

Još jedna od poznatijih otvorenih normi u vezi s redovima poruka je protokol **AMQP**²⁵³ (engl. *Advanced Message Queuing Protocol*). AMQP je otvorena norma protokola aplikacijskog sloja međuopreme zasnovane na redovima poruka koja definira i posredničke usluge prema modelu nazvanom Advanced Message Queuing Protocol Model. Protokol se definira na „razini žice“ (engl. *wire-level*), što znači da ne opisuje programsko sučelje, već ga nadopunjuje konverzijskim sekvencijama bajtova koji se razmjenjuju mrežom kako bi ga mogle koristiti mnoge druge tehnologije. Jedna od poznatijih implementacija je Apache Qpid.

Najpoznatiji komercijalni produkti zasnovani na redovima poruka su IBM WebSphere MQ, Oracle Streams, Microsoft Message Queue Server, BEA MessageQ i TIBCO Enterprise Message Service. Od rješenja zasnovanih na otvorenom kodu izdvojili bismo JBoss Messaging, ObjectWeb JORAM, ActiveMQ, RabbitMQ, Open Source Message Queue, Apache Qpid, RabbitMQ i Red Hat Enterprise MRG.

Kada je potrebno ispostaviti sigurnu i pouzdanu asinkronu komunikaciju između heterogenih sustava i aplikacija na nesukladnim platformama, međuoprema zasnovana na redovima

²⁵² Java Message Service Specification Documentation, Oracle, <http://java.sun.com/products/jms/docs.html>

²⁵³ Advanced Message Queuing Protocol, <http://www.amqp.org>

poruka često je jedno od boljih rješenja, jer omogućava neosjetljivost na pogreške u komunikaciji i privremenu nedostupnost mreže.

5.5.2.1 RabbitMQ

RabbitMQ je softver za komunikaciju porukama još poznat kao broker poruka (engl. *message broker*) ili upravitelj redova poruka (engl. *queue manager*)²⁵⁴. Primarno služi za definiranje redova poruka na koje se spajaju druge aplikacije kako bi mogle slati i primiti poruke.

Svaka poruka može uključivati razne informacije o procesu ili zadatku koji bi mogao pokrenuti drugu aplikaciju ili logiku na drugom poslužitelju. Upravitelj redova poruka sprema poruke dok se ne spoji aplikacija koja čita poruka iz reda i nakon toga je ta aplikacija procesira.

Broker poruka ima ulogu posrednika (engl. *middleman*) za razne servise ili web aplikacije. Osnovna arhitektura reda poruka se sastoji od klijentskih aplikacija koje se nazivaju procedurama koje kreiraju poruke i šalju ih brokeru poruka. Ostale aplikacije koje se nazivaju potrošačima (engl. *consumers*) spajaju se na red poruka i pretplaćuju (engl. *subscribe*) na primanje određenih poruka kao što je prikazano na sljedećoj slici:

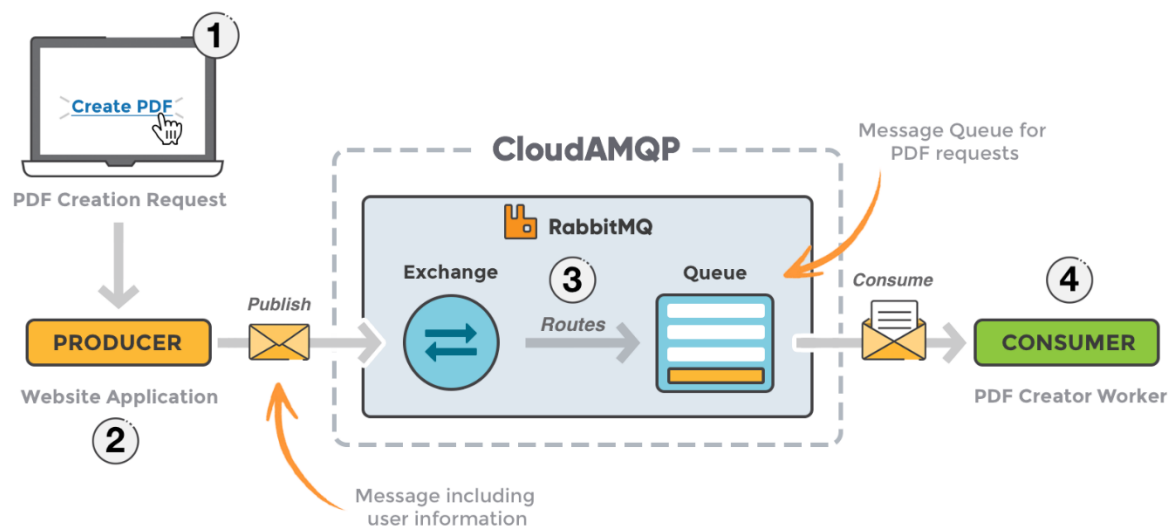


Slika 9. Arhitektura reda poruka

Korištenje redova poruka omogućava web poslužiteljima da vrlo brzo odgovaraju na zahtjeve klijenata, umjesto da obrađuju dugotrajne procedure. Također se vrlo često koristi u slučajevima kad je potrebno distribuirati poruke na veliki broj potrošača ili u slučajevima kad je potrebno balansirati opterećenje među čvorovima za izvršavanje (engl. *workers*).

Potrošač (engl. *consumer*) uzima poruku iz reda poruka i započinje procesiranje (npr. PDF dokument). U isto vrijeme proizvođač (engl. *producer*) stavlja poruke u red. Dvije aplikacije mogu komunicirati putem poruka koje međusobno izmjenjuju, kao što je prikazano na sljedećoj slici:

²⁵⁴ Uvod u RabbitMQ, <https://www.cloudamqp.com/blog/2015-05-18-part1-rabbitmq-for-beginners-what-is-rabbitmq.html>



Slika 10. Scenarij korištenja brokera poruka

Na slici se može primijetiti nekoliko koraka kroz koje se obavlja komunikacija:

1. Korisnik šalje zahtjev za kreiranje PDF-a web aplikaciji.
2. Web aplikacija koja ima ulogu proizvođača, proslijeđuje RabbitMQ-u poruku koja uključuje podatke iz zahtjeva kao što su ime i adresa e-pošte.
3. Aplikacija „Exchange“ prima poruku od proizvođača i preusmjerava (engl. *routes*) je u ispravni red poruka za kreiranje PDF dokumenta.
4. PDF dokument se preuzima od strane potrošača.

6. Poglavlje: Arhitektura orijentirana uslugama (SOA)

U ovom poglavlju naučit ćete:

- ✓ osnovna načela orijentiranosti usluzi
- ✓ poslovanje i orijentiranost usluzi
- ✓ sprega usluga
- ✓ otkrivanje usluga
- ✓ slaganje usluga

Porastom složenosti poslovnih sustava, procesa koji se provode u takvim sustavima te sve većom konkurencijom i porastom troškova, tvrtke traže nova tehnološka rješenja i razvijaju nove usluge kako bi povećale produktivnost i smanjile troškove. Jedno od rješenja koje se predlaže je uvođenje koncepta orijentiranosti usluzi. Sam pojam, arhitektura orijentirana usluzi (engl. *Service-Oriented Architecture* – SOA) predstavlja model u kojemu se logika potrebna za djelovanje složenog sustava rastavlja u manje, zasebne dijelove logike. Te je manje dijelove moguće dalje neovisno raspodijeliti po mreži.

SOA je zapravo nov pogled na arhitekturu softvera koji definira korištenje usluga za ostvarenje zahtjeva koje je postavio korisnik. Premda sam koncept postoji već dulje vrijeme, 1996. su Schulte i Natis definirali SOA-u kao način višeslojnog programiranja koji omogućava dijeljenje logike i podataka među aplikacijama kao i različitim organizacijama²⁵⁵. Jednako tako, SOA omogućuje iskorištavanje logike ili podataka na razne načine, ovisno o tome tko ih koristi. SOA je nastala ponajprije iz potrebe za unapređenjem i racionalizacijom poslovnih sustava, čime će se smanjiti troškovi i povećati dobit tvrtke.

Usluga je osnovni element svakog sustava zasnovanog na načelima SOA-e. SOA sustav zapravo pruža uslugu koja se sastoji od jedne ili više elementarnih usluga, često zvanih i sastavnicama (engl. *components*). Sastavnice su male logičke cjeline koje obavljaju neku operaciju.

SOA-u je najlakše opisati kao standardizirano okruženje načinjeno na načelima orijentiranosti usluzi, u kojemu se mogu izvršavati procesi koji koriste usluge. Trenutno ne postoji jedinstvena definicija što je to SOA. Razlog tome je što ne postoji jedinstveno normizacijsko tijelo koje bi definiralo načela koja treba poštovati želi li se stvoriti sustav orijentiran usluzi. Umjesto toga, razne profesionalne organizacije, konzultantske kuće i proizvođači imaju svoja viđenja i definicije što je to SOA. Naglasimo još da treba razlikovati SOA-u od usluga Weba, o čemu će kasnije biti još riječi.

6.1 Osnovna načela orijentiranosti usluzi

Orijentiranost usluzi izvedena je iz teorije programskog inženjerstva nazvane „**razdvajanje zaduženja**“ (engl. *separation of concerns*)²⁵⁶. Ona govori da veći zadatak treba razdijeliti u više manjih zadataka. Tako se logika potrebna za rješavanje problema razlaže na manje složene i međusobno ovisne dijelove. Svaki takav mali logički dio obavlja jedno zaduženje. Orijentiranost usluzi može se sagledati kao specifična izvedba teorije razdvajanja zaduženja.

1. Iako ne postoji pravno definirana norma, osam se načela pokazalo najprimjerenijima za sustave orijentirane usluzi: usluge trebaju biti nanovo iskoristive – preporučuje se da oblikovanje usluga bude takav da ih je moguće koristiti za buduće potrebe,
2. usluge dijele formalni ugovor – da bi usluge međusobno djelovale potreban je ugovor u kojem je usluga opisana, određen način i uvjeti razmjene podataka

²⁵⁵ Schulte, R. W., Natis, Y. V., „Service Oriented Architectures“, SSA Research Note SPA-401-068, 1996.

²⁵⁶ Dijkstra, Edsger W. (1982), "On the role of scientific thought", in Dijkstra, Edsger W., *Selected writings on Computing: A Personal Perspective*, New York, NY, USA: Springer-Verlag New York, Inc., pp. 60–66, ISBN 0-387-90652-5

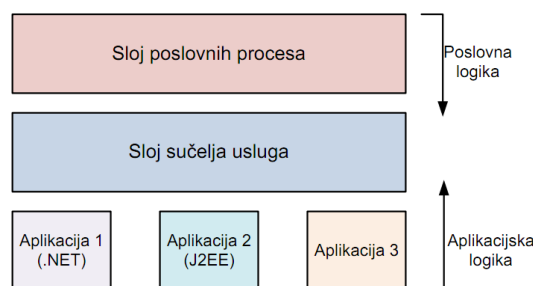
3. usluge su labavo povezane – moraju biti dizajnirane tako da nema potrebe za čvrstim vezama među različitim uslugama,
4. usluge skrivaju temeljnu logiku – samo je dio usluge opisan ugovorom vidljiv vanjskom svijetu,
5. usluge je moguće presložiti – usluga se može sastojati od više usluga, čime je dobivena različita zrnatost (engl. *granularity*) logike i potaknuto ponovno korištenje usluga i stvaranje apstraktnih razina
6. usluge su neovisne – logika sadržana u usluzi mora biti zadržana u jasnim granicama, čime je osigurana neovisnost *prema ostalim* uslugama i zadržana mogućnost promjene i razvoja usluge,
7. usluge ne čuvaju stanje – usluge ne smiju brinuti o informacijama o stanjima jer to može utjecati na njihovu sposobnost labavog povezivanja; ako je ipak potrebno imati mehanizam za upravljanje stanjima, on mora biti izveden na drugom mjestu,
8. usluge je moguće pronaći – opisi usluga moraju se moći pronaći i biti razumljivi bilo kome tko traži uslugu.

Od osam opisanih načela, neovisnost, labava povezivost, apstraktnost i potreba za formalnim ugovorom su temeljna načela koja tvore osnovu za izradu sustava temeljenih na orijentiranosti usluzi.

6.2 Poslovanje i orijentiranost usluzi

Logiku poslovnog sustava možemo podijeliti na poslovnu i aplikacijsku logiku²⁵⁷. Poslovna logika je procesno strukturirana izvedba poslovnih zahtjeva zajedno s ograničenjima, ovisnostima i vanjskim utjecajima na te zahtjeve. Aplikacijska logika automatizira rješenja poslovnih procesa kroz različite tehnologije. Sadržava rješenja potrebna za uspješno svladavanje tokova poslovnih procesa i obično je realizirana kroz aplikacije zasnovane na pogodnim tehnološkim platformama.

SOA uvodi treći sloj između poslovne i aplikacijske logike, sloj sučelja usluga (Slika 6.1). Time je postignuta mogućnost ućahurivanja (engl. *encapsulation*) aplikacijske i poslovne logike. Aplikacije se razdvajaju prema platformama za koje su namijenjene te se otkrivaju pojedine sastavnice, koje usluge sada mogu koristiti putem sučelja.



Slika 6.1: Sloj sučelja usluga

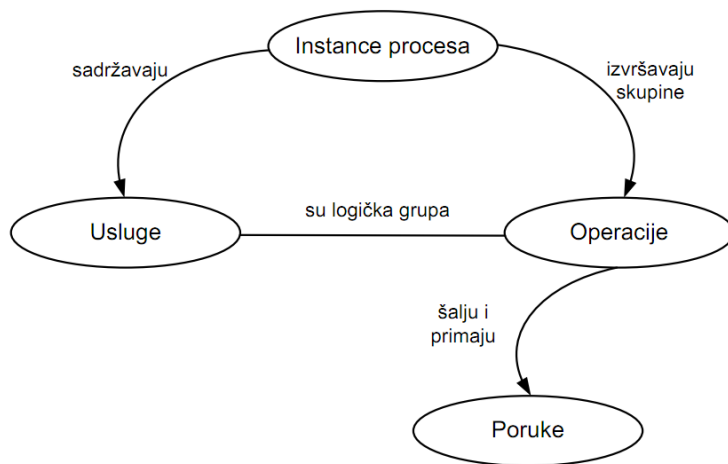
²⁵⁷ Erl, T., Service.Oriented Architecture: Concepts, Technology and Design, Prentice Hall, New Jersey, 2005., ISBN 0-13-185858-0

Već smo rekli da je usluga osnovni element svakog sustava zasnovanog na načelima SOA-e. SOA sustav zapravo pruža uslugu koja se sastoji od jedne ili više elementarnih usluga, često zvanih i sastavnicama. Sastavnice su male logičke cjeline koje obavljaju neki posao. Za njihovo korištenje u sustavima zasnovanim na SOA-i sastavnice moraju osigurati određen stupanj skrivanja poslovne logike, tako da možemo reći da:

- poruka označava podatak potreban da se obavi posao ili neki njegov dio,
- operacija predstavlja logiku potrebnu za procesiranje poruke da bi se obavila jedinica posla,
- usluga je logički skup operacija sposobnih izvršiti povezane jedinice posla,
- proces sadržava poslovna pravila koja određuju koje se usluge koriste da bi se obavio neki posao.

Sastavnice u SOA-i međusobno se odnose na sljedeći način:

- operacija šalje i prima poruke kako bi obavila neki posao, najčešće je definirana porukama koje obrađuje,
- usluga je skup srodnih operacija, najčešće je definirana operacijama koje sadržava,
- instanca procesa može sačinjavati uslugu, ali nije nužno definirana uslugom budući da može zahtijevati samo dio funkcionalnosti koje usluga nudi.



Slika 6.2: Međusobni utjecaj SOA sastavnica

Arhitektura orijentirana usluzi je standardizirano okružje načinjeno na načelima orijentiranosti usluzi, u kojemu se procesi koji koriste usluge mogu izvršavati.

6.2.1 Ugovori

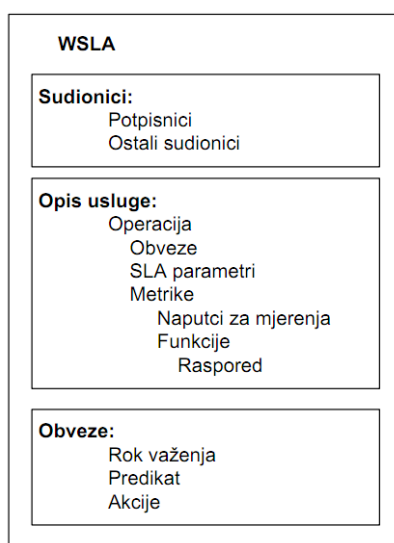
Prije nego što se usluga može početi koristiti, uobičajena je praksa da se potpiše formalni ugovor koji određuje prava i obveze obje strane – pružatelja i korisnika usluge. U sustavima zasnovanim na načelima SOA-e svaka usluga ima definirani vlastiti ugovor, koji sadržava metapodatke koji je opisuju. Ugovor sadržava formalnu definiciju sučelja usluge prema vanjskom svijetu, popis (tj. definiciju) operacija koje usluga izvršava, ulazne i izlazne poruke za svaku operaciju te pravila i obilježja usluge i operacija koje izvodi. Ovakvi se ugovori često formalno nazivaju SLA – *Service Level Agreement*.

Za stvaranje ugovora koristi se neki od jezika zasnovanih na XML-u. Najčešće je to WSDL (engl. *Web Services Description Language* – više o WSDL-u bit će govora u sljedećem poglavlju) kojim se daje apstraktni i konkretni opis usluge, a mogu se koristiti i XSD (engl. *XML Schema Definition*) shema kao i polica (engl. *policy*)²⁵⁸.

Svi oni sadržavaju metapodatke o usluzi i kolektivno čine ugovor – skup pravila koje potencijalni korisnik usluge mora prihvatiti i udovoljiti im. Na ovim je načelima razvijeno nekoliko prijedloga normi za izradu ugovora u SOA sustavima^{259, 260}, međutim niti jedan za sada nije prevladavajući.

Web Service Level Agreement (WSLA) je XML specifikacija uvjeta koji moraju biti ispunjeni da bi se neka usluga mogla uspješno pružiti. Tipični WSLA dokument (Slika 6.3) sastoji se od popisa sudionika, definicije usluge te obveza sudionika u ugovoru. Uz potpisnike SLA ugovora, u WSLA dokumentu moguće je navesti i ostale sudionike koji, iako nisu formalnim dijelom ugovora, podupiru uslugu na neki način, poput pružatelja telekomunikacijskih usluga ili sl.

U dijelu koji opisuje uslugu nalazi se opis obilježja usluge razvrstan po operacijama koje se izvode. Za svaku se operaciju definira jedna ili više obveza te SLA parametri s odgovarajućim metrikama. Metrike se mogu definirati kao naputci za mjerenje i funkcije koje predstavljaju algoritme koji specificiraju kako se određena metrika računa. Rasporedom se određuje vremenski okvir u kojem se funkcije moraju izvršiti. Zadnji dio svakog WSLA dokumenta čine obveze svih stranaka koje potpisuju WSLA dokument, koje uključuju rok važenja te akcije koje su stranke suglasne izvesti u slučaju raskida ugovora.



Slika 6.3. WSLA dokument

²⁵⁸ Keller, A., Ludwig, H., The WSLA Framework: Specifying and Monitoring Service Level Agreements for Web Services, Journal of Network and System Management, Volume 11, Number 1, Springer-Verlag, March 2003.

²⁵⁹ Dan, A., Davis, D., Kearney, R., King, R., Keller, A., Kuebler, D., Ludwig, H., Polan, M., Spreitzer, M. and Youssef, A., Web Services on demand: WSLA-driven Automated Management, IBM Systems Journal, Special Issue on Utility Computing, Volume 43, Number 1, pages 136-158, IBM Corporation, March, 2004.

²⁶⁰ Gambiagi, P., Owe, O., Schneider, G., Ravn, A. P., Language-based support for service oriented architectures: Future directions, 1st International Conference on Software and Data Technologies (ICSOT 2006), pages: 339-344, 11-14 Sep 2006, Setúbal, Portugal

Pogledamo li primjer isječka jednog WSLA dokumenta, koji je dan u nastavku, vidimo da je zapravo riječ o dobro strukturiranom XML dokumentu. Pažljivim čitanjem samog koda vrlo je lako zaključiti na što se pojedini dio ugovora odnosi.

Ovaj se ugovor odnosi na uslugu izlistavanja dionica, čija je funkcionalnost opisana u pripadajućem opisu usluge u WSDL datoteci. Prikazan je odsječak u kojem su definirane obveze pružatelja usluge. One su podijeljene u nekoliko razina (*ServiceLevelObjective*), definirano je razdoblje kada obveza vrijedi (u oznakama *<StartDate>* i *<EndDate>*), kao i što se događa ako se neka od obaveza prekrši (parametar *Action*).

```
<?xml version="1.0"?>
<SLA xmlns="http://www.ibm.com/wsla"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.ibm.com/wsla
c:\Projects\WSLA\wsla.xsd"
name="StockquoteServiceAgreement12345" >
. . .
<!--ovdje je ispušten dio koda-->
  <Obligations>

    <ServiceLevelObjective name="g1" serviceObject="WSDLSOAPGetQuote">
      <Obligated>
        provider
      </Obligated>
      <Validity>
        <StartDate>2001-08-15:1400</StartDate>
        <EndDate>2001-09-15:1400</EndDate>
      </Validity>
      <Expression>
        <Predicate xsi:type="wsla:Less">
          <SLAParameter>
            AverageResponseTime
          </SLAParameter>
          <Value>
            5
          </Value>
        </Predicate>
      </Expression>
      <EvaluationEvent>
        NewValue
      </EvaluationEvent>
    </ServiceLevelObjective>

    <ActionGuarantee name="g2">
      <Obligated>
        ms
      </Obligated>
      <Expression>
        <Predicate xsi:type="wsla:Violation">
          <ServiceLevelObjective>
            g1
          </ServiceLevelObjective>
        </Predicate>
      </Expression>
      <EvaluationEvent>
        NewValue
      </EvaluationEvent>
```

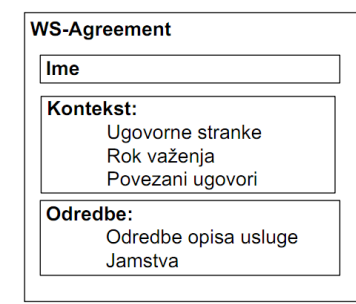
```

    <QualifiedAction><Party>
      customer
    </Party>
    <Action actionName="notification" xsi:type="Notification">
      <NotificationType>
        Violation
      </NotificationType>
      <CausingGuarantee>
        g1
      </CausingGuarantee>
      <SLAParameter> AverageResponseTime
      </SLAParameter>
    </Action>
  </QualifiedAction>
  <ExecutionModality>
    Always
  </ExecutionModality>
</ActionGuarantee>
<!--ovdje je ispušten dio
koda-->
</SLA>

```

WS-Agreement²⁶¹ koncept je ugovora razvijen na temelju WSLA i predložen od strane Global Grid Foruma (GGF). Razvijen je s ciljem da se pruži standardni sloj koji bi pomogao razvoj SOA sustava zasnovanih na dogovorima. U WS-Agreementu se definiraju elementi neovisni o platformi na kojoj se izvode, pomoću kojih se na jednostavan način provodi ugovaranje²⁶². Oni opisuju koncepte i uvjete generičkog dogovora kao elemente najviše razine²⁶³. Ako je potrebno, generičke se definicije mogu proširiti platformsko ili domenski specifičnim elementima.

Slično kao i WSLA, i WS-Agreement je XML dokument koji sadržava opis onoga što je dogovoreno, kontekst na koji se ugovor odnosi i definiciju predmeta ugovora (Slika 6.4).



Slika 6.4: WS-Agreement dokument

²⁶¹ Global Grid Forum. Web Services Agreement Specification (*WS-Agreement*), version 2005/09 edition, September 2005

²⁶² Bertolino, A., De Angelis, G., Polini, A., Automatic Generation of Test-beds for Pre-Deployment QoS Evaluation of Web Services, **Proceedings of the 6th international workshop on Software and performance**, pages: **137-140**, Buenos Aires, Argentina.

²⁶³ Ludwig, H., WS-Agreement Concepts and Use –Agreement-Based Service-Oriented Architectures. Technical report, IBM, May 2006.

Element Kontekst (engl. *Context*) koristi se za opis ugovornih strana u sporazumu kao i ostalog sadržaja dogovora koji ne predstavlja obveze stranaka, kao što je to primjerice rok do kojeg dogovor vrijedi. Dogovor može sadržavati jedan ili više konteksta.

Odredbe (engl. *Terms*) kojima se definiraju prava i obveze stranaka čine jezgru svakog dogovora načinjenog prema WS-Agreement specifikaciji. Za njihovo stvaranje moguće je koristiti i specijalne povezne elemente (kao npr. AND/OR/XOR operatore), čime se omogućava specificiranje alternativnih puteva, kao i ugnježđavanje elemenata dogovora.

Obveze stranaka u dogovoru organizirane su u dvama logičkim cjelinama. Prva određuje koje su usluge predmet dogovora, a druga mjerljiva jamstva za pruženu uslugu. Odredbe opisa usluge opisuju uslugu i način na koji će ona biti pružena s funkcionalnog gledišta.

Ako je usluga specifično vezana za neku domenu, tada i njen opis mora sadržavati uvjete koji moraju biti ispunjeni kako bi se usluga mogla pružiti. Poseban oblik Odredbe opisa usluge je Referenca usluge, kojom se definira pokazivač na opisu usluge, čime se izbjegava njeno eksplicitno definiranje u dogovoru.

Zadnji se dio dogovora odnosi na odredbe kojima se definiraju mjerljiva jamstva, koja tijekom procesa korištenja usluge mogu biti poštovana ili prekršena. Jamstvo se sastoji od ugovornih strana (npr. pružatelj usluge, korisnik usluge), liste usluga na koje se to jamstvo odnosi, Booleov izraz kojim se određuje pod kojim uvjetima jamstvo vrijedi, stvarne tvrdnje za koju se može jamčiti (Service Level Objective) te skupa poslovnih vrijednosti za opisanu uslugu (primjerice: važnost, kazne, performanse). Opis jamstvenih obveza najčešće se iskazuje u jeziku specifičnom upravo za domenu na koju se odnosi.

Sljedeći primjer prikazuje XML kod ugovora zasnovan na specifikaciji WS-Agreement.

```
<wsag:Agreement
  Name="job123"
  xmlns:wsag="...">

  <wsag:AgreementContext>
    <wsag:AgreementInitiator>
      xs:anyURI
    </wsag:AgreementInitiator>
    <wsag:AgreementProvider>
      xs:anyURI
    </wsag:AgreementProvider>
    <wsag:TerminationTime>
      xs:DateTime
    </wsag:TerminationTime>
    <wsag:ServiceReference>
      <wsa:EndpointReference xmlns:wsa="...">
        <wsa:Address>
          xs:anyURI
        </wsa:Address>
        <wsa:PortType>
          xs:QName
        </wsa:PortType>
      </wsa:EndpointReference>
    </wsag:ServiceReference>
  </wsag:AgreementContext>
</wsag:Agreement>
```

```

        <wsag:RelatedAgreements>
        . . .
        </wsag:RelatedAgreements>
    </wsag:AgreementContext>
    <wsag:TerminationTerms>
    . . .
    </wsag:TerminationTerms>
    <wsag:MonitoringTerms>
    . . .
    </wsag:MonitoringTerms>
    <wsag:GuaranteeTerms>
    . . .
    </wsag:GuaranteeTerms>
    <tns:domain specific terms>
    . . .
    </tns:domain specific terms>

</wsag:Agreement>

```

Na početku unutar *oznaka (engl. tag)* AgreementContext definiran je kontekst ugovora, oznake TerminationTems definiraju uvjete pod kojima se ugovor može raskinuti. Oznakama MonitoringTerms definiraju se uvjeti za praćenje provedbe ovog ugovora, dok se unutar oznaka GuaranteeTerms preciziraju jamstva. Na kraju se nalazi popis domenski specifičnih zahtjeva.

Kao što vidimo, u ugovorima se definira velik skup različitih obveza i funkcionalnosti koje usluge, njihovi proizvođači i korisnici moraju poštovati u međusobnom odnosu. Zbog takve širine, često na njihovom definiranju radi cijela skupina stručnjaka, pored onih računalne struke (koji opisuju same funkcionalnosti) vrlo su bitni i oni ekonomskog i pravnog usmjerenja, koji definiraju prava, obveze stranaka u sporazumu i naplatu korištenja usluge. Iz toga možemo iščitati da je većina ovako sklopljenih ugovora zapravo namijenjena surađivanju različitih poslovnih subjekata na korištenju različitih usluga.

Kao i za većinu specifikacija, i za ovdje spomenute specifikacije normi za izradu ugovora postoje neka već gotova rješenja zasnovana na otvorenom kodu. Tako za WSLA postoji već gotovi besplatni *plugin*²⁶⁴, koji se može uključiti u platformu Eclipse te se može koristiti prilikom stvaranja ugovora. Za WS-Agreement razvijen je primjerice programski okvir WSAG4J²⁶⁵, koji se može integrirati u bilo koji sadržnik Weba (primjerice Apache Tomcat) te tako oslobađa programera od implementacije cijelog sloja podrške za usluge Weba koje zahtijeva specifikacija WS-Agreement.

6.3 Sprega

Pod pojmom sprega (engl. *coupling*) podrazumijevamo u kakvom se međusobnom odnosu nalaze dvije stvari. U našem ćemo slučaju govoriti o međusobnom odnosu dviju (ili više) usluga u sustavima s arhitekturom orijentiranom uslugama. S gledišta programskog inženjerstva,

²⁶⁴ http://www.eclipseplugincentral.com/Web_Links-index-reg-viewlink-cid-1235.html

²⁶⁵ <http://packcs-e0.scai.fraunhofer.de/wsag4j/index.html>

svaki odnos dijelova programskog koda može se izraziti stupnjem njihove međusobne ovisnosti jedan o drugome.

Kako se u SOA sustavima uobičajeno koristi nešto širi pojam sprege među uslugama, koji podrazumijeva i način povezivanja usluga, postavlja se pitanje kakve vrste sprege postoje među uslugama. Uobičajeno se definiraju dvije vrste:

- čvrsto povezane usluge (engl. *tight coupled*), kada je za njihovo međusobno djelovanje potrebno poznavati internu izvedbenu strukturu (primjerice, jedna usluga dohvaća podatak koji je lokalno zapisan na drugoj usluzi),
- labavo povezane usluge (engl. *loose coupled*), gdje usluge zapravo ne ovise jedna o drugoj, već se sva međusobna ovisnost rješava putem komunikacije, uobičajeno putem poruka (ovakav se način povezivanja koristi još od povezivanja terminala sa *mainframe* sustavima, a sličan se oblik koristi i danas kod tradicionalnih klijent-poslužitelj aplikacija).

Vidljivo je da kod čvrsto povezanih usluga svaka promjena u nekom ponašanju (a to može biti dodavanje novih ili promjena postojećih funkcionalnosti) jedne usluge može rezultirati nepredvidljivim radom, pa čak i pogreškom u radu druge usluge. Što je veća međusobna ovisnost usluga, to je teže takve usluge ponovno iskoristiti, pogotovo ako na njihovom razvoju radi druga programerska skupina (možda čak i iz neke druge tvrtke!).

Nasuprot tome, ako je sustav projektiran tako da se koriste slabo povezane usluge, tada promjena jedne usluge zapravo nema nikakvog utjecaja na drugu uslugu koja je koristi, sve dok se ne mijenja format u kojem se poruke razmjenjuju (to se tiče sučelja usluga, o čemu ćemo više govoriti nešto kasnije).

Kako je pojašnjeno u prethodnom odjeljku, važan dio procesa korištenja usluge je i ugovor o njenom korištenju. Kako je i sprega važna u pravilno projektiranim sustavima zasnovanim na uslugama, treba promotriti odnos sprege i samih ugovora. Pritom je nužno raščlaniti pojam usluga na pojedine izvedbene sastavnice (programska logika, tehnološka izvedba, funkcionalnosti, implementacija) te uvesti i korisnika usluge (koji je ujedno i potpisnik samog ugovora).

Programska logika se prema ugovoru može staviti u čvrsti oblik sprege, no obrnuto se ne preporučuje jer bi to podrazumijevalo da se ugovor sastavljao prije programske izvedbe same usluge, što vrlo često dovodi do problema kod definiranja prava i obveza ugovornih strana.

U idealnom slučaju, ugovor o korištenju usluge i tehnologija u kojoj je usluga programski izvedena ne bi smjeli biti ni u kakvom odnosu (bez sprege – engl. *decoupled*) jer jedino tako osiguravamo podršku za sve otvorene tehnologije i standarde koji se i inače koriste u izvedbi primjerice usluga Web-a.

Odnos ugovora prema funkcionalnostima usluge također ni bi smio biti spregnut, čime se osigurava mogućnost programeru da ponovno koristi uslugu za neku drugu svrhu. Treba istaknuti da određeni tipovi usluga mogu zahtijevati čvršću vezu prema funkcionalnosti usluge.

Sprega ugovora s implementacijom usluge nije poželjna, osobito ako uzmemo u obzir da usluge tijekom svog izvršavanja mogu koristiti i neke vanjske i dijeljene resurse. Ovdje možemo pogledati i odnos korisnika prema implementacijskim potankostima, koji također nije

poželjan jer je vrlo teško odrediti kakvu će opremu (hardver i/ili softver) korisnik neke usluge imati na raspolaganju u trenutku korištenja usluge.

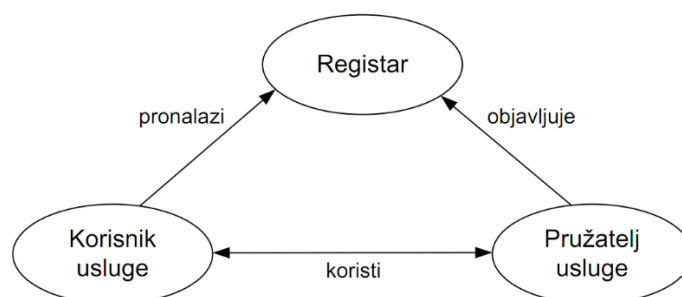
Odnos korisnika i ugovora predstavlja pozitivan oblik sprege, no uspjeh toga tipa sprege izravno ovisi o tome kako je uspjelo izbjegavanje prije opisanih neželjenih oblika sprege. U ovom je odnosu zanimljivo definirati i zrnatost (engl. *granularity*), koja može utjecati na mogućnosti same usluge, njene podatke ili ograničenja u njezinu korištenju. Ako ugovor omogućava korištenje više odvojenih funkcionalnosti, koja svaka za sebe troši određene resurse i izvodi neki posao, korisnik usluge će možda smatrati da izvođenje same usluge traje predugo (doživjet će to kao produljeno procesiranje). S druge pak strane, ako je usluga finije granulirana, korisnik usluge može biti nezadovoljan zbog toga što će morati više puta izvršavati određenu funkcionalnost kako bi tek na kraju bio zadovoljan rezultatom koji je dobio od usluge.

6.4 Otkrivanje usluga

Vrijeme je da se pozabavimo još jednim iznimno važnim područjem vezanim uz sustave zasnovane na uslugama, a to je kako otkriti neku uslugu.

Pretpostavimo da radite u nekoj maloj tvrtki i da ste upravo isprogramirali (prateći naravno sve norme i pravila struke!) uslugu Weba za koju pretpostavljate da će biti zanimljiva velikom broju potencijalnih klijenata. No kako ćete im dati do znanja da upravo vaša tvrtka nudi tu uslugu? Naravno, možete se reklamirati po poznatim portalima i ostalim sjedištima Weba, dati oglase na razne društvene mreže i slično, gdje ćete uputiti korisnike na vašu stranicu Weba. Ako je riječ o tzv. *killer*-aplikaciji²⁶⁶, riskirate potpun kolaps informatičkih resursa vaše tvrtke kada nebrojeni korisnici pristupali na poslužitelje u želji da baš oni isprobaju vašu uslugu. No hoće li to biti tako ili će vašu aplikaciju zamijetiti samo vrlo mali broj budućih korisnika?

Upravo kako bi se izbjegao ovakav, prilično nepouzdan način oglašavanja usluge, u sustavima zasnovanim na uslugama primjenjuje se načelo jedinstvenog (ovo jedinstveno shvatite sa zadržkom, što ćemo kasnije komentirati) direktorija ili registra, u kojem bi se nalazili opisi svih objavljenih usluga kako bi ih se lagano moglo pronaći. Slika 6.5 shematski prikazuje odnos između korisnika usluge, njenog tvorca (koristit ćemo uobičajeni naziv pružatelj usluge) i registra u kojem se objavljuju usluge.



Slika 6.5: Otkrivanje usluga/pružatelj

²⁶⁶ Dasgupta, P., Killer Applications (overview), Arizona State University in Tempe, AZ, May 2002.

Suradnja ovih triju entiteta odvija se na sljedeći način:

1. Pružatelj usluge razvio je novu uslugu, koju je opisao u WSDL formatu.
2. Pružatelj usluge objavljuje opis svoje nove usluge u registru.
3. U registru se nalazi baza s podacima o svim uslugama koje su njihovi pružatelji registrirali.
4. Kada korisnik poželi koristiti neku uslugu, kontaktira registar te ga pretražuje u potrazi za opisom usluge koja odgovara onome što traži.
5. Kada se u registru pronađe odgovarajuća usluga, tada se njen opis, zajedno s ostalim parametrima nužnim za njezino korištenje (treba napomenuti da je jedan od najbitnijih takvih parametara URI gdje se usluga nalazi, jer bez njega korisnik neće znati gdje treba potražiti uslugu), vraća Korisniku.
6. Korisnik može početi koristiti uslugu koju je pronašao u registru (prije čega, ovisno o tipu usluge, mora potpisati ugovor o njezinom korištenju, ako tako zahtijeva pružatelj usluge).

Proces otkrivanja usluge jednako koristi i krajnjim korisnicima neke usluge u sustavima orijentiranim u usluzi, kao i programerima u tvrtkama koji žele iskoristiti već postojeće usluge u svojim rješenjima (napomenimo da će oboje, sa stajališta usluge koju koriste i onoga tko je tu uslugu stvorio, biti smatrani korisnicima takve usluge).

Razmislimo na trenutak koje prednosti mehanizam otkrivanja usluga donosi programeru u našoj već spominjanoj maloj tvrtci. Prije nego što pokrene proces otkrivanja usluge, programer ne zna postoji li neka usluga koja udovoljava traženim funkcionalnostima. Ako pretraga za traženom uslugom polučiti traženi rezultat (i usluga se pronađe), moći će zatražiti njeno korištenje (naravno, prvo mora pristati na uvjete iz ugovora i platiti traženu cijenu), uštedeći tako na razvoju dijela aplikacije koji zapravo već postoji. S druge pak strane, ako pretraga za uslugom pokaže da takva funkcionalnost još nije razvijena, moguće je preciznije planirati trošak njenog razvoja i možda čak i dodatne zarade jer se takva usluga može nuditi i drugima na tržištu usluga.

Proces otkrivanja usluge omogućavaju informacije koje se nalaze u opisu same usluge, eventualno postojećem ugovoru o njenom korištenju te ostalim dostupnim metapodacima o usluzi. Većina postupaka i tehničkih rješenja za otkrivanje usluge ipak ostaje na čovjeku kao zadnjem donositelju odluke radi li pronađena usluga upravo ono što se od nje traži.

Uzmimo za primjer tri usluge i označimo ih kao Usluga1, Usluga2 i Usluga3. Pretpostavimo da je svaka usluga opisana s dovoljnim skupom podataka o uslugama. Možemo reći da o svakoj usluzi znamo koje funkcionalnosti može izvršiti.

Programer pokreće proces otkrivanja usluga. On raspolaže vlastitim opisom koje funkcionalnosti treba imati usluga. Za otkrivanje može koristiti automatizirani alat (bilo bi vrlo teško pretraživati desetke tisuća različitih usluga ručno, jednu po jednu), a rezultat je da Usluga1 ne ispunjava njegove zahtjeve, dok Usluga2 i Usluga3 prolaze tu prvu provjeru te mogu biti uzete na daljnje razmatranje. U ovom trenutku treba potanje provjeriti stvarne mogućnosti promatranih usluga, uzevši u obzir i eventualna ograničenja u njihovoj izvedbi i korištenju (poput cijene ili načina korištenja). Daljnjim pregledom dostupnih metapodataka o uslugama programer zaključuje da Usluga2 ne ispunjava u potpunosti sve postavljene zahtjeve, dok Usluga3 može u potpunosti ispuniti sve zahtjeve koje je programer postavio u

svom zahtjevu. Stoga je Usluga3 konačni odabir programera nakon provedenog procesa otkrivanja usluge.

Iz opisanog primjera vidimo da je cijeli proces otkrivanja usluge moguće automatizirati, pa se tako posljednjih godina mnogo istraživalo upravo na tom području. Mnoge su velike softverske kompanije pokušale stvoriti vlastiti registar usluga i na njemu zasnovan vlastiti pretraživač objavljenih usluga, no svi ti pokušaji nisu rezultirali šire prihvaćenim rješenjem.

Kako vidimo, za što pouzdaniji opis otkrivanja usluge ključni su dobar opis funkcionalnosti koje nudi, njen ugovor, metapodaci kojima su pobliže opisana njena ograničenja i načini na koji je korisnici mogu koristiti. Kako je otkrivanje nužno da bi se sama usluga mogla i koristiti, to zapravo osigurava da će se pružatelj usluge potruditi što bolje opisati točno što i na koji način želi ponuditi svojim potencijalnim korisnicima. S tim u vezi, a kako bi se izbjegao potencijalni rizik od pogrešne interpretacije opisa usluge, treba osigurati suradnju tehničkog i pravnog osoblja pružatelja usluge na cijelom procesu izrade pravnih dokumenata o kojima je bilo riječi u ovom poglavlju.

Već smo spomenuli potrebu postojanja jedinstvenog registra gdje bi korisnici usluga Weba s lakoćom mogli pronaći usluge koje ih zanimaju te među mnoštvom istovrsnih usluga odabrati one koje će kasnije koristiti, prema vlastito definiranom kriteriju (pritom vodeći brigu kako o cijeni korištenja usluge tako i o kvaliteti njezina pružanja od strane davatelja usluge). No, nažalost, takav registar već neko vrijeme ne postoji.

Kako smo pokazali, ovaj registar vrlo je važan dio cijelog procesa na kojem se temelje usluge Weba. Stoga su brojne kompanije, uključujući Microsoft, IBM i SAP, u vremenu kada su brojni programeri širom svijeta prihvaćali mnogobrojne pogodnosti koje je nudila nova paradigma usluga Weba, uspostavljale vlastite javne registre na kojima su nudile objavu informacija o uslugama Weba²⁶⁷. Tako su programerima širom zajednice omogućile da, nakon što razviju vlastitu uslugu Weba, objave na njihovom registru njezin opis (bilo je moguće dodati i neki oblik dokumentacije o usluzi Weba). Takve bi usluge ostali programeri mogli pretraživati i uključivati u svoje aplikacije, stvarajući tako vlastita rješenja temeljena na javno dostupnim uslugama Weba.

Nažalost, sve su kompanije odustale od ovakvih projekata; jedino još tvrtka SAP nudi, u okviru jednoga aplikacijskog poslužitelja, rješenje koje obuhvaća registar za usluge Weba (SAP Netweaver²⁶⁸).

Razlog ovakvom raspletu jednog dobro smišljenog sustava je što su takvi javni registri usluga Weba počeli nalikovati kokošinjcu, a ne dobro uređenom popisu (točnije rečeno bazi) usluga Weba. Kako su se pojavile razne specifikacije i alati koji su omogućavali jednostavno dodavanje (i pregledavanje) javno dostupnih registara, programeri širom svijeta su, u fazi testiranja i učenja, objavljivali te usluge tako da se za vrlo kratko vrijeme u registru pojavio velik broj usluga nazvanih test2, koje su kao ime vlasnika prikazale „stvaran“ podatak poput

²⁶⁷ Hewitt, E., Java SOA Cookbook, O'Reilly Media, Inc., 2009.

²⁶⁸ <http://www.sap.com/platform/netweaver/index.epx>

someone@hotmail.com. Naravno, bilo je sve teže pronaći neku korisnu uslugu pa je zanimanje korisnika počelo nestajati.

U međuvremenu, brojne su se tvrtke okrenule jednostavnijim, vlastitim rješenjima za registar vlastitih usluga Weba koje su mogle objaviti i koristiti za potrebe vlastitih razvojnih skupina ili poslovnih partnera. Ovakva su rješenja nudila i povećanu kvalitetu (jer se znalo tko je programer neke usluge koja će se koristiti u ključnom tvrtkinom softveru), kao i višu razinu sigurnosti (registar nije bio javno dostupan i često je bio zaštićen i vatrozidom). Ovakav je način objave opisa usluga Weba i danas dominantan.

U novije se vrijeme pojavljuju tendencije da velike kompanije, poput Googlea, Amazona, Facebooka i sličnih, nude razne usluge Weba kao dijelova vlastitih programskih platformi. Na taj se način pruža određena razina uvida u sam poslovni model te kompanije, no povećavanje broja korisnika usluge donosi i povećanu zaradu.

U nastavku donosimo kratak prikaz dvaju mehanizama za otkrivanje usluga.

6.4.1 Universal Description Discovery and Integration (UDDI)

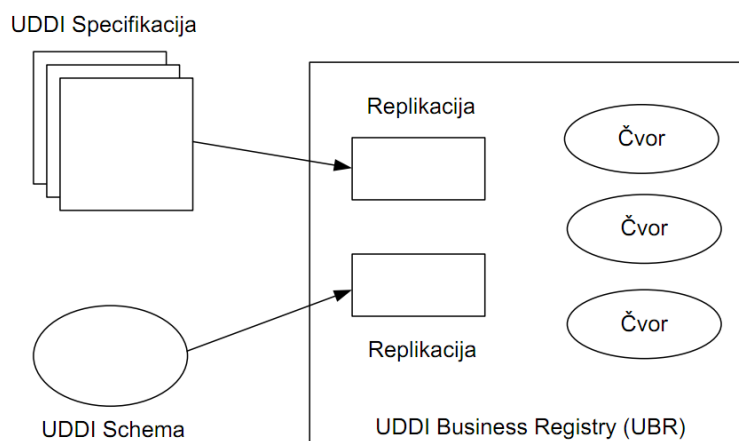
Universal Description Discovery and Integration (UDDI)²⁶⁹ bio je prvi standardiziran registar usluga Weba, čija je prva verzija objavljena 2000. godine. Ukupno su objavljene 3 verzije, a posljednja (u inačici 3.0.2) postala je dio specifikacije WS-Interoperability (više o WS-* standardima reći ćemo u sljedećem poglavlju).

UDDI je specifikacija raspodijeljenog registra usluga Weba. Zasnovan je na jeziku XML te neovisan o platformi na kojoj se izvodi. U registru se objavljuju metapodaci o uslugama koje pružatelji žele ponuditi na tržištu.

UDDI registar moguće je pretraživati, a pritom se koriste SOAP, CORBA ili Java RMI. Zainteresirani čitatelj može u samoj specifikaciji pogledati koja se XML Schema koristi i provjeriti specifikaciju API-ja.

Bilo je zamišljeno da nekoliko globalnih tvrtki uspostavi sustav nazvan UDDI Business Registry – UBR. UBR osigurava implementaciju UDDI standarda i sinkronizaciju svih podataka na automatski način.

²⁶⁹ <http://uddi.org/>



Slika 6.6: UDDI Arhitektura

UBR je također bio poznat kao „Javni oblak“ (engl. *Public Cloud*), i njegovi ga korisnici vide kao jedan sustav sagrađen od više čvorova koji rade s istim podacima čiji se integritet osigurava replikacijom. Ovaj je oblak trebao biti fizički raspodijeljen, pa se replikacijom osiguravalo da svi čvorovi unutar određenog roka raspolažu istim podacima. Taj je rok obično bio 24 sata. UDDI specifikacija dopušta i uspostavu vlastitih privatnih UDDI registara, koje su često koristile velike kompanije za rad s vlastitim uslugama Web-a. No kako takvi registri nisu bili sinkronizirani s javnim registrima, nije bilo moguće pronaći usluge objavljene u njima, te nisu bili smatrani dijelom javnog UDDI oblaka.

Većina je velikih softverskih kompanija (uključujući Microsoft, IBM, HP itd.) implementirala vlastiti UBR, no s vremenom svi su ga ili isključili ili integrirali u neki vlastiti komercijalni proizvod, otprilike u vrijeme kada je objavljena UDDI specifikacija 3.0 2005. godine. Razlog isključenju je što skoro nitko zapravo nije koristio usluge UBR-a.

Pretraživanje UDDI registra radilo se putem ugrađenih metoda u UDDI API-ju. Jedna od njih `find_business` kao parametar prima neku oznaku tvrtke koju tražimo (to može biti ime, adresa, kategorija u kojoj tvrtka djeluje itd.).

Ako u UDDI registru tražimo sve registrirane tvrtke čije ime počinje s Visoka, upotrijebit ćemo sljedeći programski odsječak:

```
<find_business generic="2.0" xmlns="urn:uddi-org:api_v2">
  <name>
    Visoka
  </name>
</find_business>
```

Ovakva će pretraga na kraj teksta koji pretražujemo dodati *, te će kao rezultat vratiti sve tvrtke čije ime počinje traženim riječima. Prema UDDI specifikaciji, pretraga `find_business` vratit će svoj rezultat u obliku liste `businessList`, koja će sadržavati po jedan `businessInfo` element za svaku tvrtku koja udovoljava uvjetu iz pretrage koju smo izvršili. Dobit ćemo primjerice ovakav ispis:

```
<businessList generic="2.0"
  operator="uddi.sourceOperator"
  truncated="true">
```

```

    xmlns="urn:uddi-org:api_v2">
<businessInfos>
  <businessInfo businessKey="F94D3591C691">
    <name>
      Visola škola za primjenjeno računarstvo
    </name>
    <serviceInfos>
      <serviceInfo serviceKey="1234567890">
        <name>
          Referada
        </name>
      </serviceInfo>
    </serviceInfos>
  </businessInfo>
  <businessInfo>
    ...
  </businessInfo>
</businessInfos>
</businessList>

```

Među dobivenim rezultatima valja izdvojiti `businessKey`, prema kojemu se mogu dobiti daljnje informacije o pronađenoj tvrtki kao i uslugama koje je registrirala na UDDI registru. Dalje možemo pretraživati registar uspoređujući sučelja usluga koje tražimo, te dobivamo liste ključeva usluga koje odgovaraju traženim parametrima.

Kao konačni rezultat pretrage za uslugom dobivamo sljedeću listu usluga:

```

<serviceList generic="2.0"
  operator="uddi.sourceOperator"
  xmlns="urn:uddi-org:api_v2">
  <serviceInfos>
    <serviceInfo
      serviceKey="064B4170-D5F5-11DA-8170-A74C17FA61A7"
      businessKey="1A3DB880-D5F4-11DA-B880-F94D3591C691">
        <name>
          Referada
        </name>
      </serviceInfo>
    </serviceInfos>
  </serviceList>

```

Pristup UDDI registru moguće je ostvariti i programski. Tijekom godina razvijeno je mnogo inačica besplatnih UDDI registara i na njima zasnovanih programskih sučelja. Primjerice, poslužitelj aplikacija Apache Geronimo²⁷⁰ u sebi već ima jUDDI²⁷¹, UDDI registar zasnovan na programskom jeziku Java.

²⁷⁰ <http://geronimo.apache.org/>

²⁷¹ <http://ws.apache.org/juddi/>

6.4.2 Web Services Dynamic Discovery (WS-Discovery)

Usporedno sa sve većom popularnošću korištenja usluga Weba, razvijale su se i norme koji su propisivali način dizajniranja takvih usluga. Tako je nastao skup WS-* normi (više o njima naći ćete u sljedećem poglavlju), a ovdje ćemo spomenuti Web Services Dynamic Discovery (WS-Discovery) normu²⁷², koja se odnosi na otkrivanje usluga.

Ova tehnička specifikacija (koja je podržala organizacija OASIS) definira višeodredišni (engl. *multicast*) protokol za otkrivanje usluga na uređajima (ovdje se namjerno koristi riječ uređaj, koja obuhvaća širi skup od pojma računalo) koje korisnik koji traži neku uslugu vidi na mreži. Uslugu je moguće otkriti prema tipu tražene usluge ili prema njenom imenu. Za proces otkrivanja usluga koristi se protokol SOAP²⁷³.

Specifikacija definira dva načina u kojima se pomoću WS-Discovery protokola mogu otkriti usluge, tzv. *ad-hoc* način i upravljani (engl. *managed*) način.

U *ad-hoc* načinu otkrivanja usluge, ako se pretraga vrši prema tipu usluge (očito je kako se pretpostavlja da će ovakav način pretrage rezultirati većim brojem odgovora), klijent koji vrši pretraživanje šalje poruku s upitom (engl. *probe message*) prema višeodredišnoj grupi, a usluge koje odgovaraju postavljenom kriteriju odgovaraju izravno klijentu koji je postavio upit. Ako klijent želi otkriti uslugu prema njenom imenu, klijent šalje poruku razlučivanja (engl. *resolution message*) na istu višeodredišnu grupu, a pronađena usluga ponovno vraća odgovor izravno klijentu koji vrši pretragu.

Iz osnovnog poznavanja mrežne arhitekture i protokola koji se koriste u umrežavanju da bi striktna primjena ovdje opisanog modela izazvala zagušenje mreže, u normi je definirano da usluga, kada se priključuje na mrežu, šalje svojoj višeodredišnoj grupi najavnu poruku (engl. *announcement message*), a klijenti trebaju oslušivati poruke koje se pojavljuju u grupi kako bi na vrijeme mogli otkriti koje su nove usluge dostupne na mreži, bez potrebe za neprestanim slanjem poruka.

Kako bi ovaj način mogao funkcionirati i u većim mrežama, norma definira i rad u tzv. upravljanom načinu rada. U ovom se načinu uvodi posrednik za otkrivanje usluga (engl. *discovery proxy*) pa usluge mogu slati najavne poruke izravno posredniku koji se nalazi u njihovoj mreži, te on preuzima daljnji proces najave za usluge koje je registrirao. Kada neki klijent želi otkrivati usluge, kontaktira izravno dostupne posrednike, prebacuje svoj način izvođenja u upravljani način te pretražuje na sličan način kao i u *ad-hoc* načinu rada (putem poruka s upitom i poruka razlučivanja). Ako posrednik iz bilo kojeg razloga postane nedostupan, klijent se vraća u *ad-hoc* način rada te nastavlja pretraživati svoju višeodredišnu grupu.

Pažljivijim čitanjem same norme možemo uočiti u čemu je razlika među otkrivanjem usluga putem UDDI URB registra i putem WS-Discovery norme. Dok je kod UDDI registra bilo moguće otkriti samo one usluge koje je njihov vlasnik registrirao, WS-Discovery nalaže uslugama da

²⁷² OASIS Standard, Web Services Dynamic Discovery (WS-Discovery) Version 1.1, 1. July 2009, <http://docs.oasis-open.org/ws-dd/discovery/1.1/os/wsdd-discovery-1.1-spec-os.pdf>

²⁷³ Beatty, J. et al., Web Services Dynamic Discovery (WS-Discovery), Microsoft Corporation, April 2005.

se same „objave“, bilo izravno klijentima, bilo putem dostupnog posrednika. Tako je kod UDDI registra bilo moguće da brojne usluge budu dostupne u mreži, ali nisu mogle biti otkrivene jer nitko za njih nije proveo proces registracije u nadležnom UDDI registru. Naravno, ako klijent nije mogao otkriti uslugu u UBR-u, nije ju mogao ni koristiti.

Ovakav je način otkrivanja usluga, koji je zasnovan na razmjeni poruka i (relativno) intenzivnoj komunikaciji putem mreže, podložan sigurnosnim prijetnjama, poput promjene poruka, ograničavanja u isporuci usluge (engl. *Denial of Service*) ili lažnog predstavljanja. U samoj WS-Discovery normi definirana je mogućnost potpisivanja SOAP poruka koje se razmjenjuju putem UDP *unicast* ili *multicast* poruka. Kako bi povećali sigurnost u vlastitim rješenjima otkrivanja usluga zasnovanog na WS-Discovery normi, mogu se koristiti i ostali dostupni mehanizmi iz WS-* grupe normi, poput WS-Security, WS-Trust i ostalih.

Ova se norma primjenjuje i u nekim uslugama modernih inačica Windows operativnog sustava, poput usluge „*People near me*“ ili pretrage za dostupnim pisačima na mreži.

Sljedeći programski kôd koristi se pri pretraživanju na mreži dostupnih pisača koju klijent šalje na dostupnu višedrežnu grupu:

```
<s:Envelope
  xmlns:a="http://schemas.xmlsoap.org/ws/2004/08/addressing"
  xmlns:d="http://schemas.xmlsoap.org/ws/2005/04/discovery"
  xmlns:i="http://printer.example.org/2003/imaging"
  xmlns:s="http://www.w3.org/2003/05/soap-envelope" >

  <s:Header>
    <a:Action>
      http://schemas.xmlsoap.org/ws/2005/04/discovery/Probe
    </a:Action>
    <a:MessageID>
      uuid:0a6dc791-2be6-4991-9af1-454778a1917a
    </a:MessageID>
    <a:To>

      </a:To>
    </s:Header>
    urn:schemas-xmlsoap-org:ws:2005:04:discovery

    <s:Body>
      <d:Probe>
        <d:Types>

          i:PrintBasic
        </d:Types>
        <d:Scopes>
          MatchBy=http://schemas.xmlsoap.org/ws/2005/04/discovery/ldap>
          ldap:///ou=programiranje,o=racunarstvo,c=hr
        </d:Scopes>
      </d:Probe>
    </s:Body>

  </s:Envelope>
```

Oznakom s:Action definirano je da se kao shema za ovaj upit koristi poruka s upitom, dok oznaka d:Types definira da klijent želi dobiti popis onih usluga koje omogućuju ispisivanje

dokumenata. Kao odgovor na ovakav upit, ako postoji neka usluga koja omogućuje ispis, stići će ovakva poruka:

```
<s:Envelope
  xmlns:a="http://schemas.xmlsoap.org/ws/2004/08/addressing"
  xmlns:d="http://schemas.xmlsoap.org/ws/2005/04/discovery"
  xmlns:i="http://printer.example.org/2003/imaging"
  xmlns:s="http://www.w3.org/2003/05/soap-envelope" >

  <s:Header>
    <a:Action>
      http://schemas.xmlsoap.org/ws/2005/04/discovery/ProbeMatches
    </a:Action>
    <a:MessageID>
      uuid:e32e6863-ea5e-4ee4-997e-69539d1ff2cc
    </a:MessageID>
    <a:RelatesTo>
      uuid:0a6dc791-2be6-4991-9af1-454778a1917a
    </a:RelatesTo>
    <a:To>
      http://schemas.xmlsoap.org/ws/2004/08/addressing/role/anonymous
    </a:To>
    <d:AppSequence InstanceId="1077004800" MessageNumber="2" />
  </s:Header>

  <s:Body>
    <d:ProbeMatches>
      <d:ProbeMatch>
        <a:EndpointReference>
          <a:Address>
            uuid:98190dc2-0890-4ef8-ac9a-5940995e6119
          </a:Address>
        </a:EndpointReference>
        <d:Types>
          i:PrintBasic i:PrintAdvanced
        </d:Types>
        <d:Scopes>
          ldap:///ou=programiranje,o=racunarstvo,c=hr
          ldap:///ou=prizemlje,ou=soba5,o=racunarstvo,c=hr
          http://vspr/programiranje/ispis/2010-07-30
        </d:Scopes>
        <d:XAddr>
          http://prn-example/PRN42/b42-1668-a
        </d:XAddr>
        <d:MetadataVersion>
          75965
        </d:MetadataVersion>
      </d:ProbeMatch>
    </d:ProbeMatches>
  </s:Body>
</s:Envelope>
```

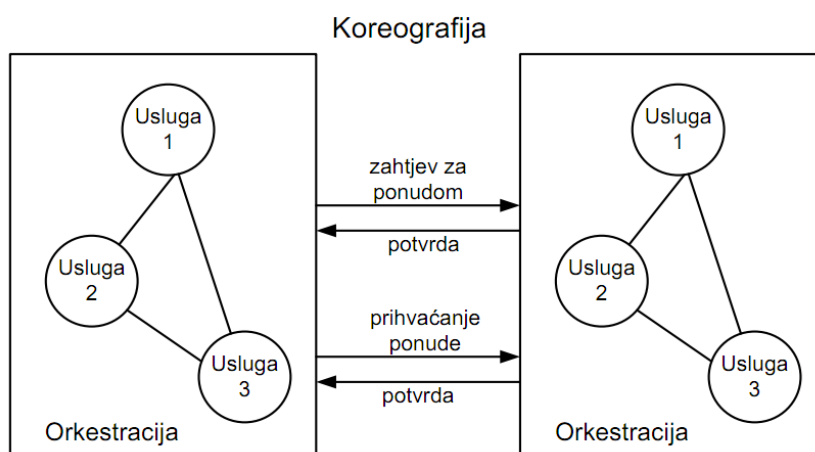
Vidimo da smo dobili dosta precizne podatke o dostupnom pisaču (lokaciju putem LDAP kazala i adresu na kojoj se usluga ispisa može dobiti).

6.5 Slaganje usluga

Do sada smo u ovom poglavlju objasnili osnovne stvari koje se tiču poslovanja u sustavima orijentiranim na uslugu, ugovore koje treba potpisati kako bi se koristile usluge te načine otkrivanja nekih dostupnih usluga na mreži. Sljedeći korak u razvoju poslovnog sustava naše male programerske tvrtke je pojasniti kako, nakon što smo pronašli neke usluge na tržištu koje bismo željeli uključiti u naše rješenje te riješili pravne formalnosti, izvršiti slaganje dijelova u novu uslugu koju ćemo ponuditi na tržištu.

Kako bi se ovaj proces mogao uspješno izvoditi, neobično je važno imati poslovni sustav otvoren prema korištenju dijelova programskog koda koji je nastao u drugim organizacijama²⁷⁴. Kako su tvrtke često jako osjetljive na takvu vrstu suradnje, trebalo bi osmisliti posebne mehanizme kako bi se prevladalo nepovjerenje među poslovnim organizacijama.

Osmišljeni su različiti pristupi u rješavanju ovog problema i za sada nijedan nije postao prevladavajućom normom. No dva su se pristupa izdvojila, nazvana orkestracija i koreografija²⁷⁵ (engl. *Orchestration and Coreography*). Ova se dva pristupa djelomice preklapaju, a njihov međusobni odnos prikazan je na slici (Slika 6.7).



Slika 6.7: Orkestracija i koreografija

Orkestracija je model za povezivanje raznih usluga u kojem je sva kontrola zadržana na jednoj strani (npr. kod pružatelja usluge). Veze prema uslugama ostalih strana odvijaju se preko poruka.

Takve poruke sadržavaju dijelove poslovne logike i poredak izvršavanja poslova, koji je potreban kako bi usluga ispravno završila svoj rad.

S druge strane, koreografija usluga zasnovana je na suradnji različitih tvrtki. Ona svakoj strani dopušta opis svojeg dijela interakcije. Koreografija usluga prati razmjenu sljedova poruka

²⁷⁴ Srivastava, B., Koehler, J., Web Service Composition - Current Solutions and Open Problems, Proceedings of ICAPS 2003, 2003.

²⁷⁵ Peltz, C., Web services orchestration and choreography. Computer Volume 36, Issue 10, Oct. 2003 Page(s): 46 – 52

između više strana, a ne samo poslovni proces za koji je zainteresiran samo jedan sudionik interakcije.

Već smo objasnili da je za međusobno razumijevanje dviju strana (koje ne moraju nužno „pričati“ istim jezikom) potrebno definirati jezik koji će obje strane neupitno razumjeti. Iz istog je razloga za potrebe slaganja usluga razvijeno nekoliko na XML-u zasnovanih jezika. Prvo su Microsoft svojim XML zasnovanim jezikom XLANG²⁷⁶ te IBM svojim Web Services Flow Languageom (WSFL)²⁷⁷ pokušali definirati normu jezika za opis poslovnih procesa. Kompanije su kasnije združile ove prijedloge kako bi stvorile Business Process Execution Language for Web Services (BPEL4WS ili BPEL ili WS-BPEL)²⁷⁸.

BPEL je gramatika zasnovana na XML-u za opis kontrolne logike potrebne za koordinaciju usluga koje sudjeluju u nekom poslovnom procesu. Orkestracijski mehanizam izvršava tu gramatiku, koordinirajući i kompenzirajući cjelokupni proces u slučaju pogrešaka. BPEL se nalazi u sloju iznad WSDL-a. WSDL sučelje opisuje koje su operacije dopuštene, a BPEL određuje kojim će redoslijedom te operacije biti izvršene.

WSDL su opisani dostupni ulazi i izlazi iz svakog BPEL procesa, dok WSDL tipovi podataka opisuju informaciju razmijenjenu tijekom izvedbe poslovnog procesa. WSDL-om se mogu definirati koje usluge dostupne kod ostalih pružatelja usluga BPEL može zahtijevati kako bi se poslovni proces uspješno završio. BPEL također osigurava robusni mehanizam za izvršavanje transakcija i obradu iznimaka.

Sličan na XML-u zasnovan jezik za opis poslovnih procesa²⁷⁹ je i Business Process Management Language (BPML)²⁸⁰. Prvotno zamišljen kao potpora sustavu za upravljanje poslovnim procesima, u posljednjoj verziji u normu je dodana i podrška za Web Services Coreography Interface (WSCI)²⁸¹. Oba spomenute norme dijele isti model izvršavanja procesa, pa stoga razvojni programeri mogu koristiti WSCI za opis javnih interakcija između poslovnih procesa, dok BPML mogu sačuvati za razvoj privatnih implementacija.

BPML i BPEL imaju slične načine konstrukcije toka procesa. Definirane su osnovne aktivnosti za slanje, primanje i pozivanje usluga, kao i aktivnosti potrebne za izbor uvjetnih odluka, sekvencijalnih i paralelnih aktivnosti, spajanja usluga i petlji usluga. BPML podržava i vremensko planiranje kada će se neka operacija izvršiti. Podržane su i brze i spore transakcije, zajedno s ugnježđivanjem procesa i transakcija, te robusan sustav za obradu iznimki.

Bez obzira na rješenje koje će se koristiti u konkretnoj implementaciji prilagodbe poslovnog procesa prednostima koje nudi korištenje usluga, potrebno je u jednom od jezika za opis poslovnih procesa definirati proces, sudionike u tom procesu te njihove uloge (koje ovise o problemu koji želimo riješiti) u njemu. Također treba definirati i tok informacija među

²⁷⁶ Thatte, S., XLANG:Web Services for Business Process Design, Microsoft, Redmond, USA, 2001.

²⁷⁷ Leymann, F., Web Services Flow Language (WSFL 1.0), IBM, May 2001.

²⁷⁸ Curbera, F. et al., Business Process Execution Language for Web Services (Version 1.0), IBM, July 2002.

²⁷⁹ Shapiro, R., A Technical Comparison of XPD, BPML and BPEL4WS, Cape Visions, USA, August 2002.

²⁸⁰ Arkin, A. et al., Business Process Modeling Language (BPML), Version 1.0, 2002.

²⁸¹ Arkin et al., Web Service Choreography Interface 1.0 (WSCI), 2002.

sudionicima procesa. Za primjer, u jeziku BPEL to se čini pomoću varijabli. Treba uspostaviti i mehanizam za korelaciju među različitim zahtjevima.

Kao ključni dio opisa poslovnog procesa definiraju se koraci potrebni za procesiranje zahtjeva. Sam proces treba definirati vrlo precizno. Pažljivo treba precizirati način, format poruka koje se razmjenjuju među sudionicima poslovnog procesa te obradu potencijalnih pogrešaka u ovakvom sustavu. Brojna istraživanja provedena u ovom području upućuju na to da se kao najbolje rješenje pokazala uspostava nekog transakcijskog mehanizma, koji će se brinuti da cijeli proces završi na pravilan i predviđeni način.

6.6 Upravljanje i nadzor nad sustavom

Kako bi u cijelosti iskoristili sve u ovom poglavlju iznesene prednosti koje donosi implementacija načela arhitekture SOA, trebamo još definirati kako cijelim tim sustavom možemo upravljati na što jednostavniji i učinkovitiji način. Jedan od mehanizama koji se predlaže nazvan je SOA Governance²⁸².

Jedna od djelatnosti koja je prva počela široko primjenjivati sustav orijentiran uslugama bili su telekomunikacijski operatori. Telekomunikacijske sustave obilježava vrlo velik broj korisnika te vrlo malo vrijeme u kojima takvi sustavi smiju biti nedostupni (čest je zahtjev da dostupnost telekomunikacijskog sustava bude 99.9999% ili više – da je to zaista i ostvarivo, možete se uvjeriti i sami – kada ste zadnji put pokušali nazvati prijatelja, a da putem telefona, bilo fiksnog ili mobilnog, niste bili u mogućnosti ostvariti poziv?).

Operatori, osobito oni u pokretnom dijelu, pod dodatnim su pritiskom kako bi sve brže izdavali nove usluge te snižavali cijene u borbi za svakog pretplatnika. Sve to dovodi do želje operatora da snize troškove koje su prisiljeni izdvajati za održavanje mreže svojih usluga, i to je glavni razlog zašto su se okrenuli ka SOA sustavima, čija je to jedna od glavnih prednosti.

SOA Governance predstavlja koncept upravljanja IT sustavima koji koriste sustave zasnovane na uslugama, te osigurava uvođenje pravila i kontrola u takve sustave. Njome se definiraju i vode procesi razvoja i održavanja ponovno iskoristivih usluga, a brine se i o procesima izrade ugovora između pružatelja i korisnika usluge²⁸³. Model je nastao na načelima koja su u programskom inženjerstvu poznata u području životnog ciklusa programskog proizvoda.

Sustav zasnovan na načelima orijentiranosti usluzi po prvi puta u telekomunikacijsko okruženje donosi otvaranje dijelova IT sustava prema ostalim sudionicima u poslovnom procesu na razini pojedinih sastavnica. Kako se sastavnice dijele s aplikacijama ostalih sudionika na telekomunikacijskom tržištu, posebna se pozornost mora posvetiti dizajnu, implementaciji i isporuci sastavnica i usluga, kako ne bi došlo do smetnji u korištenju usluga. Slika 6.8 prikazuje životni ciklus SOA sustava i osnovne elemente njegova nadzora.

²⁸² Durvasula, S. et al., SOA Practitioner's Guide, BEA Systems, Inc. 2006

²⁸³ Woolf, B., Introduction to SOA Governance, IBM, 2006.

Tri osnovna dijela životnog ciklusa (dizajn, izvedba i promjena) nadgleda nadzor životnog ciklusa usluge (engl. *Service Lifecycle Governance*).



Slika 6.8: SOA životni ciklus i SOA Governance

Nadzor dizajna (engl. *Design-Time Governance*). Nadgledanje dizajna preuzeto je iz tradicionalnih funkcija nadziranja IT sustava te uključuje uvođenje politika (engl. *policies*) za nadgledanje definiranja izvedbe usluge. Kako je u SOA sustavu cilj ponovno iskoristiti već postojeće sastavnice, dizajner mora temeljem specifikacija pretražiti kako već postojeće sastavnice, tako i one koje se već razvijaju, ali se još ne koriste. Provjere mogu uključivati i jesu li nove usluge suglašene s postojećim normama.

Nadzor za vrijeme izvođenja (engl. *Run-time Governance*). Nadgledanje u fazi izvođenja vezano je za definiranje i provedbu politika za kontrolu isporuke, prilagodbe kao i ostalih postupaka koje treba provesti nad uslugama koji se izvođe. Tipično se vezuju za nefunkcijske zahtjeve, poput zahtjeva za uspostavom povjerenja ili upravljanja kvalitetom usluge.

Nadzor promjena (engl. *Change-Time Governance*). Tijekom životnog ciklusa bilo kojeg programskog proizvoda, pa tako i usluge, neminovno dolazi do potrebe za njenom preinakom, bilo zbog promjena u poslovnim zahtjevima ili otkrivenih pogrešaka. Tradicionalni pristup je prilagodba softvera, što uključuje i razvoj potpuno novih sastavnica. Slaganje već postojećih usluga novost je koju donosi implementacija SOA sustava, jer se za kraće vrijeme mogu dobiti nove usluge ili poboljšati one stare. Aspekti ponašanja SOA sustava koji se najbrže mijenjaju su ugovori i politike. Njihovo je mijenjanje regulirano konfiguracijom, koja je ujedno i najbrža metoda promjene u SOA sustavu. Važno je naglasiti da je u ovu fazu nadzora nužno uključiti i poslovnu stranu kako bi se osiguralo da promjene na nekom dijelu sustava ne bi ugrozile cjelokupno poslovanje, na što posebno treba paziti ako su usluge koje su dostupne za korištenje ostalim poslovnim partnerima izvan vlastite tvrtke.

6.7 Primjeri sustava orijentiranih uslugama

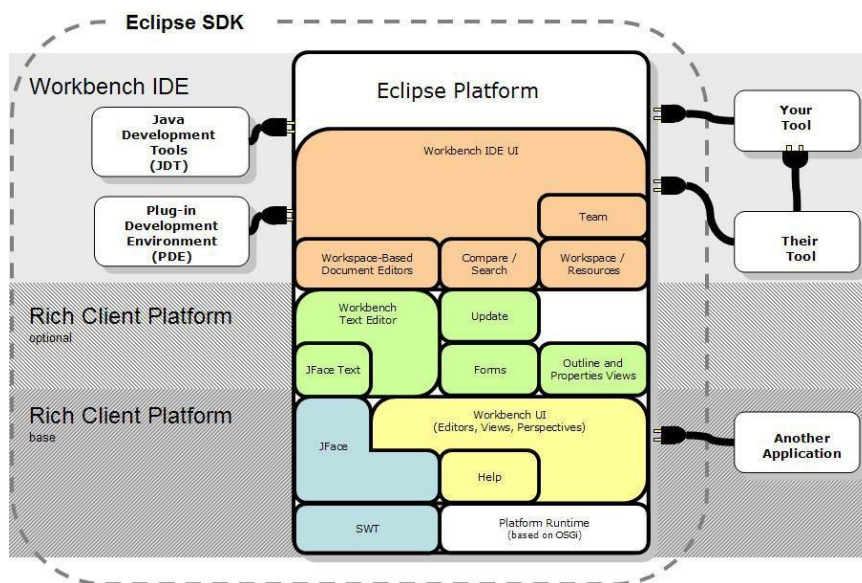
6.7.1 Arhitektura Eclipse dodataka

Jedna od najpopularnijih platformi za razvoj programskih sustava, Eclipse, sastoji se od integriranog razvojnog sustava i proširivog dijela za dodatke (engl. *plug-in*). Sama je platforma većinom napisana u programskom jeziku Java, a korisnici mogu dodavati nove funkcionalnosti korištenjem razvojnog okruženja za dodatke (engl. *plug-in development environment*).

Eclipse kao programsku osnovu koristi Equinox²⁸⁴, jednu od implementacija specifikacije programskog okruženja OSGi²⁸⁵. OSGi (engl. *Open Services Gateway Initiative*) je modularni sustav koji omogućava razvoj i implementaciju dinamičkih sastavnica, koje se nazivaju paketima (engl. *bundles*). Ti se paketi registriraju na OSGi platformi te je pomoću njih moguće kreirati usluge.

Osim jednog manjeg dijela same jezgre, zapravo je cijela platforma Eclipse sastavljena od brojnih dodataka. Upravo je to jedna od prednosti ovog sustava, jer se i dodaci koji su proizvedeni kao dio službenog izdanja smatraju jednakovrijednima onima koje su, na temelju specifikacije, napravile neke druge tvrtke. Neki od vanjskih dodataka su vrlo popularni, jer rješavaju probleme koje je prepoznala šira programerska zajednica.

Neki su od takvih dodatka ušli čak i u sljedeće izdanje platforme Eclipse. Slika 6.9 **Error! Reference source not found.** prikazuje arhitekturu Eclipse sustava²⁸⁶.



Slika 6.9: Arhitektura platforme Eclipse

²⁸⁴ Eclipse Equinox, <http://www.eclipse.org/equinox/>

²⁸⁵ OSGi, <http://www.osgi.org/>

²⁸⁶ Slika preuzeta sa

http://www.idg2e.com/idg2e_CD_for_eclipse321/documentation/html/ch08.architecture/doc/index.html. ©

Copyright International Business Machines Corporation, 2003, 2004, 2006. All Rights Reserved.

Kako bi neki program mogao postati dodatak za platformu Eclipse, mora se moći povezati na neku od postojećih točaka za proširenje (na gornjoj slici to bi bile točke gdje se utičnice s dodataka spajaju na platformu). Također, programski kôd treba dodatno opremiti dvama datotekama, tzv. *manifest-files*, u kojima je sve što je potrebno kako bi se dodatak mogao izvesti te programski kôd dodatno opremiti eventualnim novim točkama proširenja.

Primjerice za jednostavan program Hello World takav bi manifest izgledao otprilike ovako:

```
<?xml version="1.0" encoding="UTF-8"?>
<?eclipse version="3.2"?>
<plugin>
  <extension point="org.eclipse.ui.views">
    <category
      name="Hello Category"
      id="com.example.helloworld">
    </category>
    <view
      name="Hello View"
      icon="icons/sample.gif"
      category="com.example.helloworld"
      class="com.example.helloworld.views.HelloWorldView"
      id="com.example.helloworld.views.HelloWorldView">
    </view>
  </extension>
</plugin>
```

Pokretanjem platforme Eclipse zapravo se ne pokreće jedan program, već se pokreće cijeli niz programa koji su međusobno ovisni i čine jednu cjelinu. Sve te dodatke zapravo možemo smatrati uslugama, pa je cijela platforma Eclipse zapravo sustav orijentiran uslugama.

Plug-in Development Environment (PDE)²⁸⁷ osigurava alate za razvoj dodataka kojima se proširuju funkcionalnosti platforme Eclipse. Iako u pravilu dodaci sadržavaju Java kôd, oni mogu sadržavati i drugačije dijelove, poput datoteka u jeziku HTML koje služe kako bi dokumentacija o samom dodatku uvijek bila raspoloživa korisnicima („online“).

Svaki dodatak ima vlastiti učitavač klasa (engl. *Class loader*). Dodaci mogu ovisiti o drugim dodacima. Te se ovisnosti opisuju navođenjem ključne riječi *requires* u dokumentu *plugin.xml*. Ako za primjer pogledamo dodatak *org.eclipse.ui*, to izgleda kao u sljedećem programskom odsječku (naveden je samo dio dodataka o kojima ovaj dodatak ovisi):

```
<?xml version="1.0" encoding="UTF-8"?>
<plugin
  id="org.eclipse.ui"
  name="%Plugin.name"
  version="2.1.1"
  provider-name="%Plugin.providerName"
  class="org.eclipse.ui.internal.UIPlugin">
  <runtime>
    <library name="ui.jar">
    <export name="*/>
```

²⁸⁷ <http://www.aosabook.org/en/eclipse.html>

```

        <packages prefixes="org.eclipse.ui"/>
    </library>
</runtime>
<requires>
    <import plugin="org.apache.xerces"/>
    <import plugin="org.eclipse.core.resources"/>
    <import plugin="org.eclipse.update.core"/>
//ovdje je izbačen dio dodataka zbog preglednosti koda
    <import plugin="org.eclipse.text" export="true"/>
    <import plugin="org.eclipse.ui.workbench.texteditor" export="true"/>
    <import plugin="org.eclipse.ui.editors" export="true"/>
</requires>
</plugin>

```

Kako je Eclipse zajednica razvijala sve više dodataka, javila se potreba za ujednačavanjem i proširenjem postojećih točaka za proširenje. U novim je verzijama moguće je navesti koje od vlastitih klasa (koje se nalaze u našem dodatku za platformu Eclipse) želimo proglasiti takvima, odnosno omogućiti drugima ograničen uvid u njihov dizajn.

Za primjer uzmimo da želimo dodati jedan element u alatnu traku. Koristiti ćemo `actionSets` točku za proširenje koje se nalaze u dodatku `org.eclipse.ui`:

Za primjer uzmimo da želimo dodati jedan element u alatnu traku. Koristiti ćemo `actionSets` točku za proširenje koje se nalaze u dodatku `org.eclipse.ui`:

```

<extension-point id="actionSets"
    name="%ExtPoint.actionSets"
    schema="schema/actionSets.exsd"/>
<extension-point id="commands"
    name="%ExtPoint.commands"
    schema="schema/commands.exsd"/>
<extension-point id="contexts"
    name="%ExtPoint.contexts"
    schema="schema/contexts.exsd"/>
<extension-point id="decorators"
    name="%ExtPoint.decorators"
    schema="schema/decorators.exsd"/>
<extension-point id="dropActions"
    name="%ExtPoint.dropActions"
    schema="schema/dropActions.exsd"/>

```

Proširenje našeg dodatka kojim dodajemo element u alatnu traku (točnije u `org.eclipse.ui.actionSet` točku za proširenje) izgleda ovako:

```

<?xml version="1.0" encoding="UTF-8"?>
<plugin
    id="com.example.helloworld"
    name="com.example.helloworld"
    version="1.0.0">
    <runtime>
        <library name="helloworld.jar"/>
    </runtime>
    <requires>
        <import plugin="org.eclipse.ui"/>
    </requires>
    <extension
        point="org.eclipse.ui.actionSets">

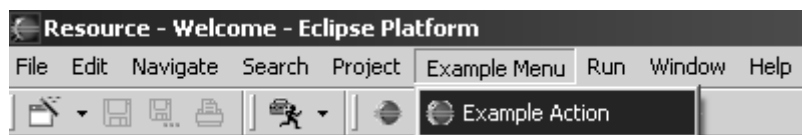
```

```

<actionSet
    label="Example Action Set"
    visible="true"
    id="org.eclipse.helloworld.actionSet">
    <menu
        label="Example &Menu"
        id="exampleMenu">
        <separator
            name="exampleGroup">
        </separator>
    </menu>
    <action
        label="&Example Action"
        icon="icons/example.gif"
        tooltip="Hello, Eclipse world"
        class="com.example.helloworld.actions.ExampleAction"
        menubarPath="exampleMenu/exampleGroup"
        toolbarPath="exampleGroup"
        id="org.eclipse.helloworld.actions.ExampleAction">
    </action>
</actionSet>
</extension>
</plugin>

```

Kada se Eclipse pokreće, pretražuju se svi postojeći manifesti svih dodataka koji se nalaze u vašem okruženju, te se stvara (engl. *Builds*) registar dodataka koji se sprema u memoriju. Priključne točke i pripadajuća proširenja označavaju se imenima. Tako stvoren registar dodataka može se potom referencirati kroz API same platforme. Za buduće potrebe, kopija registra snima se i na disk, kako bi bila dostupna prilikom sljedećeg pokretanja. Važno je napomenuti da se svi dodaci učitavaju u registar, no ne aktiviraju se (nema učitavanja klasa), dok njihov kôd ne postane potreban platformi za izvođenje (npr. kada korisnik otvori neki dodatak). Takav pristup zovemo lijena aktivacija (engl. *Lazy activation*). Tako se naš dodatak u alatnu traku neće učitati dok korisnik ne odabere upravo njega u alatnoj traci.



Slika 6.10: XML-RPC arhitektura

U nastavku je primjer koda kojim se generira ovaj dodatak u alatnoj traci:

```

package com.example.helloworld.actions;

import org.eclipse.jface.action.IAction;
import org.eclipse.jface.viewers.ISelection;
import org.eclipse.ui.IWorkbenchWindow;
import org.eclipse.ui.IWorkbenchWindowActionDelegate;
import org.eclipse.jface.dialogs.MessageDialog;

public class ExampleAction implements IWorkbenchWindowActionDelegate {
    private IWorkbenchWindow window;

    public ExampleAction() {
    }
}

```

```

    public void run(IAction action) {
        MessageDialog.openInformation(
            window.getShell(), "org.eclipse.helloworld",
            "Hello, Eclipse architecture world");
    }

    public void selectionChanged(IAction action, ISelection selection) {
    }

    public void dispose() {
    }

    public void init(IWorkbenchWindow window) {
        this.window = window;
    }
}

```

Kada korisnik odabere novi element u alatnoj traci, Eclipse šalje upit na registar proširenja kako bi pronašao koji dodatak je odgovoran za tu funkcionalnost. Tada se dodatak instancira i učitavaju se potrebne klase. Iz oda je vidljivo da će se otvoriti novi okvir s porukom „Hello, Eclipse architecture world“.

Kao što smo naveli, doprinos ne mora sadržavati samo izvorni kod, već se tu primjerice mogu nalaziti i datoteke za pomoć, dokumentaciju i sl. U Eclipse-u je to riješeno na način da se nadograđuje sustav za pomoć. On je sagrađen od malih jedinica informacije koji se nazivaju teme (engl. *Topics*). Tema se sastoji od oznake i reference na svoju lokaciju. Pri tome, lokacije može primjerice biti HTML datoteka, ili načelno bilo kakav XML dokument koji opisuje dodatne izvore. Teme združujemo u liste znane pod imenom tablica sadržaja ToC (engl. *Table of contents*). Ako na primjer želimo dodati sadržaj pomoći svojoj aplikaciji, povezujemo se na točku proširenja definiranu kao `org.eclipse.help.toc`, kao što prikazuje sljedeći programski odsječak.

```

<?xml version="1.0" encoding="UTF-8"?>
<?eclipse version="3.0"?>
<plugin>
<!-- ===== -->
<!-- Definiramo glavni TOC -->
<!-- ===== -->
    <extension
        point="org.eclipse.help.toc">
        <toc

            file="toc.xml"
            primary="true">
            <index path="index"/>
            </toc>
        </extension>
<!-- ===== -->
<!-- Definiramo preostale TOCs -->
<!-- ===== -->
    <extension
        point="org.eclipse.help.toc">
        <toc
            file="topics_Guide.xml">

```

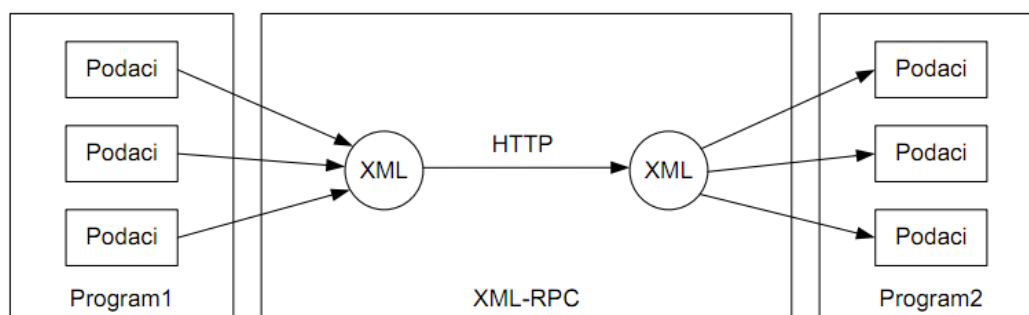
```

</toc>
<toc
    file="topics_Reference.xml">
</toc>
<toc
    file="topics_Porting.xml">
</toc>
<toc
    file="topics_Questions.xml">
</toc>
<toc
    file="topics_Samples.xml">
</toc>
</extension>

```

6.7.2 XML-RPC

XML-RPC²⁸⁸ ili XML za udaljeni poziv procedura (engl. *remote procedure call*) je specifikacija koja omogućava programima koji se izvode na različitim operativnim sustavima i u različitim okruženjima da međusobno izmjenjuju pozive procedura preko mreže. Kao prijenosni protokol koriste široko podržani protokol HTTP, dok se same poruke izvode korištenjem jezika XML.



Slika 6.11: Arhitektura XML-RPC

Slika 6.11 prikazuje načelnu arhitekturu na kojoj funkcionira XML-RPC. S obzirom na to da je cilj XML-RPC-a osigurati komunikaciju među različitim programskim sustavima, pokazujemo ga ovdje iako zapravo nije riječ o pravoj usluzi.

Pokažimo kako XML-RPC radi na jednom jednostavnom primjeru. Imamo program koji zbraja dva broja. Poštujući načelo objektne orijentiranosti, postoji zasebna metoda koja izvodi operaciju zbrajanja. Kako bi neki drugi program mogao udaljeno pozvati ovu metodu, ona mora biti implementirana kao javna metoda u programskom kodu, a ostatak programa mora znati prepoznati takav poziv.

```

import org.apache.xmlrpc.*;
public class Zbrajalo {

    public Integer sum(int x, int y) {
        return new Integer(x+y);
    }
}

```

²⁸⁸ XML-RPC, <http://www.xmlrpc.com/>

```

    }

    public static void main (String [] args) {
        try {
            System.out.println("Attempting to start XML-RPC Server...");
            WebServer server = new WebServer(80);
            server.addHandler("sample", new JavaServer()); server.start();
            System.out.println("Started successfully.");
            System.out.println("Accepting requests. (Halt program to stop.)");
        } catch (Exception exception) {
            System.err.println("JavaServer: " + exception);
        }
    }
}

```

Pogledajmo kako izgleda druga strana:

```

import java.util.*;
import org.apache.xmlrpc.*;

public class Klijent {
    public static void main (String [] args) {
        try {
            XmlRpcClient server = new XmlRpcClient("http://localhost/RPC2");
            Vector params = new Vector();
            params.addElement(new Integer(17));
            params.addElement(new Integer(13));
            Object result = server.execute("sample.sum", params);
            int sum = ((Integer) result).intValue();
            System.out.println("The sum is: " + sum);
        } catch (Exception exception) {
            System.err.println("Klijent: " + exception);
        }
    }
}

```

Pozivanjem naredbe `server.execute` klasa `Klijent` će poslati sljedeću poruku na klasu `Zbrajalo`:

```

<?xml version="1.0" encoding="ISO-8859-1"?>
<methodCall>
  <methodName>
    sample.sum
  </methodName>
  <params>
    <param>
      <value>
        <int>
          17
        </int>
      </value>
    </param>
    <param>
      <value>
        <int>
          13
        </int>
      </value>
    </param>
  </params>
</methodCall>

```

```
</param>
</params>
</methodCall>
```

Vidimo da se u njoj nalaze parametri koje želimo prenijeti na Zbrajalo. Po primitku poruke Zbrajalo će obraditi poruku, pozvat će se tražena metoda te će se Klijent-u uputiti sljedeća poruka s rezultatom, a dobiveni se kôd može dalje koristiti:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<methodResponse>
  <params>
    <param>
      <value>
        <int>
          30
        </int>
      </value>
    </param>
  </params>
</methodResponse>
```

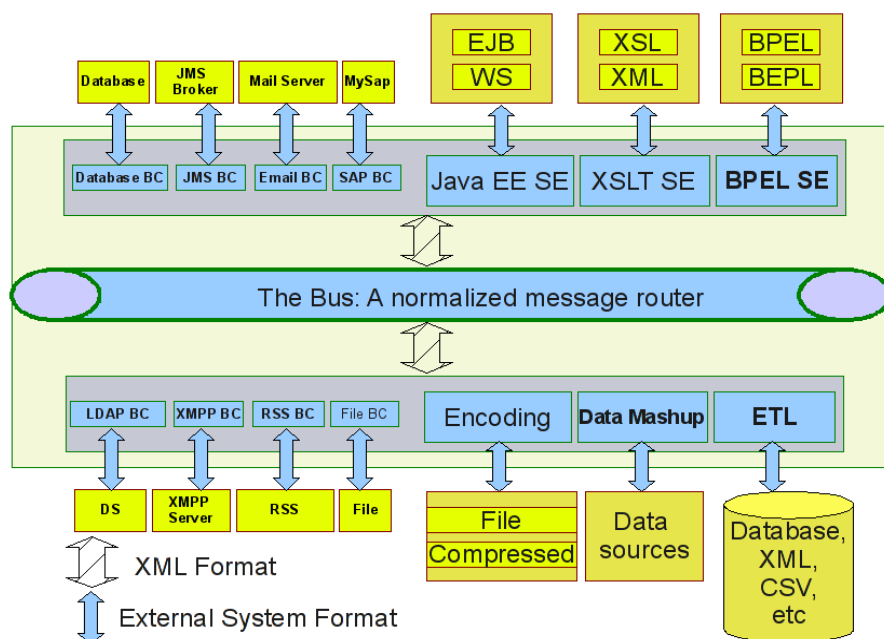
6.7.3 Otvorena sabirnica Open ESB

Pod pojmom **Enterprise Service Bus (ESB)**²⁸⁹ podrazumijevamo raspodijeljenu infrastrukturu koja se koristi za integraciju poslovnih aplikacija. Ona sadrži sučelja prema različitim aplikacijama, a one međusobno komuniciraju putem ugrađenog sustava poruka. ESB se brine za transformaciju, orkestraciju, pregovaranje i usmjeravanje poruka kao i za centralizirano upravljanje sustavom.

Kako je model ESB dovoljno generički napravljen, postoji nekoliko desetaka implementacija, što komercijalnih, a što u domeni sustava otvorenog koda. Radi jednostavnosti i šire dostupnosti, ovdje ćemo ukratko prikazati jednu od najpoznatijih implementacija, OpenESB.

Arhitektura **OpenESB** omogućuje korisnicima da međusobno povežu aplikacije koje su dizajnirane da podatke razmjenjuju u različitim formatima te putem različitih protokola. Kao osnova za to koriste se povezujuće komponente BC (engl. *Binding Components*), koje osiguravaju prilagodbu među različitim protokolima (npr. FTP, HTTP, CORBA, vlasnički protokoli itd.). Druga glavna funkcionalna cjelina naziva se uslužni centar SE (engl. *Service Engine*) koja se brine o prevođenju poruka i višoj razini poslovne logike (npr. XSLT, BPEL, prevoditelji i sl.). Normalizirani usmjernik poruka (engl. *Normalized Message Router*) osigurava razdvajanje modula i osigurava jedinstveni komunikacijski model. Uz već dokazani aplikacijski poslužitelj otvorenog koda GlassFish, OpenESB koristi i druge otvorene standarde, poput Java Business Integration (JBI), WSDL, XML Scheme i drugih.

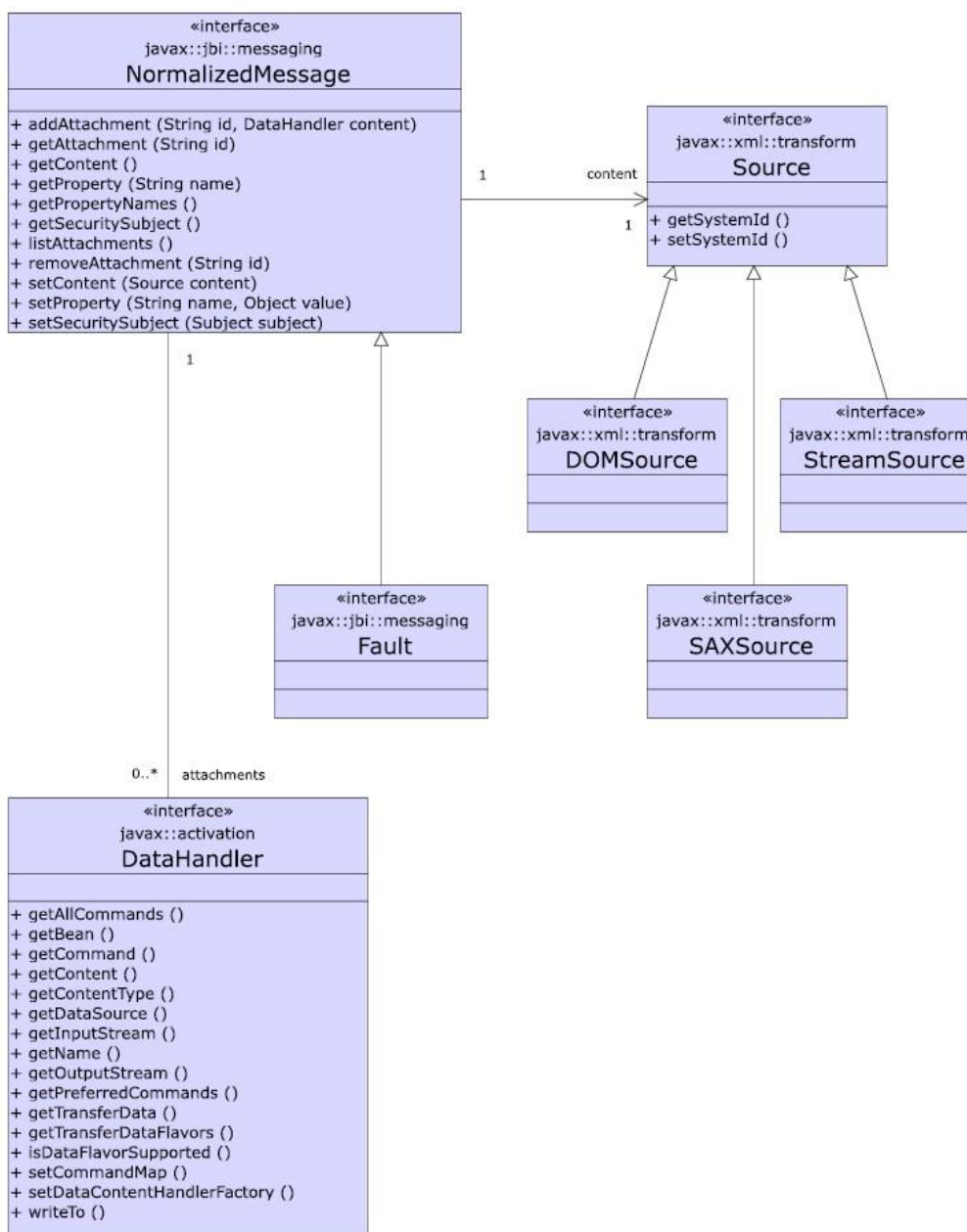
²⁸⁹ David Chappell, Enterprise Service Bus, O'Reilly, June 2004

Slika 6.12: Arhitektura OpenESB²⁹⁰

Kako se BC i SE dosta razlikuju kod pojedinih implementacija, nećemo ulaziti u detalje na koji su način izvedene, već se može više o njima saznati kroz literaturu. Posvetimo zato nekoliko trenutaka trećem osnovnom dijelu arhitekture OpenESB, a to je NMR.

Normalizirani usmjernik poruka NMR ključ je za integraciju među komponentama. On osigurava funkcionalnost poznatiju pod imenom medijacijska razmjena poruka (engl. *Mediated Message Exchange*). U tom se procesu poruke normaliziraju, te se najčešće razmjenjuju apstraktne poruke, kod kojih se korisni dio najčešće nalazi zapisan u obliku XML, a moguće je razmijeniti i metapodatke, odnosno podatke o samim porukama, poput svojstava poruka (engl. *Properties*). Sustav omogućuje i kreiranje predložaka razmjene poruka (engl. *Message Exchange Patterns*), koje podržavaju jednostavne komunikacijske primitive (poput *Get*, *Send*, *Delete* i sl.). Slika prikazuje NMR Message API.

²⁹⁰ Preuzeto sa <http://kalali.me/introducing-opensb-from-development-to-administration-and-management/>



Slika 6.13: NRM Message API

7. Poglavlje: Usluge Weba (Web Services)

U ovom poglavlju naučit ćete:

- ✓ temeljne pojmove
- ✓ arhitekturu usluga Weba
- ✓ SOAP
- ✓ WSDL
- ✓ REST
- ✓ usluge Weba druge generacije

7.1 Uvod u usluge Web

Prethodno se poglavlje bavilo konceptima sustava čija je arhitektura zasnovana na uslugama, na jednoj prilično konceptualnoj razini. U ovom ćemo poglavlju potanje obraditi usluge Web (engl. *Web Services*), koje su trenutno najrašireniji oblik uslužno orijentiranih sustava.

Definirajmo na početku usluge Web prema World Wide Web konzorciju²⁹¹:

„**Usluga Web** je programski sustav dizajniran na način da podržava interoperabilne (pod ovime podrazumijevamo odnos stroj prema stroju) interakcije putem mreže. Ona sadrži sučelja koja su opisana u obliku koji je moguće obraditi u računalu (prvenstveno WSDL). Drugi sustavi mogu komunicirati sa uslugom Web na način koji je opisan u njenom opisu korištenjem SOAP poruka, najčešće korištenjem protokola HTTP sa XML serijalizacijom u suradnji sa ostalim Web- orijentiranim normama.“²⁹²

Iščitavanjem ove službene definicije možemo odmah otkloniti nekoliko zabluda koje se obično pojavljuju kada se govori o uslugama Web. Prvo, vidimo da one nisu namijenjene kako bi ih koristili krajnji korisnici (ako želite stvarati vaš program za to, tada bismo trebali govoriti o aplikacijama Web), već je njihova namjena omogućiti korištenje njihovih podataka i operacija drugim programima. Ti drugi programi mogu biti ostale usluge Web, Web ili obične aplikacije. Drugo, vidimo da je ova definicija, premda i dalje općenita, ipak nešto konkretnija od definicija SOA sustava. Zbog toga usluge Web nisu SOA u punom smislu riječi, iako ih često poistovjećujemo. SOA sustav je mnogo širi pojam, dok usluge Web obuhvaćaju samo jedan manji dio jednog takvog sustava.

Kako ova definicija ostavlja uslugu Web prilično apstraktnom, norma definira njenu konkretnu implementaciju kao softverski (ili rijetko hardverski) sustav koji može primati i slati poruke. Takav se program prema specifikaciji naziva agentom. Usluga je pritom resurs koji je opisan apstraktnim skupom funkcionalnosti koje je u mogućnosti pružiti. Ovo zapravo znači da usluga ostaje ista bez obzira na to koji se agent koristi kako bi joj se moglo pristupiti (pritom je čak moguće mijenjati i programski jezik u kojem je agent izveden).

Kod usluga Web i uloge sudionika u njenom izvođenju (Slika 6.5) nešto su konkretnije nego kod SOA-e. Usluga Web izvodi svoju funkcionalnost za svog vlasnika (to može biti osoba ili tvrtka). Tako se *pružateljem* (engl. *service provider*) usluge u kontekstu usluga Web naziva upravo vlasnik usluge Web, točnije onaj koji je implementirao uslugu putem nekog agenta. S druge strane, *zahtjevatelj* (engl. *service requester*) može biti osoba ili tvrtka koja želi koristiti uslugu Web koju je ponudio pružatelj usluge. Zahtjevatelj usluge koristi vlastitog agenta koji će moći komunicirati s agentom pružatelja usluge. Upravo ovakav način komunikacije svojstven je za usluge Web, jer nije moguće da korisnik izravno pristupa nekoj od ponuđenih funkcionalnosti, već to može činiti isključivo putem svojeg (najčešće programskog) agenta.

Kako bi agenti znali oblik u kojem se međusobne poruke trebaju izmjenjivati, norma definira i način opisivanja usluge (WSD, engl. *Web Service Description*). WSD mora biti računalno

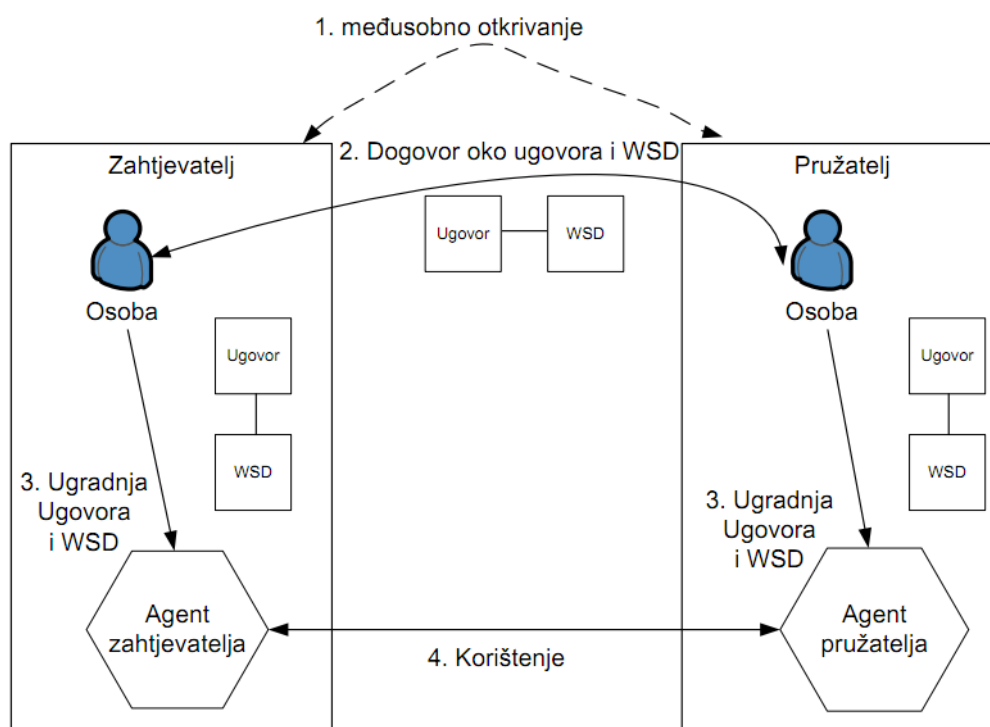
²⁹¹ World Wide Web Consortium, W3C, <http://www.w3.org/>

²⁹² W3C Working Group Note, Web Service Architecture, <http://www.w3.org/TR/ws-arch/>

obrađiva pa je definiran i zaseban jezik za opis usluga (WSDL, engl. *Web Service Description Language*). Potanje o WSDL-u govorit ćemo malo kasnije.

Kod usluga Weba također postoji ugovor o njihovom korištenju, sklopljen između zahtjevatelja i pružatelja usluge. Naravno, formalizam kod usluga Weba je umanjnjen u odnosu na formu ugovora u slučaju SOA-a, koja je prilično striktna. Ugovor za korištenje usluga Weba precizira dogovor dviju strana uključenih u korištenje usluge te način međusobnog komuniciranja njihovih agenata. Ovdje sam ugovor ne mora nužno biti u pisanoj formi ili posebno dogovoren. Ugovor može biti eksplicitan ili implicitan, u usmenom ili pisanom obliku, podoban za obradu na računalu ili napisan da bude razumljiv ljudima, a može biti legalno obvezujući ili samo informativan. Kako bi obje strane mogle u cijelosti razumjeti svoja prava i obveze koje proizlaze iz korištenja usluge, iznimno je važno da semantika koja se koristi prilikom sklapanja ugovora bude dobro definirana.

Slika 7.1 načelno prikazuje proces na koji zahtjevatelj počinje koristiti uslugu koju je ponudio pružatelj.

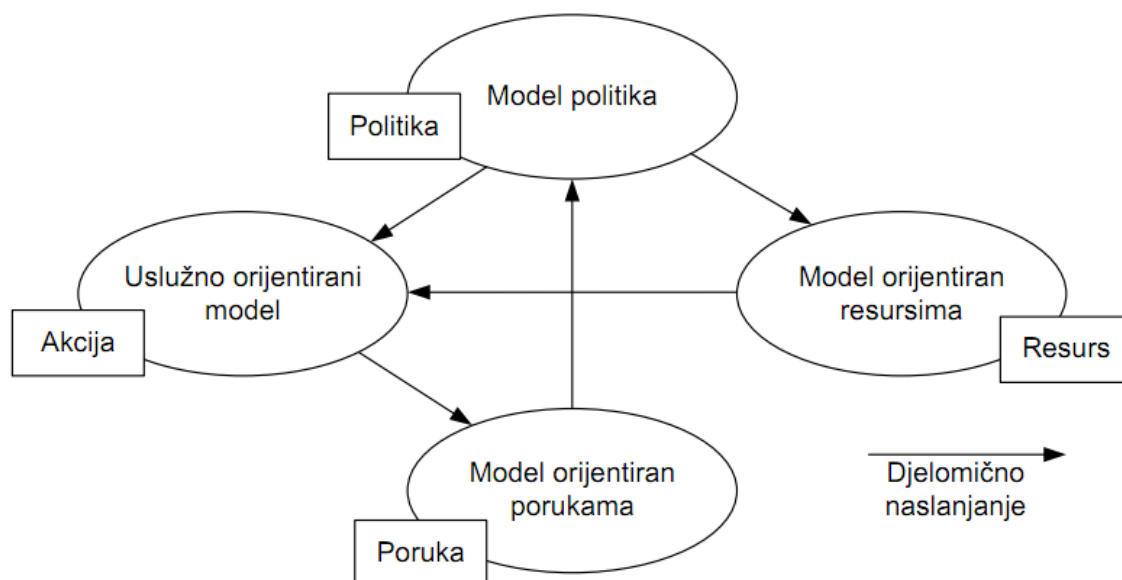


Slika 7.1: Model procesa korištenja usluge Weba

Prvi korak koji se mora obaviti kako bi se usluga mogla početi koristiti je međusobno otkrivanje sudionika u procesu. Norma usluga Weba olakšava ovo jer dopušta da samo jedna strana bude poznata drugoj (najčešće zahtjevatelj treba otkriti pružatelja). U sljedećem koraku treba pronaći neki način dogovora oko opisa sučelja usluge i ugovora za njeno korištenje kako bi se usuglasio način komunikacije među agentima zahtjevatelja i pružatelja usluge. U trećem je koraku potrebno ove usuglašene uvjete programski ugraditi u svaki od agenata. Naposljetku, u četvrtom koraku agenti mogu početi međusobno izmjenjivati poruke, tj. zahtjevatelj može početi koristiti uslugu. Neki od ovih koraka mogu se u potpunosti automatizirati, dok se neki (poput implementacije agenta) moraju provesti ručno.

7.2 Arhitektura usluga Web

Norma definira arhitekturu usluga Web kao metamodel koji se sastoji od 4 cjeline, kao što prikazuje Slika 7.2.



Slika 7.2: Metamodel arhitekture usluga Web

Model arhitekture usluga Web podijeljen je na četiri dijela, od kojih svaki definira pojedini pogled na cjelokupnu arhitekturu. Tako se model politika oslanja ponajprije na politike koje se koriste u sustavu (politike su zapravo previla igre). Uslužno orijentirani model bavi se funkcionalnostima koje sustav pruža, definirajući akcije koje se izvode u sustavu. Model orijentiran porukama dominantno se bavi porukama, njihovim formatom i strukturom, dok se model orijentiran resursima usredotočuje na resurse koji su potrebni ili nastaju tijekom rada samog sustava. U nastavku ćemo nešto opširnije predstaviti svaki od ovih četiriju modela. Napomenimo da ovdje donosimo pojednostavljeni pregled, dok zainteresirani čitatelj detaljnije opise ovih modela može pronaći u samoj normi²⁹³.

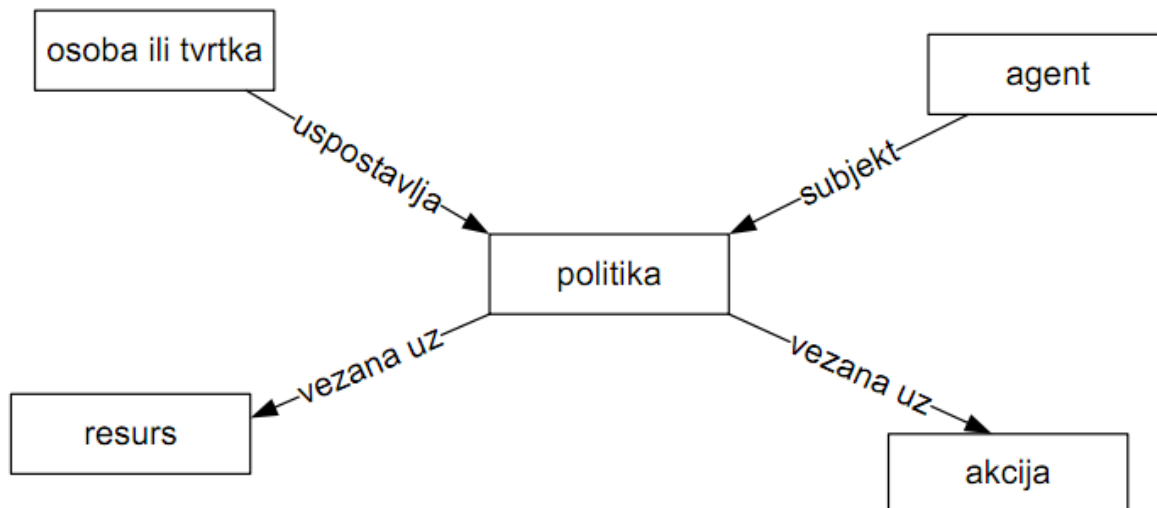
7.2.1 Model politika

Model politika (engl. *Policy Model*) bavi se ograničenjima u ponašanju agenata i usluga. Najšire gledano, politiku možemo primijeniti na resurse koje koristimo u našem sustavu. Pod resursima podrazumijevamo dokumente (primjerice opise usluga) kao i uobičajene računalne resurse. Naravno, dobro uspostavljena politika definirat će i model sigurnosti kao i uspostaviti kriterij za kvalitetu usluge.

²⁹³ W3C Working Group Note, Web Service Architecture, <http://www.w3.org/TR/ws-arch/>

Sukladno tome, model politika djelomično se oslanja (što u programerskom kontekstu možemo poistovjetiti sa korištenjem) na uslužno orijentiran model i model orijentiran resursima, dok se na njega oslanja model orijentiran porukama, što je prikazano na slici 3.

Slika 7.3 prikazuje pojednostavljeni model politika.



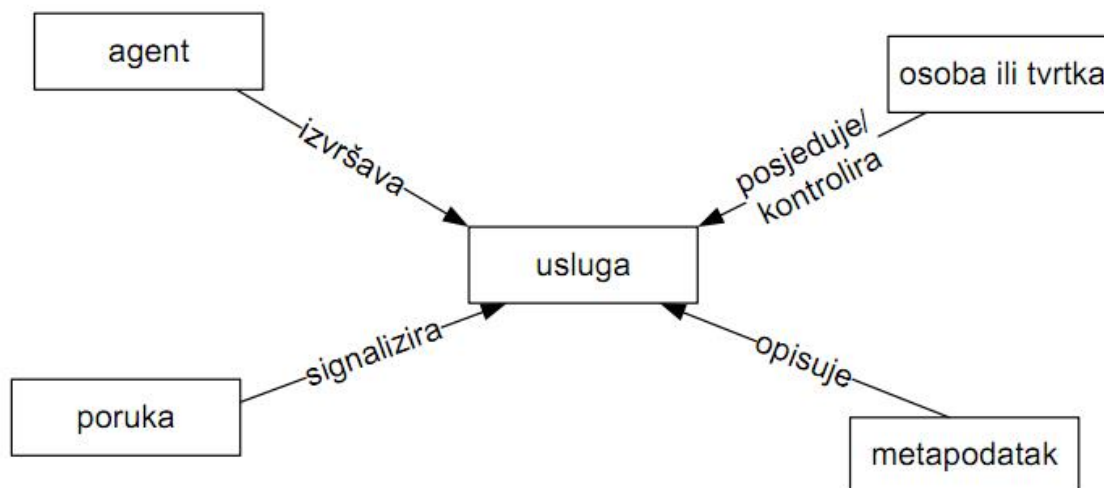
Slika 7.3: Pojednostavljeni model politika

Prema ovom modelu u središtu modela je politika. Osoba ili tvrtka, jednostavnije rečeno pružatelj usluge, mora definirati samu politiku, područje koje ona pokriva sa svim parametrima koje želi definirati u njoj, a koji su na raspolaganju u proširenom modelu (što uključuje dozvole za korištenje usluge i nadzor nad njenim korištenjem). Agent je definiran kao subjekt same politike, a politika je vezana uz resurs i konkretnu akciju koja se obavlja izvođenjem usluge.

7.2.2 Uslužno orijentirani model

Kako se može zaključiti već prema njegovom imenu, uslužno orijentirani model usluga Weba (engl. *Service Oriented Model*) dominantno se bavi samom uslugom kao i akcijama na kojima se temelji njena funkcionalnost. Kako je ovdje riječ o raspodijeljenom sustavu, usluge ne mogu biti dobro izvedene ako se ne oslanjaju na neki oblik razmjene poruka. No obrnuta tvrdnja ne vrijedi, poruke se ne moraju vezati uz usluge.

Donja slika donosi pojednostavljeni uslužno orijentirani model usluga Weba.



Slika 7.4: Pojednostavljeni uslužno orijentirani model

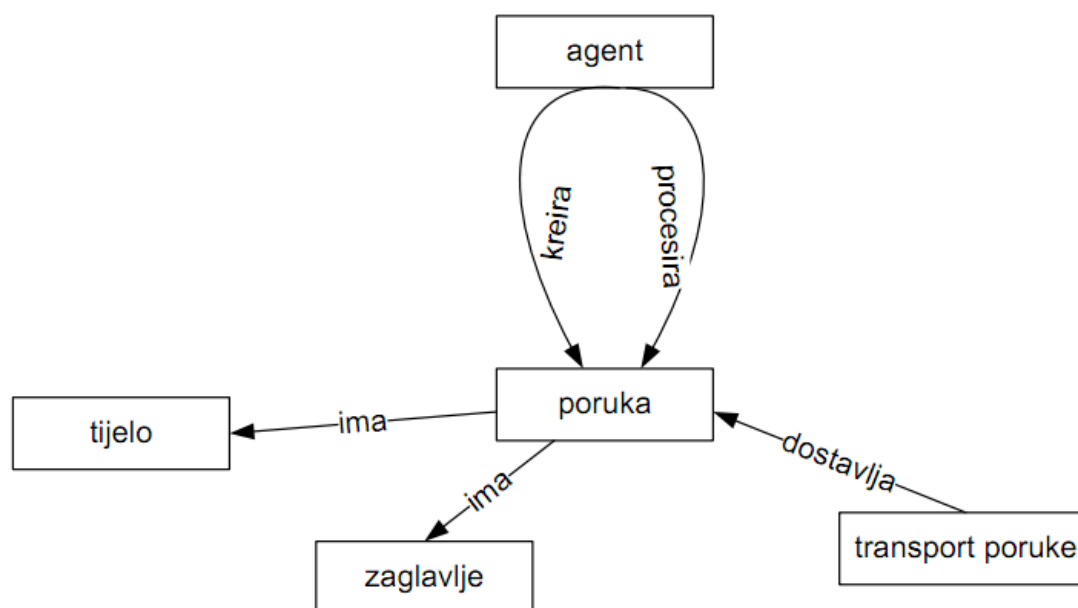
Usluga je realizirana, a i izvršava se putem agenta (u jednom slučaju to je agent pružatelja, a u drugom agent zahtjevatelja usluge). Ta dva agenta međusobno komuniciraju razmjenjujući poruke. Ovaj model definira i tko posjeduje uslugu (što može biti vrlo važno za pravnu stranu korištenja same usluge). Zadnji dio pojednostavljenog modela čine metapodaci, koji se koriste kako bi se opisali razni aspekti usluge, uključujući podatke o sučeljima i eventualnim ograničenjima u korištenju usluge.

Cjelovit uslužno orijentirani model najsloženiji je od svih četiriju modela, koji su dio metamodela arhitekture usluga Web. Djelomično se oslanja na model orijentiran porukama, dok se na njega oslanjaju modeli politika i model orijentiran resursima. On se također dotiče sigurnosnih pitanja korištenja usluga Web, ali i opisa i načina korištenja usluga, njihovog slaganja i funkcionalnosti koje osiguravaju.

7.2.3 Model orijentiran porukama

Model orijentiran porukama (engl. *Message Oriented Model*) u cijelosti se bavi porukama, njihovom strukturom i prijenosom poruka, ali ne ulazi u razloge nastajanja poruka u sustavu niti raspravlja o njihovom značenju. Poruke su definirane kao sredstvo pomoću kojeg agenti međusobno razmjenjuju informacije koje su im potrebne za izvršavanje (ili korištenje) usluge.

Donja slika prikazuje pojednostavljeni model orijentiran porukama.



Slika 7.5: Pojednostavljeni model orijentiran porukama

Agent je odgovoran za nastajanje, primanje i obradu primljene poruke. Model definira strukturu poruke, koja se sastoji od zaglavlja i tijela (u proširenom modelu definira se i omotnica). U konačnici, pojednostavljeni model specificira mehanizme koji su na raspolaganju kako bi se poruka dostavila na odredište.

Jedna od velikih prepreka kod ovog modela je što je, sa sigurnosnog stajališta, idealan kako bi se izvršio jedan od poznatijih sigurnosnih napada, „čovjek u sredini“ (engl. *Man-In-The-Middle Attack*). Zato je iznimno važno kod implementacije ovog modela u vlastiti sustav usluga Weba paziti na ispravne procedure u razmjeni poruka, a nije naodmet razmisliti o dodatnim sigurnosnim zaštitama kod njihove razmjene.

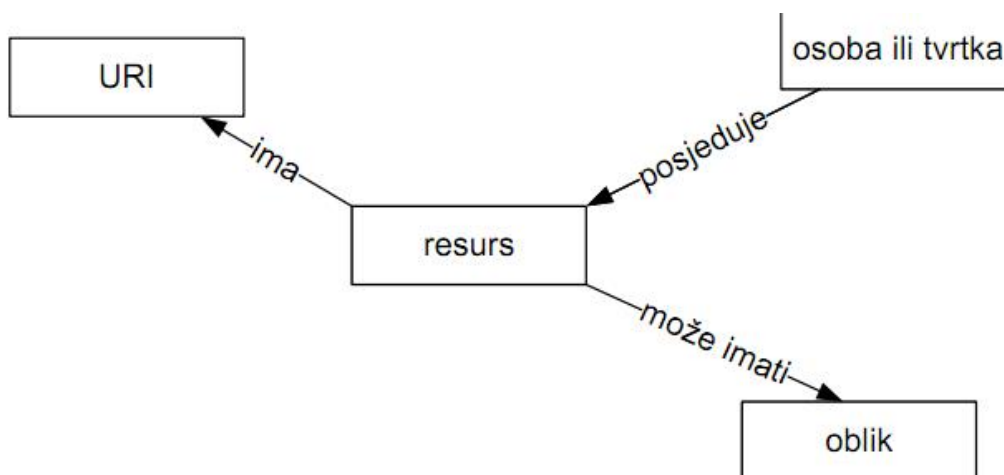
Model orijentiran porukama djelomice se oslanja na model politika, dok se na njega oslanja uslužno orijentirani model.

7.2.4 Model orijentiran resursima

Za razliku od uslužno orijentiranog modela, model koji je orijentiran resursima (engl. *Resource Oriented Model*) definira način korištenja resursa kojima raspolaže usluga (ili čak i njen vlasnik koji neke resurse može dati usluzi na korištenje). Ovdje podrazumijevamo da je resurs definiran kao bilo kakva element koji se može identificirati identifikatorom²⁹⁴.

Slika 7.6 prikazuje pojednostavljeni model orijentiran resursima.

²⁹⁴ Berners-Lee, T., Fielding, R., Masinter, L., Uniform Resource Identifiers (URI): Generic Syntax, IETF RFC 2396, August 1998, <http://ietf.org/rfc/rfc2396.txt>



Slika 7.6: Pojednostavljeni model orijentiran resursima

Ovaj je model vrlo važan za pojašnjavanje samog Weba te njegovog odnosa s uslugama Weba (primijetimo da je ovaj model vrlo sličan modelu sjedišta Weba neke tvrtke ili osobe). Ovaj model za usluge Weba izvodi istu stvar koju usluga DNS izvodi za internet. Važno je naglasiti da iako su resursi neovisan koncept, upravljanje resursima zapravo je instanca uslužnog modela – kada jasno definiramo o kojim je uslugama i konkretnim akcijama riječ.

Korištenjem ovog modela omogućuje se i samo korištenje usluga Weba. On daje svojevrsni identitet uslugama Weba, jer usluga za koju nitko ne zna da postoji, nitko ne zna što radi, nitko ne zna kako bi je pronašao i identificirao, zapravo je beskorisna.

Ovaj se model oslanja na uslužno orijentirani model, dok se na njega djelomice oslanja model politika.

7.2.5 Umjesto zaključka

Kao što se vidi, čak je i smanjenje specifikacija sustava usluga Weba još uvijek vrlo složeno (osobito ako se gleda prošireni model) i zahtjevno za dizajnera i programera takvog sustava. Na temelju ovih načela različiti su proizvođači softvera izveli svoja rješenja koja, nažalost, nisu u potpunosti kompatibilna te zahtijevaju savladavanje različitih programskih tehnologija i alata.

Nastavak ovog priručnika, a osobito konkretni primjeri koji će biti izneseni u njemu, bavit će se primjenama usluga Weba u programskom jeziku Java (čest naziv za takve usluge Weba je *Java Web Services* – JWS). JWS je skup alata, a ne konkretan okvir za programiranje aplikacija (engl. *framework*). U ovom ćemo poglavlju spomenuti i neke od alata koji će nam biti potrebni za izvođenje primjera navedenih u priručniku. Zbog ograničenog prostora, nećemo se detaljnije baviti sintaksom jezika Java, već pretpostavljamo da je čitatelj upoznat s njom barem do srednje razine (što uključuje i osnove Java EE platforme).

7.3 Simple Object Access Protocol – SOAP

U različitim smo prigodama u dosadašnjem tekstu ovog priručnika spominjali korištenje protokola SOAP tijekom komunikacije s uslugama Weba. Vrijeme je da pobliže definiramo ovaj protokol, kao i način njenog korištenja.

Simple Object Access Protocol – SOAP je komunikacijski protokol zasnovan na jeziku XML, koji služi za izmjenu informacija među entitetima na mreži²⁹⁵. Kao transportni protokol najčešće se koristi protokol HTTP. Informacija se prenosi putem SOAP poruka u kojima je u XML-u zapisana tekstualna informacija koju jedan entitet želi prenijeti drugome. Iz navedenog proizlazi da se SOAP uvelike oslanja na XML i njegove norme, poput *XML Scheme* i *XML Namespace* (detaljnije možete pročitati u poglavlju vezanom za XML²⁹⁶).

Prvotna specifikacija protokola SOAP nastala je 2000. godine na poticaj velikih softverskih tvrtki (uključivši HP, IBM, Microsoft i SAP) s ciljem unaprjeđivanja komunikacije među aplikacijama koje su s poslužiteljem komunicirale putem interneta preko protokola HTTP. Trenutno je važeća inačica 1.2 SOAP norme. Razlike među te dvije normama uključuju mogućnost korištenja *XSD Scheme* i korištenje ostalih protokola za transport (nije ograničeno na korištenje isključivo protokola HTTP) u inačici 1.2.

Kako je sama norma nastala u suradnji različitih kompanija, a oslonjen je na standardiziranja rješenja poput HTTP-a i XML-a, protokol je neovisan o operacijskim sustavima i tehnologijama u kojima je sama aplikacija izvedena.

7.3.1 SOAP poruke

Pogledajmo što sve mora sadržavati SOAP poruka koju želimo poslati nekoj drugoj aplikaciji.

Već smo rekli da se u SOAP poruci nalaze informacije koje su kodirane u jeziku XML. Podsjetimo li se što znamo o XML-u (već smo nešto o tome rekli u ovom priručniku, a i u nekim drugim predmetima tijekom studija), znamo da je istu informaciju XML-om moguće prikazati na više načina. Pogledajmo sljedeće zapise podataka o studentima u XML-u:

```
<student> Ivan Horvat 1234567</student>

<student>
  <ime>Ivan</ime>
  <prezime>Horvat</prezime>
  <mbr>1234567</mbr>
</student>

<student ime="Ivan" prezime="Horvat" mbr="1234567" />
```

Vidimo da su sva tri načina zapisa podataka u potpunosti valjana sa stajališta XML-a. Prisjetimo se da XML nema predefinirano značenje te je programeru ostavljeno da se u cijelosti brine o sintaksi koju će koristiti za označavanje elemenata koji su mu potrebni u njegovoj

²⁹⁵ SOAP standard, <http://www.w3c.org/TR/soap/>

²⁹⁶ Vidi poglavlje 3.3.4 i 3.4.1

aplikaciji. Korištenje SOAP poruka osigurava razumijevanje prenesenih informacija od jedne aplikacije prema drugoj.

SOAP poruka je (izvana gledano) običan XML dokument koji se sastoji od sljedećih dijelova:

- omotnice, pomoću koje definiramo da je XML dokument zapravo SOAP poruka,
- zaglavlja, u kojem se definira na koji će način informacije iz poruke biti procesirane (ovaj element nije obavezan),
- tijela poruke, gdje se nalazi sama poruka koja se prenosi,
- pogreške, koja sadržava informaciju o pogrešci koja se mogla dogoditi tijekom prijenosa i obrade informacije (ovaj element nije obavezan).

Svi su ovi elementi precizno definirani u osnovnim prostorima imena za SOAP omotnicu te SOAP kodiranje i tipove podataka^{297, 298}. Sljedeća slika prikazuje kako je poruka ućahurena u elemente SOAP poruke.



Slika 7.7: Struktura poruke SOAP

Vidimo kako za postizanje željene funkcionalnosti na korisno tijelo poruke moramo dodati velik dio popratnih informacija, koje će biti iskorištene kako bi se razumio korisni dio informacije koji se prenosi. Taj dio nazivamo viškom prenesene informacije (engl. *overhead*). Neobvezni dijelovi poruke (SOAP zaglavlje i pogreška) na slici su označeni sivom bojom.

Omotnica poruke označava se pomoću `xmlns:soap`, koji uvijek ima vrijednosti `http://www.w3.org/2001/12/soap-envelope/`, čime je poruka koju sadržava definirana kao poruka protokola SOAP. Već prema izgledu ovog elementa moguće je zaključiti koja se inačica protokola SOAP koristi (gornji se imenički prostor koristi za SOAP 1.2). Ako se u poruci želi koristiti SOAP 1.1, tada će u navedenoj oznaci kao prostor imena biti

²⁹⁷ SOAP Envelope, <http://www.w3.org/2001/12/soap-envelope>

²⁹⁸ SOAP Encoding, <http://www.w3.org/2001/12/soap-encoding>

navedeno `http://schemas.xmlsoap.org/soap/envelope/`. Također, svaka SOAP omotnica može sadržavati točno jedno SOAP tijelo (koje može biti dalje granano u skladu s potrebama u komunikaciji). Ako se koristi SOAP zaglavlje (također je, sukladno specifikaciji, dopuštena pojava samo jednoga ovakvog elementa), ono se u kodu XML-a nalazi neposredno ispod omotnice.

U zaglavlju se obično navode informacije o svrsi poruke, identifikatoru prijenosa kao i informacije o pošiljatelju i primatelju poruke. Informacija koja se nalazi u tijelu poruke za krajnjeg korisnika kodirana je u skladu s potrebama aplikacije kojoj je namijenjena. Ako je potrebno, moguće je koristiti prethodno definirani vlastiti imenički prostor. Prostor namijenjen pogreškama može sadržavati neku od četiri predefinirane pogreške:

- `VersionMismatch` – je li riječ o pogrešnoj inačici protokola koji se koristi,
- `MustUnderstand` – poruka nije razumljiva,
- `Server` – dogodila se neka pogreška na poslužitelju,
- `Client` – pojavio se neki problem sa samom porukom)²⁹⁹.

Da bi poruka udovoljavala definiciji poruke protokola SOAP, ona mora udovoljavati sljedećim jednostavnim pravilima. Pored činjenice da se za njeno kodiranje mora koristiti jezik XML, kao i odgovarajući imenički prostori, takva poruka ne smije sadržavati DTD referencu niti naredbe za procesiranje XML dokumenata.

Pogledajmo u nastavku na jednostavnom primjeru komunikaciju dvaju Javinih programa putem SOAP poruka^{300, 301}.

Na poslužiteljskoj se strani nalazi jednostavna „Hello, SOAP“ usluga (klasa `HelloWorlsSOAPService`) koja se sastoji od jedne jedine metode (`sayHello()`), koja od klijentske strane (naravno, putem protokola SOAP i poruke) prima ime korisnika i vraća korisniku personaliziranu pozdravnu poruku. Ovdje nećemo ulaziti u mehanizme obrade poruke na samom poslužitelju.

Ovako izgleda kod klase `HelloWorlsSOAPService`:

```
package samples.HelloWorld;

public class HelloWorldSOAPService
{
    public String sayHello(String firstName, String lastName) throws Exception
    {
        String aString = firstName + " " + lastName + ", ovako izgleda jedan
jednostavan primjer SOAP poruke koju je poslala usluga!";
        return aString;
    }
}
```

²⁹⁹ Snell, J., Tidwell, D., Kulchenko, P., Programming Web services with SOAP, O'Reilly Media, Inc., 2002.

³⁰⁰ Cerami, E., Web Services Essentials, O'Reilly Media, Inc., 2002.

³⁰¹ Yan, Y., Web Services, The Eighth International Conference on Electronic Commerce, August 14-16, 2006., Fredericton, New Brunswick, Canada

```

    public String addString(String symbol, String dataType) throws Exception
    {
        String aString = symbol + dataType;
        return aString;
    }
}

```

Vidimo da ova klasa sadržava dvije metode. Prva metoda kako parametre prima dva String objekta, a može baciti iznimku³⁰². Kako je sama metoda tipa String, ona će onome tko je pozove vratiti String objekt. Metoda će povratni objekt kreirati korištenjem podataka koji su prispjeli kod poziva te metode, te će pomoću operacije konkatencije s dodatnim tekstom stvoriti konačni oblik povratnog objekta. Zadnja će naredba vratiti rezultat onome tko je pozvao metodu. Pokažimo u nastavku kako izgleda klijentska strana:

```

public class TestClient
{
    public static void main(String args[])
    {
        try
        {

            Options opts = new Options( args );
            args = opts.getRemainingArgs();
            Service service = new Service();
            Call call = (Call) service.createCall();
            call.setTargetEndpointAddress( new java.net.URL(opts.getURL()) );
            if( args[0].equals("1") )
                call.setOperationName( new QName("urn:HelloWorld",
                    "sayHello"));
            else

                call.setOperationName( new
                    QName("urn:HelloWorld", "addString"));
            call.addParameter( "p1", XMLType.XSD_STRING,
                ParameterMode.IN );
            call.addParameter( "p2", XMLType.XSD_STRING,
                ParameterMode.IN );
            call.setReturnType( XMLType.XSD_STRING );
            call.setUsername( opts.getUser() );
            call.setPassword( opts.getPassword() );
            String res = (String) call.invoke( new Object[]
                { args[1], args[2] } );
            System.out.println( "Return is: " + res );

        }

        catch( Exception e )
        {
            e.printStackTrace();
        }

    }
}

```

³⁰² Zbog jednostavnosti ovdje nije naveden dio koda koji obrađuje iznimku – više o iznimkama možete pogledati u predmetima koji se bave programskim jezikom Java

Vidimo da je klijentska aplikacija nešto složenija, stoga objasnimo njene najbitnije dijelove. Na početku *try-catch* bloka učitavaju se parametri koji su predani pri pokretanju samog programa. Instancira se novi objekt tipa `Service` (to je zapravo neka od klasa dostupnih u istome paketu), te se nad njome kreira poziv (koristi se metoda `createCall()`, koja je jedna od metoda unutra klase `Service`). Pozivom metode `getURL()` iz paketa `java.net.URL` postavljamo URL na kojem se nalazi tražena usluga. Nakon toga slijedi `if` petlja koja provjerava vrijednosti koje su unesene kao parametar prilikom pokretanja programa. Vidimo da je jedini definirani parametar broj 1. Dakle, ako je korisnik klijentski program pokrenuo pomoću naredbe:

```
java TestClient 1 Ivan Horvat
```

izvest će se prvi uvjet u `if` petlji te će se pozvati metoda `setOperationName()` s navedenim parametrima, odnosno odabrat će se da je primatelj naše poruke metoda `sayHello()` na poslužitelju. Ako se pak navede neki drugi parametar (umjesto broja 1), neovisno o tome što se navede izvest će se preostali dio `if` petlje (onaj nakon naredbe `else`) te će se izvesti procedura vezana za metodu `addString()` na poslužiteljskoj strani. Na kraju³⁰³ će se izvršiti poziv usluge kojoj će biti predani parametri `args[1]` i `args[2]` – to su zapravo ime i prezime korisnika koji je pokrenuo klijentski program. Ako se kod pozivanja programa navede još parametara (oni će biti smješteni u polje `args[]` na pozicije od pozicije 3 nadalje), s obzirom na to da se ne koriste u programu, oni će jednostavno biti zanemareni. Metoda `invoke()` zapravo će poslati SOAP poruku na poslužiteljsku stranu. Ta će poruka izgledati ovako:

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

<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

  <soapenv:Body>
    <ns1:sayHello
      soapenv:encodingStyle=http://schemas.xmlsoap.org/soap/encoding/
      xmlns:ns1="urn:HelloWorld">
      <p1 xsi:type="xsd:string">
        Ivan
      </p1>
      <p2 xsi:type="xsd:string">
        Horvat
      </p2>
    </ns1:sayHello>
  </soapenv:Body>
</soapenv:Envelope>
```

U ovoj poruci jasno vidimo dva odvojena dijela svake SOAP poruke, omotnicu i tijelo poruke. Vidimo da se u dijelu koji definira omotnicu nalazi element `xmlns:soapenv`, za koji smo rekli da je obavezan. Možemo također vidjeti da je ovdje riječ o inačici SOAP 1.1.

³⁰³ Nakon pozivanja ostalih dostupnih metoda – moguće je čak uvesti i sigurnost putem odabira korisničkog imena i lozinke

Odmah nakon početka tijela poruke, definiramo kojoj je metodi namijenjena poruka (`ns1:sayHello`), reference na shemu kodiranja koja se koristi te ime usluge kojoj je poruka namijenjena (`ns1="urn:HelloWorld"`). Sljedeća dva retka definiraju parametre koji se prenose u poruci. Specificirano je da su oni tipa `String`. Ovaj tip mora odgovarati onome što je definirano u opisu usluge u pripadajućem WSDL dokumentu. Vidimo da su njihove vrijednosti upravo one koje smo naveli pri pokretanju klijentskog programa. Nakon što se poruka primi na poslužitelju, SOAP poslužitelj će je obraditi te je, sukladno navedenim podacima u njenom tijelu, predati na daljnju obradu odgovarajućoj metodi usluge `HelloWorld`. Kako naša poruka ima dva parametra, a metoda `sayHello()` očekuje dva parametra, ona će, nakon izvođenja svog programskog odsječka, pripremiti `String` u kojem će pisati:

"Ivan Horvat, ovako izgleda jedan jednostavan primjer SOAP poruke koju je poslala usluga!"

Ovaj će se objekt vratiti SOAP poslužitelju, koji će je zapakirati natrag u SOAP poruku kako bi je dostavio klijentu. Povratna SOAP poruka izgledat će ovako:

```
<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

  <soapenv:Body>
    <ns1:sayHelloResponse
      soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
      xmlns:ns1="urn:HelloWorld">
      <ns1:sayHelloReturn xsi:type="soapenc:string"
        xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">
        Ivan Horvat, ovako izgleda jedan jednostavan primjer SOAP poruke koju
        je poslala usluga!
      </ns1:sayHelloReturn>
    </ns1:sayHelloResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

7.3.2 Prijenos SOAP poruka

Kao što se vidi, možda i najbolje iz navedenog primjera, protokol SOAP definira samo oblik i ostali kontekst poruke, dok njime nije specificirano kako će sama poruka biti prenesena od pošiljatelja prema primatelju. SOAP je zapravo vrlo fleksibilan prema prijenosnom protokolu koji programer aplikacija želi koristiti za sam proces prijenosa. Postoje implementacije protokola SOAP koje kao prijenosni protokol koriste HTTP, FTP, SMTP, POP3, pa čak i izravno putem protokola TCP ili protokola za trenutno poručivanje Jabber – XMPP³⁰⁴.

Kako se na internetu najveća količina prometa prenosi upravo protokolom HTTP, on je dominantan i kod prijenosa SOAP poruka. Sama specifikacija, iako ne zahtijeva eksplicitno korištenje tog protokola, ipak daje neke smjernice prenošenja poruka SOAP putem protokola

³⁰⁴ XMPP Protocol Stack, <http://xmpp.org/protocols/>

HTTP. Korištenje tog protokola zapravo je najprirodnije (osim u slučajevima nekih posebnih zahtjeva na strani same aplikacije) jer SOAP i HTTP u potpunosti odgovaraju jedan drugome jer su oboje dizajnirani kao protokoli zahtjev-odgovor (engl. *request-response protocol*). Pritom se poruka klijenta prema poslužitelju šalje putem HTTP zahtjeva (engl. *HTTP request*), dok se odgovor šalje putem HTTP odgovora (engl. *HTTP response*). Dakle, u prethodnom primjeru na XML dio poruke koju je klijentski program uputio prema poslužitelju treba dodati dio koji inače na svaki zahtjev dodaje protokol HTTP, pa će ta poruka izgledati ovako:

```
POST /HelloWorld HTTP1.1

Content-Type: text/xml
Content-Lenght: nnnn

SOAPAction: urn:HelloWorld
<soapenv:Envelope
xmlns:soapenv=http://schemas.xmlsoap.org/soap/envelope/>
. . .
</soapenv:Envelope>
```

Sličan će dodatak dobiti i poruka odgovora, s razlikom što će na početku umjesto ključne riječi

```
HTTP/1.1 200 OK
. . .
<soapenv:Envelope
xmlns:soapenv=http://schemas.xmlsoap.org/soap/envelope/>
. . .
</soapenv:Envelope>
```

Vidimo da se ovdje u HTTP zaglavlje dodaje atribut SOAPAction, koji je definiran u spomenutom dijelu u kojem se specificira korištenje protokola HTTP. On zapravo govori poslužitelju HTTP zahtjeva što treba napraviti prije nego što pokuša dekodirati dobivenu poruku. Taj se atribut također može koristiti kako bi poslužitelj prepoznao legitimne zahtjeve i poruke koje su došle uslugama za koje on zna, a odbacio one namijenjene nepoznatim uslugama.

Ovime zaključujemo dio priručnika koji se odnosi na protokol SOAP i poruke kojima komuniciramo među uslugama Weba. Napomenimo da je, kako bi komunikacija porukama SOAP radila u konkretnoj implementaciji sustava usluga Weba, potrebno također instalirati odgovarajuću programsku podršku na objema stranama koje sudjeluju u procesu komunikacije. Gotovo svi vodeći poslužitelji namijenjeni smještanju usluga Weba imaju (bilo već ugrađeno, bilo u obliku zasebnog modula koji treba posebno instalirati) dostupnu implementaciju protokola SOAP. Način instalacije i korištenja tog modula ovisi o tome koji ćemo poslužitelj koristiti, tako da se više informacija o konkretnim koracima koje treba provesti da bi se na poslužitelju osigurala podrška za protokol SOAP možemo pronaći u dokumentaciji samog poslužitelja.

7.4 Web Services Description Language

Web Services Description Language – WSDL opisni je jezik zasnovan na jeziku XML koji služi za opis usluga Weba. Softverske kompanije (Aruba, IBM, Microsoft) predložile su standard 2000. godine, a on je 2001. prihvaćen kao preporuka W3C-a³⁰⁵. U toj inačici, označenoj kao WSDL 1.1, slovo D bilo je skraćenica za riječ *Definition*. Godine 2007. kao norma je prihvaćena inačica WSDL 2.0³⁰⁶, koja se od prethodne dosta razlikovala, a najvažnije su razlike bile dodavanje više razine semantike u opisni jezik, pojednostavljenje poruka, ukidanje podrške za preopterećenje te preimenovanje pojedinih ključnih riječi. Tako su *PortTypes* preimenovani u sučelja (engl. *interfaces*) dok su *Ports* preimenovani u zaključne točke (engl. *endpoints*). Jednako tako, WSDL 2.0 za razliku od prethodne inačice može koristiti sve metode zahtjeva koje koristi HTTP, dok je inačica WSDL 1.1 bila ograničena samo na korištenje metoda GET i POST.

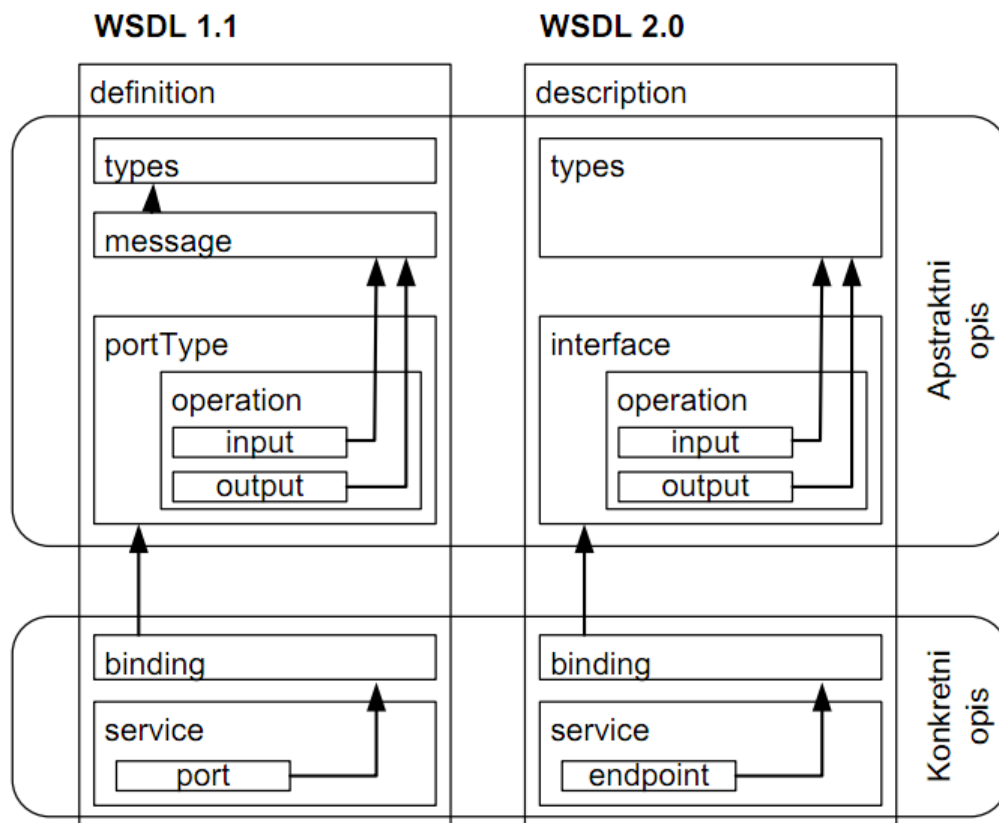
WSDL omogućuje opis usluga Weba, konkretnije pruža okvir za opis četiriju ključnih dijelova svake usluge. To su:

- sučelja takve usluge, kojima se definiraju sve javno dostupne funkcije koje je takva jedna funkcija u stanju pružiti svojim korisnicima,
- oblik podataka, koji se koristi u svim porukama koje takva jedna usluga prima i šalje,
- protokol, koji treba koristiti kako bi se pristupilo samoj usluzi,
- adresne informacije, kako bi se usluga mogla pronaći i početi koristiti.

Premda je već neko vrijeme važeći norma inačica WSDL 2.0, inačica WSDL 1.1 još se široko primjenjuje pa pokažimo stoga konkretno u čemu je razlika među opisima usluga pomoću ove dvije inačice norme. Slika 7.8 pokazuje kako izgleda opis usluge u WSDL 1.1 odnosno 2.0.

³⁰⁵ WSDL 1.1, <http://www.w3.org/TR/wsdl>

³⁰⁶ WSDL 2.0, <http://www.w3.org/TR/wsdl20-primer/>



Slika 7.8: Odnos opisa usluga Weba u WSDL 1.1 i WSDL 2.0

Kako vidimo, postoji šest dijelova koje moramo definirati prema normi WSDL da bismo opisali neku uslugu Weba. Promotrimo svaki od tih dijelova.

Svaki se WSDL dokument sastoji od dvaju dijelova, apstraktnog opisa i konkretnog opisa usluge. Apstraktni dio opisa usluge koristi se za općenitiji opis usluge, dok se u konkretnom dijelu opisa specificiraju dijelovi točno određeni upravo za tu uslugu.

Glavni je element svakog WSDL dokumenta označen ključnom riječju `description` (kod WSDL 1.1 taj je element `definition`). Pomoću njega definiramo koje je ime usluge Weba koja je opisana tim dokumentom, označujemo koje smo imeničke prostore koristili za opis funkcionalnosti usluge. Ovaj element zatvara sve ostale dijelove opisa usluge i mora se nalaziti na početku (engl. *root element*) svakog WSDL dokumenta.

Element `types` opisuje sve tipove podataka koji će se koristiti tijekom izvođenja opisane usluge Weba. Ovdje se navodi XML Schema koja će se koristiti unutar dokumenta. Ako se koristi standardna W3C XML Schema te u njoj definirani jednostavni tipovi podataka (`integer`, `String`), oni se podrazumijevaju te ih ne treba posebno definirati.

U normi WSDL 1.1 element `message` je izdvojen kao zasebna cjelina, dok ga u specifikaciji WSDL 2.0 nema, gdje se za poruke koristi XML Schema. Element `message` opisuje poruku, ne vodeći pritom brigu hoće li poruka biti poslana ili primljena. Definira se ime poruke te nijedan ili više dijelova poruke, u kojima se specificiraju parametri same poruke (ili očekivani segmenti odgovora na poruku).

Već smo rekli da je element `portType` iz specifikacije WSDL 1.1 preimenovan u `interface` u specifikaciji WSDL 2.0. Ovaj se element koristi kako bi se definirale operacije koje usluga može izvesti. Unutar jednog elementa moguće je definirati više operacija (zapravo onoliko koliko će ih usluga pružati svojim korisnicima). Svaka je od operacija definirana pomoću svojih ulaza i izlaza. Element `binding` opisuje na koji će konkretan način sama usluga biti izvedena, na koji će se način koristiti poruke SOAP i koji će se protokol koristiti za njihovo slanje.

Naposljetku, zadnji element koji definira specifikacije WSDL-a je element `service`. On definira adresu na kojoj se može pronaći tražena usluga. Najčešće je riječ o URL-u, gdje je dostupna njena funkcionalnost.

Pogledajmo na primjeru kako to konkretno izgleda, te objasnimo svaki od elemenata. U nastavku slijedi WSDL 2.0 dokument koji opisuje uslugu Web-a zamišljenog hotela koji je svojim gostima osigurao dvije funkcionalnosti, jednu za provjeru slobodnih mjesta u hotelu (`CheckAvailability`), dok preko druge mogu rezervirati sobu u hotelu za određeno razdoblje (`MakeReservation`).³⁰⁷

```
<?xml version="1.0" encoding="utf-8" ?>
<description
  xmlns="http://www.w3.org/ns/wsd1"
  targetNamespace= "http://greath.example.com/2004/wsd1/resSvc"
  xmlns:tns= "http://greath.example.com/2004/wsd1/resSvc"
  xmlns:ghns = "http://greath.example.com/2004/schemas/resSvc"
  xmlns:soap= "http://www.w3.org/ns/wsd1/soap"
  xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
  xmlns:wsd1x= "http://www.w3.org/ns/wsd1-extensions">

  <documentation>
    This document describes the Greath Web service. Additional
    application-level requirements for use of this service --
    beyond what WSDL 2.0 is able to describe -- are available
    at http://greath.example.com/2004/reservation-documentation.html
  </documentation>

  <types>
    <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
      targetNamespace="http://greath.example.com/2004/schemas/resSvc"
      xmlns="http://greath.example.com/2004/schemas/resSvc">

      <xs:element name="checkAvailability" type="tCheckAvailability"/>
      <xs:complexType name="tCheckAvailability">
        <xs:sequence>
          <xs:element name="checkInDate" type="xs:date"/>
          <xs:element name="checkOutDate" type="xs:date"/>
          <xs:element name="roomType" type="xs:string"/>
        </xs:sequence>
      </xs:complexType>

      <xs:element name="checkAvailabilityResponse" type="xs:double"/>

      <xs:element name="invalidDataError" type="xs:string"/>
    </xs:schema>
  </types>
</description>
```

³⁰⁷ Pierce, M. WSDL 2.0, Community Grids Lab, Indiana University

```

    </xs:schema>
  </types>

  <interface name = "reservationInterface" >

    <fault name = "invalidDataFault"
      element = "ghns:invalidDataError"/>

    <operation name="opCheckAvailability"
      pattern="http://www.w3.org/ns/wsd1/in-out"
      style="http://www.w3.org/ns/wsd1/style/iri"
      wsdlx:safe = "true">
      <input messageLabel="In"
        element="ghns:checkAvailability" />
      <output messageLabel="Out"
        element="ghns:checkAvailabilityResponse" />
      <outfault ref="tns:invalidDataFault" messageLabel="Out"/>
    </operation>

  </interface>

  <binding name="reservationSOAPBinding"
    interface="tns:reservationInterface"
    type="http://www.w3.org/ns/wsd1/soap"
    wsoap:protocol="http://www.w3.org/2003/05/soap/bindings/HTTP/">

    <fault ref="tns:invalidDataFault"
      wsoap:code="soap:Sender"/>

    <operation ref="tns:opCheckAvailability"
      wsoap:mep="http://www.w3.org/2003/05/soap/mep/soap-response"/>

  </binding>

  <service name="reservationService"
    interface="tns:reservationInterface">

    <endpoint name="reservationEndpoint"
      binding="tns:reservationSOAPBinding"
      address = "http://greath.example.com/2004/reservation"/>

  </service>

</description>

```

Promotrimo svaki pojedini element u ovom opisu usluge.

Kao i svaki dobro definirani WSDL dokument, tako i naš primjer započinje oznakom `<description>`. Unutar tog elementa treba definirati imenički prostor koji će se koristiti pri izradi WSDL dokumenta.

Kao prvi imenički prostor navodi se `xmlns=http://www.w3.org/ns/wsd1`, kojim smo zapravo odredili da koristimo imenički prostor kojim je definiran sam WSDL 2.0, odnosno da će svi elementi koje navodimo u dokumentu, a koji neće imati vlastiti prefiks, biti upravo elementi WSDL 2.0 specifikacije. U sljedećim recima definiramo dodatne imeničke prostore, poput ciljnog prostora (engl. *target namespace*). Njih ćemo kasnije koristiti kod pojedinih elemenata za detaljniji opis usluge. Napomenimo da bi ovi prostori imena trebali biti pod

izravnim nadzorom vlasnika usluge jer je važno da klijenti koji dohvaćaju opis usluge to čine dohvaćajući njene dijelove iz autoritativnog izvora.

Iz definicije same usluge možemo zaključiti kako će naša usluga slati i primiti nekakve poruke iz vanjskog svijeta, tako da je sljedeći element kojeg moramo definirati `types`. WSDL specifikacija dopušta kreiranje poruka na dva načina. Prvi, podrazumijevani način je kreirati ga kao dijete elementa `description`, a drugi je korištenjem mehanizma `import` dostupnog u XML Schemi. Svaka poruka (neovisno o vrsti poruke – normalna poruka ili poruka o pogrešci) definirana prema WSDL 2.0 normi mora biti definirana pomoću jednog element `taga` (naravno, ovisno o potrebama, njezina unutarnja struktura može biti po volji složena).

Sljedeći korak pri kreiranju opisa naše usluge je opisati njenu funkcionalnost. WSDL specifikacija omogućuje razdvajanje apstraktnih funkcionalnosti od konkretnih detalja na koji je način izvedena i kako je usluga pruža. Ovo je izravna posljedica potrebe da se usluga (i njen opis) može nanovo iskoristavati. Element `interface` koristi se za apstraktni opis usluge Weba, a sastoji se od više apstraktnih operacija, gdje svaka takva operacija predstavlja jednostavno sučelje između usluge i njenog klijenta. Ovdje se također opisuje i slijed poruka (engl. *pattern*) koje će biti razmjenjivane. Naziv koji koristimo za sučelje ne smije sadržavati razmak niti dvotočku.

Nakon što smo definirali operacije i poruke koje se razmjenjuju, moramo opisati način na koji će se to učiniti. Za taj se dio opisa koristi element `binding`. Pomoću njega ćemo točno opisati kako će izgledati poruka (njezin oblik) i koji će se protokol koristiti za prijenos. Pritom je takvu informaciju potrebno specificirati za svaku poruku (bilo normalnu, bilo poruku o pogrešci) koju smo naveli u opisu sučelja usluge Weba. Primjerice, za opis SOAP poruke napisati ćemo `type="http://www.w3.org/ns/wSDL/soap"`, dok ćemo, ako mislimo za njen prijenos koristiti protokol HTTP, morati dodati (`wsoap` se nalazi u gornjem popisu imeničkih prostora) `wsoap:protocol="http://www.w3.org/2003/05/soap/bindings/HTTP/"`.

Korištenjem elementa `service` definira se gdje je usluzi moguće pristupiti. Norma WSDL 2.0 omogućuje specificiranje jednog sučelja koje će usluga imati te liste završnih točaka (engl. *endpoints*) na kojima je moguće pristupiti usluzi, s informacijom o korištenim protokolima i prijenosnim oblicima za svaku točku. Naziv koje koristimo za uslugu kao i naziv završne točke moraju biti jedinstveni u korištenom imeničkom prostoru.

Pomoću elementa `address="http://greath.example.com/2004/reservation"/>` definiramo fizičku adresu na kojoj se usluga nalazi, te joj klijenti mogu pristupiti putem definiranog sučelja.

Kao što možemo vidjeti, WSDL opis neke usluge zapravo pruža vrlo ograničen uvid u samu uslugu. Ovdje smo definirali samo osnovni način korištenja usluge – naveli smo tipove poruka koje možemo koristiti u komunikaciji s ovom uslugom, koje protokole trebamo koristiti, kao i adresu gdje se sama usluga nalazi, no za potpuno razumijevanje načina funkcioniranja usluge treba ipak definirati dokumentaciju o usluzi u nešto većem obimu. Takva dokumentacija može uključivati svrhu i način korištenja same usluge, stvarno značenje svih poruka koje se koriste, ograničenja u radu usluge i slične informacije koje su potrebne korisniku kako bi razumio i ispravno koristio uslugu. WSDL 2.0 pomoću svog elementa `documentation` omogućuje kreiranje ljudima čitljivog opisa usluge. Ovaj se element često koristi kako bi se osigurala

poveznica na vanjski poslužitelj (najčešće neki od poslužitelja Weba), gdje se nalazi detaljna informacija koja može poslužiti razvijateljima klijentskih aplikacija kako bi mogli učinkovito koristiti uslugu.

Nakon što smo završili s opisom usluge (napravili WSDL dokument i svu potrebnu ostalu dokumentaciju za neometano korištenje naše usluge Weba), moramo na neki način zatvoriti ciklus (Slika 6.5 iz prethodnog poglavlja).

Da bi naša usluga dobila svoje prve korisnike, prvo njen opis moramo objaviti na registru (to može biti bilo kakva implementacija koja je javno dostupna – više potankosti o načinu objave pročitajte u prethodnom poglavlju). Na taj će način potencijalni klijenti moći pronaći našu uslugu te je na kraju, ako mu odgovaraju njezine funkcionalnosti, i početi koristiti (dok će vlasnik usluge to potencijalno moći naplatiti).

Naravno, postoje i neka ograničenja u korištenju WSDL-a, zbog čega se pojavljuju nove inicijative i predlaže daljnji razvoj specifikacije³⁰⁸. Iz dosadašnjeg teksta o ovom jeziku možemo zaključiti da iako on na vrlo jednostavan način precizno definira način korištenje i pristupa usluge, pomoću opisa neke usluge u WSDL obliku ne može se ništa reći o tome što usluga zapravo radi. Dakle, potencijalni korisnik na semantičkoj razini ne zna hoće li mu pronađena usluga odgovarati. Sve što korisnik putem pronađenog WSDL-a može odrediti je jesu li specificirane poruke i sučelja u skladu s onim što traži. Pritom je na programeru klijentske aplikacije da temeljem vlastitog iskustva i uzdanja u pružatelja usluge na osnovi primjerice imena sučelja ili usluga zaključi što pronađena usluga zapravo radi i kako je može iskoristiti za vlastite potrebe.

Stoga je skupina autora sa Sveučilišta u Georgiji uz suradnju sa zaposlenicima IBM-a World Wide Web konzorciju predložila prihvaćanje proširenja specifikacije WSDL, nazvanu Web Service Semantics – WSDL-S³⁰⁹. U njemu se predlaže da se za uslugu definira odgovarajući semantički model, koji bi se potom samo uključivao u opis usluge u WSDL-u (dakle, ne mijenja se struktura WSDL-a, već se samo koriste semantički opisi u obliku WSDL proširenja). Kao ontologija predlaže se korištenje jezika OWL-S³¹⁰. Na taj se način osigurava da semantička informacija koja je sadržana u opisu usluge dobro definira preduvjete za rad, ulaze i izlaze usluge Weba kao i učinke izvedbe usluge Weba. Zbog ovih bi svojstava korištenje WSDL-S-a trebalo na jednostavan način riješiti problem što bržeg (automatiziranog) razumijevanja o kojoj je funkcionalnosti riječ u pronađenoj usluzi Weba. Jednako tako, zbog minimalnih intervencija u normu, ni implementacija ovog proširenja u brojnim alatima ne bi smjela biti velik problem.

7.5 REST API

REST (engl. *Representational State Transfer*) predstavlja arhitekturni stil namijenjen dizajniranju mrežnih aplikacija. Osnovna ideja se temelji na korištenju postojećeg HTTP

³⁰⁸ Kalin, M., *Java Web Services, Up and Running*, O'Reilly Media, Inc., 2009

³⁰⁹ WSDL-S, <http://www.w3.org/Submission/WSDL-S/>

³¹⁰ OWL-S, <http://www.daml.org/services/owl-s/>

protokola i iskorištavanju metoda GET, POST, PUT i DELETE, pri čemu su pružene usluge nevjесne stanja (engl. *stateless*).

Od SOAP-a se razlikuju po tome što podržavaju XML i JSON, dok SOAP omogućava korištenje samo XML kao formata podataka. Omogućava vrlo lako spajanje raznih klijenata koji mogu biti mobilne aplikacije, klijentske aplikacije implementirane u tehnologijama poput Angular, ReactJS ili Vue.js.

URL-ovi kod REST API-a se fokusiraju na resurs, a ne na akciju. Na primjer, ako je potrebno obavljati neku akciju nad resursom iz skupine “student” i s identifikatorom “id=1”, URL za identifikaciju tog resursa bio bi:

<http://posluzitelj:8080/Aplikacija/student/1>.

Akcija koja se obavlja nad tim resursom nije ugrađena u sam URL, već je definirana HTTP zahtjevom i može biti GET, POST, PUT i DELETE te koristi detalje o samom entitetu koji su spremjeni u zaglavlju ili tijelu HTTP zahtjeva.

Na primjer, ako je potrebno razviti REST API sučelje za dohvat podataka o jelima, prvo je potrebno implementirati klasu (npr. U C#-u) koja predstavlja model podataka:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Runtime.Serialization;
using System.Web;

namespace REST.API.Models
{
    [DataContract]
    public class Jelo
    {
        [DataMember]
        public string Naziv { get; set; }
        [DataMember]
        public string Vrsta { get; set; }
        [DataMember]
        public int BrojKilokalorija { get; set; }
        [DataMember]
        public int Količina { get; set; }
        [DataMember]
        public string NacinPripreme { get; set; }
        [DataMember]
        public List<string> Namirnice { get; set; }
    }
}
```

Klasa “Jelo” sadržava podatke kao što su naziv, vrsta, broj kilokalorija, količina, način pripreme i lista namirnica. Radi jednostavnosti, kreirat će se klasa “JeloRepository” koja inicijalizira dva objekta klase “Jelo” i stavlja ih na listu:

```
using System.Collections.Generic;
namespace REST.API.Models
```

```
{
    public class JeloRepository
    {
        private List<Jelo> jela;

        public JeloRepository()
        {
            jela = new List<Jelo>()
            {
                new Jelo
                {
                    BrojKilokalorija = 123,
                    Količina = 1,
                    Namirnice = new List<string>()
                    {
                        "Meso",
                        "Rajčica",
                        "Tijesto"
                    },
                    Naziv = "Lazanje",
                    NacinPripreme = "Pečenje",
                    Vrsta = "Meso"
                },
                new Jelo
                {
                    BrojKilokalorija = 1234,
                    Količina = 3,
                    Namirnice = new List<string>()
                    {
                        "Meso",
                        "Rajčica",
                        "Tortilje",
                        "Paprika"
                    },
                    Naziv = "Tortilje",
                    NacinPripreme = "Pečenje",
                    Vrsta = "Meso"
                }
            };
        }

        public List<Jelo> GetJela()
        {
            return jela;
        }
    }
}
```

Sljedeći korak je implementacija klase REST API *controller* koja će implementirati REST metode za dohvat svih jela korištenjem URL-a “api/jela/”, što je postavljeno na razini klase te pomoću GET metode vraća sva spremljena jela. Dodavanjem parametra naziva ili iznosa

kilokalorija u URL obavlja se filtriranje podataka prema zadanim kriterijima. Korištenjem metode POST i dodatka URL-u "novo" kreira se novi zapis o jelu i dodaje u listu s ostalim zapisima.

```
using REST.API.Models;
using System.Collections.Generic;
using System.Linq;
using System.Web.Http;

namespace KarloFabijanić.API.Controllers
{
    [RoutePrefix("api/jela")]
    public class JelaController : ApiController
    {
        private readonly JeloRepository _jeloRepository;

        public JelaController()
        {
            _jeloRepository = new JeloRepository();
        }

        [HttpGet]
        [Route("")]
        public IEnumerable<Jelo> Jela()
        {
            return _jeloRepository.GetJela();
        }

        [HttpGet]
        [Route("{naziv}")]
        public List<Jelo> JelaByNaziv(string naziv)
        {
            return _jeloRepository.GetJela().Where(j=>j.Naziv == naziv).ToList();
        }

        [HttpPost]
        [Route("novo")]
        public IEnumerable<Jelo> Jela(Jelo jelo)
        {
            var jela = _jeloRepository.GetJela();
            jela.Add(jelo);

            return jela;
        }

        [HttpGet]
        [Route("kcal/{kcal}")]
        public IEnumerable<Jelo> JelaByKcal(int kcal)
        {
            return _jeloRepository.GetJela().Where(
                j => j.BrojKilokalorija < kcal).ToList();
        }
    }
}
```

```
}  
}
```

Klijentski dio sustava koji kroz upute iz naredbene linije poziva REST API servise prikazan je u nastavku:

```
using System;  
using System.Collections.Generic;  
using System.IO;  
using System.Linq;  
using System.Net;  
using System.Text;  
using System.Xml;  
using System.Xml.Serialization;  
  
namespace REST.Client  
{  
    class Program  
    {  
        private static readonly string path = "http://localhost:50000/api/jela";  
  
        static void Main(string[] args)  
        {  
            int option;  
            do  
            {  
  
                ShowMenu();  
                option = ReadOption();  
  
                switch (option)  
                {  
                    case 1:  
                        GetJela();  
                        break;  
                    case 2:  
                        GetJeloByNaziv();  
                        break;  
                    case 3:  
                        InsertJelo();  
                        break;  
                    case 4:  
                        GetJeloByKcal();  
                        break;  
                    default:  
                        break;  
                }  
  
                } while (option != 5);  
        }  
    }  
}
```

```
private static void GetJeloByKcal()
{
    Console.WriteLine("Upisite broj kalorija: ");
    int kcal = int.Parse(Console.ReadLine());

    GetDataFromApi($"{path}/kcal/{kcal}");
}

private static void InsertJelo()
{
    Console.WriteLine("Unesite naziv jela: ");
    string naziv = Console.ReadLine();

    Console.WriteLine("Unesite vrstu: ");
    string vrsta = Console.ReadLine();

    Console.WriteLine("Unesite broj kilokalorija: ");
    int brojKiloKalorija = int.Parse(Console.ReadLine());

    Console.WriteLine("Unesite količinu");
    int kolicina = int.Parse(Console.ReadLine());

    Console.WriteLine("Unesite način pripreme: ");
    string nacinPripreme = Console.ReadLine();

    Console.WriteLine("Unesite namirnice (odvojene zarezom) : ");
    List<string> namirnice = Console.ReadLine().Split(',').ToList();

    Jelo jelo = new Jelo(naziv, vrsta, brojKiloKalorija, kolicina,
                        nacinPripreme, namirnice);

    XmlSerializer serializer = new XmlSerializer(typeof(Jelo));
    StringWriter writer = new StringWriter();
    serializer.Serialize(writer, jelo);

    string xml = serializer.ToString();

    byte[] data = Encoding.UTF8.GetBytes(xml);
}

private static void GeJeloByNaziv()
{
    Console.WriteLine("Upisite naziv jela: ");
    string name = Console.ReadLine();

    GetDataFromApi($"{path}/{name}");
}

private static void GetJela()
{

```

```
        GetDataFromApi(path);
    }

    private static void GetDataFromApi(string path)
    {
        HttpWebRequest request = (HttpWebRequest)WebRequest.Create(path);
        request.ContentType = "application/xml";

        HttpWebResponse reponse = (HttpWebResponse)request.GetResponse();

        XmlDocument doc = new XmlDocument();
        doc.Load(reponse.GetResponseStream());
        doc.Save(Console.Out);

        DisplayContinueOption();
    }

    private static void DisplayContinueOption()
    {
        Console.WriteLine("\n\n");
        Console.WriteLine("Pritisnite enter za nastavak");
        string next = Console.ReadLine();
        Console.Clear();
    }

    private static int ReadOption()
    {
        Console.WriteLine("Unesite opciju: ");

        return int.Parse(Console.ReadLine());
    }

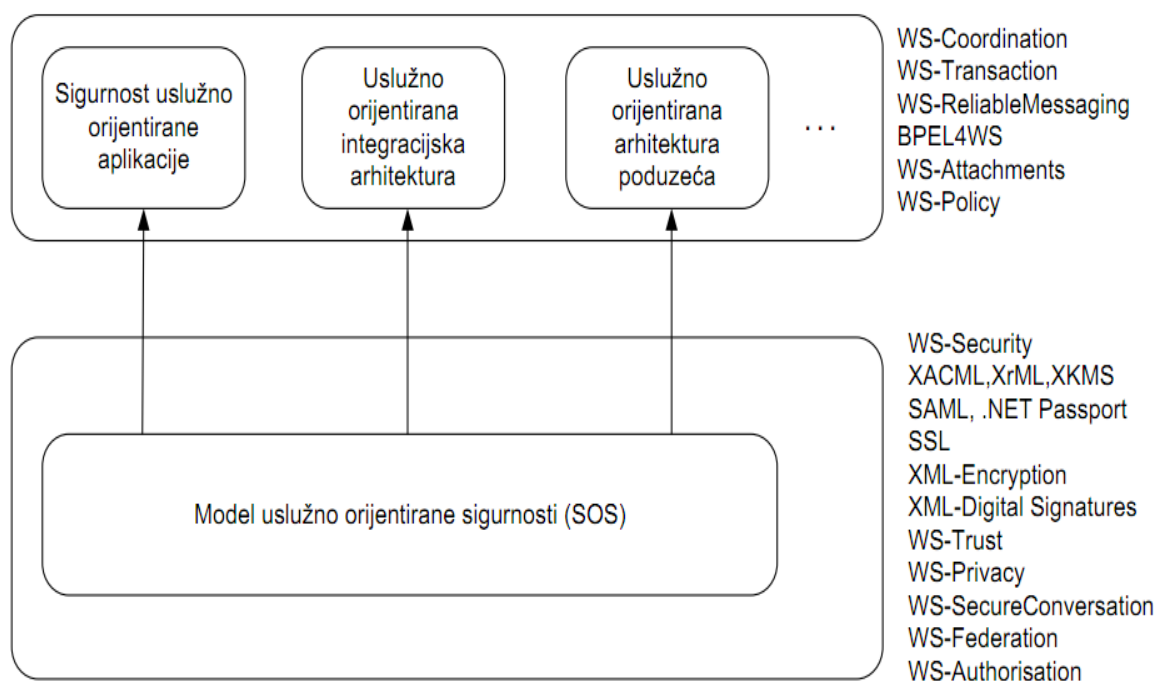
    private static void ShowMenu()
    {
        Console.WriteLine("MENU JELA");
        Console.WriteLine("-----");
        Console.WriteLine("Odaberite opciju: ");
        Console.WriteLine("1. Ispis svih jela");
        Console.WriteLine("2. Ispis jela prema nazivu");
        Console.WriteLine("3. Unos novog jela");
        Console.WriteLine("
            4. Ispis jela s brojem kilokalorija manjim od zadanog");
        Console.WriteLine("5. Prekid rada programa");
    }
}
}
```

7.6 Razvoj standarda usluga Weba

Slično kao i u slučaju WSDL-a, gdje je tijekom razdoblja njegova korištenja uočen manjak podrške u vezi sa semantičkim vrednovanjem opisa usluga Weba, tako je i tijekom razdoblja iznimne popularnosti i velikog porasta korištenja različitih usluga Weba uočen razmjerno velik prostor za različita poboljšanja. Prva generacija usluga Weba nije bila posve integrirana, te je bilo razmjerno teško uključiti druge tvrtke u njihovo korištenje.

7.6.1 Usluge Weba druge generacije

Tijekom vremena predložen je velik broj specifikacija usluga Weba druge generacije, koji su bili usmjereni na rješavanje pojedinih pitanja. Kako su u njihovom stvaranju često sudjelovali i veliki proizvođači softvera, koji su specifikaciju odmah ugradili u svoj alat, stvorena je početna baza korisnika, koja je s vremenom proširivana. Jedan od glavnih ciljeva uspostave druge generacije usluga Weba bila je stvaranje temelja za tzv. uslužno orijentirano poduzeće (engl. *service-oriented enterprise*)³¹¹. Slika 7.9 prikazuje kako se pojedine specifikacije odnose prema modelu jednog takvog poduzeća.



Slika 7.9. Specifikacije usluga Weba druge generacije

Model uslužno orijentiranog poduzeća koncentriran je naravno oko usluge koja je primaran generator nove vrijednosti koje to poduzeće stvara. On je logički podijeljen u dva sloja, od kojih je donji sloj primarno orijentiran prema sigurnosti, dok je onaj gornji namijenjen za integraciju s poslovnim sustavima poduzeća koji su partneri (koji nude usluge potrebne za uredan rad

³¹¹ Erl, T., Service-Oriented Architecture, A Field Guide to Integrating XML and Web Services, Prentice Hall, 2004

primarne usluge poduzeća). Sa desne se strane nalazi popis nekih specifikacija za usluge Weba druge generacije.

Specifikacije usluga Weba druge generacije u literaturi se često nazivaju WS-* (u izgovoru se koristi termin WS-Star). Njihova je osnovna namjena povećati stupanj automatizacije poslovnog procesa, što nije moguće (ili možda bolje, nije bilo pretjerano lako izvesti) korištenjem usluga Weba prve generacije. Jedan od pokretača razvoja specifikacija usluga Weba druge generacije dugo je vremena bila udruga Web Services Interoperability Organization (poznata po kratici WS-I)³¹², koja je sredinom 2010. združena s već spominjanom organizacijom OASIS. U nastavku ćemo ukratko opisati neke od njih, s naglascima probleme koje su pokušali riješiti.

Usluge Weba prve generacije nisu imale mogućnost zadržavanja informacije o stanju u kojem se usluga nalazila u nekom trenutku njezina izvođenja. Takve se aplikacije inače u programskom inženjerstvu nazivaju *stateless* aplikacijama. Ako je pak neka aplikacija svjesna stanja u kojem se nalazi u nekom trenutku, tada govorimo o *statefull* aplikacijama³¹³. Ova analogija vrijedi i za usluge Weba. Kako je praćenje stanja jedna od osnova za izvođenje raspodijeljenih transakcija, vidimo da specifikacije prve generacije usluga Weba nisu mogle podržati usluge koje su zahtijevale transakcijske mehanizme. Nije bilo moguće ni podržati jasan kontekst u kojem se usluga odvija. Kako bi se riješili opisani problemi, predložene su dvije nove specifikacije za usluge Weba druge generacije, nazvane WS-Coordination³¹⁴ i WS-Transaction³¹⁵.

Specifikacija WS-Coordination osigurava mehanizme pomoću kojih usluga može sačuvati kontekst u kojem se odvija neka njezina aktivnost. Kako bi se to provelo, uvodi se novi koordinacijski model za usluge koji se sastoji od jednostavnih usluga koje su odgovorne za kreiranje konteksta, odabir protokola te registraciju koordinatora kontekstom. Kako bi ostale usluge Weba mogle koristiti ovu funkcionalnost, osigurana je mogućnost korištenje nove definicije nazvane Coordination Type. Tu mogućnost koristi i specifikacija WS-Transaction, koja definira dva različita koordinacijska tipa, jedan za kratkotrajne, tzv. atomične transakcije, a drugi za dugotrajne, poslovne aktivnosti. Tijekom razvoja je za atomične transakcije izdana zasebna specifikacija, WS-AtomicTransaction, kako bi se još detaljnije iskazale specifičnosti koje vrijede za takve transakcije. Ova specifikacija osigurava koordinacijske protokole za provođenje širokog spektra transakcija poslovnih aktivnosti, kao i podršku za jezike za modeliranje poslovnih procesa.

Modeliranje poslovnih procesa i jezici povezani s njime također su bili relativno slabo podržani u prvoj generaciji usluga Weba. Stoga je bilo potrebno definirati novi jezik koji će osigurati mogućnost pretvaranja poslovnog procesa neke tvrtke pomoću strukturiranog dijagrama toka u uslugu Weba. Najčešće se za ovu svrhu koristi jezik, nastao kao jedna od specifikacija za usluge Weba druge generacije, nazvan *Business Process Execution Language for Web Services* (BPEL4WS)³¹⁶. Tom se specifikacijom definira rječnik koji omogućuje da opis nekog

³¹² Web Services Interoperability Organisation, WS-I, <http://www.ws-i.org>

³¹³ Vidi poglavlje 5.2.1

³¹⁴ Web Services Coordination (WS-Coordination), <http://docs.oasis-open.org/ws-tx/wscoor/2006/06>

³¹⁵ WebServices Transaction (WS-TX), <http://www.ibm.com/developerworks/library/specification/ws-tx/>

³¹⁶ Business Process Execution Language for Web Services, <http://www.ibm.com/developerworks/library/specification/ws-bpel/>

poslovnog procesa bude prenesen u oblik izvršivih skripta, koje se mogu izvesti u jednom sustavu usluga Weba druge generacije. Takve se skripte najčešće izvode u nekom obliku procesa orkestracije. Opis poslovnog procesa u jeziku BPEL4WS opisuje njegov tijek koristeći slijed događaja zasnovan na predefiniranim uvjetima i ugrađenoj logici. Aktivnosti je moguće opisati korištenjem jednog od dvaju oblika aktivnosti, osnovnog i strukturiranog. Osnovne aktivnosti omogućuju opisivanje jednostavnih događaja, poput *receive*, *invoke*, *reply*, *throw*, *wait*. Strukturirane aktivnosti omogućuje kreiranje dijelova poslovne logike, pomoću aktivnosti kao što su *sequence*, *flow*, *switch* ili *while*.

Jedna od najvećih prepreka širem korištenju usluga Weba prve generacije bila je vrlo slaba podrška za bilo kakav oblik sigurnosti, osim onih osnovnih (koje nisu bile dijelom norme, već su se koristili na drugi način dostupni mehanizmi). Kako nisu mogle biti sigurne tko će, na koji način i s kojom svrhom koristiti njihovu uslugu, mnoge su se tvrtke vrlo teško odlučivale objaviti neku uslugu i dati je na javno korištenje korisnicima interneta. Bojazan je bila tim veća ako bi takva usluga uključivala izlaganje nekog dijela vlastitoga poslovnog procesa javnosti. Kao moguće poboljšanje predložena je specifikacija WS-Security³¹⁷, koja uvodi cjelovit model sigurnosti temeljen na mnoštvu komplementarnih normi.

Specifikacija osigurava sigurnosne mjere kako bi se osigurale SOAP poruke na cijelom njihovom putu te uspostavio određen stupanj povjerenja među stranama koje koriste i kreiraju usluge. Ova se specifikacija u velikoj mjeri oslanja na veći broj XML sigurnosnih normi, poput XML Key Managementa (XKMS)³¹⁸ ili Extensible Access Control markup Language (XACML)³¹⁹.

Usluge Weba prve generacije nisu posvećivale dovoljno pažnje načinu isporuke poruka u sustavu te taj sustav, prema mišljenju mnogih korisnika, nije bio dovoljno pouzdan. Kako bi se osiguralo da ne dođe do pogrešaka u sustavu, nužno je osigurati potvrdu o prispjeću poruke na odredište, bez obzira na to kakav je bio rezultat slanja poruke (je li poruka ispravno stigla na odredište ili je došlo do neke pogreške). Cilj specifikacije WS-ReliableMessaging³²⁰ je osigurati pouzdan sustav isporuke poruka u drugoj generaciji usluga Weba. Ona definira pravila za isporuku poruka kao i redoslijed dostavljanja poruka na odredište. Kako bi osigurala pouzdanu isporuku poruka, ova se specifikacija oslanja na specifikaciju WS-Policy³²¹, u kojoj su implementirana poslovna pravila o načinu izvješćivanja o dostavljenoj poruci.

Ako se neko poduzeće u cijelosti želi posvetiti razvoju uslužno orijentiranog poslovanja, tada bi cijeli skup vlastitih poslovnih pravila, sigurnosnih politika i ostalih propisanih postupaka trebali objediniti u jedinstven skup politika koje bi se tada mogle primjenjivati na apstraktnoj visokoj razini na usluge koje nudi na tržištu. Kako bi se omogućilo korištenje takvih mogućnosti, razvijena je specifikacija WS-Policy, koja ima definirane WS-PolicyAssertion i WS-PolicyAttachments.

³¹⁷ Web Services Security, <http://www.ibm.com/developerworks/webservices/library/specification/ws-secure/>

³¹⁸ XML Key Management Specification (XKMS), <http://www.w3.org/TR/xkms/>

³¹⁹ Extensible Access Control markup Language (XACML), http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-core-spec-os.pdf

³²⁰ Web Services Reliable Messaging, <http://www.ibm.com/developerworks/library/specification/ws-rm/>

³²¹ Web Services Policy, <http://www.w3.org/Submission/WS-Policy/>

Njihova je namjena osigurati uspostavu politika korištenjem policy konstruktora, uspostavom osiguranja dostave uspostavljenih politika te dodavanje postojećih konstrukta u kreirane politike.

Do sada smo naučili da se u sustavu usluga Weba intenzivno komunicira putem razmjene poruka. No komunikacija putem poruka protokola SOAP, o kojima smo pisali ranije u ovome poglavlju, prilično je ograničena te takvim porukama nije moguće prenijeti baš sve oblike podataka koje bi pružatelji i korisnici usluga Weba voljeli imati na raspolaganju. Kako bi se omogućilo prenošenje i drugih oblika podataka putem poruka protokolom SOAP, predložena je specifikacija WS- Attachments, koja tipove podataka koje tradicionalno nije moguće prenijeti putem SOAP-a pretvara u tzv. privitke (engl. *attachments*) klasičnim SOAP porukama.

Korištenjem ove specifikacije moguće je putem protokola SOAP prenijeti na primjer binarnu datoteku ili neki drugi XML dokument. WS-Attachments koristi učajurivanje DIME (engl. *Direct Internet Message Encapsulation*), koja se od mnogo poznatije učajurivanje MIME (engl. *Multipurpose Internet Mail Extensions*) razlikuje po mnogo jednostavnijem formatu za učajurivanje podataka, te je procijenjeno da je mnogo prikladnija za uporabu s porukama SOAP. Jedna druga specifikacija nazvana SOAP Messages with Attachments (SwA) koristi upravo MIME učajurivanje. Izbor načina korištenja prijenosa poruka je na programeru usluge, odnosno konkretnim potrebama funkcionalnosti usluge Weba koju projektira.

Još je jedna zanimljiva inicijativa u pogledu definiranja mogućih razina interoperabilnosti potekla od organizacije WS-I, nazvana WS-Basic Profile (WS-BP)³²². Prva je inačica profila predložena 2002. godine, a trenutno je aktualna inačica WS-Basic Profile 2.0 objavljena u ožujku 2010. U njemu se nalazi lista dostupnih specifikacija za usluge Weba (raspodijeljeno prema inačicama jer svaka inačica nije nužno kompatibilna s onom prethodnom)³²³. Njegovim su nam korištenjem na raspolaganju smjernice za implementaciju usluga Weba kako bi razvojem zaista postigli da je naša usluga u potpunosti interoperabilna. Korištenje ovog profila ne prejudicira koji ćemo od raspoloživih normi koristiti za razvoj usluge Weba, već taj izbor u potpunosti prepušta programeru. Ovaj je profil (u različitim svojim inačicama) dostupan u velikom broju različitih tehnoloških rješenja, poput IBM WebSphere³²⁴ ili .NET³²⁵.

Promotrimo ova proširenja u kontekstu uslužno orijentirane arhitekture (Slika 6.5). Kao što smo rekli, usluge Weba najčešće su korišten (iako nešto ograničenih mogućnosti u odnosu na SOA model) implementacijski oblik uslužno orijentiranih sustava.

Jedna od glavnih ideja koje su bile pokretači razvoja sustava okrenutih prema uslugama bila je mogućnost njihova slaganja, tj. sastavljanja složenijih usluga iz proizvoljnog broja jednostavnih usluga, koje su dostupne putem registra i koje su kreirali različiti pružatelji usluga. Naravno, opisane specifikacije druge generacije usluga Weba služe kako bi složene usluge (i

³²² Basic Profile 1.0, <http://www.ws-i.org/deliverables/workinggroup.aspx?wg=basicprofile>

³²³ Senthil Kumar, K. M., Saurav Das, A., Padmanabhuni, S., WS-I Basic Profile: A Practitioner's View, Proceedings of the IEEE International Conference on Web Services (ICWS'04), 2004

³²⁴ IBM WebSphere Application Server, <http://www-01.ibm.com/software/websphere/>

³²⁵ Microsoft .NET Framework, <http://www.microsoft.com/net/>

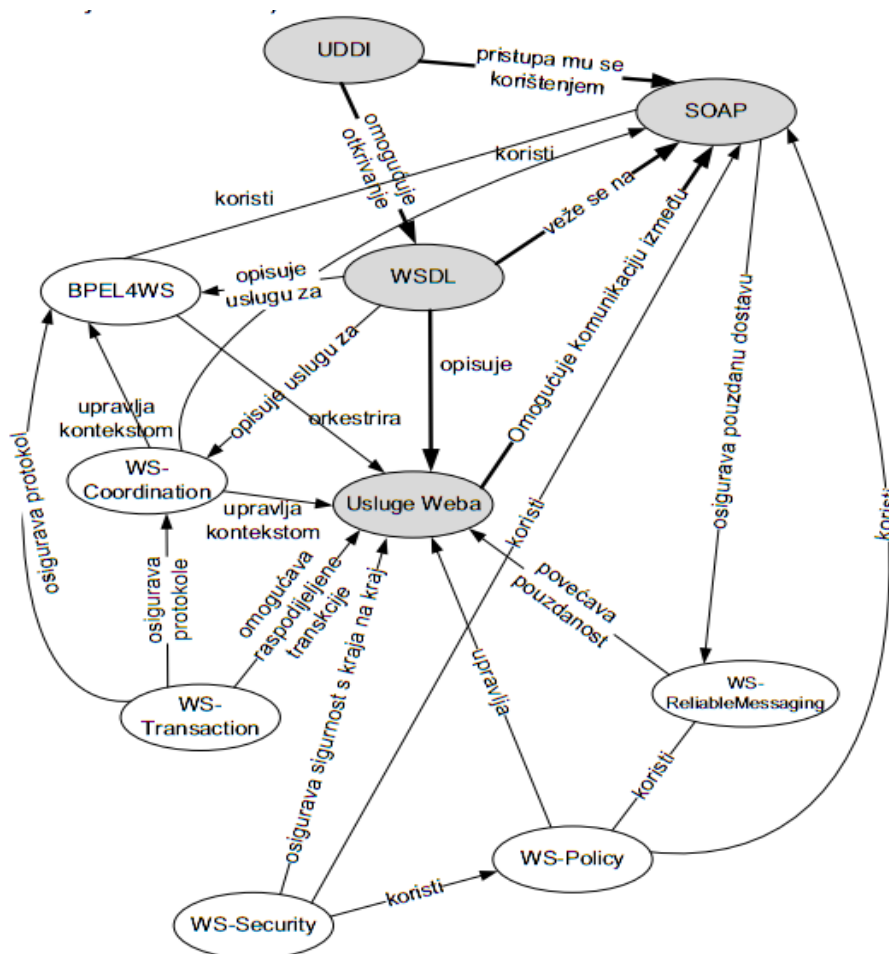
naravno, one jednostavnije od kojih su građene) bile što učinkovitije i bolje usmjerene prema cilju automatizacije poslovnih procesa u nekom uslužno orijentiranom poduzeću.

Kako bi ovaj cilj bio ostvaren, nužno je držati se određenih pravila tijekom faze projektiranja usluge Weba. Ta pravila uključuju:

- ograničiti korištenje specifikacija usluga Weba samo na one koje su nužne kako bi se ostvarila funkcionalnost za koju se projektira usluga,
- unutar aplikacijske logike projektirane usluge implementirati samo one dijelove specifikacije koji su potrebni za njen nesmetani rad,
- kako bi se pospješila komunikacija s drugim uslugama, SOAP poruke ne treba opterećivati nepotrebnim informacijama u njenom zaglavlju.

Dakle, možemo zaključiti da je jedan od glavnih zadataka uspješnog dizajna usluga Weba druge generacije poznavanje specifikacija, kako bi na temelju njih i iskustva uspješno mogli procijeniti koje od (brojnih) specifikacija i njihovih dijelova treba koristiti kako bi se ostvarila željena funkcionalnost kod projektirane usluge Weba.

Slika 7.10 prikazuje međusobni odnos najbitnijih specifikacija usluga Weba druge generacije prema komponentama poznatim iz njihove prve generacije (njih smo označili sivo, a njihov je odnos označen debljim strelicama).



Slika 7.10: Odnos među specifikacijama usluga Weba prve i druge generacije

Vidimo da se situacija prvotno jednostavno zamišljenog sustava (Slika 6.5 iz prošlog poglavlja) komplicira uvođenjem novih specifikacija. No, kako smo objasnili, svaka od novouvedenih specifikacija omogućuje rješavanje određenog problema koji nije moguće (barem ne jednostavno) riješiti pomoću prve generacije. Nemojte smetnuti s uma ni da će, ovisno o funkcionalnosti konkretne implementacije usluge Weba, ova slika izgledati bitno jednostavnije jer će samo rijetke usluge Weba zahtijevati implementaciju baš svih spomenutih specifikacija.

7.6.2 RESTful usluge Weba

Na kraju ovog poglavlja ukratko ćemo predstaviti jednu nešto drukčiju vrstu usluga Weba. Ove se usluge Weba oslanjaju na *Representational State Transfer* (REST)^{326, 327}. Ovaj arhitekturni model uveo je Roy Fielding u svojoj doktorskoj disertaciji, objašnjavajući pomoću njega izvrsnu skalabilnost koju je iskazao protokol HTTP u svojoj inačici 1.0. Kako je on bio i jedan od tvorca specifikacije protokola HTTP, i kasniji, trenutno važeći standard HTTP 1.1 prati arhitekturu REST- a. Kada želimo reći da neka usluga Weba prati ovaj stil arhitekture, zovemo je RESTful uslugom Weba.

REST se u osnovi oslanja na jednostavne, standardizirane elemente. Oni koriste URI za identifikaciju dostupnih resursa. Pomoću URI-ja odmah se rješava i problem globalnoga adresnog prostora kao i otkrivanje usluga i resursa. Nakon što je neki resurs pronađen, njime se može manipulirati korištenjem operacija za kreiranje, čitanje, ažuriranje i brisanje resursa (PUT, GET, POST, i DELETE). Zanimljivo je vidjeti da izvođenjem operacija GET i POST zapravo dobivamo, odnosno mijenjamo trenutno stanje resursa nad kojim izvodimo operaciju. Sami se resursi odvajaju od njihove implementacije kako bi se omogućilo da poruke koje se prenose sadržavaju dovoljno informacija kako bi se njihov sadržaj mogao ispravno predočiti na bilo koji način (primjerice, omogućava se da se neki HTML dokument pročita kao XML datoteka ili kao čisti tekst dokument). Pritom se stanje u kojem je resurs ne prati pomoću poruka, već korištenjem nekog od eksternih mehanizama (prepisivanje URI-ja, kolačići itd.).

Prevladavajuće je mišljenje da su RESTful usluge Weba općenito jednostavnije za programiranje i korištenje jer se u njihovom dizajnu koriste samo W3C/IETF standardi (poput HTML-a, XML-a ili URI-ja). Uz veliku količinu standardnih klijenata koji ih podržavaju, razvijen je i velik broj jednostavnih alata, čime je osigurano jednostavno usvajanje za sve koji ih žele koristiti. Korištenjem mehanizama ugrađenih u arhitekturni model REST (poput korištenja priručne memorije i raspodjele opterećenja) RESTful usluge Weba istovremeno može koristiti razmjerno velik broj klijenata. Naravno, postoje i određena ograničenja. Primjerice, moderni vatrozidovi koji su u širokoj uporabi neće dopustiti promet (bez dodatnog podešavanja – nepraktično za veliku većinu običnih korisnika) koji sadržava HTTP upite PUT i DELETE.

Zaseban je problem i ako je potrebno na ovaj način prenijeti veću količinu podataka. U ovoj inačici standarda ta je količina ograničena na 4 kB, te podatke koji zauzimaju više prostora nije

³²⁶ Fielding, R.. Architectural Styles and The Design of Network-based Software Architectures. PhD thesis, University of California, Irvine, 2000.

³²⁷ Richardson, L., Ruby, S., RESTful web services, O'Reilly Media, Inc., 2007.

moguće kodirati u URI. U tom slučaju poslužitelj jednostavno odbacuje takav URI kako neispravan.

Trenutno u svijetu programera usluga Weba vlada veliko razmimoilaženje oko toga koja je usluga bolja, jednostavnija itd. Pobjednik u toj debati još uvijek je neizvjestan.

Pokažimo na primjeru kako bi izgledala jedna svima poznata usluga (*e-mail* – e-pošta) kada bi je se dizajniralo korištenjem načela REST-a, opisanih u ovom poglavlju³²⁸.

Pretpostavimo da smo svome prijatelju napisali poruku koju želimo poslati *e-mailom*. Za slanje te poruke u obliku *e-maila* poslužitelj će koristiti metodu PUT. Poslužitelj (definiran kao lokalni izlazni *mail* poslužitelj) tada će zapravo otkriti URI na kojem se može naći odlazna poruka. Naravno, ugrađeni mehanizam sigurnosti brine o tome da jedino autorizirani korisnik smije iskoristiti taj URI kako bi pokupio poruku. Naglasimo da poruka u stvarnosti nikad ne napusti odlazni poslužitelj dok god je primatelj ne obriše (njoj se pristupa putem njenog URI-ja).

Da bi primatelj pročitao poruku, koristit će metodu GET kako bi dohvatio sve ispravno oblikovane poveznice na kojima se kriju njemu namijenjene poruke. Izvršavanjem naredbe GET nad jednim od tih poveznica odvodi se korisnika na poslužitelj poruka elektroničke pošte koji čuva njemu namijenjenu poruku kako bi je mogao dohvatiti i pročitati. Korištenjem naredbe DELETE s poslužitelja može izbrisati poruku koju je pročitao i više je ne želi čuvati. Ako se neka poruka želi spremiti, tada je se pomoću naredbe POST sprema u odgovarajući direktorij (koji je također određen svojim URI-jem). Za prosljeđivanje poruke također koristimo POST, jedino što kao odredište ovog zahtjeva koristimo URI nečijega poštanskog sandučića. O prispjeću poruke moguće je korištenjem POST-a poslati jednostavnu poruku koja sadržava samo URI kako bi se primatelj obavijestio da je primio novu poruku. Ovaj pristup pruža i mogućnost da se jednom poslana pošta mijenja (sjetite se koliko ste puta zaboravili poslati privitak uz poruku te ste je morali poslati još jednom). Kao što vidimo, ovakav pristup donosi neke novosti i poboljšanja o kojima korisnici često pričaju.

³²⁸ Cowan, J., RESTful Web Services: An introduction to building Web Services without tears (i.e., without SOAP or WSDL).

8. Poglavlje: Sigurnost interoperabilnosti i usluga

U ovom poglavlju naučit ćete:

- ✓ osnovne pojmove i zahtjeve nad sigurnost kod interoperabilnosti i usluga
- ✓ o kriptografiji u službi sigurnosti usluga
- ✓ o korištenju elektroničkog potpisa i elektroničke oмотnice
- ✓ osnovne napade na sigurnost komunikacije i usluga
- ✓ o sigurnosti interoperabilnih usluga

8.1 Osnovni pojmovi sigurnosti

U dosadašnjim poglavljima zaključili smo da se interoperabilnost zasniva na komunikaciji dviju strana, i to na siguran i pouzdan način. Zaključili smo i da se interoperabilnost većinom izvodi nekom vrstom usluge. U ovom poglavlju predstavljamo metode i načela koja se koriste kako bi se osigurale usluge i interoperabilnost između subjekata koja se ne može narušiti slučajnom ili namjernom promjenom podataka koji se razmjenjuju.

Svako elektroničko poslovanje mora zadovoljiti niz sigurnosnih preduvjeta kako bi bilo učinkovito, sigurno i pouzdano:

- informacije se moraju štititi od neovlaštenog pristupa,
- poruke se ne smiju mijenjati,
- poruke koje su poslone i primljene ne mogu se poreći,
- svaka strana mora biti sigurna u identitet druge strane.

8.1.1 Sigurnosni zahtjevi kod interoperabilnosti

U priručniku „Sigurnost informacijskih sustava“ upoznali ste se s osnovnim sigurnosnim trokutom nazvanim trijadom CIA (akronim *Confidentiality, Integrity and Availability*) koji se sastoji od triju pojmova: povjerljivosti, cjelovitosti i dostupnosti. Kasnije su neki autori taj koncept pokušali proširiti, pa je jedan od takvih pokušaja tzv. Parkerova heksada³²⁹ (engl. *Confidentiality, Possession (or Control), Integrity, Authenticity, Availability and Utility*), koja predstavlja šest osnovnih značajki informacije – **povjerljivost, posjedovanje, cjelovitost, izvornost, dostupnost i korisnost**. Ove značajke su atomarne, ne razlažu se na dijelove, ne preklapaju se po smislu i označavaju jedinstvene aspekte informacije.

Kako bi bolje razumjeli koncepte i načela predstavljena u ovom poglavlju, ponovit ćemo definicije određenih pojmova, važnih za razumijevanje problematike sigurnosti kod interoperabilnosti i sigurnosti usluga:

- **povjerljivost** (engl. *confidentiality*) i **tajnost** (engl. *secrecy*) – odnosi se na očuvanje sadržaja poruke povjerljivim i tajnim. Sadržaj poruke treba biti razumljiv samo pošiljatelju i primatelju kojem je poruka namijenjena.
- **cjelovitost, očuvanost** ili **integritet** (engl. *integrity*) – odnosi se na nepostojanje promjena sadržaja poruke tijekom njenog prijenosa. Ako do promjene i dođe, mora se uočiti i primatelju dati na znanje da je poruka promijenjena. Promjene se dijele na namjerne i nenamjerne. Nenamjerne su poruke najčešće izazvane pogreškama u prijenosu, dok su namjerne promjene posljedica sigurnosnih napada na komunikaciju porukom. No cjelovitost informacije nema nikakav utjecaj na njenu ispravnost (točnost).
- **dostupnost** usluge (engl. *availability*) – odnosi se na osiguravanje funkcioniranja određene usluge i njenog korištenja s određenom razinom pouzdanosti, odnosno vjerojatnosti njenog rada. Informacije trebaju biti pravodobno dostupne, odnosno onda kada ih korisnik zatraži, a vrijeme isporuke informacije treba biti prihvatljivo korisniku.

³²⁹ Parker, Donn B., „Toward a New Framework for Information Security“ u Bosworth, Seymour; Kabay, M.E., *The Computer Security Handbook*, 4. izdanje, New York; John Wiley & Sons, 2002

Dostupnost se rješava nizom tehnika koje omogućavaju dostupnost usluga, najčešće pomoću aktivnog sprječavanja napada.

- **Posjedovanje** (engl. *possession*) – odnosi se na kontrolu nad objektom koju osoba posjeduje i njegovo korištenje. Posjedovanje može biti zakonito ili nezakonito. Ako netko ukrade ključ i time ga posjeduje, to ne znači nužno i da ga je upotrijebio, ali samo posjedovanje omogućuje da ga potencijalno upotrijebi. Posjedovanje treba razlikovati od vlasništva, jer vlasništvo omogućava i pravno raspolaganje.
- **Izvornost, ovjera ili autentičnost** (engl. *authenticity*) – odnosi se na sposobnost određivanja izvora poruke, odnosno strane koja je poruku poslala. Smatra se da je pošiljalac ovjerio, odnosno autenticirao poruku prilikom njenog slanja te da primatelj može biti siguran u izvor poruke, odnosno koji je pošiljalac poslao poruku. Ako dolazi do promjene izvora poruke, pošiljalac mora toga biti svjestan.
- **Korištenje** (engl. *utility*) – povezano s korisnošću (engl. *usefulness*) određene informacije. Ako se informaciju ne može koristiti, ona je u načelu bezvrijedna. Povreda korištenja događa se kada je onemogućeno korištenje određene informacije tako da nije ugrožena njena dostupnost, nego ako je primjerice korisnik izgubio ključ potreban za autorizirani pristup. Povreda korištenja je i ako je informacija pohranjena u neprikladnom obliku, pa su potrebne dodatne transformacije kako bi se mogla koristiti. Korištenje treba razlikovati od dostupnosti, jer i dostupnu informaciju nije nužno moguće koristiti.
- **Neporicljivost ili neporecivost** (engl. *non-repudiation*) – odnosi se na nemogućnost poricanja slanja poruke koju je poslao određeni pošiljalac i primio primatelj. Neporicljivost je povezana s autentičnošću budući da određeni pošiljalac, za kojeg se može utvrditi identitet, ne može poricati da je baš on poslao poruku. Neki autori ovaj termin prevode i kao nepobitnost, nepovratnost i neodbijanje.
- **Kontrola pristupa** (engl. *access control*) – odnosi se na sposobnost zabrane prava pristupa određenoj funkciji ili resursu, odnosno dodjele tog prava. Kontrola pristupa omogućava korištenje resursa na pouzdan način samo od autoriziranih korisnika.

8.2 Korištenje kriptografije

U priručniku „Sigurnost informacijskih sustava“ upoznali ste se s kriptografijom i kriptografskim algoritmima, čega ćemo se podsjetiti u ovom poglavlju. No i nadopunit ćemo ono što je tamo prikazano, jer će nam to trebati u objašnjavanju sigurnosti nad uslugama Weba i općenito sigurnosti kod interoperabilnosti.

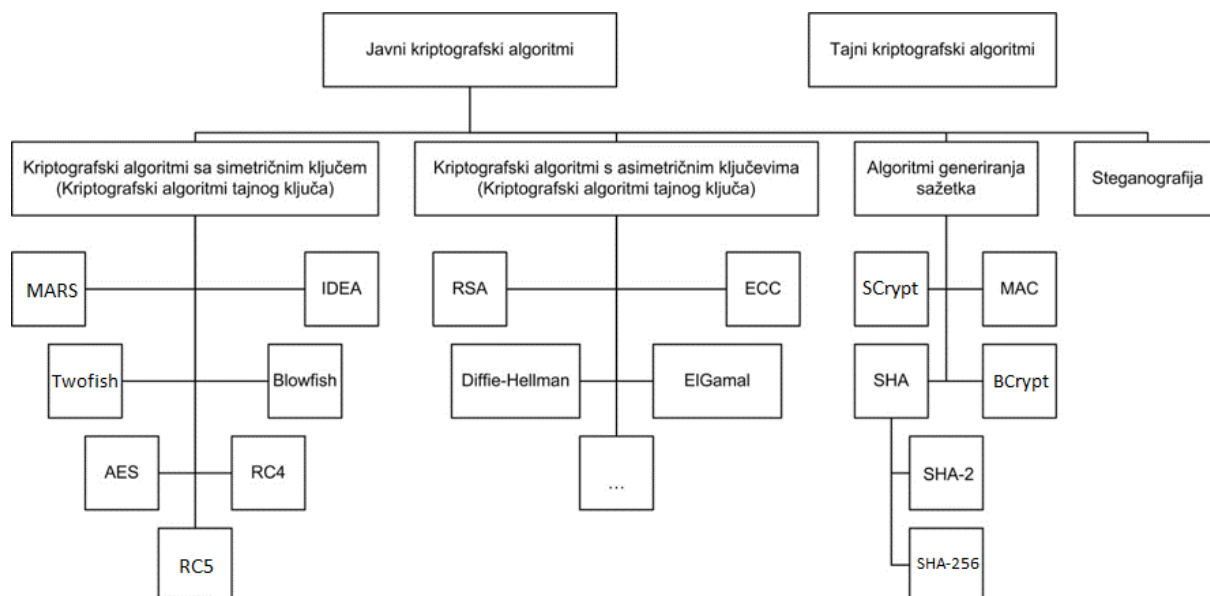
Kriptiranje ili šifriranje je proces kojim se izvorna informacija ili čisti tekst (engl. *plain text*) transformira u kriptirani ili šifrirani tekst (engl. *cipher text*), a dekriptiranje ili dešifriranje je obrnuti postupak. **Kriptografski algoritam** definira metodu transformacije, a **ključ** je podatak koji omogućava transformaciju u jednom ili drugom smjeru postupka kriptiranja. Iako su tijekom povijesti postojali, a i danas postoje različiti tajni kriptografski algoritmi, od njih se većinom odustaje, jer se otkrivanjem samog algoritma, odnosno metoda transformacija, praktički otkriva i kako dekriptirati sve buduće tekstove ili dokumente kriptirane tim tajnim algoritmom. Zato su većinom u uporabi **javni kriptografski algoritmi**, koji postoje u različitim oblicima.

8.2.1 Javni kriptografski algoritmi

Kao što ste već djelomično vidjeli u priručniku „Sigurnost informacijskih sustava“, **javni kriptografski algoritmi** dijele se na:

- **Kriptografske algoritme s ključem** (engl. *key cryptographic algorithms*) – algoritmi koji koriste određeni ključ prilikom kriptiranja i dekriptiranja. Dijelev se na:
 - **Kriptografske algoritme sa simetričnim ključem** (engl. *symmetric key cryptographic algorithms*) ili simetrične kriptografske algoritme, a koji se još nazivaju i **kriptografskim algoritmima tajnog ključa** (engl. *secret key cryptographic algorithms*), od kojih su najpoznatiji predstavnici algoritmi DES (engl. *Data Encryption Standard*), 3-DES, odnosno Triple DES ili TDEA (engl. *Triple Data Encryption Algorithm*), AES (engl. *Advanced Encryption Standard*) temeljen na algoritmu Rijndael, RC4 ili ARC4 (engl. *Alleged RC4*) korišten kod SSL-a i WEP-a, Twofish, Blowfish, CAST5 ili CAST-128, Serpent i IDEA (engl. *International Data Encryption Algorithm*).
 - **Kriptografske algoritme s asimetričnim ključevima** (engl. *asymmetric key cryptographic algorithms*) ili asimetrične kriptografske algoritme, a koji se još nazivaju i **kriptografskim algoritmima javnog ključa** (engl. *public key cryptographic algorithms*), od kojih je najpoznatiji algoritam RSA (Rivest, Shamir i Adleman), ali postoje i mnogi drugi, poput algoritama Diffie-Hellman i ElGamal ili tehnika eliptičnih krivulja ECC (engl. *elliptic curves cryptography*).
- **Algoritme generiranja sažetka** (engl. *hash*), odnosno kriptografske funkcije sažetka (engl. *cryptographic hash function*) – koriste se za generiranja sažetaka poruka. Sažeci, često nazvani i skraćenim prikazom poruke (engl. *message digest*), su točno određene duljine. Najpoznatiji predstavnici algoritama generiranja sažetka su MD5 (engl. *Message Digest algorithm 5*), porodica algoritama SHA (engl. *Secure Hash Algorithm*), od kojih se trenutno upotrebljavaju algoritam SHA-1 i grupa algoritama SHA-2 (algoritmi SHA-224, SHA-256, SHA-384 i SHA-512), i porodica algoritama MAC (engl. *message authentication code*).
- **Steganografiju** (engl. *steganography*) ili metodu sakrivanja tekstualnih, slikovnih ili multimedijalnih poruka u druge tekstove, slike ili multimedijalne poruke, a koji se često koristi u slučaju zabrane kopiranja ili zadržavanja informacije o autorstvu (engl. *copyright*), kao što je u slučaju raznih tehnologija DRM (engl. *digital rights management*).

Ova se podjela može prikazati i grafički kao na sljedećoj slici:



Slika 8.1: Podjela javnih kriptografskih algoritama

8.2.2 Usporedba značajki kriptografskih algoritama sa simetričnim i asimetričnim ključem

Kako ne bismo ulazili u potankosti algoritama koje ste već vidjeli u priručniku „Sigurnost informacijskih sustava“, usporedit ćemo svojstva algoritama bitna za njihovu primjenu, a koja je bitna za interoperabilnost.

Kriptografski algoritmi sa simetričnim ključem koriste isti ključ za kriptiranje i dekriptiranje. Taj ključ je tajan, znaju ga samo pošiljatelj i primatelj i oni ga dijele (engl. *share*). Osim o odabiru algoritma, *sigurnost ovisi o samom ključu*, odnosno matematički gledano o njegovoj *duljini*, o *sigurnoj pohrani ključa* na obje strane i o mehanizmu *dijeljenja*, odnosno razmjene ključa između strana. U kontekstu interoperabilnosti mora postojati način, odnosno protokol *sigurnog prenošenja ključa*. Iako je problem sigurnog prenošenja ključa manji od problema sigurnog prenošenja cijele poruke, ipak je načelno riječ o istom mehanizmu. Čitav mehanizam ovisi o dogovoru dviju strana o ključu, koji zajednički dijele i razmjenjuju nekim drugim oblikom sigurnog kanala.

Prednosti kriptografskih algoritama sa simetričnim ključem su *velika raznolikost* algoritama prilikom odabira i *velika brzina* temeljem načelno manje računske složenosti i jednostavnije programske ili sklopovske implementacije. Mane svakako uključuju *problem sigurne distribucije ključa*, odnosno skupa ključeva kod više uključenih strana.

Kriptografski algoritmi s asimetričnim ključem, odnosno javnim ključem koriste par ili parove javnih i tajnih ključeva. Sigurnost i ovdje, osim o odabiru algoritma, *ovisi o duljini ključa te sigurnoj pohrani tajnih ključeva*, dok se javni ključevi javno obznanjaju. Javno obznajivanje javnih ključeva je i najveća prednost kriptografskih algoritama s asimetričnim ključem, s

obzirom na to da se tajni ključevi ne moraju dijeliti i razmjenjivati na siguran način, već se samo trebaju sigurno pohraniti. No mane kriptografskih algoritama s asimetričnim ključem su svakako sporost zbog velike računske složenosti i kompleksna implementacija.

8.2.3 Ključevi

Različite metode zaštite, sigurnosni mehanizmi i kriptografski algoritmi koriste različite vrste ključeva, pa tako ključeve možemo podijeliti na:

- **kratke ključeve** koji su najčešće predstavljeni znakovno-broječnim nizovima i često su lako pamtljivi te razlikujemo:
 - **lozinke** ili **zaporke** – kratki nizovi znakova, pamtljivi, često se izvode iz nekih drugih osobama poznatih podataka, pa se zato uvode razne sigurnosne mjere prilikom njihovog odabira ili promjene,
 - **ključeve za jednokratnu upotrebu** ili **OTP** (*one-time password*) – najčešće brojevni nizovi koje generira neki sigurnosni uređaj (npr. token) ili se odabiru iz tablica s mnogo takvih ključeva na temelju nekih kriterija,
 - **osobne identifikacijske brojeve** ili **PIN-ove** (engl. *personal identification number*) – kratki pamtljivi brojevni nizovi (najčešće od 4 ili 6 znamenaka) koji služe za brzu identifikaciju i autorizaciju, najčešće u sustavima autorizacije plaćanja,
- **dulji ključevi** predstavljeni znakovno-broječnim nizovima koji nisu lako pamtljivi, a dijele se na:
 - **sažetke** – skraćeni prikaz poruke izveden algoritmom generiranja sažetaka,
 - **identifikatore** – različiti identifikacijski podaci koji identificiraju neki entitet,
 - **certifikate** – elektronička potvrda kojom se potvrđuje identitet potpisnika,
- biometrijske ključeve, a najčešći su:
 - **otisci prsta ili dlana,**
 - **uzorci šarenica ili lica,**
 - **uzorci glasa,**
 - **uzorci DNK.**

8.3 Elektronički potpis

Ne europskoj je razini još 1999. godine donesena direktiva EU 1999/93/EC³³⁰ o elektroničkom potpisu. Razvojem problematike slijedi niz normi, radnih sporazuma i drugih dokumenata, koje ćemo zbog njihove opširnosti ovdje samo nabrojiti, a o njima se više može naći pretraživanjem:

- norme ISO 9796-2, 9797-2, 10118-2, 11770, 13888, 14888, 15496, 15000-2, 7498-2, 10181-4, 9594-8, 19785-1 i 19794-2,
- dokumenti CWA 14167-1, 14167-2, 14167-3, 14168, 14169, 14170, 14171, 14172-1, 14172-2, 14172-3, 14172-4, 14172-5, 14355 i 14365,
- norme ETSI TS 101 733, TS 101 861, TS 101 862, TS 101 903, TS 101 456, TR 102 038, TS 102 042 i TS 102 280,

³³⁰ Directive 1999/93/EC on a Community framework for electronic signatures, 1999-12-13, EU, <http://portal.etsi.org/esi/Documents/e-sign-directive.pdf>

- dokumenti RFC 2251, 2256, 3280 i 3739,
- norme PKCS 1, 5, 7, 9, 10, 11 i 15,
- niz preporuka W3C-a,

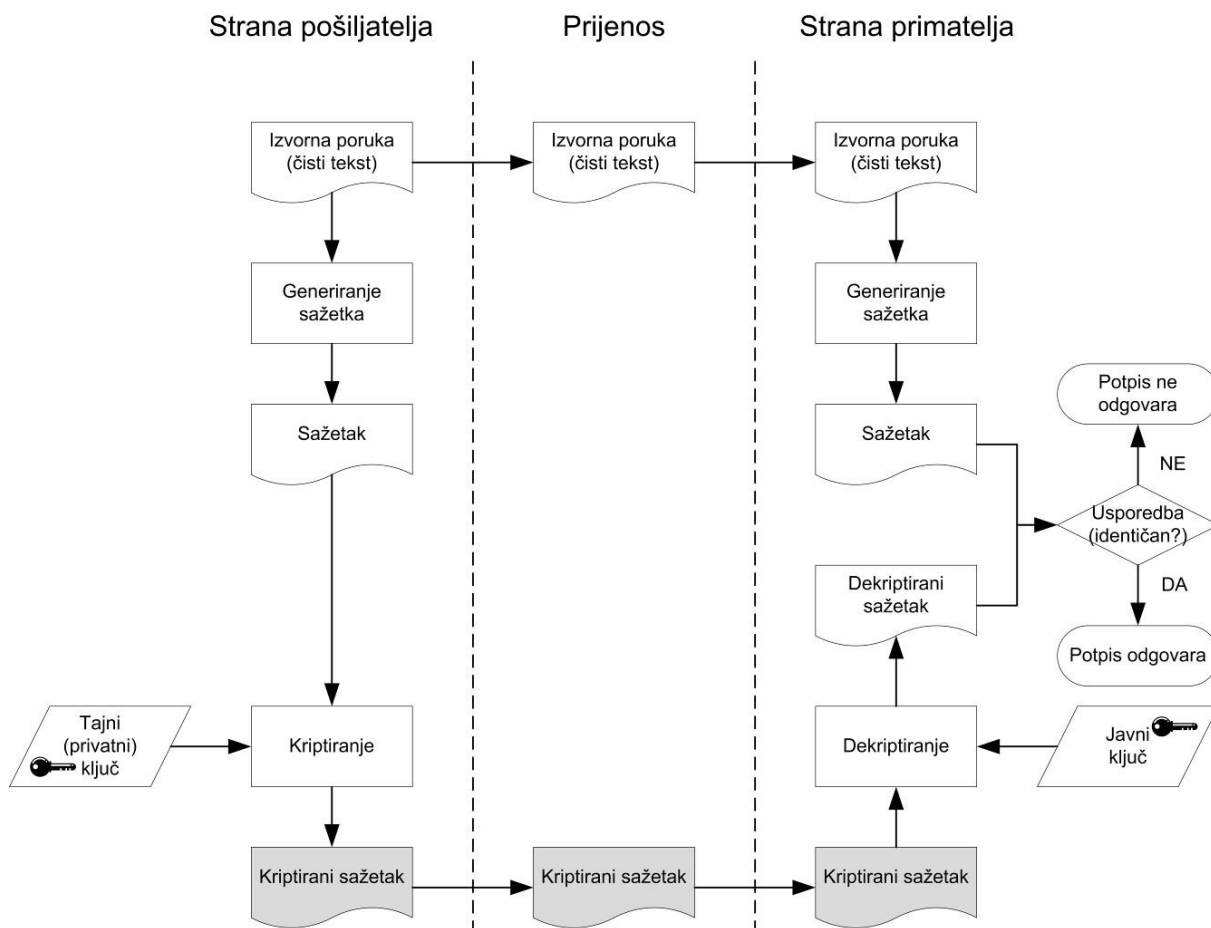
Republika Hrvatska je 24. siječnja 2002. godine donijela **Zakon o elektroničkom potpisu**³³¹. Kao što se već vidjeli u poglavlju 1.6.1, Zakon o elektroničkom potpisu definira niz pojmova, kao što su:

- elektronički potpis i napredan elektronički potpis,
- vremenski žig,
- elektronički zapis,
- podaci za izradu elektroničkog potpisa i sredstvo za izradu (naprednog) elektroničkog potpisa,
- podaci za verificiranje elektroničkog potpisa i sredstvo za verificiranje potpisa,
- certifikat i kvalificirani certifikat,
- davatelj usluga certificiranja,
- sredstvo za elektronički potpis.

U procesima stvaranja i provjere elektroničkog potpisa koriste se kriptografski algoritmi. Osnovni proces uključuje dvije strane, pošiljatelja i primatelja. Kod generiranja elektroničkog potpisa pošiljatelj generira sažetak poruke koju želi poslati. Taj sažetak zatim kriptira svojim tajnim (privatnim) ključem. Nakon toga pošiljatelj šalje poruku i kriptirani sažetak primatelju. S druge strane primatelj dekriptira sažetak javnim ključem pošiljatelja. I primljene poruke na isti način i istim mehanizmom, odnosno algoritmom generiranja sažetka (engl. *hash*), generiraju sažetak primljene poruke. Ako su dekriptirani sažetak i generirani sažetak identični, poruka je istovjetna izvorniku. Ovaj se proces može prikazati i grafički (Slika 8.2).

Prilikom dekriptiranja sažetka javnim ključem pošiljatelja, s obzirom na to da samo pošiljatelj posjeduje tajni ključ kojim je kriptiran sažetak, utvrđuje se izvornost (ovjera, autentičnost), a pošiljatelj ne može poreći da je on generirao sažetak, a time ni da je autor poruke, pa je utvrđena i neporicljivost.

³³¹ Zakon o elektroničkom potpisu, Narodne novine, br. 10/02. i Zakon o izmjenama i dopunama Zakona o elektroničkom potpisu, Narodne novine, br. 80/08.



Slika 8.2: Elektronički potpis

8.3.1 Elektronički potpis i napredni elektronički potpis

Kao što se može primijetiti iz popisa definicija pojmova Zakona o elektroničkom potpisu i definicija iz poglavlja 1.6.1, razlikuju se elektronički potpis i napredni elektronički potpis. **Elektronički potpis** definira se kao skup podataka u elektroničkom obliku koji služe za identifikaciju potpisnika i potvrdu vjerodostojnosti potpisanoga elektroničkog zapisa. **Napredni elektronički potpis** proširuje tu definiciju tako da definira da je potpis:

- povezan isključivo s potpisnikom,
- nedvojbeno identificira potpisnika,
- nastaje korištenjem sredstava kojima potpisnik može samostalno upravljati i koja su isključivo pod nadzorom potpisnika,
- sadržava izravnu povezanost s podacima na koje se odnosi, i to na način koji nedvojbeno omogućava uvid u bilo koju izmjenu izvornih podataka.

Napredni elektronički potpis ima istu pravnu snagu i zamjenjuje vlastoručni potpis, odnosno vlastoručni potpis i otisak pečata. U pravnom smislu, **ne može se osporiti** prihvatanje dokumenta samo zbog toga što je sačinjen i izdan u elektroničkom obliku s elektroničkim potpisom ili naprednim elektroničkim potpisom, čak ni u sudskim postupcima.

Sredstvo za izradu naprednoga elektroničkog potpisa mora osigurati da se podaci za izradu naprednoga elektroničkog potpisa mogu pojaviti samo jednom, da je ostvarena njihova sigurnost, da se ne mogu ponoviti, da se od strane potpisnika mogu pouzdano zaštititi protiv korištenja od strane drugih te da je potpis zaštićen od krivotvorenja pri korištenju postojeće raspoložive tehnologije. Sredstvo za izradu naprednoga elektroničkog potpisa ne smije prilikom izrade naprednoga elektroničkog potpisa promijeniti podatke koji se potpisuju ili onemogućiti potpisniku uvid u te podatke prije izrade naprednoga elektroničkog potpisa.

8.3.2 Certifikat i vremenski žig

Certifikat je svaka elektronička potvrda kojom se potvrđuje identitet potpisnika u postupcima razmjene elektroničkih zapisa. **Kvalificirani certifikat** je svaka elektronička potvrda kojom davatelj usluga izdavanja kvalificiranih certifikata potvrđuje napredni elektronički potpis. On mora sadržavati:

- oznaku o tome da je riječ o kvalificiranom certifikatu,
- identifikacijski skup podataka o izdavatelju certifikata,
 - osobno ime, ime oca ili majke, datum rođenja, prebivalište odnosno boravište te naziv pravne osobe i sjedište, ako certifikat izdaje pravna osoba,
- identifikacijski skup podataka o potpisniku,
 - osobno ime, ime oca ili majke, datum rođenja, prebivalište odnosno boravište,
- podatke za verificiranje elektroničkog potpisa koji odgovaraju podacima za izradu elektroničkog potpisa koji su pod kontrolom potpisnika,
- podatke o početku i kraju važenja certifikata,
- identifikacijsku oznaku izdanog certifikata,
 - brojčanu ili drugu oznaku i datum izdavanja,
- napredni elektronički potpis davatelja usluge izdavanja kvalificiranih certifikata,
- ograničenja vezana za uporabu certifikata, ako ih ima,
- ograničenja na vrijednost poslovnih događaja za koje se daje certifikat, ako ih ima.

Vremenski žig (engl. *time stamp*) je dodatni podatak uz elektronički potpis koji potvrđuje sadržaj podataka u određenom vremenu, odnosno vremenskom razdoblju. Na vremenski se žig i s njim povezane usluge primjenjuju odredbe koje reguliraju certifikat, a za napredan vremenski žig odredbe koje se odnose na kvalificirani certifikat.

8.3.3 Davatelji usluga certificiranja i izdavanja kvalificiranih certifikata

Davatelj usluga certificiranja ne treba posebnu dozvolu za obavljanje usluga certificiranja, a Ministarstvo gospodarstva, rada i poduzetništva upisuje davatelje usluga certificiranja u Evidenciju davatelja usluga certificiranja u Republici Hrvatskoj. Davatelji usluga izdavanja kvalificiranih certifikata kojima je izdana dozvola upisuju se u Registar davatelja usluga izdavanja kvalificiranih certifikata u Republici Hrvatskoj.

Davatelj usluga izdavanja kvalificiranih certifikata za obavljanje usluga treba dozvolu od Ministarstva gospodarstva, rada i poduzetništva, i mora, osim navedenih uvjeta za davatelja usluga certificiranja, ispunjavati i sljedeće uvjete:

- dokazanu sposobnost sigurne provedbe usluga certificiranja,
- osigurane uvjete djelovanja sigurnog i ažurnog registra potpisnika te provedbu sigurnog i trenutnog prekida, odnosno opoziva usluge certificiranja na zahtjev potpisnika,
- osigurano točno utvrđivanje datuma i vremena izdavanja ili opoziva certifikata,
- osiguranu provjeru identiteta i dodatnih obilježja osobe za koju se izdaje certifikat,
- pouzdane sustave i proizvode koji su zaštićeni od preinaka i osiguravaju tehničku i kriptografsku sigurnost procesa,
- pouzdane mjere protiv krivotvorenja te zaštićen i povjerljiv proces generiranja podataka elektroničkog potpisa,
- dostatne financijske resurse za rad i prikladno osiguranje za štete,
- sustav pohrane relevantnih informacija (kvalificirane certifikate) za određeno vrijeme i pružanje evidencije certifikata,
- sigurnosni sustav koji onemogućuje pohranjivanje i kopiranje podataka za izradu potpisa za osobe u ime kojih se pruža usluga certificiranja,
- sustav informiranja korisnika usluga certificiranja o točnim uvjetima korištenja usluga i ograničenjima u korištenju, kao postupke za rješavanje pritužbi i žalbi,
- pouzdani sustav pohranjivanja certifikata u obliku koji omogućuje provjeru kako bi:
 - unos i promjene radile samo ovlaštene osobe,
 - informacije mogle biti provjerene za autentikaciju,
 - certifikat bio javno raspoloživ za pretraživanje samo kada postoje ovlaštenja,
 - tehnička promjena koja može narušiti sigurnosne zahtjeve bila poznata davatelju usluge certificiranja.

Davatelj usluga certificiranja može u kvalificiranom certifikatu naznačiti ograničenja uporabe certifikata, kao i naznačiti granicu vrijednosti transakcija za koje se certifikat može koristiti, pod uvjetom da je to ograničenje vidljivo i trećim stranama.

8.3.4 Prava i obveze potpisnika i davatelja usluga certificiranja

Kvalificirani certifikat može se izdati svakoj osobi na njen zahtjev te na osnovi nesumnjivo utvrđenog identiteta i ostalih detaljnijih podataka o osobi (utvrđuje se na druge načine, npr. usporedbom s osobnom iskaznicom, putovnicom, domovnicom i sl.).

Potpisnici mogu svojevrijedno odrediti davatelja usluga certificiranja. Potpisnik može koristiti usluge certificiranja od jednog i više davatelja usluga certificiranja, a s odabranim davateljima usluga zaključuje ugovore. Usluge certificiranja i izdavanja kvalificiranih certifikata mogu obavljati i davatelji usluga certificiranja sa sjedištem u inozemstvu.

Davatelj usluga certificiranja ima obvezu:

- osigurati da svaki kvalificirani certifikat sadržava sve potrebne podatke,
- provesti potpunu provjeru identiteta potpisnika za kojeg provodi usluge certificiranja,
- osigurati točnost i cjelovitost podataka koje unosi u evidenciju izdanih certifikata,
- u svaki certifikat unijeti osnovne podatke o sebi,
- omogućiti svakoj zainteresiranoj osobi uvid u identifikacijske podatke i dozvolu za izdavanje kvalificiranih certifikata,
- voditi točnu i zaštićenu evidenciju certifikata koja mora biti javno dostupna,
- voditi točnu i zaštićenu evidenciju opozvanih ili nevažećih certifikata,

- osigurati vidljiv podatak o točnom datumu i vremenu izdavanja odnosno opoziva certifikata u evidenciji izdanih certifikata,
- čuvati sve podatke i dokumentaciju o izdanim certifikatima najmanje 10 godina.

Davatelj usluga certificiranja dužan je prekinuti uslugu certificiranja, odnosno izvršiti opoziv certifikata, potpisnicima koji su to izričito zatražili, za koje je utvrđena netočnost ili nepotpunost podataka u evidenciji certifikata i za koje je primljena službena obavijest o smrti ili o gubitku poslovne sposobnosti.

Kvalificirani certifikati izdani od davatelja usluga sa sjedištem u Europskoj uniji jednako su valjani kao i kvalificirani certifikati izdani u Hrvatskoj. Usluge certificiranja mogu obavljati i drugi davatelji usluga certificiranja sa sjedištem u inozemstvu izvan EU-a ako ispunjavaju uvjete propisane Zakonom o elektroničkom potpisu, ako su dobrovoljno akreditirani za izdavanje kvalificiranih certifikata, ako domaći izdavalac jamči za takve certifikate i ako je davatelj usluga priznat na temelju bilateralnog ili multilateralnog sporazuma između EU-a i trećih zemalja.

8.3.5 Elektronički potpis u obliku XML

Elektronički potpis u obliku XML naziva se jednostavno **XML Signature**, a često i skraćeno **XMLDsig**, **XML-DSig** ili **XML-Sig**. Potpis u obliku XML definiran je sintaksom u jeziku XML za elektronički potpis, koja je objavljena kao preporuka W3C-a **XML Signature Syntax and Processing**³³². Elektronički potpisi u obliku XML koriste se u mnogim drugim tehnologijama, kao što su SOAP, SAML i mnoge druge. Propisivanjem pravila sintakse i obrade elektroničkog potpisa u obliku XML osigurava se izvornost i cjelovitost poruke. Elektronički se potpisi mogu primijeniti na bilo koji oblik elektroničkog, odnosno digitalnog sadržaja, uključujući dokumente u jeziku XML, ali i bilo koji objekt dohvatljiv putem URL-a. Potpisom u obliku XML može se potpisati objekt izvan XML dokumenta, što se onda zove **odijeljeni ili odvojeni potpis** (engl. *detached signature*). Osim toga može se potpisati dio dokumenta koji sadržava te se tada naziva **omotani potpis** (engl. *enveloped signature*), a ako se potpisuju podaci u samom potpisu, onda se naziva **omotački ili omotavajući potpis** (engl. *enveloping signature*).

Osnovni proces stvaranja potpisa u obliku XML je sljedeći: poruka ili podatkovni objekt se obrađuju i rezultat te obrade stavlja se u element zajedno s drugim podacima. Taj se element obrađuje i kriptografski potpisuje. Na kraju se potpis u obliku XML stavlja u element Signature. Prikazat ćemo jedan oblik elektroničkog potpisa u jeziku XML:

```
<SignatureId="MojPotpis" xmlns="http://www.w3.org/2000/09/xmldsig#">
  <SignedInfo>
    <CanonicalizationMethod Algorithm="http://www.w3.org/2006/12/xml-c14n11"/>
    <SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#dsa-sha1"/>
    <Reference URI="http://www.w3.org/TR/2000/REC-xhtml1-2000126/">
      <Transforms>
        <Transform Algorithm="http://www.w3.org/2006/12/xml-c14n11"/>
      </Transforms>
    <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
    <DigestValue>dGhprzTRtcyBpcyBub3QgYSBzaWduYXR1cmUK...</DigestValue>
  </SignedInfo>
</SignatureId>
```

³³² XML Signature Syntax and Processing, Second Edition, W3C Recommendation, W3C, 2008-06-10, <http://www.w3.org/TR/xmldsig-core/>

```

        </Reference>
    </SignedInfo>
    <SignatureValue>...</SignatureValue>
    <KeyInfo>
        <KeyValue>
            <DSAKeyValue>
                <P>...</P><Q>...</Q><G>...</G><Y>...</Y>
            </DSAKeyValue>
        </KeyValue>
    </KeyInfo>
    <Object>...</Object>
</Signature>

```

Iz prethodnog je koda primjetno da se element Signature definira u prostoru naziva <http://www.w3.org/2000/09/xmldsig#>. Element SignedInfo sadržava reference na potpisane podatke i specificira algoritam koji se koristio. Elementa Reference može biti više i svaki predstavlja resurs koji je potpisan. Dodatno se može definirati transformacija uporabom elementa Transforms. Element DigestMethod navodi algoritam za generiranje sažetka koji se koristi, dok element DigestValue sadržava sam sažetak. Element SignatureValue sadržava rezultat potpisivanja. Opcionalni element KeyInfo omogućava slanje i ključeva za verifikaciju potpisa, najčešće u obliku X.509 certifikata, a opcionalni element Object sadržava potpisane podatke kod omotavajućeg potpisa.

8.4 Elektronička (digitalna) omotnica

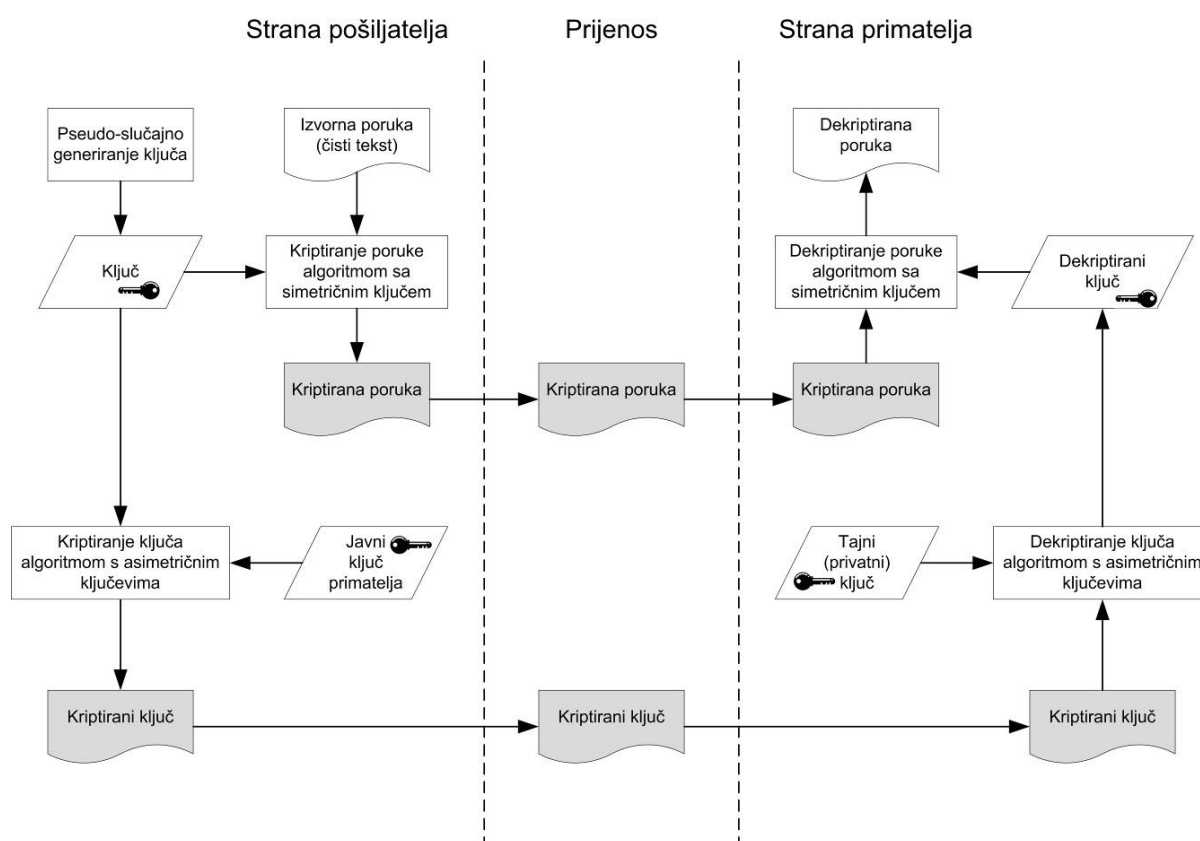
Elektronička ili digitalna omotnica (engl. *digital envelope*, *electronic envelope*) je slična stvarnoj omotnici ili koverti koja štiti poruku koja se u njoj nalazi, dok se na vanjskoj strani vidi samo informacija o primatelju (adresa). U pozadini se koriste kriptografske usluge algoritama simetričnog ključa, ali i algoritama asimetričnih ključeva i infrastrukture javnih ključeva. Elektronička se omotnica, za razliku od poruke s elektroničkim potpisom gdje poruka ne mora biti kriptirana, sastoji od kriptirane poruke i kriptiranog ključa koji se koristi za dekriptiranje poruke. Na ovaj se način rješava problem distribucije ključa kojim je kriptirana poruka, jer se i on sam kriptira korištenjem kriptografije javnog ključa, odnosno para javnog i tajnog ključa, te šalje zajedno s porukom.

Ova se metoda koristi jer se za kriptiranje same poruke može koristiti neki od kriptografskih algoritama sa simetričnim ključem, koji su znatno brži, pogotovo kod velike količine podataka u poruci. No ključ koji se koristi za kriptiranje i dekriptiranje poruke nekako se mora distribuirati drugoj strani. Jedan od načina je da se sam ključ kriptira uporabom nekog od kriptografskih algoritama s asimetričnim ključevima. Ovdje se ne narušava učinkovitost procesa kriptiranja i dekriptiranja jer se sporijim algoritmom kriptira samo ključ koji je podatak male duljine. Na kraju se i poruka i ključ zajedno šalju na drugu stranu gdje se dekriptiraju, prvo ključ, a zatim i poruka.

Primjena ovog načela jednostavno je izvediva kod kriptiranja elektroničke pošte, a nema ni problema kod slanja na više primatelja, jer se ključ kojim je kriptirana poruka treba samo kriptirati onoliko puta koliko ima primatelja. Ključ se također može promijeniti pri svakoj sljedećoj poruci, odnosno komunikaciji, ali i drugom, željenom rjeđom frekvencijom obnavljanja.

Kod generiranja elektroničke omotnice pošiljatelj prvo odabranim ključem kriptira poruku koju želi poslati, i to kriptografskim algoritmom sa simetričnim ključem. Ključ koji se pritom koristi generira se slučajno, odnosno najčešće pseudoslučajnim generatorom. Pošiljatelj zatim kriptira taj ključ kriptografskim algoritmom s asimetričnim ključevima korištenjem javnog ključa primatelja. Ovime se izbjegava problem distribucije ključa. Nakon toga pošiljatelj primatelju zajedno šalje kriptiranu poruku i kriptirani ključ.

S druge strane primatelj prvo dekriptira primljeni ključ korištenjem kriptografskog algoritma s asimetričnim ključevima tako da upotrijebi svoj privatni (tajni) ključ. Tim postupkom dobiva ključ koji mu služi da korištenjem kriptografskog algoritma sa simetričnim ključem dekriptira poruku. Čitav se proces može prikazati i grafički (Slika 8.3).



Slika 8.3: Elektronička omotnica

Kao što je već spomenuto, proces je učinkovit jer se prilikom kriptiranja i dekriptiranja poruke, koja u smislu količine podataka može biti velika, koristi kriptografski algoritam sa simetričnim ključem, koji je načelno brži od kriptografskog algoritma s asimetričnim ključevima. Sporiji kriptografski algoritam s asimetričnim ključevima koristi se samo za kriptiranje i dekriptiranje ključa. Utvrđena je i tajnost, jer samo primatelj posjeduje privatni ključ za dekriptiranje ključa koji dekriptira poruku.

8.5 Napadi

Kao što ste već vidjeli u priručniku „Sigurnost informacijskih sustava“, kriptanalizu čine različite metode napada na kriptografske algoritme, odnosno kriptirane poruke, kojima je cilj otkriti izvornu poruku. U kontekstu kriptanalitičkih napada navedeni su sljedeći modeli (engl. *attack models*) ili tipovi napada:

- **napad poznatim izvornim tekstom (KPA, engl. *known plaintext attack*)**, gdje napadač poznaje samo izvorni tekst i odgovarajući kriptirani tekst,
- **napad odabranim izvornim tekstom (CPA, engl. *chosen plaintext attack*)**, gdje napadač odabire izvorni tekst i može pokrenuti metodu kriptiranja (te time dobiti i odgovarajući kriptirani tekst),
 - **adaptivan napad odabranim izvornim tekstom** (engl. *adaptive chosen plaintext attack*), gdje napadač na temelju prethodnih kriptanaliza odabire sljedeći izvorni tekst i može pokrenuti metodu kriptiranja,
- **napad poznatim kriptiranim tekstom (COA, engl. *ciphertext-only attack* ili *known ciphertext attack*)**, gdje napadač poznaje samo kriptirani tekst,
- **napad odabranim kriptiranim tekstom (CCA, engl. *chosen ciphertext attack*)**, gdje napadač odabire kriptirani tekst i može pokrenuti metodu dekriptiranja (te time dobiti i odgovarajući izvorni tekst),
 - **adaptivan napad odabranim kriptiranim tekstom** (engl. *adaptive chosen ciphertext attack*), gdje napadač na temelju prethodnih kriptanaliza odabire sljedeći kriptirani tekst i može pokrenuti metodu dekriptiranja,
- **napad korištenjem srodnih ključeva** (engl. *related-key attack*), gdje napadač uporabom srodnih ključeva, koji nisu nužno poznati, ali je poznata njihova srodnost (npr. promjena u jednom bitu ili znaku), kriptira poruke i uspoređuje rezultate.

Osim navedenih kriptanalitičkih napada koristi se i niz drugih, od kojih ćemo spomenuti najpoznatije:

- **napad pogađanjem ključeva ili lozinki grubom silom** (engl. *brute force attack*) – metoda isprobavanja svih mogućih ključeva, vrlo spora i neučinkovita, ali jednostavna i pouzdana metoda napada (ako se ima dovoljno vremena),
- **napad rječnikom** (engl. *dictionary attack*) – varijanta napada grubom silom koji ne koristi sve moguće ključeve, već isprobava ključeve iz određenog rječnika, odnosno ključeve s većom vjerojatnošću da budu pogođeni prema nekoj metodi odabira.

Postoji i niz drugih sigurnosnih ugrožavanja i napada koji nisu nužno usmjereni na metode kriptanalize i određeni algoritam ili poruke te se koriste i u kombinaciji s drugim metodama. Navest ćemo najpoznatije oblike narušavanja sigurnosti i napada vezanih uz računala i usluge:

- **Prisluškivanje** (engl. *eavesdropping, tapping, interception*) – odnosi se na praćenje mrežnog (često nekriptiranog) prometa i preuzimanje bitnih podataka.
- **Ubacivanje u komunikaciju ili napad „čovjeka u sredini“ (MITM, engl. *man-in-the-middle attack*)** – oblik aktivnog prisluškivanja gdje su moguće promjene poruka koje se prenose ili postavljanja lažne komunikacije i slanje odnosno primanje novih poruka bez da je to vidljivo bilo kojoj strani uključenoj u komunikaciju.
- **Lažno predstavljanje drugog korisnika** (engl. *impersonation of user*) – koriste se autentikacijski mehanizmi drugog korisnika kako bi se pristupilo sustavu. Najčešće se

izvodi krađom autentikacijskih uređaja (pametne kartice, tokena) ili podataka (PIN-a, lozinke, identiteta), probijanjem sigurnosti sustava ili pogađanjem korisničkih lozinki.

- **Lažno predstavljanje drugog računala** (engl. *impersonation of computer*) – najčešće poslužitelja ili klijenta u komunikaciji, odnosno pružatelja usluge. Često se lažiraju mrežne IP i MAC adrese računala, a mogu se koristiti metode napada MITM i promjene poruka te različiti oblici pogađanja certifikata i ključeva (napad grubom silom, CCA, CPA ...).
- **Lažno predstavljanje usluge** (engl. *phishing*) – izvodi se metodama gdje se preusmjerava korištenje usluge na napadačevo računalo ili se mijenja povratna poruka od usluge korištenjem napada MITM. Najčešće se koristi za pribavljanje lozinke i informacija, kao što su podaci s kreditnih kartica i financijski podaci, metodama koje uključuju lažne poruke elektroničke pošte, lažne poruke IM (engl. *instant messaging*), lažna sjedišta Weba ili lažne sigurnosne usluge (elektronička plaćanja, upite za lozinke i slično).
- **Uskraćivanje usluge (DoS, engl. *denial of service*)** – zasniva se na uskraćivanju usluge kroz razne oblike preopterećenja poslužitelja, najčešće generiranjem velike količine prometa, odnosno poziva poslužitelja. Moguće je privremeno ili trajno uskraćivanje usluge, kao što je određeno sjedište Weba, radi nanošenja štete. Ako je usluga i dalje dostupna, poslužitelj pod napadom vrlo sporo obrađuje stvarne zahtjeve jer troši velike količine resursa na lažne napade. Posljedica može biti zauzeće sve postojeće memorije, a često i rušenje sustava, aplikacije i operacijskog sustava. **Raspodijeljena varijanta napada uskraćivanjem usluge DDoS** (engl. *distributed denial of service*) koristi višestruka računala koja napadaju, katkad tisuće različitih računala koja su zaražena programom koji izvodi napad.
- **Odugovlačenje ili ponavljanje poruke** – izvodi se napadom ponavljanjem poruka (engl. *replay attack*) koji postojeće poruke, odnosno podatke, ponavlja niz puta ili namjerno odugovlači s isporukom.
- **Zamjena poruke** – izvodi se korištenjem napada zamjenom poruke ili dijela poruke (engl. *substitution attack*) podacima koje je napadač izmijenio u odnosu na izvornu poruku.

Svi ovi napadi i ugrožavanja sigurnosti rješavaju se različitim sigurnosnim metodama koje omogućavaju povjerljivost, cjelovitost, dostupnost, izvornost i neporecivost na razini poruka, usluga i komunikacije. Zaštita se izvodi na više razina, a možemo je podijeliti na:

- zaštitu sustava
- zaštitu komunikacije
- zaštitu poruka.

Zaštita na razini sustava izvodi se na razne načine. Prije svega stvaranjem sigurnosne infrastrukture i arhitekture računalne mreže uporabom demilitariziranih zona ili DMZ-a (engl. *demilitarized zones*) te vatrozidova (engl. *firewalls*), što onemogućava upade u lokalnu mrežu. Treba zaštititi i sve bežične mrežne pristupne točke (WLAN), modemske ulaze i žične mrežne priključke. Na mreži se stalno trebaju pratiti sve neuobičajene mrežne aktivnosti te se trebaju koristiti antivirusni i drugi sigurnosni alati. Postoje razni oblici zaštite sustava, od razine operacijskog sustava, preko mrežne komunikacije, zaštite datoteka i baza podataka, do razine zaštite različitih aplikacijskih sustava. Kod informacijskih sustava treba zaštititi sve javne usluge i sučelja, zapisivati aktivnosti nad sustavom i izvoditi sigurnosna testiranja sustava.

Zaštita komunikacijskog kanala zasniva se na sigurnosti mrežne komunikacije uporabom različitih protokola, kao što su SSL (engl. *Secure Sockets Layer*), HTTPS i IPSEC, a često i stvaranjem virtualnih privatnih mreža VPN (engl. *virtual private network*) te softverskim ili sklopovskim kriptiranjem komunikacije, npr. korištenjem uređaja HSM (engl. *High Security Modules*). Općenito, svi pristupi informacijskim sustavima i njegovim sučeljima trebaju koristiti autentikacijske i autorizacijske mehanizme, i to na razini korisnika, klijenta i poslužitelja.

Zaštita poruka najčešće se ostvaruje korištenjem elektroničkog potpisa i elektroničke omoćnice te kriptiranjem i generiranjem sažetaka opisanih u prethodnim potpoglavljima.

8.6 Sigurnost usluga

Sigurnost elektroničkog poslovanja ovisi o sigurnosti svih njenih dijelova. U prvom dijelu ovog poglavlja navedene su sigurnosne norme elektroničkog potpisa koje se koriste kod sigurnosti poslovanja. No, u kontekstu sigurnosti usluga, izdvajaju se i norme sigurne interoperabilnosti, norme vezane uz jezik XML i norme vezane uz sigurnost usluga Web Services.

Najvažnije **norme sigurnosti korištenja vezane uz jezik XML**, osim već spomenute norme elektroničkog potpisa u XML obliku (XML-DSig), uključuju sintaksu i obradu pri kriptiranju XML-a (XML-Enc) opisanu u potpoglavlju 0, označni jezik sigurnosnih tvrdnji SAML opisan u potpoglavlju 8.6.2, kontrolu pristupa (XACML) te upravljanje ključevima (XKMS). Jezik **XML Access Control Markup Language**³³³ (XACML) koristi pravila za kontrolu pristupa XML dokument. Sustav može

odbiti ili prihvatiti zahtjev korisnika za pristupom, ali može se dopustiti ili odbiti pristup dijelu XML dokumenta do razine elementa. Specifikacija **XML Key Management**³³⁴ (XKMS) definira protokole distribucije i registracije javnih ključeva. Konkretno se definiraju poruke za zahtjev, odgovor, registraciju, povlačenje i promjenu ključa. Specifikacija XKMS sastoji se od dvaju dijelova, prvi pod nazivom XML Key Information Service Specification (X-KISS), koji definira protokole obrade informacija o ključu, i drugi pod nazivom XML Key Registration Service Specification (X_KRSS), koji definira usluge registracije ključeva uporabom pouzdanih registracijskih poslužitelja.

Norme vezane uz sigurnost usluga temeljenih na skupu tehnologija Web Services uključuju normu WS-Security, opisanu u potpoglavlju 8.6.3, normu WS-I Basic Security Profile, opisanu u potpoglavlju 8.6.4, te norme WS-ReliableMessaging, WS-Policy, WS-Trust, WS-Privacy, WS-SecureConversation, WS-Federation i WS-Authorization, koje su dio okvira nadogradnji **WS-Security** i koje su većinom još u razvoju. Norma **WS-ReliableMessaging**³³⁵ omogućava pouzdanu dostavu poruka i definira model pouzdane isporuke poruka (engl. *Reliable Messaging Model*). Norma **WS-Authorization** služiti će kod autorizacija i kontrole

³³³ XACML 2.0 Specification Set, OASIS, 2005-02-01, http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml#XACML20

³³⁴ XML Key Management Specification (XKMS 2.0), W3C Recommendation, W3C, 2005-06-28, <http://www.w3.org/TR/xkms2/>

³³⁵ Web Services Reliable Messaging (WS-ReliableMessaging) Version 1.2, OASIS Standard, 2009-02-02, <http://docs.oasis-open.org/ws-rx/wsrn/v1.2/wsrn.html>

pristupa. Norma **WS-Federation**³³⁶ postoji i specificira modele i načine uspostave federacije između različitih domena. Norma **WS-SecureConversation**³³⁷ također je objavljena i definira osnovne mehanizme semantike sigurne razmjene višestrukih poruka. Objavljena norma **WS-Trust**³³⁸ opisuje metode izdavanja, obnavljanja i validacija sigurnosnih znački (engl. *token*) te stvaranje odnosa povjerenja između domena. Norma **WS-Policy**³³⁹ omogućava uporabu sigurnosnih i drugih politika kod usluga, a norma **WS-Privacy** još je u razvoju i omogućit će uporabu politika privatnosti.

8.6.1 Kriptiranje sadržaja u obliku XML

Metoda kriptiranja sadržaja i zapisa u obliku XML naziva se **XML kriptiranje** ili **enkripcija** (engl. *XML encryption*) i opisana je u preporuci W3C-a **XML Encryption Syntax and Processing**³⁴⁰. Iako se metoda kriptiranja može primijeniti na bilo koji oblik podatka, a ne samo XML dokument, strukturu u jeziku XML ili pojedini element, s obzirom na to da se sam rezultat kriptiranja zapisuje kao struktura u jeziku XML, nazvana je **XML Encryption** ili skraćeno **XML-Enc**. Rezultat kriptiranja je sam kriptirani sadržaj u obliku kriptiranoga podatkovnog XML elementa, ali i dodatne informacije koje primatelju omogućavaju dekriptiranje.

Slično kao i kod elektroničkog potpisa u jeziku XML, koristi se određena struktura koja sadržava podatke o podacima, metodi kriptiranja, ključevima i druge značajke. Slijedi primjer osnovnog oblika strukture:

```
<EncryptedData Id="..." xmlns="..." Type="..." MimeType="..." Encoding="...">
  <EncryptionMethod/>
  <ds:KeyInfo>
    <EncryptedKey/>
    <AgreementMethod/>
    <ds:KeyName/>
    <ds:RetrievalMethod/>
  </ds:KeyInfo>
  <CipherData>
    <CipherValue>...</CipherValue>
    <CipherReference URI/>
  </CipherData>
  <EncryptionProperties>...</EncryptionProperties>
</EncryptedData>
```

Glavni elementi su **EncryptedData** ili **EncryptedKey**, koji sadržavaju podstrukturu u kojoj je bitno uočiti element **CipherData** i podelement **CipherValue** koji sadrži kriptirane podatke.

³³⁶ Web Services Federation Language, BEA, BMC, CA, IBM, Layer 7 Microsoft, Novell i Verisign, <http://www.ibm.com/developerworks/library/specification/ws-fed/>

³³⁷ WS-SecureConversation 1.4, OASIS standard, OASIS, 2009-02-02, <http://docs.oasis-open.org/ws-sx/ws-secureconversation/v1.4/ws-secureconversation.html>

³³⁸ WS-Trust 1.4, OASIS standard, OASIS, 2009-02-02, <http://docs.oasis-open.org/ws-sx/ws-trust/v1.4/ws-trust.html>

³³⁹ Web Services Policy 1.5 – Framework, W3C Recommendation, W3C, 2007-09-04, <http://www.w3.org/TR/ws-policy/>

³⁴⁰ XML Encryption Syntax and Processing, W3C Recommendation, W3C, 2002-12-10, <http://www.w3.org/TR/xmlenc-core/>

Za primjer ćemo kriptirati dokument koji sadržava neku narudžbu i podatke o plaćanju:

```
<PurchaseOrder>
  <Order>
    <Item>Knjiga</Item>
    <Id>1234567890</Id>
    <Quantity>1</Quantity>
  </Order>
  <Payment>
    <CardId>1234-567890-1234</CardId>
    <CardName>Diners</CardName>
    <ValidDate>2011-09-01</ValidDate>
  </Payment>
</PurchaseOrder>
```

Ako se želi kriptirati cijeli navedeni odsječak, odnosno XML dokument, to bi izgledalo ovako:

```
<?xml version="1.0" ?>
<EncryptedData xmlns="http://www.w3.org/2001/04/xmlenc#"
  Type="http://www.isi.edu/in-notes/iana/assignments/media-types/text/xml">
  <CipherData>
    <CipherValue>A1B2C3D4...</CipherValue>
  </CipherData>
</EncryptedData>
```

Glavni element je EncryptedData, koji sadržava i prostor naziva koji se koristi kod kriptiranja i tip podataka u kojem su bili podaci prema definicijama naziva IANA, a kriptirani podaci pojavljuju se kao sadržaj elementa CipherValue unutar elementa CipherData.

Na isti se način mogu kriptirati bilo kakvi drugi dokumenti koji nisu zapisani u obliku XML, kao što su dokumenti uređivača tekstova, dokumenti tabličnih kalkulacija, dokumenti u obliku PDF, slike u JPEG, PNG ili GIF obliku i mnogi drugi, samo se pritom treba pravilno oblikovati tip podatka, odnosno dokumenta koji je kriptiran prema definicijama naziva IANA.

Ako se želi kriptirati samo jedan element ili dio strukture XML dokumenta, onda se to može prikazati sljedećim primjerom:

```
<?xml version="1.0" ?>
<PurchaseOrder>
  <Order>
    <Item>Knjiga</Item>
    <Id>1234567890</Id>
    <Quantity>1</Quantity>
  </Order>
  <EncryptedData xmlns="http://www.w3.org/2001/04/xmlenc#"
    Type="http://www.w3.org/2001/04/xmlenc#Element">
    <CipherData>
      <CipherValue>A1B2C3D4E5F6...</CipherValue>
    </CipherData>
  </EncryptedData>
</PurchaseOrder>
```

Ovdje se vidi da dio strukture koji se odnosi na narudžbu (element Order) nije kriptiran, dok je kriptiran samo dio koji se odnosi na podatke o plaćanju (element Payment). Umjesto elementa Payment pojavljuje se element EncryptedData, koji sadržava i prostor naziva, a tip

podatka koji je kriptiran postavljen je na `http://www.w3.org/2001/04/xmlenc#Element`, što je definirano preporukom. Kriptirani podaci pojavljuju se i dalje kao sadržaj elementa `CipherValue` unutar elementa `CipherData`.

Postoji i mogućnost kriptiranja samo sadržaja pojedinog elementa, pa bi to u slučaju broja kartice, odnosno elementa `CardID`, izgledalo ovako:

```
<?xml version="1.0" ?>
<PurchaseOrder>
  <Order>
    <Item>Knjiga</Item>
    <Id>1234567890</Id>
    <Quantity>1</Quantity>
  </Order>
  <Payment>
    <CardId>
      <EncryptedData xmlns="http://www.w3.org/2001/04/xmlenc#"
        Type="http://www.w3.org/2001/04/xmlenc#Content">
        <CipherData>
          <CipherValue>A1234567890B12345...</CipherValue>
        </CipherData>
      </EncryptedData>
    </CardId>
    <CardName>Diners</CardName>
    <ValidDate>2011-09-01</ValidDate>
  </Payment>
</PurchaseOrder>
```

U prethodnom primjeru primjećuje se da je tip podatka koji je kriptiran postavljen na `http://www.w3.org/2001/04/xmlenc#Content`.

8.6.2 Označni jezik sigurnosnih tvrdnji – SAML

Označni jezik sigurnosnih tvrdnji SAML (engl. *Security Assertion Markup Language*) je norma za razmjenu autentikacijskih i autorizacijskih podataka između sigurnosnih domena, koju je razvila organizacija OASIS. Jezik SAML se zapravo zasniva na nizu drugih normi, koje uključuju jezik XML, XML sheme, XML Signature, XML Encryption, protokole HTTP i SOAP i druge. Tvrdnje u obliku XML definiraju protokole, povezivanja i profile. Naziv **jezgra jezika SAML** (engl. *SAML Core*) odnosi se na općenitu sintaksu i semantiku pisanja tvrdnji, kao i na protokol koji se koristi za prijenos tvrdnji između sustava u smislu objekata koji se prenose. Povezivanja (engl. *bindings*) korištenjem jezika SAML definiraju kako se zahtjevi i odgovori preslikavaju na standardne poruke komunikacijskih protokola, kao što su protokoli SOAP i HTTP. Profili SAML su zapravo konkretni slučajevi korištenja kombinacija tvrdnji, protokola i povezivanja. Jedna od izravnih koristi norme SAML je korištenje jednostruke prijave SSO (engl. *Single Sign On*).

Tvrdnje u jeziku SAML, koje se najčešće šalju od davatelja identifikacije (engl. *identity provider*) prema davateljima usluga (engl. *service provider*), sadržavaju izjave (engl. *statement*) koje se dijele na tri vrste:

- izjave o autentikaciji (engl. *authentication statements*),
- izjave o atributima (engl. *attribute statements*),
- izjave o odlukama autorizacije (engl. *authorization decision statements*).

U kontekstu sigurnosti SAML koristi već postojeće mehanizma, kao što su SSL ili TLS na razini prijenosa podataka te XML Encryption i XML Signature na razini poruke.

8.6.3 Protokol i norma WS-Security

Protokol i norma **WS-Security** ili skraćeno **WSS**, što dolazi od **Web Services Security**, je nadogradnja protokola SOAP koji osigurava sigurnost primjene tehnologija Web Services. Dvije godine nakon što je nastao skup tehnologija Web Services tvrtke IBM, Microsoft i VeriSign su predložili specifikaciju sigurnosnog protokola temeljenog na protokolu SOAP. Danas je to protokol WS-Security, koji je objavljen kao norma tijela OASIS, trenutno u inačici 1.1³⁴¹. Osnovna ideja protokola i norme WS-Security je očuvanje cjelovitosti i povjerljivosti poruka, a pritom koristi mnoge druge sigurnosne norme i protokole, kao što su SAML, Kerberos i elektronički certifikati X.509. U svojoj osnovi protokol WS-Security opisuje kako pripojiti elektronički potpis, zaglavlje i sigurnosne tokene na poruku protokola SOAP. Za to se koristi prilagodljivi skup mehanizama i postojećih sigurnosnih protokola. Norma WS-Security opisuje tri osnovna mehanizma:

- **potpisivanje poruka SOAP** kako bi se očuvala cjelovitost i neporecivost,
- **kriptiranje poruka SOAP** kako bi se osigurala tajnost,
- **pridruživanje sigurnosnih tokena** porukama SOAP.

S obzirom na to da norma WS-Security omogućava korištenje različitih postojećih sigurnosnih mehanizama i protokola, moguće je koristiti razne oblike potpisa, različite algoritme kriptiranja, višestruke domene povjerenja i različite oblike sigurnosnih tokena, od jednostavnih korisničkih imena i lozinki, preko certifikata X.509, Kerberos tokena i tvrdnji SAML, do korisnički definiranih tokena.

Norma WS-Security zapravo uključuje više specifikacija i dokumenata:

- SOAP Message Security 1.1 – poboljšanje poruke SOAP očuvanjem cjelovitosti i autentikacijom, kao mehanizmima prihvata sigurnosnih tokena,
- UsernameToken Profile 1.1 – korištenje identifikacije korisničkim imenom i proizvoljnom lozinkom,
- X.509 Token Profile 1.1 – korištenje certifikata X.509,
- SAML Token Profile 1.1 – korištenje tvrdnji jezika SAML kao sigurnosnih tokena,
- Kerberos Token Profile 1.1 – korištenje Kerberos tokena,
- Rights Expression Language Token Profile 1.1 – korištenje jezika REL,
- SOAP with Attachments Profile 1.1 – korištenje poruka SwA.

³⁴¹ WS-Security Core Specification 1.1, Web Services Security: SOAP Message Security 1.1 (WS-Security 2004), OASIS Standard Specification, OASIS, 2006-02-01, <http://www.oasis-open.org/committees/download.php/16790/wss-v1.1-spec-os-SOAPMessageSecurity.pdf>

No primjena norme WS-Security ima i svojih mana, jer dodaje dosta podataka u samu poruku SOAP, pa obrada poruke traje dulje i troši više procesorskog vremena. Ovo je osobito izraženo kad se istovremeno koriste mehanizmi potpisivanja i kriptiranja.

8.6.4 Specifikacija WS-I Basic Security Profile

Skup nevlasničkih specifikacija tehnologija i normi Web Services koji je 2007. godine donijela organizacija Web Services – Interoperability (WS-I) naziva se **WS-I Basic Security Profile**³⁴², a trenutno je aktualna inačica 1.1. Skup specifikacija WS-I Basic Security Profile zapravo je interoperabilni profil, odnosno niz specifikacija vezanih uz tehnologije i norme Web Services koji omogućavaju siguran prijenos i razmjenu poruka SOAP. Usklađenost sa specifikacijama izvodi se poštovanjem niza zahtjeva i kriterija opisanih u ovom dokumentu. U dokumentu postoji niz izjava označenih kao „Rnnnn“, gdje je „nnnn“ broj pojedine izjave, koje definiraju što se točno mora, smije, preporučuje, može i slično. Za većinu izjava postoji primjer i objašnjenje, a često i dopune.

Primjer izjave bio bi:

R3103 A SIGNATURE SHOULD be a Detached Signature as defined by the XML Signature specification.
Detached signatures are encouraged.

ili prevedeno na hrvatski:

R3103 POTPIS BI TREBAO biti odijeljeni potpis kao što je definirano u specifikaciji XML Signature.
Odijeljeni potpisi se potiču (za korištenje).

S obzirom na to da se skup specifikacija WS-I Basic Security Profile odnosi na druge specifikacije, ovo se smatra zahtjevima koje treba poštovati kako bi izgrađene **usluge bile interoperabilne**.

Konkretna specifikacija na koje se referencira uključuju:

- XML Signature Syntax and Processing
- XML Encryption Syntax and Processing
- Specifikacije norme WS-Security
 - Web Services Security: SOAP Message Security 1.1
 - Web Services Security: UsernameToken Profile 1.1
 - Web Services Security: Rights Expression Language (REL) Token Profile 1.1
 - Web Services Security: Kerberos Token Profile 1.1
 - Web Services Security: SAML Token Profile 1.1
 - Web Services Security: X.509 Certificate Token Profile 1.1
 - Web Services Security: SOAP Messages with Attachments (SwA) Profile 1.1

³⁴² WS-I Basic Security Profile Version 1.1, WS-I, 2010-01-24, <http://www.ws-i.org/Profiles/BasicSecurityProfile-1.1.html>

- Specifikacije korištenja sigurnosnih protokola i certifikata
 - RFC 2818: HTTP over TLS
 - RFC 2246: The TLS Protocol Version 1.0
 - The SSL Protocol Version 3.0
 - RFC2459: Internet X.509 Public Key Infrastructure Certificate and CRL Profile
- Neke druge specifikacije
 - Simple SOAP Binding Profile Version 1.0 (SSBP1.0)
 - XPointer Framework, W3C Recommendation, 25 March 2003
 - Information technology "Open Systems Interconnection" The Directory: Public-key and attribute certificate frameworks Technical Corrigendum 1
 - RFC4514 Lightweight Directory Access Protocol: String Representation of Distinguished Names
 - Attachments Profile Version 1.0 (AP1.0).

9. Poglavlje: Primjer interoperabilnosti

U ovom poglavlju naučit ćete:

- ✓ primjer interoperabilnosti iz prakse

9.1 Primjer implementacije interoperabilnosti – NIAS autentikacija

Kao što smo već prije objasnili, NIAS je informacijsko-tehnološki sustav za središnju identifikaciju i autentikaciju korisnika u pristupu elektroničkim javnim uslugama u umreženoj upravi, a za upravljanje NIAS-om nadležno je Ministarstvo uprave kao središnje tijelo državne uprave nadležno za poslove e-Hrvatske (MURH), dok je posao uspostave i operativnog vođenja NIAS-a povjeren Financijskoj agenciji (FINA).

U sklopu uspostave NIAS-a, MURH i FINA su izradili poslovno-tehničku dokumentaciju, potrebnu za lakše uključivanje tijela javnog sektora i drugih subjekata u sustav NIAS, a koja uključuje:

- program razvoja elektroničkih usluga u okviru projekta e-Građani³⁴³
- protokol rada NIAS-a za e-Građane i e-Poslovanje³⁴⁴
- kriterije za određivanje razine osiguranja kvalitete autentikacije u okviru NIAS-a³⁴⁵
- tehničke specifikacije za integraciju vjerodajnica u sustav NIAS³⁴⁶
- tehničke specifikacije za integraciju e-usluga u sustav NIAS³⁴⁷
- tehničke specifikacije korisničkih atributa za e-Građane³⁴⁸
- tehničke specifikacije korisničkih atributa za e-Poslovanje³⁴⁹
- protokol rada e-Ovlaštenja³⁵⁰
- tehničke specifikacije za dohvat autorizacijskih podataka³⁵¹
- tehničke specifikacije za integraciju registracijskog obrasca u e-Poslovanje³⁵²
- tehničke specifikacije za „Single Sign-Out” e-usluge³⁵³
- i druge dokumente.



Većina ovog potpoglavlja izravno se odnosi na gore navedene dokumente, odnosno preuzima određene opise te u slučaju potencijalnih promjena svakako treba obratiti pažnju postoje li novije inačice dokumenata od onih koje su ovdje navedene.

Osim navedenog, postoji i skup dokumenata za operativnu provedbu i funkcioniranje sustava, koji čine dokumenti, kao što su mnogi obrasci, procedure, mišljenja, odluke, sporazumi, scenariji i nalozi, koje nećemo ovdje pojedinačno navoditi, a njihov popis je dan u navedenoj dokumentaciji.

³⁴³ Program razvoja elektroničkih usluga u okviru projekta e-Građani, verzija 1.0, rujan 2013.

³⁴⁴ Protokol rada NIAS-a za e-Građane i e-Poslovanje, verzija 1.6.2., 31.07.2019.

³⁴⁵ Kriteriji za određivanje razine osiguranja kvalitete autentikacije u okviru NIAS-a, verzija 1.2, 06.12.2013.

³⁴⁶ Tehnička specifikacija za integraciju vjerodajnica u sustav NIAS

³⁴⁷ Tehnička specifikacija za integraciju e-usluga u sustav NIAS, verzija 1.1, 02.12.2013.

³⁴⁸ Tehnička specifikacija korisničkih atributa za e-Građane

³⁴⁹ Tehnička specifikacija korisničkih atributa za e-Poslovanje

³⁵⁰ Protokol rada e-Ovlaštenja, verzija 0.2, 31.07.2019.

³⁵¹ Tehnička specifikacija za dohvat autorizacijskih podataka

³⁵² Tehnička specifikacija za integraciju registracijskog obrasca u e-Poslovanje

³⁵³ Tehnička specifikacija za „Single Sign-Out” e-usluge, verzija 1.0, 20.11.2013.



Veći dio dokumentacije o Projektu e-Građani i Projektu e-Poslovanje, javno je objavljen na internetskim stranicama Vlade Republike Hrvatske u okviru platforme e-Građani i e-Poslovanje.

9.1.1 Subjekti u procesu identifikacije i autentikacije

NIAS razlikuje nekoliko vrsta subjekata koje međusobno povezuje sa svrhom središnje autentikacije:

- izdavatelji elektroničkih vjerodajnica,
- pružatelje e-usluga,
- korisnike, odnosno prekogranične korisnike,
- čvor,
- e-Ovlaštenja i
- navigacijsku traku sustava e-Građani i e-Poslovanje.

Pritom NIAS ima ulogu posrednika između Korisnika, odnosno Prekograničnog korisnika u procesu prijave na e-usluge, neovisno o tome nalazi li se e-usluga u Republici Hrvatskoj ili u zemljama koje su se uključile u prekograničnu autentikaciju temeljem **Uredbe eIDAS**³⁵⁴, u kojem se slučaju razmjena poruka obavlja putem Čvora koristeći poruke XML sukladno normi SAML 2.0.

Dodatno, svaka e-usluga na platformi e-Poslovanja mora biti integrirana u NIAS, a ako e-usluga dodatno treba dobiti i autorizacijske podatke o autenticiranom Korisniku, potrebno je provesti i integraciju s podsustavom e-Ovlaštenja, sukladno Protokolu rada e-Ovlaštenja i pripadajućim Tehničkim specifikacijama.

Navigacijska traka sustava e-Građani i e-Poslovanje je posebni dio Portala e-Građani/e-Poslovanje koji se integrira unutar svake e-usluge te sadrži osnovne informacije o Korisniku koji je prijavljen na sustav (ime, prezime, OIB, razina sigurnosti vjerodajnice). Navigacijska traka omogućuje Korisniku da promijeni sustav djelovanja e-Građani ili e-Poslovanje te e-uslugu na koju se želi prijaviti. Dodatno, navigacijska traka sustava e-Poslovanje sadrži i informacije za koji poslovni subjekt Korisnik ima pravo djelovati (jedinствени identifikator i naziv poslovnog subjekta) te omogućuje promjenu poslovnog subjekta u čije ime Korisnik ima pravo djelovati.

9.1.2 Elektronički identiteti i vjerodajnice

Skup podataka o elektroničkom identitetu Korisnika koji NIAS prosljeđuje e-usluzi dovoljan je za jednoznačnu identifikaciju Korisnika iz evidencija Pružatelja atributa, kao što je primjerice Evidencija o osobnim identifikacijskim brojevima (OIB). Skup podataka o Prekograničnom korisniku koji NIAS prosljeđuje e-usluzi je reguliran Uredbom eIDAS, koja definira obvezni i

³⁵⁴ eIDAS – Uredba Europskog parlamenta i Vijeća br. 910/2014 o elektroničkoj identifikaciji i uslugama povjerenja za elektroničke transakcije na unutarnjem tržištu i stavljanju izvan snage Direktive 1999/93/EZ, 23.07.2014.

opcionalni skup podataka za autentikaciju, a zemlja pošiljateljica je obvezna poslati najmanje obvezni skup podataka.

Ključna karakteristika vjerodajnica u procesu autentikacije je njihova **sigurnosna razina**, odnosno razina osiguranja kvalitete autentikacije. Uredba eIDAS propisuje tri razine osiguranja identiteta za sredstva elektroničke identifikacije izdana u okviru prijavljenog sustava elektroničke identifikacije, a svaka od navedenih razina pruža odgovarajući stupanj pouzdanja u odnosu na traženi ili utvrđeni identitet osobe:

- **niska razina** – pruža ograničen stupanj pouzdanja
- **značajna razina** – pruža značajan stupanj pouzdanja
- **visoka razina** – pruža viši stupanj pouzdanja.

Integracijom vjerodajnice u NIAS, takva vjerodajnica postaje nacionalno priznata kao sredstvo elektroničke identifikacije u pristupu e-uslugama koje su integrirane u NIAS, a koje prihvaćaju njezinu razinu sigurnosti.



Uz ispunjavanje određenih uvjeta, pojedinim prihvatljivim izdaviteljima vjerodajnica tipa „korisničko ime/lozinka” omogućava se da njihovi već postojeći podaci o korisničkim računima budu prihvaćeni u središnji sustav ePASS³⁵⁵ koji nakon toga preuzima sve poslove vezane uz daljnje upravljanje elektroničkim identitetima tih Korisnika (Prihvat podataka o korisničkim računima drugih izdavalaca vjerodajnica u središnji sustav ePASS), iako ovi izdavalci nisu Klijenti NIAS-a jer se prihvat podataka radi u sustavu ePASS. Uz to, oni trajno prestaju s poslovima izdavanja takve vjerodajnice.

9.1.3 Pristup korisnika sustavu putem vjerodajnice

Da bi Korisnik pristupio NIAS-u, mora imati odgovarajuću vjerodajnicu koja je integrirana u sustav NIAS. Može se koristiti: jedna ili više vjerodajnica, uz jednostruku autentikaciju za pristup različitim e-uslugama koje se mogu nalaziti na više različitih web stranica, odnosno domena povezanih s NIAS-om (*Single Sign-on*); jednostruka odjava sa svih e-usluga na kojima je Korisnik prijavljen ako je ona podržana (*Single Sign-out*); zaštita kroz ugrađene mehanizme za otkrivanje uljeza; zaštita privatnosti osobnih podataka prilikom razmjene podataka te uvid u vrstu osobnih podataka koji će biti prosljeđeni e-usluzi te mogućnost obustave procesa razmjene podataka od strane Korisnika i njegovo odustajanje od pristupa e-usluzi putem NIAS-a. Stoga Svaki Korisnik prilikom prve autentikacije putem NIAS-a *on-line* prihvaća Uvjete korištenja NIAS-a, nakon čega mu se odmah generira korisnički račun u sustavu mojID koji se veže se uz OIB Korisnika fizičke osobe.

³⁵⁵ ePASS, <https://epass.gov.hr>



Kroz korisnički sustav e-usluge **mojID**³⁵⁶, Korisniku NIAS-a omogućava personalizaciju korisničkog računa u NIAS-u te profiliranje određenih aktivnosti vezanih uz rad NIAS-a, kao što je unos e-mail adresa za primanje obavijesti o aktualnim autentikacijama, uvid u povijest podataka razmijenjenih s e-uslugom prilikom svake prijave na e-uslugu i podatak o vrsti vjerodajnice na temelju koje je ta prijava napravljena te podešavanje postavki za davanje dozvole NIAS-u za automatsko prosljeđivanje osobnih podataka Pružatelju e-usluge.

9.1.4 Integracija usluge

Tehnička integracija e-usluge na uslužnom poslužitelju Pružatelja e-usluge s NIAS-om obavlja se na način da Pružatelj e-usluge pribavi odgovarajući **aplikacijski i poslužiteljski X509 certifikat** za e-uslugu te da implementira **protokol SAML** kojim se ostvaruje komunikacija između uslužnog poslužitelja i NIAS-a. Aplikacijski certifikat X509 osigurava sigurnu razmjenu podataka između NIAS-a i e-usluge, a javni ključ tog certifikata X509 mora biti dostupan NIAS-u. Analogno, javni ključ NIAS-ovog poslužiteljskog certifikata X509 mora biti dostupan e-usluzi. Razmjena certifikata obavlja se prije početka integracije e-usluge s NIAS-om.



Za implementaciju integracije Pružatelj e-usluge može se koristiti NIAS-ov integracijski modul koji sadrži programske biblioteke protokola SAML za različite razvojne platforme ili se može izraditi vlastite biblioteke koje podržavaju protokol SAML. Biblioteke za komunikaciju s NIAS-om sadrže niz funkcionalnosti pomoću kojih se šalje zahtjev za porukama SAML te se prima odgovor na taj isti zahtjev, kao i funkcionalnosti pomoću kojih se dohvaćaju korisne informacije iz odgovora na zahtjev za porukom SAML. Integracijski modul NIAS-a se isporučuje kao predložak koji onda Pružatelj e-usluge prilagođava specifičnostima svojeg sustava te ga nadalje samostalno održava.

Pružatelj e-usluge unutar svojeg sustava treba stvoriti web stranicu koja će mu služiti za slanje zahtjeva za porukom SAML te za zaprimanje odgovora na taj zahtjev. Prije postupka integracije, Pružatelj e-usluge dostavlja URL web stranice na kojoj je dostupna e-usluga koja se integrira i URL za odjavu, minimalnu razinu sigurnosti, podatke o aplikativnom certifikatu te o namjeni korištenja za građane koji pristupaju vjerodajnicom iz Republike Hrvatske i drugih zemalja.

Postupak integracije za potrebe testiranja se radi na način da se e-usluga prvo integrira s testnim sustavom NIAS, a kasnije se za produkcijsku integraciju e-usluga integrira s produkcijskim sustavom NIAS, pri čemu se s obje strane koriste produkcijski aplikacijski certifikati X509.

Ako, osim autentikacijskih podataka, e-usluga u okviru e-Poslovanja za potrebe pristupa Korisnika traži i autorizacijske podatke, provest će integraciju s podsustavom e-Ovlaštenja sukladno Protokolu rada za e-Ovlaštenja, Tehničkoj specifikaciji za dohvat autorizacijskih

³⁵⁶ Sustav mojID, <https://mojid.gov.hr/>

podataka te, opcionalno, Tehničkoj specifikaciji za integraciju registracijskog obrasca u ePoslovanje.

9.1.5 Integracija vjerodajnice

Za postupak integracije nove vjerodajnice u NIAS, Izdavalatelj vjerodajnice prvo sklapa trostrani sporazum s Finom i MURH-om, kako bi mogao integrirati svoj autentikacijski poslužitelj s NIAS-om. Tehnička integracija autentikacijskog poslužitelja Izdavalatelja vjerodajnice s NIAS-om obavlja se na način da Izdavalatelj vjerodajnice pribavi odgovarajući aplikacijski i poslužiteljski X509 certifikat za autentikacijski poslužitelj te da implementira protokol SAML kojim se ostvaruje komunikacija između autentikacijskog poslužitelja i NIAS-a.



Slično kao i za integraciju e-usluge, za integraciju vjerodajnice može se koristiti NIAS-ov integracijski modul koji sadrži programske biblioteke protokola SAML za različite razvojne platforme ili se može izraditi vlastite biblioteke koje podržavaju SAML standard. Biblioteke za komunikaciju s NIAS-om sadrže niz funkcionalnosti pomoću kojih se šalje zahtjev za porukama SAML te prima odgovor na taj isti zahtjev, kao i funkcionalnosti pomoću kojih se dohvaćaju korisne informacije iz odgovora na zahtjev za porukom SAML. Integracijski modul NIAS-a se isporučuje kao predložak koji onda Izdavalatelj vjerodajnice prilagođava specifičnostima svojeg sustava te ga nadalje samostalno održava.

Prije postupka integracije, Izdavalatelj vjerodajnice dostavlja URL web stranice na kojoj je dostupan autentikacijski poslužitelj Izdavalatelja vjerodajnice koji se integrira te podatke o aplikativnom certifikatu. Tehnička integracija poslužitelja e-usluge s NIAS-om obavlja se u nekoliko koraka na način da pružatelj e-usluge:

1. implementira e-uslugu na poslužitelj e-usluge u formi web aplikacije dostupne putem Interneta
2. pribavi aplikacijski certifikat X509 za e-uslugu kojim će štititi komunikaciju s NIAS-om
3. preuzme javni ključ NIAS-ovog aplikacijskog certifikata X509 kojim NIAS štiti komunikaciju
4. implementira protokol SAML kojim se ostvaruje komunikacija između e-usluge i NIAS-a (SAMLRequest) te NIAS-a i e-usluge (SAMLResponse) korištenjem aplikacijskog certifikata X509 i porukama prema specifikaciji
5. dostavi NIAS-u URL web stranice na kojoj e-usluga očekuje odgovor od NIAS-a, a koja implementira protokol SAML i omogućuje zaprimanje i obradu poruke SAMLResponse od NIAS-a
6. dostavi NIAS-u javni ključ pribavljenog aplikacijskog certifikata X509 za e-uslugu kojim će e-usluga štititi komunikaciju s NIAS-om.

Postupak integracije za potrebe testiranja se radi na način da Izdavalatelj vjerodajnice prvo integrira autentikacijski poslužitelj s testnim sustavom NIAS, a kasnije za produkcijsku integraciju integrira autentikacijski poslužitelj s produkcijskim sustavom NIAS, pri čemu se s obje strane koriste produkcijski aplikacijski certifikati X509. Nakon uspješne integracije, Korisnici vjerodajnice koja je integrirana u NIAS mogu koristiti novu vjerodajnicu posredstvom NIAS-a.

9.1.6 Komunikacija

Tijek komunikacije u procesu autentikacije Korisnika detaljnije je prikazan na slici Sekvencijalni dijagram rada NIAS-a u procesu autentikacije Korisnika, koja prikazuje tijek komunikacije između Korisnika, Pružatelja elektroničke usluge, NIAS-a i Izdavatelja vjerodajnice.

Proces komunikacije inicira Korisnik koji putem svojeg web preglednika traži pristup e-usluzi slanjem poruke protokolom HTTP (korak 1.). Kako bi e-usluga identificirala Korisnika koji traži pristup, preusmjerava ga na NIAS kako bi se autenticirao slanjem poruke SAMLRequest (korak 2.). NIAS Korisniku prikazuje popis osobnih podataka koje e-usluga traži u svrhu prijave te od Korisnika traži dozvolu za njihovo slanje. Ako Korisnik da dozvolu za slanje prikazanih osobnih podataka, NIAS ga upućuje na odabir vjerodajnica kojima je moguće izvršiti prijavu na odabranu e-uslugu (korak 3.). Korisnik odabire vjerodajnicu kojom se želi autenticirati (korak 4.), a NIAS ga preusmjerava Izdavatelju vjerodajnice porukom SAMLRequest (korak 5.). Ako Korisnik ne dozvoli NIAS-u prosljeđivanje prikazanih podataka, NIAS obustavlja pokrenuti proces.

Nadalje, Korisnik se na stranicama Izdavatelja vjerodajnice (korak 6.) autenticira odabranom vjerodajnicom (korak 7.). Izdavatelj vjerodajnice s autentikacijskog poslužitelja radi provjeru validnosti i valjanosti izdane vjerodajnice uspoređujući ih s onima koje drži pohranjene u svojoj bazi podataka. Nakon što autentikacijski poslužitelj izvrši provjeru podataka, obavještava NIAS o validnosti i valjanosti korisničkih podataka slanjem OIB-a Korisnika porukom SAMLResponse (korak 8.). Na kraju, NIAS provjerava korisnički račun i šalje e-usluzi odgovor porukom SAMLResponse (korak 9.), a e-usluga omogućuje pristup autenticiranom korisniku (korak 10.).



Slika 9.1 Sekvencijalni dijagram rada NIAS-a u procesu autentikacije Korisnika³⁵⁷

Izdavatelji poslovnih vjerodajnica uz OIB Korisnika obavezno šalju i druge podatke koji su vezani uz poslovni subjekt kao što su: OIB poslovnog subjekta (OIB pravne ili fizičke osobe koja djeluje kao poslovni subjekt), identifikator poslovnog subjekta kod izdavatelja vjerodajnice – PSID, te DN ukoliko se autentikacija obavlja certifikatom.



Za pravnu osobu PSID je MB pravne osobe, za obrt PSID je matični broj obrta (MBO), za obiteljska poljoprivredna gospodarstva PSID je matični identifikacijski broj poljoprivrednog gospodarstva (MIBPG), za slobodne djelatnosti PSID je matični broj s Državnog zavoda za statistiku (MB), a za sporedna zanimanja PSID je redni broj odobrenja koji dodjeljuje Gradski ured za gospodarstvo, rad i poduzetništvo.

³⁵⁷ Protokol rada NIAS-a za e-Građane i e-Poslovanje, verzija 1.6.2., 31.07.2019.

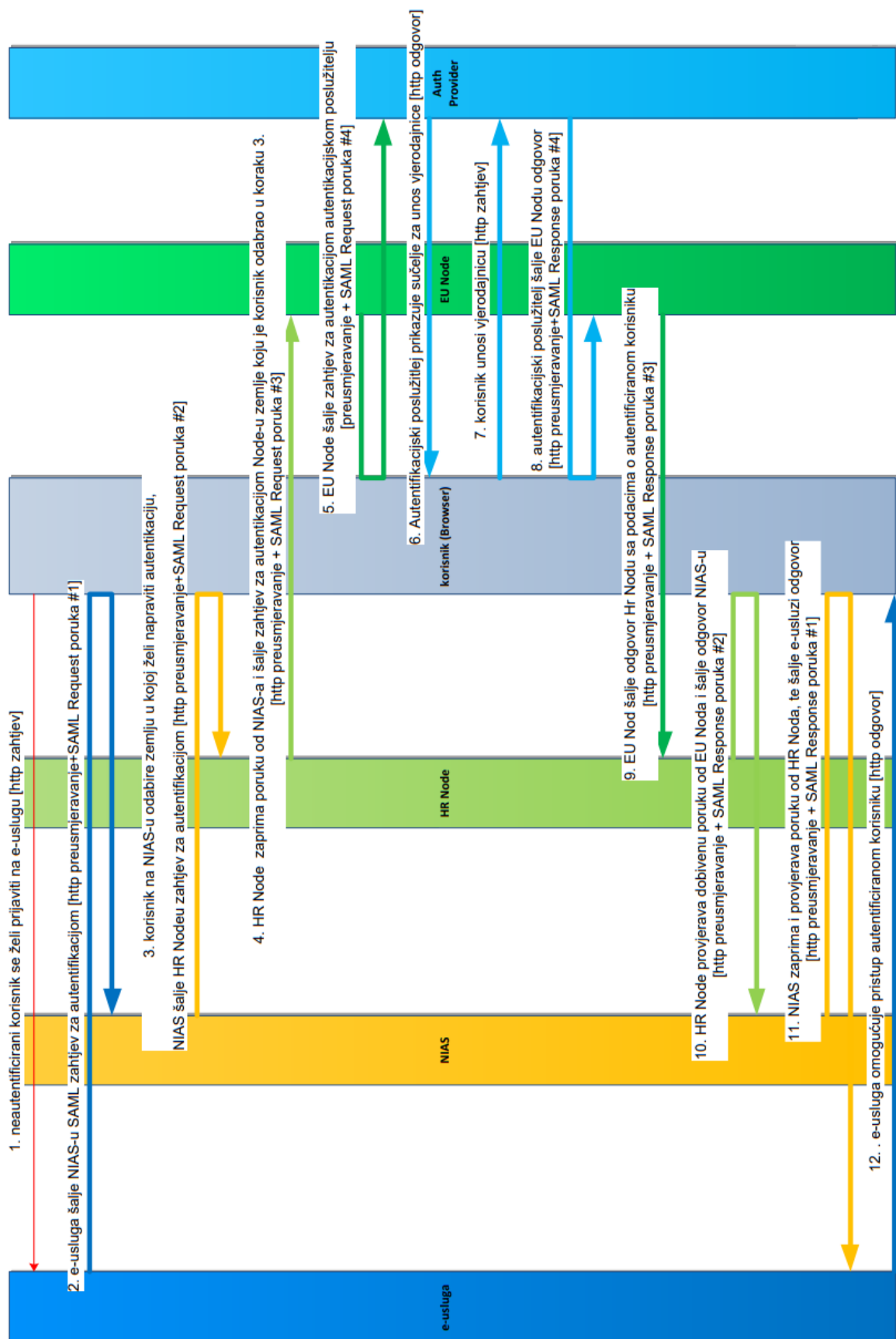
Temeljem podataka o OIB-u Korisnika, NIAS radi provjeru postojanja i aktivnosti Korisnika u OIB sustavu te dohvaća i ostale zatražene atribute. Ako provjera ne zadovolji, autentikacijski poslužitelj NIAS-u šalje poruku o grešci te NIAS obustavlja proces autentikacije. Kod prijave poslovnom vjerodajnicom, NIAS temeljem dostavljenih podataka od Izdavatelja vjerodajnice – OIB-a poslovnog subjekta s vjerodajnice i identifikatora poslovnog subjekta kod izdavatelja vjerodajnice (PSID), radi provjeru postojanja o poslovnom subjektu kod Pružatelja atributa, te dohvaća zatražene atribute poslovnog subjekta. Ako provjera zadovolji, NIAS će sukladno poslovnim pravilima kreirati Jedinostveni identifikator poslovnog subjekta (JIPS). Jedinostveni identifikator poslovnog subjekta kombinacija je dva podataka, identifikatora poslovnog subjekta - IPS (nakon provjere u povezanom registru) i izvora u registru (IZVOR_REG).



Identifikator poslovnog registra nakon provjere u povezanom registru (IPS) ne treba biti identičan Identifikatoru poslovnog subjekta koji je za njega dostavio Izdavatelj vjerodajnice (PSID) (npr. Izdavatelj vjerodajnica dostavlja MB pravne osobe, a Identifikator poslovnog registra koji je sastavni dio Jedinostvenog identifikatora poslovnog subjekta za pravnu osobu je OIB pravne osobe). Ako provjera ne zadovolji, autentikacijski poslužitelj NIAS-u šalje poruku o grešci te NIAS obustavlja proces autentikacije.

Pružatelju e-usluge NIAS dostavlja SAML odgovor s podacima o autentifikaciji. Ako su Pružatelju e-usluge dovoljni podaci o Korisniku koji se autentificirao posredstvom NIAS-a, Pružatelj e-usluge odobrava pristup Korisniku. Proces autentikacije se smatra završenim nakon što NIAS Pružatelju usluge isporuči SAML poruku koja je digitalno potpisana od strane NIAS-a. Ako Pružatelj poslovne e-usluge za nastavak rada zatraži autorizacijske podatke, temeljem dobivenih autentikacijskih podataka dodatno generira upit prema podsustavu e-Ovlaštenja sukladno Tehničkim specifikacijama koje se odnose na e-Ovlaštenja. Sustav e-Ovlaštenja poslovnoj e-usluzi dostavlja podatke o poslovnim subjektima u čije ime Korisnik ima pravo djelovati, sukladno upitu Pružatelja e-usluge. Jedinostveni identifikator poslovnog subjekta (JIPS) ključ je u identifikaciji poslovnih subjekata u projektu e-Poslovanje. Podatak o OIB-u poslovnog subjekta (koji nije sastavni dio JIPS-a) koristi se samo u dohvat autentikacijskih podataka, dok se ne koristi u dohvat autorizacijskih podataka. Zbog potreba prepoznavanja fizičkih osoba koje djeluju kao poslovni subjekt (npr. obrt i sl.), e-usluga može dodatno zatražiti podatke od e-Ovlaštenja o svim aktivnim vlasnicima tih poslovnih subjekata.

U slučaju Prekograničnog korisnika, tijek komunikacije je donekle izmijenjen odnosno proširen. U ovom slučaju od Prekograničnog korisnika se traži da definira zemlju, odnosno Prekogranični čvor s kojeg dolazi te se prikazuje popis osobnih podataka koje e-usluga traži u svrhu prijave. Prekogranični korisnik odabire zemlju izdavanja vjerodajnice kojom se predstavlja, te preko Čvora biva preusmjeren na odgovarajući Prekogranični čvor pošiljateljice podataka. NIAS od Prekograničnog korisnika ujedno traži i dozvolu za slanje zatraženih podataka e-usluzi u svrhu prijave. Ako Prekogranični korisnik ne dozvoli slanje tih podataka, NIAS obustavlja proces. Proces autentikacije Prekograničnog korisnika detaljnije je prikazan na slici Sekvencijalni dijagram rada NIAS-a u procesu autentikacije Prekograničnog korisnika.



Slika 9.2 Sekvencijalni dijagram rada NIAS-a u procesu autentikacije Prekograničnog korisnika

9.1.7 Protokoli komunikacije

Kao što smo već objasnili, komunikacija se temelji na razmjeni poruka između četiri različita entiteta: Korisnika, Pružatelja e-usluge, Izdavatelja vjerodajnice i NIAS-a. Uloga korisnika je zapravo pasivna te on služi samo kao transportni sloj u razmjeni poruka, ostali entiteti su aktivni (engl. *endpoints*), što znači da se, prilikom dobivanja zahtjeva, očekuje dati odgovor. Razmjena poruka odvija se temeljem izdvojenih protokola koje definira standard SAML 2.0, a za potrebe NIAS-a su odabrani protokoli HTTP-Redirect i HTTP-POST *redirect binding*.

Protokol **HTTP Redirect Binding** propisuje razmjenu poruka preko korisnika pomoću HTTP/HTTPS transportnog sloja. Poruke se razmjenjuju tako da se potpisuju na strani poslužitelja te se korisnik zajedno s porukom preusmjerava na poslužitelj kojemu je ta poruka namijenjena. Iako je korisnik nositelj poruke o svojoj autentifikaciji, na ovaj je način razmjena poruka u potpunosti osigurana te su poruke nepromjenjive. Protokol se dijeli na razmjenu poruka pomoću metoda HTTP Redirect i HTTP POST.

Metoda **HTTP Redirect** podrazumijeva slanje svih parametara poruke SAML putem URL-a. Poruka se može izravno predati na način da se korisniku na stranici generira poveznica s URL-om koja korisnika zajedno sa SAML porukom preusmjeruje na određeni poslužitelj, te mora sadržavati sljedeće parametre:

- Poruka **SAMLRequest** ili **SAMLResponse** – u ovisnosti o tome koja poruka se šalje
- **RelayState** – parametar koji sadrži specifični identifikator pošiljatelja, a prilikom odgovora SAML ovo polje mora biti jednako vrijednosti koje je poslano zahtjevom SAML
- **SigAlg** – URI koji je definiran standardom XML-Sig, a opisuje algoritam potpisivanja koji je korišten prilikom potpisa XML poruke SAML (DSAwithSHA1 ili RSAwithSHA1)
- **Signature** – vrijednost potpisa.

Formiranje URL-a HTTP GET izvodi se tako da se iz poruke SAML odstranjuje element `<ds:Signature>` koji tu poruku potpisuje kako bi poruka bila manja. Zatim se poruka SAML sažima algoritmom DEFLATE³⁵⁸, a sažeta poruka se kodira algoritmom Base64³⁵⁹. Poruka u obliku Base64 se URL kodira te se sprema u parametar GET poruke SAMLRequest ili SAMLResponse, u ovisnosti o tipu poruke. Dodaje se parametar RelayState te se kodira URL.

U slučaju da je SAML poruka bila potpisana, potrebno je dodati i potpis parametara GET. Potpis parametara GET se vrši slaganjem prethodno definiranih parametara kako slijedi:

```
SAMLRequest=value&RelayState=value&SigAlg=value
```

Dobiveni znakovni niz potpisuje se pomoću algoritma definiranog poljem SigAlg, a potom se kodira po Base64, prilagođuje formatu URL te na kraju dodaje metodi GET kao parametar Signature. Konačni URL mora izgledati ovako:

```
https://nias.eid.com.hr/authenticate.aspx?SAMLRequest=value&RelayState=value&SigAlg=value&Signature=value
```

³⁵⁸ RFC 1951, DEFLATE Compressed Data Format Specification version 1.3, <https://tools.ietf.org/html/rfc1951>

³⁵⁹ RFC 2045, Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies - Base64 Content-Transfer-Encoding, <https://tools.ietf.org/html/rfc2045#page-24>

Drugi oblik protokola HTTP-Redirect Binding koji koristi metodu POST internetskog preglednika za slanje podataka je **HTTP-POST**, a u tom slučaju je potrebno korisniku stvoriti stranicu HTML s obrascem koji će inicirati slanje podataka na određeni poslužitelj. Obrazac treba sadržavati parametre SAMLRequest / SAMLResponse, u ovisnosti o poruci koja se šalje, i RelayState, odnosno elemente HTML input tog oblika.

Stvaranje obrasca HTTP POST izvodi se kako slijedi:

- parametar obrasca `method` je potrebno je postaviti na vrijednost `POST`
- parametar obrasca `action` je potrebno postaviti na određenu adresu
- potrebno je dodati skriveni element `<input name="SAMLRequest">` ili `<input name="SAMLResponse">` koji sadrži poruku SAMLRequest / SAMLResponse kodiranu po Base64
- potrebno je dodati skriveni `<input name="RelayState">` koji sadrži vrijednost poslanu porukom SAMLRequest ili proizvoljnu vrijednosti generiranu od strane koja šalje poruku SAMLRequest.



Obrazac može sadržavati kod JavaScript koji radi automatsko slanje podataka (engl. *submit*), ali i onda mora postojati gumb kojim korisnik može sam pokrenuti slanje podataka s obrasca u slučaju da je JavaScript blokiran u internetskom pregledniku.

9.1.8 Oblik poruke

Komunikacija između pojedinih entiteta odvija se putem poruka, a u sklopu NIAS-a integrirano je 6 parova poruka, zahtjeva i odgovora između NIAS-a i poslužitelja e-usluge za potrebe jedinstvene prijave korisnika na e-uslugu (engl. *Single Sign-On*), poruke između NIAS-a i autentikacijskog poslužitelja za potrebe autentikacije korisnika te poruke između NIAS-a i poslužitelja e-usluge za potrebe jedinstvene odjave korisnika (engl. *Single Sign-Out*).

Poruka **AuthnRequest** za zahtjev za autentikacijom korisnika od drugog poslužitelja je dio standarda SAML te propisuje uvjete i načine autentikacije koje autentikacijski poslužitelj mora napraviti kako bi uspješno autentificirao korisnika za pojedinu uslugu. NIAS koristi izdvojeni set standarda SAML za poruku AuthnRequest, kao u sljedećem primjeru:

```
<?xml version="1.0" encoding="utf-8"?>
<AuthnRequest xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" ID="c831b14f-85d3-4858-
b1b0-2e7297e5177b" Version="2.0" IssueInstant="0001-01-01T00:00:00"
Destination="https://localhost/eid/authenticate.aspx"
ProtocolBinding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST"
AssertionConsumerServiceURL="https://localhost/myid/default.aspx"
xmlns="urn:oasis:names:tc:SAML:2.0:protocol">
  <Issuer Format="urn:oasis:names:tc:SAML:1.1:nameid-format:X509SubjectName"
xmlns="urn:oasis:names:tc:SAML:2.0:assertion">CN=mojID, OU=FINA 00332852,
OU=Poslovni, OU=DEMO, O=FINA, C=HR</Issuer>
  <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
    <SignedInfo>
      <CanonicalizationMethod Algorithm="http://www.w3.org/TR/2001/REC-xm1-
c14n20010315" />
```

```

    <SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1" />
    <Reference URI="#c831b14f-85d3-4858-b1b0-2e7297e5177b">
      <Transforms>
        <Transform Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature"
/>
        <Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
      </Transforms>
      <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
      <DigestValue>4vX2Uy0qJdvYEuS+4qBaVoP8YVQ=</DigestValue>
    </Reference>
  </SignedInfo>
  <SignatureValue>...</SignatureValue>
  <KeyInfo>
    <X509Data>
      <X509Certificate>...</X509Certificate>
    </X509Data>
  </KeyInfo>
</Signature>
<NameIDPolicy Format="urn:oasis:names:tc:SAML:2.0:nameid-format:persistent" />
<Conditions NotBefore="2012-08-20T12:48:00.0771924Z" NotOnOrAfter="2012-08-
20T13:14:00.0771924Z" xmlns="urn:oasis:names:tc:SAML:2.0:assertion">
  <OneTimeUse />
  <Condition xmlns:q1="http://nias.eid.com.hr/2012/07/saml20Extension"
xsi:type="q1:NiasConditionType" MinAuthenticationSecurityLevel="1" />
</Conditions>
</AuthnRequest>

```

Elementi poruke SAMLRequest:

- Element **AuthnRequest** je osnovni (*root*) element
 - atribut **ID** – GUID, jedinstveni identifikator poruke
 - atribut **Version** – oznaka verzije SAML standarda koji se koristi odnosno 2.0
 - atribut **IssueInstant** – trenutak izdavanja SAML poruke izražen vremenskom oznakom UTC
 - atribut **Destination** – odredišna adresa prema kojoj se SAML poruka šalje
 - atribut **ProtocolBinding** – protokol kojim se poruka šalje, a NIAS dopušta samo protokole `urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST` i `urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Redirect`
 - atribut **AssertionConsumerServiceURL** – adresa poslužitelja koji prihvaća poruku SAMLResponse
- Element **Issuer**
 - atribut **Format** – format zapisa o izdavatelju `urn:oasis:names:tc:SAML:1.1:nameid-format:X509SubjectName`
 - vrijednost elementa – SubjectName aplikacijskog certifikata kojim se usluga predstavlja
- Element Signature:
 - vrijednost elementa – definiran je standardom XML-Sig, a NIAS zahtijeva da SAML poruka bude potpisana aplikacijskim certifikatom namijenjenim za komunikaciju s NIAS-om
- Element **NameIDPolicy**
 - atribut **Format** – identifikator kojim će korisnik biti predstavljen usluzi, a NIAS podržava `urn:oasis:names:tc:SAML:2.0:nameid-format:persistent` (korisnik je predstavljen jedinstvenim identifikatorom koji je nepromjenjiv, dvije različite usluge će imati dva različita identifikatora iste osobe), `urn:oasis:names:tc:SAML:2.0:nameid-format:entity` (korisnik je predstavljen jedinstvenim identifikatorom koji nepromjenjiv, dvije različite usluge će imati isti identifikator za istu osobu) i `urn:oasis:names:tc:SAML:2.0:nameid-format:transient` (korisnik je predstavljen identifikatorom koji je promjenjiv)
- Element **Conditions**
 - atribut **NotBefore** – vrijeme prije kojega poruka ne vrijedi
 - atribut **NotOnOrAfter** – vrijeme nakon kojega poruka ne vrijedi
 - element **OneTimeUse** – poruka se smije samo jednom iskoristiti, a usluga koja čita poruku u tom slučaju mora sama voditi popis svih iskorištenih ID-ova te ne smije dopustiti ponavljanje iste poruke

- element **Condition** – NIAS-ova ekstenzija standarda SAML koja definira atribut **MinAuthenticationSecurityLevel** pomoću kojeg se može definirati razina sigurnosti koju usluga zahtijeva

Poruka **SAMLResponse** (element XML Response) je odgovor na određeni zahtjev za autentikaciju, a poveznica sa zahtjevom na koji je odgovor vezan se nalazi u polju **InResponseTo**. Kada entitet primi autentikacijski odgovor, na njemu je potrebno provjeriti sigurnosne elemente te je li status poruke jednak **Success**. Tek nakon tih radnji korisnik se smatra uspješno autenticiranim te se može prijeći na čitanje atributa o korisniku odnosno identifikaciju korisnika u vlastitom sustavu.