UDC: 004.42.045:004.738.5 INFO M: str. 29-37

KOMPARATIVNA ANALIZA JAVA I .NET WEB SERVISA
COMPARATIVE ANALYSIS OF JAVA AND .NET WEB SERVICES

Nebojša Ristin, Siniša Vlajić, Ilija Antović, Miloš Milić, Vojislav Stanojević

REZIME: Ovaj rad predstavlja rezultat istraživanja sprovedenog sa ciljem da se postavi formalni model arhitekture web servisa i izvrši komparativna analiza performansi web servisa realizovanih korišćenjem dve vodeće platforme za razvoj softvera – Java i .NET. Sa ciljem davanja pune slike o standardnoj arhitekturi web servisa, njihove teorijske i tehnološke osnove su opisane preko formalnog modela, predstavljenog UML dijagramom. Nakon toga dat je opis uočenih elemenata i veza u formalnom modelu, sa posebnim fokusom na ulogu elemenata i formulaciju njihovih međusobnih zavisnosti. Posmatrane platforme za razvoj softvera prikazane su kroz tehnologije za implementaciju Web servisa, i prezentovani su rezultati poređenja različitih tehnologija unutar svake od platformi. Korišćenjem metoda dinamičke analize i softverskih profajlera, vršen je monitoring aplikacija i analiza njihovih performansi. Posmatrane platforme su upoređene sa aspekta izvršavanja web servisa u JAX-WS i WCF implementacijama, i predstavljeni su dobijeni rezultati. Na kraju je prikazana diskusija u kojoj su objašnjene uočene prednosti i mane Java i .NET realizacija Web servisa.

KLJUČNE REČI: Web servisi, arhitektura Web servisa, model Web servisa, Java, .NET, JAX-WS, WCF, komparativna analiza, softverski profajler

ABSTRACT: This paper is the result of research carried out in order to define a formal model of Web services architecture and perform a comparative analysis of performance of the web services implemented using two leading software development platforms - Java and. NET framework. With the goal of providing a full picture of the standard Web services architecture, their theoretical and technological basis are described by formal model, presented by UML diagram. Furthermore, the observed elements and relationships from the formal model are elaborated, with focus on roles of elements and formulation of their interdependence. The considered software development platforms are presented through Web services development technologies available in each of them, and the results of comparison of different technologies inside each platform. By using dynamic analysis methods and software profilers, application monitoring and performance analysis were executed. The considered platforms are compared in terms of execution of web services in JAX-WS and WCF implementations, and the results are presented. At the end, there is the discussion which explains the perceived advantages and disadvantages of Java and. NET Web services implementation.

KEY WORDS: Web service, Web services architecture, Web services model, Java, .NET, JAX-WS, WCF, comparative analysis, software profiler

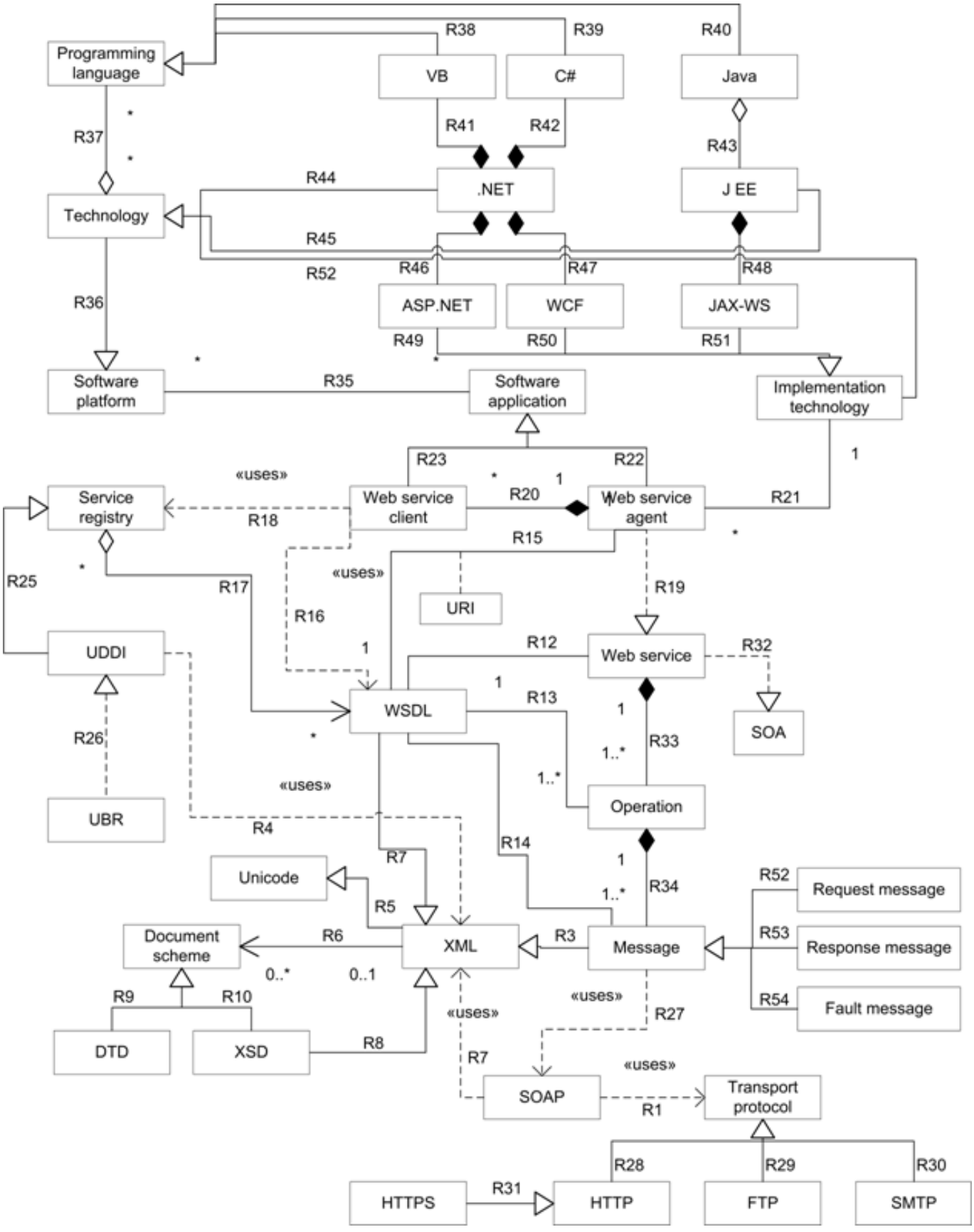
1. UVOD

Oblast informacionih tehnologija i razvoj softvera kao njen deo karakteriše upotreba različitih i međusobno nekompatibilnih tehnologija. Web servisi su se nametnuli kao rešenje jednog od ključnih problema – kako međusobno različite sisteme povezati u funkcionalnu celinu. Apstrahujući softversku platformu na kojoj se sistem izvršava i tehnologije korišćene za njegov razvoj, oni omogućavaju da sistemi razmenjuju informacije na standardizovan način.

Web servisi predstavljaju korak napred u evoluciji distribuiranih sistema, kao i u evoluciji XML aplikacija – inicijalna uloga XML tehnologije u struktuiranju podataka proširena je na struktuiranje poruka, odnosno od statičke uloge postala je procesna, dinamička. Zbog toga bi se web servisi mogli posmatrati kao spoj distribuiranih softverskih arhitektura i XML tehnologije, i upravo navedeni spoj donosi ono što je u web servisima revolucionarno – puna platformska nezavisnost i interoperabilnost distribuiranih aplikacija. Razlog neuspeha drugih distribuiranih tehnologija bio je upravo u zahtevima za zajedničkom platformom za izvršavanje aplikacija, što je potpuno onemogućavalo integraciju raznorodnih sistema. Web servisi međusobno razmenjuju XML poruke, pa je XML jedina platforma od koje zavise – razvojne tehnologije, operativni sistemi ili arhitektura računarskog hardvera ne predstavljaju prepreku za realizaciju ovakvih distribuiranih aplikacija. Upotreba web servisa upravo iz ovog razloga obuhvata širok spektar aplikacija, od specifičnih poslovnih rešenja, do pretraživača i digitalnih biblioteka [Santos08].

Predmet istraživanja ovog rada je komparativna analiza web servisa realizovanih korišćenjem *Java* i *.NET* tehnologija. Postavljeni ciljevi su: 1) kroz formalni model predstaviti međuzavisnosti između tehnologija i koncepata koji se vezuju za web servise i 2) Izvršiti komparativnu analizu web servisa realizovanih korišćenjem *Java* i *.NET* tehnologija korišćenjem softverskih profajlera, i na osnovu rezulata analize uporediti i oceniti posmatrane tehnologije sa aspekta brzine izvršavanja web servisa.

S obzirom na rivalitet i borbu za tržišni udeo koja se vodi između Java i .NET tehnologija, njihove implementacije web servisa su konstantno razvijane i unapređivane. Upoređivanje .NET i Java tehnologija sa aspekta web servisa može dati odgovor koja tehnologija je bolje prilagođena razvoju web servisa, kao i koja pruža bolje performanse razvijenih aplikacija. Da bi se razrešila dilema o performansama web servisa



Slika 1. – *Model web servisa*

realizovanih u ovim tehnologijama, biće razvijena po jedna demonstraciona aplikacija u svakoj od tehnologija, koje će zatim biti upoređene po nekoliko kriterijuma.

2. WEB SERVISI - ARHITEKTURA I KONCEPTI

Za temeljno proučavanje web servisa potrebno ih je posmatrati iz različitih perspektiva – od najvišeg nivoa apstrakcije, kao interfejsa prema nekom servisu van specifičnog softverskog sistema, do posmatranja konkretnih procesa i tehnologija. Postoji niz međusobno povezanih elemenata koji se mogu posmatrati kao gradivna materija web servisa i njihovih funkcionalnosti. Upravo ta detaljnost i složenost usmerava na proučavanje web servisa kao celine sastavljene iz funkcionalnih delova, i opisivanje međuzavisnosti između tih delova kroz formalni model. Za elemente modela biće definisane veze sa drugim elementima, na osnovu odgovora na četiri pitanja: *Definicija - Šta je? Uloga - Za šta se koristi? Preduslovi - Šta koristi, od čega zavisi? Zavisni elementi - Šta zavisi od njega?* Tako određene međuzavisnosti biće prikazane u jedinstvenom *UML* dijagramu klasa, uz konceptualni i tehnološki opis značajnih elemenata modela u svrhu razjašnjenja njihove uloge u arhitekturi web servis aplikacija.

Model web servisa prikazan je dijagramom na Slici 1.

2.1 *Elementi modela web servisa*

U nastavku su prezentovana detaljna objašnjenja značajnih elemenata modela. Uloge onih elemenata koji nisu detaljnije razrađeni su jasne i bez daljeg objašnjenja.

Element: **Web servis**.

Definicija:. Web servis je softverski sistem projektovan da omogući interakciju između više računara preko mreže. Način pristupa i korišćenja web servisa se opisuje u formatu koji se može tumačiti od strane računara (konkretno *WSDL*). Drugi sistemi dolaze u interakciju sa web servisom na način određen njegovim opisom, i to preko *SOAP* poruka, obično razmenjivanim preko *HTTP* protokola sa *XML* serijalizacijom, a uz korišćenje i drugih standarda povezanih sa web-om.

Preduslovi: **Servisno orijentisana arhitektura -** web servisi su implementacija SOA (*Veza R32*: Realizacija – web servisi *implementiraju* SOA)

Zavisnosti: **Agent web servisa** (*Veza R19*: Realizacija - Agent web servisa *realizuje* web servis). **Operacija** (*Veza V33*: Agregacija – web servis *pruža* jednu ili više operacija).

Element: **Servisno orijentisana arhitektura**.

Definicija: Arhitekturalni stil za razvoj softverskih aplikacija koje koriste servise dostupne na mreži poput web-a. [Erl21].

Zavisnosti: **Web servis –** web servisi su najčešća implementacija SOA (*Veza R32*: Realizacija – web servisi *implementiraju* SOA).

Element: **Agent web servisa**.

Definicija: Konkretan hardversko-softverski sistem koji implementira web servis. [W3C2] *(Veza R22*: nasleđivanje - agent web servisa *je* softverska aplikacija).

Uloga: **Realizacija web servisa –** Agent web servisa je softverska aplikacija koja realizuje web servis. Agent prima poruku u XML formatu, tumači je, izvršava odgovarajuću operaciju i vraća rezultat izvršavanja u vidu XML poruke. (*Veza R19*: Realizacija - Agent web servisa *realizuje* web servis).

Preduslovi: **Implementaciona tehnologija –** agent web servisa, kao softverska aplikacija, se implementira u odabranoj konkretnoj tehnologiji (*Veza R21*: Asocijacija – Agent web servisa se implementira u konkretnoj tehnologiji). **WSDL dokument -** WSDL opis servisa sadrži lokaciju agenta web servisa na osnovu koje klijent zna na kojoj lokaciji može da pristupi agentu (*Veza R15*: Asocijacija klase - WSDL locira agenta web servisa *preko* URI adrese).

Zavisnosti: **Klijent web servisa -** Klijent web servisa šalje XML poruke agentu servisa i od njega prima odgovor sa rezultatom poziva operacije (*Veza R20*: Agregacija – Klijent pristupa agentu web servisa; agentu može pristupiti više klijenata).

Element: **Klijent web servisa**.

Definicija: Softverska aplikacija koja preko SOAP protokola razmenjuje XML poruke sa agentom web servisa. [W3C2] (*Veza R23*: Nasleđivanje (Klijent web servisa *je* softverska aplikacija).

Uloga: **Izvršenje operacija servisa –** Klijent web servisa šalje XML poruke agentu servisa i od njega prima odgovor sa rezultatom poziva operacije (*Veza R20*: Agregacija - Klijent pristupa agentu web servisa; agentu web servisa može pristupati više klijenata).

Preduslovi: **Registar servisa -** Klijent koristi registar servisa da pronađe WSDL opise web servisa koji pružaju uslugu koja je njemu potrebna (*Veza R18*: Korišćenje - klijent *koristi* registar servisa da pronađe opise servisa). **WSDL dokument -** Klijent koristi WSDL opis servisa za lociranje web servisa i pozivanje njegovih operacija. Na osnovu specifikacije operacija i poruka date u WSDL dokumentu, klijent zna na koji način da formira XML poruke zahteva i kakve poruke da očekuje kao odgovor (*Veza R16*: Korišćenje - klijent *koristi* WSDL dokument za povezivanje sa servisom).

Zavisnosti: Nema.

Element: **Transportni protokol**.

Definicija: Protokol za razmenu poruka između aplikacijskih procesa koji se izvršavaju na različitim računarima.

Uloga: **Transport poruka od jedne mrežne lokacije na drugu -** SOAP poruke koje se razmenjuju između web servisa i klijenta transportuju se od jedne krajnje tačke na drugu preko transportnog protokola.

Preduslovi: None.

Zavisnosti: **SOAP -** SOAP protokol koristi transportne protokole za transport SOAP poruka. Za ostvarivanje funkcija

koje SOAP protokol ne pruža (pouzdanosti transporta, zaštite, rutiranja) koriste se postojeće funkcionalnosti transportnih protokola (*Veza R1:* Korišćenje - SOAP protokol *koristi* transportni protokol).

Tipovi: **HTTP**, **HTTPS**, **FTP**, **SMTP** (*Veze R28, R29, R30*: Nasleđivanje - *HTTP*, *FTP*, *SMTP su* transportni protokoli; *Veza R31*: Nasleđivanje - *HTTPS je* sigurni *HTTP* protokol).

Element: **XML - Extensible Markup Language**.

Definicija*:* Standardizovan proširivi jezik za označavanje tekstualnih dokumenata. [Erl21]**.**

Ulogas*:* **Struktuiranje poruka** – Poruke koje se razmenjuju u komunikaciji između web servisa i klijenta su u XML formatu. Preko XML sintakse je u poruci poziva servisa specifikovana operacija koja se poziva i vrednosti parametara, a u povratnoj poruci rezultat izvršavanja operacije. XML sintaksa koristi se za predstavljanje složenih tipova podataka u porukama (*Veza R3*: Nasleđivanje - Poruka *je* XML dokument). **Čuvanje podataka u UDDI registrima** – UDDI registri web servisa skladište podatke struktuirane u XML dokumentima (*Veza R4*: Korišćenje - UDDI registar *koristi* XML za struktuiranje skladištenih podataka).

Preduslovi*:* **Unicode standard –** XML dokumenti su tekstualne datoteke sa Unicode predstavljanjem karaktera. Platforma na kojoj se vrši kreiranje, obrada i čitanje XML dokumenata mora podržavati rad sa Unicode tekstualnim datotekama (*Veza R5*: Nasleđivanje - XML datoteka *je* Unicode datoteka). **XML šeme** – XML šeme definišu pravila strukture i sadržaja koja se primenjuju na XML dokumente koji ih referenciraju. XML dokument se smatra validnim ako poštuje sva pravila navedena u šemi koju referencira (*Veza R6*: Usmerena asocijacija - XML dokument *prati* šemu; šema ne zna za XML dokument)

Zavisnosti*:* **SOAP -** SOAP je protokol za razmenu informacija zasnovan na XML-u. SOAP poruke koje se razmenjuju između web servisa i klijenta su XML dokumenti sa strukturom koja prati specifikaciju SOAP protokola (*Veza R2*: Korišćenje - SOAP *koristi* XML za struktuiranje poruka). **WSDL -** WSDL dokumenti koji se koriste za opisivanje web servisa su XML dokumenti koji prate specifikaciju WSDL jezika (*Veza R7*: Nasleđivanje - WSDL dokument *je* XML dokument). **UDDI -** UDDI specifikacija definiše da se informacije u UDDI registrima web servisa čuvaju u XML formatu (*Veza R4*: Korišćenje - UDDI registar *koristi* XML za struktuiranje skladištenih podataka). **XML Schema -** XML Schema dokumenti su XML dokumenti koji referenciraju prostor imena koji određuje specifikacija XML Schema-e (*Veza R8*: Nasleđivanje - XML Schema dokument *je* XML dokument).

Element: **Šema XML dokumenata**.

Definicija*:* Opis klase XML dokumenata koji definiše pravila strukture i sadržaja.

Uloga*:* **Validacija poruka -** Za XML poruke se može definisati šema na osnovu koje se sadržaj poruke validira po pravilima strukture i sadržaja. SOAP poruke i WSDL dokumenti, kao XML datoteke, validiraju se po odgovarajućoj šemi.

Preduslovi*:* None.

Zavisnosti*:* **XML -** XML šeme definišu pravila strukture i sadržaja koja se primenjuju na XML dokumente koji ih referenciraju. XML dokument se smatra validnim ako poštuje sva pravila navedena u šemi koju referencira (*Veza R6*: Usmerena asocijacija - XML dokument prati šemu; šema ne zna za XML dokument).

Vrste: **DTD -** DTD je vrsta XML šeme koja može da definiše jednostavna pravila strukture i sadržaja za validaciju XML dokumenata (*Veza R9*: Nasleđivanje - DTD *je* vrsta XML šeme); **XML Schema -** XML Schema je vrsta XML šeme koja može da definiše složena pravila strukture i sadržaja za validaciju XML dokumenata (*Veza R10*: Nasleđivanje - XML Schema *je* vrsta XML šeme).

***Element*: XML Schema**.

Definicija*:* Šema XML dokumenata.

Uloga*:* **Formiranje poruka -** Poruke koje se razmenjuju između web servisa i klijenta mogu da uključuju podatke složenih tipova, koji izlaze iz okvira standardnih XML tipova podataka. Mapiranjem klasa iz objektno-orijentisanih programskih jezika u XML Schema datoteke omogućava se transformacija objekata u XML datoteke i obrnuto.

Preduslovi*:* **XML -** XML Schema dokumenti su XML dokumenti koji referenciraju prostor imena koji određuje specifikacija XML Schema-e (*Veza R8*: Nasleđivanje - XML Schema dokument *je* XML dokument).

Zavisnosti*:* None.

***Element*: SOAP**.

Definicija*:* Protokol za razmenu informacija u distribuiranim softverskim sistemima. [W3C1]

Uloga*:* **Razmena poruka između web servisa i klijenta -** Prilikom poziva operacije web servisa, klijent šalje SOAP poruku sa sadržajem koji određuje koju operaciju servisa i sa kojim parametrima treba pozvati. Rezultat izvršanja operacije šalje se u povratnoj SOAP poruci od web servisa do klijenta (*Veza R27*: Korišćenje - SOAP *se koristi* za razmenu poruka).

Preduslovi*:* **XML -** Svaka SOAP poruka je XML dokument sa strukturom koja prati specifikaciju SOAP protokola. U okviru specifikacijom definisanog XML elementa za sadržaj poruke nalaze se podaci koji se razmenjuju između učesnika u komunikaciji (*Veza R2*: Korišćenje - SOAP *koristi* XML za struktuiranje poruka); **Transportni protokoli -** Transport SOAP poruka se vrši korišćenjem nekog od transportnih protokola (kao što su HTTP, FTP, SMTP ili drugi). Za ostvarivanje funkcija koje SOAP protokol ne pruža (pouzdanosti transporta, zaštite, rutiranja) koriste se postojeće funkcionalnosti transportnih protokola (*Veza R1*: Korišćenje - SOAP protokol *koristi* transportni protokol).

Zavisnosti*:* None.

***Element*: WSDL - Jezik za opisivanje web servisa**.

***Definicija**: WSDL* pruža model i XML format za opisivanje web servisa. [W3C3]

***Uloga**:* **Interfejs web servisa -** WSDL dokument sadrži opis sistema koji predstavlja interfejs web servisa prema klijentima. Na osnovu specifikovanog WSDL opisa klijenti obavljaju komunikaciju sa servisom (*Veza R12*: Asocijacija - *WSDL* dokument opisuje web servis). **Specifikacija operacija web servisa -** WSDL dokument sadrži formalnu definiciju operacija koje web servis može da obavlja, sa listom parametara koje primaju i vrednosti koje vraćaju (*Veza R13*: Agregacija - WSDL dokument *sadrži* definiciju operacija). **Specifikacija strukture poruka -** WSDL dokument daje specifikaciju strukture poruka koje se razmenjuju između klijenta i servisa u toku obavljanja operacije servisa. Opisane su poruke poziva operacije, poruke vraćanja vrednosti i poruke greške (*Veza R14*: Agregacija - WSDL dokument *sadrži* specifikaciju poruka). **Lociranje agenta web servisa -** WSDL dokument daje informaciju o lokaciji na kojoj se može pristupiti web servisu. Lokacija je data u vidu URI adrese agenta web servisa (*Veza R15*: Asocijacija klase - WSDL locira agenta web servisa *preko* URI adrese).

***Preduslovi**:* **XML -** WSDL dokumenti su XML dokumenti koji prate specifikaciju WSDL jezika (*Veza R7*: Nasleđivanje - WSDL dokument *je* XML dokument).

***Zavisnosti**:* **Klijent web servisa -** Klijentska aplikacija koja koristi usluge web servisa na osnovu WSDL dokumenta dobija informacije o servisu. Podaci o dostupnim operacijama, tipovima podataka, formatima poruka i lokaciji, koji su klijentu potrebni za povezivanje sa web servisom, nalaze se u WSDL dokumentu (*Veza R16*: Korišćenje - klijent *koristi* WSDL dokument za povezivanje sa servisom). **Registar servisa -** Registri servisa sadrže opise web servisa u WSDL formatu. Pretragom registra klijenti mogu pronaći WSDL opise servisa za koje su zainteresovani (*Veza R17*: Usmerena agregacija - registar servisa *sadrži* WSDL opise servisa; WSDL dokument ne zna za registar servisa).

***Element*: Registar servisa**.

***Definicija**:* Centralizovani registar web servisa, u kom se u propisano struktuiranoj formi nalaze informacije o web servisima.

***Uloga**:* **Objavljivanje servisa -** Obavljivanje servisa je proces u kom pružalac usluge čini servis dostupnim potencijalnim klijentima [Graham5]. U registar servisa se postavljaju WSDL opisi web servisa da bi se učinili dostupnim potencijalnim klijentima. (*Veza R17*: Usmerena agregacija - registar servisa *sadrži* WSDL opise servisa; WSDL dokument ne zna za registar servisa). **Otkrivanje servisa –** proces pronalaženja pružaoca usluge i pribavljanja dokumenata koji opisuju serviris [Erl21]. Pretragom registra servisa klijenti mogu pronaći WSDL opise servisa za čije su usluge zainteresovani (*Veza R18*: Korišćenje - klijent *koristi* registar servisa da pronađe opise servisa).

***Preduslovi**:* None

***Zavisnosti**:* UDDI (*Veza R25*: Nasleđivanje – UDDI *je* servisni registar).

***Element*: Universal Description Discovery and Integration**.

***Definicija**:* Platformski nezavistan poslovni registar web servisa zasnovan na XML jeziku. [Bean10] (*Veza R24*: Nasleđivanje - UDDI *je* servisni registar)

***Preduslovi**:* **XML**

UDDI stores information structured in XML files. (*Veza R4*: Usage - UDDI registry *uses* XML for storing structured data)

***Zavisnosti**:* None

Realizations: UBR (*Veza R26*: Realization - UBR *je realizacija* UDDI registra)

3. IMPLEMENTACIJE WEB SERVISA

Sun Microsystems Java i *Microsoft .NET Framework* su dve vodeće tehnologiji za razvoj složenih poslovnih aplikacija. Podrška za razvoj web servisa je uključena u ove platforme preko biblioteka koje omogućavaju razvoj web servis aplikacija bez poznavanja tehnologija kao što su XML, SOAP, WSDL i druge.

Imajući u vidu realizaciju web servis aplikacije, mogu se postaviti sledeće veze između elemenata modela web servisa koje se tiču njihove implementacije:

Veza 35: Asocijacija (softverska aplikacija može da se izvršava na jednoj ili više softverskih platformi; na svakoj platformi izvršava se više aplikacija).

Veza 36: Nasleđivanje (tehnologija *je* softverska platforma).

Veza 37: Agregacija (tehnologija *uključuje* programske jezike).

Veza 38: Nasleđivanje (Visual Basic *je* programski jezik).

Veza 39: Nasleđivanje (C# *je* programski jezik).

Veza 40: Nasleđivanje (*Java je* programski jezik).

Veza 41: Kompozicija (Visual Basic *je deo* .NET tehnologije).

Veza 42: Kompozicija (C# *je deo* .NET tehnologije).

Veza 43: Kompozicija (Java EE *je jedna od* Java platformi).

Veza 44: Nasleđivanje (.NET *je* tehnologija).

Veza 45: Nasleđivanje (Java EE *je* tehnologija).

Veza 46: Kompozicija (ASP.NET *je deo* .NET tehnologije).

Veza 47: Kompozicija (WCF *je deo* .NET tehnologije).

Veza 48: Kompozicija (JAX-WS *je deo* Java EE).

Veza 49: Nasleđivanje (ASP.NET je tehnologija za implementaciju web servisa).

Veza 50: Nasleđivanje (WCF je tehnologija za implementaciju web servisa).

Veza 51: Nasleđivanje (JAX-WS je tehnologija za implementaciju web servisa).

Veza 52: Nasleđivanje (Tehnologija za implementaciju servisa *je* tehnologija).

3.1. JAVA WEB SERVISI

Podrška za web servise uključena je u Java Enterprise Edition, koji pruža osnovu za razvoj poslovnih aplikacija. Globalni model *Java EE* aplikacija, koji podrazumeva trosloj-

nu arhitekturu (sloj klijenta, servera i podataka) uključuje web servise, pri čemu agent web servisa predstavlja serverski sloj, a klijent web servisa klijentski sloj. Podrška za web servise u okviru *Java EE* zasnovana je na grupi *API*-ja povezanih sa tehnologijama web servisa: *Java API for XML Processing* (*JAXP*), *Java Architecture for XML Binding* (*JAXB*), *The SOAP with Attachments API for Java* (*SAAJ*), i *Java API for XML Web Services* (*JAX-WS*).

Tabela 2 – predstavlja najviše korišćene implementacije Java web servisa [WordPress1].

Name	Company	Latest version	License
JAX-WS Reference Implementation	Sun	2.2.1	open-source
Apache Axis2	Apache Software Foundation	1.5.2	open-source
Apache CFX	Apache Software Foundation	2.3.0	open-source

Tabela 2 – *Najviše korišćene implementacije Java web servisa*

Rezultati istraživanja performansi web servisa sprovedenih upravo sa ciljem upoređivanja *JAX-WS RI* i drugih implementacija mogu poslužiti kao dobra osnova za izbor konkretne implementacije. Istraživanje dostupno na portalu *Java* zajednice [Java1] uključilo je poređenje vremena izvršavanja operacija *JAX-WS RI* i *Axis2* web servisa, i prezentovalo rezultate po kojima su performanse *JAX-WS RI* web servisa značajno (od 30% do 100%) bolje prilikom izvršavanja svih operacija, bez obzira da li one vraćaju jednu povratnu vrednost ili skup od više vrednosti. Istraživanja sprovedena od strane *IBM* potvrđuju da su performanse *JAX-WS RI* web servisa značajno bolje u odnosu na performanse *Axis2* web servisa [IBM1], i neznatno bolje od CFX web servisa [IBM2], koji su se pokazali boljim od *JAX-WS RI* samo u operacijama sa velikom količinom povratnih podataka. Na osnovu predočenog, kao najbolji izbor za implementaciju *Java* web servisa nameće se *JAX-WS RI* implementacija, i zbog toga može biti referentna tačka za poređenje sa *.NET* implementacijama

3.2. *.NET web servisi*

Microsoft *.NET* tehnologija omogućava razvoj web servisa korišćenjem dva *API*-ja: *ASP.NET web services*, i *WCF* (*Windows Communication Foundation*). Za *WCF* [InfoM1], koji je predstavljen u verziji 3.5, bi se moglo reći da predstavlja funkcionalno i tehnološki nadograđeni *ASP.NET*, jer uključuje niz funkcionalnosti za koje je *ASP.NET* koristio druge komponente. Imajući u vidu da je *WCF* tehnologija zamišljena kao naslednik *ASP.NET* web servisa, prirodno je očekivati da su njene performanse pri izvršavanju na višem nivou. Ova pretpostavka potvrđena je i testiranjima sprovedenim od strane kompanije Microsoft [Microsoft1]. WCF implementacija pokazala se performantnijom za približno 35% pri izvršavanju standardnih poziva web servisa, i zbog toga je odabrana za poređenje sa JAX-WS web servisima u ovoj analizi.

4. DINAMIČKA ANALIZA

Proces dinamičke analize softverskog sistema podrazumeva istraživanje ponašanja programa na osnovu informacija prikupljenih tokom izvršavanja. U te svrhe definisan je plan testa koji uključuje:

1. Monitoring aplikacije
   a) Analizu upravljanja memorijom
   b) Analizu upotrebe procesorske snage
   c) Analiza upravljanja nitima
   d) Analiza upravljanja objektima
2. Analizu performansi aplikacije
   a) Analizu performansi kompletnog ciklusa izvršenja operacije web servisa
   b) Analizu performansi izvršavanja operacija poslovne logike

4.1 *Okruženje za analizu*

Imajući u vidu da je cilj dinamičke analize da se izvrši komparacija izvršavanja operacija web servisa, test će biti postavljen tako da se u što je moguće većoj meri izoluje aplikacija web servisa od ostalih faktora koji mogu da utiču na performanse.

Java je tehnologija koja je platformski nezavisna od operativnog sistema, a operativni sistem za pokretanje aplikacija u *.NET* tehnologiji je gotovo uvek *Microsoft Windows* [1], pa je očigledan izbor za operativni system za poređenje upravo Windows. Konkretno odabrana verzija je Window 7 32-bit.

Rešenje za potpunu izolaciju aplikacija bilo bi korišćenje aplikativnog servera koji uključuje podršku i za *Java* i za *.NET* obe tehnologiju, i na kom bi se mogle pokrenuti obe realizacije agenta web servisa. Međutim, s obzirom na potpuno drugačiju osnovu ovih tehnologija, aplikativni serveri se uvek namenjuju za jednu od njih. Kao rešenje se iz tog razloga nameće pokretanje aplikacija na različitim serverima, odabranim tako da pruže najbolje performanse. Dakle, izolovanu atomsku jedinicu posmatranja činiće par aplikacija – server. .NET web servisi će biti pokretani na serveru *Microsoft IIS* (*Internet Information Services*) verzija 6.1, koji je *de facto* standard za pokretanje *.NET* web aplikacija. Za pokretanje Java serverske aplikacije koristiće se *Apache Tomcat 6.0*, koji strogo postmatrano ne predstavlja aplikacioni server u punom značenju, već servlet kontejner koji omogućava izršavanje aplikacija koje se baziraju na servletima i *JSP* tehnologiji, kao i drugim *API*-jima *Java* tehnologije koji ne zavise od *EJB* kontejnera, pružajući tako bolje performanse i manju zahtevnost za resursima sistema.

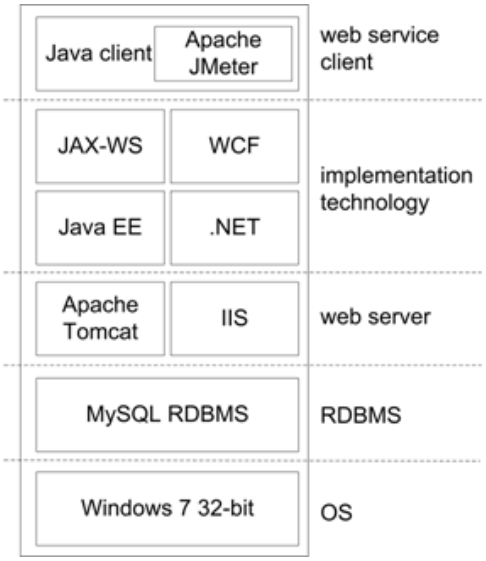
Podrška za rad sa različitim vrstama baza podaka zajednička je za *Java* i *.NET* tehnologije. Gotovo sve baze podataka koje su u široj upotrebi (*MySQL*, *Microsoft SQL Server*, *Oracle*, *Postgre* i druge) mogu se uz pomoć odgovarajućih *API*-ja kori-

[1] implementations of .NET for other operative systems do exists (Mono and Portable.NET), but neither provide full .NET framework support nor are used in any greater extent

stiti iz *Java* i *.NET* aplikacija. Na osnovu male zahtevnosti za resursima sistema za implementaciju studijskog primer izabran je *MySQL* sistem za upravljanje bazama podataka.

Da bi se izbegle bilo kakve razlličitosti u performansama klijenta web servisa, isti klijent će biti korišćen za pozive oba agenta web servisa. Zahvaljujući tome, možemo reći da će ponašanje klijentske aplikacije biti isto prilikom poziva oba servisa, odnosno da će uticaj klijenta na performanse biti podjednak nezavisno od toga da li se radi o web servisu realizovanom u *Java* ili *.NET* tehnologiji. To nam omogućava da razlike u izmerenim performansama posmatramo isključivo kao razlike u performansama web servisa

Kompletna platforma za dinamičku analizu prikazana je na slici 6.



Slika 6 – *Platforma za dinamičku analizu*

4.2. *Monitoring aplikacija*

Aplikacije će biti posmatrane u nekoliko stanja: 1) neaktivna aplikacija, kada nema poziva web servisa; 2) jedan poziv operacije sa jednom povratnom vrednošću; 3) 20 uzastopnih poziva operacije sa jednom povratnom vrednošću; 4) 1000 poziva operacije sa jednom povratnom vrednošću, od kojih 100 uzastopnih poziva sa 10 klijenata istovremeno; 5) 1000 poziva operacije sa više povratnih vrednosti, od kojih 100 uzastopnih poziva sa 10 klijenata istovremeno; 6) 10000 poziva operacije sa više povratnih vrednosti, od kojih 1000 uzastopnih poziva sa 10 klijenata istovremeno. Parametri posmatranja biće: 1) zauzeće procesora u procentima; 2) zauzeće heap memorije u MB; 3) zauzeće non-heap memorije u procentima; 4) broj aktivnih niti; 5) broj učitanih klasa.

Tabela 4 prikazuje rezultate dobijene obavljenim monitoringom aplikacije:

Scenario		Java - MAX	.NET - MAX
Neaktivna aplikacija	CPU (%)	0	1
	Heap (MB)	23	3.1
	Non-heap (MB)	21	110
	Niti (kom)	44	10
	Klase (kom)	3894	5985
Jedan poziv	CPU (%)	8	24
	Heap (MB)	24	3.3
	Non-heap (MB)	23	115
	Niti (kom)	45	13
	Klase (kom)	4102	6523
20 poziva	CPU (%)	3	8
	Heap (MB)	23	4
	Non-heap (MB)	23	115
	Niti (kom)	45	13
	Klase (kom)	4114	6526
1000 poziva	CPU (%)	17	17
	Heap (MB)	36	27
	Non-heap (MB)	24	133
	Niti (kom)	44	15
	Klase (kom)	4114	6528
1000 složenih poziva	CPU (%)	12	17
	Heap (MB)	41	27
	Non-heap (MB)	25	140
	Niti (kom)	44	15
	Klase (kom)	4121	6541
10000 složenih poziva	CPU (%)	18	23
	Heap (MB)	50	36
	Non-heap (MB)	26	146
	Niti (kom)	46	13
	Klase (kom)	4119	6541

Tabela 4 – *Rezultati monitoringa*

Na osnovu izvršenog monitoringa mogu se izvesti sledeći zaključci o ponašanju JAX-WS i WCF web servisa: 1) Upravljanje memorijom: a) Zauzeće heap memorije od strane web servis aplikacije nešto je veće kod web servisa realizovanog u Java tehnologiji. b) Zauzeće non-heap memorije znatno je veće kod web servisa realizovanog u .NET tehnologiji. c) Ukupno zauzeće memorije znatno je veće kod web servisa realizovanog u .NET tehnologiji. d) Aktivnost sakupljača đubreta češća je kod web servisa realizovanog u .NET tehnologiji. e) Efekti sakupljača đubreta su približno isti kod obe tehnologije. 2) Upotreba procesorske snage: a) Prosečno zauzeće procesora, kao i maksimalno zauzeće procesora veće je kod web servisa realizovanog u .NET tehnologiji. Prosečna razlika u zauzeću procesora je na nivou od 5%. b) Aktivnost sakupljača đubreta ima veći udeo u ukupnom zauzeću procesora kod .NET web servisa. 3) Upravljanja nitima: a) Broj aktivnih niti znatno je veći kod JAX-WS aplikacije. b) Varijacije broja aktivnih niti vrlo su male kod obe aplikacije (ispod 10%). 4) Upravljanje objektima a) Broj učitanih klasa veći je kod WCF aplikacije. b) Varijacije broja učitanih klasa su na približnom nivou za obe aplikacije (10%-15%).

4.3. *Analiza performansi*

U okviru analize performansi aplikacije biće posmatrano izvršavanje kompletnih operacija servisa i operacija poslovne logike. Analiza performansi podrazumeva posmatranje vremena izvršenja svake metoda koje se pozivaju u toku izvršenja operacije.

Analiza performansi obuhvatiće: 1) analizu izvršavanja kompletnih operacija web servisa pomoću profajlera; 2) analizu izvršavanja operacija poslovne logike pomoću profajlera, I 3) analizu izvršavanja kompletnih operacija web servisa preko Apache JMeter alata.

S obzirom da se web servis kao softverski sistem sastoji iz dve softverske aplikacije, za merenje vremena svih operacija potrebno je posmatrati i serverski deo, i klijenta web servisa. U okviru agenta web servisa biće posmatrano izvršavanje metoda poslovne logike, koje uključuju poziv sistemske operacije i interakciju sa bazom podataka preko brokera baze podataka. Posmatranje klijenta web servisa daće uvid u kompletan ciklus izvršavanja operacije web servisa.

Svi softverski profajleri imaju negativan uticaj na performanse aplikacija (eng. *overhead*). On se ogleda u usporavanju izvršavanja operacija izazvanim samim procesom merenja, i nastaje kao posledica aktivnih procesa koji vrše merenje i prikaz izmerenih rezultata, a koji, kao i svi drugi procesi, u određenoj mere troše resurse. Iz tog razloga će za kompletniju analizu biti korišćene i druge tehnike. Analiza izvršavanja kompletnih operacija biće obavljena pomoću *Apache JMeter* alata, koji omogućava prikaz vremena izvršenja ciklusa poziva operacije web servisa.

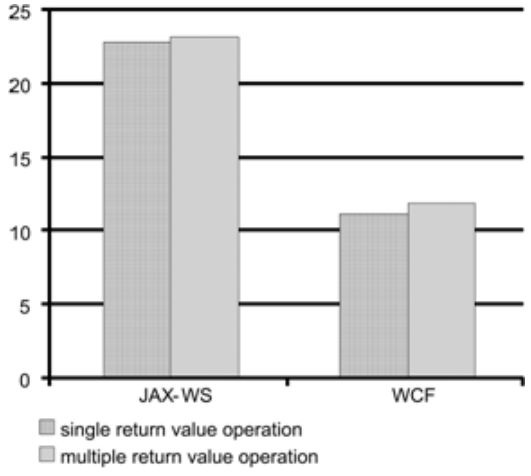
Tabela 5 prikazuje podatke o trajanju kompletnog poziva web servisa i vremenu izvršavanja operacije, na osnovu čega možemo zaključiti o performansama web servisa.

Tehnologija	Operacija	Kompletan poziv	Operacija	Δ (Mehanizam web servisa)
Java	Jedna povratna vrednost	22.5 ms	12.5 ms	10 ms
	Više povratnih vrednosti	35.5 ms	15.3 ms	20.5 ms
.NET	Jedna povratna vrednost	9.56 ms	3 ms	6.56 ms
	Više povratnih vrednosti	12.91 ms	4 ms	8.91 ms

Tabela 5 – *Rezultati analize performansi vršene softverskim profajlerima*

Razlike u performansama samih mehanizama web servisa su vrlo male, do 15 ms. Značajno bolje performanse WCF web servis je pokazao prilikom izvršavanja operacije sa više povratnih vrednosti, gde je 12 ms brži od JAX-WS web servisa, dok se performanse web servisa pri izvršavanju operacije sa jednom povratnom vrednošću razlikuju za nešto više od 3 ms.

Slika 3 prikazuje prosečna vremena izvršavanja operacije web servisa izmerena preko *JMeter* alata.



Slika 3 – *prosečna vremena izvršavanja operacije web servisa*

Očigledno je da su vremena izvršavanja posmatranih operacija WCF web servisa niža za u proseku 12ms od vremena izvršavanje istih operacija web servisa realizovanih preko JAX-WS tehnologije.

4.4. *Diskusija*

Predočeni rezultati impliciraju značajne razlike u ponašanju aplikacija realizovanih u različitim tehnologijama, pri čeum svaka od njih ima određene prednosti – JAX-WS web servisi u zauzeću memorije i procesora, kao i broju učitanih klasa, a WCF web servisi u zauzeću heap memorije i broju aktivnih niti. Iz perspektive monitoringa, ni jedna od tehnologija ne može se izdvojiti kao bolja, već bi na izbor mogle da utiču prvenstveno karakteristike sistema na kom treba izvršavati web servise, odnosno dostupni resursi.

Na osnovnu kompletne analize performansi, možemo zaključiti da su performanse .NET WCF web servisa nešto bolje od Java JAX-WS web servisa. Razlike su, na nivou jednog poziva, u redu veličine od 10ms, ali posmatrano pri izvršavanju većeg broja operacija donose značajniju prednost WCF tehnologiji, jer se poziv 100 operacija web servisa izvršava za 1 sekund brže od poziva 100 operacija JAX-WS web servisa.

5. ZAKLJUČAK I BUDUĆI PRAVCI ISTRAŽIVANJA

U svrhu pružanja kompletne slike o web servisima, predstavljen je model web servisa koji uključuje osnovne koncepte i tehnologije. Ovaj model ima ulogu formalnog opisivanja arhitekture web servisa. UML dijagram modela omogućava jasan i direktan uvid u elemente i veze modela, dajući sliku o funkcionalnom i implementacionom aspektu web servisa. Kao takav, on se može koristiti kao dobra osnova za detaljnije upoznavanje sa web servisima i povezanim tehnologijama.

Na osnovu izvršene dinamičke analize izveden je zaključak da aplikacija realizovana korišćenjem *Java JAX-WS* tehnologije ima nešto manje zauzeće sistemskih resursa, dok se aplikacija razvijena u *.NET WCF* tehnologiji izvršava nešto brže. Globalno gledano, izmerene razlike bile su veoma male i ne mogu dati definitivan odgovor na pitanje koja tehnologija je bolji izbor za realizaciju web servisa.

Budući pravci istraživanja mogli bi da budu proširivanje formalnog modela web. Model predstavljen u okviru ovog rada uključuje naznačajnije koncepte i tehnologije vezane za web servise, ali nije sveobuhvatan. S obzirom da web servisi imaju i napredne funkcionalnosti koje se realizuju korišćenjem odgovarajućih tehnologija i definisanih specifikacija, proširenjem modela u njega bi se uključili dodatni elementi i povećao broj opisanih veza. Takav model mogao bi da posluži kao celovita osnova za proučavanje arhitekture i tehnologija web servisa. Drugi pravac istraživanja bila bi komparativna analiza unapređenih tehnologija za razvoj web servisa. Pošto su web servisi relativno mlada tehnologija, postoji stalni napredak domenu novih funkcionalnosti definisanih preko specifikacija i optimizacije implementacija servisa. U bliskoj budućnosti

mogu se očekivati nove verzije implementacija *Java* i *.NET* web servisa, kao i implementacije namenjene drugim platformama, i poređenje njihovih performansi u moglo bi da dâ odgovore na pitanja koja tehnologija je više napredovala i da li se neka od njih izdvojila kao vodeća u oblasti web servisa.

LITERATURA:

[1] [Bean10] Bean, J., 2010.: SOA and Web Services Interface Design: Principles, Techniques and Standard, Elsevier Morgan Kaufmann Publishers.

[2] [Erl21] Erl, T., 2005. Service-Oriented Architecture, A Field Guide to Integrating XML and WebServices, Prentice Hall.

[3] [Graham5] Graham, S., Davis, D., et al, 2005. Building Web Services with Java, Sams Publishing

[4] [InfoM1] Minović, M., Radojčić, S., 2006. Arhitektura orijentisana ka servisima – Windows Communication Foundation (WCF), InfoM, br 19.

[5] [IBM1] Sosnoski, D., 2010. Java Web services: Metro vs. Axis2 performance <http://www.ibm.com/developerworks/java/library/j-jws11/index.html>.

[6] [IBM2] Sosnoski, D., 2010. Java web services: CXF performance comparison <http://www.ibm.com/developerworks/java/library/j-jws14/index.html>.

[7] [Java1] Kavaguchi, K., 2007. JAX-WS RI 2.1 benchmark detail http://weblogs.java.net/blog/kohsuke/archive/2007/02/jaxws_ri_21_ben.html, (20.9.2010).

[8] [Microsoft1] Gupta, S., 2007. A Performance Comparison of Windows Communication Foundation (WCF) with Existing Distributed Communication Technologies, <http://msdn.microsoft.com/en-us/library/bb310550.aspx#wcfperform_topic1>.

[9] [Santos08] Santos, R. L. T., Roberto, P. A., et al. 2008. A Web services-based framework for building componentized digital libraries, Journal of Systems and Software 81/5.

[10] [W3C1] W3C, 2007. SOAP Version 1.2 Part 1: Messaging Framework (Second Edition), <http://www.w3.org/TR/SOAP12>.

[11] [W3C2] W3C, 2004. Web Services Architecture, <http://www.w3.org/TR/ws-arch/>.

[12] [W3C3] W3C, 2007. Web Services Description Language (WSDL) Version 2.0, <http://www.w3.org/TR/2007/REC-wsdl20-20070626>.

[13] [WordPress1] SOAPRPC, 2007. Web service implementations <http://soaprpc.wordpress.com/2009/04/25/web-service-implementations/>.

M.Sc. Nebojša Ristin, Fakultet organizacionih nauka,
Kontakt: nebojsa.ristin@yahoo.com
Oblasti interesovanja: Softversko inženjerstvo, Web servisi, Java EE aplikacije,

Dr Sinisa Vlajić, docent FON-a
Kontakt:: vlajic@fon.bg.ac.yu
Oblasti interesovanja: razvoj informacionih sistema, uzori, softversko inženjerstvo

Mr Ilija Antović, Fakultet organizacionih nauka,
Kontakt: ilijaa@fon.bg.ac.rs
Oblasti interesovanja: Softversko inženjerstvo, Softverski paterni, Generatori koda, Softverski zahtevi, Kvalitet softvera

M.Sc. Miloš Milić, Fakultet organizacionih nauka,
Kontakt: mmilic@fon.bg.ac.rs
Oblasti interesovanja: Softversko inženjerstvo, Softverski paterni, Generatori koda, Softverski zahtevi, Web tehnologije,

Vojislav Stanojević, Fakultet organizacionih nauka
Kontakt: vojkans@fon.bg.ac.rs
Oblasti interesovanja: Softversko inženjerstvo, Softverski paterni, Generatori koda, Softverski zahtevi, Kvalitet softvera

UPUTSTVO ZA PRIPREMU RADA

1. Tekst pripremiti kao Word dokument, A4, u kodnom rasporedu 1250 latinica ili 1251 ćirilica, na srpskom jeziku, bez slika. Preporučen obim – oko 10 strana, single prored, font 11.
2. Naslov, abstakt (100-250 reči) i ključne reči (3-10) dati na srpskom i engleskom jeziku.
3. Jedino formatiranje teksta je normal, bold, italic i bolditalic, VELIKA i mala slova (tekst se naknadno prelama).
4. Mesta gde treba ubaciti slike, naglasiti u tekstu (Slika1...)
5. Slike pripremiti odvojeno, VAN teksta, imenovati ih kao u tekstu, radi identifikacije, u sledećim formatima: rasterske slike: jpg, tif, psd, u rezoluciji 300 dpi 1:1 (fotografije, ekranski prikazi i sl.), vektorske slike – cdr, ai, fh,eps (šeme i grafikoni).
6. Autor(i) treba da obavezno priloži svoju fotografiju (jpg oko 50 Kb), navede instituciju u kojoj radi, kontakt i 2-4 oblasti kojima se bavi.
7. Maksimalni broj autora po jednom radu je 5.

Redakcija časopisa Info M