

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Matija Podravec

**OTKRIVANJE I POSTAVLJANJE USLUGA
U SUSTAVIMA ZASNOVANIM NA
USLUGAMA**

MAGISTARSKI RAD

Zagreb, 2006.

Magistarski rad je izrađen na Zavodu za elektroniku,
mikroelektroniku, računalne i inteligentne sustave.

Mentor: prof. dr. sc. Siniša Srbljić

Magistarski rad ima 153 stranice.

Rad br. _____

Povjerenstvo za ocjenu u sastavu:

1. prof. dr. sc. Nikola Bogunović – predsjednik
2. prof. dr. sc. Siniša Srbljić – mentor
3. doc. dr. sc. Darko Huljenić – Ericsson Nikola Tesla, Zagreb

Povjerenstvo za obranu u sastavu:

1. prof. dr. sc. Nikola Bogunović – predsjednik
2. prof. dr. sc. Siniša Srbljić – mentor
3. doc. dr. sc. Darko Huljenić – Ericsson Nikola Tesla, Zagreb

Datum obrane: 26. siječnja 2006.

Zahvaljujem prof. dr. sc. Siniši Srbljiću na motivaciji tijekom izrade ovog rada.

Zahvaljujem kolegama Miroslavu Popoviću, Ivanu Bencu, Ivanu Skuliberu, Andri Milanoviću, Ivanu Gavranu, Dejanu Škvorcu i Danielu Skrobi na korisnim savjetima koji su pridonijeli kvaliteti teksta.

Posebno hvala Marini na svemu ostalom.

Sadržaj

1	UVOD.....	1
2	RAČUNARSTVO ZASNOVANO NA USLUGAMA	3
2.1	KONCEPT USLUGE	3
2.2	ARHITEKTURA ZASNOVANA NA USLUGAMA	6
2.3	TRŽIŠTE USLUGA.....	7
2.4	WEB USLUGE	9
2.4.1	<i>Osnovni standardi Web usluga.....</i>	10
2.4.2	<i>Sigurnost</i>	14
2.4.3	<i>Kompozicija usluga.....</i>	16
2.4.4	<i>Web usluge i Grid.....</i>	17
2.4.5	<i>Semantičke Web usluge</i>	19
2.5	USPOREDBA OBJEKATA, KOMPONENTI I WEB USLUGA	21
3	POSTAVLJANJE USLUGA.....	24
3.1	GENERIČKI PROCES POSTAVLJANJA PROGRAMSKE POTPORE	24
3.2	IZAZOVI U POSTAVLJANJU APLIKACIJA.....	27
3.3	PRISTUPI U POSTAVLJANJU USLUGA	28
3.3.1	<i>Postavljanje usluga opisano skriptom.....</i>	28
3.3.2	<i>Postavljanje usluga opisano jezikom.....</i>	29
3.3.3	<i>Postavljanje usluga opisano modelom</i>	30
3.3.4	<i>Postavljanje usluga upravljano agentima</i>	31
3.3.5	<i>Usporedba pristupa postavljanja usluga.....</i>	33
3.4	TEHNOLOGIJE POSTAVLJANJA USLUGA	34
3.4.1	<i>Distributed Ant</i>	35
3.4.2	<i>SmartFrog</i>	36
3.4.3	<i>Radia</i>	38
3.4.4	<i>Microsoft Systems Management Server 2003.....</i>	40
4	OTKRIVANJE USLUGA	43
4.1	ARHITEKTURE KATALOGA USLUGA.....	43
4.1.1	<i>Centralizirana arhitektura</i>	44
4.1.2	<i>Hijerarhijska arhitektura</i>	46
4.1.3	<i>Arhitektura zasnovana na mrežama ravnopravnih sudionika</i>	48
4.1.4	<i>Otkrivanje usluga u zgodom oblikovanim i pokretnim mrežama</i>	52
4.2	ODABIR USLUGA	54
4.2.1	<i>Odabir usluga na osnovi kategorija</i>	54
4.2.2	<i>Semantički odabir usluga</i>	55
4.2.3	<i>Kvaliteta odabira usluge</i>	58
5	PRIVIDNA RASPODIJELJENA RAČUNALNA OKOLINA	61
5.1	ARHITEKTURA PRVIDNE RASPODIJELJENE RAČUNALNE OKOLINE	61
5.1.1	<i>Prividna logička mreža</i>	63
5.1.2	<i>Sustav za povezivanje usluga.....</i>	64
5.1.3	<i>Sustav za siguran pristup uslugama</i>	67
6	SUSTAV ZA POSTAVLJANJE I OTKRIVANJE USLUGA.....	69
6.1	FUNKCIONALNOSTI SUSTAVA ZA POSTAVLJANJE I OTKRIVANJE USLUGA	69
6.1.1	<i>Otkrivanje usluga</i>	69

6.1.2	<i>Spremanje usluga</i>	70
6.1.3	<i>Postavljanje usluga</i>	70
6.1.4	<i>Konfiguriranje usluga</i>	71
6.2	ŽIVOTNI CIKLUS APLIKACIJSKIH USLUGA	71
6.3	ARHITEKTURA SUSTAVA ZA POSTAVLJANJE I OTKRIVANJE USLUGA	73
6.3.1	<i>Konceptualna arhitektura usluge Skladište</i>	75
6.3.2	<i>Konceptualna arhitektura usluge Katalog</i>	77
6.3.3	<i>Konceptualna arhitektura usluge Postavljач</i>	82
7	PROGRAMSKO OSTVARENJE SUSTAVA ZA POSTAVLJANJE I OTKRIVANJE USLUGA	84
7.1	KORIŠTENE TEHNOLOGIJE	84
7.1.1	<i>.NET radni okvir</i>	85
7.1.2	<i>Relacijske baze podataka</i>	86
7.1.3	<i>Prijenos binarnih podataka SOAP protokolom</i>	89
7.1.4	<i>Sigurnost pokretnog izvršnog kôda</i>	90
7.2	APLIKACIJSKE USLUGE	91
7.3	USLUGA SKLADIŠTE	93
7.3.1	<i>Programska arhitektura</i>	94
7.3.2	<i>Struktura instalacijske skripte</i>	97
7.3.3	<i>Ostvarenje</i>	98
7.4	USLUGA KATALOG	99
7.4.1	<i>Programska arhitektura</i>	100
7.4.2	<i>Organizacija podataka u relacijskoj bazi podataka</i>	104
7.4.3	<i>Organizacija podataka u XML katalogu</i>	106
7.4.4	<i>Ostvarenje</i>	108
7.5	USLUGA POSTAVLJAČ	110
7.5.1	<i>Programska arhitektura</i>	110
7.5.2	<i>Postupak postavljanja aplikacijske usluge</i>	112
7.5.3	<i>Postupak uklanjanja aplikacijske usluge</i>	117
7.5.4	<i>Postupak konfiguriranja aplikacijske usluge</i>	118
7.5.5	<i>Ostvarenje</i>	119
7.6	PRAĆENJE POGREŠAKA	120
7.7	WEB KORISNIČKO SUČELJE	121
7.8	INSTALACIJA SUSTAVA ZA POSTAVLJANJE I OTKRIVANJE USLUGA	123
8	ZAKLJUČAK	126
9	LITERATURA	128
10	ŽIVOTOPIS	137
11	SAŽETAK	138
12	SUMMARY	139
13	KLJUČNE RIJEČI	140
14	DODATAK A	141
15	DODATAK B	150

1 Uvod

U posljednjih desetak godina globalni informacijski sustav Internet postao je jedan od osnovnih načina komunikacije i razmjene elektroničkog sadržaja među ljudima. Zbog velikog interesa korisnika razvijene su brojne elektroničke usluge koje omogućuju kvalitetnu komunikaciju među ljudima te različite oblike elektroničkog poslovanja. Tijekom izvođenja složenih poslovnih procesa elektroničke usluge međusobno komuniciraju. Raznorodnost računalnih platformi, programskih jezika i operacijskih sustava otežava mogućnost jedinstvene komunikacije elektroničkih usluga.

Tehnologije računarstva zasnovanog na uslugama pojednostavnjuju komunikaciju elektroničkih usluga. Računarstvo zasnovano na uslugama nova je paradigma razvoja raspodijeljenih aplikacija prema kojoj se aplikacije grade povezivanjem postojećih usluga. Tehnologije računarstva zasnovanog na uslugama definiraju otvorene i standardizirane protokole koji omogućuju komunikaciju raznorodnih usluga. Osim komunikacijskog protokola, za razvoj i izvođenje složenih raspodijeljenih aplikacija potrebni su i dodatni mehanizmi. Primjer takvih mehanizama su mehanizmi za otkrivanje i postavljanje usluga.

Otkrivanje usluga je postupak pronađaska informacija o uslugama dostupnim unutar računalne okoline. Postupkom otkrivanja usluga pronađaze se sve informacije nužne za njeno uspješno korištenje, poput lokacije i opisa programskog sučelja. Na osnovi prikupljenih informacija se tijekom razvoja raspodijeljene aplikacije ostvaruju veze između aplikacijskih usluga. Mehanizmi za otkrivanje usluga također se koriste i tijekom izvođenja raspodijeljene aplikacije s ciljem prilagođavanja aplikacije dinamičkim promjenama u računalnoj okolini. Na primjer, u slučaju nedostupnosti pojedine usluge, raspodijeljena aplikacija postupkom otkrivanja pronađazi prikladnu zamjensku uslugu koju koristi u nastavku svog izvođenja.

Postavljanje usluge je postupak instalacije izvodivog oblika usluge na računalo računalne okoline. Postupkom postavljanja usluga raspodijeljena se aplikacija priprema za izvođenje. Ručno postavljanje raspodijeljene aplikacije, zbog potencijalno velikog broja usluga i čvorova računalne okoline, vremenski je zahtjevan i pogreškama podložan postupak. Uporabom mehanizama za automatizirano postavljanje usluga smanjuje se vrijeme postavljanja raspodijeljene aplikacije te se smanjuje mogućnost pogrešaka. Mehanizmi za postavljanje usluga osnova su mehanizama za dinamičko prilagođavanje raspodijeljene aplikacije, poput upravljanja opterećenjem (engl. load balancing) i pogreškama (engl. fault tolerance). Na primjer, veliko

opterećenje pojedine usluge moguće je smanjiti postavljanjem njene preslike na slobodni čvor računalne okoline.

U magistarskom radu oblikovan je, opisan i ostvaren *Sustav za postavljanje i otkrivanje usluga*. Razvijeni sustav sastavni je dio *Prividne raspodijeljene računalne okoline* (engl. Virtual Distributed Environment) koja pruža potporu izgradnji, postavljanju i izvođenju raspodijeljenih aplikacija zasnovanih na uslugama. *Prividna raspodijeljena računalna okolina* razvijena je u suradnji tvrtke Ericsson Nikola Tesla d.d. i Zavoda za elektroniku, mikroelektroniku, računalne i inteligentne sisteme, Fakulteta elektrotehnike i računarstva u Zagrebu. *Sustav za postavljanje i otkrivanje usluga* pruža mehanizme za otkrivanje i postavljanje usluga. Sustav je oblikovan kao raspodijeljena aplikacija koja se sastoji od usluga *Katalog*, *Skladište* i *Postavljač*. *Katalog* ostvaruje potporu otkrivanju usluga, dok *Repozitorij* i *Postavljač* u suradnji s *Katalogom* ostvaruju potporu postavljanju aplikacijskih usluga.

Magistarski rad sastoji se od sedam poglavlja koja se grupiraju u dvije cjeline. Prva cjelina opisuje istraživanja i tehnologije u području postavljanja i otkrivanja usluga, a sastoji se od drugog do četvrtog poglavlja. Drugo poglavlje opisuje računarstvo zasnovano na uslugama s pripadnim tehnologijama Web usluga. Treće poglavlje opisuje i uspoređuje uobičajene pristupe u postavljanju usluga. Dodatno se iznose mogućnosti i svojstva dostupnih alata za postavljanje usluga. Četvrto poglavlje opisuje uobičajene arhitekture kataloga usluga te navodi metode odabira prikladne usluge iz skupa dostupnih usluga.

Druga cjelina opisuje praktični dio magistarskog rada, a sastoji se od petog do sedmog poglavlja. Peto poglavlje opisuje arhitekturu *Prividne raspodijeljene računalne okoline* te navodi ulogu i smještaj *Sustava za postavljanje i otkrivanje usluga* unutar računalne okoline. Šesto poglavlje opisuje funkcionalnosti i konceptualnu arhitekturu *Sustava za postavljanje i otkrivanje usluga*, dok sedmo poglavlje opisuje korištene tehnologije i programsko ostvarenje sustava. Na kraju rada izneseni su zaključak i pregled korištene literature.

2 Računarstvo zasnovano na uslugama

Računarstvo zasnovano na uslugama (engl. Service-Oriented Computing, SOC) [1] je programska paradigma prema kojoj se raspodijeljene aplikacije ostvaruju povezivanjem usluga smještenih na računalnoj mreži. Usluge su nezavisni programski blokovi koji putem otvorenih sučelja pružaju svoje funkcionalnosti. Sučelje se smatra otvorenim ako je ostvareno u skladu s javnim standardima. Izgradnja raspodijeljene aplikacije zasniva se na otkrivanju postojećih usluga te definiranju njihovog međudjelovanja. *Arhitektura zasnovana na uslugama* (engl. Service-Oriented Architecture, SOA) definira arhitekturu raspodijeljene aplikacije s obzirom na uobičajene uloge i međudjelovanja njenih sastavnih elemenata.

Standardi Web usluga (engl. Web Services) omogućuju ostvarenje raspodijeljenih aplikacija zasnovanih na uslugama. Osnovni skup standarda Web usluga definira jedinstven jezik za komunikaciju među uslugama te mehanizme za otkrivanje i opisivanje usluga. Standardi Web usluga zasnivaju se na XML (eXtensible Markup Language) jeziku koji se ustalio kao standard za strukturiranje podataka prisutnih na javnoj mreži Internet. Osim osnovnog skupa standarda definirani su i dodatni standardi koji omogućuju napredne uzorke međudjelovanja usluga, poput kompozicije usluga ili sigurne komunikacije.

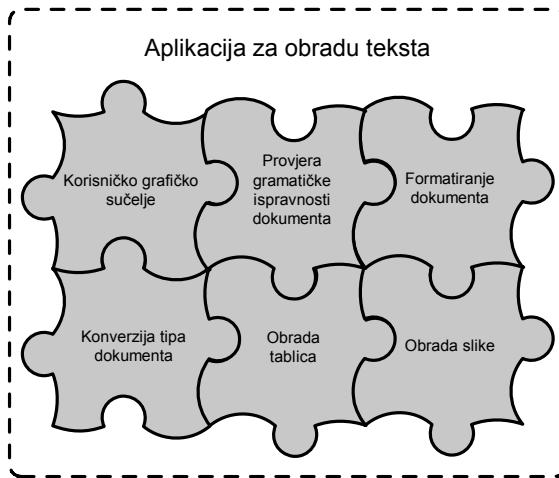
U nastavku poglavlja se na osnovi koncepta usluge naglašavaju razlike između tradicionalne izgradnje aplikacija i izgradnje zasnovane na uslugama. Nadalje, opisuje se arhitektura zasnovana na uslugama te se razmatraju svojstva tržišta usluga sa pripadnim zakonima ponude i potražnje. Na kraju poglavlja opisane su tehnologije Web usluga koje omogućuju ostvarenje arhitektura zasnovanih na uslugama.

2.1 Koncept usluge

Unatoč razvoju programskih jezika i razvojnih okruženja, osnovna paradigma izgradnje i održavanja programskih aplikacija nije se značajnije promijenila od 1960-tih godina. Proces izgradnje programske aplikacije sastoji se od pisanja programskog kôda u višem programskom jeziku, prevodenja u izvršni kôd te instaliranja izvršnog oblika aplikacije na korisničko računalo. Aplikacija se u tom procesu ostvaruje kao monolitni sustav koji se na korisničko računalo instalira sa svim potrebnim komponentama.

Primjer monolitnog sustava je aplikacija za obradu teksta. Korisnik aplikaciju kupuje u obliku CD-a s instalacijskim paketom kojeg instalira na osobno računalo. Aplikacija za obradu teksta sastoji se od mnoštva funkcionalnosti, poput gramatičke provjere teksta, pretvaranja

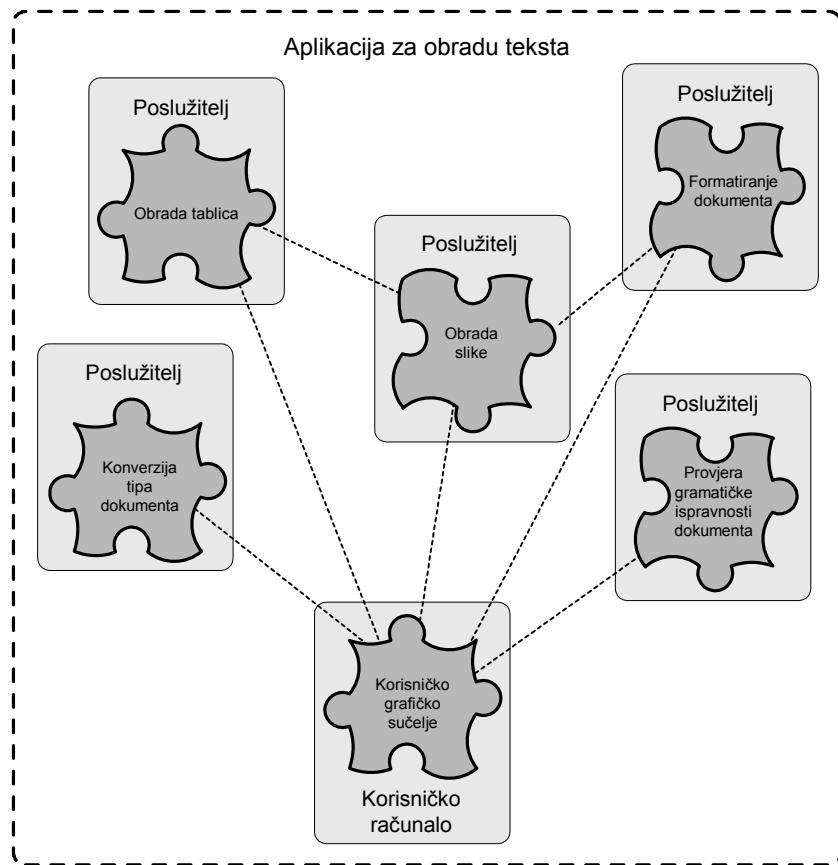
dokumenata iz jednog formata u drugi, obrade tablica i slično. Na korisničkom računalu aplikacija se instalira sa svim dostupnim funkcionalnostima, bez obzira što se pojedine funkcionalnosti vjerojatno nikad neće koristiti. Slika 2-1 naglašava čvrstu povezanost sastavnih dijelova aplikacije unutar monolitnog sustava.



Slika 2-1: Monolitna aplikacija za obradu teksta

Umrežavanje računala proširuje mogućnosti programskih aplikacija. Podaci i računalni procesi više se ne nalaze na jednom računalu već su raspodijeljeni po računalima u mreži. Raspodjela podataka i procesa na više računala zahtijeva novu paradigmu razvoja i održavanja računalnih aplikacija. Rezultat navedene potrebe je računarstvo zasnovano na uslugama. Prema računarstvu zasnovanom na uslugama aplikacija se ostvaruje povezivanjem usluga smještenih na mreži, čime se izvođenje aplikacije raspodijeljuje na više računala računalne mreže. Računarstvo zasnovano na uslugama uvodi i novi način plaćanja. Umjesto kupnje CD-a s instalacijskim paketom usluge, korisnik plaća iznajmljivanje aplikacije. Svako korištenje aplikacije posebno se naplaćuje.

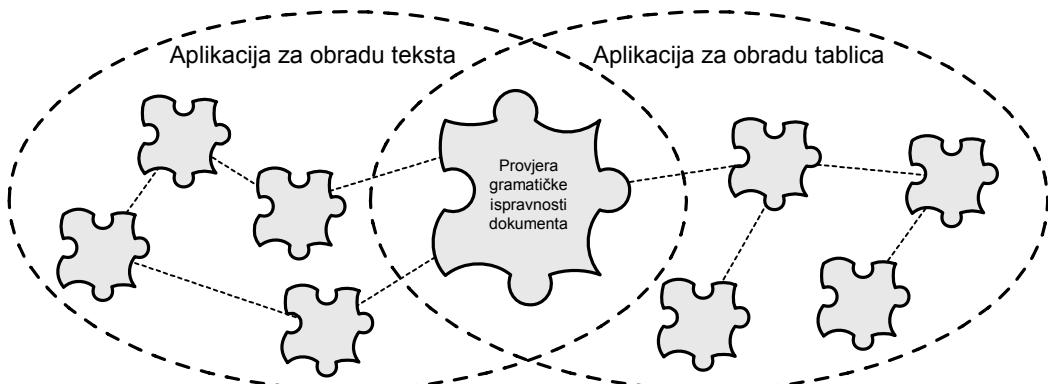
Slika 2-2 prikazuje aplikaciju za obradu teksta ostvarenu u skladu s računarstvom zasnovanom na uslugama. Pojedine funkcionalnosti aplikacije izdvajaju se i ostvaruju kao usluge. Povezivanje usluga ostvaruje se tijekom izvođenja aplikacije. Na primjer, ako se za vrijeme izvođenja raspodijeljene aplikacije pojavi potreba za gramatičkom provjerom dokumenta, onda aplikacija pronalazi potrebnu uslugu te putem otvorenih komunikacijskih protokola koristi njene funkcionalnosti. Ostvarenje raspodijeljene aplikacije pomoću usluga pojednostavnjuje održavanje aplikacije. Na primjer, ispravljanje uočene pogreške u raspodijeljenoj aplikaciji izvodi se za sve korisnike samo na jednom mjestu, odnosno unutar specifične usluge. U slučaju monolitne aplikacije, svaki korisnik samostalno dobavlja datoteke s ispravcima i instalira ih na svoje računalo.



Slika 2-2: Aplikacija za obradu teksta zasnovana na uslugama

Osnovno svojstvo računarstva zasnovanog na uslugama je interoperabilnost (engl. interoperability) usluga. Interoperabilnost usluga je mogućnost komunikacije usluga ostvarenih na raznorodnim računalnim platformama i operacijskim sustavima. Svojstvo interoperabilnosti omogućuju sučelja usluga ostvarena u skladu s otvorenim i javnim standardima. Informacije o strukturi ulaznih i izlaznih poruka tako ostvarenog sučelja usluge dovoljne su za uspješno pristupanje usluzi. Ovo su ujedno i jedine vidljive informacije o programskom ostvarenju usluge.

Sljedeće svojstvo računarstva zasnovanog na uslugama je slaba povezanost usluga (engl. loose coupling). Svojstvo slabe povezanosti naglašava autonomnost usluga unutar raspodijeljene aplikacije. Usluge se ostvaruju kao zasebne cjeline nezavisne od poslovnog procesa aplikacije pa ih je moguće upotrijebiti i u drugim raspodijeljenim aplikacijama. Na primjer, uslugu gramatičke provjere dokumenta moguće je iskoristiti i u aplikaciji za obradu tablica. Slika 2-3 prikazuje primjer uporabe usluge za gramatičku provjeru dokumenta u aplikaciji za obradu tablica i u aplikaciji za obradu teksta. Za razliku od slabo povezanih, jako povezane usluge ovisne su o poslovnom procesu aplikacije te njihova uporaba izvan konteksta tog poslovnog procesa nije učinkovita.



Slika 2-3: Uporaba iste usluge u različitim aplikacijama

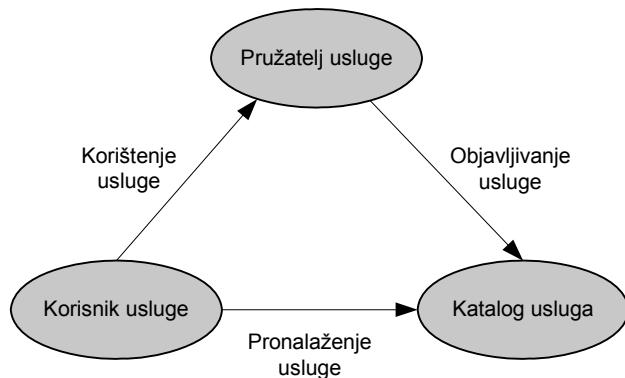
2.2 Arhitektura zasnovana na uslugama

Arhitektura zasnovana na uslugama definira se na osnovi karakterističnih međudjelovanja entiteta raspodijeljene aplikacije. Entiteti aplikacije u arhitekturi zasnovanoj na uslugama mogu poprimiti jednu od tri osnovne uloge: *pružatelj usluge*, *korisnik usluge* i *katalog usluga*.

Pružatelj usluge je organizacija koja posjeduje uslugu i ostvaruje poslovnu logiku usluge. Korisnik usluge je organizacija koja putem otvorenih sučelja poziva i koristi poslovnu logiku usluge. Ako pružatelj usluge koristi neku drugu uslugu, onda on istovremeno poprima i ulogu korisnika usluge. Pružatelji i korisnici usluga nisu dužni održavati znanje o ostalim uslugama u mreži. Znanje o uslugama u mreži spremi se u katalog usluga. Svaka usluga spremi u katalog usluga svoj opisnik koji sadrži informacije nužne za uspješnu komunikaciju s uslugom.

Između entiteta arhitekture zasnovane na uslugama postoje sljedeći oblici međudjelovanja: *objavljivanje usluge*, *pronalaženje usluge* i *korištenje usluge*. Slika 2-4 prikazuje entitete arhitekture zasnovane na uslugama i pripadne oblike međudjelovanja.

Pružatelj usluge nakon postavljanja usluge na mrežu objavljuje njezino postojanje. Tijekom objavljivanja usluge opisnik usluge se spremi u katalog usluga. Informacije u opisniku usluge dijele se u tri osnovne kategorije: *poslovne informacije*, *informacije o funkcionalnosti usluge* i *tehničke informacije*. Poslovne informacije specificiraju organizaciju koja je vlasnik usluge. Informacije o funkcionalnosti usluge definiraju funkcionalna svojstva usluge. Tehničke informacije opisuju format ulaznih i izlaznih poruka usluge. Na osnovi poslovnih informacija i informacija o funkcionalnosti usluga korisnik pretražuje katalog usluga. Na osnovi tehničkih informacija korisnik uspostavlja komunikaciju s uslugom.



Slika 2-4: Arhitektura zasnovana na uslugama

Pronalaženje usluge sastoji se od *pretraživanja* i *odabira* usluge. Korisnik prilikom pretraživanja kataloga usluga postavlja upit za određenim uslugama. Kriteriji za pretraživanje mogu biti opis funkcionalnosti usluge, kvaliteta usluge, ugled usluge i slično. Procesom odabira se iz skupa usluga dobivenih pretraživanjem izabire prikladna usluga.

Posljednji proces je korištenje usluge. Komunikacija između korisnika i pružatelja usluge uspostavlja se dinamički na osnovi tehničkih informacija iz opisnika usluge. Tijekom komunikacije korisnik i usluga razmjenjuju poruke zahtjeva i odgovora.

2.3 Tržište usluga

Povećanjem broja usluga i njihovih korisnika uspostavlja se tržište usluga. Tržište se definira kao mjesto gdje prodavači i kupci razmjenjuju dobra ili usluge. Osnovna arhitektura sustava zasnovanih na uslugama korisniku omogućuje promjenu pružatelja usluge. Ako korisnik nije zadovoljan kvalitetom ostvarene usluge, onda u svakom trenutku može odabrati novog pružatelja usluge. Na tržištu usluga pružatelji usluga oglašavaju usluge koje pružaju, dok korisnici pregovaraju s pružateljima usluga i koriste usluge. Pregovaranje je potraga za uslugom koja svojim funkcijskim i nefunkcijskim osobinama najbolje odgovara korisniku.

Proces pregovaranja, kao ključni proces tržišta usluga, sastoji se od tri dijela: *predpregovaranje*, *pregovaranje* i *korištenje usluge* [17]. Tijekom predpregovaranja korisnik prikuplja informacije o tržištu. Usluge se na tržištu međusobno razlikuju prema kombinaciji funkcijskih i nefunkcijskih osobina. Uspoređivanje usluga na osnovu funkcijskih osobina provodi se jednostavnom provjerom da li usluga ostvaruje ili ne ostvaruje određenu funkcionalnost. Problem se javlja prilikom usporedbe usluga na osnovi nefunkcijskih osobina koje mogu poprimiti širok raspon vrijednosti. Tijekom predpregovaranja korisnik prema vlastitim potrebama uspoređuje usluge raspoložive na tržištu i odabire prikladnu uslugu.

U procesu pregovaranja korisnik i usluga razmjenjuju poruke s ciljem postizanja dogovora. Za vrijeme pregovaranja utvrđuju se uvjeti korištenja usluge, razmjenjuju se sigurnosni podaci, utvrđuje se identitet korisnika, dogovara se očekivana kvaliteta usluge i slično. Rezultat pregovaranja je ugovor o korištenju usluge.

Ako su korisnik i usluga tijekom pregovaranja postigli dogovor, slijedi proces korištenja usluge. Na osnovi rezultata dobivenih korištenjem usluge i ugovora postignutog pregovaranjem, korisnik stvara izvještaj o kvaliteti usluge i spremi ga u vlastitu bazu znanja o uslugama. Iskustvo o korištenju usluge također utječe na buduće korištenje usluge. Ako je korisnik nezadovoljan kvalitetom usluga, onda će u budućnosti odabrati novog pružatelja usluge.

Proces pregovaranja usko je povezan s svojstvima tržišta usluga. Tržište usluga obilježeno je sljedećim svojstvima: *zrnatost usluga, kritičnost vremena, korištenje složenih usluga, nepotpuno znanje, nepredvidljivost i uspoređivanje usluga*.

Zrnatost usluge je određena opsežnošću njenih funkcionalnosti. Pružatelji usluga moraju pažljivo odabrati prikladnu zrnatost usluga. Funkcionalnosti usluga trebaju biti dovoljno male opsežnosti kako bi korisnici mogli precizno definirati složene radnje koje se sastoje od poziva više različitih usluga. S druge strane, mala zrnatost usluga uzrokuje povećanje njihove brojnosti na tržištu, čime se otežava proces pregovaranja.

Korisnicima je bitno u što kraćem vremenu doći do usluge. Korisnici mogu pregovarati s pružateljima usluga prilikom svake potrebe za nekom uslugom. Ovakvo pregovaranje rezultira najpovoljnijim ugovorom u tom trenutku, ali ujedno traje duže. Korisnicima je tijekom pregovaranjem prikladnije postići ugovor koji vrijedi i za nekoliko sljedećih poziva usluge.

Složene usluge nastaju povezivanjem više jednostavnih usluga. Primjer složene usluge je usluga za planiranje putovanja koja se sastoji od usluge za rezervaciju avionskog leta, usluge za rezervaciju hotela te usluge za iznajmljivanje automobila. Složene usluge otežavaju proces pregovaranja, jer je teže utvrditi nefunkcijske osobine složene usluge koje ovise o nefunkcijskim osobinama svih njenih sastavnih usluga.

Sudionici velikih tržišta teško mogu održavati znanje o ostalim sudionicima tržišta. U najboljem slučaju mogu održavati znanje o podskupu sudionika. Uz nepotpuno znanje o sudionicima tržišta javlja se i problem povjerenja. Pojedini pružatelji usluga mogu lagati o osobinama svojih usluga. Jedno od rješenja za održavanje znanja i povjerenja je uvođenje nezavisnih tijela koja prikupljaju informacije o sudionicima tržišta te izdavanjem ovjenih iskaznica jamče kvalitetu usluga.

Stanje tržišta usluga je nepredvidljivo. Pružatelji usluga i korisnici slobodno ulaze i izlaze s tržišta, zbog čega se neprestano mijenja trenutna ponuda i potražanja za uslugama. Česte

promjene u ponudi i potražnji usluga istovremeno uzrokuju česte promjene u cijeni korištenja usluga. Također, usluge zbog tehničkih problema mogu postati nedostupne određeno vrijeme. Dinamičnost tržišta usluga potiče korisnike na često pregovaranje za najprikladnjom uslugom.

Uspoređivanje usluga ostvaruje se na osnovi funkcijskih i nefunkcijskih osobina, poput cijene, pouzdanosti ili dostupnosti usluge. Tijekom pregovaranja korisnik treba uzeti u obzir što više osobina usluga te na njihovoj osnovi odabrati optimalnu uslugu. Funkcijski jednake usluge ne moraju pružati isti skup nefunkcijskih osobina što unosi dodatne poteškoće u proces pregovaranja.

2.4 Web usluge

Skup standarda Web usluga omogućuje ostvarenje arhitekture zasnovane na uslugama. Definiranje standarda Web usluga rezultat je udruženog rada mnogih proizvodača programske potpore, kao što su primjerice Microsoft, IBM, Sun, BEA, SAP i drugi. Osim industrije, razvoju standarda Web usluga pridonosi i akademска zajednica. Raznorodnost razvojne zajednice pridonijela je pojavi različitih pogleda na tehnologije Web usluge pa stoga postoje i različite definicije Web usluga. Slika 2-5 prikazuje neke od definicija Web usluga.

Dijelovi poslovne logike dostupni putem javne mreže Internet korištenjem otvorenih standarda. (Microsoft)

Slabo povezane programske komponente koje međusobno dinamički komuniciraju putem standardnih Internet tehnologija. (Gartner)

Aplikacije programske potpore identificirane URI-jem, čija se sučelja i veze mogu definirati, opisati i otkrivati putem XML dokumenata te koje podržavaju direktnu interakciju s ostalim aplikacijama programske potpore putem poruka zasnovanih na XML jeziku i protokola zasnovanih na Internetu. (W3C)

Slika 2-5: Definicije Web usluga

Početni razvoj Web usluga potaknut je uspjehom World Wide Web tehnologija, u prvom redu HTTP protokola [74] i HTML jezika [77]. HTTP protokol omogućuje prijenos podataka javnom mrežom Internet, dok HTML jezik omogućuje njihov prikaz. Osnovna svojstva HTTP protokola i HTML jezika su njihova jednostavnost, zasnovanost na tekstu te dostupnost programskih ostvarenja za različite operacijske sustave i računalne platforme.

U ostvarivanju međudjelovanja između računalnih programa, standardi Web usluga primjenjuju ideje i principe koji se uspješno koriste u World Wide Web standardima. Poput World Wide Web standarda, Web usluge ostvaruju komunikaciju putem skupa osnovnih

protokola i zasnovane su na tekstu. Svi dokumenti vezani uz uslugu, poput opisa usluge ili komunikacijske poruke između usluga, tekstualno su formatirane. Osnovna razlika između Web usluga i World Wide Web standarda je djelokrug djelovanja. HTTP protokol i HTML jezik prvenstveno su namijenjeni prikazu podataka ljudskim korisnicima, dok su Web usluge namijenjene razmjeni podataka između računalnih programa.

2.4.1 Osnovni standardi Web usluga

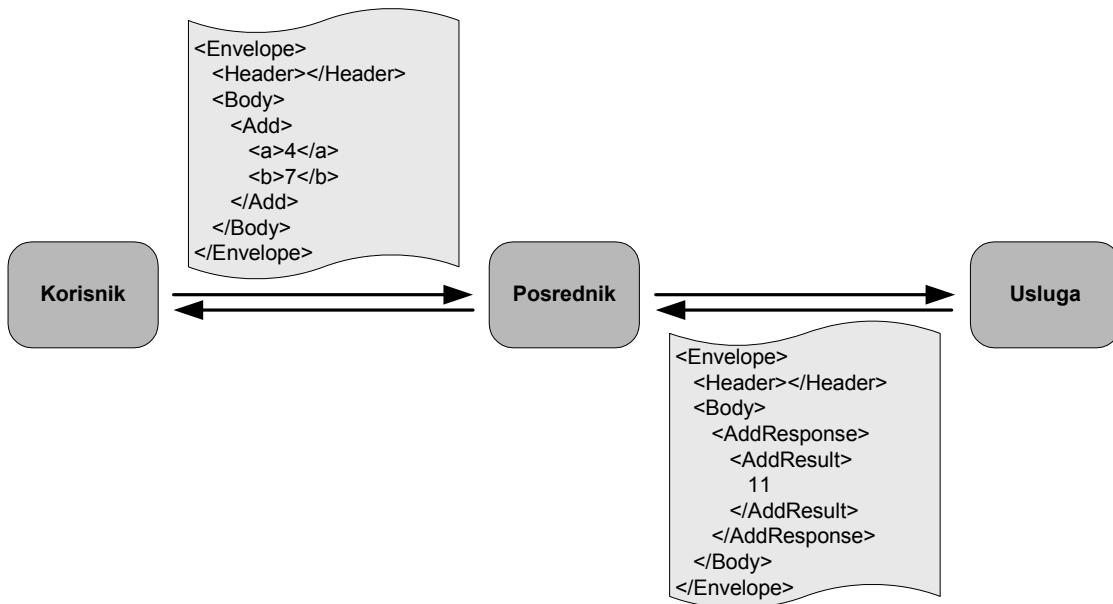
Standardi Web usluga omogućuju ostvarenje arhitektura zasnovanih na uslugama. Arhitektura zasnovana na uslugama, prikazana na slici 2-4, ostvaruje se sljedećim standardima Web usluga: SOAP, WSDL i UDDI. SOAP standard definira komunikacijski protokol za razmjenu podatka između usluge i korisnika. WSDL opisuje sučelje Web usluge, dok UDDI definira protokol za registraciju i pretraživanje kataloga usluga. U nastavku se detaljnije opisuje svaki od standarda.

SOAP

SOAP [79] je komunikacijski protokol Web usluga. Primarna namjena protokola je razmjena strukturiranih informacija u raspodijeljenoj okolini. Web usluge se često pogrešno smatraju infrastrukturom za komuniciranje raspodijeljenim objektima. Osnovna razlika između sustava za komuniciranje raspodijeljenim objektima i Web usluga je životni ciklus. Objekti u sustavima za komuniciranje raspodijeljenim objektima imaju životni ciklus: na korisnički zahtjev se instanciraju, korisnik im putem javnih metoda mijenja stanje te se kasnije na korisnički zahtjev ili djelovanjem unutarnjih mehanizama uništavaju. Opisani životni ciklus ne postoji kod Web usluga. Web usluge su stalno aktivne i spremne za obradu ulaznih XML dokumenata. Svaki ulazni XML dokument u pravilu je nezavisan od prijašnjih dokumenata koji su stigli Web usluzi na obradu. Detaljna analiza razlika između Web usluga i sustava za komuniciranje raspodijeljenim objektima može se pronaći u [10].

SOAP protokol zasniva se na razmjeni XML poruka. Svaka XML poruka sastoji se od tri elementa: *omotnica* (engl. envelope), *zaglavljje* (engl. header) i *tijelo* (engl. body) poruke. Omotnica je korijenski element u koji se ugrađuju neobavezno zaglavljje i obavezno tijelo poruke. Za razliku od ostalih mrežnih arhitektura, gdje aplikacijska logika nema pristup upravljačkim informacijama mrežne infrastrukture smještenim u zaglavljju poruke, kod SOAP protokola je aplikacijskoj logici dopušten pristup u zaglavljju poruke. Pristup zaglavljju poruke osnova je prilagodljivosti SOAP protokola.

Poruka se na svom putu do odredišta može djelomično obrađivati na jednom ili više posredničkih čvorova. Podaci namijenjeni posrednicima smještaju se unutar zaglavlja poruke, dok se unutar tijela poruke smještaju podaci namijenjeni odredišnom primatelju. Slika 2-6 pruža primjer komunikacije korisnika i usluge SOAP porukama putem posredničkog čvora. U ovom slučaju korisnik poziva uslugu za zbrajanje dva broja. Gornja poruka je SOAP poruka zahtjeva, a donja poruka je SOAP poruka odgovora. Parametri ulazne i izlazne poruke prenose se u tekstualnom XML formatu. Pošto su u ovom primjeru zaglavlja SOAP poruka prazna, posrednički čvor ne obrađuje poruke već ih prosljeđuje sljedećim entitetima u komunikacijskom lancu.



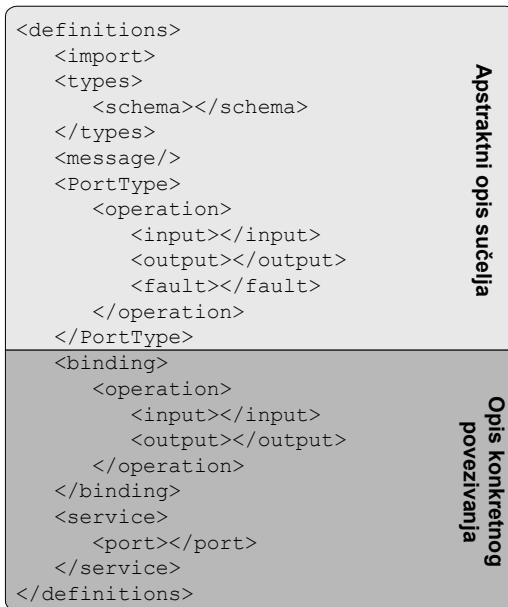
Slika 2-6: Komunikacija SOAP porukama

SOAP protokol je nezavisan o transportnom protokolu. Trenutno je uobičajena metoda prijenosa SOAP poruka HTTP protokolom, iako je poruke moguće prenositi i drugim protokolima, poput TCP [77] ili SMTP protokola [76].

WSDL

WSDL (Web Service Description Language) [80] jezik je standard za opisivanje sučelja Web usluga. Jezik je zasnovan na XML formatu pa je stoga neovisan o računalnoj platformi i pogodan za automatsku računalnu obradu. Slika 2-7 prikazuje osnovnu strukturu WSDL dokumenta. Potpun WSDL opis Web usluge sadrži dvije vrste informacija: *apstraktni opis sučelja* (engl. abstract description) i *opis konkretnog povezivanja* (engl. concrete binding information).

Apstraktni opis sučelja definira strukturu ulaznih i izlaznih poruka usluge. Nadalje, ovaj dio sučelja određuje i uzorke razmjene poruka tijekom komunikacija usluge i korisnika. Na početku apstraktog opisa sučelja navode se strukture tipova podataka koji se pojavljuju u komunikaciji (element `<types>`). Navedeni tipovi podataka se nakon toga koriste u definiranju struktura ulaznih i izlaznih poruka (element `<message>`). Definirane strukture poruka koriste se u opisivanju mogućih međudjelovanja korisnika i usluge (element `<portType>`), tako da se za svaku operaciju Web usluge navodi tip ulazne i izlazne poruke.



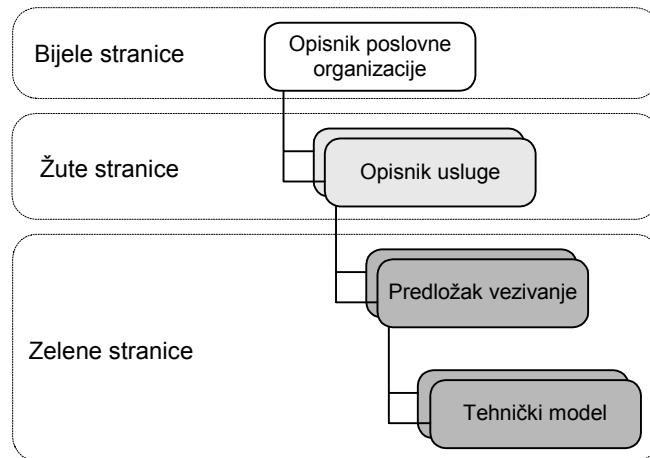
Slika 2-7: Struktura WSDL dokumenta

Opis konkretnog povezivanja definira specifičnosti fizičke razmjene poruka između korisnika i usluge. Unutar opisa konkretnog povezivanja određuje se komunikacijski protokol te format zapisa poruka (element `<binding>`). Nadalje, definira se i fizička adresa Web usluge (element `<service>`).

WSDL jezik nije dovoljan za kvalitetan opis svih svojstava Web usluge. Primjer svojstava Web usluge koja se ne mogu izraziti WSDL jezikom su: operacijska svojstva usluge (npr. usluga podržava SOAP inačicu 1.2), dostupnost usluge (npr. usluga je dostupna svaki dan osim srijede) ili sigurnosna svojstva usluge (npr. usluga koristi kriptiranje podataka). Za opisivanje dodatnih funkcionalnih i nefunkcionalnih svojstava Web usluga razvijene su dodatne specifikacije koje nadopunjaju mogućnosti WSDL jezika. Primjer takve specifikacije je WS-Policy [88] koja omogućuje definiranje odredbi (engl. policy) za opisivanje zahtjeva i mogućnosti usluge.

UDDI

UDDI (Universal Description, Discovery and Integration) [81] standard definira mehanizme za otkrivanje i registraciju Web usluga putem središnjeg kataloga usluga. Standard određuje strukturu informacija o uslugama te programsko sučelje (engl. application program interface, API) za pretraživanje i promjenu sadržaja kataloga. Pristup UDDI katalogu ostvaruje se SOAP porukama koje sadrže UDDI upite za pretraživanje i osvježavanje kataloga. Informacije o uslugama se prema analogiji s telefonskim imenikom dijele u tri vrste: "bijele stranice" koje sadrže podatke o pružateljima usluge, "žute stranice" koje sadrže podatke o kategoriji usluga te "zelene stranice" koje sadrže tehničke podatke o uslugama.



Slika 2-8: Struktura podataka UDDI kataloga

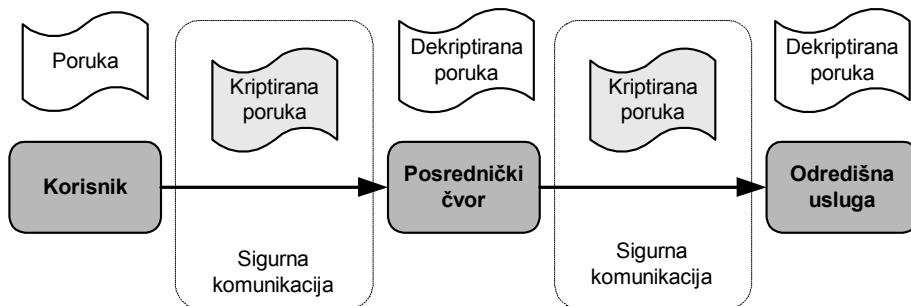
Slika 2-8 prikazuje strukturu informacija o Web uslugama unutar UDDI kataloga. *Opisnik poslovne organizacije* (engl. businessEntity) odgovara "bijelim stranicama" i sadrži podatke poput imena i kontakt informacija o pružatelju usluge. Svaki *Opisnik poslovne organizacije* sadrži jedan ili više *Opisnika usluge* (engl. businessService) koji odgovaraju "žutim stranicama". Svaki *Opisnik usluge* sadrži jedan ili više *Predložaka vezivanja* (engl. bindingTemplate) koji definiraju pristupne točke Web usluga. Unutar predloška vezivanja nalazi se *Tehnički model* (engl. tModel) koji sadrži informacije koje odgovaraju "zelenim stranicama". U *Tehničkom modelu* nalaze se podaci o sučelju Web usluge, poput adrese na kojoj se nalazi WSDL opis sučelja usluge. Struktura *Tehničkog modela* dovoljno je prilagodljiva za spremanje informacija i o bilo kojem drugom sredstvu, a ne samo o Web uslugama.

Postoje tri kategorije UDDI kataloga: *javni*, *organizacijski* i *međuorganizacijski*. Javni katalog (engl. UDDI Business Registry, UBR) zajednički održava više organizacija, pod vodstvom Microsofta, IBM-a i SAP-a. Javni katalog dopušta svim pružateljima usluga na javnoj mreži Internet registraciju usluga. Organizacijski katalog sadrži samo informacije o uslugama

koje pripadaju istoj organizaciji, a međuorganizacijski katalog sadrži informacije o uslugama organizacija koje djeluju kao poslovni partneri.

2.4.2 Sigurnost

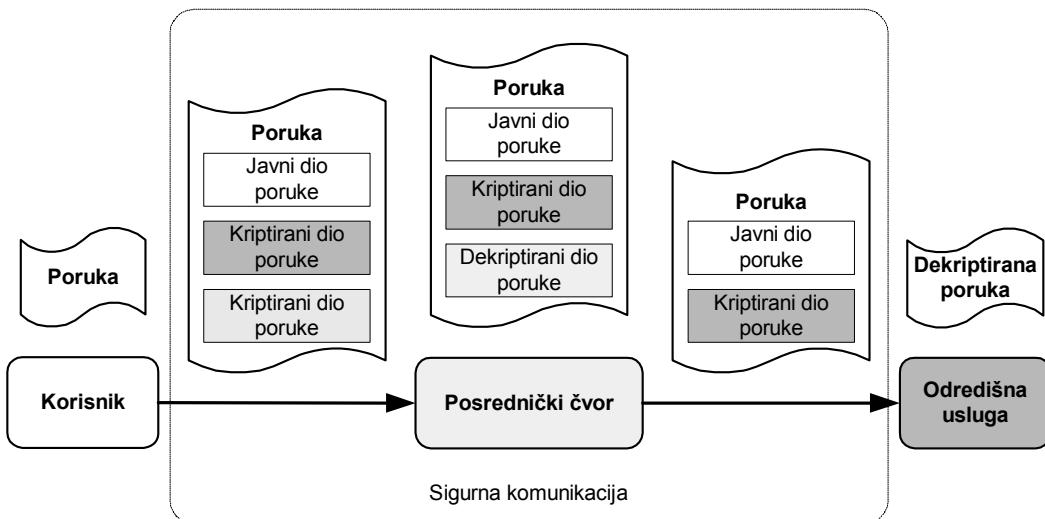
Sigurnost je osnovni zahtjev raspodijeljenih računalnih sustava smještenih na javnoj mreži Internet. Neki od sigurnosnih zahtjeva su: autentikacija, autorizacija, praćenje rada, cjelovitost poruke te povjerljivost poruke. Autentikacija je provjera identiteta pojedinog entiteta. Autorizacijom se potvrđuju ovlasti entiteta za korištenjem zatražene usluge. Praćenje rada obuhvaća neprekidno praćenje i spremanje podataka o korištenju usluge. Cjelovitost poruke zahtijeva da se poruka ne može promijeniti tijekom prijenosa, a da odredišni čvor to ne primjeti. Povjerljivost poruke zahtijeva da poruku tijekom transporta ne može pročitati neki treći neautorizirani entitet.



Slika 2-9: Prijenos SOAP poruka sigurnim prijenosnim protokolom

Najjednostavniji način ostvarenja sigurne komunikacije Web usluga je slanje SOAP poruka putem sigurnog prijenosnog protokola, poput SSL (Secure Socket Layer) [82] ili TLS (Transport Layer Security) [83] protokola. Slika 2-9 prikazuje prijenos SOAP poruke sigurnim prijenosnim protokolom. Sigurni prijenosni protokoli kriptiraju i digitalno potpisuju poruke koje se šalju mrežom te time ostvaruju zahtjeve cjelovitosti i povjerljivosti poruka. Međutim, uporaba sigurnog prijenosnog protokola nije prikladna za aplikacije zasnovane na Web uslugama, jer sigurni prijenosni protokol ostvaruje sigurnu komunikaciju samo između dviju točaka. SOAP poruke na svom putu do odredišta moguće je djelomično obraditi i na posredničkim čvorovima. Ako se upotrijebi siguran prijenosni protokol, onda je poruke na posredničkim čvorovima potrebno u potpunosti dekriptirati. Posrednički čvorovi ne moraju pripadati istoj organizaciji kao i odredišni čvor pa stoga nije primjerno da posrednički čvor ima pristup dijelovima poruke koji nisu namijenjeni isključivo njemu. Dakle, sigurna komunikacija se na ovaj način ostvaruje samo po dijelovima komunikacijskog lanca, jer posrednički čvorovi imaju pristup cjelovitoj poruci.

Opisani problem rješava WS-Security specifikacija [84]. WS-Security je skup nadogradnji SOAP protokola koji omogućuju autentikaciju, integritet i povjerljivost poruka. U oblikovanju WS-Security specifikacije posebna pažnja je posvećena proširivanju dodatnim sigunosnim protokolima. WS-Security ostvaruje sigurnost na razini poruke. Integritet poruke ostvaruje se uporabom digitalnih potpisa, pri čemu se digitalno mogu potpisati različiti dijelovi poruke. Povjerljivost poruke ostvarena je kriptiranjem pojedinih dijelova poruke. Autentikacija je ostvarena uporabom digitalnih potpisa. Slika 2-10 prikazuje uporabu WS-Security specifikacije u komunikaciji korisnika i Web usluge putem posrednika. Korisnik šalje SOAP poruku u kojoj su posebno kriptirani dijelovi namijenjeni posredničkom čvoru i posebno su kriptirani dijelovi namijenjeni odredišnoj usluzi. Posrednički čvor može dekriptirati samo dijelove poruke namijenjene njemu. Iz primljene poruke posrednički čvor uklanja njemu pripadne dijelove i promijenjenu poruku šalje odredišnoj usluzi. Usluga dekriptira preostale dijelove poruke i izvodi zadane operacije. Opisana metoda ostvaruje sigurnu komunikaciju od početnog do odredišnog čvora.

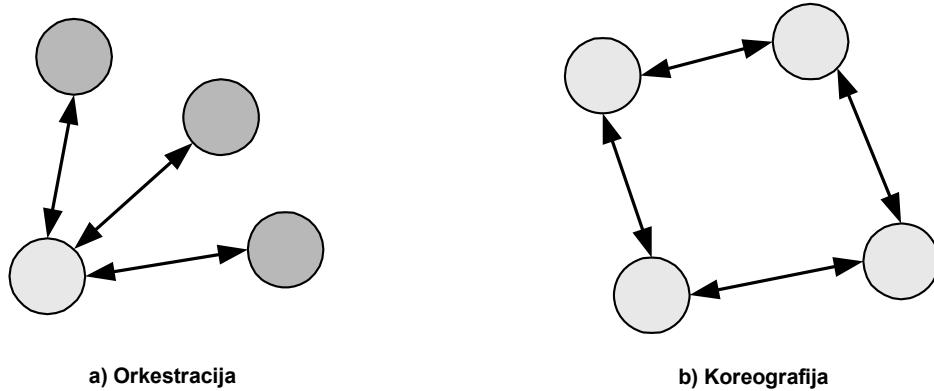


Slika 2-10: Sigurna komunikacija zasnovana na WS-Security standardu

Osim WS-Security specifikacije razvijene su i dodatne specifikacije koje ostvaruju specifične zahteve sigurnosti Web usluga. Većina dodatnih specifikacija su proširenja WS-Security specifikacije. Tako WS-SecureConversation [85] specifikacija pruža mehanizme za ostvarivanje sigurne komunikacije unutar sjednica tijekom koje korisnik i usluga u kratkom vremenu razmjenjuju više poruka. WS-Trust [86] specifikacija definira mehanizme za upravljanjem sigurnosnim značkama (engl. tokens). WS-Federation [87] definira mehanizme za potvrđivanje identiteta mrežnog entiteta, korisničkih računa i ostvarivanje autorizacije informacije putem povjerljivih organizacijskih domena.

2.4.3 Kompozicija usluga

Složene poslovne aplikacije zasnivaju se na kompoziciji Web usluga. Kompoziciju usluga čini formalni opis izmjene poruka između Web usluga. Orkestracija i koreografija su dva pristupa ostvarenja kompozicije Web usluga. Orkestracija opisuje poslovni proces koji se sastoji od međudjelovanja Web usluga. Međudjelovanje Web usluga se odvijaju na razini poruka. Orkestracija predstavlja kontrolu poslovnog procesa iz stajališta jedne središnje stranke. Koreografija se zasniva na suradnji Web usluga, gdje svaka usluga definira vlastiti udio u cjelokupnom međudjelovanju. U koreografiji Web usluga nema centralne stranke. Slika 2-11 prikazuje koreografiju i orkestraciju kao pristupe u stvaranju kompozicije usluga.



Slika 2-11: Pristupi kompozicije usluga

Kompozicija usluga je iznimno važan koncept za uspješnu i široku uporabu tehnologija Web usluga. Razvijeno je više specifikacija za opisivanje kompozicije usluga. Trenutno najrašireniji mehanizam za opisivanje kompozicije Web usluga je BPEL (Business Process Execution Language) [89] jezik. BPEL jezik podržava izvršne i apstraktne modele procesa. Izvršni model procesa definira ponašanje sudionika tijekom izvođenja zadatka, što odgovara orkestraciji. Apstraktni model procesa definira razmjenu poruka između sudionika što odgovara koreografiji. BPEL jezik sadrži niz konstrukata kojima je moguće definirati sinkrone i asinkrone pozive Web usluga, definirati varijable za spremanje rezultata obrade usluge, upravljati iznimkama, sekvensijalno i paralelno pozivati više Web usluga te definirati petlje i grananja. BPEL jezik zasnovan je na XML formatu pa je stoga izrazito neprikladan za ručno opisivanje poslovnih procesa. Zbog toga se prilikom definiranja poslovnog procesa upotrebljavaju grafički alati koji pojednostavuju stvaranje BPEL opisa kompozicije usluga.

2.4.4 Web usluge i Grid

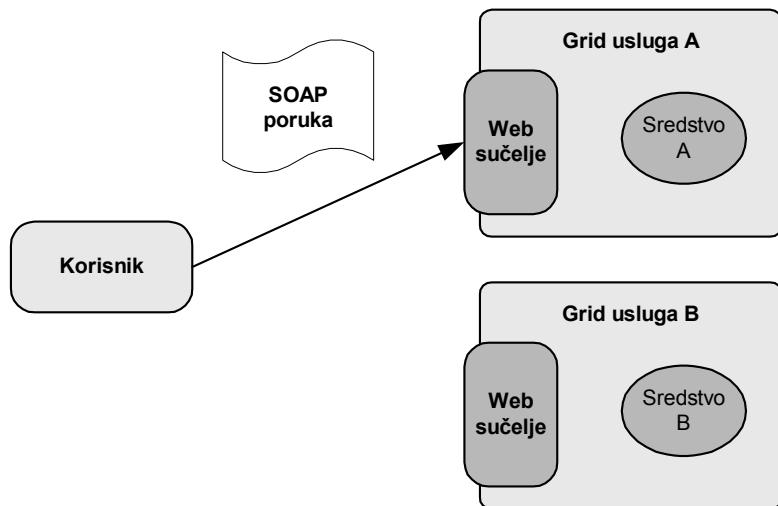
Grid tehnologije pružaju potporu dijeljenoj i koordiniranoj uporabi raznolikih sredstava u dinamičkim prividnim organizacijama (engl. virtual organisations). Prividna organizacija definira se kao grupa pojedinaca ili institucija koje su geografski raspodijeljene, ali djeluju kao jedna unificirana organizacija [14]. U početku su Grid tehnologije bile usmjerene prema znanstvenim aplikacijama, ali s vremenom im se primjena proširila i na poslovne aplikacije. Neke od primjena Grid računarstva su: računalno zahtjevne simulacije na udaljenim super-računalima, kooperativne simulacije velikih znanstvenih skupova podataka, raspodijeljeno procesiranje računarski zahtjevnih podatkovnih analiza te povezivanje znanstvenih instrumenata sa udaljenim računalima i arhivama podataka.

Za izgradnju Grid aplikacija razvijeno je više programskih okruženja, od kojih je najpoznatiji Globus Toolkit [74]. Globus Toolkit pruža infrastrukturu za sigurnost, gospodarenje informacijama, gospodarenje sredstvima, upravljanje iznimkama u sustavu te pouzdanu komunikaciju.

U početku su se Grid tehnologije i tehnologije Web usluga nezavisno razvijale. Međutim, Web usluge su ostvarile interoperabilnost između aplikacija ostvarenih na različitim platformama što je jedan od ključnih zahtjeva Grid računarstva. Grid tehnologije ostale su usko vezane uz specifične računalne platforme. Težnja za konvergencijom Grid tehnologija i tehnologija Web usluga rezultirala je Open Grid Services Architecture (OGSA) modelom [14]. OGSA model se zasniva na procesu prividnosti (engl. virtualisation), odnosno na procesu omatanja sredstva javnim otvorenim sučeljem. Proces prividnosti omogućava jednoličan pristup sredstvima koja postoje na raznorodnim računalnim platformama. Sredstva u OGSA modelu mogu biti fizička (procesor, proces, diskovni prostor, datotečni sustav) ili logička (iskaznice, ugovori). Sredstva imaju svojstva koja su promjenjiva s vremenom, kao na primjer postotak opterećenost procesora ili veličina slobodnog prostora na tvrdom disku. Sredstva također mogu biti dinamički stvarana i uništavana. Na primjer, sredstvo može biti proces koji se pokreće na korisnički zahtjev, a prekida nakon ostvarenog procesiranja.

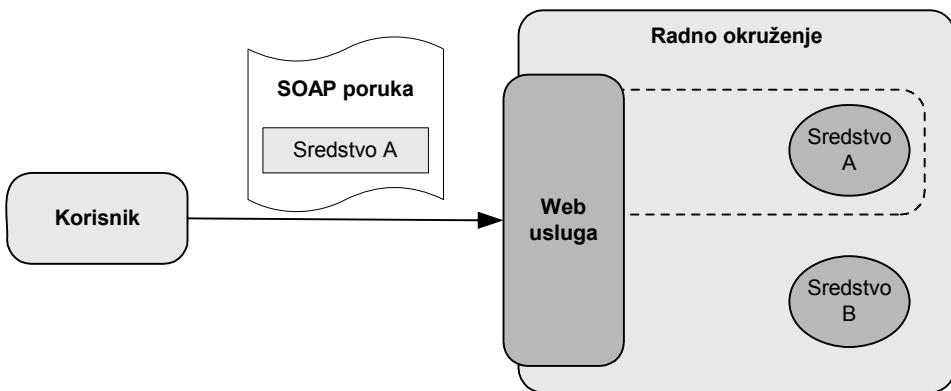
Web usluge u svom osnovnom obliku nisu prikladne za modeliranje sučelja sredstva. Prema SOAP protokolu i WSDL jeziku svaka poruka zahtjeva nezavisna je od prijašnjih poruka. Zbog toga se za Web usluge kaže da ne čuvaju stanje (engl. stateless). Omatanje sredstva sučeljem zasnovanim na SOAP i WSDL specifikacijama nije dovoljno za unificiran pristup svojstvima sredstava i za kontrolu životnog ciklusa sredstva. Iz tog razloga predložena je OGSI (engl. Open Grid Service Infrastructure) specifikacija [90]. OGSI specifikacija uvodi koncept Grid usluge, što je proširenje osnovnog oblika Web usluga. Grid usluge uvode standardizirano

sučelje za pristup vrijednostima svojstava te za upravljanje životnim ciklusom usluge, što je osnova za ostvarivanje Web usluga sa čuvanjem stanja (engl. stateful). Za razliku od Web usluga koje nemaju životni vijek i cijelo vrijeme su spremne za obradu SOAP poruka, Grid usluge imaju životni vijek te se stvaraju i uništavaju prema potrebi. Slika 2-12 prikazuje pristup korisnika instanci prividnog sredstva. Svaka instanca sredstva korisnicima je vidljiva kao zasebna Grid usluga. Korisnik pristupa pojedinoj instanci sredstva slanjem poruka na njezino Web sučelje.



Slika 2-12: Pristup sredstvu prema OGSI specifikaciji

OGSI specifikacija nije se u potpunosti uklopila sa postojećim standardima Web usluga. Specifikacija je uvela proširenje Web usluga u smjeru sustava raspodijeljenih objekata. Sustavi raspodijeljenih objekata (npr. CORBA [91] i DCOM [92]) intenzivnije su korišteni u prošlosti i nisu se ispostavili prikladnim rješenjem na globalnoj mreži Internet. Specifikacije Web usluga su oblikovane s ciljem da riješe probleme sustava raspodijeljenih objekata. Usklađivanje OGSI specifikacije i osnovnih standarda Web usluga rezultiralo je WSRF radnim okvirom (engl. Web Services Resource Framework) [15]. WSRF je skup specifikacija koje ostvaruju iste ciljeve kao i OGSI specifikacija, ali drugačijim pristupom. Kod WSRF radnog okvira razdvajaju se Web usluga i sredstvo. Dok je kod OGSI specifikacije svaka instanca sredstva posjedovala vlastito Web sučelje, kod WSRF radnog okvira sve instance iste vrste sredstva posjeduju zajedničko Web sučelje. Slika 2-13 prikazuje pristup pojedinoj instanci sredstva. Korisnik šalje SOAP poruku u kojoj navodi identifikator instance sredstva. Na osnovi identifikatora instance sredstva, Web usluga zaključuje za koju je instancu poruka namijenjena te nad njom izvodi operaciju navedenu u tijelu SOAP poruke.



Slika 2-13: Pristup sredstvu prema WSRF skupu specifikacija

WSRF skup specifikacija definira sučelja i format poruka za upravljanje životnim ciklusom sredstva, upravljanje svojstvima sredstva, definiranje grupa sredstava te ostvarivanje jednoznačnog upravljanja greškama. WSRF radni okvir osnova je trenutne inačice Globus Toolkit računalne platforme za izgradnju Grid usmjerenih računalnih sustava.

2.4.5 Semantičke Web usluge

Semantičke Web usluge (engl. semantic web services) su Web usluge čija se svojstva i mogućnosti opisuju na jednoznačan i računalu-razumljiv način. Opis mogućnosti i svojstava Web usluga osnova je za automatizaciju zadataka koje ljudi obično izvode ručno, poput otkrivanja i pozivanja usluga, te definiranja kompozicije usluga. Prilikom otkrivanja usluga ljudski korisnici pretražuju popis usluga te na osnovi kratkih opisa u prirodnom jeziku odabiru uslugu koja zadovoljava zadane uvjete. Pozivanje usluga također zahtijeva sudjelovanje ljudskog korisnika koji priprema ulazne parametre te tumači rezultat obrade. Definiranje kompozicije usluga obuhvaća operacije otkrivanja i pozivanja usluga pa samim time zahtijeva sudjelovanje ljudskog korisnika. Za automatizaciju navedenih zadataka nužno je ulazne i izlazne parametre te opis usluge opisati na računalu-razumljiv način.

Osnova za ostvarenje računalnog znanja su ontologije. Ontologije su oblici predstavljanja znanja o nekoj domeni. Unutar ontologije definiraju se ključni pojmovi, njihove semantičke veze i pravila zaključivanja. Ontologija se često izražava semantičkom mrežom, odnosno grafom čiji su čvorovi objekti domene, a veze grafa su odnosi među tim objektima. Mreža se dodatno proširuje svojstvima, ograničenjima, funkcijama i pravilima koja detaljno definiraju ponašanje objekata. Primjer ontologije je predstavljanje znanja o automobilu. Definiranje dijelova automobila (npr. kotač, motor, volan), njihovog smještaja unutar automobila (npr. klipovi su dio motora) te njihovog međudjelovanja (npr. motor pokreće kotače, volan usmjerava kotače) dovoljno je za konceptualizaciju znanja o automobilu.

Za predstavljanje ontologija razvijeno je više jezika, poput RDF (Resource Description Framework) [23], RDF Schema [24], DAML (DARPA Agent Modeling Language) [22], DAML+OIL (DAML + Ontology Interchange Language) [21] te u novije vrijeme OWL (Web Ontology Language) [25] jezik. Na osnovi OWL jezika razvijena je OWL-S [26] ontologija koja je posebno prilagođena opisivanju svojstava Web usluga.

OWL-S ontologija

Cilj OWL-S ontologije je definirati semantičke pojmove kojima se detaljno opisuju svojstva Web usluga. OWL-S ontologija sastoji se od tri međusobno povezane ontologije: *profil usluge* (engl. service profile), *model procesa* (engl. process model) i *veza usluge* (engl. service grounding).

Profil usluge opisuje funkcionalnosti usluge u svrhu reklamiranja i otkrivanja usluge. Ovaj dio opisa odgovara "žutim stranicama" u UDDI katalogu usluga. Profil usluge definira tri osnovne vrste informacija: informacije o pružatelju usluge, informacije o funkcionalnostima usluge, te informacije o svojstvima usluge. Informacije o funkcionalnostima usluge definiraju ulaze, izlaze, uvjete i rezultate usluge. Na primjer, za uslugu "Internet knjižara" ulazi mogu biti naslov knjige, korisnikova adresa i broj korisnikove kreditne kartice, a uvjet usluge može biti valjanost korisnikove kreditne kartice. Izlaz usluge može biti elektronička potvrda o kupnji, a rezultat izvedene radnje je oduzimanje novca s kreditne kartice te slanje knjige korisniku. Informacije o svojstvima usluge definiraju različite nefunkcijske karakteristike usluge, poput kategorije usluge, ugleda usluge, geografskog položaja pružatelja usluge, procijenjenog maksimalnog vremena obrade korisničkih zahtjeva i slično. Uporabom konstrukata OWL jezika moguće je profil usluge proširiti informacijama koje su specifične za pojedino ostvarenje usluge.

Model procesa definira način na koji korisnik ostvaruje međudjelovanje s uslugom. Međudjelovanje korisnika i usluge definira se putem semantičkog opisa ulaznih poruka, te preduvjeta i posljedica izvođenja usluge. Uloga modela procesa je pružiti potporu odabiru, pozivanju, kompoziciji i nadgledanju usluge. Model procesa dijeli se u dva podrazreda: *atomske procese* (engl. atomic process) i *složeni procesi* (engl. composite process). Atomske procese su opis usluge koja prima ulaznu poruku, izvodi pripadni proces obrade, te vraća izlaznu poruku. Složeni proces sastoji se od više atomskega procesa. Za složeni proces je karakteristično održavanje stanja prilikom obrade korisničkih poruka kroz sastavne atomske procese. Unutar modela procesa također se definiraju ulazi, izlazi, uvjeti i rezultate usluge. OWL-S specifikacija predviđa da model procesa obuhvaća i proširuje definicije ulaza, izlaza, uvjeta i rezultata usluge koji su definirani unutar profila usluge.

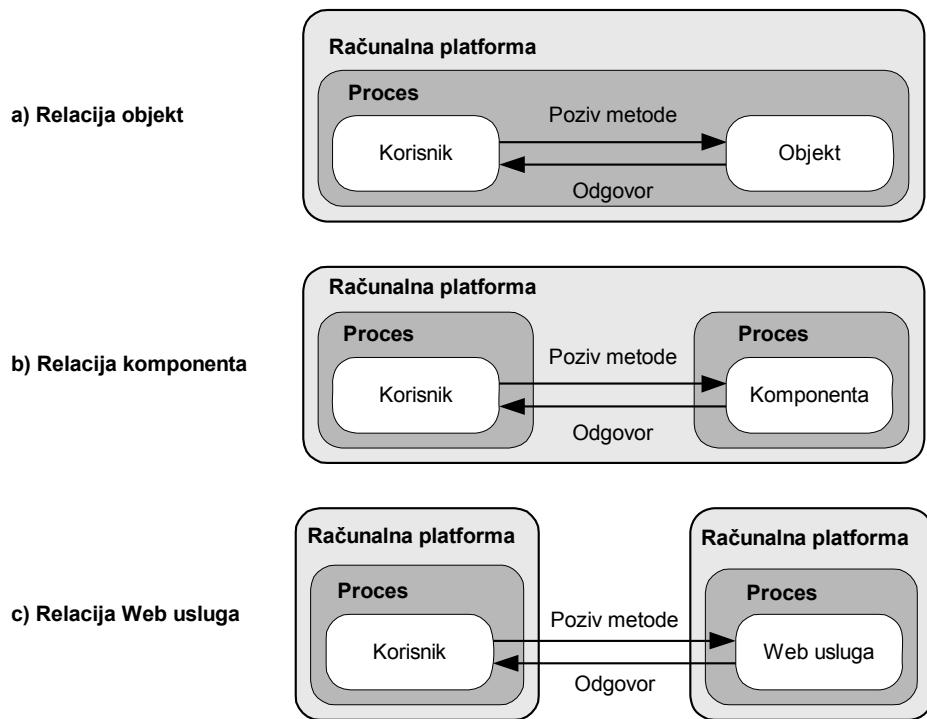
Veza usluge specificira način pristupa usluzi, odnosno specificira komunikacijski protokol, detaljne komunikacijske parametre te jednoznačan način razmjene podataka. Veza usluge zasniva se na WSDL opisu konkretnog povezivanja usluge koji se proširuje semantičkim informacijama.

2.5 *Usporedba objekata, komponenti i Web usluga*

Web usluge samo su jedan od načina izgradnje aplikacijskih blokova. Izvodivi aplikacijski blokovi mogu se ostvariti kao objekti, komponente ili Web usluge. Svaki model izgradnje ima prednosti i nedostatke, a time i prikladno područje primjene. Objekti, komponente i Web usluge imaju niz zajedničkih točaka: posjeduju sučelje koje opisuje njihove funkcionalnosti, izvode se unutar nekog procesa te im korisnici pristupaju pozivom metoda sučelja.

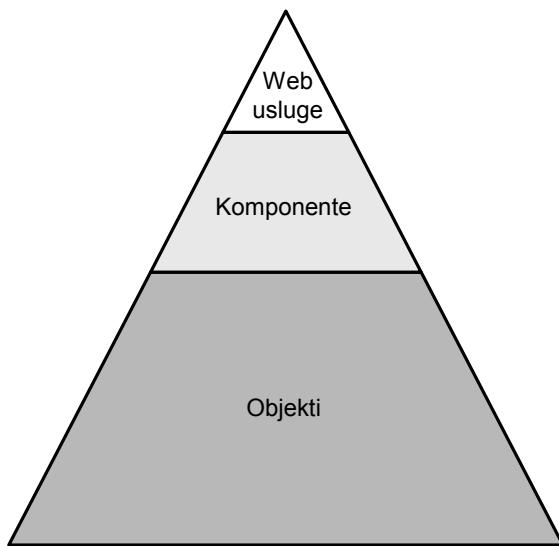
Razlike između navedena tri modela zasnivaju se na dva čimbenika: lokaciji i okruženju. Lokacija se odnosi na mjesto izvođenja procesa korisnika i procesa aplikacijskog bloka. Okruženje se odnosi na računalnu platformu na kojoj se izvode procesi korisnika i aplikacijskog bloka (npr. IBM WebSphere, Microsoft .NET). Kada su korisnik i aplikacijski blok smješteni unutar istog procesa, onda se govori o relaciji objekt. U tom slučaju aplikacijski blok i korisnik moraju biti u istom okruženju, jer se isti proces ne može istovremeno izvoditi na dvije računalne platforme. Ako se korisnik i aplikacijski blok izvode u različitim procesima, ali u istom okruženju, onda se govori o relaciji komponenta. Ako se korisnik i aplikacijski blok izvode u različitim procesima i različitim okruženjima onda se govori o relaciji Web usluga. Slika 2-14 prikazuje navedene osnovne razlike između objekata, komponenti i Web usluga.

Komunikacijski mehanizmi Web usluga omogućuju ostvarenje svih oblika komunikacije koji se također mogu ostvariti komunikacijskim mehanizmima objekata i komponenti. Glavni razlog za uporabu sva tri modela izgradnje aplikacijskih blokova je učinkovitost. Objekti se izvode u istom procesu kao i njihov korisnik pa je komunikacija između njih brza, jer korisnik i objekt imaju pristup istim memorijskim lokacijama. Komponente se izvode u različitom procesu od korisnika pa su za njihovu komunikaciju potrebni međuprocesni mehanizmi koji su sporiji od komunikacije među objektima. Web usluge koriste najsporiji oblik komunikacije jer sa korisnikom komuniciraju putem otvorenih protokola, a podatke zapisuju u tekstualni XML format.



Slika 2-14: Osnovne razlike između objekta, komponente i Web usluga

Brzina komunikacije između korisnika i aplikacijskog bloka određuje učestalost uporabe pojedinih modela izgradnje aplikacijskog bloka. Modeli koji omogućuju bržu komunikaciju s korisnikom češće se upotrebljavaju u odnosu na modele sa sporijom komunikacijom. Učestalost uporabe pojedinih modela može se prikazati piramidom na slici 2-15. Piramida naglašava hijerarhiju uporabe objekata, komponenti i Web usluge. Web usluge nalaze se na vrhu piramide i prema površini zauzimaju njen mali dio što je znak rijetke uporabe u odnosu na komponente i objekte. Komponente se nalaze u središtu piramide te se prema zauzetoj površini upotrebljavaju češće od Web usluga, ali rjeđe od objekata. Objekti se nalaze na dnu i zauzimaju dvije trećine površine piramide, što naglašava njihovu raširenu uporabu. Objekti se obično upotrebljavaju za izgradnju komponenata i Web usluga.



Slika 2-15: Hijerarhija uporaba objekata, komponenti i Web usluga

Uporaba komponenti je prikladna u izgradnji raspodijeljenih sustava unutar jedne organizacije. Računala unutar organizacije su povezana brzom lokalnom mrežom, pod kontrolom su iste administratorske domene te su zbog učinkovitosti orijentirana prema istoj računalnoj platformi. Za ostvarivanje izvršnih blokova koji komuniciraju između organizacija prikladno je koristiti Web usluge, jer organizacije ne koriste nužno iste računalne platforme, a Web usluge omogućuju komunikaciju između računalnih procesa ostvarenih za različite računalne platforme. Uz navedeno, Web usluge posebno su prilagodene komunikaciji putem globalne mreže Internet.

3 Postavljanje usluga

Postavljanje usluge (engl. service deployment) obuhvaća niz akcija kojima se usluga u obliku skupa izvršnih datoteka prenosi na odredišno udaljeno računalo, integrira u radnu okolinu i priprema za korištenje. Ako se postavljanje usluga složene raspodijeljene aplikacije izvodi "ručno", onda postupak postaje vremenski zahtjevan i podložan pogreškama. Ručno postavljanje usluga podrazumijeva da korisnik sve akcije izvodi alatima operacijskog sustava, poput alata za udaljenu kontrolu računala ili alata za prijenos datoteka. Prilikom ručnog postavljanja usluga pogreške se često pojavljuju u postupku podešavanja radnih parametara usluga. Glavni uzrok pogreškama je brojnost radnih parametara te njihova međusobna zavisnost.

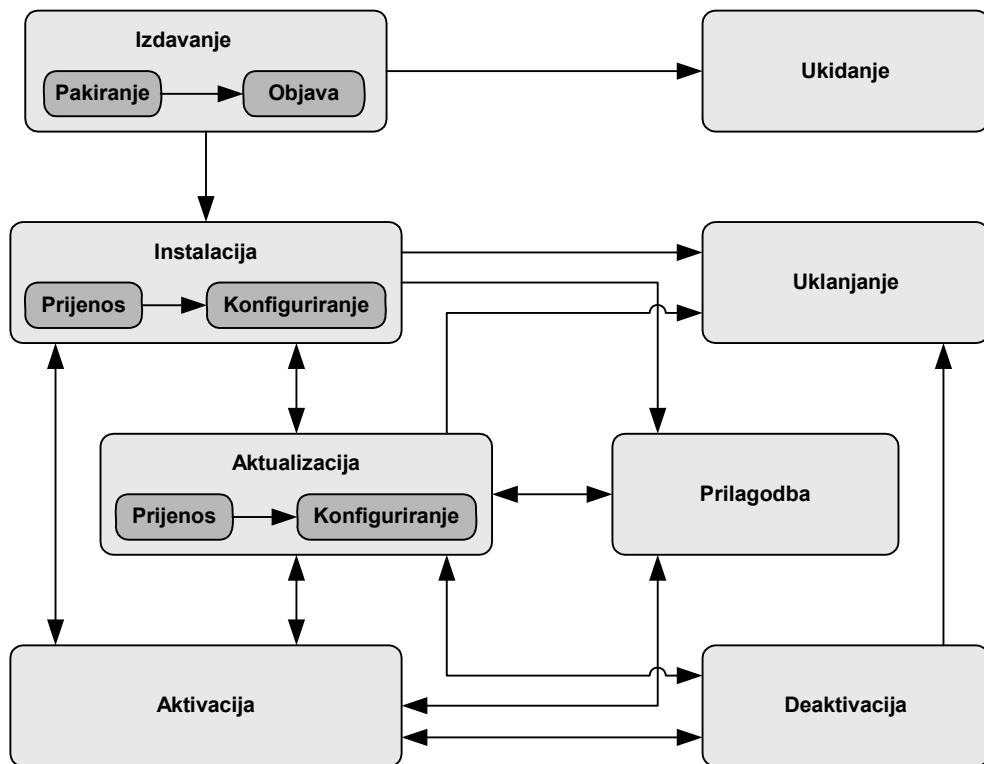
Specijalizirani alati za postavljanje usluga uklanjuju probleme ručnog postavljanja usluga. Automatizacija prijenosa i instalacije izvršnih datoteka smanjuje vrijeme postavljanja usluga, a podešavanje radnih parametara raspodijeljene aplikacije iz jednog središnjeg mesta smanjuje pojavu slučajnih pogrešaka. Uporaba specijaliziranih alata za postavljanje usluga omogućuje postavljanje raspodijeljenih aplikacija građenih od više stotina usluga na geografski raspodijeljena računala javne mreže Internet.

Postavljanje usluga poseban je slučaj općenitog postupka postavljanja programske potpore (engl. software). Sva razmatranja iznesena u nastavku poglavlja jednakov vrijede za postavljanje programske potpore kao i za postavljanje usluga. U nastavku poglavlja formalno se definira općenit proces postavljanja programske potpore, te se opisuju neki od izazova u procesu postavljanja. Automatizirano postavljanje usluga moguće je ostvariti primjenom više različitih pristupa. Za svaki od uobičajenih pristupa postavljanju usluga navode se osnovna svojstva i mogućnosti. Također, opisani pristupi postavljanju usluga uspoređuju se na osnovi kvantitativne i kvalitativne metrike. Na kraju poglavlja pruža se pregled trenutnih tehnologija i alata za postavljanje usluga.

3.1 Generički proces postavljanja programske potpore

Proces postavljanja programske potpore je složen proces. Slika 3-1 prikazuje proces postavljanja programske potpore koji se sastoji od sljedećih aktivnosti: *izdavanje* (engl. release), *instalacija* (engl. installation), *aktivacija* (engl. activation), *deaktivacija* (engl. deactivation), *aktualizacija* (engl. update), *prilagodba* (engl. adaptation), *uklanjanje* (engl. uninstallation) i *ukidanje* (engl. de-release). Strelice na slici označavaju moguće prijelaze između pojedinih aktivnosti. Na primjer, nakon izdavanja programske aplikacije moguće je izvesti aktivnosti

instalacije ili ukidanja. Detaljniju granulaciju aktivnosti nije moguće provesti zbog prevelike zavisnosti pojedinih aktivnosti o samoj prirodi aplikacije. Stoga se proces sa slike promatra kao generički proces postavljanja koji se specijalizira ovisno o potrebama i svojstvima specifičnog slučaja.



Slika 3-1: Generički proces postavljanja programske potpore

Izdavanje je prijelazna aktivnost između procesa razvoja (engl. development) i procesa postavljanja aplikacije. U ovoj aktivnosti se aplikacija priprema za prijenos na korisničko računalo. Potrebno je utvrditi koja sve sredstva aplikacija zahtijeva za uspješno izvođenje na korisničkom računalu. Ako korisničko računalo ne zadovoljava minimalne zahtjeve za sredstvima (npr. količina slobodnog prostora diskovnog spremnika), onda nema smisla postavljati aplikaciju na korisničko računalo.

Akcija izdavanja uključuje operaciju pakiranja (engl. packaging), odnosno pretvaranje aplikacije u oblik pogodan za prijenos do korisničkog računala. Instalacijski paket treba sadržavati izvršne datoteke aplikacije, opis aplikacije koji uključuje zahtjeve za potrebnim računalnim sredstvima i drugim vanjskim komponentama, zapis instalacijske procedure i sve ostale važne informacije potrebne za upravljanje aplikacijom. Drugi korak izdavanja je operacija objave aplikacije koja uključuje oglašavanje informacija o aplikaciji zainteresiranim korisnicima.

Instalacija obuhvaća postavljanje aplikacijskog izvršnog kôda na korisničko računalo. Ovo je najsloženija aktivnost u procesu postavljanja te ujedno i akcija s najširom potporom specijaliziranih alata. Instalacija se sastoji od dvije procedure. Prva procedura je *prijenos* datoteka od proizvođača do korisnika. Na primjer, datoteke je do korisnika moguće prenijeti putem CD-a ili dohvaćanjem sa javne mreže Internet. Druga procedura je *konfiguriranje* instalirane aplikacije. Konfiguriranje obuhvaća podešavanje početnih radnih parametara aplikacije.

Aktivacija je akcija pokretanja izvršnih komponenti aplikacije na korisničkom računalu. Za jednostavnu aplikaciju aktivacija podrazumijeva pokretanje jedne izvršne komponente. Složena aplikacija sastoji se od više izvršnih komponenti koje se ne moraju nalaziti na istom računalu. U tom slučaju je izvršne komponente potrebno aktivirati određenim redoslijedom.

Deaktivacija je inverzna akcija od aktivacije i obuhvaća zaustavljanje svih izvršnih komponenti instalirane aplikacije. Ova akcija često se izvodi prije ostalih akcija postavljanja programske potpore (npr. aktualizacije).

Aktualizacija je akcija postavljanja nove inačice instalirane aplikacije i može se promatrati kao specijalan slučaj instalacije. Postavljanje nove inačice instalirane aplikacije podrazumijeva zamjenu podskupa datoteka novim datotekama. Ova akcija je jednostavnija od akcije instalacije jer je većina važnih informacija za uspješno izvođenje aplikacije već utvrđena za vrijeme instalacije aplikacije. Prije izvođenja aktualizacije se aplikacija obično zaustavi, a nakon aktualizacije se ponovno aktivira. Akcija aktualizacija uključuje procedure prijenosa i konfiguriranja.

Prilagodba ostvaruje, kao i akcija aktualizacije, promjene na instaliranoj aplikaciji. Za razliku od aktualizacije, koja se aktivira na udaljene događaje (npr. obavijest proizvođača o novoj inačici aplikacije), prilagodba se aktivira na lokalne događaje kao što su promjene u okolini korisničkog računala. Akcija prilagodbe se pokreće s ciljem ispravljanja i održavanja ispravnog rada instalirane aplikacije.

Uklanjanje pokreće korisnik kada više nema potrebu za korištenjem instalirane aplikacije. Postupak uklanjanja uključuje brisanje datoteka aplikacije s diskovnog spremnika te eventualno brisanje informacija o aplikaciji iz operacijskog sustava.

Ukidanje je akcija proglašavanja aplikacije zastarjelom, a provodi ju proizvođač aplikacije. U ovoj akciji proizvođač je dužan obavijestiti sve poznate korisnike o povlačenju aplikacije s tržišta. Povlačenjem aplikacije s tržišta proizvođač ukida korisničku potporu.

3.2 Izazovi u postavljanju aplikacija

Okolina u kojoj se aplikacija izvodi podložna je promjenama pa je stoga potrebno osigurati mogućnost prilagodbe aplikacije na nastale promjene. Na primjer, sklopovski moduli računala, poput procesora ili mrežne kartice, mogu se zamijeniti novim modelima s ciljem postizanja boljih radnih svojstava računala. Operacijski sustav računala mora biti u stanju prilagoditi se na nastale promjene u sklopovskoj opremi. Prilagodba operacijskog sustava se svodi na instalaciju pogonskih programa (engl. drivers) koji povezuju sklopovski modul i operacijski sustav.

U procesu postavljanja aplikacije posebnu pažnju je potrebno posvetiti utvrđivanju i poštivanju veza između programskih komponentni. Aplikacije se obično izgrađuju modularno pa je moguće da više različitih aplikacija koristi isti modul. Ovakav pristup pojednostavnjuje proces razvoja aplikacije, te ujedno smanjuje veličinu postavljene aplikacije. Na primjer, aplikacija za obradu teksta i aplikacija za obradu prezentacija imaju zajedničkih elemenata, poput funkcionalnosti za obradu tablica. Funkcionalnost obrade tablica može se ostvariti kao zaseban modul. Ako se na računalo instaliraju obje aplikacije, onda je dovoljno modul za obradu tablica instalirati samo jednom.

Procedura prijenosa instalacijskih paketa sastavni je dio akcija instalacije i aktualizacije. Prijenos instalacijskih paketa obično se ostvaruje putem mrežne infrastrukture. Ako se instalacijski paketi postave samo na jednom poslužitelju, onda se javlja opasnost zagrušenja poslužitelja korisničkim zahtjevima. Prikladniji način je uvišestručavanje instalacijskih paketa na više poslužitelja u mreži.

Javna mreža Internet postala je globalno elektroničko tržište. Mogućnosti javne mreže Internet mogu se iskoristiti u procesu postavljanja programske potpore. Proizvođači putem Interneta mogu oglašavati vlastite produkte. Korisnici putem Interneta mogu dohvaćati instalacijske pakete, obavještavati proizvođače o iskustvima prilikom korištenja aplikacije, prijaviti pogreške u radu aplikacije (engl. bug) ili dati komentare.

Uporaba javne mreže Internet istovremeno otvara i pitanje sigurnosti. U procesu postavljanja programske potpore putem globalne mreže Internet zahtijeva se rješavanje sljedećih sigurnosnih problema: privatnost, autentikacija i integritet. Privatnost zahtijeva da instalacijski paketi koji se prenose mrežom mogu biti čitljivi samo odredišnim entitetima. Zahtjev za privatnošću razrješava se enkripcijom podataka. Autentikacija podrazumijeva utvrđivanje i potvrđivanje identiteta mrežnog entiteta. Ovaj zahtjev osigurava da samo određeni entiteti mogu pristupati zaštićenim sredstvima. Integritet zahtijeva da instalacijski paketi tijekom prijenosa do

odredišta ne mogu biti promijenjeni, a da odredišni entitet to ne primjeti. Ovaj zahtjev se ostvaruje uporabom digitalnih potpisa.

3.3 Pristupi u postavljanju usluga

Današnji alati za postavljanje usluga, odnosno programske potpore općenito, pružaju različite razine automatiziranosti. Tipično se razine automatiziranosti dijele prema četiri pristupa: *postavljanje opisano skriptom*, *postavljanje opisano jezikom*, *postavljanje opisano modelom* i *postavljanje upravljano agentima*. U nastavku odjeljka detaljnije se opisuju svojstva i mogućnosti svakog od pristupa, a na kraju odjeljka se daje njihova usporedba.

3.3.1 Postavljanje usluga opisano skriptom

Ručno postavljanje programske potpore zahtijeva od korisnika potpuni angažman nad procesom postavljanja. Povećanjem složenosti aplikacije i broja računala osjetno se povećava vrijeme koje korisnik mora uložiti u postavljanje aplikacije, a sam postupak postaje podložan pogreškama.

Prvi korak u automatizaciji procesa postavljanja programske potpore je skriptiranje. Skriptiranje je povezivanje gotovih programskih procedura i vanjskih programa tako da zajedno ostvare neki složeni zadatak. U skriptu se upisuju naredbe koje se inače upisuju u upravljačku konzolu (engl. command console) operacijskog sustava. Skriptu izvodi interpretator operacijskog sustava. Moderni operacijski sustavi podržavaju barem jedan oblik skriptnog jezika. Skriptni jezici obično sadrže konstrukte za kontrolu tijeka izvođenja naredbi, konstrukte za kontrolu pogrešaka te konstrukte za obradu ulaznih parametara. Skripta se izvodi u neinteraktivnom načinu rada (engl. batch processing), odnosno za vrijeme izvođenja skripte nema komunikacije s korisnikom. Radni parametri skripte se navode prilikom njenog poziva.

Slika 3-2 daje primjer skripte Windows operacijskog sustava koja automatizira postavljanje jednostavne aplikacije. Na početku skripte se definiraju vrijednosti varijabli *passwd* i *user*. U nastavku se stvara datoteka *script.ftp* sa slijedom naredbi za dohvat datoteke *pie.zip* sa udaljenog računala. Nakon toga se poziva program *ftp* koji se spaja na udaljeno računalo *pie.fer.hr* i izvodi naredbe navedene unutar datoteke *script.ftp*. Naredbom *del* obriše se stvorena datoteka *script.ftp*, dok se naredbom *md* stvara staza *pie* u koju se pomoću *winrar* alata sažimanja smješta sadržaj datoteke *pie.zip*. Na kraju se naredbom *del* obriše datoteka *pie.zip*.

```

@ECHO OFF
set passwd = UserPassword
set user = UserName

> script.ftp ECHO %user%
>>script.ftp ECHO %passwd%
>>script.ftp ECHO binary
>>script.ftp ECHO get pie.zip
>>script.ftp ECHO close
>>script.ftp ECHO bye

FTP -s:script.ftp pie.fer.hr
DEL script.ftp
MD pie
WINRAR x pie.zip .\pie
DEL pie.zip

```

Slika 3-2: Primjer Windows skripte

Nixes alat [37] je primjer postavljanja programske potpore skriptom. *Nixes* alat je skup skripti za instalaciju, održavanje, upravljanje i nadgledanje aplikacija na *PlanetLab* mreži [36]. *Nixes* alat osim skupa skripti uključuje konfiguracijsku datoteku i javno Web sabiralište. U konfiguracijskoj datoteci se nalaze parametri koje *Nixes* skripte koriste za izvođenje. Javno Web sabiralište je skladište izvršnih datoteka aplikacije. *Nixes* alat je ostvaren za Linux operacijski sustav, skripte su pisane u *bash* (Bourne again shell) [40] skriptnom jeziku, a postavljanje instalacijskih paketa je automatizirano uporabom RPM (Red Hat Package Manager) [41] tehnologije. RPM je sustav za upravljanje instalacijskim paketima te je primarno razvijen za Linux operacijski sustav. RPM sustav pruža potporu instalaciji, nadogradnji, uklanjanju i verifikaciji programskih komponenti. *Nixes* alat nije prikladan za velike i složene sustave jer ne pruža automatizirane mehanizme za postavljanje aplikacije čiji su dijelovi raspodijeljeni po mreži i zahtijevaju posebno konfiguriranje.

3.3.2 Postavljanje usluga opisano jezikom

Napredak u postavljanju usluga je ostvaren razvojem posebnih jezika koji opisuju postupak postavljanja. Za razliku od skripti, koje su proceduralnog tipa, jezici za postavljanje usluga su deklarativnog tipa. Skripte zbog proceduralnosti odgovaraju na pitanje "*Kako postaviti uslugu?*" pa stoga sadrže naredbe poput "*stvori direktorij XYZ*" ili "*kopiraj datoteku ABC na lokaciju XYZ*". Jezici za opis postavljanja usluga su zbog deklarativnosti više usmjereni na pitanje "*Što postaviti?*". Jezikom za opis postavljanja usluga se aplikacija opisuje kao skup usluga, definiraju se vrijednosti njihovih radnih parametara te se definiraju njihove međusobne veze. Zbog općenitosti, jezici za opisivanje postupka postavljanja usluga podržavaju i mehanizme za definiranje proceduralnih akcija. Proces postavljanja opisan jezikom se kao i skripta zapisuje u tekstualnu datoteku koju izvodi interpretator. Interpretator na osnovi opisnika

postavljanja ima dovoljno informacija za uspješno postavljanje aplikacijskih usluga na jedno ili više računala.

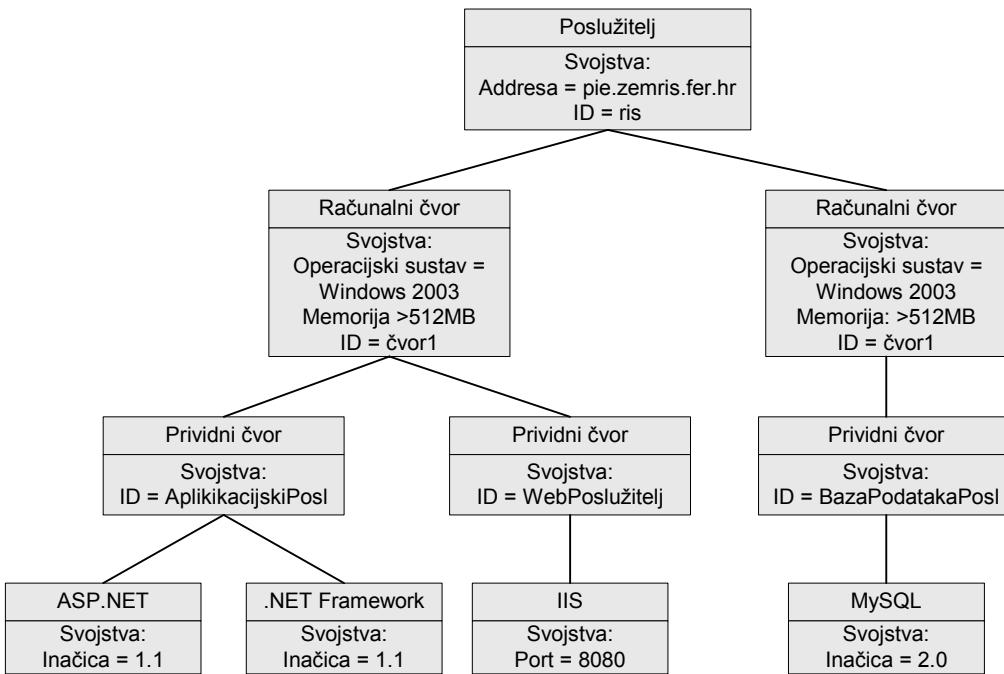
```
webServer extends{
    sfProcessHost "ris.zemris.fer.hr";
    port 8080;
}
```

Slika 3-3: Primjer SmartFrog opisnika aplikacijske usluge

Slika 3-3 prikazuje primjer opisnika postavljanja aplikacijske usluge u *SmartFrog* jeziku. SmartFrog je deklarativan jezik za opisivanje postavljanja usluga, a detaljnije se opisuje u poglavljju 3.4.2. Prikazani opisnik definira osnovne parametre za postavljanje usluge Web poslužitelja. Element *sfProcessHost* definira adresu računala na koje se postavlja usluga Web poslužitelja. Element *port* definira vrijednost specifičnog radnog parametra usluge koji se podešava tijekom postavljanja usluge. U ovom slučaju vrijednost elementa *port* označava da postavljena usluga Web poslužitelja očekuje korisničke zahtjeve na pristupu (engl. port) 8080.

3.3.3 Postavljanje usluga opisano modelom

Postavljanje usluga opisano modelom pruža najvišu razinu apstrakcije procesa postavljanja usluga. Kod ovog pristupa korisnik zadaje model raspodijeljene aplikacije u kojem definira razmještaj aplikacijskih usluga, njihova svojstva te njihovu međusobnu povezanost. Postavljanje opisano modelom zahtijeva pripadni radni okvir koji automatizira što je moguće više zadataka. Jedna od funkcija radnog okvira je održavanje jednoznačnog stanja raspodijeljene aplikacije u odnosu na zadani model. To konkretno znači da radni okvir nadgleda komponente raspodijeljene aplikacije te ukoliko su pojedine datoteke iz nekog razloga obrisane ili izmijenjene, onda ih radni okvir automatski zamijeni originalnim datotekama. Radni okvir također bi trebao pružiti potporu automatiziranoj instalaciji novih inačica i zakrpa komponenti raspodijeljene aplikacije.



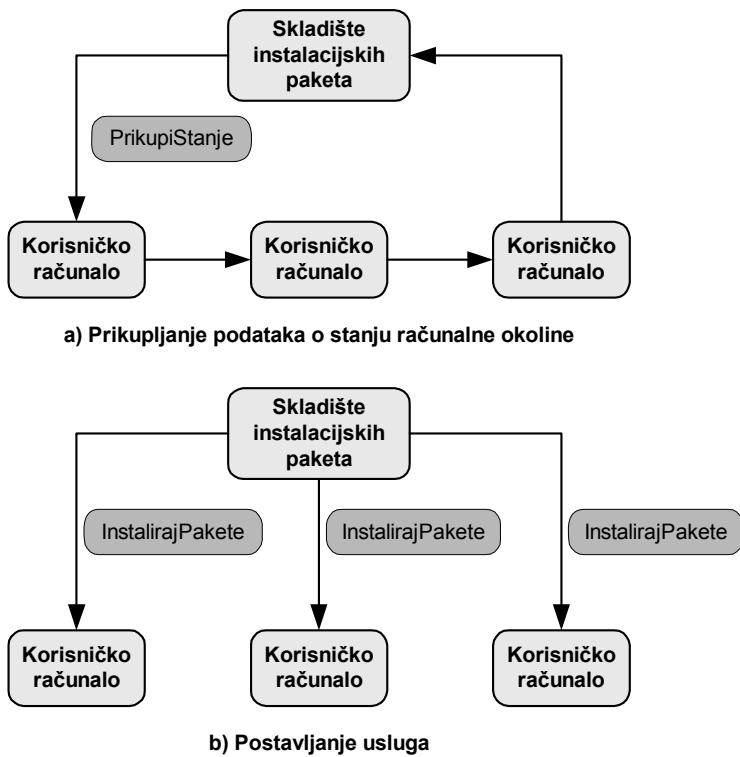
Slika 3-4: Primjer aplikacije opisane modelom

Primjeri sustava koji podržavaju postavljanje programske potpore modelom su *Radia* i *Microsoft Systems Management Server 2000*, a koji se detaljnije opisuju u poglavljima 3.4.3 i 3.4.4. Slika 3-4 prikazuje grafički način zadavanja modela raspodijeljene aplikacije. Na slici je prikazan model Web aplikacije dostupne na adresi *pie.zemris.fer.hr* koja je postavljena na dva računala. Na jednom računalu je postavljena aplikacijska logika i Web poslužitelj, a na drugom računalu je postavljena baza podataka.

3.3.4 Postavljanje usluga upravljano agentima

Pokretni agent se definira kao program koji se za vrijeme izvođenja autonomno kreće po mrežnim računalima. Prilikom kretanja agenta premješta se cijeli programski proces, odnosno premješta se izvršni kôd agenta, trenutno stanje procesa te dodatni podaci potrebni izvođenju procesa. Svojstvo autonomnosti označava da agent samostalno donosi odluku o selidbi na drugi mrežni čvor.

Primjena pokretnih agenata u postavljanju usluga zasniva se na činjenici da je neučinkovito i neprikladno izgraditi generički sustav za postavljanje usluga, jer proces postavljanja usluga ovisi o računalnim platformama i specifičnostima samih usluga. Prikladnije je za svaku uslugu razviti specifičnu računalnu logiku koja postavlja uslugu na mrežna računala. Tehnologije pokretnih agenata su jedan od prikladnih načina za ostvarenje navedene računalne logike.



Slika 3-5: Arhitektura sustava za postavljanje usluga zasnovanog na agentima

U [47] se opisuje primjer sustava za postavljanje programske potpore zasnovan na pokretnim agentima. Sustav je razvijen za TACOMA (Tromsø And CORnell Mobile Agents) platformu pokretnih agenata [48]. Slika 3-5 prikazuje arhitekturu sustava. *Skladište instalacijskih paketa* je središnji dio sustava i u njemu se nalaze instalacijski paketi usluga. Instalacijski paketi sastoje se od senzorskog i instalacijskog agenta. Senzorski agent provjerava da li je pripadni instalacijski paket postavljen na računalu, dok instalacijski agent instalira pripadni paket na računalo.

Skladište instalacijskih paketa u pravilnim vremenskim razmacima prikuplja podatke o stanju računalne okoline (slika 3-5a), odnosno za svaki čvor računalne okoline prikuplja popis postavljenih usluga. Prikupljanje podataka o stanju računalne okoline izvodi agent *PrikupiStanje* kojeg stvara *Skladište instalacijskih usluga*. Agent *PrikupiStanje* sadrži listu računala koje treba posjetiti te listu senzorskih agenata instalacijskih paketa. Kada agent *PrikupiStanje* stigne na korisničko računalo, aktivira senzorske agente koji provjere postojanje pripadnih usluga na korisničkom računalu. Agent *PrikupiStanje* sakupi informacije od senzorskih agenata i preseli se na sljedeće računalo. Nakon što obide sva računala, agent se vraća u *Skladište instalacijskih paketa*.

Skladište instalacijskih paketa na osnovi prikupljenih podataka o stanju računalne okoline stvara za svaki čvor računalne okoline agent *InstalirajPakete*. Agent *InstalirajPakete* sastoji se od skupa instalacijskih agenata. Svaki instalacijski agent je zadužen za instalaciju pripadne usluge. Agensi *InstalirajPakete* usporedno se šalju na čvorove računalne okoline. Kada agent *InstalirajPakete* stigne na korisničko računalo, aktivira instalacijske agente koji izvode instalaciju pripadnih usluga. Nakon instalacije usluga, agent *InstalirajPakete* završava svoj životni ciklus.

Drugi primjer postavljanja usluga upravljan agentima je *Sustav za udaljeno održavanje programske potpore* (engl. Remote Maintenance Shell, RMS) [49] razvijen u suradnji Fakulteta elektrotehnike i računarstva u Zagrebu te tvrtke Ericsson Nikola Tesla. Sustav ostvaruje postavljanje, testiranje, te održavanje više inačica iste aplikacije na računalnom čvoru. Svaku od osnovnih operacija izvodi specifičan pokretni agent.

3.3.5 Usporedba pristupa postavljanja usluga

Svaki od opisanih pristupa problemu postavljanja usluga ima svoje prednosti i nedostatke. Za uspješnu usporedbu različitih pristupa postavljanja potrebna je metrika prikladnosti pojedinog pristupa. Primjer kvalitativne i kvantitativne metrike za uspoređivanje pristupa postavljanja definiran je u radu [27].

Kvantitativno se pojedini pristupi mogu uspoređivati na osnovi broja linija konfiguracijskog kôda kojim se opisuje postupak postavljanja usluge. Broj linija konfiguracijskog kôda neizravan je pokazatelj složenosti postavljanja usluge. Odnos broja linija i složenosti postavljanja usluge je razmjeran, odnosno veći broj linija konfiguracijskog kôda označava složeniji postupak postavljanja. Kvantitativnu usporedbu moguće je provesti i na osnovi broja koraka koje administrator treba poduzeti u postupku postavljanja usluge. Broj koraka u postavljanju usluge razmjeran je vremenu potrebnom za postavljanje usluge, odnosno manji broj koraka označava kraće vrijeme postavljanja usluge.

Eksperimenti omogućuju utvrđivanje odnosa između pojedinih pristupa postavljanja usluga s obzirom na opisane kvantitativne metrike. Logično je očekivati da povećanjem složenosti raspodijeljene aplikacije raste i broj linija konfiguracijskog kôda. Ispitivanja su pokazala da je taj porast izraženiji u slučaju postavljanja usluga skriptom, nego prilikom postavljanja usluga jezikom. Također je logično za očekivati da porastom složenosti raspodijeljene aplikacije raste i broj koraka u postupku postavljanja usluge. Ispitivanja su i za ovaj slučaj pokazala da je porast izraženiji u slučaju postavljanja opisanog skriptom, nego prilikom postavljanja opisanog jezikom ili zasnovanog na agentima. Ispitivanja su također

pokazala da je broj koraka u slučaju postavljanja usluga na osnovi modela neovisan o složenosti aplikacije. Administrator uvijek poduzima sličan broj koraka.

Kvalitativno svojstvo usporedbe pristupa za postavljanje usluge je stupanj automatizacije procesa postavljanja. Sustavi za postavljanje zasnovani na jeziku omogućuju automatiziranu aktivaciju raspodijeljene aplikacije. Prilikom aktivacije potrebno je usluge pokrenuti točno određenim redoslijedom, jer se pojedine usluge za svoje izvođenje zahtijevaju funkcionalnosti drugih usluga. Najveći stupanj automatizacije pružaju sustavi zasnovani na modelu.

Iznimno važno svojstvo je prilagodba na pogreške u postupku postavljanja usluga. Skriptiranje pruža rudimentarnu potporu upravljanju greškama. Postavljanje opisano jezikom naprednije upravlja pogreškama u procesu postavljanja. Sustavi za postavljanje zasnovani na modelu prilikom pojave pogreške mogu automatizirano promijeniti konfiguraciju raspodijeljene aplikacije pa umjesto na zadano računalo, uslugu mogu postaviti na neko drugo računalo.

Važno svojstvo sustava za postavljanje usluga je vrijeme potrebno administratoru da nauči upravljati sustavom. Za postavljanje usluga opisano skriptom administrator treba uložiti malo vremena za učenje postupka. U slučaju postupka postavljanja usluga zasnovanog na jeziku administrator treba uložiti nešto više vremena kako bi se upoznao sa specifičnim konstruktima jezika. Postavljanje usluga zasnovano na agentima zahtijeva slično vrijeme za učenje kao i u slučaju postavljanja zasnovanog na jeziku, dok postupak postavljanja usluga zasnovan na modelu zahtijeva od administratora najviše vremena za učenje.

Ni jedan od opisanih pristupa nije optimalno rješenje, odnosno svaki od pristupa ima svoje područje primjene. U slučaju malog broja računala i jednostavne raspodijeljene aplikacije najprikladije je usluge postaviti ručnim pristupom ili uporabom skripti. Za takva jednostavna postavljanja administrator ne mora upoznavati nikakve nove tehnologije. U slučaju veće računalne mreže i složene raspodijeljene aplikacije prikladna je uporaba pristupa zasnovanog na jeziku ili pristupa zasnovanog na modelu, jer pružaju najveći stupanj automatizacije složenog postupka postavljanja usluga.

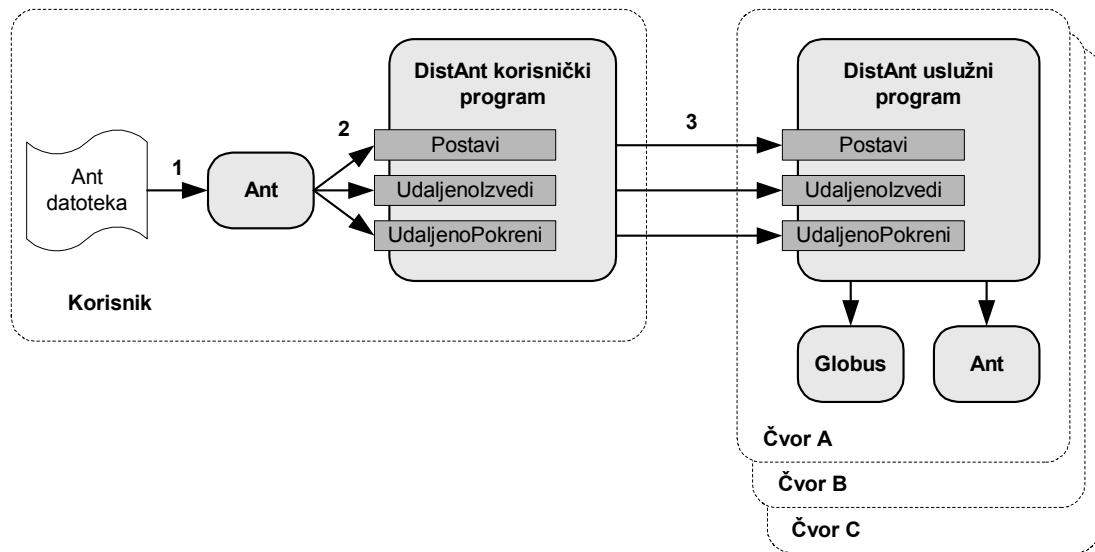
3.4 Tehnologije postavljanja usluga

Razvijeno je više alata i tehnologija za postavljanje usluga. U nastavku se opisuje samo nekoliko popularnih alata. *Distributed Ant* je primjer alata za postavljanje usluga koji koristi napredni skriptni pristup. *SmartFrog* radni okvir ostvaruje postavljanje usluga na osnovi opisnog jezika. *Radia* i *Microsoft Systems Management Server 2003* primjeri su naprednih sustava koji ostvaruju postavljanje usluga na osnovi modela.

3.4.1 Distributed Ant

Distributed Ant (DistAnt) [35] je sustav za postavljanje raspodijeljenih aplikacija posebno prilagođen Grid aplikacijama. Sustav omogućuje postavljanje aplikacija korisnicima koji nemaju administratorske ovlasti na udaljenom računalu, što je uobičajena situacija u Grid okolini. Autori sustava su primjetili sličnost između postupka gradnje (prevodenje izvornog kôda, povezivanje i pakiranje) i postupka postavljanja (prijenos datoteka, konfiguriranje i pokretanje) složene raspodijeljene aplikacije, zbog čega su za postupak postavljanja aplikacije odlučili iskoristiti postojeći sustav za gradnju aplikacije. DistAnt koristi Globus Toolkit 3 (GT3) i Ant [34] tehnologije. Ant je XML-zasnovan alat za automatiziranu gradnju (engl. build-management tool) aplikacija. Osnovna prednost Ant alata je platformska neovisnost, jer je ostvaren u Java programskom jeziku i koristi XML jezik za zapis ulaznih podataka.

Ant ulazna datoteka sastoji se od zadataka, ciljeva i svojstava. Zadatak definira akcije, poput kopiranja datoteka, prevodenja izvornog kôda ili pokretanja programa. Cilj objedinjuje više zadataka u jednu logičku jedinicu. Primjer cilja može biti postupak prevodenja ili postupak povezivanja izvršnih dijelova aplikacije. Svojstva su korisnički definirane varijable. Primjer svojstva je lokacija direktorija na diskovnom spremniku ili mrežno ime računala.



Slika 3-6: Arhitektura DistAnt sustava za postavljanje usluga

Slika 3-6 prikazuje osnovnu arhitekturu DistAnt sustava. Sustav se sastoji od tri glavne komponente: korisnički Ant zadaci, DistAnt korisnički programa i DistAnt uslužni program. Ant alat omogućuje definiranje korisničkih zadataka. DistAnt koristi navedeno svojstvo i uvodi vlastite zadatke *postavi*, *udaljenoIzvedi* i *udaljenoPokreni*. Zadatak *postavi* definira mehanizam prijenosa datoteka na udaljenu lokaciju. Zadatak *udaljenoIzvedi* definira proceduru koja će se

izvesti na udaljenom računalu, pri čemu je procedura jedan od standarnih Ant zadataka. Zadatak *udaljenoPokreni* definira mehanizam za pokretanje postavljene aplikacije na udaljenom računalu. Korisnički DistAnt zadaci navode se unutar ulazne Ant datoteke. DistAnt korisnički program je lokalni program koji koordinira procesom postavljanja raspodijeljene aplikacije na udaljena računala. DistAnt uslužni program se nalazi na udaljenim računalima i izvodi postavljanje aplikacije na udaljeno računalo.

Postupak postavljanja aplikacije na udaljena računala započinje obradom ulazne Ant datoteke (1). Nailaskom na korisničke DistAnt zadatke u ulaznoj datoteci, Ant alat poziva odgovarajuće metode DistAnt korisničkog programa (2). DistAnt korisnički program prosljeđuje korisničke DistAnt zadatke udaljenom DistAnt uslužnom programu koji ih izvodi (3). Za izvođenje korisničkih DistAnt zadataka, poput prijenosa datoteka i pokretanja postavljenih usluga, DistAnt uslužni program koristi mehanizme Globus Toolkit i Ant alata.

DistAnt sustav je specijaliziran za Grid okolinu, prvenstveno izgrađenu na Globus Toolkit 3 platformi. Glavni nedostatak mu je što ne pruža direktnu potporu konfiguriranju i povezivanju komponenti aplikacije. Korisnik je primoran konfiguriranje aplikacije ostvariti neizravno putem Ant zadataka, što je neprikladno za krajnjeg korisnika.

3.4.2 SmartFrog

SmartFrog (Smart Framework for Object Groups) [28] je radni okvir koji pruža potporu konfiguriranju, automatskom postavljanju, pokretanju i zaustavljanju raspodijeljenih aplikacija. Radni okvir osigurava da su aplikacijske komponente postavljene na prikladna računala, da su ispravno konfiguirirane te da su međusobno ispravno povezane u cjelokupni sustav.

SmartFrog radni okvir sastoji se od sljedeća tri glavna dijela: *opisni jezik*, *komponentni model* i *infrastruktura za postavljanje*. *Opisni jezik* omogućava definiranje deklarativnog opisa raspodijeljene aplikacije. Deklarativni opis raspodijeljene aplikacije sadrži popis aplikacijskih komponenti, njihove konfiguracijske parametre te pravilan redoslijed njihovog pokretanja. Komponente raspodijeljene aplikacije obično zavise jedna o drugoj pa se zato najprije pokreću nezavisne komponente, a nakon njih se pokreću određenim redoslijedom preostale komponente.

Slika 3-7 daje jednostavan primjer SmartFrog opisnika raspodijeljene aplikacije koja se sastoji od dva Web poslužitelja i baze podataka. SmartFrog jezik se zasniva na predlošcima u kojima se definiraju podrazumijevane vrijednosti aplikacijskih komponenti. U detaljnoj definiciji raspodijeljene aplikacije predlošci se specijaliziraju ovisno o kontekstu. Primjer sa slike definira dva predloška: predložak Web poslužitelja (*webservertemplate.sf*) i predložak baze podataka (*dbtemplate.sf*). Unutar definicije cjelokupne aplikacije (*system.sf*) predlošci Web poslužitelja i

baze podataka se specijaliziraju tako da se podrazumijevane vrijednosti svojstava iz predloška zamijene specifičnim vrijednostima. Ako nije zadana specifična vrijednost svojstva, onda se koristi podrazumijevana vrijednost. U primjeru se tako definiraju dva Web poslužitelja koji se umjesto na lokalno računalo postavljaju na računala sa adresama 192.168.0.15 i 192.168.0.16.

```
//webservertemplate.sf
webServerTemplate extends{
    sfProcessHost "localhost";
    port 80;
    useDB;
}

//dbtemplate.sf
dbTemplate extends{
    userTable extends{
        columns 4;
        rows 3;
    }
    dataTable extends {
        columns 2;
        rows 5;
    }
}

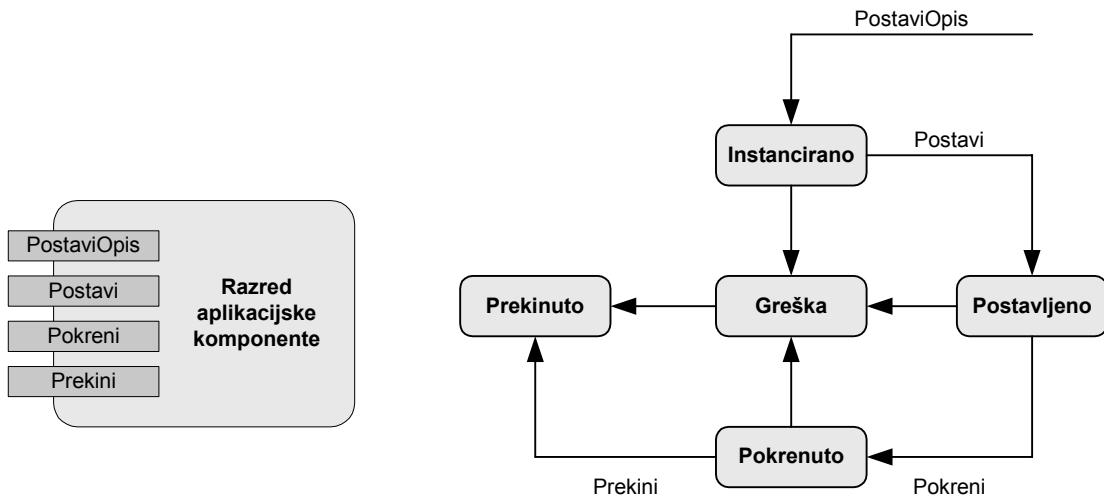
//system.sf
system extends{
    ws1 extends webServerTemplate{
        sfProcessHost "192.168.0.15";
    }
    ws2 extends webServerTemplate{
        sfProcessHost "192.168.0.16";
        port 8080;
        type "backup";
    }
    db extends dbTemplate{
        userTable:rows 6;
    }
}
```

Slika 3-7: Primjer SmartFrog opisnika raspodijeljene aplikacije

Komponentni model SmartFrog razvojnog okvira definira pojedinosti ostvarenja aplikacijskih komponenti. Od aplikacijskih komponenti se zahtjeva da su ostvarene u Java programskom jeziku i da ostvaruju sljedeće metode (slika 3-8): *postaviOpis*, *postavi*, *pokreni* i *zaustavi*. Značenje metoda sučelja komponenti opisano je u [30]. Zajednički podskup metoda sučelja omogućuje SmartFrog radnom okviru unificiranu kontrolu aplikacijskih komponenti. SmartFrog radni okvir predaje komponentama njihove opisnike u SmartFrog jeziku, a od njih se očekuje da samostalno ostvare specifičnosti vlastitog postavljanja i konfiguriranja. Unatoč navedenim zahtjevima, putem SmartFrog okvira je moguće upravljati i tzv. *ne-SmartFrog* komponentama, odnosno komponentama koje nisu ostvarene u Java programskom jeziku i ne ostvaruju zadane metode sučelja. Primjer ne-SmartFrog komponente je Apache Web poslužitelj. U takvom slučaju je potrebno razviti valjanu SmartFrog komponentu koja će biti posrednik (engl. wrapper) između SmartFrog razvojnog okvira i ne-SmartFrog komponente.

Važan aspekt SmartFrog komponentnog modela je životni ciklus komponenti koji se definira konačnim automatom (slika 3-8). Prijelazi automata odgovaraju pozivima metoda sučelja komponente, a stanja automata odgovaraju stanjima komponente. Životni ciklus aplikacije koja se sastoji od više komponenti je kombinacija životnih ciklusa svih njenih dijelova. SmartFrog radni okvir osigurava da se aplikacija ne može pokrenuti prije nego se

postave sve njezine sastavne komponente. Prilikom zaustavljanja aplikacije nije bitan redoslijed zaustavljanja pojedinih komponentni.



Slika 3-8: Komponentni model SmartFrog radnog okvira

Infrastruktura za postavljanje aplikacije je treći dio SmartFrog radnog okvira. Infrastrukturu čine SmartFrog procesi (engl. daemons) pokrenuti na svim računalima mreže. Infrastruktura osigurava da komponente raspodijeljene aplikacije u skladu s opisnikom aplikacije budu smještene na odgovarajuće lokacije te da imaju pravilno podešene konfiguracijske postavke. SmartFrog procesi komuniciraju s komponentama u svrhu kontrole njihovog životnog ciklusa. Infrastruktura je potpuno raspodijeljena, bez središnje kontrolne točke, što osigurava razmjeran rast SmartFrog radnog okvira.

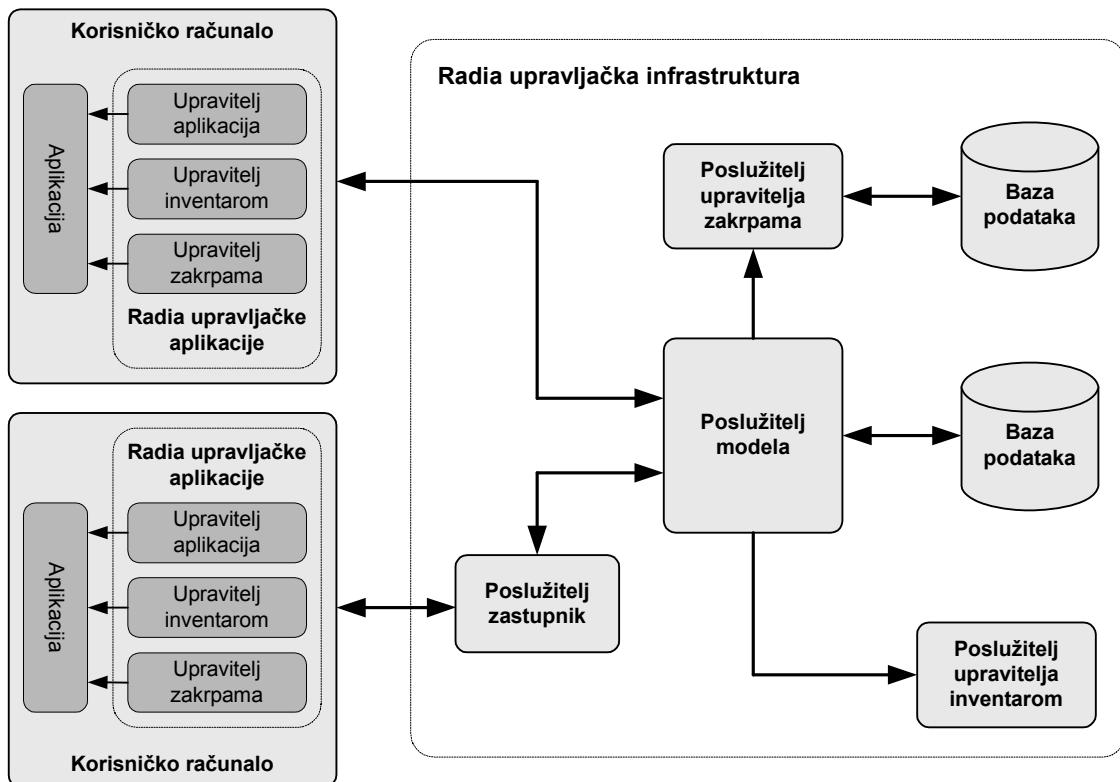
SmartFrog je razvijen u istraživačkim laboratorijima tvrtke Hewlett Packard. Usmjerenost razvoja SmartFrog sustava prema radnom okviru, a ne prema aplikaciji, osigurao mu je iznimnu prilagodljivost koja dolazi do izražaja u procesu postavljanja i konfiguriranja složenih raspodijeljenih aplikacija. Prilagodljivost SmartFrog radnog okvira pokušava se iskoristiti u postavljanju složenih aplikacija na Grid računalne okoline [45].

3.4.3 Radia

Radia [33] sustav je produkt tvrtke Novadigm koja je postala dijelom Hewlett Packard korporacije. Radia je komercijalni sustav za automatizirano postavljanje programske potpore na mrežna računala. Sustav je primjer postavljanja programske potpore na osnovi modela.

Radia sustav se sastoji od dvadesetak komponenti. Administrator ovisno o potrebama i topologiji mreže odabire komponente Radia sustava i razmješta ih po lokalnoj mreži. Slika 3-9 je primjer konfiguracije Radia sustava. Komponente Radia sustava se dijele u dvije kategorije:

Radia upravljačke aplikacije i Radia upravljačka infrastruktura. Radia upravljačke aplikacije smještene su na korisničkim računalima i pružaju potporu postavljanju aplikacijskih paketa na korisnička računala. Radia upravljačku infrastrukturu čine komponente koje pružaju potporu održavanju zadanog modela raspodijeljene aplikacije. Komponente upravljačke infrastrukture mogu biti u bilo kojoj kombinaciji raspodijeljene po mreži.



Slika 3-9: Primjer konfiguracije Radia sustava

Aplikaciju je potrebno pripremiti u odgovarajući oblik kako bi se mogla postaviti na mrežna računala putem Radia sustava. Uporabom Radia alata aplikacija se oblikuje u instalacijske pakete koji se spremaju u bazu podataka *Poslužitelja modela*. *Poslužitelj modela* je središnji dio Radia infrastrukture. Osim instalacijskih paketa, u bazu podataka *Poslužitelja modela* spremi se i model raspodijeljene aplikacije. Model definira željeni razmještaj aplikacijskih komponenti na fizička računala.

Postupkom sinkronizacije se uspostavlja jednakost između zadanog modela i fizičkog stanja na korisničkim računalima. Postupak sinkronizacije pokreće *Upravitelj aplikacija* ili *Poslužitelj modela*. *Upravitelj aplikacija* je smješten na korisničkom računalu, a uloga mu je održavanje aplikacijskih komponenti na korisničkom računalu. U postupku sinkronizacije *Upravitelj aplikacija* najprije zatraži od *Poslužitelja modela* očekivano stanje aplikacijskih komponenti. Nakon što primi odgovor, *Upravitelj aplikacija* uspoređuje fizičko i očekivano

stanje aplikacijskih komponenti, na osnovi kojih određuje popis datoteka koje treba nadodati ili ukloniti s računala.

Upravitelj aplikacija dohvaća nove datoteke iz *Poslužitelja modela*. S ciljem rasterećenja *Poslužitelja modela* datoteke moguće je dohvaćati i putem *Poslužitelja zastupnika*. *Poslužitelj zastupnik* je privremeno spremište datoteka aplikacije. Ako zastupnik ne posjeduje traženu datoteku, onda ju dohvati iz *Poslužitelja modela* i lokalno spremi. *Poslužitelj zastupnik* u slučaju ispunjenog lokalnog spremnika briše stare datoteke.

Nakon što se aplikacija postavi na korisnička računala, postoji potreba za naknadnim postavljanjem aplikacijskih zakrpa (engl. patch). Radia sustav pruža potporu postavljanju zakrpa putem *Upravitelja zakrpama* i *Poslužitelja zakrpama*. *Upravitelj zakrpama* je smješten na korisničkom računalu i uloga mu je postavljenja aplikacijskih zakrpa. Aplikacijske zakrpe se spremaju u bazu podataka *Poslužitelja zakrpama*. *Poslužitelj zakrpama* obavještava *Upravitelje zakrpama* o novim zakrpama, na što oni reagiraju postavljanjem potrebnih zakrpa.

Radia sustav također pruža potporu stvaranju izvještaja o stanju postavljenih aplikacija i sklopovske opreme (engl. hardware). Izvještaji su u obliku web stranica, a stvara ih *Poslužitelj upravitelja inventarom*. Podaci o stanju korisničkih računala se prikupljaju od *Upravitelja inventarom* koji su smješteni na korisničkim računalima. *Upravitelji inventarom* prikupljaju informacije o stanju sklopovske opreme, poput količine slobodne radne memorije, svojstvima procesora ili slobodnom prostoru diskovnog spremnika.

Središnji dio infrastrukture je *Poslužitelj modela* koji je potencijalno usko grlo sustava, odnosno prepreka razmjernom rastu sustava. S ciljem ostvarenja razmjerog rasta sustava, Radia nudi posebne komponente koje omogućuju sinkronizaciju sadržaja između više *Poslužitelja modela*. Radia sustav, osim mogućnosti postavljanja aplikacijskih komponenti, također pruža potporu postavljanju operacijskih sustava na mrežna računala.

3.4.4 Microsoft Systems Management Server 2003

Systems Management Server 2003 (SMS) [31] je komercijalni aplikacijski paket Microsoft korporacije. Sustav automatizira postavljanje i konfiguriranje aplikacija na mrežnim računalima s Windows operacijskim sustavom. Osim postavljanju i konfiguriranju aplikacija, SMS pruža potporu prikupljanju informacija o stanju mrežnih računala i postavljenih aplikacija, udaljenoj kontroli računala te stvaranju izvještaja.

Podaci o stanju mrežnih računala i postavljenih aplikacija koriste se u svrhu planiranja nadogradnje sustava ili praćenja promjena na mrežnim računalima. Na primjer, prije postavljanja nove aplikacije, administrator sustava može saznati koja računala imaju dovoljno procesorske

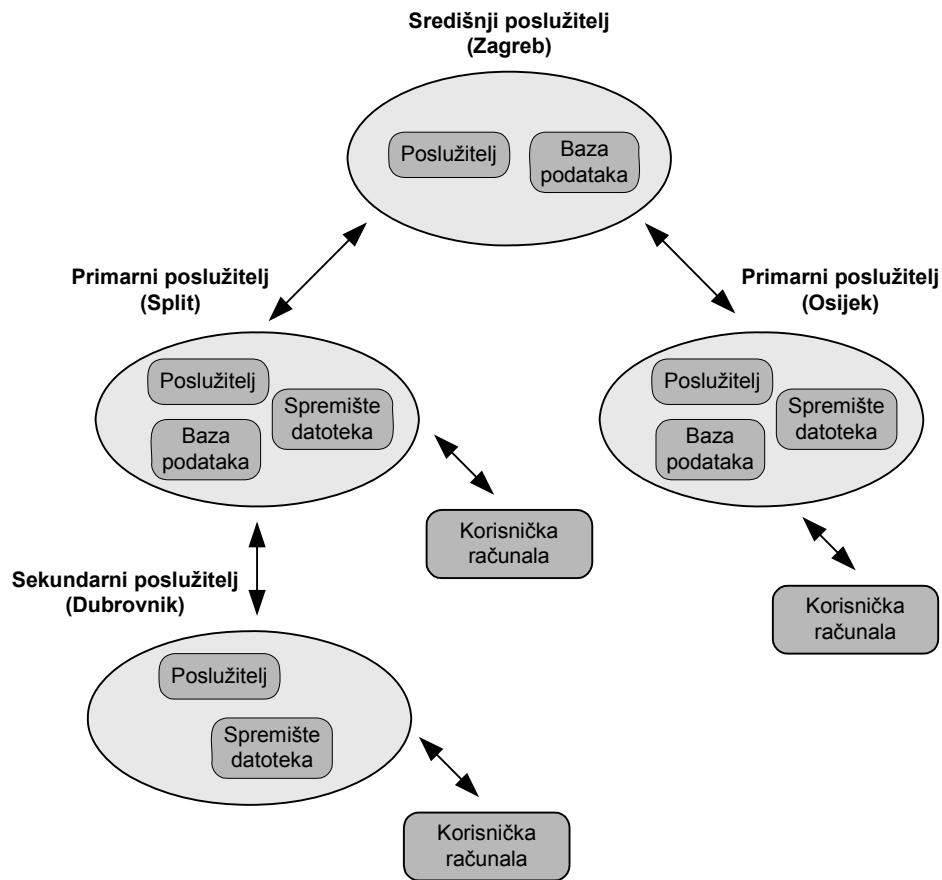
snage i radne memorije za novu aplikaciju. Podatke o stanju mrežnih računala SMS sustav prikuplja putem Windows Management Instrumentation tehnologije koja je standardno uključena u Windows operacijski sustav. SMS sustav također pruža bogatu potporu stvaranju izvještaja koji se prezentiraju u obliku Web stranica. Administratoru je na raspolaganju više unaprijed oblikovanih izvještaja putem kojih ostvaruje detaljan uvid u stanje pojedinog mrežnog računala.

Postavljanje i konfiguriranje aplikacija na mrežnim računalima je središnja mogućnost SMS sustava. Početni korak u postavljanju aplikacije je stvaranje instalacijskih paketa koji sadrže datoteke aplikacije, početne konfiguracijske postavke te skriptirane akcije koje će se poduzeti tijekom instaliranja aplikacije. SMS pruža više alata i tehnologija za stvaranje instalacijskih paketa. Stvoreni paket se spremi u SMS sustav koji ga uviše stvara na više fizičkih lokacija čime se smanjuje opterećenje čvorova. Administrator sustava aktivira postavljanje aplikacija definiranjem skupa računala na koja će se aplikacija postaviti. SMS pruža više načina definiranja ciljnih računala. Na primjer, skup ciljnih računala mogu biti sva računala koja imaju preko 500 MB radne memorije i 1GB slobodnog prostora na tvrdom disku. U ovom primjeru se skup ciljnih računala definira na osnovi podataka o stanju mrežnih računala.

SMS sustav podržava upravljanje aplikacijskim zakrpama. Sustav putem izvještaja pruža administratoru uvid u skup postavljenih zakrpa na pojedinim računalima. Na osnovi tih informacija administrator definira prioritetu listu zakrpa koje je potrebno postaviti na računala.

U svrhu jednostavnijeg ispravljanja problema na udaljenim računalima, SMS sustav pruža skup alata kojima administrator može ostvariti potpunu kontrolu nad udaljenim problematičnim računalom. Ti alati omogućavaju pokretanje naredbi na udaljenom računalu, tekstualnu komunikaciju s korisnikom udaljenog računala, prijenos datoteka na udaljeno računalo te nadgledanje mrežnog prometa na udaljenom računalu.

SMS sustav se zasniva na hijerarhijskoj arhitekturi koja omogućuje razmjeran rast sustava. Sustav se sastoji od jednog centralnog poslužitelja ili ga je moguće raspodijeliti na više računala ovisno o potrebama poslovne organizacije. Raspodjela SMS sustava pozitivno utječe na propusnost mreže, raspodjeljuje obradu podataka i ostvaruje zalihost podataka. Važan aspekt korištenja SMS sustava je pažljivo planiranje postavljanja čitavog sustava. Nepravilno postavljen sustav teže se proširuje, a radna svojstva čitave mreže su smanjena.



Slika 3-10: Hijerarhijska arhitektura SMS sustava

Slika 3-10 prikazuje primjer hijerarhijske arhitekture SMS sustava. Hijerarhijska arhitektura je prikladna za povezivanje geografski udaljenih organizacijskih jedinica. Osnovni elementi SMS sustava su poslužitelji (engl. site) na slici označeni elipsom. Poslužitelji se mogu nalaziti na jednom računalu ili se mogu po potrebi raspodijeliti na više računala. Dva su tipa poslužitelja: *primarni* i *sekundarni*. Primarni poslužitelji uključuju bazu podataka, a sekundarni su bez baze podataka. Primarni poslužitelj na vrhu hijerhije se naziva središnji poslužitelj. Sekundarni poslužitelji prosljeđuju podatke o podređenim korisničkim računalima svom nadređenom primarnom poslužitelju. Spremišta datoteka su skladišta instalacijskih paketa i ostalih datoteka. Ova skladišta su strateški raspoređena po mreži kako bi se smanjilo opterećenje poslužitelja.

4 Otkrivanje usluga

Raspodijeljeni sustav ostvaren u skladu s paradigmom zasnovanom na uslugama sastoji se od skupa usluga koje međusobnom komunikacijom ostvaruju složene zadatke. Svaka usluga raspodijeljenog sustava mora poznavati lokacije ostalih usluga s kojima komunicira. Lokacije se obično zapisuju unutar konfiguracijskih datoteka. Međutim, ovakav pristup ima nedostatke. Prvi nedostatak je čvrsta povezanost usluga. Čvrsta povezanost se očituje tijekom izvođenja složenog zadatka kada neka od usluga postane nedostupna pa se složeni zadatki ne može u potpunosti ostvariti. Sljedeći nedostatak je složeno održavanje raspodijeljenog sustava. Prilikom premještanja usluge s jednog računala na drugo potrebno je novu lokaciju navesti u konfiguracijskim datotekama svih usluga koje koriste premještenu uslugu.

Postupak otkrivanja usluga (engl. service discovery) rješava navedene probleme čvrste povezanosti i složenosti održavanja raspodijeljenog sustava. Postupkom otkrivanja, korisnici i usluge saznavaju informacije o drugim uslugama dostupnim u raspodijeljenom sustavu. Otkrivanje usluga omogućuje korisnicima pronalaženje prikladne zamjene za nedostupnu uslugu. Na primjer, umjesto nedostupne usluge moguće je iskoristiti neku drugu funkcionalnu uslugu. Informacije o uslugama raspodijeljenog sustava spremaju se u katalog usluga. Uloga kataloga je prikupljanje i održavanje podataka o dostupnim uslugama raspodijeljenog sustava. Katalog usluga ostvaruje se kao usluga, ali za razliku od ostalih usluga njegova lokacija i sučelje su općepoznati svim korisnicima raspodijeljenog sustava. Ako se neka usluga premjesti s jednog računala na drugo, njena nova lokacija zabilježi se u katalogu usluga. Novu lokaciju premještene usluge korisnici otkrivaju putem kataloga usluga.

U ovom poglavlju opisuju se rješenja i pristupi problemu otkrivanja usluga. Opisuju se uobičajene arhitekture kataloga usluga te metode odabira prikladne usluge iz skupa dostupnih usluga.

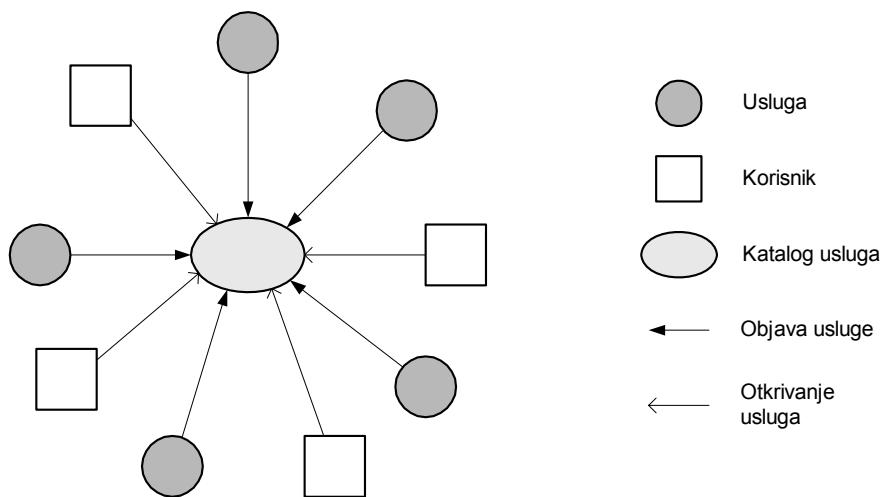
4.1 Arhitekture kataloga usluga

Osnovna funkcionalnost kataloga usluga je pružanje mehanizama za spremanje i pretraživanje informacija o raspodijeljenom sustavu. Osnovni zahtjev na katalog usluga je svojstvo razmernog rasta, odnosno radna svojstava kataloga ne smiju ovisiti o količini spremljenih podataka. Svojstvo razmernog rasta moguće je ostvariti pažljivom organizacijom arhitekture kataloga usluga. Postoje tri uobičajene arhitekture kataloga usluga: *centralizirana*, *hijerarhijska* i *arhitektura zasnovana na mrežama ravnopravnih sudionika*. Centralizirana

arhitektura je najjednostavnija, ali ne ostvaruje svojstvo razmijernog rasta. Hijerarhijska arhitektura i arhitektura zasnovana na mrežama ravnopravnih sudionika omogućuju ostvarenje kataloga usluga sa svojstvom razmijernog rasta.

4.1.1 Centralizirana arhitektura

Centralizirana arhitektura najjednostavniji je pristup izgradnje kataloga usluga. Slika 4-1 prikazuje centraliziranu arhitekturu kataloga usluga. U prikazanoj arhitekturi postoji jedan javno poznati mrežni čvor na kojem se nalazi katalog usluga s popisom prisutnih usluga u računalnoj okolini. Sve usluge računalne okoline objavljaju svoje informacije u navedeni jedinstveni katalog usluga. Korisnici otkrivaju informacije o uslugama također putem istog kataloga usluga.



Slika 4-1: Centralizirana arhitektura kataloga usluga

Centralizirani pristup izgradnje kataloga Web usluga ima nekoliko nedostataka. Prvi nedostatak su slaba radna svojstva kataloga usluga. Povećanjem broja usluga i korisnika povećava se opterećenje središnjeg kataloga usluga te on postaje usko grlo cijelokupnog sustava. Drugi nedostatak je da katalog postaje jedinstvena točka pogreške sustava (engl. single point of failure). Ako čvor na kojem se nalazi katalog usluga postane nedostupan, onda cijeli raspodijeljeni sustav postaje nedostupan, jer korisnici više ne mogu pronaći, a time niti koristiti željene usluge. Treći nedostatak je visoka cijena održavanja kataloga usluga. Centraliziran katalog usluga zahtijeva snažno sklopolje, s mnogo procesorske snage i memorijskog kapaciteta.

Jedno od rješenja navedenog problema centralizirane arhitekture kataloga usluga je uvišestručavanje kataloga usluga (engl. replication). Uvišestručavanjem kataloga usluga na nekoliko mrežnih čvorova raspodijeljuje se opterećenje pretraživanja kataloga te katalog ostaje dostupan u slučaju nedostupnosti pojedine preslike kataloga. Međutim, uvišestručavanje

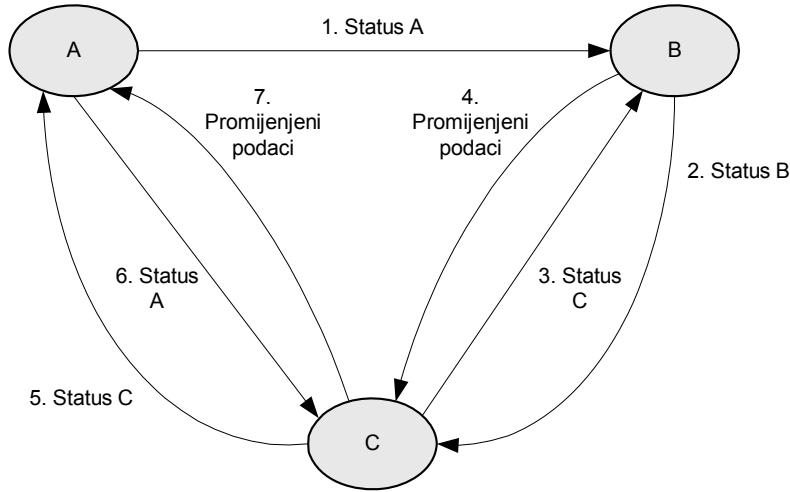
podataka uvodi problem nadzora preslike podataka. Promjene na podacima jedne preslike kataloga potrebno je paralelno ostvariti i na svim preostalim preslikama kako bi se očuvala jednoznačnost podataka. Postupak osvježavanja podataka zahtjeva komunikaciju između kataloga te se izvodi na svim preslikama uvišestručenog kataloga usluga. Radna svojstva kataloga usluga rastu samo u slučaju male učestalosti osvježavanja podataka kataloga usluga. Usklajivanje podataka u preslikama kataloga zahtjeva uporabu naprednih komunikacijskih i transakcijskih algoritama.

Postoji više protokola uvišestručavanja podataka. Jedna od strategija uvišestručavanja podataka kataloga usluga je strategija *lijenog uvišestručavanja* (engl. lazy replication) [52]. Prema strategiji lijenog uvišestručavanja korisnik koji dodaje zapis u katalog usluga naziva se vlasnik zapisa, a čvor u koji se zapis dodaje naziva se početni čvor. Samo vlasniku zapisu omogućeno je promijeniti zapis i to samo na početnom čvoru. Prilikom promjene zapisu, početni čvor poveća sljedni broj izmjene zapisu te izvede lokalnu izmjenu zapisu. Slijedni broj izmjene zapisu jedinstveno određuje izmjenu zapisu na početnom čvoru.

Osvježavanje zapisu izvodi se periodički kroz sve preslike uvišestručenog kataloga. Čvorovi kataloga se povezuju u logički prsten, tako da svaki čvor poznaje svog desnog i lijevog susjeda. Osvježavanje zapisu započinje jedan od čvorova kataloga tako da svog desnog susjeda obavijesti o stanju zapisu koje posjeduje. Ako susjed zaključi da ne posjeduje najnovije zapise, onda ih dohvati od svog lijevog susjeda, lokalno ih spremi, te svog desnog susjeda obavijesti o stanju zapisu koje posjeduje. Obavijest o greškama prosljeđuje se u suprotnom smjeru kroz logički prsten. Svaki čvor kataloga održava tablice *Status* i *DnevnikIzmjena*. Čvorovi kataloga u tablici *Status* održavaju za svaki zapis njegov posljednji sljedni broj izmjene. Tablica *DnevnikIzmjena* sadrži podatke o svim izmjenama na čvoru kataloga.

Slika 4-2 prikazuje primjer osvježavanja promjena zapisu kroz logički prsten uvišestručenog kataloga usluga koji se sastoji od tri čvora (A, B, C). Čvor A započinje osvježavanje zapisu tako da čvoru B pošalje svoju *Status* tablicu (1). Čvor B uspoređuje sadržaje dviju *Status* tablica te ako utvrdi da čvor A ne sadrži nove inačice zapisu, onda čvoru C šalje svoju *Status* tablicu (2). Ako čvor C na osnovi usporedbe *Status* tablica utvrdi da čvor B sadrži nove inačice zapisu, onda ih čvor C zatraži od čvora B tako da mu pošalje svoju *Status* tablicu (3). Čvor B usporedi svoju *Status* tablicu sa *Status* tablicom čvora C te odredi zapisu za koje ima novije inačice u odnosu na čvor C. Novije inačice zapisu čvor B šalje čvoru C (4). Nakon promjene zapisu, čvor C proslijedi istim postupkom nove zapisu čvoru A (5, 6, 7). U prvom prolazu kroz logički prsten čvor B nije svjestan promjena koje su vlasnici zapisu napravili na

čvoru C. Čvor A nakon određenog vremena započinje novi ciklus u kojem će čvor B biti obavješten o promjenama na čvoru C iz prethodnog ciklusa.



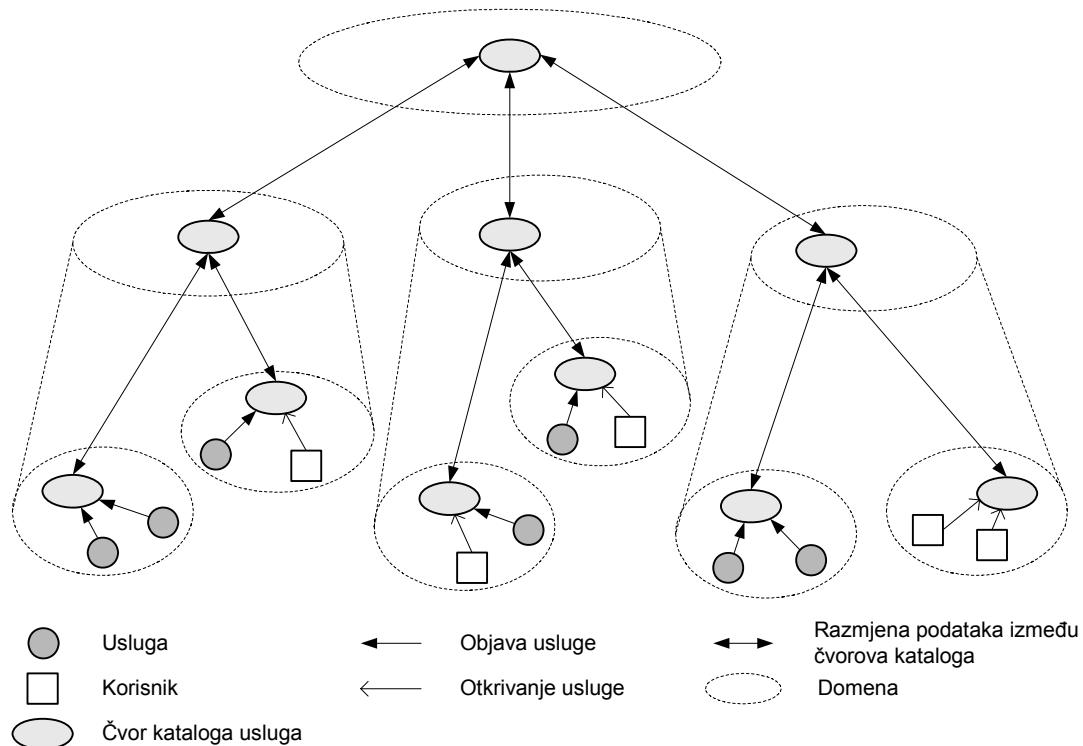
Slika 4-2: Uvišestručavanje podataka kataloga usluga

Vrlo važan parametar uvišestručenog kataloga je vrijeme između dva ciklusa osvježavanja podataka. Ako je vremenski razmak predug, onda je uvišestručeni katalog duže vrijeme u nejednoznačnom stanju. Ako je vremenski razmak prekratak, onda komunikacija među čvorovima negativno utječe na radna svojstva cjelokupnog kataloga. Zbog zadovoljavajućih radnih svojstava opisana strategija se često upotrebljava za replikaciju kataloga usluga smještenih na globalnoj mreži Internet. Strategija lijelog uvišestručavanja kataloga usluga iskorištena je u sklopu UDDI standarda.

4.1.2 Hijerarhijska arhitektura

Hijerarhijska arhitektura prikladan je pristup u izgradnji kataloga usluga za globalne mreže poput Interneta. Katalog usluga ostvaren hijerarhijskom arhitekturom posjeduje svojstvo razmjernog rasta.

Slika 4-3 prikazuje osnovnu hijerarhijsku arhitekturu kataloga usluga. Globalno okruženje se hijerarhijski dijeli u domene, tako da se domena na višoj razini sastoji od više domena niže razine. Osnovni elementi radnog okruženja su korisnici, usluge i čvorovi kataloga. Korisnici komuniciraju samo sa lokalnim čvorom kataloga koji pripada istoj domeni kao i korisnik. Usluge objavljaju svoje postojanje u lokalni čvor kataloga. Svaka domena posjeduje jedan čvor kataloga. Slično kao i domene, čvorovi kataloga su također organizirani u hijerarhiju. Samo lokalni čvorovi kataloga komuniciraju s korisnicima i pružateljima usluga. Čvorovi kataloga na višim razinama komuniciraju isključivo s ostalim čvorovima kataloga.



Slika 4-3: Hijerarhijska arhitektura kataloga usluga

Hijerarhijski katalog usluga zasniva se na dva ključna mehanizma: *nepotpunom sakupljanju informacija o uslugama* te na *raspodijeljenoj obradi upita*. Nepotpuno sakupljanje informacija o uslugama je širenje podataka o uslugama od lokalnih čvorova kataloga prema čvorovima na višim razinama. Postupak započinje lokalni čvor koji svom nadređenom čvoru šalje združene informacije o uslugama pripadne domene. U združene informacije ugrađuju se osnovne informacije o uslugama domene. Odabir osnovnih informacija o uslugama ovisan je o specifičnom programskom ostvarenju kataloga usluga. Postupkom združivanja informacija smanjuje se količina podataka koja se šalje nadređenom čvoru te se time uklanja opasnost preopterećenja komunikacije. Roditeljski čvor spremi združene podatke od svih svojih podređenih čvorova kataloga. Sakupljene združene podatke čvor ponovno združuje i šalje svom nadređenom čvoru. Postupak se rekursivno ponavlja sve do korijenskog čvora kataloga. Lokalni čvorovi na kraju postupka nepotpunog sakupljanja informacija sadrže potpune informacije o uslugama, dok čvorovi kataloga na višim razinama sadrže osnovne informacije o uslugama.

Raspodijeljena obrada upita uključuje prosljeđivanje upita kroz hijerarhiju čvorova kataloga s ciljem pronađaska čvora kataloga koji sadrži potpuni zapis o traženom pružatelju usluge. Korisnik šalje upit svom lokalnom čvoru kataloga. Ako lokalni čvor nema potpune informacije o traženoj usluzi, onda prosljeđuje upit svom nadređenom čvoru. Upit se rekursivno prosljeđuje sve višim razinama hijerarhije sve dok se ne dođe do čvora kataloga koji sadrži

osnovne informacije o traženoj usluzi. Kada se pronađe čvor kataloga s osnovnim informacijama o usluzi, upit se prosljeđuje nižim razinama hijerarhije kako bi se došlo do lokalnog čvora koji sadrži potpune informacije o traženoj usluzi. Na kraju pretraživanja svi lokalni čvorovi kataloga s potpunim informacijama o traženoj usluzi vraćaju potrebne informacije lokalnom čvoru kataloga koji je započeo pretragu. Originalni čvor vraća rezultate pretraživanja korisniku.

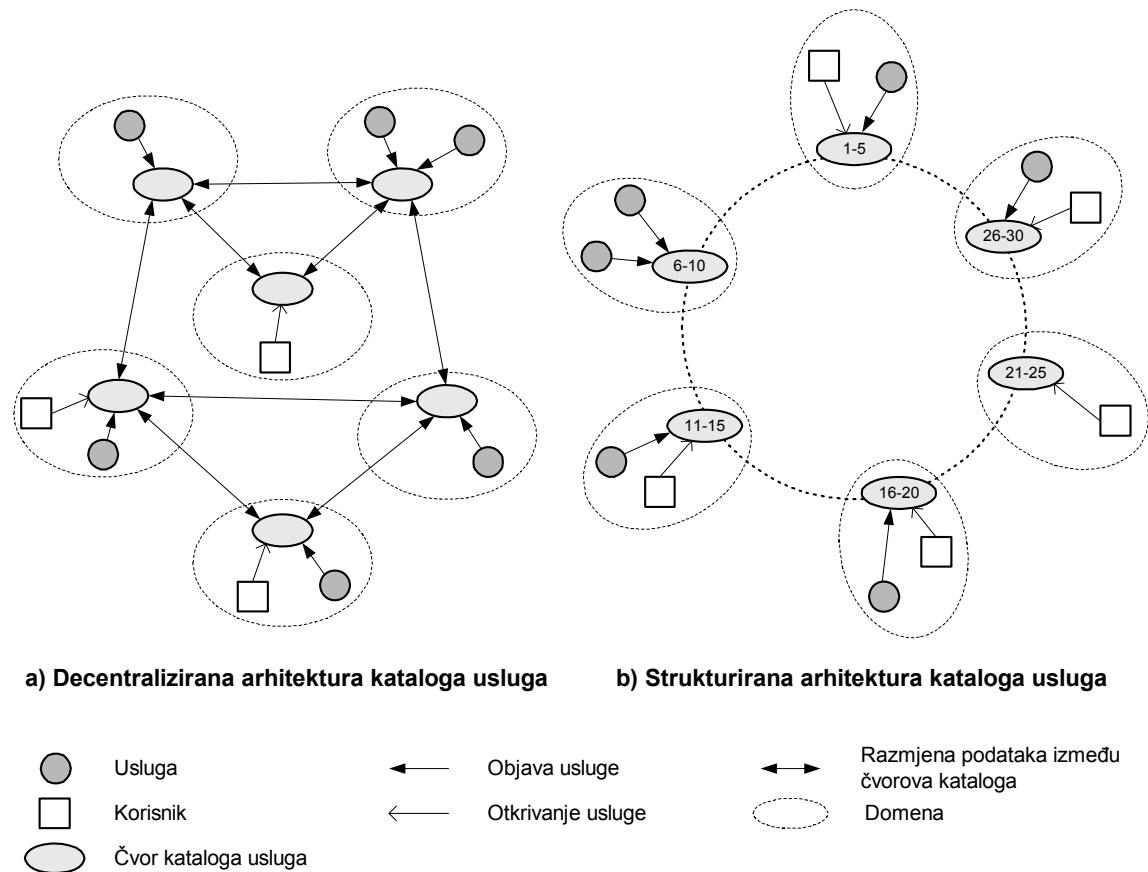
Opisana arhitektura kataloga usluga osigurava razmjeran rast kataloga, ali ima za posljedicu dugo vrijeme pretraživanja. Vrijeme obrade upita raste s 'udaljenošću' između korisnika i pružatelja usluge, pri čemu se udaljenost izražava kao broj čvorova kataloga koje upit mora proći na svom putu kroz hijerarhiju. U [53] se predlaže rješenje opisanog problema. Osnovna tehnika u poboljšanju osnovne arhitekture je uporaba privremenog spremanja podataka o uslugama (engl. caching) te propagiranje potpunih informacija o uslugama u čvorove viših razina kataloga.

Primjer uspješne uporabe hijerarhijske arhitekture je *Monitoring and Discovery System* (MDS) [70] informacijski sustav koji je sastavni dio Globus Toolkit skupa alata za izgradnju Grid aplikacija. Sustav je prilagođen otkrivanju informacija o sredstvima unutar Grid računalne okoline. MDS se zasniva na dvorazinskoj hijerarhiji informacijskih usluga. Na dnu hijerarhije su *pružatelji informacija* koji sadrže detaljne informacije o pojedinim sredstvima u računalnom okruženju. Iznad pružatelja informacija se nalaze *sakupljački direktori* koji sadrže samo osnovne informacije o pojedinim sredstvima u računalnom okruženju. *Sakupljački direktori* su specijalizirani za određenu vrstu sredstava. Korisnici šalju *sakupljačkim direktorijima* upite o željenim sredstvima. *Sakupljački direktori* kao rezultat vraćaju osnovne informacije o sredstvima. Detaljnije informacije o sredstvima korisnik dohvata od pripadnog *pružatelja informacija*.

4.1.3 Arhitektura zasnovana na mrežama ravnopravnih sudionika

Mreže ravnopravnih sudionika (engl. peer-to-peer networks, P2P) uklanjaju podjelu mrežnih sudionika na poslužitelje i korisnike, odnosno svaki sudionik P2P mreže istovremeno je poslužitelj i korisnik. Izvorna i najrasprostranjena uporaba P2P mreža je razmjena datoteka. Dva su osnovna mehanizma P2P mreže: mehanizmi razmjene datoteka i mehanizmi otkrivanja datoteka. Sudionici mreže dohvataju potrebne datoteke izravno od ostalih sudionika. Decentralizirana razmjena datoteka ima svojstvo razmernog rasta jer se porastom potrebe za određenom datotekom istovremeno povećava i broj sudionika koji posjeduju njezinu presliku. Svojstvo razmernog rasta teže je ostvariti u slučaju mehanizama otkrivanja datoteka. Mehanizmi otkrivanja datoteka u postojećim P2P sustavima dijele se u tri kategorije:

centralizirani, decentralizirani i strukturirani. Arhitektura centraliziranog mehanizma otkrivanja datoteka nema svojstvo razmjernega rasta, a identična je arhitekturi opisanoj u odjeljku 4.1.1. Arhitekture decentraliziranog i strukturiranog mehanizma otkrivanja datoteka u P2P mrežama moguće je iskoristiti u oblikovanju kataloga usluga.



Slika 4-4: Arhitekture kataloga usluga zasnovane na mehanizmima P2P mreža

Slika 4-4a prikazuje arhitekturu kataloga usluga zasnovanu na decentraliziranim P2P sustavima. Korisnici i usluge grupiraju se u domene tako da svaka domena posjeduje jedan čvor kataloga usluga. Čvor kataloga održava informacije o uslugama pripadne domene. Čvor kataloga također posjeduje znanje o lokacijama susjednih čvorova kataloga. Prilikom otkrivanja usluga, korisnik šalje upit domenskom čvoru kataloga. Čvor kataloga upit prosljeđuje susjednim čvorovima koji ga dalje rekurzivno prosljeđuju svojim susjedima. Opisanim postupkom katalog usluga se preplavljuje upitima. Ako čvor kataloga posjeduje informacije o traženoj usluzi, onda ih šalje istim putem kojim je stigao upit. Povratak informacija istim putem omogućen je privremenim spremanjem informacija o prosljeđenim upitimima koje se spremaju na čvorovima

kataloga. Privremeno spremanje informacija o prosljeđenim upitima također sprječava višestruko slanje istih upita.

Opisana arhitektura kataloga usluga omogućuje jednostavno pristupanje i odstupanje čvorova iz kataloga usluga. Novom čvoru kataloga je za pristup nužno znanje o lokaciji barem jednog postojećeg čvora kataloga. Novi čvor kroz nekoliko iteracija obaveštava postojeće čvorove o svom postojanju pri čemu saznaće lokacije njima susjednih čvorova. Ako je čvor kataloga nedostupan ili je odstupio iz kataloga usluga, onda ga ostali čvorovi kataloga izbacuju iz svog popisa susjednih čvorova. Opisana arhitektura kataloga usluga otporna je na pogreške uslijed nedostupnosti pojedinih čvorova kataloga, ali ne ostvaruje svojstvo razmijernog rasta. Prilikom otkrivanja usluga potrebno je ograničiti preplavljivanje kataloga zbog čega neke informacije mogu ostati neotkrivene.

Slika 4-4b prikazuje arhitekturu kataloga usluga zasnovanu na strukturiranim P2P sustavima. Osnovni princip ove arhitekture kataloga usluga je raspodjela tablice raspršenog adresiranja između čvorova kataloga usluga. Zasnovanost na tehnikama raspršenog adresiranja ograničava ovu arhitekturu kataloga na upis i pretraživanje informacija koje su u obliku (*ključ, vrijednost*). Na primjer, *ključ* može biti ime usluge, a *vrijednost* može biti lokacija usluge. Otkrivanje informacija izvodi se na osnovi *ključa* pa u ovom primjeru korisnik mora unaprijed znati ime usluge čija ga lokacija zanima.

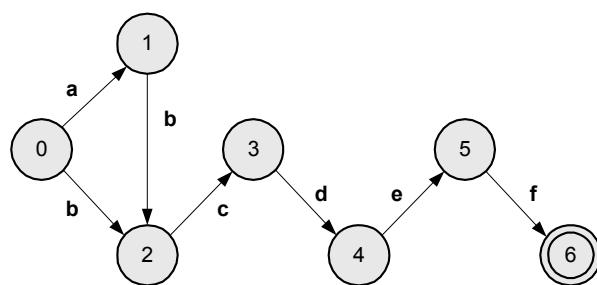
Korisnici i usluge grupiraju se u domene tako da svaka domena posjeduje jedan čvor kataloga. Svaki čvor kataloga zadužen je za održavanje dijela tablice raspršenog adresiranja. Čvorovi kataloga dodatno održavaju informacije o njima poznatim čvorovima kataloga usluga. Za svaki poznati čvor kataloga usluga održavaju se informacije o lokaciji i pripadnom dijelu tablice raspršenog adresiranja. Usluga objavljuje informacije o svom postojanju u obliku uređenog para (*ključ, vrijednost*) u čvor kataloga pripadne domene. Čvor kataloga na osnovi *ključa* i funkcije raspršenog adresiranja (engl. hash function) određuje poziciju podatka *vrijednost* u tablici raspršenog adresiranja. Ako čvor kataloga nije odgovoran za izračunati dio tablice, onda uređeni par proslijedi poznatom čvoru čiji je dio tablice raspršenog adresiranja najbliži izračunatoj poziciji. Ponavljanjem navedenog postupka pronađe se čvor kataloga u čiji se dio tablice raspršenog adresiranja spremi informacija o novoj usluzi. Prilikom otkrivanja usluge, korisnik šalje čvoru kataloga pripadne domene upit o usluzi. Upit o usluzi je podatak *ključ* spremjenog uređenog para. Pronalazak podatka *vrijednost* uređenog para u tablici raspršenog adresiranja izvodi se analognim algoritmom kao i u slučaju spremanja uređenog para.

Čvorovi kataloga mogu nesmetano pristupati i odstupati iz kataloga usluga. Katalog usluga prilikom pristupanja ili odstupanja čvorova samostalno osvježava raspodijeljenu tablicu

adresiranja. Opisana arhitektura ostvaruje svojstvo razmjernega rasta jer se prilikom otkrivanja usluge upiti obrađuju samo na podskupu čvorova kataloga.

Primjer izgradnje kataloga usluga zasnovanog na decentraliziranim P2P sustavima opisan je u radu [71]. Opisani katalog usluga izgrađen je uporabom Gnutella protokola [72]. Gnutella protokol je osnova Gnutella P2P sustava za razmjenu datoteka. Unutar kataloga usluga spremi se opis usluge zasnovan na DAML-S [63] ontologiji. Otkrivanje usluge izvodi se na osnovi *profila usluge* koji je podskup DAML-S opisa usluge. Osnovni nedostatak opisanog kataloga usluga je neučinkovito preraživanje kataloga jer se upit obrađuje na svim čvorovima kataloga usluga.

- a:** registracija
- b:** prijava
- c:** pretraživanje knjiga
- d:** narudžba
- e:** prijedlog cijene
- f:** prihvatanje cijene



Slika 4-5: Primjer opisa usluge konačnim automatom

Primjer izgradnje kataloga usluga zasnovanog na strukturiranim P2P sustavima opisan je u radu [51]. Opisani katalog usluga zasniva se na pretpostavci da korisnik unaprijed zna slijed međudjelovanja koje treba ostvariti s nekom uslugom. Na primjer, korisnik uslugu "prodaja knjiga" vidi kroz skup međudjelovanja: registracija, prijava, pretraživanje knjiga, narudžba, prijedlog cijene i prihvatanje cijene. Međudjelovanja usluge i korisnika moguće je prikazati u obliku konačnog automata, pri čemu prijelazi automata odgovaraju međudjelovanju korisnika i usluge, a stanja automata odgovaraju stanjima usluge. Slika 4-5 prikazuje primjer konačnog automata usluge "prodaja knjiga". Korisnici ne moraju uvijek ostvariti isti slijed međudjelovanja sa uslugom. U slučaju usluge "prodaja knjiga", korisnik koji je prije koristio uslugu ne mora se ponovno registrirati pa postoje dva slijeda mogućih međudjelovanja usluge i korisnika:

<*registracija, prijava, pretraživanje knjiga, narudžba, prijedlog cijene, prihvatanje cijene*>,
<*prijava, pretraživanje knjiga, narudžba, prijedlog cijene, prihvatanje cijene*>.

Slijed međudjelovanja usluge i korisnika odgovara jednom putu kroz konačni automat od početnog do konačnog stanja. Informacije o usluzi spremaju se u katalog usluga u obliku uređenih parova (*ključ, podaci*), gdje je *ključ* mogući slijed međudjelovanja, a *podaci* su informacije o usluzi. Svaki slijed međudjelovanja usluge i korisnika ključ je jednog uređenog para. Informacije o usluzi "prodaja knjiga" spremaju se dva puta u katalog usluga, za svaki slijed

međudjelovanja po jednom. Otkrivanje informacija usluga izvodi se na osnovi *ključa* uređenog para, odnosno na osnovi mogućih slijedova međudjelovanja usluge i korisnika.

4.1.4 Otkrivanje usluga u zgodom oblikovanim i pokretnim mrežama

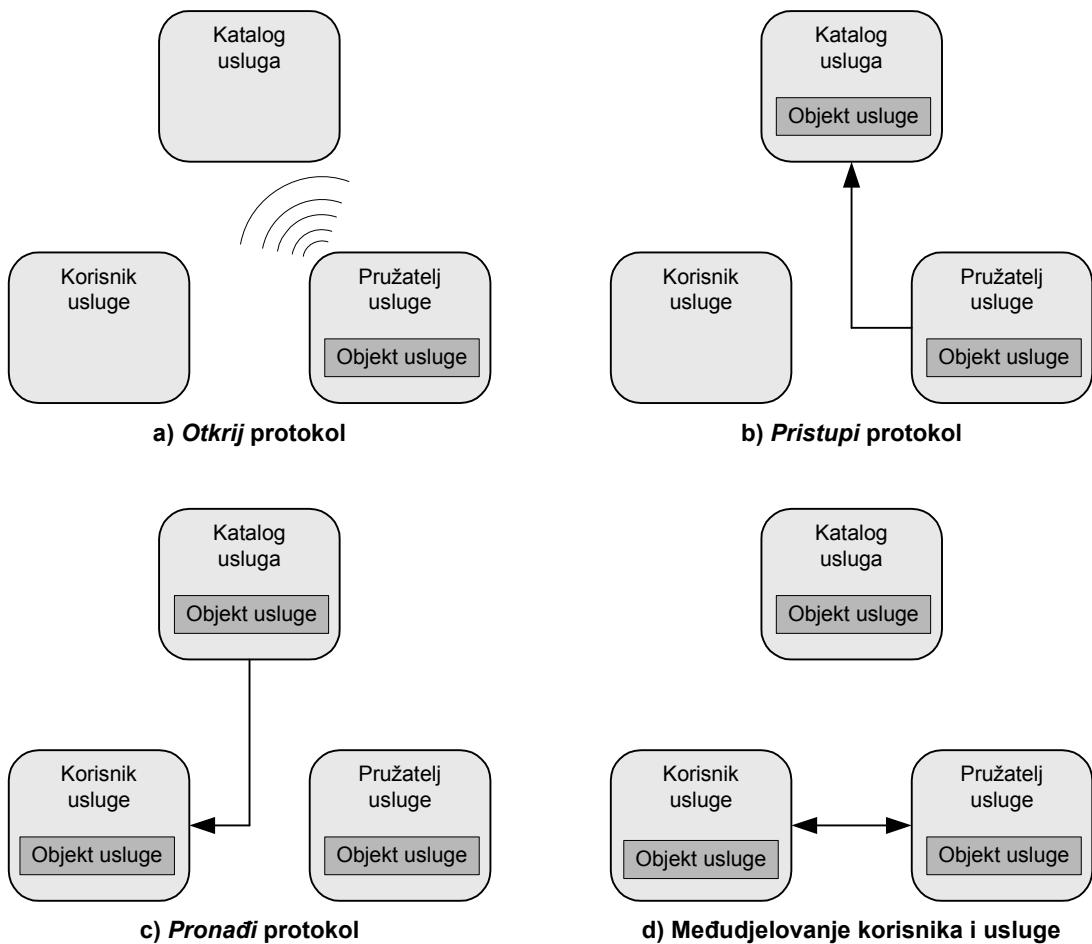
U zgodom oblikovanim i pokretnim mrežama (engl. ad hoc networks) otkrivanje usluga je poseban izazov. Za zgodom oblikovane i pokretne mreže svojstveno je nepostojanje fiksne mrežne infrastrukture. Čvorovi slobodno pristupaju i odstupaju iz takvih mreža, zbog čega je topologija logičke mreže nepredvidiva i spontana. Takve mreže se obično zasnivaju na bežičnoj komunikaciji među čvorovima. Otkrivanje usluga u takvim mrežama je problematično zbog nepostojanja fiksne mrežne infrastrukture, a time niti unaprijed poznatog kataloga usluga.

Alternativne metode otkrivanja usluga koje zaobilaze statične adrese kataloga usluga i koje su prikladne u zgodom oblikovanim i pokretnim mrežama zasnivaju se na grupnom slanju poruka mrežom (engl. multicast and broadcast). Ove metode primjenjive su u mrežama s malim brojem čvorova. Uporaba ovih metoda omogućava međusobno otkrivanje usluga u računalnom okruženju te automatizira pripremu usluga za međusobnu komunikaciju. Navedeni pristup uvelike olakšava administrirajuću i održavanje računalne okoline.

Primjer sustava koji se zasniva na dinamičkom otkrivanju usluga je Jini radni okvir [69]. Jini radni okvir je razvijen u Sun Microsystems korporaciji i zasniva se na mogućnostima Java računalne platforme. Cilj Jini radnog okvira je pružiti jednostavan pristup do sredstava smještenih u mreži te pri tome dopustiti korisnicima da nesmetano mijenjaju vlastitu lokaciju. Jini radni okvir je specifično ostvarenje arhitekture zasnovane na uslugama, što znači da u sustavu postoje tri osnovne vrste entiteta: usluge, korisnici usluga i katalog usluga. Komunikacija između korisnika i usluga ostvaruje se Java mehanizmom za pozivanje udaljenih metoda (engl. Remote Method Invocation, RMI). Osnova Jini sustava su tri protokola koja se koriste u komunikaciji entiteta s katalogom usluga: *otkrij*, *pristupi* i *pronadi*. *Otkrij* protokol pronalazi katalog usluga. Protokol *pristupi* izvodi registraciju usluge u katalog usluga. Protokol *pronadi* koriste korisnici usluga prilikom pretraživanja kataloga usluga.

Slika 4-6 prikazuje proces otkrivanja usluga u Jini radnom okviru. Prvi dio je registracija usluge u Jini radno okruženje. Pružatelj usluge prilikom registracije najprije uporabom *otkrij* protokola saznaće lokaciju kataloga usluga (slika 4-6a). *Otkrij* protokol se zasniva na slanju grupnih poruka svim čvorovima na mreži. Sljedeći korak u registraciji je spremanje objekta usluge u katalog usluga uporabom *pristupi* protokola (slika 4-6b). Objekt usluge je sučelje usluge izraženo u Java programskom jeziku. Sučelje usluge sadrži opisne atribute usluge te popis metoda usluge koje korisnik može pozvati. Nakon registracije, usluga postaje spremna za

korištenje. Korisnici putem *pronadi* protokola otkrivaju uslugu na osnovi sučelja zapisanog u Java programskom jeziku (slika 4-6c). Rezultat otkrivanja usluge je dohvaćanje preslike objekta usluge iz kataloga usluga te učitavanje tog objekta u korisničku aplikaciju. Završni korak otkrivanja usluge je pozivanje metoda udaljene usluge (slika 4-6d).



Slika 4-6: Otkrivanje usluga u Jini računalnoj okolini

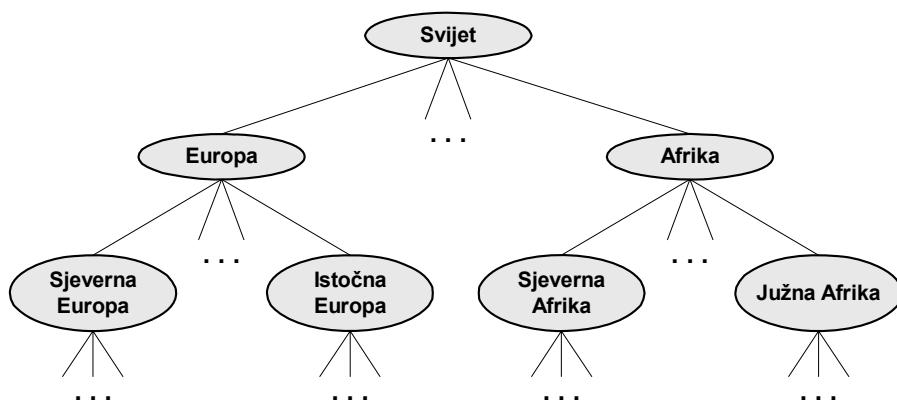
Još jedan primjer otkrivanja usluga u zgodom oblikovanim i pokretnim mrežama je *Bonjour* tehnologija [68] (prije poznata kao *Rendezvous*), razvijena u istraživačkim laboratorijima Apple korporacije. *Bonjour* tehnologija omogućuje automatsko spajanje elektroničkih uređaja na mrežu i njihovo međudjelovanje, a da pri tome nije potrebno posebno konfigurirati uređaje. Bonjour tehnologija je namijenjena lokalnim bežičnim i ožičanim mrežama. Tehnologija je sastavni dio Mac OS X operacijskog sustava te se koristiti u aplikacijama za razmjenu datoteka, automatsko otkrivanje mrežnih pisača te za komunikaciju s ostalim ljudima putem računalne mreže.

4.2 Odabir usluga

Proces otkrivanja usluga korisniku vraća popis svih usluga koje zadovoljavaju određene zahtjeve. Iz dobivenog popisa usluga korisnik odabire uslugu koju će koristiti. Zbog nejednoznačnosti ljudskih jezika nije prikladno usluge odabrati samo na osnovi njihovog imena. Funkcijski jednake usluge mogu imati različita imena, a također i funkcijski različite usluge mogu imati jednaka imena. Opisi usluga u katalogu usluga moraju sadržavati i podatke koji omogućuju kvalitetan odabir prikladne usluge.

4.2.1 Odabir usluga na osnovi kategorija

Kategorija se definira kao skup objekata koji su jednaki s obzirom na određena svojstva. Obično se kategorijama pridružuje ime (npr. *životinje, psi*). Taksonomija je hijerarhijski sustav koji definira veze između kategorija prema principu uključivanja. Svaka kategorija unutar taksonomije je u potpunosti uključena u drugu kategoriju, osim ako se ne radi o korijenskoj kategoriji sustava. Slika 4-7 prikazuje primjer geografske taksonomije kojom se svijet dijeli na kontinente, regije, države itd.



Slika 4-7: Geografska taksonomija

Principle kategorija moguće je iskoristiti unutar kataloga usluga. Razvrstavanjem usluga u kategorije korisniku se pruža dodatni prostor informacija u odabiru prikladne usluge. Na primjer, ako se usluge u katalogu razvrstaju prema geografskoj taksonomiji sa slike 4-7, onda korisnik svoj odabir može usmjeriti na njemu geografski najbliže usluge. Odabir usluga na osnovi kategorija podrazumijeva znanje korisnika o značenju kategorija.

Primjeri unaprijed definiranih taksonomija koje se koriste u javnim UDDI katalozima usluga su: NAICS, UNSPSC i ISO 3166. NAICS (North American Industry Classification System) taskonomija je standard za kategorizaciju industrijskog poslovanja. UNSPSC (Universal Standard Products and Services Code System) taksonomija koristi se za kategorizaciju produkata

i usluga. ISO 3166 (International Organization for Standardization Geographic Taxonomy) taksonomija omogućuje geografsku klasifikaciju. Usluge unutar UDDI kataloga moguće je istovremeno kategorizirati s obzirom na više taksonomija. UDDI specifikacija dodatno omogućuje definiranje korisničkih taksonomija unutar kataloga usluga.

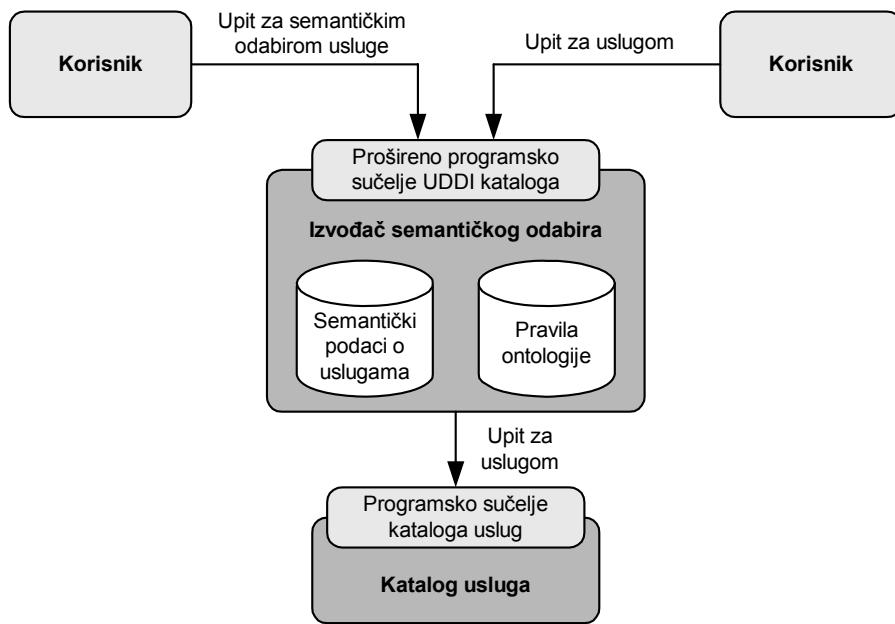
4.2.2 Semantički odabir usluga

Cilj semantičkog odabira usluga je automatizirati postupak otkrivanja i odabira usluga. Uobičajene metode otkrivanja i odabira usluga zahtijevaju djelovanje programera koji odabire prikladnu uslugu na osnovi opisa izraženog u prirodnom jeziku. Ovakve metode ograničavaju postupak otkrivanja i odabira usluga na vrijeme izgradnje raspolijeljene aplikacije. Metode semantičkog odabira usluga omogućuju otkrivanje i odabir usluga tijekom izvođenja raspolijeljene aplikacije.

Usluga zapisuje u katalog opis svojih mogućnosti i ograničenja. Opis usluge uključuje uvjete koje korisnik mora zadovoljiti prije komunikacije s uslugom, ograničenja u izvođenju usluge te definiciju komunikacijskog protokola. Definicija komunikacijskog protokola obuhvaća semantiku ulaznih i izlaznih poruka usluge. Opis usluge zapisuje se u katalog usluga uporabom unaprijed dogovorenih ontologija. U slučaju Web usluga obično se koriste OWL-S ili DAML-S ontologije.

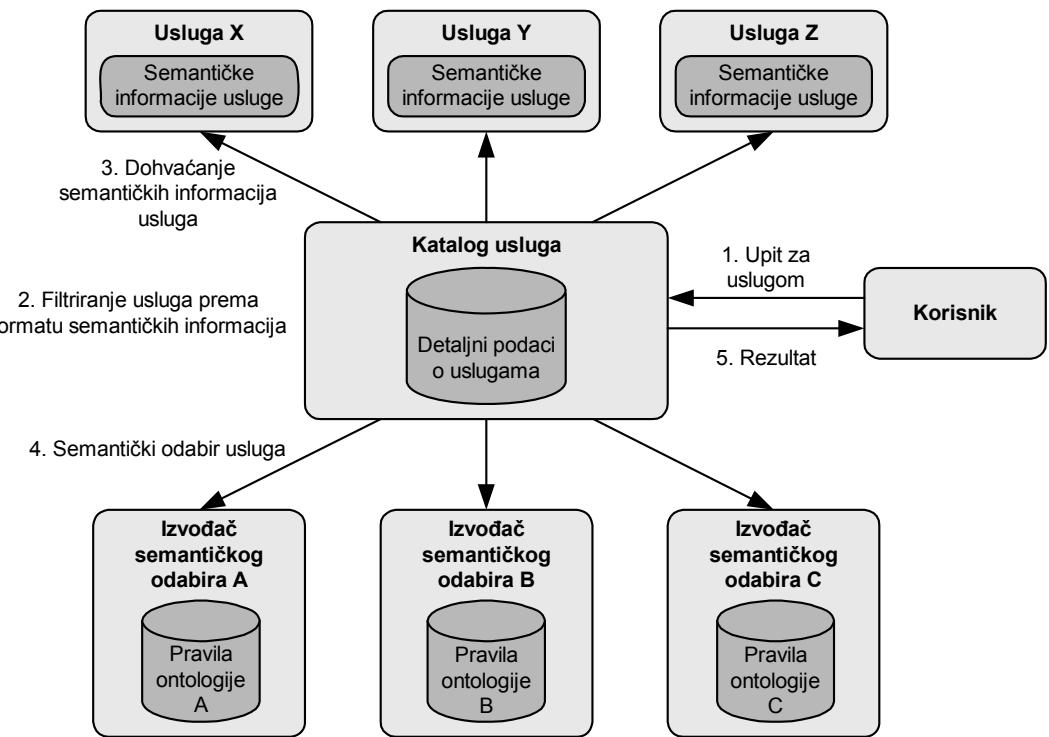
Korisnik prilikom pretraživanja kataloga usluga navodi poželjna svojstva usluga. Poželjna svojstva usluge uključuju strukturu ulaznih i izlaznih poruka, potporu sigurnoj komunikaciji ili željenu razinu kvalitete. Upiti za uslugama se također izražavaju putem dogovorenih ontologija. Katalog usluga prikladnu uslugu odabire na osnovi primljenog korisničkog upita, spremlijenih opisa usluga i pravila zaključivanja. Pravila zaključivanja dio su dogovorenih ontologija. Katalog usluga korisniku vraća opis usluge koji najviše odgovara zadanom upitu.

Umjesto izgradnje specifičnih semantičkih kataloga usluga prikladnije je proširiti postojeće kataloge usluga mehanizmima za semantičko otkrivanje i odabir usluga. Na primjer, UDDI standard definira mehanizme za otkrivanje Web usluga. Međutim, mehanizmi UDDI standarda ne ostvaruju potporu semantičkom otkrivanju Web usluga. Zbog općeprihvaćenosti UDDI standarda prikladnije je postojeće UDDI kataloge proširiti semantičkim mehanizmima nego ih zamijeniti specijaliziranim semantičkim katalozima.



Slika 4-8: Prilagodba kataloga usluga semantičkom otkrivanju i odabiru usluga

Slika 4-8 prikazuje primjer proširenja postojećeg kataloga usluga mehanizmima za semantičko otkrivanje i odabir usluga [73]. Katalog usluga u ovom sustavu ostaje nepromijenjen. Između korisnika i kataloga usluga smješten je *Izvođač semantičkog odabira* u koji se spremaju semantičke informacije o uslugama. Detaljne informacije o uslugama, poput adrese usluge ili opisa programskog sučelja, spremaju se u katalog usluga. Uz semantičke podatke o uslugama, unutar *Izvođača semantičkog odabira* nalazi se i definicija odabrane ontologije koja određuje značenje semantičkih podataka. Korisnici zahtjeve za otkrivanjem usluga šalju *Izvođaču semantičkog odabira*. *Izvođač semantičkog odabira* ostvaruje jednako programsko sučelje kao i katalog usluga uz dodatne metode vezane uz semantički odabir usluga. Ovakvo ostvarenje *Izvođača semantičkog odabira* korisnicima omogućuje pozivanje kataloga na stari način te istovremeno omogućuje uporabu mehanizama semantičkog odabira usluga. Upiti u kojima se ne zahtijeva semantički odabir usluge u potpunosti se obrađuju na katalogu usluga. *Izvođač semantičkog odabira* u tom slučaju samo proslijedi upite i rezultate obrade. Upite za semantičkim odabirom usluge korisnik priprema prema pravilima odabrane ontologije. *Izvođač semantičkog odabira* na osnovi upita i ugrađenih pravila ontologije pretražuje semantičke podatke o uslugama. Rezultat pretraživanja je usluga koja najbolje odgovara zadnom upitu. Za odabranu uslugu *Izvođač semantičkog odabira* dohvata iz kataloga usluga njene detaljne podatke i šalje ih korisniku.



Slika 4-9: Katalog usluga proširen mehanizmima za semantičko otkrivanje i odabir usluga

Opisani sustav ograničen je na jednu ontologiju prema kojoj se izvodi semantički odabir usluga. Slika 4-9 prikazuje sustav koji omogućuje odabir usluga na osnovi više ontologija. Osim uobičajenih informacija, poput lokacije i opisa programskog sučelja, usluge dodatno objavljuju lokaciju i format zapisa semantičkih informacija. Spremanjem semantičkih informacija izvan kataloga usluga uklanja se potreba za prilagodbom kataloga na različite formate zapisa semantičkih informacija. U katalog usluga dodatno se prijavljuju različiti *Izvođači semantičkog odabira* čija je uloga semantički odabir usluga. Sastavni dio *Izvođača semantičkog odabira* je skup pravila ontologije na osnovu kojih se izvodi odabir. Za svaki format zapisa semantičkih podataka, odnosno za svaku korištenu ontologiju postoji poseban *Izvođač semantičkog odabira*. Prilikom semantičkog odabira usluga korisnik šalje prikladan upit katalogu usluga (1). Katalog usluga na osnovi upita odabire skup usluga čiji format semantičkih informacija odgovara formatu upita (2). Za odabran skup usluga katalog dohvata njihove semantičke informacije (3). Skup prikupljenih semantičkih informacija o uslugama proslijeduje se zajedno s korisničkim upitom prikladnom *Izvođaču semantičkog odabira* (4). *Izvođač semantičkog odabira* na osnovi semantičkih informacija, korisničkog upita i pravila ontologije odabire najprikladniju uslugu. Na kraju, katalog usluga šalje korisniku detaljne informacije o odabranoj usluzi (5). U radu [58] opisuje se primjena ovakve arhitekture u svrhu proširenja postojećeg UDDI kataloga Web usluga mehanizmima semantičkog odabira usluga.

4.2.3 Kvaliteta odabira usluge

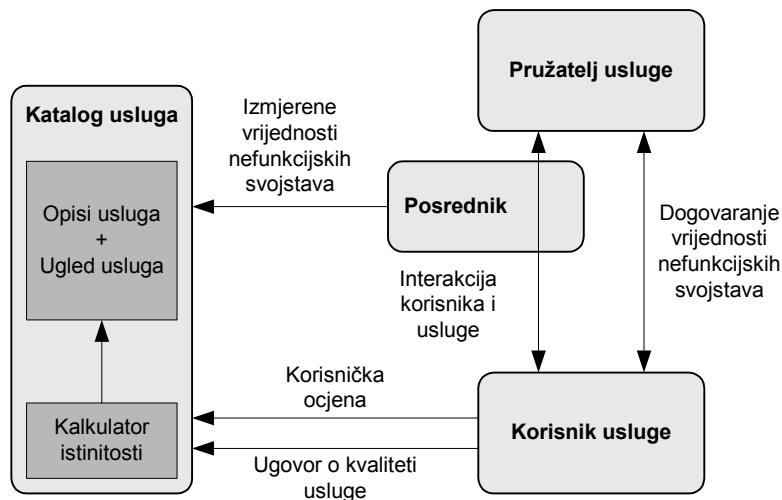
Osim funkcijskih svojstava usluge kod odabira usluge važna su i nefunkcijkska svojstva. Na osnovi nefunkcijskih svojstava usluge određuje se kvaliteta specifičnog ostvarenja usluge (engl. Quality of Service, QoS). Neka od nefunkcijskih svojstava usluge su: *vrijeme odziva, pouzdanost, razmjeri rast, kapacitet, upravljanje iznimkama, integritet, dostupnost i sigurnost*. Vrijeme odziva izražava brzinu kojom usluga obraduje korisničke zahtjeve. Pouzdanost se definira kao sposobnost usluge da pruži traženu funkcionalnost pod zadanim uvjetima i unutar zadanog vremenskog razdoblja. Razmjeran rast je usko vezan uz vrijeme odziva usluge te izražava sposobnost usluge da obradi korisničke zahtjeve unutar zadanog vremenskog intervala bez obzira na njihovu brojnost. Kapacitet se definira kao maksimalan broj istovremenih zahtjeva koje usluga obradi unutar određenog vremenskog intervala. Nemoguće je u razvoju usluge predvidjeti sve moguće iznimke u radu usluge, ali je zato moguće uporabom mehanizama za upravljanje iznimkama presresti neočekivane iznimke tijekom rada usluge. Integritet usluge osigurava da neovlaštena stranka ne može mijenjati ili pristupati podacima ili računalnim procesima usluge. Dostupnost usluge izražava se kao postotak vremena u kojem se očekuje spremnost usluge da uspješno obradi korisničke zahtjeve. Smještaj usluge na globalnu mrežu Internet automatski zahtijeva uporabu sigurnosnih mehanizama.

Kvaliteta usluge izražava se u obliku ugleda (engl. reputation) usluge. Ljudi često ugled primjenjuju u stvarnom životu. Na primjer, prilikom odabira prikladnog mjesta za ljetovanje, stranci mogu hrvatsku obalu odabrati zbog ugleda o čistom moru i ljubaznim ljudima. Ugled usluge se definira kao združeno iskustvo bivših korisnika usluge s obzirom na kvalitetu.

Tehnika ugleda obično se primjenjuje kod popularnih elektroničkih tržnica, kao što su eBay, OnSale Exchange i drugi. Elektroničke tržnice su posrednici između prodavača i kupaca pa s ciljem poboljšanja kvalitete trgovine omogućuju sudionicima transakcije da ocijene jedni druge. Ocjena se sastoji od numeričke ocjene i tekstualnog komentara. Srednja vrijednost svih ocjena daje ukupan ugled prodavača ili kupca. Određivanje ugleda ima nekoliko problema. U elektroničkom svijetu relativno je lako promijeniti identitet. Ako korisnik tijekom vremena stekne ugled niži od početnog, onda može doći u iskušenje da stvorи novi identitet i krene ispočetka. Također je vrlo važno pažljivo definirati funkciju za računanje ugleda. Na primjer, aritmetička srednja vrijednost ocjena nije najprikladnija mjera ugleda, jer korisnik koji na početku krene s lošim ocjenama teško će se u budućnosti riješiti lošeg ugleda. Ocjene dobivene u davnoj prošlosti trebaju se u postupku računanja ugleda uzimati s manjom težinom. Problem su i lažne ocjene, odnosno korisnik se može dogovoriti s "prijateljima" koji će mu dati visoke ocjene i tako mu povisiti ugled. Na isti način korisnik može povisiti vlastiti ugled stvaranjem novih

lažnih identiteta i visokim ocjenjivanjem samog sebe. Dobar sustav za praćenje ugleda korisnika mora razriješiti navedene probleme.

Prilikom računanja ugleda usluge potrebno je osim korisničkih subjektivnih ocjena uzeti u obzir i objektivne ocjene nefunkcijskih svojstava usluge. Primjer sustava koji omogućuje otkrivanje usluga na osnovi ugleda predložen je u radu [65]. Objektivne ocjene nefunkcionalnih svojstava usluge promatraju se kroz određeno vremensko razdoblje. Slika 4-10 prikazuje arhitekturu predloženog sustava. Ugled usluge spremi se u katalog usluga i sastoji se od tri numeričke vrijednosti: *korisnički rang*, *uslužnost* (engl. compliance) i *istinitost* (engl. verity).



Slika 4-10: Arhitektura sustava za otkrivanje usluga na osnovi ugleda usluge

Korisnički rang izračunava se kao aritmetička sredina numeričkih korisničkih ocjena. Korisnici šalju u katalog usluga numeričke ocjene nakon svakog međudjelovanja s uslugom. Numeričke ocjene su mjerilo korisnikove subjektivne percepcije kvalitete usluge.

Karakteristika uslužnosti je objektivna ocjena usluge koja prikazuje u kojoj je mjeri usluga zadovoljila unaprijed utvrđene kriterije kvalitete. Kriteriji kvalitete dogovaraju se između pružatelja i korisnika usluge prije samog poziva usluge. Rezultat dogovora je ugovor o kvaliteti usluge u kojem se formalno navodi pružatelj i korisnik usluge, nefunkcijska svojstva usluge, očekivane vrijednosti odabranih nefunkcijskih svojstava te moguće kazne pružatelju usluge ako ne zadovolji ugovor. Uspostavljeni ugovor o kvaliteti usluge korisnik šalje katalogu usluga. Mjerenje kvalitete pružene usluge se prepušta trećoj nezavisnoj strani, *posredniku*, koji nadgleda međudjelovanje korisnika i usluge. Posrednik prilikom međudjelovanja korisnika i usluge nadgleda vrijeme odziva, dostupnost i radna svojstva usluge. Izmjerene vrijednosti nefunkcijskih svojstava usluge posrednik šalje katalogu usluga. Katalog usluga uspoređuje izmjerene i

dogovorene vrijednosti nefunkcijskih svojstava usluge te na osnovi tih informacija računa uslužnost usluge.

Istinitost izražava dosljednost pružatelja usluge u kvalitetnom pružanju usluge. Ova karakteristika se izražava kao varijanca uslužnost za pozive usluge u nekom vremenskom razdoblju. Manja varijanca pokazuje da pružatelj usluge uspijeva održati konstantnu razinu kvalitete usluge kroz određeno vremensko razdoblje.

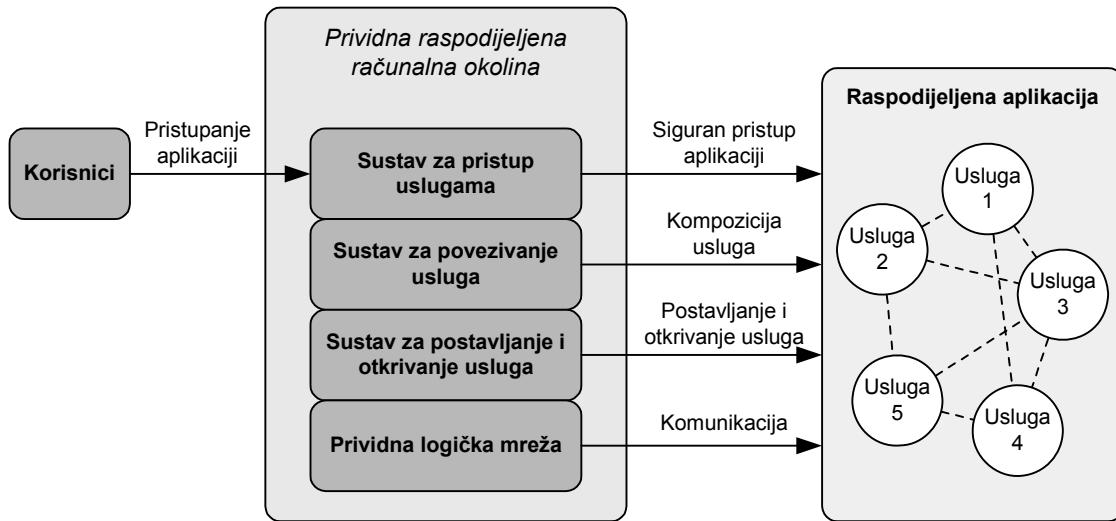
5 Prividna raspodijeljena računalna okolina

Prividna raspodijeljena računalna okolina (engl. Virtual Distributed Environment, VDE) je infrastruktura za razvoj, postavljanje i izvođenje raspodijeljenih aplikacija zasnovanih na uslugama. Razvoj raspodijeljenih aplikacija uporabom mehanizama *VDE* računalne okoline posebno je prilagođen ograničenim sposobnostima krajnjeg korisnika. Prilagodba sposobnostima krajnjih korisnika izražena je u automatizaciji složenih postupaka, poput postavljanja aplikacijskih usluga na čvorove računalne okoline, te u definiciji posebnih jezika opisivanja složenih raspodijeljenih zadataka. *VDE* računalna okolina razvijena je u suradnji Fakulteta elektrotehnike i računarstva u Zagrebu i tvrtke Ericsson Nikola Tesla, te je rezultat istraživanja u okviru nacionalnog CroGrid projekta. Računalna okolina ostvarena je u skladu s principima računarstva zasnovanog na uslugama. Sastavni dijelovi računalne okoline ostvareni su kao slabo povezane usluge koje putem otvorenih sučelja pružaju svoje funkcionalnosti.

5.1 Arhitektura Prividne raspodijeljene računalne okoline

Oblikovanje raspodijeljenih aplikacija za *VDE* računalnu okolinu zasniva se na procesima *prividnosti* (engl. virtualization) i *kompozicije*. Procesom prividnosti računalnim se sredstvima pridružuju javna i otvorena sučelja, odnosno sredstva se ostvaruju kao računalne usluge. Pojam *sredstvo* ima široko značenje. Na primjer, sredstvo može biti dio sklopolja, poput procesora, radne memorije ili tvrdog diska. Nadalje, sredstvo može biti i logičke prirode, poput aktivnog računalnog procesa, objekta ili tablice podataka. Javna i otvorena sučelja omogućuju ostalim entitetima računalne okoline upravljanje i nadgledanje sredstava. Proces kompozicije je oblikovanje složenih usluga na osnovi povezivanja skupa jednostavnih usluga. Dijelovi aplikacijske logike procesom prividnosti postaju usluge koje se procesom kompozicije međusobno povezuju u složenu raspodijeljenu aplikaciju. Uloga *VDE* računalne okoline je pružiti potporu tim osnovnim procesima izgradnje raspodijeljene aplikacije.

Slika 5-1 prikazuje arhitekturu *VDE* računalne okoline. *VDE* računalna okolina sastoji se od četiri slojevito postavljena podsustava: *Prividne logičke mreže*, *Sustava za postavljanje i otkrivanje usluga*, *Sustava za povezivanje usluga* i *Sustava za pristup uslugama*. Slojevitost arhitekture naglašava zavisnost pojedinih podsustava. Podsustav na višoj razini koristi funkcionalnosti podsustava na nižim razinama. Na dnu arhitekture je *Prividna logička mreža* koja omogućava raspodjelu programskih usluga u logičkom komunikacijskom prostoru neovisnom o adresama čvorova fizičke mreže.



Slika 5-1: Arhitektura Prividne raspodijeljene računalne okoline

Sustav za postavljanje i otkrivanje usluga nadovezuje se na logičku mrežu. Otkrivanje sredstava je nužna funkcionalnost za izgradnju i izvođenje raspodijeljenih aplikacija u dinamičkim računalnim okolinama. *Sustav za postavljanje i otkrivanje usluga* ostvaruje funkcionalnosti kataloga usluga u koji se spremaju lokacije usluga prisutnih u *VDE* računalnoj okolini. Dodatno, sustav automatizira postupak postavljanja usluga.

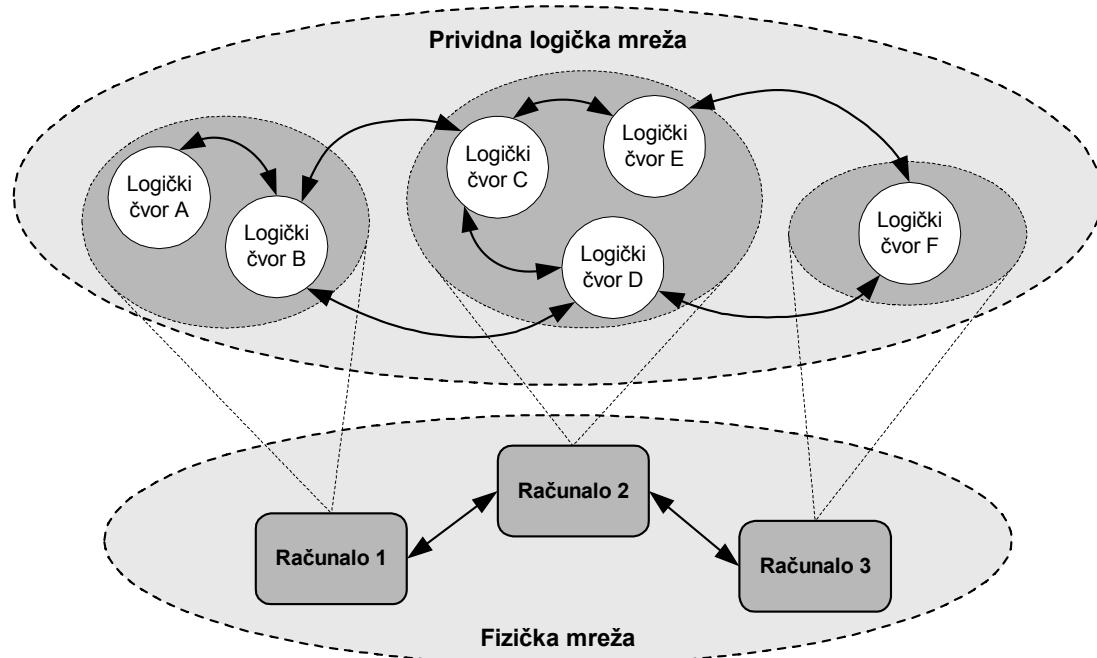
Sustav za povezivanje usluga ostvaruje proces kompozicije usluga. Usluge postavljene putem *Sustava za otkrivanje i postavljanje usluga* međusobno se povezuju i grade složenu raspodijeljenu aplikaciju. Za opisivanje kompozicije usluga definirani su posebni jezici te su izgrađeni jezični prevodioci kojima se opis kompozicije prevodi u izvršni oblik. *Sustav za povezivanje usluga* također pruža mehanizme za sinkronizaciju i komunikaciju složenih aplikacijskih procesa.

Sustav za pristup uslugama pruža mehanizme sigurnog pristupa uslugama unutar *VDE* računalne okoline. Sigurnosni mehanizmi *Sustava za pristup uslugama* ostvaruju funkcionalnosti autentikacije i autorizacije. U procesu autentikacije provjerava se identitet korisnika, dok se u procesu autorizacije provjeravaju korisnikova prava pristupa usluzi. Također, *Sustav za pristup uslugama* ostvaruje praćenje pristupanja korisnika uslugama.

U nastavku poglavljaju opisuju se svojstva i osnovna arhitektura *Prividne logičke mreže*, *Sustava za povezivanje usluga* i *Sustava za pristup uslugama*. *Sustav za postavljanje i otkrivanje usluga* tema je ovog rada te se detaljnije opisuje u poglavljima 6 i 7.

5.1.1 Prividna logička mreža

Prividna logička mreža ostvaruje potporu komunikaciji među uslugama VDE računalne okoline. Osnovni cilj *Prividne logičke mreže* je ostvariti nezavisnost raspodijeljene aplikacije s obzirom na fizičku mrežnu infrastrukturu. Navedena nezavisnost ostvarena je uvođenjem koncepta *logičkog čvora*. Komunikacijski mehanizmi logičkog čvora zamjenjuju komunikacijske mehanizme fizičkog računala te time ostvaruju nezavisnost komunikacije među uslugama s obzirom na fizičku računalnu mrežu. Logički čvorovi nesmetano pristupaju i odjavljaju se iz *Prividne logičke mreže* što ima za posljedicu dinamičku prirodu mrežne topologije. Nesmetan pristup i odjava čvorova iz mreže uvodi izazove u održavanju znanja o mrežnoj topologiji. Svaki logički čvor *Prividne logičke mreže* održava znanje o topologiji samo jednog dijela mreže. Zbog ograničenog znanja o mreži, logički čvorovi ne mogu uvijek izravno poslati korisničke zahtjeve na odredište. U takvim slučajevima se korisnički zahtjevi šalju posebnim usmjeravajućim čvorovima koji odlučuju o dalnjem usmjeravanju zahtjeva. Slika 5-2 prikazuje primjer konfiguracije *Prividne logičke mreže*. U primjeru je šest logičkih čvorova smješteno na tri fizička računala. Kao što je vidljivo sa slike, čvorovi *Prividne logičke mreže* sposobni su izravno komunicirati samo s podskupom logičkih čvorova.



Slika 5-2: Struktura *Prividne logičke mreže*

Prividna logička mreža unosi specifičan način adresiranja usluga smještenih na logičkim čvorovima. Adresa usluge je uređeni par oblika (*ime čvora, ime usluge*), pri čemu *ime čvora* jedinstveno identificira logički čvor mreže, a *ime usluge* jedinstveno identificira uslugu

postavljenu na logičkom čvoru. Ime čvora određuje se nezavisno o fizičkoj mrežnoj adresi. Ovako definirano adresiranje usluga uvodi ograničenje prema kojem na istom logičkom čvoru može postojati samo jedna preslika usluge. Prilikom putovanja korisničkog zahtjeva do odredišne usluge, prvi dio adrese koristi se za usmjeravanje zahtjeva do odredišnog logičkog čvora, dok se drugi dio adrese koristi tek na odredišnom logičkom čvoru u prosljeđivanju korisničkog zahtjeva usluzi.

Jedna od prednosti korištenja *Prividne logičke mreže* je prenosivost raspodijeljene aplikacije s jednog skupa računala na drugi. Umjesto promjena na raspodijeljenoj aplikaciji potrebno je samo postaviti odgovarajuću topologiju *Prividne logičke mreže*. Detaljniji pregled svojstava i programskog ostvarenja infrastrukture *Prividne logičke mreže* nalazi se u radu [93].

5.1.2 Sustav za povezivanje usluga

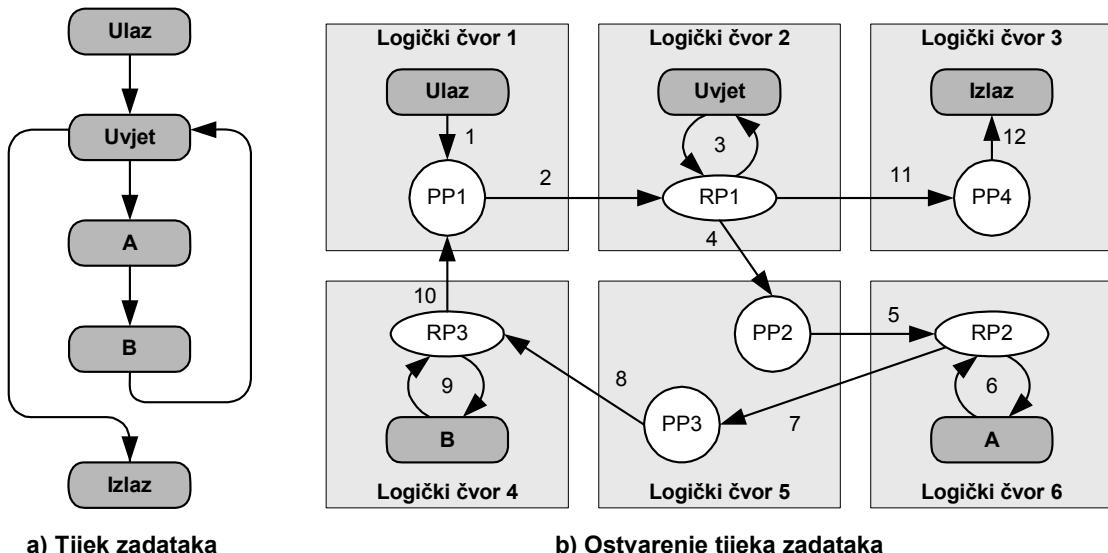
Sustav za povezivanje usluga ostvaruje potporu oblikovanju i izvođenju složenih raspodijeljenih aplikacija nastalih kompozijom aplikacijskih usluga. Postoje dva pristupa u ostvarenju kompozicije usluga: orkestracija i koreografija. Orkestracija ostvaruje poslovni proces koji se nadgleda i upravlja iz centralnog mjesta. U slučaju koreografije ne postoji centralno mjesto upravljanja poslovnim procesom. *Sustav za povezivanje usluga* kombinira navedena dva pristupa u ostvarivanju kompozicije aplikacijskih usluga.

Oblikovanje raspodijeljene aplikacije zasniva se na odvajanju aplikacijske i koordinacijske logike pa se tako raspodijeljena aplikacija oblikuje kao skup aplikacijskih usluga, raspodijeljenih programa i koordinacijskih usluga. Aplikacijske usluga ostvaruju specifičnu aplikacijsku logiku. Raspodijeljeni programi ostvaruju kompoziciju aplikacijskih usluga. Kompozicija se zasniva na definiranju toka poruka između aplikacijskih usluga. Koordinacijske usluge ostvaruju komunikacijske i sinkronizacijske mehanizme. Razvijene su tri koordinacijske usluge: semafor, poštanski pretinac i usmjernik događaja. Usluga semafor koristi se za sinkronizaciju i međusobno isključivanje raspodijeljenih programa. Usluga poštanski pretinac koristi se za razmjenu poruka između raspodijeljenih programa. Ova usluga omogućuje asinkronu komunikaciju između raspodijeljenih programa. Usluga usmjernik događaja koristi se za komunikaciju zasnovanu na događajima.

Slika 5-3a prikazuje primjer tijeka zadatka (engl. workflow) raspodijeljene aplikacije koja se sastoji od pet aplikacijskih usluga: *Ulaz*, *Uvjet*, *A*, *B* i *Izlaz*. Prikazani tijek zadatka je primjer programskog uzorka petlje. Usluga *Ulaz* prikuplja korisničke podatke i šalje ih usluzi *Uvjet* koja provjerava određeni uvjet nad ulaznim podacima. Ako uvjet nije ispunjen šalje se prikidan odgovor usluzi *Izlaz* koja obavještava korisnika o završetku procesa. Ako je uvjet

ispunjen, onda se poziva najprije usluga *A* pa nakon nje usluga *B*. Rezultat izvođenja usluge *B* prosljeđuje se usluzi *Uvjet* koja ponovno provjerava određeni uvjet nad ulaznim podacima.

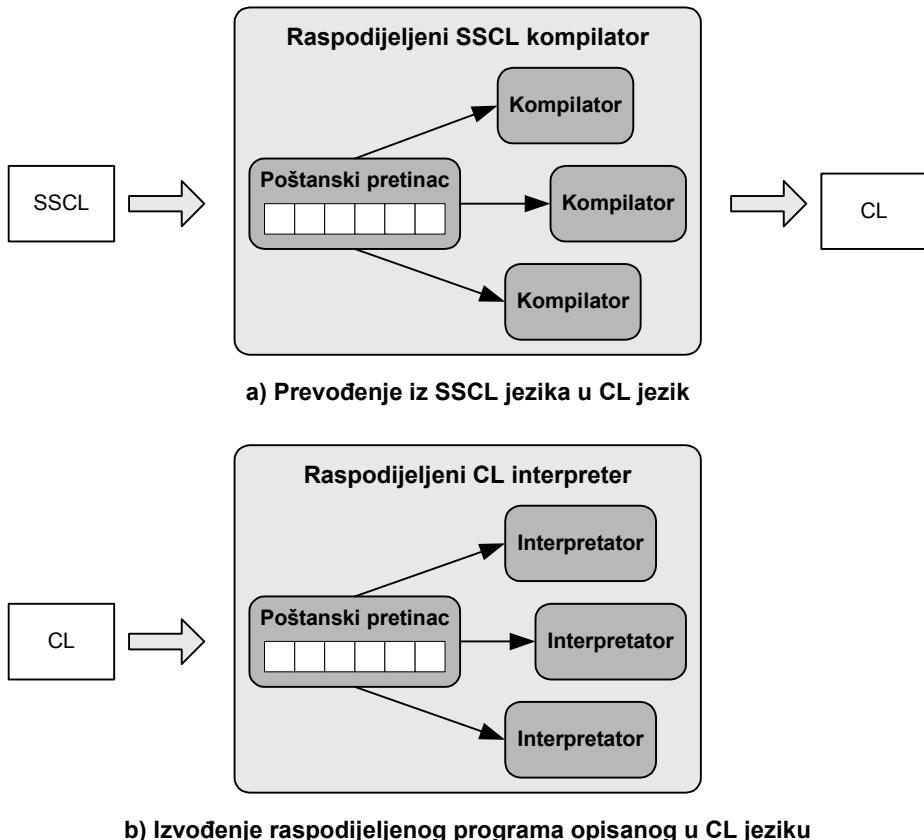
Slika 5-3b prikazuje ostvarenje opisanog tijeka zadataka u *Sustavu za povezivanje usluga*. Za ostvarenje tijeka zadataka potrebne su četiri usluge poštanskog pretinca (*PP1*, *PP2*, *PP3* i *PP4*) i tri raspodijeljena programa (*RP1*, *RP2* i *RP3*). Poštanski pretinci se koriste za asinkronu komunikaciju između raspodijeljenih programa. Usluga *Ulaz* šalje ulazne podatke u poštanski pretinac *PP1* (1), odakle ih uzima raspodijeljeni program *RP1* (2) i šalje na provjeru usluzi *Uvjet* (3). Ako je uvjet ispunjen *RP1* šalje podatke u poštanski pretinac *PP2* (4), odakle ih uzima raspodijeljeni program *RP2* (5). *RP2* dobivene podatke šalje na obradu usluzi *A* (6), nakon čega se rezultat obrade šalje u poštanski pretinac *PP3* (7), odakle ga uzima raspodijeljeni program *RP3* (8) koji ga prosljeđuje na obradu usluzi *B* (9). Rezultat obrade usluge *B*, *RP3* šalje u *PP1* (10), odakle ga ponovno čita *RP1* (2). Ako uvjet nije zadovoljen *RP1* šalje poruku o rezultatu obrade poštanskom pretincu *PP4* (11), odakle ga uzima usluga *Izlaz* i prikazuje korisniku (12).



Slika 5-3: Primjer raspodijeljene aplikacije

Za oblikovanje raspodijeljenih aplikacija definirana su dva programska jezika: *Jednostavan jezik za kompoziciju usluga* (engl. Simple Service Composition Language, SSCL) i *Jezik za koordinaciju i natjecanje usluga* (engl. Coopetition Language, CL). Ovi jezici se koriste za ostvarenje raspodijeljenih programa. SSCL jezik omogućuje opisivanje međudjelovanja usluga, a razvijen je sa ciljem da krajnjem korisniku omogući definiranje kompozicije usluga. CL jezik također služi za opisivanje međudjelovanja usluga, ali na nižoj razini u odnosu na SSCL jezik. Ovaj jezik se zasniva na kombinaciji BPEL4WS i WSDL jezika. Opis relativno

jednostavnog procesa zahtijeva više tisuća linija CL kôda pa je stoga CL jezik neprimjeren za uporabu krajnjem korisniku.



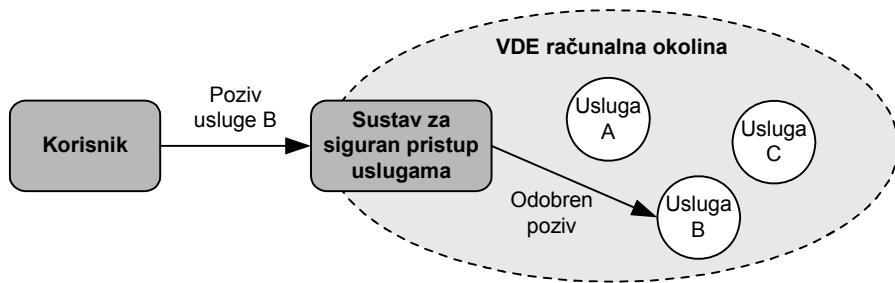
Slika 5-4: Proces prevodenja i izvođenja raspodijeljenog programa

Za izvođenje raspodijeljenih programa opisanih ovim jezicima potrebno ih je prevesti u izvršni oblik. Slika 5-4 prikazuje proces prevodenja i izvođenja raspodijeljenog programa. Prevodenje se izvodi u dva koraka. U prvom koraku se prevodi opis raspodijeljenog programa iz SSCL jezika u CL jezik (slika 5-4a). Prevodenje ostvaruje *Raspodijeljeni SSCL kompilator* koji se sastoji od *Poštanskog pretinca* i skupa *SSCL kompilatora*. *SSCL kompilatori* se registriraju u *Poštanski pretinac*. Korisnik šalje raspodijeljeni program napisan u SSCL jeziku u *Poštanski pretinac*, odakle ga uzimaju *SSCL kompilatori* ovisno o redoslijedu kojim su se registrirali. Rezultat prevodenja je raspodijeljeni program u *CL* jeziku. U drugom koraku se raspodijeljeni program preveden u *CL* jezik prenosi u *Raspodijeljeni CL interpretator* na izvođenje (slika 5-4b). *Raspodijeljeni CL interpretator* sastoji se od *Poštanskog pretinca* i skupa *CL interpretatora* koji se registriraju na *Poštanski pretinac*. Ovisno o redoslijedu registracije *CL interpretatori* uzimaju raspodijeljene programe opisane u *CL* jeziku iz *Poštanskog pretinca* i izvode ih. Ovakva arhitektura infrastrukture za izvođenje kompozicije usluga omogućuje decentralizirano izvođenje raspodijeljene aplikacije. Mjesto izvođenja raspodijeljenog programa utvrđuje se tek

za vrijeme izvođenja. Detalji o *Sustavu za povezivanje usluga* opisani su u radovima [94], [95], [97], [99] i [101].

5.1.3 Sustav za siguran pristup uslugama

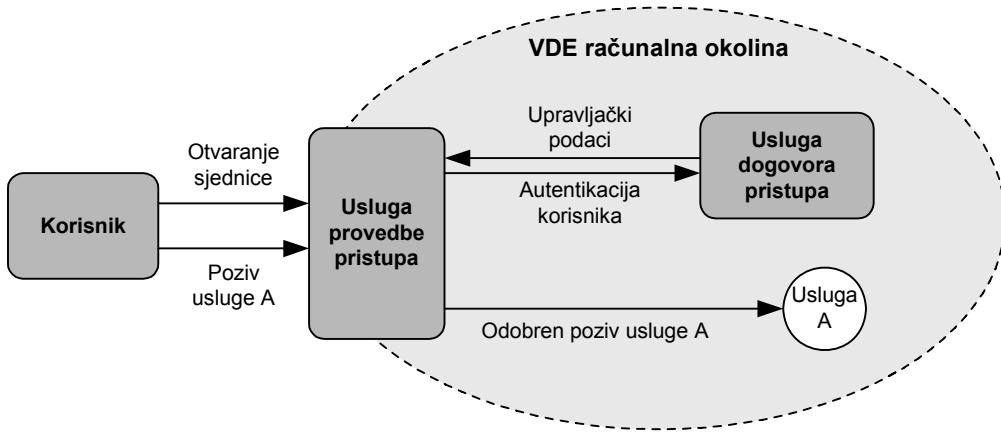
Sustav za siguran pristup uslugama ostvaruje potporu sigurnom pristupu uslugama. Pristup do unutarnjih usluga ostvaruje se putem specifičnih *pristupnih točaka* (engl. Points-of-Presence, PoP) koje su istovremeno sudionici *VDE* računalne okoline i javne mreže kao što je Internet.



Slika 5-5: Smještaj *Sustava za siguran pristup uslugama* unutar *VDE* računalne okoline

Slika 5-5 prikazuje smještaj *Sustava za siguran pristup uslugama* unutar *VDE* računalne okoline te njegovu osnovnu funkcionalnost. Raspodijeljena infrastruktura *Sustava za siguran pristup uslugama* smještena je na pristupnim točkama i čvorovima *VDE* računalne okoline. Korisnici prilikom pristupa uslugama *VDE* računalne okoline šalju zahtjeve putem *Sustava za siguran pristup uslugama* koji odobrava ili zabranjuje pristup usluzi. *Sustav za siguran pristup uslugama* donosi odluku o propuštanju ili odbacivanju korisničkog zahtjeva na osnovi identiteta vanjskog korisnika i njegovih prava pristupa usluzi. Osim odobravanja ili zabrane, provodi se i praćenje pristupa unutarnjoj usluzi, odnosno spremaju se značajne informacije iz korisničkih zahtjeva za buduću analizu.

Slika 5-6 prikazuje osnovnu arhitekturu *Sustava za siguran pristup uslugama*. Sustav se sastoji od *Usluge provedbe pristupa* i *Usluge dogovora pristupa*. Ovakva arhitektura razdvaja logiku odlučivanja od provedbene logike na analogan način kao što se u stvarnom svijetu razdvaja zakonodavnu vlast (odлука) od izvršne vlasti (provedba). *Usluga provedbe pristupa* se nalazi na pristupnoj točki *VDE* računalne okoline i provodi odluku pristupa usluzi, odnosno prosljeđuje korisnički zahtjev do usluge ili ga odbacuje. *Usluga dogovora pristupa* se nalazi unutar *VDE* računalne okoline i donosi odluke o pristupu usluzi.



Slika 5-6: Arhitektura *Sustava za siguran pristup uslugama*

Korisnici i usluge najprije se registriraju u sustav. Korisnici tijekom registracije navode informacije o svom identitetu, dok usluge tijekom registracije navode prava pristupa pojedinim metodama svog sučelja za pojedine korisnike. Podaci dobiveni registracijom spremaju se unutar *Usluge dogovora pristupa*. Prilikom korištenja usluge vanjski korisnik najprije započinje sjednicu tijekom koje koristi određenu uslugu. Tijekom otvaranja sjednice korisnika se autenticira, odnosno provjerava se njegov identitet. Autentikaciju korisnika provodi *Usluga dogovora pristupa* koja kao rezultat autentikacije *Usluzi provedbe pristupa* vraća upravljačke podatke. Upravljački podaci sadrže korisnikova prava pristupa pojedinim unutarnjim uslugama. Korisnik poziva unutarnje usluge šaljući zahtjeve putem *Usluge provedbe pristupa*. Za svaki korisnički zahtjev provodi se autorizacija u kojoj *Usluga provedbe pristupa* na osnovi korisnikovih upravljačkih podataka odobrava ili blokira korisnički zahtjev. *Usluga provedbe pristupa* također u svrhu praćenja prometa bilježi značajne informacije iz korisničkog zahtjeva. Detaljan opis *Sustava za siguran pristup uslugama* nalazi se u radovima [96] i [98].

6 Sustav za postavljanje i otkrivanje usluga

Sustav za postavljanje i otkrivanje usluga pruža potporu automatiziranom postavljanju i otkrivanju usluga u *VDE* računalnoj okolini. Aplikacijske usluge potrebno je postaviti na čvorove *VDE* računalne okoline. Potreba za automatiziranim postavljanjem usluga proizlazi iz činjenice da je ručno postavljanje usluga vremenski zahtjevan i greškama podložan proces. Postupkom postavljanja usluga mijenja se razmještaj usluga unutar *VDE* računalne okoline. Uvid u trenutni razmještaj usluga *VDE* računalne okoline korisnici ostvaruju postupkom otkrivanja usluga.

Sustav je ostvaren kao raspodijeljena aplikacija koja se sastoji od tri vrste usluga: usluga *Katalog*, usluga *Skladište* i usluga *Postavljač*. Usluga *Katalog* pruža mehanizme za otkrivanje usluga, dok usluge *Skladište* i *Postavljač* u suradnji s uslugom *Katalog* pružaju mehanizme za postavljanje usluga.

U ovom poglavlju opisuju se konceptualna arhitektura *Sustava za postavljanje i otkrivanje usluga*. U opisu arhitekture naglašavaju se samo važni funkcionalni dijelovi, bez dubljeg pregleda u samo ostvarenje. Detaljan pregled ostvarenja sustava s uvidom u korištene tehnologije daje se u poglavlju 7.

6.1 Funkcionalnosti Sustava za postavljanje i otkrivanje usluga

Sustav za postavljanje i otkrivanje usluga razvijen je za potrebe *VDE* računalne okoline. Osnovne funkcionalnosti sustava su *otkrivanje*, *spremanje*, *postavljanje* i *konfiguriranje* usluga.

6.1.1 Otkrivanje usluga

Otkrivanje usluga je proces tijekom kojeg sudionik računalne okoline saznaće informacije o nekoj usluzi računalne okoline. Otkrivanje usluga jedan je od osnovnih procesa u arhitekturama zasnovanim na uslugama. Za održavanje znanja o stanju računalne okoline brine se zasebni podsustav - *katalog usluga*. Pretraživanjem kataloga usluga sudionici računalne okoline dolaze do podataka o uslugama. Katalog usluga je i sam ostvaren kao usluga, ali u odnosu na ostale usluge njegova lokacija i programsko sučelje su općepoznati svim sudionicima računalne okoline.

Aplikacijske usluge *VDE* računalne okoline su premjestive, odnosno mogu se postavljati i uklanjati sa čvorova logičke mreže. Strukture podataka standarnih kataloga usluga prilagođene su spremanju podataka o statičnim uslugama, pa stoga standardni katalozi usluga nisu prikladni

za *VDE* računalnu okolinu. Katalog usluga *VDE* računalne okoline prilagođen je dinamičkoj prirodi aplikacijskih usluga tako da osim informacija o postavljenim uslugama održava informacije i o spremištima izvršnog kôda. Nadalje, katalog usluga *VDE* računalne okoline prilagođen je specifičnom načinu adresiranja postavljenih usluga unutar *Prividne logičke mreže*. Uz navedeno, katalog usluga omogućava spremanje specifičnih informacija o *VDE* računalnoj okolini, poput popisa logičkih čvorova.

6.1.2 Spremanje usluga

Spremanje usluge je postupak tijekom kojeg administrator računalne okoline sprema izvršni kôd aplikacijske usluge unutar *Sustava za postavljanje i otkrivanje usluga*. Usluge se spremaju na posebne čvorove računalne okoline koji ostvaruju ulogu skladišta aplikacijskih usluga. Spremljena usluga je spremna za prijenos i postavljanje na čvorove računalne okoline. Usluge se spremaju u neaktivnom obliku koji se naziva *instalacijski paket*. Instalacijski paket omogućava automatizaciju postupka postavljanja usluge na čvorove računalne okoline. Sa ciljem smanjenja diskovnog prostora instalacijskog paketa u skladištu usluga, koriste se algoritmi sažimanja. Također, postupkom sažimanja smanjuje se mrežni promet prilikom prijenosa instalacijskog paketa.

Uslugu je istovremeno moguće spremiti u više skladišta. Spremanjem usluge u više računala osigurava se razmjeran rast sustava. Na primjer, korisnici prilikom dohvata instalacijskih paketa mogu odabrati geografski najbliže skladište te time osigurati kraće vrijeme dohvata instalacijskih paketa. Također, spremanjem usluge na više računala smanjuje se opterećenje skladišta, jer neće svi korisnici dohvaćati uslugu uvijek iz istog skladišta.

Suprotan postupak spremanju usluga je uklanjanje usluga iz skladišta usluga. Postupkom uklanjanja se instalacijski paket briše iz skladišta usluga.

6.1.3 Postavljanje usluga

Postavljanje usluge sastoji se od prijenosa instalacijskog paketa iz skladišta usluga na čvor računalne okoline te njegove instalacije. Postupak postavljanja usluge pokreće administrator računalne okoline ili neka druga usluga. Usluga nakon postavljanja postaje spremna za izvođenje, odnosno postaje spremna za obrađivanje korisničkih zahtjeva. Uslugu je istovremeno moguće postaviti na više čvorova računalne okoline. Postupak postavljanja usluge opisuje se posebno definiranom instalacijskom skriptom.

Postupak postavljanja ostvaren je u obliku transakcije. Transakcijski model postavljanja usluga osigurava da se postupak obavi u cijelosti ili nikako. U slučaju pogreške u procesu

postavljanja, čvor računalne okoline se vraća u stanje u kojem je bio prije početka postavljanja usluge.

Postupak postavljanja usluga prilagođen je specifičnostima aplikacijskih usluga koje se pojavljuju u *VDE* računalnoj okolini. Također, unutar *Sustava za postavljanje i otkrivanje usluga* primjenjene su sigurnosne metode za ograničavanje sposobnosti potencijalno zlonamjernim usluga. Suprotan proces postavljanju usluge je uklanjanje usluge s čvora računalne okoline.

6.1.4 Konfiguriranje usluga

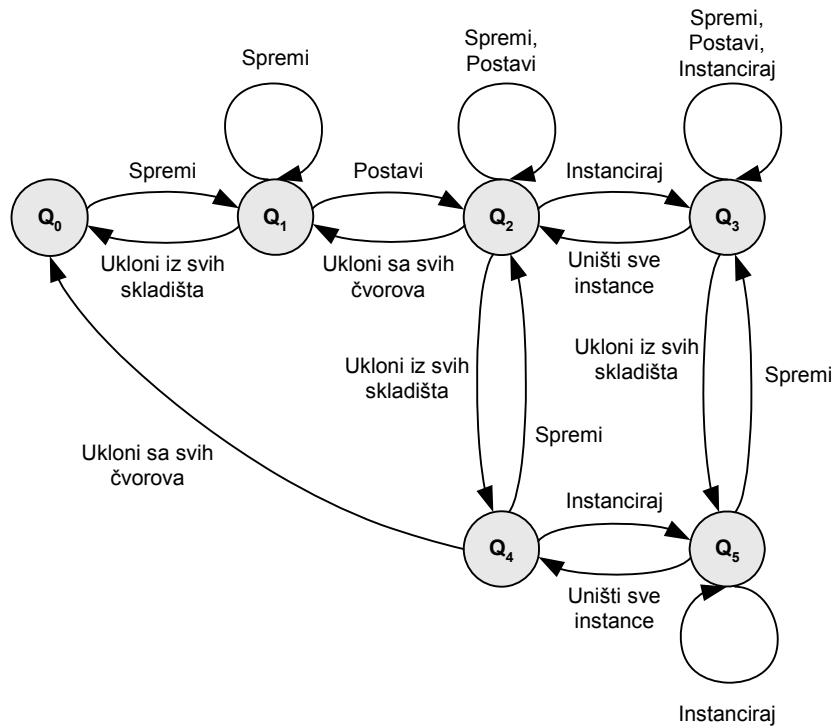
Konfiguriranje usluga je postupak podešavanja radnih parametara postavljene aplikacijske usluge. Primjer radnog parametra usluge je lokacija kataloga usluga. Lokacija kataloga usluga je nužna informacija za otkrivanje ostalih usluga računalne okoline. Usluge na različite načine upravljaju konfiguracijskim podacima. Na primjer, konfiguracijski podaci spremaju se u datoteku proizvoljnog imena i proizvoljnog formata. S ciljem ostvarenja uspješnog konfiguriranja definiran je uniforman način zapisa konfiguracijskih podataka za aplikacijske usluge u *VDE* računalnom okruženju. Od aplikacijskih usluga se zahtjeva da slijede definirani format konfiguracijskih podataka.

6.2 Životni ciklus aplikacijskih usluga

Aplikacijska usluga tijekom svog postojanja unutar *VDE* računalne okoline poprimaju više oblika. Ako je aplikacijska usluga spremljena u barem jednom skladištu usluga kao instalacijski paket, onda je usluga u *spremljenom* obliku. Ako je usluga postavljena na barem jedan čvor računalne okline, onda je usluga u *postavljenom* obliku. Ako je aplikacijska usluga pokrenuta i održava instance na barem jednom čvoru, onda je usluga u *instanciranom* obliku. Ne mora svaka usluga podržavati instance. Primjer usluge koja podržava instance je usluga reda poruka. Osnovne funkcionalnosti ove usluge su spremanje poruka u internu strukturu reda poruka, uzimanje poruka iz reda te njihova predaja korisniku. Instance ove usluge su interni redovi poruka, pri čemu je svaka instanca određena identifikatorom. Korisnik prilikom slanja poruke u red poruka mora navesti identifikator instance u koju želi spremiti poruku. Za aplikacijske usluge *VDE* računalne okoline je karakteristično da se pojavljuju u više oblika istovremeno. Na primjer, usluga u pojedinom trenutku može istovremeno biti spremljena u skladištu usluga i postavljena na nekom od čvorova računalne okoline.

Funkcionalnosti *Sustava za postavljanje i otkrivanje usluga* pružaju potporu pretvaranju aplikacijske usluge iz jednog oblika u drugi. Također, uloga *Sustava za postavljanje i otkrivanje usluga* je održavanje informacija o aktivnim oblicima aplikacijske usluge. Formalna

definicija životnog ciklusa aplikacijske usluge definira utjecaj sustava na životni ciklus usluge. Slika 6-1 prikazuje formalnu definiciju životnog ciklusa aplikacijske usluge izraženu pomoću determinističkog konačnog automata (DKA). Stanje automata identificira u kojim sve oblicima usluga može biti istovremeno. Na primjer, stanje Q_5 označava da je usluga postavljena na barem jednom čvoru i da barem jedna od tih postavljenih usluga održava instance.



- Q_0 - Usluga nije prisutna u VDE računalnoj okolini
- Q_1 - Usluga je u spremjenom obliku
- Q_2 - Usluga je u spremjenom i postavljenom obliku
- Q_3 - Usluga je u spremjenom, postavljenom i instanciranom obliku
- Q_4 - Usluga je u postavljenom obliku
- Q_5 - Usluga je u postavljenom i instanciranom obliku

Slika 6-1: Formalni opis životnog ciklusa usluge

Prijelazi automata predstavljaju moguće akcije nad oblicima usluge. Akcija *Spremi* predstavlja postupak spremanja usluge u skladište usluga. Akcija *Postavi* predstavlja postupak postavljanja spremljene usluge na čvor računalne okoline. Akcija *Instanciraj* predstavlja postupak stvaranja instance postavljenе usluge. Svaka od navedenih akcija posjeduje svoju suprotnu akciju. Rezultat akcije je pojava novih oblika usluge. Na primjer, rezultat akcije *Spremi* je pojava *spremljenog* oblika usluge. Informacije o rezultatima akcija spremaju se u *Sustav za postavljanje i otkrivanje usluga*. Na primjer, *Sustav za postavljanje i otkrivanje usluga* omogućuje korisniku da u svakom trenutku sazna u kojim je sve skladištima usluga spremljena.

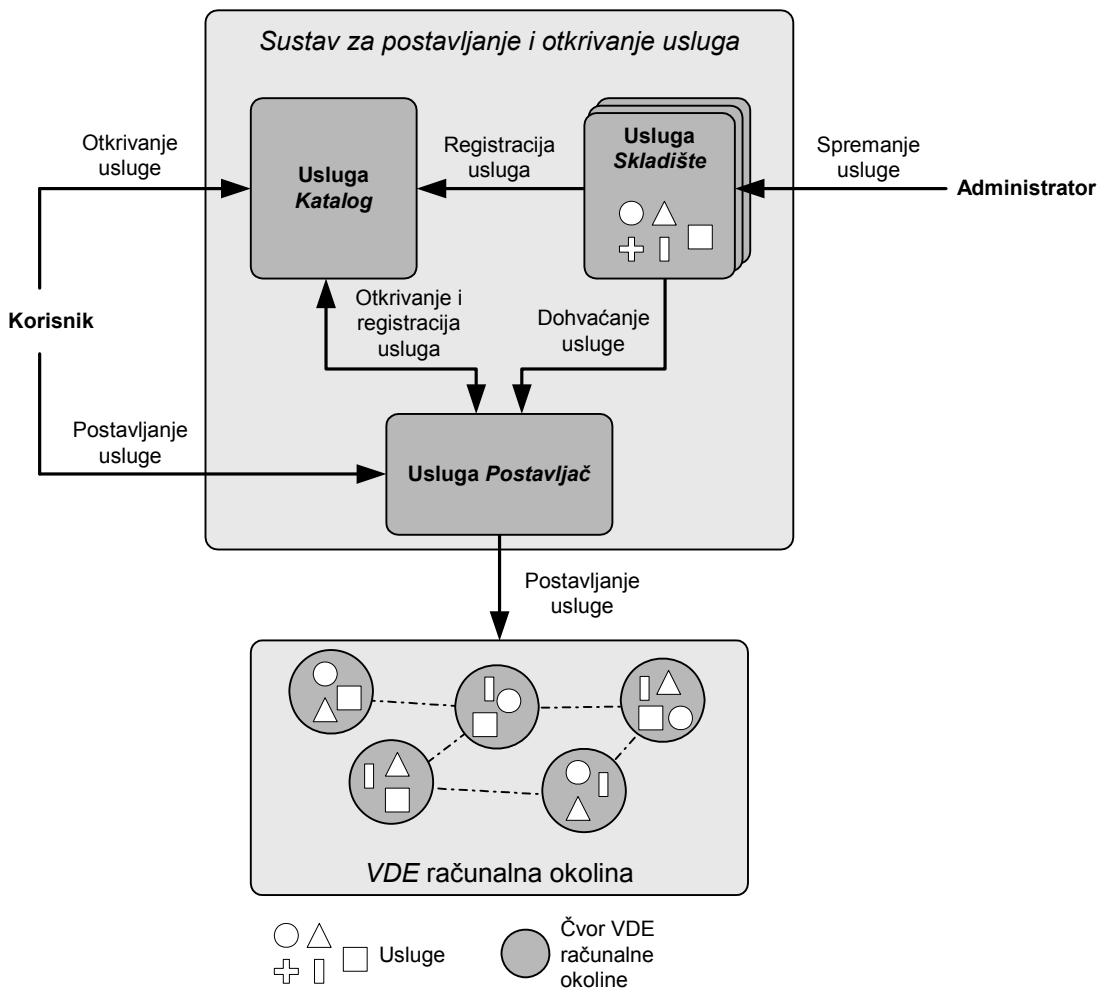
Akcije *Spremi*, *Postavi*, *Ukloni iz svih skladišta* i *Ukloni sa svih čvorova* su direktno podržane od *Sustava za postavljanje i otkrivanje usluga*. Akcije *Instanciraj* i *Uništi sve instance* specifične su akcije aplikacijskih usluga. Aplikacijske usluge ne moraju nužno ostvariti akcije *Instanciraj* i *Uništi sve instance*. Rezultat akcije *Instanciraj* je stvaranje instance usluge. Ako usluga podržava instance, onda je dužna informacije o rezultatu akcija *Instanciraj* i *Uništi sve instance* objaviti *Sustavu za postavljanje i otkrivanje usluga*. Akcija *Instanciraj* zahtijeva postojanje aplikacijske usluge na čvoru *VDE* računalne okoline. Usluga prestaje postojati u *VDE* računalnoj okolini kada se uklone njeni instalacijski paketi iz svih skladišta te kada se usluga ukloni sa svih čvorova na kojima je postavljena.

6.3 Arhitektura sustava za postavljanje i otkrivanje usluga

Arhitektura *Sustava za postavljanje i otkrivanje usluga* prikazana je na slici 6-2. Prikazana arhitektura zasniva se na principima arhitektura zasnovanih na uslugama. Svi elementi *Sustava za postavljanje i otkrivanje usluga* su autonomne i slabo povezane usluge. Arhitektura sustava se sastoji od usluga *Katalog*, *Skladište* i *Postavljač*.

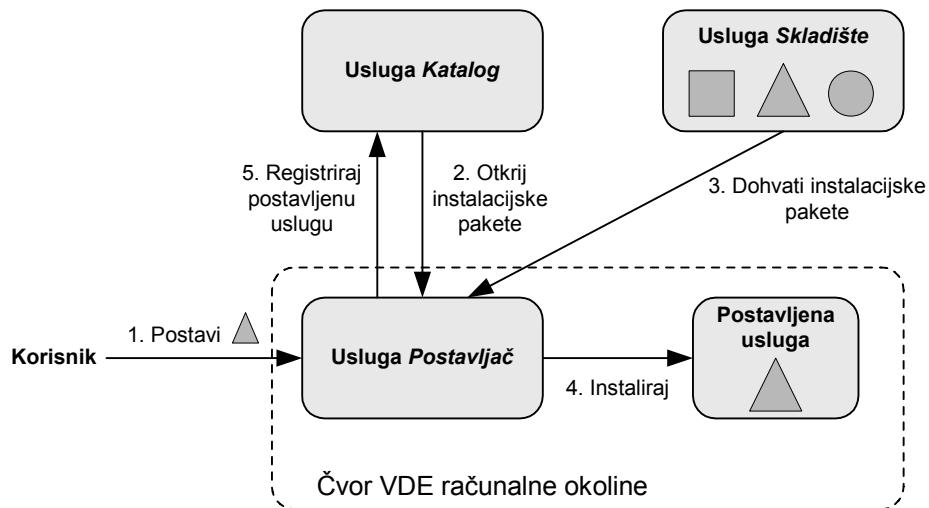
Usluga *Skladište* ostvaruje funkcionalnosti skladišta instalacijskih paketa usluga. U sustavu može postojati više preslika usluge *Skladište*, pri čemu one ne moraju sadržavati instalacijske pakete istog skupa aplikacijskih usluga. Raspodjelom usluge *Skladište* osigurava se razmjeran rast sustava, a također se uklanja jedinstvena točka pogreške. Ako prestane s radom jedna usluga *Skladište*, onda je moguće instalacijske pakete aplikacijske usluge pronaći u nekoj drugoj preslici usluge *Skladište*.

Usluga *Katalog* je informacijska usluga *Sustava za postavljanje i otkrivanje usluga*. *Katalog* proširuje osnovnu ulogu kataloga usluga u arhitekturama zasnovanim na uslugama, gdje se katalog koristi isključivo za otkrivanje postavljenih usluga. *Katalog* se osim otkrivanja lokacija postavljenih usluga koristi i za otkrivanje ostalih oblika aplikacijskih usluga, poput spremljenih ili instanciranih usluga. Osim toga, *Katalog* održava i druge informacije bitne *Sustavu za postavljanje i otkrivanje usluga* te *VDE* računalnoj okolini, poput popisa čvorova računalne okoline ili lokacija svih usluga *Skladište*.



Slika 6-2: Arhitektura *Sustava za postavljanje i otkrivanje usluga*

Usluga *Postavljač* zadužena je za automatizirano postavljanje i konfiguiriranje aplikacijskih usluga. Na svakom čvoru *VDE* računalne okoline nalazi se preslika usluge *Postavljač*. Slika 6-3 prikazuje proces postavljanja usluge na razini međudjelovanja usluga *Sustava za postavljanje i otkrivanje usluga*. Proces postavljanja započinje korisnički zahtjev za postavljanjem usluge (1). *Postavljač* najprije putem *Kataloga* saznaće lokaciju preslike usluge *Skladište* u kojoj je spremljena aplikacijska usluga (2). Nakon toga *Postavljač* dohvata iz *Skladišta* instalacijske pakete aplikacijske usluge (3), odpakirava ih, te izvodi instalaciju usluge na lokalni čvor (4). Za svaku aplikacijsku uslugu provodi se specifičan postupak instalacije. Postupak instalacije aplikacijske usluge izvodi se na osnovi specifične instalacijske skripte koja je sastavni dio instalacijskog paketa. Ako je postupak instalacije uspješno završen, onda usluga *Postavljač* obavještava uslugu *Katalog* o novo postavljenoj usluzi (5).



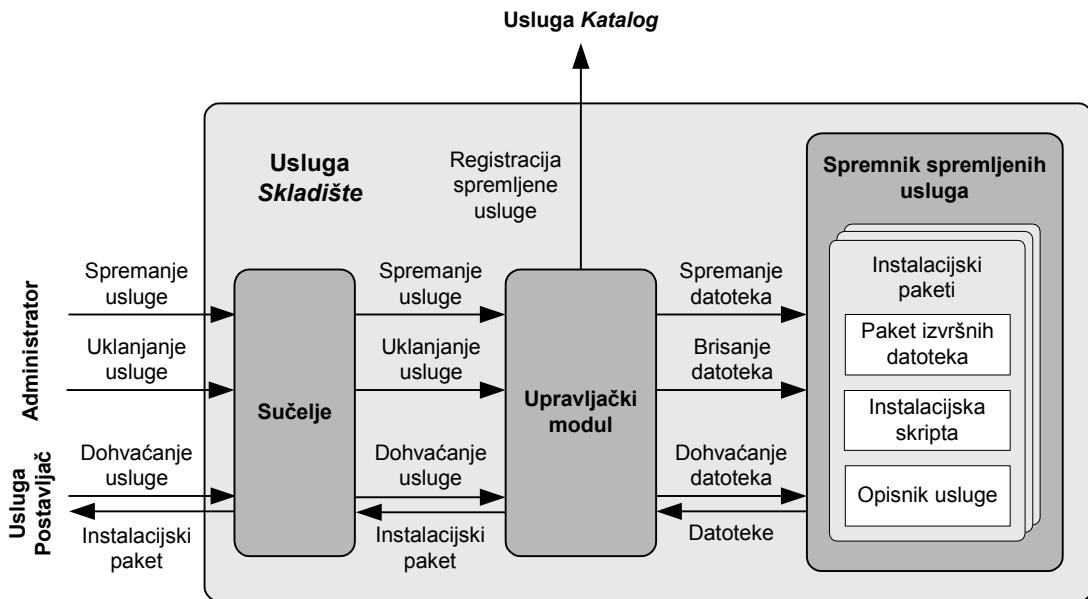
Slika 6-3: Proces postavljanja usluge

Predložena arhitektura automatizira postupak postavljanja aplikacijskih usluga na čvorove *VDE* računalne okoline te pruža potporu otkrivanju različitih oblika aplikacijskih usluga. Arhitektura sustava se zasniva na uslugama što olakšava održavanje programskog ostvarenja usluge. Na primjer, ako se prilikom izgradnje nove inačice usluge *Katalog* zadrži staro programsko sučelje, onda ostale usluge *Sustava za postavljanje i otkrivanje usluga* nije potrebno mijenjati, jer se nije promijenio način komunikacije s uslugom *Katalog*.

6.3.1 Konceptualna arhitektura usluge *Skladište*

Usluga *Skladište* je spremište spremljenih aplikacijskih usluga. Slika 6-4 prikazuje konceptualnu arhitekturu usluge *Skladište*. Konceptualna arhitektura usluge *Skladište* sastoji se od *Sučelja*, *Upravljačkog modula* i *Spremnika spremljenih usluga*. Usluga *Skladište* ostvaruje tri funkcionalnosti: *spremanje usluge*, *dohvaćanje usluge* i *uklanjanje usluge*.

Postupak spremanja usluge započinje administrator slanjem instalacijskog paketa na *Sučelje* usluge *Skladište*. *Sučelje* usluge prima instalacijski paket i prosljeđuje ga *Upravljačkom modulu* koji ga sprema u *Spremnik spremljenih usluga*. *Upravljački modul* nadgleda i provodi aplikacijsku logiku usluge *Skladište*. *Spremnik spremljenih usluga* sprema datoteke instalacijskog paketa na diskovni prostor te ih registrira u vlastitu listu spremljenih instalacijskih paketa. Ako je postupak spremanja datoteka instalacijskog paketa uspješno završio, *Upravljački modul* šalje usluzi *Katalog* obavijest o spremljenoj aplikacijskoj usluzi.



Slika 6-4: Konceptualna arhitektura usluge *Skladište*

Postupak dohvaćanja usluge obično započinje usluga *Postavljac* tijekom postupka postavljanja usluge na čvor računalne okoline. Nakon što *Sučelje* primi zahtjev za instalacijskim paketom, proslijedi ga *Upravljačkom modulu* koji zatraži datoteke instalacijskog paketa iz *Spremnika spremljenih usluga*. *Spremnik spremljenih usluga* pronađe u svojoj listi instalacijskih paketa traženi paket, učita ga sa diskovnog prostora i proslijedi *Upravljačkom modulu*. *Upravljački modul* instalacijski paket proslijedi do *Sučelja* koje ga šalje usluzi *Postavljac*.

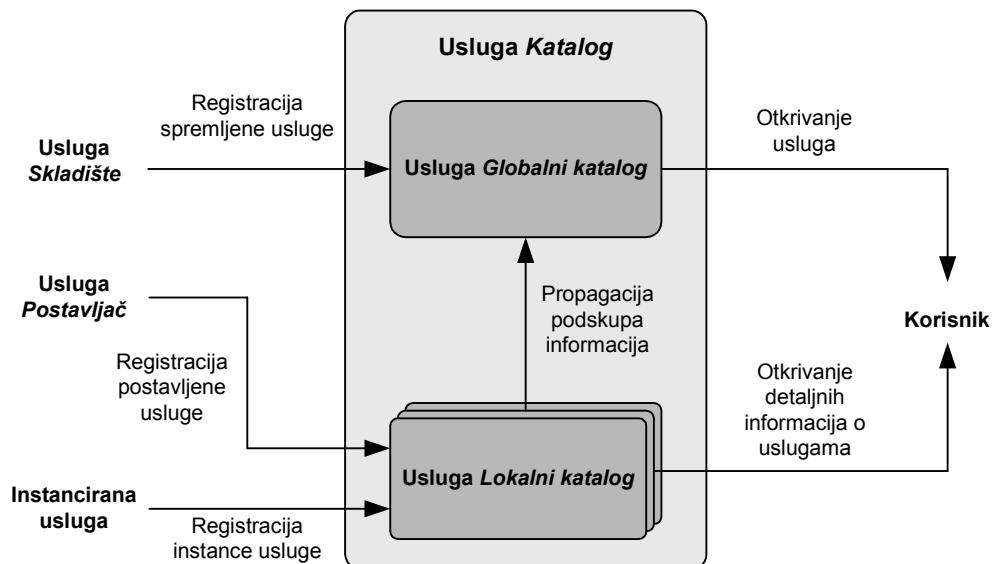
Uklanjanje usluge započinje administrator *VDE* računalne okoline. Rezultat uklanjanja usluge je brisanje instalacijskih paketa iz usluge *Skladište*, odnosno iz *Spremnika spremljenih usluga*. Ako je brisanje instalacijskog paketa uspješno završilo, *Upravljački modul* obavještava uslugu *Katalog* o uklonjenoj aplikacijskoj usluzi.

Aplikacijska usluga se sprema u obliku instalacijskog paketa. Instalacijski paket se sastoji od tri komponente: *paketa datoteka*, *instalacijske skripte* i *opisnika usluge*. Paket datoteka je sažeta datoteka koja sadrži izvršne datoteke aplikacijske usluge. Budući da se u instalacijski paket spremaju već prevedene izvršne datoteke, unaprijed je određena računalna platforma na kojoj se može izvoditi aplikacijska usluga. Osim izvršnih datoteka, paket datoteka sadrži i ostale datoteke, poput konfiguracijskih datoteka ili raznih podatkovnih datoteka. Instalacijska skripta je tekstualna datoteka koja sadrži opis instalacijskog postupka. Struktura i konstrukti instalacijske skripte prilagođeni su potrebama *VDE* računalne okoline. Prilikom definiranja konstrukata instalacijske skripte poseban je naglasak stavljen na nezavisnost skripte s obzirom na operacijski sustav i računalnu platformu. Detalji o strukturi instalacijske skripte se mogu pronaći u poglavljju

7.3.2. Opisnik usluge je tekstualna datoteka koja definira programsko sučelje aplikacijske usluge. U opisniku se definira struktura valjanih ulaznih i izlaznih poruka aplikacijske usluge. Opisnik usluge je važan korisnicima aplikacijske usluge za uspostavu komunikacije s uslugom.

6.3.2 Konceptualna arhitektura usluge *Katalog*

Usluga *Katalog* je složena usluga koja pruža potporu održavanju i otkrivanju informacija o aplikacijskim uslugama. *Katalog* je oblikovan kao kompozicija usluga, čime je ostvareno raspodijeljeno upravljanje informacijama o aplikacijskim uslugama. Osim raspodijeljenog upravljanja informacijama, raspodijeljeno je i njihova fizička lokacija. Raspodjelom informacija na više mjesta u *VDE* računalnoj okolini ostvaruje se svojstvo razmjernog rasta usluge *Katalog*, odnosno radna svojstva sustava ne ovise o količini spremiljenih podataka.



Slika 6-5: Konceptualna arhitektura usluge *Katalog*

Slika 6-5 prikazuje konceptualnu arhitekturu usluge *Katalog*. Arhitektura *Kataloga* slična je arhitekturi *MDS* informacijskog sustava [70]. Usluga *Katalog* oblikovana je u dvorazinsku hijerarhijsku strukturu. Na dnu hijerarhije su usluge *Lokalni katalog*, koje se nalaze na svakom čvoru *VDE* računalne okoline. Na vrhu hijerarhije je usluga *Globalni katalog* koja se nalazi na posebno odabranom čvoru. Lokacija usluge *Globalni katalog* poznata je svim sudionicima *VDE* računalne okoline.

Usluga *Lokalni katalog* održava detaljne informacije o aplikacijskim uslugama postavljenim na lokalnom čvoru. Primjer informacije koja se spremaju u *Lokalni katalog* je opisnik sučelja postavljene usluge. Informacije u *Lokalni katalog* upisuju usluga *Postavljac* i aplikacijske usluge smještene na lokalnom čvoru. Usluga *Postavljac* postupkom registracije

upisuje podatke o postavljenim aplikacijskim uslugama u *Lokalni katalog*, dok postavljene aplikacijske usluge upisuju informacije o stvorenim instancama. Podskup informacija iz usluge *Lokalni katalog* prosljeđuje se usluzi *Globalni katalog*. Na osnovi prikupljenih podataka iz *Lokalnih kataloga*, usluga *Globalni katalog* omogućuje otkrivanje različitih oblika aplikacijskih usluga prisutnih u *VDE* računalnoj okolini. Osnovna informacija koja se postupkom otkrivanja dobije iz *Globalnog kataloga* je lokacija aplikacijske usluge. Detaljne informacije o usluzi, poput opisnika sučelja postavljene usluge, dohvaćaju se iz *Lokalnog kataloga* koji se nalazi na istom čvoru kao i postavljena usluga. Osim *Lokalnih kataloga*, informacije u *Globalni katalog* upisuje i usluga *Skladište* prilikom postupka registracije spremiljenih aplikacijskih usluga. Uz osnovne informacije o oblicima prisutnih aplikacijskih usluga, *Globalni katalog* sadrži i specifične informacije *VDE* računalne okoline, poput popisa računalnih čvorova.

Usluga *Lokalni katalog* prosljeđuje *Globalnom katalogu* podatke koji zauzimaju mali memorijski prostor. Opisnik postavljene usluge je dokument koji ovisno o složenosti sučelja usluge može postati relativno velik pa se stoga opisnik usluge ne prosljeđuje usluzi *Globalni katalog*. Pažljivom raspodjelom podataka između usluga *Globalni katalog* i *Lokalni katalog* moguće je osigurati bolja radna svojstva cjelokupnog *Kataloga*. Sljedeći odjeljci opisuju osnovni skup podataka o aplikacijskim uslugama u *VDE* računalnoj okolini te daju funkcionalnu arhitekturu usluga *Lokalni* i *Globalni katalog*.

Osnovni podaci o aplikacijskim uslugama

Aplikacijska usluga u *VDE* računalnoj okolini se može pojaviti u spremljenom, postavljenom ili instanciranom obliku. Za aplikacijsku uslugu u spremljenom obliku nužno je održavati popis *Skladišta* koji sadrže instalacijske pakete aplikacijskih usluga. Za aplikacijsku uslugu u postavljenom obliku nužno je održavati popis čvorova na kojima je usluga postavljena. Za aplikacijsku uslugu u instanciranom obliku nužno je održavati podatke o pristupu pojedinoj instanci usluge.

Točan oblik zapisa ovih osnovnih podataka o aplikacijskim uslugama uvelike ovisi o specifičnom načinu adresiranja usluga unutar *Prividne logičke mreže*. Prilikom slanja poruke postavljenoj usluzi na čvoru *Prividne logičke mreže*, korisnik ju adresira pomoću dva podatka: *imenom čvora* i *imenom usluge*. Ime čvora na jedinstven način određuje čvor *Prividne logičke mreže*, pri čemu je to ime nezavisno od fizičke adrese računala. Ime usluge na jedinstven način određuje postavljenu uslugu.

Osnovni podaci o aplikacijskim uslugama unutar *Kataloga* grupiraju se u tri skupine: podaci o spremljenim uslugama, podaci o postavljenim uslugama i podaci o instanciranim

uslugama. Za spremjene usluge održava se tablica s uređenim parovima oblika (*ime čvora, ime usluge*), gdje *ime čvora* identificira čvor unutar *Prividne logičke mreže*, a *ime usluge* identificira uslugu. Navedeni uređeni par određuje ime čvora na kojem se nalazi usluga *Skladište*, a u kojoj je spremljena imenovana usluga. U jedno *Skladište* se može spremiti više aplikacijskih usluga, a jedna usluga se može spremiti u više *Skladišta*.

Za postavljenе usluge se održava slična tablica s uređenim parovima oblika (*ime čvora, ime usluge*), gdje *ime čvora* identificira čvor *Prividne logičke mreže* na kojem je postavljena usluga s imenom *ime usluge*. Na jedan čvor se može postaviti više različitih aplikacijskih usluga, a jedna usluga se može postaviti na više čvorova.

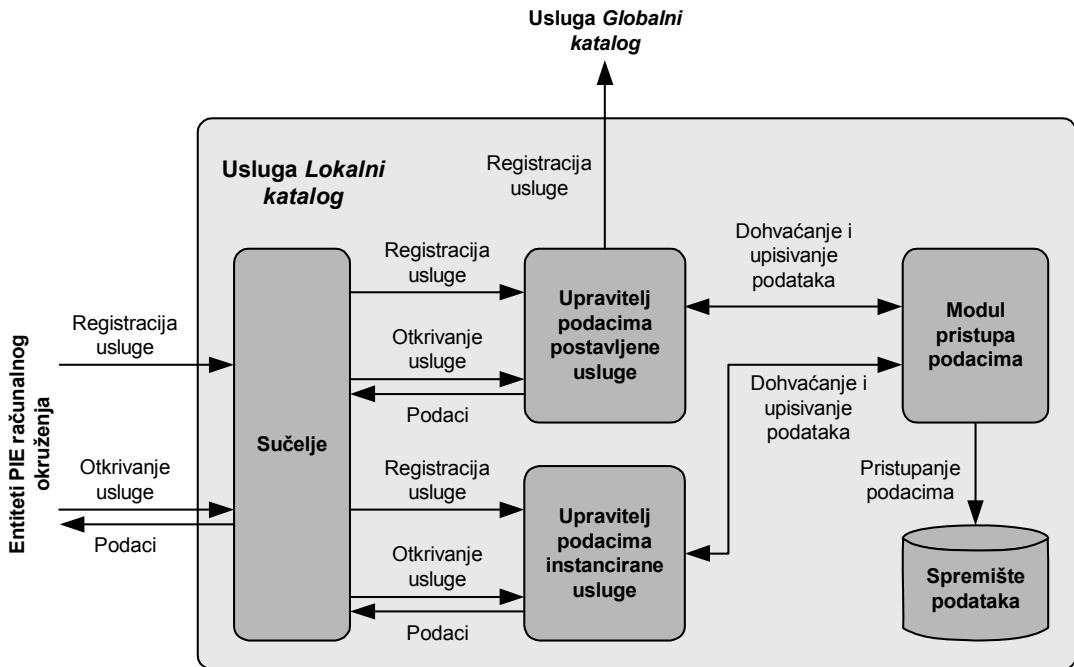
Za instancirane usluge održava se tablica s uređenim trojkama oblika (*ime čvora, ime usluge, ime instance*), gdje *ime čvora* identificira čvor mreže, *ime usluge* identificira uslugu, a *ime instance* identificira instancu usluge. Navedena uređena trojka jedinstveno adresira instancu usluge unutar *VDE* računalne okoline. Tablica instanciranih usluga je proširenje tablice postavljenih usluga, jer instanca usluge ne može postojati ako usluga nije postavljena. Na primjer, za uređenu trojku (*cvorA, uslugaN, instancaXYZ*) u tablici instanciranih usluga mora postojati uređeni par (*cvorA, uslugaN*) u tablici postavljenih usluga.

Može se pretpostaviti da spremjeni oblik aplikacijske usluge ima najduži životni vijek unutar *VDE* računalne okoline, a instancirani oblik najkraći. Instanca usluge obično se stvara za vrijeme nekog raspodijeljenog procesa te se na kraju procesa uništava. Spremljena usluga se uništava kad u potpunosti nestane potrebe za njenom uporabom, odnosno kad se proglaši zastarjelom. Iz navedenog slijedi da se tablica instanciranih usluga puno češće mijenja u odnosu na tablicu spremjenih usluga. S ciljem postizanja boljih radnih svojstava usluge *Katalog*, tablice spremjenih i postavljenih usluga spremaju se u uslugu *Globalni katalog*, a tablice instanciranih i postavljenih usluga spremaju se u usluge *Lokalni katalog*. Opisanom raspodjelom osnovnih podataka aplikacijskih usluga smanjuje se učestalost osvježavanja sadržaja usluge *Globalni katalog*.

Arhitekture usluga Lokalni i Globalni katalog

Arhitekture usluga *Lokalni* i *Globalni katalog* oblikovane su prema modelu *troslojne arhitekture*. Troslojna arhitektura se sastoji od sloja podataka, sloja aplikacijske logike i prezentacijskog sloja. Sloj podataka ostvaruje fizički pristup podacima koji su spremjeni na neki medij, poput datoteke ili baze podataka. Sloj aplikacijske logike izvodi potrebne operacije nad podacima. Prezentacijski sloj prima podatke od sloja aplikacijske logike, pretvara ih u prikidan

oblik i šalje korisniku. Ovakva raspodjela aplikacijskih komponenti omogućuje jednostavnije održavanje cjelokupne aplikacije.

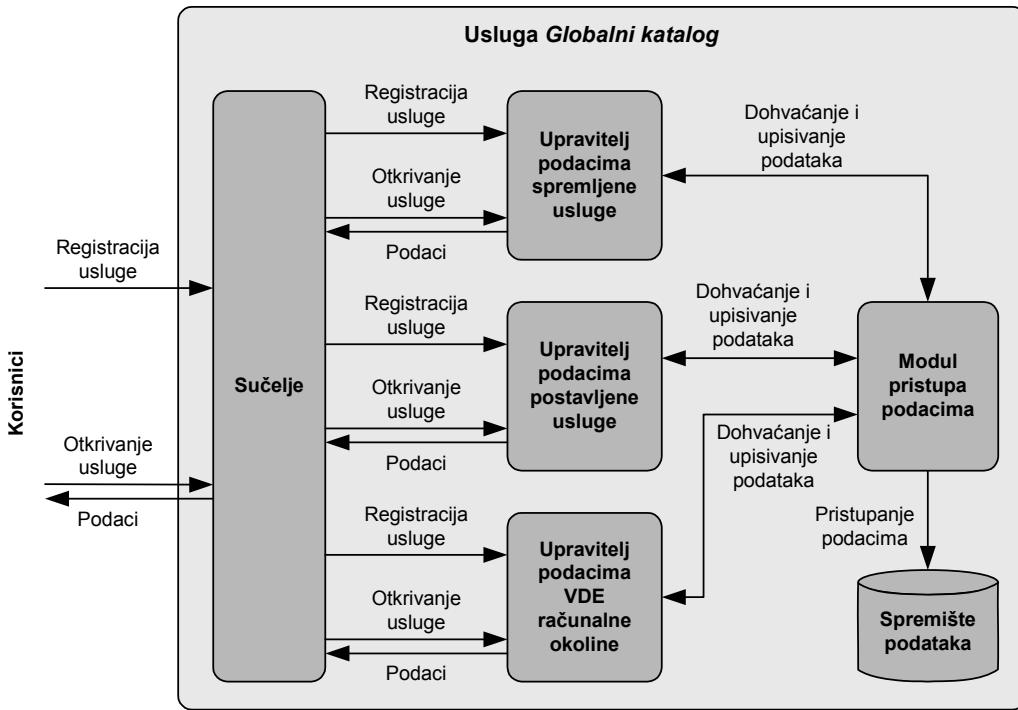


Slika 6-6: Konceptualna arhitektura usluge *Lokalni katalog*

Slika 6-6 prikazuje konceptualnu arhitekturu usluge *Lokalni katalog*. *Lokalni katalog* je namijenjen održavanju i spremanju podataka o postavljenim iinstanciranim oblicima aplikacijskih usluge prisutnih na lokalnom računalu. Sloj podataka usluge sastoji se od *Modula pristupa podacima* i *Spremista podataka*. Za svaku vrstu *Spremista podataka* potrebno je izgraditi specifičan *Modul pristupa podacima*, ali tako da svaki od modula ostvaruju jednako programsko sučelje. Na primjer, podatke je moguće spremiti u bazu podataka ili u datoteke. Pristup podacima u bazi podataka ostvaruje se putem jednog *Modula pristupa podacima*, a pristup podacima u datotekama se ostvaruje putem drugog *Modula pristupa podacima*. Aplikacijski sloj se sastoji od *Upravitelja podacima postavljene usluge* i *Upravitelja podacima instancirane usluge*. Prezentacijski sloj čini *Sučelje*.

Korisničke zahtjeve *Lokalni katalog* prima putem *Sučelja* koje ih proslijedi prikladnom modulu sloja aplikacijske logike. Ako se korisnički zahtjev odnosi na postavljene usluge, onda se zahtjev proslijedi *Upravitelju podacima postavljene usluge*. Ako se korisnički zahtjev odnosi na instancirane usluge, onda se zahtjev proslijedi *Upravitelju podacima instancirane usluge*. Moduli sloja aplikacijske logike ovisno o zahtjevu ili upisuju podatke u sloj podataka ili dohvaćaju podatke iz sloja podataka. Prilikom zahtjeva za registracijom usluge podaci se upisuju u sloj podataka, dok se prilikom zahtjeva za otkrivanjem usluga podaci dohvaćaju iz sloja

podataka. Prilikom registracije postavljene usluge, *Upravitelj podacima postavljene usluge* osim upisivanja podataka o usluzi prosljeđuje podatke do usluge *Globalni katalog*. Također, upravitelji podacima provjeravaju valjanost podataka o uslugama prilikom registracije usluge. Na primjer, ako u *VDE* računalnoj okolini ne postoji čvor *X*, onda podatak da je usluga *A* postavljena na čvor *X* nije valjan.

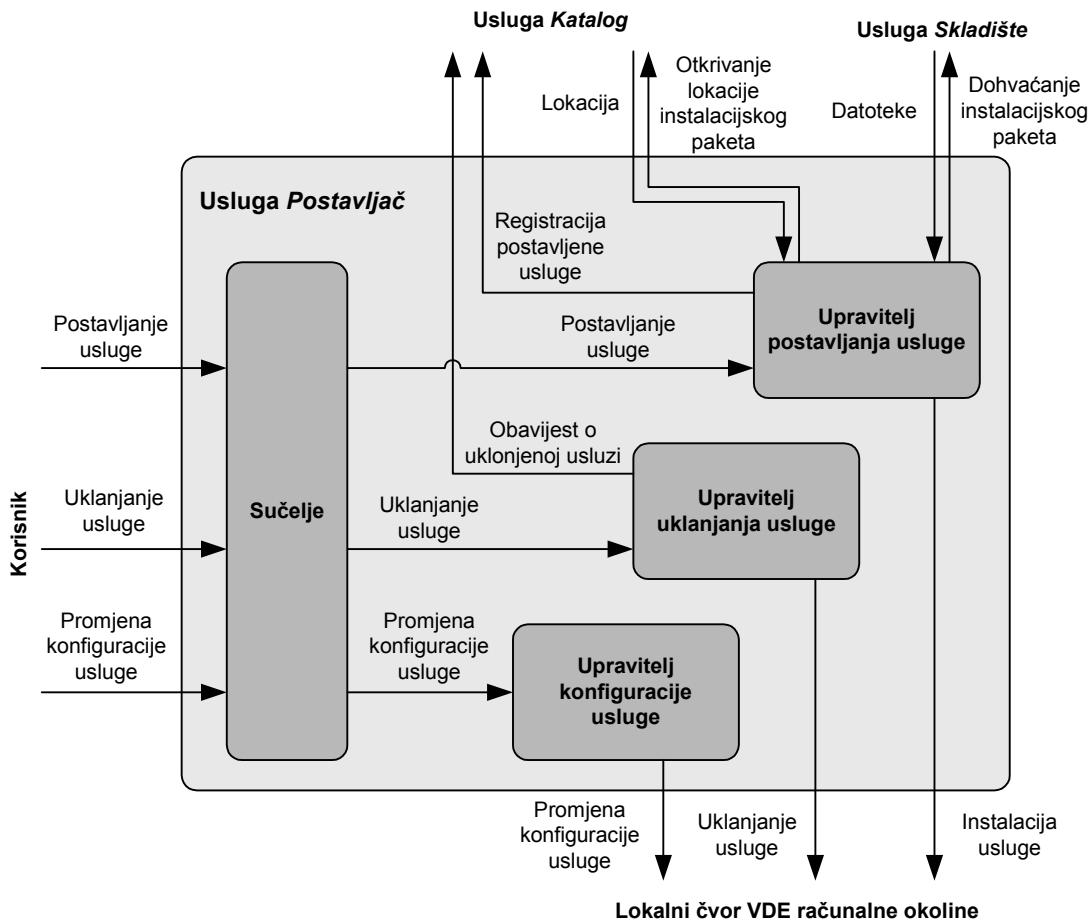


Slika 6-7: Konceptualna arhitektura usluge *Globalni katalog*

Slika 6-7 prikazuje konceptualnu arhitekturu usluge *Globalni katalog*. Usluga *Globalni katalog* održava i sprema podataka o spremjenim i postavljenim uslugama u *VDE* računalnoj okolini. Dodatno, ova usluga održava i sprema specifične podatke o *VDE* računalnoj okolini. Arhitektura usluge je slična arhitekturi usluge *Lokalni katalog*. Razlika je u aplikacijskom sloju, koji se kod usluge *Globalni katalog* sastoji od *Upravitelja podacima spremljene usluge*, *Upravitelja podacima postavljene usluge* i *Upravitelja podacima VDE računalne okoline*. Svaki upravljački modul aplikacijskog sloja zadužen je za podatke jedne skupine podataka o aplikacijskim uslugama prisutnim unutar *VDE* računalne okoline. Korisnici šalju zahtjeve na *Sučelje* koje ih prosljeđuje odgovarajućem upravljačkom modulu u aplikacijskom sloju usluge. Na osnovi korisničkih zahtjeva upravljački moduli u sloju aplikacijske logike dohvaćaju ili spremaju podatke u podatkovni sloj usluge.

6.3.3 Konceptualna arhitektura usluge Postavljač

Usluga *Postavljač* ostvaruje funkcionalnosti postavljanja, uklanjanja i konfiguriranja aplikacijskih usluga. Na svakom čvoru *VDE* računalne okoline izvodi se preslika usluge *Postavljač*, pri čemu svaka preslika ostvaruje navedene funkcionalnosti na lokalnom čvoru. Slika 6-8 prikazuje konceptualnu arhitekturu usluge *Postavljač*.



Slika 6-8: Konceptualna arhitektura usluge *Postavljač*

Sučelje prihvata korisničke zahtjeve i proslijeđuje ih odgovarajućem upravitelju. Ako se korisnički zahtjev odnosi na postavljanje nove usluge na čvor, onda *Sučelje* poziva *Upravitelj postavljanja usluge*. Ako se korisnički zahtjev odnosi na uklanjanje postojeće usluge sa lokalnog čvora, onda *Sučelje* poziva *Upravitelj uklanjanja usluge*. Ako se korisnički zahtjev odnosi na konfiguriranje postavljene usluge, onda *Sučelje* poziva *Upravitelj konfiguracije usluge*.

Postupak postavljanja aplikacijske usluge je najznačajniji postupak *Sustava za postavljanje i otkrivanje usluga*. Na početku postavljanja aplikacijske usluge, *Upravitelj postavljanja usluge* putem usluga *Katalog* i *Skladište* dohvaća instalacijski paket aplikacijske

usluge. Nakon uspješnog dohvaćanja instalacijskog paketa slijedi postupak instalacije usluge. Postupak instalacije aplikacijske usluge vođen je instalacijskom skriptom koja je sastavni dio instalacijskog paketa. Instalacijska skripta definira korake instalacijskog procesa, kao što je preslikavanje izvršnih datoteka usluge na odgovarajuće lokacije ili registracija aplikacijske usluge u operacijski sustav lokalnog računala. Nakon uspješno obavljene instalacije aplikacijske usluge, *Upravitelj instalacije usluge* obavijesti uslugu *Katalog* o novo postavljenoj usluzi. *Upravitelj instalacije usluge* obavijest šalje *Lokalnom katalogu* koji dio informacija prosljeđuje usluzi *Globalni katalog*. Unutar usluge *Lokalni katalog* spremaju se detaljne informacije o postavljenoj usluzi koje se koriste u postupcima uklanjanja i konfiguriranja usluge. Primjer tih detaljnih informacija su lokacije lokalno spremljene instalacijske skripte i konfiguracijske datoteke. Postupak postavljanja usluge ostvaren je kao transakcija, odnosno usluga se postavlja u potpunosti ili se ne postavlja. U slučaju jednog neuspješnog koraka u postupku postavljanja usluge, *Upravitelj instalacije usluge* vraća lokalno računalo u stanje prije početka postavljanja usluge. Također, postupak postavljanja usluge ne prekida rad ostalih postavljenih usluga na lokalnom računalu.

Postupak uklanjanja usluge suprotan je postupku postavljanja usluge. Prilikom primanja zahtjeva za uklanjanjem usluge, *Upravitelj uklanjanja usluge* dohvati iz *Lokalnog kataloga* lokaciju lokalno spremljene instalacijske skripte aplikacijske usluge. Za svaku akciju instalacije usluge postoji suprotna akcija pa tako instalacijska skripta ujedno definira i postupak uklanjanja usluge. Nakon uspješnog uklanjanja usluge, *Upravitelj uklanjanja usluge* obavijesti uslugu *Katalog* o uklonjenoj usluzi. Postupak uklanjanja usluge također ne prekida izvođenje ostalih postavljenih usluga na lokalnom računalu.

Proces konfiguriranja usluge podrazumijeva u prvom redu promjenu početnih parametara u konfiguracijskoj datoteci usluge. Nakon što primi zahtjev za promjenom konfiguracije usluge, *Upravitelj konfiguracije usluge* dohvati iz *Lokalnog kataloga* lokaciju konfiguracijske datoteke postavljene usluge. Nakon toga upravitelj učita datoteku i promijeni vrijednost parametra usluge. Od aplikacijskih usluga se zahtijeva da konfiguracijske datoteke zadovoljavaju određeni format.

7 Programsко ostvarenje *Sustava za postavljanje i otkrivanje usluga*

Sustav za postavljanje i otkrivanje usluga pruža mehanizme postavljanja i otkrivanja aplikacijskih usluga u *VDE* računalnoj okolini. Sustav je programski ostvaren uporabom standarda Web usluga. Standardi Web usluga omogućuju interoperabilnost, odnosno omogućuju komunikaciju između procesa koji se izvode na različitim računalnim platformama i operacijskim sustavima. Time je uporaba *Sustava za postavljanje i otkrivanje usluga* postala neovisna o računalnim platformama i operacijskim sustavima. Zasnovanost na standardima Web usluga također olakšava održavanje i buduće nadogradnje sustava.

U ovom poglavlju opisano je programsko ostvarenje *Sustava za postavljanje i otkrivanje usluga*. Na početku poglavlja navode se tehnologije korištene u izgradnji sustava. Ograničenja korištenih tehnologija otežavaju ostvarenje generičkog postupka postavljanja usluga. Zbog toga razloga *Sustav za postavljanje i otkrivanje usluga* je ograničen na postavljanje aplikacijskih usluga koje ostvaruju unaprijed zadatu arhitekturu. Mehanizmi postavljanja i otkrivanja usluga ostvareni su unutar usluga *Skladište*, *Katalog* i *Postavljač*. Za svaku od sastavnih usluga sustava navodi se programska arhitektura, definiraju se strukture spremišta podataka te se detaljno opisuju značajni unutarnji procesi. Nadalje, opisuju se razvijeni mehanizmi za otkrivanje i dijagnosticiranje pogrešaka nastalih radom sustava. S ciljem olakšavanja nadzora i upravljanja sustavom ostvareno je prikladno Web korisničko sučelje. Opis Web korisničkog sučelja te kratak opis postupka instalacije sustava navodi se na kraju poglavlja.

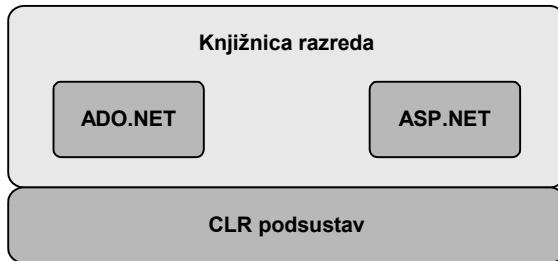
7.1 Korištene tehnologije

Raspodijeljene aplikacije zasnovane na standardima Web usluga moguće je ostvariti uporabom različitih tehnologija. U izgradnji *Sustava za postavljanje i otkrivanje usluga* korištene su tehnologije .NET radnog okvira [102]. Povod odabiru tehnologija .NET radnog okvira je bogata potpora razvoju raspodijeljenih aplikacija zasnovanih na standardima Web usluga. Nadalje, u postupku otkrivanja usluga zahtijeva se brz i učinkovit pristup spremiljenim podacima. Spremanje i pretraživanje podataka uobičajan je problem u suvremenim informacijskim sustavima, a obično se rješava uporabom tehnologija relacijskih baza podataka. Ugrađena potpora relacijskim bazama podataka unutar .NET radnog okvira potaknula je uporabu baza podataka unutar *Sustava za postavljanje i otkrivanje usluga*. Osim prijenosa informacija o aplikacijskim uslugama, unutar sustava se prenose i instalacijski paketi usluga. Komunikacija

između usluga sustava ostvaruje se SOAP protokolom koji zbog tekstualnog formata poruka nije prikladan za prijenos binarnih datoteka instalacijskih paketa. Problem prijenosa binarnih podataka SOAP protokolom uočen je od strane različitih Internet organizacija koje su predložile nekoliko rješenja. Rješenja se međusobno razlikuju prema učinkovitosti i slaganju s osnovnim principima standarda Web usluga. Postavljanje aplikacijskih usluga ujedno otvara problem sigurnosti. Bez sigurnosnih mehanizama *Sustav za postavljanje i otkrivanje usluga* olakšava širenje zlonamjernih aplikacijskih usluga računalnom okolinom. Identičan sigurnosni problem pojavljuje se u svim sustavima koji omogućuju premještanje izvršnog kôda mrežnim računalima. U sklopu istraživanja sustava za prijenos izvršnog kôda predloženo je nekoliko pristupa rješavanju problema sigurnosti.

7.1.1 .NET radni okvir

.NET radni okvir (engl. .NET Framework) je proizvod Microsoft korporacije za potporu razvoju i izvođenju raspodijeljenih aplikacija usmjerenih javnoj mreži Internet. Slika 7-1 prikazuje osnovnu arhitekturu .NET radnog okvira. .NET radni okvir sastoji se od *podsistava za potporu izvođenju aplikacija* (engl. Common Language Runtime, CLR) i *knjižnice razreda*.



Slika 7-1: Arhitektura .NET radnog okvira

CLR podsustav je posrednik između .NET aplikacija i operacijskog sustava. Komponente .NET aplikacije moguće je ostvariti različitim programskim jezicima, a CLR podsustav se prilikom izvođenja brine o njihovom pravilnom povezivanju. Na primjer, jednu komponentu aplikacije moguće je napisati u jeziku VB.NET, a druga u jeziku C#. Ovakav višejezični način izgradnje aplikacija zasniva se na prevodenju izvornog kôda aplikacijskih komponenti u platformski neovisan međujezik (engl. Common Intermediate Language, CIL). CLR podsustav tijekom izvođenja .NET aplikacije prevodi njene komponente iz međujezika u strojni jezik računala. Sljedeća važna funkcionalnost CLR podsustava je "sakupljanje otpadaka" (engl. garbage collector). CLR podsustav nadgleda uporabu objekata za vrijeme izvođenja .NET aplikacije. Ako se objekt više ne koristi, odnosno niti jedan drugi objekt više ne posjeduje njegovu referencu, CLR sustav ga automatski uklanja iz memorije.

Knjižnica razreda sa preko dvije tisuće ostvarenih razreda pruža programeru širok skup funkcionalnosti. Tako je u knjižnici moguće pronaći standardne razrede, poput razreda s matematičkim funkcijama, ali i specijalizirane razrede, poput razreda za pristup Windows registry bazi podataka. U izgradnji *Sustava za postavljanje i otkrivanje usluga* korištena su dva podskupa knjižnice: *programski model za pristup relacijskim bazama podataka* (ADO.NET) i *programski model za razvoj Web aplikacija usmjerenih prema globalnoj mreži Internet* (ASP.NET).

ADO.NET programski model pruža jedinstven način pristupa svim sustavima relacijskih baza podataka. Ovaj programski model usko je vezan uz programski model za pristup XML dokumentima. Uzrok te povezanosti je sve veća popularnost Web aplikacija u kojima je uobičajeno da se podaci mrežom prenose zapisani u XML formatu. ADO.NET programski model pruža mehanizme za automatsku pretvorbu podataka dohvaćenih iz baze podataka u XML format što olakšava izgradnju raspodijeljenih aplikacija zasnovanih na Web uslugama i XML tehnologijama. Osim čvrsto povezanog pristupa bazi podataka, ADO.NET podržava i slabo povezan pristup. Kod slabo povezanog pristupa bazi podataka svaki zahtjev je nezavisan od prethodnih zahtjeva i veza s bazom podataka se zatvara nakon obrade zahtjeva. Slaba povezanost karakterističan je pristup kod Web usluga pa je stoga upotrijebljen i u oblikovanju ADO.NET programskog modela.

ASP.NET programski model pruža potporu izgradnji i izvođenju Web aplikacija. Web aplikacije je moguće podijeliti na Web usluge i dinamičke Web stranice. Razlika je u prikazu rezultata obrade korisničkih zahtjeva. Kod Web usluga rezultat obrade je SOAP poruka, dok je kod dinamičkih Web stranica rezultat HTML stranica. ASP.NET potpora izvođenju Web aplikacija ugrađuje se u Web poslužitelj, odnosno u Microsoft Internet Information Services (IIS). Posljedica ugradnje ASP.NET potpore u IIS poslužitelj je dolazak korisničkih zahtjeva putem IIS poslužitelja do Web aplikacije. ASP.NET potpora izvođenju Web usluga u trenutnoj inačici .NET radnog okvira (1.1) pruža mehanizme za korištenje tek osnovnog skupa protokola Web usluga: SOAP, WSDL i UDDI. Ostale standarde Web usluga, poput WS-Security, trenutno je moguće koristiti putem posebnih proširenja osnovnog radnog okvira.

7.1.2 Relacijske baze podataka

U matematici, relacija je skup n -torki, pri čemu se za n -torke iz skupa kaže da zadovoljavaju relaciju. Princip relacija iskorišten je u relacijskom modelu baza podataka. Relacija se u relacijskom modelu baze podataka ostvaruje u obliku tablice, zbog čega se riječi

relacija i tablica često koriste kao sinonimi. Reci tablice su n -torke relacije, dok stupci tablice sadrže istovrsne elemente n -torki.

Tablica Osoba

Ime	Prezime	JMBG	Mjesto	PoštanskiBroj
Ivan	Ivić	120397065204	Zagreb	10000
Pero	Perić	200596521400	Zagreb	10000
Marko	Markić	231098031607	Zagreb	10000
Ana	Anić	280897012914	Koprivnica	48000
Branko	Anić	070996831001	Koprivnica	48000

Slika 7-2: Primjer relacijske baze podataka

Slika 7-2 prikazuje primjer relacijske baze podataka koja se sastoji od jedne tablice, odnosno relacije *Osoba*. Svaki redak tablice sadrži podatke o nekoj osobi pa se kaže da svaki redak zadovoljava relaciju *Osoba*. Osoba je određena s pet svojstava, odnosno atributa: *Ime* (ime osobe), *Prezime* (prezime osobe), *JMBG* (jedinstveni matični broj građana), *Mjesto* (mjesto prebivališta) i *PoštanskiBroj* (poštanski broj mjesta prebivališta).

Osnovni nedostatak relacije sa slike je redundancija podataka. Kao što se vidi, vrijednost *Zagreb* se pojavljuje na tri mesta zajedno s pripadnim poštanskim brojem. Vrijednost *Koprivnica* se također pojavljuje dva puta u tablici. Uklanjanje redundancije podataka ostvaruje se postupkom *normalizacije* baze podataka. Postupkom normalizacije tablica se dijeli na više manjih tablica koje zajedno odražavaju početnu veliku tablicu. Važna pomoć u procesu normalizacije su *primarni ključevi tablice*. Primarni ključ tablice se definira kao skup atributa koji jedinstveno određuju redak tablice. U primjeru sa slike atribut *JMBG* je primarni ključ tablice, jer se ne može dogoditi da postoje dvije osobe s istim jedinstvenim matičnim brojem građana. Tablicu je moguće podijeliti tako da zadovoljava različite stupnjeve normalizacije. Teorija relacijskih baza podataka definira prvu, drugu, treću, četvrту i petu normalnu formu [103]. U praksi se obično koristi normalizacija tablice na treću normalnu formu, čime tablica istovremeno zadovoljava prvu i drugu normalnu formu. Normalizacija na četvrstu i petu normalnu formu provodi se samo u posebnim slučajevima.

Slika 7-3 prikazuje rezultat normalizacije relacije sa slike 7-2 na treću normalnu formu. Početna tablica podijeljena je na tablicu *Osoba* i tablicu *Mjesto*. Atribut *JMBG* je postavljen za primarni ključ tablice *Osoba*, a atribut *PoštanskiBroj* je postavljen za primarni ključ tablice *Mjesto*, jer dva mesta ne mogu imati isti poštanski broj. Postupkom normalizacije informacije nisu izgubljene, već su samo raspodijeljene u više tablica. Na primjer, još uvijek je jednoznačno

određeno da osoba *Ivan Ivić* stanuje u Zagrebu, samo što se ime mesta stanovanja sada nalazi u drugoj tablici.

Tablica Osoba

Ime	Prezime	*JMBG	PoštanskiBroj
Ivan	Ivić	120397065204	10000
Pero	Perić	200596521400	10000
Marko	Markić	231098031607	10000
Ana	Anić	280897012914	48000
Branko	Anić	070996831801	48000

Tablica Mjesto

Mjesto	*PoštanskiBroj
Zagreb	10000
Koprivnica	48000

Slika 7-3: Normalizirana relacijska baza podataka

Osnovna snaga relacijskog modela baza podataka je što se u bazi podataka ne definiraju statički sve moguće relacije među podacima, već se nove relacije izvode iz postojećih relacija. Izvođenje novih relacija omogućuje strukturirani jezik upita (engl. Structured Query Language, SQL). Na primjer, SQL jezikom je moguće postaviti upit o imenima svih osoba koje žive u Zagrebu. Rezultat tog upita je nova relacija koja se sastoji od jednog stupca i tri retka. U recima su vrijednosti: *Ivan*, *Pero* i *Marko*. Relacije dobivene kao rezultat upita su privremene, odnosno ne spremaju se fizički u bazu podataka. SQL jezik također omogućuje upite za mijenjanje, dodavanje ili brisanje redaka tablice koja je fizički spremljena u bazi podataka.

SQL jezik je usko povezan s relacijskim sustavom za upravljanje bazom podataka (engl. Relational Database Management Systems, RDBMS). RDBMS sustav upravlja relacijskim bazama podataka što uključuje spremanje tablica baze podataka na disk, pristup tablicama, provjera suvislosti tablica, potpora transakcijama i druge funkcije. Na primjer, pošto je nad tablicom *Mjesto* definiran primarni ključ *PoštanskiBroj*, RDBMS sustav neće dopustiti da se u tablicu upiše *n*-torka (*Split*, 10000) jer u tablici već postoji *n*-torka s vrijednošću atributa *PoštanskiBroj* 10000. Zabrana upisivanja neispravne *n*-torke je primjer održavanja suvislosti tablice.

Transakcija je nedjeljiva jedinica unutar koje se obavlja jedna ili više operacija nad bazom podataka. Transakcije posjeduju svojstva nedjeljivosti, integriteta, odvojenosti i postojanosti. Navedena svojstva se jednim imenom nazivaju ACID svojstva (engl. Atomicity, Consistency, Isolation, Durability). Svojstvo nedjeljivosti jamči da su ostvarene sve operacije transakcije ili niti jedna. Ako samo jedna operacija transakcije ne uspije, onda je potrebno poništiti sve prethodne operacije. Svojstvo integriteta jamči da je baza podataka u suvislom stanju prije i poslije transakcije. Baza je u suvislom stanju ako zadovoljava sva pravila integriteta. Svojstvo odvojenosti jamči odvojenost transakcije od ostalih transakcija. Rezultat

izvođenja paralelnih transakcija treba biti jednak rezultatu slijednog izvođenja transakcija. Svojstvo postojanosti jamči trajnost promjena baze podataka nakon obavljene transakcije. O zadovoljenju navedenih svojstava transakcija brine se RDBMS sustav.

7.1.3 Prijenos binarnih podataka SOAP protokolom

SOAP komunikacijski protokol zasniva se na XML jeziku koji unatoč svojoj prilagodljivosti i širokoj prihvaćenosti ipak nije prikladan za zapis binarnih podataka kao što su slikovne ili zvučne datoteke. Na primjer, slika 7-4 prikazuje mogući način zapisa grafičkog objekta u XML formatu. Ovako zapisan grafički objekat zauzima znatno više memoriskog prostora u odnosu na binarni zapis. Također, binarni zapis multimedijalnih podataka pruža mogućnost sažimanja podataka što je teško ostvariti kod strukturiranog XML zapisa.

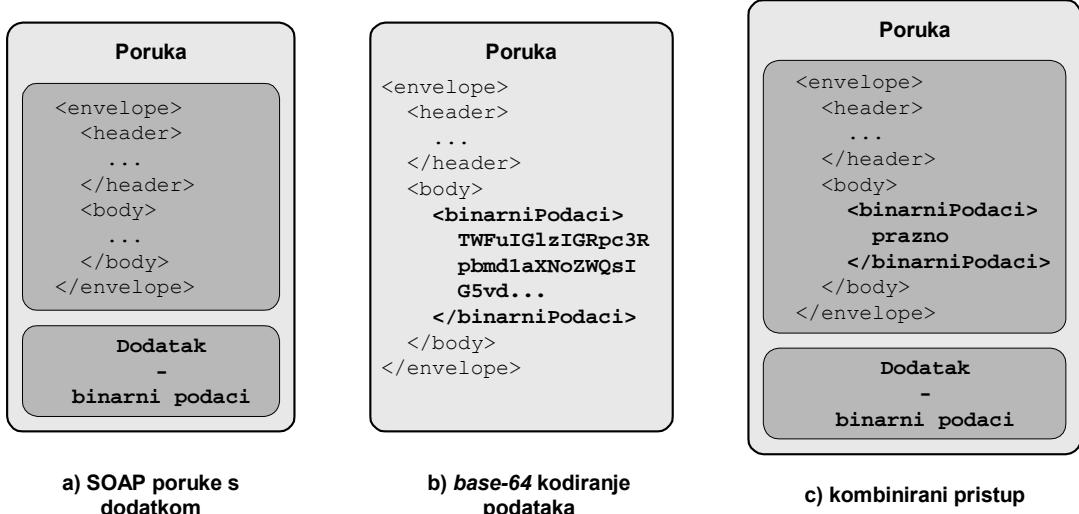
```
<slika>
  <red>
    <pixel>
      <boja>00FF00</boja>
    </pixel>
    <pixel>
      <boja>00FF00</boja>
    ...
  ...
</slika>
```

Slika 7-4: Zapisivanje grafičkog objekta u XML formatu

Predloženo je nekoliko pristupa prijenosa binarnih podataka SOAP protokolom: *SOAP poruke s dodatkom*, *base-64 kodiranje podataka u XML poruku* i *kombinirani pristup*. Slika 7-5 prikazuje strukturu SOAP poruka za svaki od navedenih pristupa. Struktura SOAP poruke s dodatkom prikazana je na slici 7-5a. Kod ovog pristupa binarni podaci nadodaju se u nepromijenjenom obliku na kraj SOAP poruke. Nastala poruka je kombinacija tekstualnih i binarnih podataka pa se stoga ne podudara s osnovnim principima Web usluga koji zahtjevaju tekstualni sadržaj poruka. Ovaj pristup prijenosa binarnih podataka korišten je u specifikacijama *SOAP with Attachments (SwA)* [105] i *WS-Attachments* [106]. Zbog neslaganja s osnovnom principima Web usluga navedene specifikacije nisu u širokoj uporabi. Neprikladnost ovog pristupa je i u nemogućnosti digitalnog potpisivanja binarnih podataka prema WS-Security standardu, jer WS-Security standard omogućuje digitalno potpisivanje samo XML dijelova poruke.

Slika 7-5b prikazuje drugi pristup prijenosa binarnih podataka SOAP protokolom. Ovdje se binarni podaci zapisuju u SOAP poruku kodiranjem *base-64* metodom. *Base-64* metoda kodiranja [107] omogućuje pretvaranje niza binarnih znakova u niz alfanumeričkih znakova. Ovim pristupom se binarni podaci pretvaraju u tekstualne podatke pa je rezultantna poruka tekstualnog formata i zadovoljava principe Web usluga. Nedostatak ovog pristupa je

neučinkovitost *base-64* kodiranja jer kodirani binarni podaci zauzimaju približno 33% više prostora u odnosu na nekodirane. Prednost ove metode u odnosu na prethodnu je mogućnost digitalnog potpisivanja binarnih podataka, pošto je cijela poruka u XML formatu.



Slika 7-5: Pristupi u prijenosu binarnih datoteka SOAP protokolom

Slika 7-5c prikazuje strukturu poruke nastalu kombinacijom prethodna dva pristupa. Kod ovog pristupa pošiljatelj gradi poruku koja se sastoji od tekstualnog i binarnog dijela. Binarni podaci se šalju u nepromijenjenom obliku kao dodatak poruci. Za razliku od prvog pristupa, ovdje se u XML dijelu poruke navodi mjesto gdje bi se podaci nalazili da su kodirani *base-64* metodom (element *<binarniPodaci>* na slici 7-5c). Na prijamnoj se strani iz takve poruke izvuče binarni dodatak te se *base-64* metodom kodira u tekstualan oblik koji se ugraditi na odgovarajuće mjesto u XML dio poruke. U nastavku obrade poruke procesira se dobiveni XML dokument. Ovakav pristup iskorištava učinkovitost slanja binarnih podataka kao dodataka SOAP poruci te ujedno slijedi osnovne principe Web usluga. Binarni sadržaj moguće je digitalno potpisati jer poruka na prijamnoj strani izgleda kao da je u cijelosti poslana u XML obliku. Opisani pristup osnova je *Message Transmission Optimization Mechanism (MTOM)* specifikacije [108].

7.1.4 Sigurnost pokretnog izvršnog kôda

Svi računalni sustavi koji omogućuju premještanje izvršnog kôda s jednog računala na drugo uvode problem širenja različitih oblika zlonamjernih programa. Trenutno postoje četiri glavna pristupa u rješavanju problema sigurnosti pokretnog izvršnog kôda [109]: *model pješčanika* (engl. sandbox), *potpisivanje izvršnog kôda* (engl. code signing), *model sigurnosne stijene* (engl. firewall) i *model izvršnog kôda s dokazom* (engl. proof-carrying code).

Modelom pješčanika ograničavaju se pokretnom izvršnom kôdu dostupna računalna sredstva. Na taj način se ograničava šteta koju bi izvršni kôd eventualno mogao ostvariti. Ovaj pristup obično se ostvaruje zabranom pristupa datotekama operacijskog sustava te ograničavanjem sposobnosti otvaranja mrežnih veza. Primjer uporabe ovog pristupa je tehnologija *Java appleta*. Java appleti su oblik Java aplikacija koje korisnik skine s Interneta i izvodi lokalno unutar svog Web preglednika. Pravidni stroj u kojem se izvodi izvršni kôd Java appleta automatski ograničava pristup kritičnim sredstvima operacijskog sustava kao što su datoteke operacijskog sustava ili mrežne veze.

Kod potpisivanja izvršnog kôda, korisnik održava listu proizvođača izvršnog kôda kojima vjeruje. Kada korisnik dohvati s mreže izvršni kôd, on može provjeriti na osnovi digitalnog potpisa identitet njegovog proizvođača. Ako je proizvođač na korisnikovoj listi povjerljivih proizvođača, onda korisnik dopušta izvođenje izvršnog kôda. Izvršni kôd prilikom izvođenja ima potpune ovlasti nad svim računalnim sredstvima. Ovakav pristup je iskorišten kod Microsoftove *ActiveX* tehnologije. ActiveX kontrole su dijelovi Web stranice koji se izvode na korisničkom računalu. Unatoč potpisivanju izvršnog kôda, ActiveX tehnologija nije doživjela široku uporabu zbog dodatnih sigurnosnih propusta.

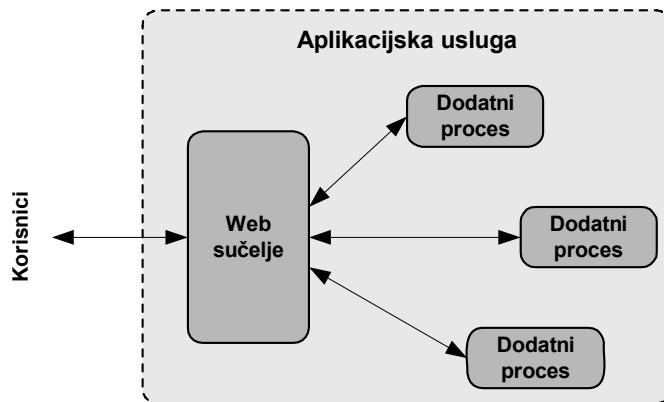
U modelu sigurnosne stijene korisnik provjerava svojstva izvršnog kôda i na osnovi toga dopušta ili zabranjuje njegovo izvođenje. Izvlačenje sigurnosnih svojstava iz izvršnog kôda nije jednostavan zadatak. Ako se problem promatra s teorijskog stajališta, onda se pokazuje da nije moguće u svakom trenutku donijeti odluku o sigurnom ili nesigurnom izvršnom kôdu. Problem je analogan problemu zaustavljanja računalnih programa (engl. halting problem) za koji je pokazano da je neodlučiv [111].

U modelu izvršnog kôda s dokazom, statičkom se analizom izvršnog kôda pokušava dokazati da li izvršni kôd zadovoljava određene sigurnosne karakteristike. Tako je za neke programe moguće dokazati da neće uzrokovati prelijevanje spremnika (engl. buffer overflow). Međutim, postoje sigurnosna svojstva koja nije moguće dokazati ovom metodom.

7.2 *Aplikacijske usluge*

Aplikacijske usluge programski ostvaruje korisnik *VDE* računalne okoline. Programska ostvarena aplikacijska usluga se putem *Sustava za postavljanje i otkrivanje usluga* postavlja na čvorove *VDE* računalne okoline. Proces postavljanja usluga je osjetljiv s obzirom na specifična svojstva usluge i na operacijski sustav za koji je usluga ostvarena. Za uspješno postavljanje aplikacijskih usluga potrebno je ograničiti programsku izvedbu aplikacijskih usluga.

Ograničavanje se svodi na definiciju tehnologija izgradnje aplikacijskih usluga i jedinstvene programske arhitekture.



Slika 7-6: Programska arhitektura aplikacijske usluge

Slika 7-6 prikazuje jedinstvenu programsku arhitekturu aplikacijskih usluga. Trenutna inačica *Sustava za postavljanje i otkrivanje usluga* podržava postavljanje aplikacijskih usluga ostvarenih tehnologijama .NET radnog okvira. Aplikacijska usluga sastoji se od Web sučelja i dodatnih procesa. Web sučelje je obavezan dio aplikacijske usluge, a programski se ostvaruje kao .NET Web usluga. Dodatni procesi su opcionalan dio aplikacijske usluge te ih unutar aplikacijske usluge može biti više od jednog. Uloga Web sučelja je da omogući vanjskim korisnicima kontrolu dodatnih procesa. Na primjer, dodatne procese moguće je iskoristiti za izvođenje dugotrajnih proračuna, koje korisnici putem Web sučelja mogu pokretati, nadgledati ili zaustavljati. Tehnologija komunikacije između Web sučelja i procesa je proizvoljna.

Uloga aplikacijskih usluga je ostvariti prividnost sredstava. Ostvarenje prividnosti sredstava zasniva se na standardima Web usluga, prvenstveno na WSRF skupu specifikacija. WSRF skup specifikacija definira mehanizme koji omogućuju pristup svojstvima sredstava, uništavanje sredstava te njihovo grupiranje. Prividnost sredstava na osnovi WSRF skupa specifikacija ostvaruje se putem Web usluge koja je posrednik između vanjskih korisnika i internog ostvarenja sredstva. Opširni opis WSRF skupa specifikacija nalazi se u poglavljju 2.4.4.

Od aplikacijske usluge se zahtijeva ostvarenje podskupa WSRF mehanizama. Jedan od tih mehanizama je pristup instanci sredstva. Vanjski korisnici pristupaju instanci sredstva putem SOAP poruke u kojoj navode identifikator instance i operaciju koja se treba izvesti nad sredstvom. Slika 7-7 prikazuje primjer takve SOAP poruke. Unutar elementa *<Header>* poruke navodi se element *<Address>* koji adresira instanca sredstva. Element *<To>* određuje adresu Web sučelja aplikacijske usluge, dok element *<Resource>* identificira instancu sredstva.

Operacija nad instancom sredstva te pripadni parametri operacije navode se unutar elementa *<Body>*.

```

<Envelope>
  <Header>
    <Address>
      <To>cvorA</To>
      <Resource>instanca1</Resource>
    </Address>
  </Header>
  <Body> ... <Body>
</Envelope>

```

Slika 7-7: Primjer SOAP poruke s adresom instance usluge

Stvaranje instance sredstva nije definirano WSRF skupom specifikacija jer postupak stvaranja instance ovisi o vrsti sredstva. Međutim, u *Sustavu za postavljanje i otkrivanje usluga* zahtijeva se od aplikacijskih usluga da registriraju nove instance kako bi im i ostali korisnici mogli pristupati. Registracija instance sredstava provodi se pozivom *Lokalnog kataloga*.

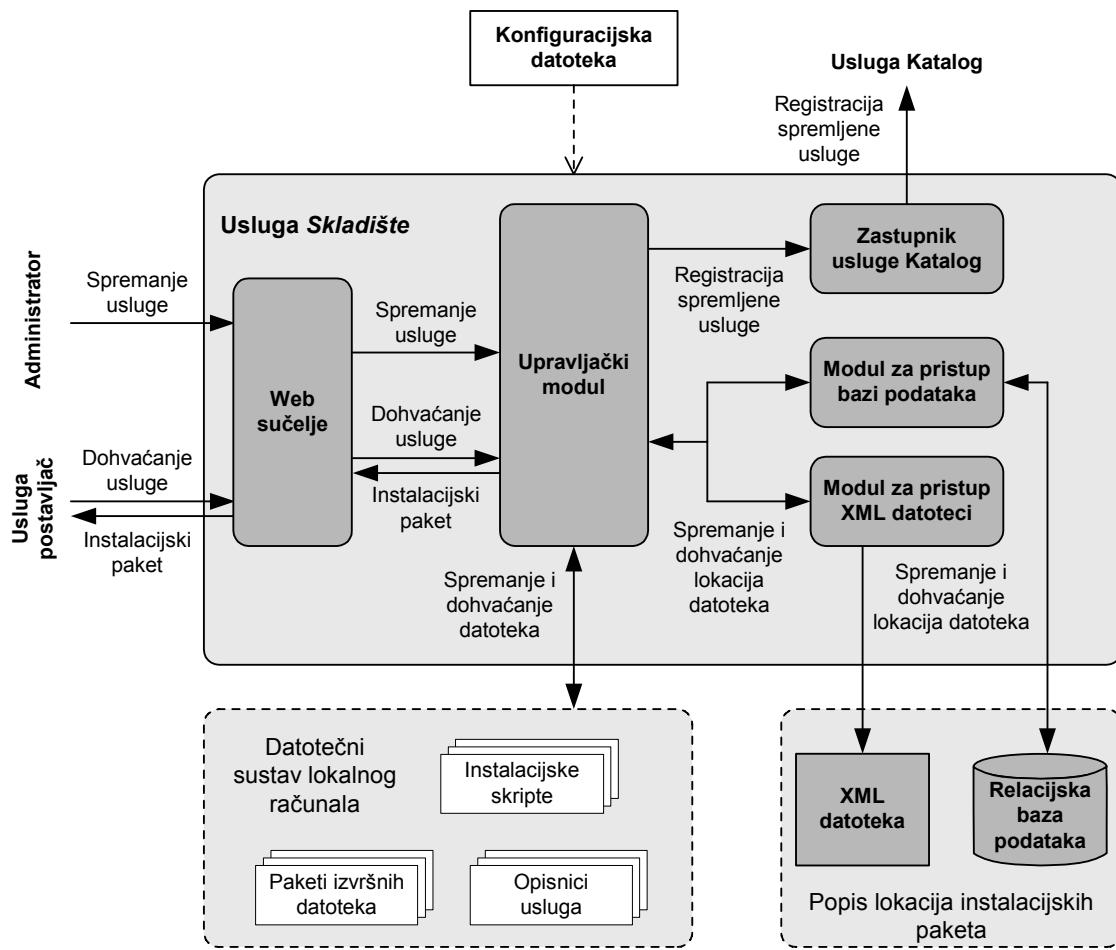
Ako se ne poduzmu sigurnosne mjere, *Sustav za postavljanje i otkrivanje usluga* postaje moćan alat za širenje zlonamjernih aplikacijskih usluge *VDE* računalnom okolinom. U *Sustavu za postavljanje i otkrivanje usluga* ostvarena je tek rudimentarna potpora sigurnosti. Iskorišten je pristup pješčanika u kojem se aplikacijskim uslugama ograničava pristup kritičnim računalnim sredstvima. Aplikacijske usluge izvode se pod posebnim korisničkim računom s ograničenim ovlastima. Podrazumijevane ovlasti navedenog korisničkog računa onemogućavaju spremanje rezultata računalne obrade u datoteke. Kako bi se uslugama omogućilo spremanje razultata obrade, ograničenom korisničkom računu je dopušten pristup određenim stazama u datotečnom sustavu. Pošto se sve aplikacijske usluge izvode pod istim korisničkim računom postoji opasnost da jedna aplikacijska usluga mijenja datoteke koje je stvorila druga aplikacijska usluga. Drugi problem je u nemogućnosti kontrole računalnih sredstava koje usluga može koristiti. Na primjer, aplikacijska usluga može namjerno zauzeti velik dio radne memorije ili slobodnog prostora na disku i time onesposobiti računalo. Navedene probleme korištenim pristupom nije moguće spriječiti.

7.3 *Usluga Skladište*

Usluga *Skladište* sprema instalacijske pakete aplikacijskih usluga. Unutar *VDE* računalne okoline postoji više *Skladišta*. Uvišestručavanjem instalacijskih paketa u više *Skladišta* osigurava se pouzdanost te bolja radna svojstva cjelokupnog sustava. U ovom poglavlju opisuje se programska arhitektura usluge *Skladište*, navode se konstrukti instalacijske skripte, te se opisuje organizacija izvornog kôda usluge.

7.3.1 Programska arhitektura

Programska arhitektura usluge *Skladište* prikazana je na slici 7-8. *Skladište* se sastoji od *Web sučelja*, *Upravljačkog modula*, *Zastupnika usluge Katalog*, *Modula za pristup bazi podataka* i *Modula za pristup XML datotekama*. *Web sučelje* je pristupna točka usluge ostvarena u skladu sa standardima Web usluga. Korisnici putem SOAP protokola šalju *Web sučelju* zahtjeve za spremanjem ili dohvaćanjem instalacijskih paketa usluge. *Web sučelje* proslijedi korisničke zahtjeve *Upravljačkom modulu* čija je uloga upravljanje procesima usluge.



Slika 7-8: Programska arhitektura usluge *Skladište*

Upravljački modul spremi i dohvati instalacijske pakete iz datotečnog sustava lokalnog računala. Lokacije datoteka instalacijskog paketa spremaju se u interni popis instalacijskih paketa. Usluga *Skladište* podržava dva oblika popisa instalacijskih paketa: XML datoteka i relacijska baza podataka. *Skladište* istovremeno koristi samo jedan od dva moguća oblika popisa instalacijskih paketa. Korišteni oblik popisa definira se u konfiguracijskoj datoteci usluge *Skladište* tijekom postupka instalacije usluge. Podrazumijevani oblik popisa instalacijskih paketa

je XML datoteka. Uporaba relacijske baze podataka zahtijeva složeniji postupak konfiguriranja usluge *Skladište* u odnosu na uporabu XML datoteke, ali zato osigurava brži dohvat podataka u slučaju velikog broja podataka.

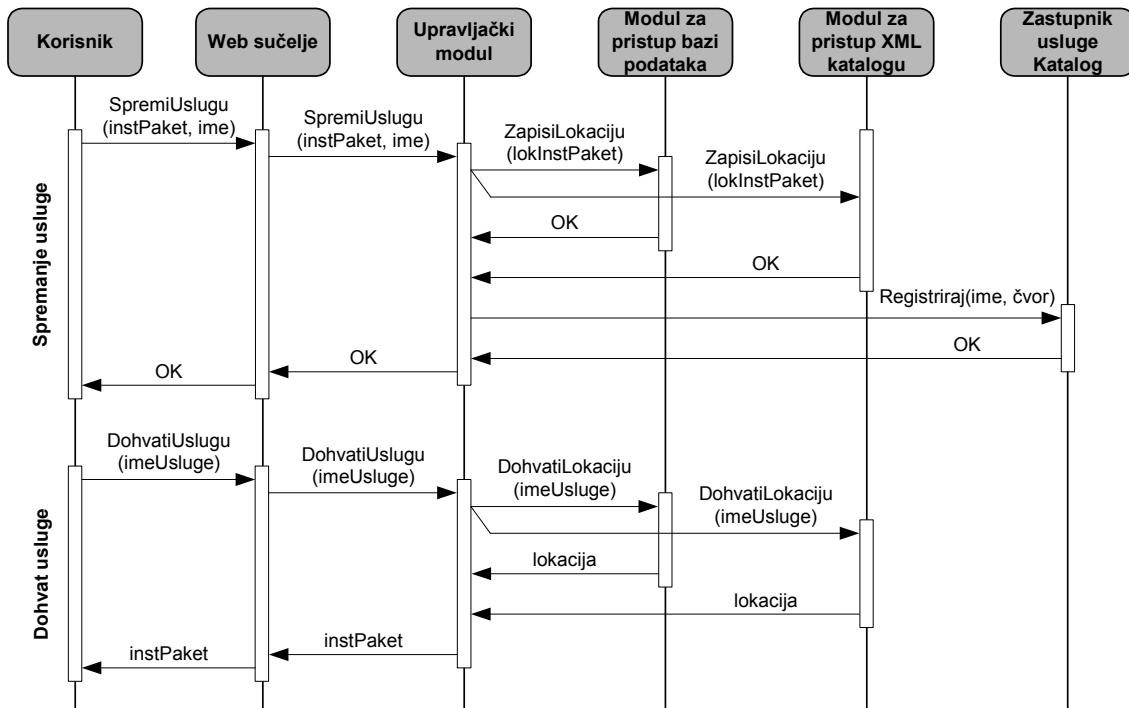
Pristup popisu instalacijskih paketa ostvaruje se putem pripadnih pristupnih modula. Pristup bazi podataka ostvaruje se putem *Modula za pristup bazi podataka*, dok se pristup XML popisu ostvaruje putem *Modula za pristup XML datoteci*. Oba modula ostvaruju isto programsko sučelje pa im *Upravljački modul* pristupa na istovjetan način.

Zastupnik usluge Katalog posreduje u komunikaciji između usluga *Skladište* i *Katalog*. Zastupnik pretvara interne međuobjekte pozive unutar *Skladišta* u odgovarajuće SOAP poruke koje šalje *Katalogu*. Zastupnik također prima SOAP poruke odgovora te ih pretvara u međuobjektne poruke odgovora.

Instalacijski paket sadrži sve informacije potrebne za uspješno postavljanje aplikacijske usluge na čvorove *VDE* računalne okoline. Dužnost administratora *VDE* računalne okoline je pripremiti datoteke instalacijskog paketa u prikladni oblik. Instalacijski paket se sastoji od paketa datoteka, instalacijske skripte i opisnika usluga. Uz izvršne datoteke aplikacijske usluge, paket datoteka sadrži sve ostale datoteke potrebne za uspješno izvođenje usluge. Paket datoteka se s ciljem smanjenja veličine sažima ZIP algoritmom [115]. ZIP algoritam sažimanja podataka odabran je zbog široke prihvaćenosti na javnoj mreži Internet. Instalacijska skripta je tekstualna datoteka koja opisuje postupak instalacije aplikacijske usluge na čvor računalne okoline. Elementi instalacijske skripte opisuju se u poglavljju 7.3.2. Opisnik usluge je tekstualna datoteka koja definira sučelje aplikacijske usluge, a zasniva se na WSDL jeziku. Opisnik usluge za razliku od uobičajenog WSDL opisa usluge ne sadrži opis konkretnog povezivanja koji prema WSDL standardu definira mrežnu adresu i prijenosne protokole. Vrijednosti opisa konkretnog povezivanja tijekom spremanja usluge je neodređen, jer je jednom spremljenu uslugu moguće postaviti na više čvorova računalne okoline.

Prilikom prijenosa binarnih datoteka instalacijskog paketa između korisnika i usluge *Skladište* potrebno je upotrijebiti jedan od mehanizama prijenosa binarnih podataka SOAP protokolom opisanih u poglavljju 7.1.3. U programskom ostvarenju usluge *Skladište* koristi se *base-64* kodiranje binarnih podataka u tekstualni oblik. Tekstualno kodirani instalacijski paketi ugrađuju se u SOAP poruku i prenose mrežom do *Skladišta*. *Web sučelje* prihvata SOAP poruke s binarnim podacima kodiranim u tekstualni oblik, pretvara ih u polje okteta koje proslijedi *Upravljačkom modulu*. Postupak kodiranje binarnih podataka *base-64* metodom u tekstualni oblik i obratno tijekom stvaranja, odnosno obrađivanja SOAP poruke automatizirano je mehanizmima .NET radnog okvira. Moguće je da datoteke instalacijskog paketa usluge postanu

prevelike za prijenos jednom SOAP porukom pa je stoga omogućen prijenos instalacijskog paketa u više dijelova. Veličina dijela instalacijskog paketa zadaje se u konfiguracijskoj datoteci usluge *Skladište*, pri čemu je podrazumijevana vrijednost 1 MB.



Slika 7-9: Vremenski dijagram operacija spremanja i dohvata usluge

Slika 7-9 prikazuje vremenski dijagram operacija spremanja i dohvata usluge. Operaciju spremanja instalacijskog paketa usluge inicira administrator *VDE* računalne okoline putem Web stranica. Binarne datoteke instalacijskog paketa se prilikom prijenosa do usluge *Skladište* na prikidan način spremaju u tekstualnu SOAP poruku. Iz primljene SOAP poruke *Web sučelje* izvuče binarne podatke instalacijskog paketa te ih proslijedi *Upravljačkom modulu* koji datoteke instalacijskog paketa spremi u datotečni sustav lokalnog računala. Rezultat spremanja su tri nove datoteke: paket datoteka, instalacijska skripta i opisnik usluge. *Upravljački modul* nakon spremanja instalacijskog paketa u datotečni sustav upisuje lokacije novih datoteka u popis instalacijskih paketa. Ako je u konfiguracijskoj datoteci navedena uporaba relacijske baze podataka za održavanje popisa instalacijskih paketa, onda *Upravljački modul* šalje popis lokacija novih datoteka *Modulu za pristup bazi podataka* koji ih spremi u relacijsku bazu podataka. Ako je u konfiguracijskoj datoteci navedena uporaba XML datoteke, onda *Upravljački modul* šalje popis lokacija novih datoteka *Modulu za pristup XML datoteci* koji ih spremi u pripadnu XML datoteku. Nakon uspješnog spremanja lokacije instalacijskog paketa, *Upravljački modul* putem

Zastupnika usluge Katalog obavještava uslugu *Katalog* o novoj spremljenoj aplikacijskoj usluzi. Web sučelje na kraju uspješnog spremanja usluge odgovara korisniku potvrđnom porukom.

Operaciju dohvaćanja instalacijskog paketa obično započinje usluga *Postavljač* tijekom procesa postavljanja aplikacijske usluge na čvor računalne okoline. *Web sučelje* primi SOAP poruku sa zahtjevom za određenom datotekom instalacijskog paketa spremljene usluge. *Web sučelje* proslijedi zahtjev *Upravljačkom modulu* koji putem odgovarajućeg pristupnog modula iz popisa instalacijskih paketa dohvati lokaciju tražene datoteke. Nakon što sazna lokaciju datoteke, *Upravljački modul* datoteku učita sa diska u radnu memoriju te ju proslijedi *Web sučelju*. *Web sučelje* učitane binarne datoteke pretvara u tekstualnu SOAP poruku koju šalje korisniku.

7.3.2 Struktura instalacijske skripte

Postupak postavljanja aplikacijskih usluga opisuje se posebno oblikovanom instalacijskom skriptom koja je prilagođena potrebama *VDE* računalne okoline. Postupak postavljanja usluge osjetljiv je na specifičnosti programskog ostvarenja usluge te je ovisan o specifičnostima operacijskog sustava na kojem se usluga izvodi. Prilikom definiranja instalacijske skripte pazilo se da konstrukti skripte budu što manje vezani uz specifičnosti .NET računalne platforme. Interpretacija instalacijske skripte opisuje se unutar odjeljaka 7.5.2 i 7.5.3 u sklopu definicija postupaka postavljanja i uklanjanja aplikacijskih usluga.

<?xml version="1.0" encoding="us-ascii"?> <ServiceInstallScript ServiceID="MailBox" ServiceDescription="MailBox Coopetition service.">	
<DeploymentInstructions> <Directory Source="bin" Destination="bin" /> <File Source="MailBox.asmx" Destination="MailBox.asmx" /> <File Source="Web.config" Destination="Web.config" /> <File Source="Container.exe" Destination="Container.exe" /> <File Source="Global.asax" Destination="Global.asax" /> <VirDir Name="MailBox" Path=".." /> </DeploymentInstructions>	Uputa za instalaciju
<ConfigurationInfo> <ConfigFile Path="Web.config" /> </ConfigurationInfo>	Podaci o konfiguraciji
<ServiceExecutionInfo> <Executable Path="Container.exe" /> <StartPage Page="MailBox.asmx" /> </ServiceExecutionInfo>	Podaci o izvođenju
</ServiceInstallScript>	

Slika 7-10: Primjer instalacijske skripte

Slika 7-10 prikazuje primjer instalacijske skripte u kojoj se koriste svi trenutno podržani konstrukti. Skripta je zasnovana na XML formatu koji pruža platformsku neovisnost. Korijenski element skripte definira identifikator usluge i kratak tekstualni opis usluge. Ovaj primjer prikazuje instalacijsku skriptu za uslugu *poštanski pretinac* koja se koristi kao mehanizam za

asinkronu komunikaciju između usluga u komunikacijskom modelu zasnovanom na porukama [101]. Unutar korijenskog elementa se konstrukti instalacijske skripte grupiraju u tri skupine: *uputa za instalaciju, podaci o konfiguraciji usluge i podaci o izvođenju usluge*.

Uputa za instalaciju smješta se unutar *<DeploymentInstructions>* elementa. Njena uloga je definiranje datoteka i direktorija iz paketa datoteka usluga koji se preslikavaju na lokaciju postavljene usluge. Element *<File>* određuje postupak preslikavanja datoteke iz paketa datoteka na lokaciju postavljene usluge. Element *<Directory>* određuje postupak preslikavanja direktorija datoteka iz paketa datoteka na lokaciju postavljene usluge. Prilikom preslikavanja direktorija se preslikava cijela struktura direktorija zajedno s poddirektorijima i pripadnim datotekama. Poseban konstrukt iz ove skupine je element *<VirDir>* kojim se definiraju parametri *prividnog direktorija* (engl. virtual directory). Prividni direktorij određuje ugradnju aplikacijske usluge ostvarene .NET radnim okvirom u IIS Web poslužitelj. Detalji o povezivanju usluge i Web poslužitelja nalazi se u odjeljku 7.5.2. Prividni direktoriji su specifičnost .NET računalne okoline pa je samim time i instalacijska skripta ovisna o računalnoj platformi.

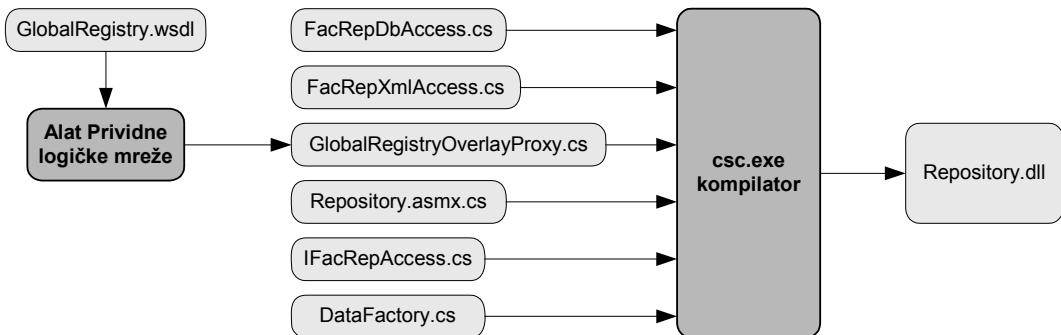
Podaci o konfiguraciji usluge smješteni su unutar *<ConfigurationInfo>* elementa. Trenutno postoji samo jedan konstrukt instalacijske skripte koji određuje konfiguraciju usluge. Element *<ConfigurationFile>* definira konfiguracijsku datoteku usluge koja se preslikava iz paketa datoteka na lokaciju postavljene usluge.

Podaci o izvođenju usluge smješteni su unutar *<ServiceExecutionInfo>* elementa. Element *<Executable>* definira ime izvršne datoteke procesa koji je potreban usluzi za izvođenje. Moguće je definirati više dodatnih procesa aplikacijske usluge. Ovaj element je opcionalan. Pretpostavlja se da su izvršne datoteke preslikane iz paketa datoteka na lokaciju postavljene usluge. Element *<StartPage>* određuje početnu stranicu aplikacijske usluge. Ovaj element je obavezan i može se pojaviti samo jednom unutar instalacijske skripte.

7.3.3 Ostvarenje

Usluga *Skladište* ostvarena je tehnologijama .NET radnog okvira. Izvorni kôd u potpunosti je napisan u C# programskom jeziku, a preveden je *csc.exe* prevodiocem. Slika 7-11 prikazuje datoteke izvornog kôda usluge *Skladište* u procesu prevodenja u izvršni oblik. U datoteci *Repository.asmx.cs* smješteni su izvorni kôdovi *Web sučelja* i *Upravljačkog modula* usluge *Skladište*. Izvorni kôd *Modula za pristup bazi podataka* smješten je u datoteci *FacRepDbAccess.cs*. Ostvarenje *Modula za pristup bazi podataka* zasniva se na ADO.NET programskom modelu za pristup bazama podataka koji omogućuje jedinstven način pristupa relacijskim bazama podataka. Zbog prilagodljivosti ADO.NET programskog modela, putem

Modula za pristup bazi podataka može se pristupati svim sustavima za upravljanje relacijskim bazama podataka koje posjeduju ODBC (Open DataBase Connectivity) sučelje [113]. Usluga *Skladište* ispitana je s *Microsoft SQL Server* sustavom za upravljanje relacijskom bazom podataka. Datoteka *DataFactory.cs* sadrži izvorni kôd čestih rutina za pristup bazi podataka, poput otvaranja veze prema relacijskoj bazi podataka i slično. Izvorni kôd *Modula za pristup XML katalogu* ostvaren je u datoteci *FacRepXmlAccess.cs*. Oba modula za pristup podacima ostvaruju isto programsko sučelje koje je definirano u datoteci *IFacRepAccess.cs*. Pošto oba modula za pristup podacima ostvaruju isto programsko sučelje, *Upravljački modul* im pristupa na identičan način. Izvorni kôd *Zastupnika usluge Katalog* nalazi se u datoteci *GlobalRegistryOverlayProxy.cs*. Izvorni kôd datoteke *GlobalRegistryOverlayProxy.cs* automatski je stvoren na osnovi WSDL opisa usluge *Globalni katalog* uporabom alata *Prividne logičke mreže*. WSDL opis usluge *Globalni katalog* nalazi se unutar datoteke *GlobalRegistry.wsdl*. Prevođenjem izvornog kôda usluge *Skladište* nastaje izvršna datoteka *Repository.dll* koja se putem prividnih direktorija ugrađuje u IIS poslužitelj.



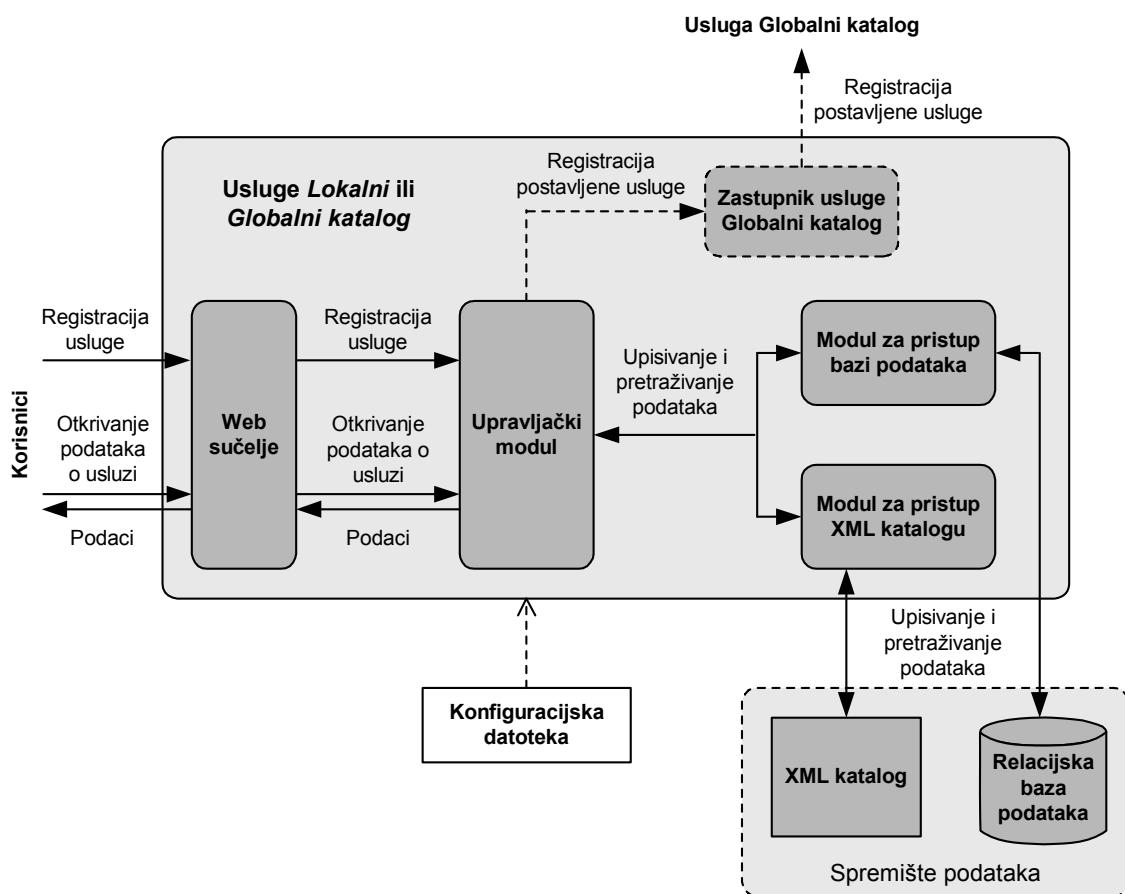
Slika 7-11: Prevodenje izvornog kôda usluge *Skladište*

7.4 Usluga *Katalog*

Usluga *Katalog* je složena usluga zadužena za održavanje informacija o aplikacijskim uslugama dostupnim unutar *VDE* računalne okoline. *Katalog* je organiziran u dvorazinsku hijerarhijsku strukturu. Na dnu hijerarhije nalaze se usluge *Lokalni katalog* koje su postavljene na svakom čvoru *VDE* računalne okoline. *Lokalni katalog* održava informacije o uslugama smještenim na lokalnom čvoru. Na vrhu hijerarhije nalazi se usluga *Globalni katalog* koja se postavlja na jedan unaprijed odabrani čvor. *Globalni katalog* održava informacije o spremlijenim i postavljenim oblicima usluga prisutnim unutar *VDE* računalne okoline.

7.4.1 Programska arhitektura

Usluge *Lokalni* i *Globalni katalog* slično su građene. Na slici 7-12 prikazana je zajednička programska arhitektura obje usluge. Razlika između usluga je u podacima koji se u njih spremaju. Usluga *Lokalni katalog* spremi podatke o postavljenim i instanciranim uslugama na lokalnom čvoru, dok usluga *Globalni katalog* spremi podatke o spremljenim i postavljenim uslugama unutar *VDE* računalne okoline. Zajednička programska arhitektura usluga *Lokalni* i *Globalni katalog* sastoji se od četiri dijela: *Web sučelje*, *Upravljački modul*, *Modul za pristup bazi podataka* i *Modul za pristup XML katalogu*. Zastupnika usluge *Globalni katalog* sadrži samo usluga *Lokalni katalog*.



Slika 7-12: Programska arhitektura usluga *Lokalni* ili *Globalni katalog*

Web sučelje je pristupna točka usluga kataloga koja prihvata SOAP poruke korisničkih zahtjeva. Korisnički zahtjevi se dijele u dvije skupine: zahtjevi za registracijom usluge i zahtjevi za otkrivanjem podataka o usluzi. Zahtjevi za registracijom usluge podrazumijevaju upis podataka u katalog, dok zahtjevi za otkrivanjem podataka o usluzi podrazumijevaju pretraživanje kataloga. Metode *Web sučelja* pokrivaju samo jednostavne upite za podacima. Složene upite za

podacima korisnik ostvaruje s nekoliko poziva različitih metoda *Web sučelja*. Rezultat obrade korisničkih zahtjeva *Web sučelje* vraća u obliku SOAP poruke.

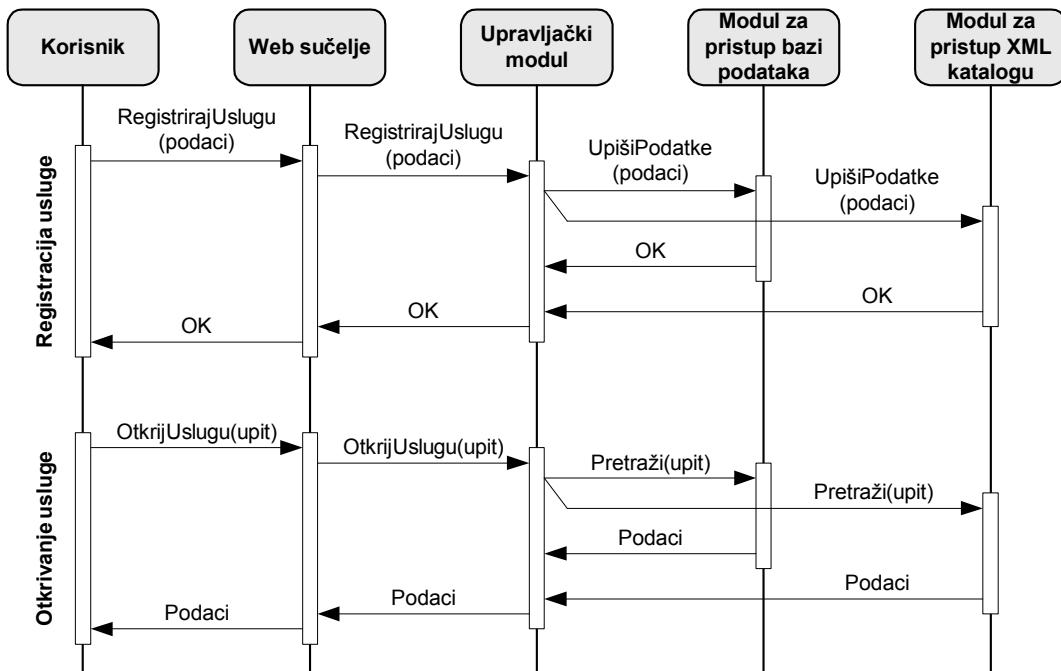
Upravljački modul nadzire upisivanje i pretraživanje podataka u pripadnom spremištu podataka. Ako se podaci upisuju na više mesta unutar spremišta podataka i ako jedan od koraka upisa ne uspije, onda *Upravljački modul* poništava cijeli postupak i vraća spremište podataka u prijašnje stanje. Oblikovana su dva oblika spremišta podataka: XML katalog i relacijska baza podataka. Usluge kataloga istovremeno koriste samo jedan oblik spremišta podataka. Korišteni oblik spremišta podataka definira se u konfiguracijskoj datoteci tijekom postupka instalacije usluga. Podrazumijevani oblik je XML katalog. *Upravljački modul* pristupa spremištu podataka putem pripadnih pristupnih modula.

Modul za pristup bazi podataka pristupa relacijskoj bazi podataka. Pristup podacima u relacijskoj bazi podataka ostvaruje se SQL upitima. Tijekom upisa podataka u više tablica relacijske baze podataka upotrebljavaju se mehanizmi transakcija koji u slučaju pogreške omogućuje poništavanje djelomično upisanih podataka. *Modul za pristup XML katalogu* pristupa XML spremištu podataka. XML spremište podataka organizirano je u XML strukturu. *Modul za pristup XML katalogu* koristi XPath jezik za pretraživanje podataka u XML katalogu. XPath jezik [112] omogućuje postavljanje upita za odabir podskupa elemenata iz XML dokumenta.

Zastupnik usluge Globalni katalog sastavni je dio usluge *Lokalni katalog*. Uloga *Zastupnika usluge Globalni katalog* je posredovanje u komunikaciji između usluga *Lokalni katalog* i *Globalni katalog*. Povezanost između usluga *Lokalni katalog* i *Globalni katalog* opisana je u odjeljku 6.3.2.

Slika 7-13 prikazuje vremenske dijagrame međudjelovanja sastavnih dijelova usluga *Lokalni* i *Globalni katalog* tijekom registracije i otkrivanja podataka o aplikacijskim uslugama. *Lokalni* i *Globalni katalog* pružaju skup metoda za registraciju različitih oblika usluga. Sve metode registracije usluga ostvarene su sličnim slijedom akcija unutar usluga kataloga pa ih se na vremenskom dijagramu prikazuje jedinstvenom metodom *RegistrirajUslugu*. Prilikom registracije usluge korisnik šalje SOAP poruku zahtjeva na *Web sučelje*. *Web sučelje* na osnovi SOAP poruke poziva odgovarajuću metodu *Upravljačkog modula*. Ovisno prema podacima u konfiguracijskoj datoteci, *Upravljački modul* učitava odgovarajući modul za pristup spremištu podataka i poziva prikladnu metodu modula za upis podataka u spremište podataka. Prilikom registracije postavljene usluge u *Lokalni katalog*, *Upravljački modul* poziva *Zastupnik usluge Globalni katalog* koji obavijesti uslugu *Globalni katalog* o novoj postavljenoj usluzi unutar *VDE* računalne okoline. Ako je postupak registracije uspješno izveden, onda se korisniku vraća SOAP

poruka potvrđnog odgovora. Ako je tijekom postupka registracije došlo do pogreške, onda se korisniku vraća SOAP poruka s detaljnim opisom pogreške.

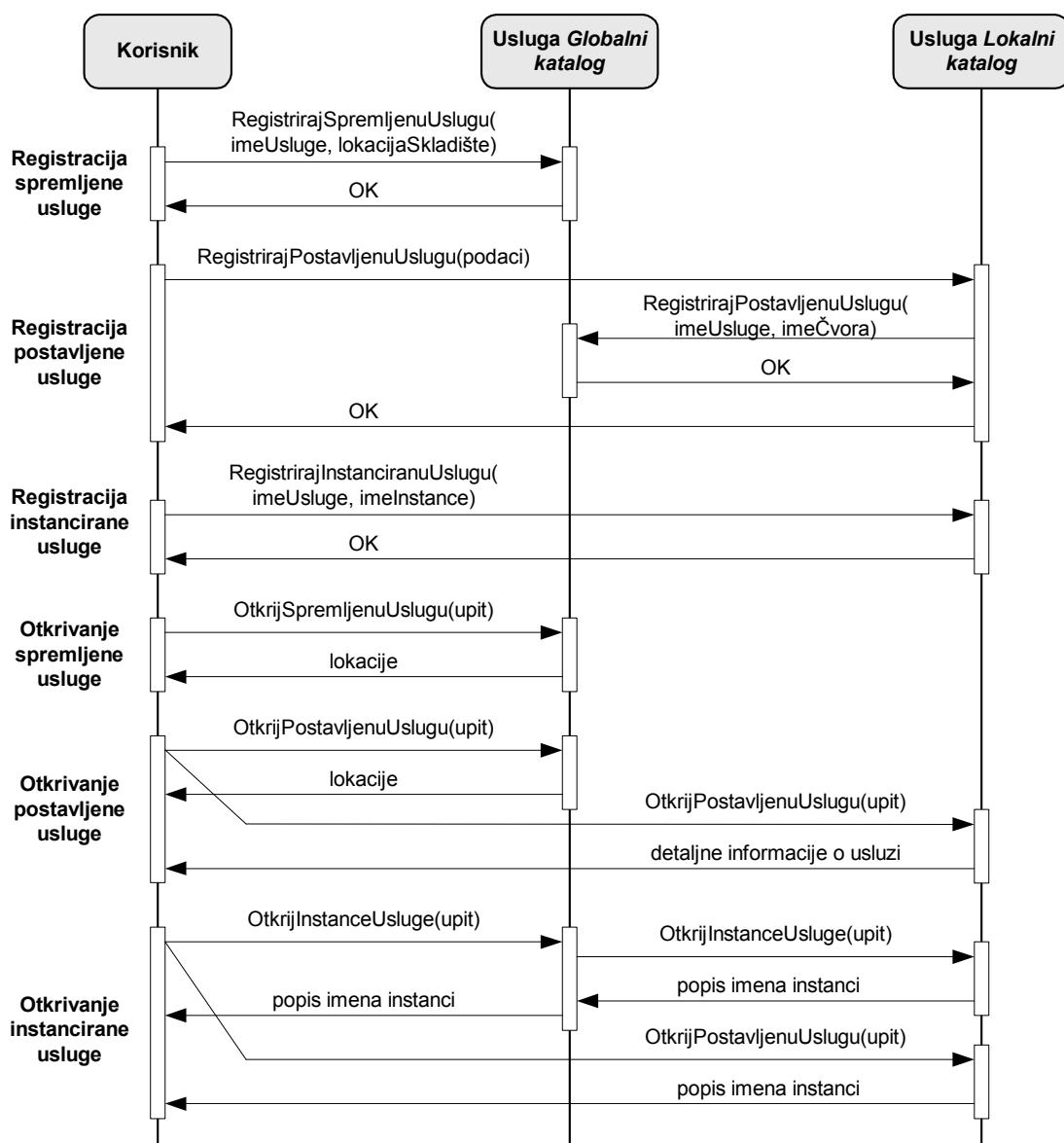


Slika 7-13: Vremenski dijagrami međudjelovanja sastavnih dijelova *Lokalnog* i *Globalnog kataloga*

Sve metode otkrivanja usluga su na vremenskom dijagramu prikazane jedinstvenom metodom *OtkrijUslugu*. Postupak otkrivanja usluge započinje korisnik slanjem SOAP poruke na *Web sučelje* usluga kataloga. *Web sučelje* na osnovi SOAP poruke poziva odgovarajući metodu *Upravljačkog modula*. *Upravljački modul* prema podacima u konfiguracijskoj datoteci poziva odgovarajući modul za pristup spremištu podataka kataloga. Modul za pristup spremištu podataka dohvata podatke i šalje ih *Upravljačkom modulu*, koji ih proslijedi *Web sučelju*. *Web sučelje* pretvara objektni zapis podataka u XML strukturu koju ugrađuje u SOAP poruku odgovora i šalje korisniku.

Slika 7-14 prikazuje međudjelovanje usluga *Lokalni* i *Globalni katalog* tijekom registracije i otkrivanja podataka o aplikacijskim uslugama. Registraciju podataka o spremljenoj usluzi započinje usluga *Skladište* pozivom odgovarajuće metode *Globalnog kataloga*. Postavljenu uslugu registrira usluga *Postavljač* pozivom *Lokalnog kataloga* koji dio informacija proslijedi *Globalnom katalogu*. Registraciju instance inicira sama usluga prilikom stvaranja instance. Usluga stvorenu instancu registrira pozivom odgovarajuće metode *Lokalnog kataloga*. Otkrivanje informacija o oblicima aplikacijskih usluga omogućeno je svim korisnicima *VDE* računalne okoline. Otkrivanje podataka o spremljenim uslugama ostvaruje se pozivom *Globalnog kataloga*, pri čemu se kao rezultat vraća popis lokacija *Skladište* koja posjeduju

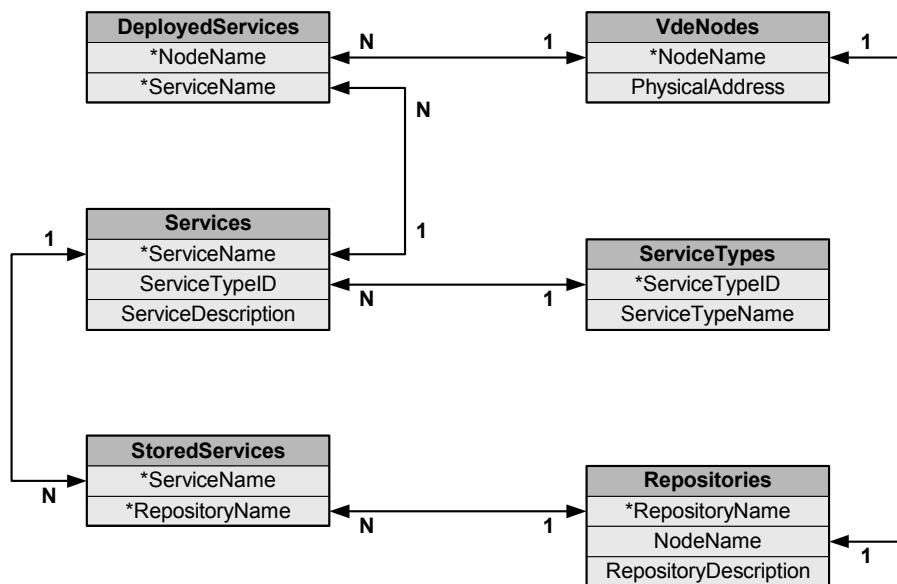
instalacijske pakete zadane aplikacijske usluge. Otkrivanje postavljenih usluga ostvaruje se pretraživanjem *Globalnog* ili *Lokalnog kataloga*. Prilikom pretraživanja *Globalnog kataloga* pronalaze se lokacije postavljenih usluga, dok se prilikom pretraživanja *Lokalnog kataloga* dohvaćaju detaljni podaci o postavljenim uslugama. Otkrivanje instanci usluge također se ostvaruje pretraživanjem *Globalnog* ili *Lokalnog kataloga*. Kod pretraživanja *Globalnog kataloga* se upit za instancom usluge prosljeđuje prikladnom *Lokalnom katalogu* koji vraća informacije oinstancama tražene aplikacijske usluge. Detaljne definicije metoda sučelja usluga *Lokalni* i *Globalni katalog* nalaze se u dodatku A.



Slika 7-14: Vremenski dijagrami medudjelovanja usluga *Lokalni* i *Globalni katalog*

7.4.2 Organizacija podataka u relacijskoj bazi podataka

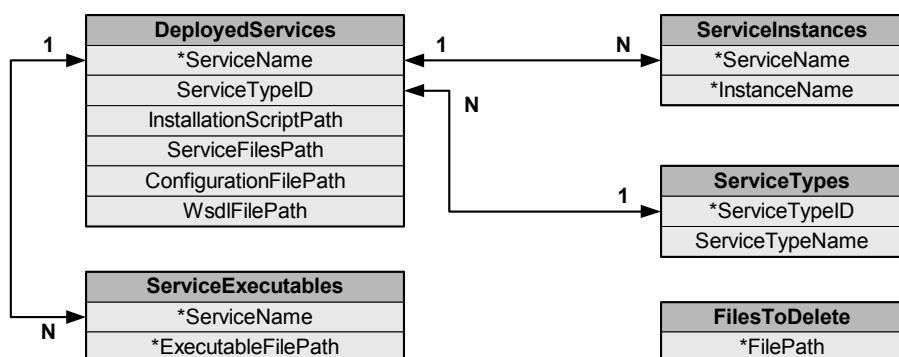
Relacijske baze podataka su zbog svojih svojstava prikladan izbor za ostvarenje spremišta podataka usluge *Katalog*. SQL jezik omogućuje jednostavno pronalaženje i promjenu podataka. Bez obzira na veličinu baze podataka, podaci se brzo pronađe. Sustav za upravljanje bazom podataka (RDBMS) osigurava da je baza uvijek u suvislom stanju. Transakcije osiguravaju izvođenje niza operacija kao jednu nedjeljivu cjelinu.



Slika 7-15: Organizacija relacijske baze podataka usluge *Globalni katalog*

Slika 7-15 prikazuje organizaciju relacijske baze podataka usluge *Globalni katalog*. Relacijska baza sastoji se od šest tablica: *VdeNodes*, *Services*, *ServiceTypes*, *Repositories*, *StoredServices* i *DeployedServices*. Baza podataka je normalizirana na treću normalnu formu kako bi se spriječila redundancija podataka. Tablica *VdeNodes* sadrži podatke o čvorovima *VDE* računalne okoline. Svaki čvor se opisuje imenom čvora (atribut *NodeName*) i fizičkom adresom pripadnog računala (atribut *PhysicalAddress*). Tablica *Services* sadrži općenite podatke o uslugama, poput imena usluge (atribut *ServiceName*), imena vrste usluge (atribut *ServiceTypeID*) i kratkog opisa usluge (atribut *ServiceDescription*). Vrsta usluge se definira unutar tablice *ServiceTypes*. Unutar tablice *Repositories* opisuju se usluge *Skladišta* dostupne unutar *VDE* računalne okoline. Svako *Skladište* određeno je vlastitim imenom (atribut *RepositoryName*), imenom čvora na kojem se nalazi (atribut *NodeName*) i kratkim opisom (atribut *RepositoryDescription*). Podaci o spremljenim uslugama smještaju se unutar tablice *StoredServices* u obliku uređenih parova imena usluge i imena skladišta. Podaci o postavljenim uslugama smještaju se unutar tablice *DeployedServices* u obliku uređenih parova imena čvora

(atribut *NodeName*) i imena usluge (atribut *ServiceName*). Elementi tablica označeni zvjezdicom (*) su primarni ključevi tablica. Veze između tablica označavaju odnose među podacima. Na primjer, tablice *VdeNodes* i *Repositories* povezane su 1-1 vezom, čime je određeno da se jedna vrijednost imena čvora može pojaviti u samo jednom retku tablice *Repositories*. Veza 1-N u slučaju tablica *ServiceTypes* i *Services* određuje da se isto ime vrste usluge može pojaviti u više različitim redaka tablice *Services*. RDBMS sustav osigurava očuvanje zadanih veza između tablica. Na primjer, RDBMS sustav onemogućava upis zapisa o usluzi u tablicu *Services* s imenom vrste usluge koja prethodno nije definirana unutar tablice *ServiceTypes*.



Slika 7-16: Organizacija relacijske baze podataka usluge *Lokalni katalog*

Slika 7-16 prikazuje organizaciju relacijske baze podataka usluge *Lokalni katalog*. Relacijska baza podataka sastoji se od četiri tablice: *DeployedServices*, *ServiceTypes*, *ServiceInstances*, *ServiceExecutables* i *FilesToDelete*. Tablica *DeployedServices* sadrži detaljne podatke o postavljenoj usluzi na lokalnom čvoru. Postavljena usluga opisuje se vlastitim imenom (atribut *ServiceName*), imenom vrste usluge (atribut *ServiceTypeID*), stazom instalacijske skripte (atribut *InstallationScriptPath*), lokacijom datoteka usluge u datotečnom sustavu (atribut *ServiceFilesPath*), stazom konfiguracijske datoteke usluge (atribut *ConfigurationFilePath*) i stazom datoteke s WSDL opisom usluge (atribut *WsdlFilePath*). Navedene detaljne podatke o postavljenoj usluzi koristi usluga *Postavljač* tijekom konfiguriranja i uklanjanja aplikacijske usluge. Vrsta usluge definira se unutar tablice *ServiceTypes*. Tablica *ServiceInstances* sadrži podatke o aktivnim instancama postavljenih usluga. Svaka instance definira se imenom usluge (atribut *ServiceName*) i imenom instance (atribut *InstanceName*). Unutar tablice *ServiceExecutables* definiraju se lokacije izvršnih datoteka dodatnih procesa usluge. Tablica *FilesToDelete* sadrži popis datoteka koje su neuspješno obrisane prilikom uklanjanja aplikacijske usluge. Podatke unutar tablice *FilesToDelete* upisuje i koristi isključivo usluga *Postavljač*.

7.4.3 Organizacija podataka u XML katalogu

Relacijske baze podataka nisu najprikladniji izbor za ostvarenje spremišta podataka. Bazu podataka potrebno je postaviti na računalo i povezati s aplikacijom koja ju koristi, a taj postupak nije moguće u potpunosti automatizirati. Postupak postavljanja baze podataka uključuje instalaciju RDBMS sustava te stvaranje tablica relacijske baze podataka. Također je aplikaciji potrebno unutar konfiguracijske datoteke zadati parametre pristupa bazi podataka. Pristupni parametri uključuju, na primjer, adresu poslužitelja baze podataka te korisničko ime i lozinku kojima se pristupa poslužitelju. Navedeni postupci zahtijevaju korisnikovo vrijeme i znanje. Problem posebno dolazi do izražaja tijekom postavljanja usluge *Lokalni katalog* koja se treba nalaziti na svim čvorovima raspodijeljene okoline.

S ciljem rješavanja opisanog problema razvijen je XML katalog koji se sastoji od skupa XML datoteka. Ako se koristi XML katalog kao spremište podataka, postupak postavljanja usluge *Katalog* uvelike je pojednostavljen i moguće ga je u potpunosti automatizirati. Postavljanje usluge *Katalog* svodi se na preslikavanje izvršnih datoteka usluge i praznih datoteka XML kataloga na čvor *VDE* računalne okoline.

```
<GlobalRegistry>
    <VdeNodes>
        <Node NodeName="..." PhysicalAddress="..."/>
    </VdeNodes>

    <Repositories>
        <Repository RepositoryName="..." NodeName="..." Description="..."/>
    </Repositories>

    <Services>
        <Service ServiceName="..." Description="..." ServiceTypeName="..."/>
        <StoredIn>
            <Repository RepositoryName="..."/>
        </StoredIn>

        <DeployedAt>
            <Node NodeName="..."/>
        </DeployedAt>
    </Service>
    </Services>
</GlobalRegistry>
```

Slika 7-17: Organizacija XML kataloga usluge *Globalni katalog*

Slika 7-17 prikazuje organizaciju XML kataloga usluge *Globalni katalog*. XML katalog usluge *Globalni katalog* održava identične podatke kao i spremište ostvareno relacijskom bazom podataka. Unutar korijenskog elementa *<GlobalRegistry>* nalaze se tri elementa: *<VdeNodes>*, *<Repositories>* i *<Services>*. Element *<VdeNodes>* sadrži podatke o čvorovima *VDE* računalne okoline, dok element *<Repositories>* sadrži podatke o uslugama *Skladište* dostupnim

unutar *VDE* računalne okoline. Element *<Services>* popisuje prisutne aplikacijske usluge. Svaka usluga posjeduje pripadni element *<Service>* u koji se putem atributa navode ime, opis i vrsta usluge. Popis usluga *Skladište* u kojima je spremljena pripadna usluga navodi se unutar elementa *<StoredIn>*. Unutar elementa *<DeployedAt>* navodi se popis čvorova *VDE* računalne okoline na koje je postavljena pripadna usluga. O suvislom stanju XML kataloga brine se *Modul za pristup XML katalogu*. Provjeravaju se tri svojstva suvislosti XML kataloga: ispravnost *<Repository>* elementa, ispravnost *<StoredIn>* elementa i ispravnost *<DeployedAt>* elementa. Kod *<Repository>* elementa se provjerava da li za vrijednost atributa *NodeName* postoji odgovarajući zapis unutar elementa *<VdeNodes>*. U slučaju *<StoredIn>* elementa se provjerava da li za vrijednost atributa *RepositoryName* postoji odgovarajući zapis unutar elementa *<Repositories>*. Kod *<DeployedAt>* elementa se provjerava da li za vrijednost atributa *NodeName* postoji odgovarajući zapis unutar elementa *<VdeNodes>*. Provjere se izvode tijekom upisa i čitanja podataka iz navedenih elemenata. Ako se ustanovi nesuvislo stanje XML kataloga, korisniku se vraća prikladna poruka greške. Tijekom upisa podataka u XML katalog dodatno se onemogućava upis jednakih zapisa. Na primjer, onemogućen je upis zapisa o čvoru *VDE* računalne okoline s imenom koje već postoji unutar elementa *<VdeNodes>*.

```

<LocalRegistry>
    <DeployedServices>
        <Service ServiceName="..." ServiceTypeName="..."
            InstallationScriptPath="..." ServiceFilesPath="..."
            ConfigurationFilePath="..." WsdlFilePath="...">
            <ServiceExecutables>
                <Executable ExecutableFilePath="..."/>
            </ServiceExecutables>
            <ServiceInstances>
                <Instance InstanceName="..."/>
            </ServiceInstances>
        </Service>
    </DeployedServices>
    <FilesToDelete>
        <File FilePath="..." />
    </FilesToDelete>
</LocalRegistry>

```

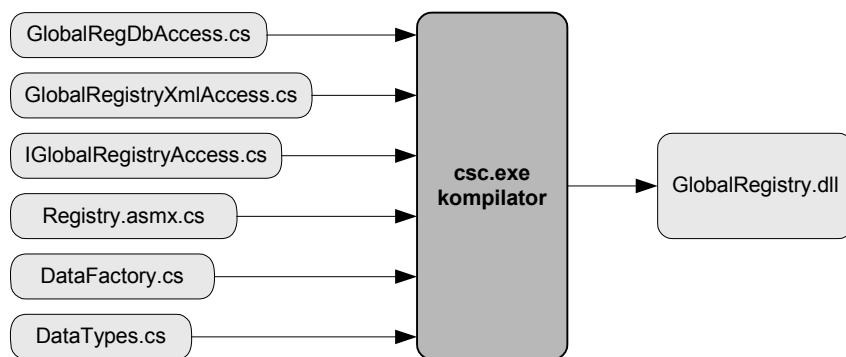
Slika 7-18: Organizacija XML kataloga usluge *Lokalni katalog*

Slika 7-18 prikazuje organizaciju XML kataloga usluge *Lokalni katalog*. XML katalog usluge *Lokalni katalog* održava identične podatke kao i spremište podataka *Lokalnog kataloga* ostvareno relacijskom bazom podataka. Unutar korijenskog elementa *<LocalRegistry>* nalaze se elementi *<DeployedServices>* i *<FilesToDelete>*. Unutar elementa *<DeployedServices>* postoji

po jedan element `<Service>` za svaku postavljenu uslugu. U element `<Service>` se putem pripadnih atributa navodi ime usluge, naziv vrste usluge, staza instalacijske skripte, lokacija izvršnih datoteka usluge u datotečnom sustavu, staza konfiguracijske datoteke i staza WSDL opisa usluge. Podaci o datotekama dodatnih procesa aplikacijske usluge navode se unutar elementa `<ServiceExecutables>`. Imena stvorenih instanci postavljene usluge spremaju se unutar elementa `<ServiceInstances>`. Element `<FilesToDelete>` sadrži podatke o neuspješno obrisanim datotekama tijekom operacije uklanjanja aplikacijske usluge. Podatke o neuspješno obrisanim datotekama isključivo koristi usluga *Postavljač*.

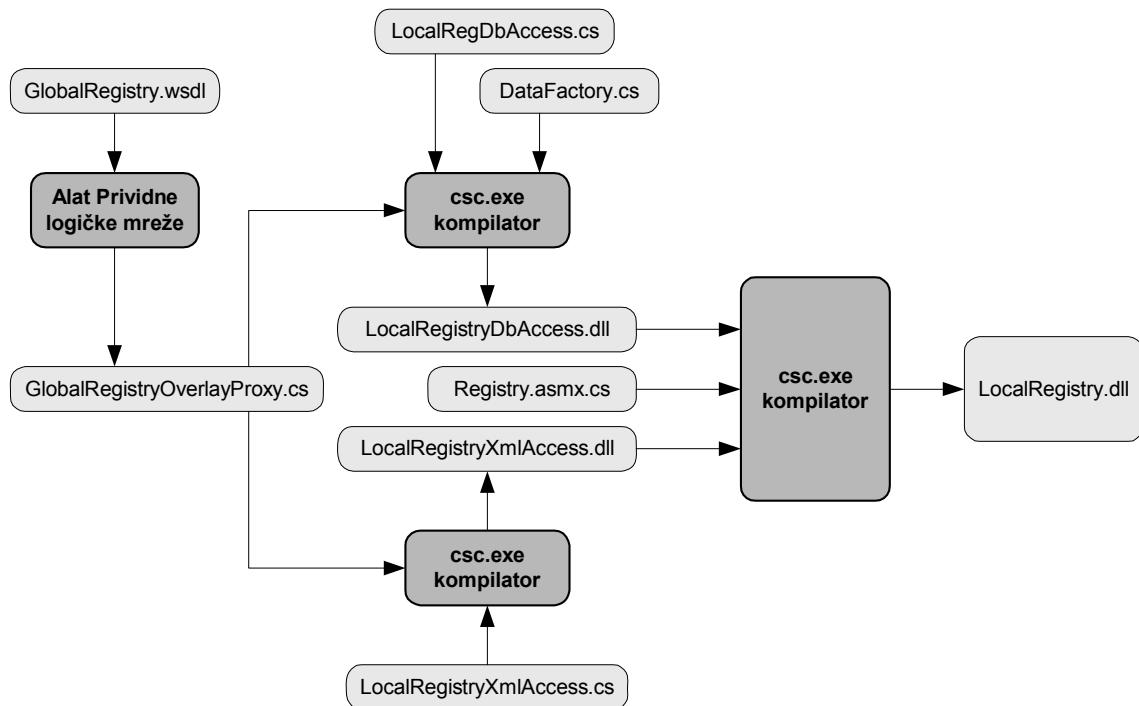
7.4.4 Ostvarenje

Usluga *Globalni katalog* je Web usluga ostvarena .NET tehnologijama. Slika 7-19 prikazuje datoteke izvornog kôda usluge *Globalni katalog* u postupku prevođenja u izvršni oblik.



Slika 7-19: Prevodenje izvornog kôda usluge *Globalni katalog*

Unutar datoteke *Registry.asmx.cs* smješteni su izvorni kôdovi *Web sučelja* i *Upravljačkog modula*. Datoteka *GlobalRegistryDbAccess.cs* sadrži ostvarenje *Modula za pristup bazi podataka*. Unutar datoteke *GlobbalRegistryXmlAccess.cs* nalazi se izvorni kôd *Modula za pristup XML katalogu*. Oba modula za pristup internom spremištu podataka ostvaruju isto programsko sučelje definirano u datoteci *IGlobalRegistryAccess.cs*. Budući da oba modula za pristup spremištu podataka ostvaruju isto programsko sučelje, *Upravljački modul* pristupa podacima na jednak način. Usluga *Globalni katalog* ispitana je s *Microsoft SQL Server* sustavom za relacijske baze podataka, ali se mogu koristiti i drugi sustavi relacijskih baza koji posjeduju ODBC sučelje [113]. Ulazne i izlazne strukture podataka usluge *Globalni katalog* definirane su u datoteci *DataTypes.cs*. Datoteka *DataFactory.cs* sadrži česte rutine za pristup bazi podataka, poput otvaranja veza prema relacijskoj bazi podataka i slično. Prevođenjem datoteka izvornog kôda nastaje datoteka *GlobalRegistry.dll* koja se ugrađuje u IIS Web poslužitelj i izvodi funkcionalnosti usluge *Globalni katalog*.



Slika 7-20: Prevodenje izvornog kôda usluge *Lokalni katalog*

Usluga *Lokalni katalog* također je ostvarena kao Web usluga .NET tehnologijama. Slika 7-20 prikazuje datoteke izvornog kôda usluge u postupku prevodenja. Unutar datoteke `Registry.asmx.cs` nalazi se izvorni kôd *Web sučelja* i *Kontrolnog modula* usluge *Lokalni katalog*. Logika pristupa spremištu podataka ostvarena je u zasebnim programskim knjižnicama. Programska knjižnica `LocalRegistryDbAccess.dll` pristupa spremištu podataka ostvarenom relacijskom bazom podataka. Ova knjižnica nastaje prevođenjem datoteke `LocalRegDbAccess.cs`, `DataFactory.cs` i `GlobalRegistryOverlayProxy.cs`. U datoteci `LocalRegDbAccess.cs` nalazi se izvorni kôd *Modula za pristup bazi podataka*. Datoteka `DataFactory.cs` sadrži često korištene rutine za pristup relacijskoj bazi podataka. Datoteka `GlobalRegistryOverlayProxy.cs` sadrži izvorni kôd *Zastupnika usluge Globalni katalog*. Ova datoteka stvorena je na osnovi WSDL datoteke usluge *Globalni katalog* uporabom posebnog alata *Prividne logičke mreže*. WSDL datoteka usluge *Globalni katalog* je `GlobalRegistry.wsdl`. Programska knjižnica `LocalRegistryXmlAccess.dll` pristupa spremištu podataka ostvarenom XML datotekama. Ova knjižnica nastaje prevođenjem datoteke `LocalRegistryXmlAccess.cs` i `GlobalRegistryOverlayProxy.cs`. Unutar datoteke `LocalRegistryXmlAccess.cs` nalazi se izvorni kôd *Modula za pristup XML katalogu*. Prevođenjem datoteke izvornog kôda i povezivanjem s knjižnicama za pristup spremištu podataka stvori se datoteka `LocalRegistry.dll` koja se ugrađuje u IIS poslužitelj i izvodi funkcionalnosti usluge *Globalni katalog*.

7.5 Usluga Postavljač

Usluga *Postavljač* ostvaruje funkcionalnosti postavljanja, uklanjanja i konfiguriranja aplikacijskih usluga na čvorove *VDE* računalne okoline. Na svakom čvoru *VDE* raspodijeljene okoline nalazi se preslika usluge *Postavljač*. U nastavku poglavila opisuje se programska arhitektura, navode se osnovni procesi usluge te se definira organizacija izvornog kôda usluge.

7.5.1 Programska arhitektura

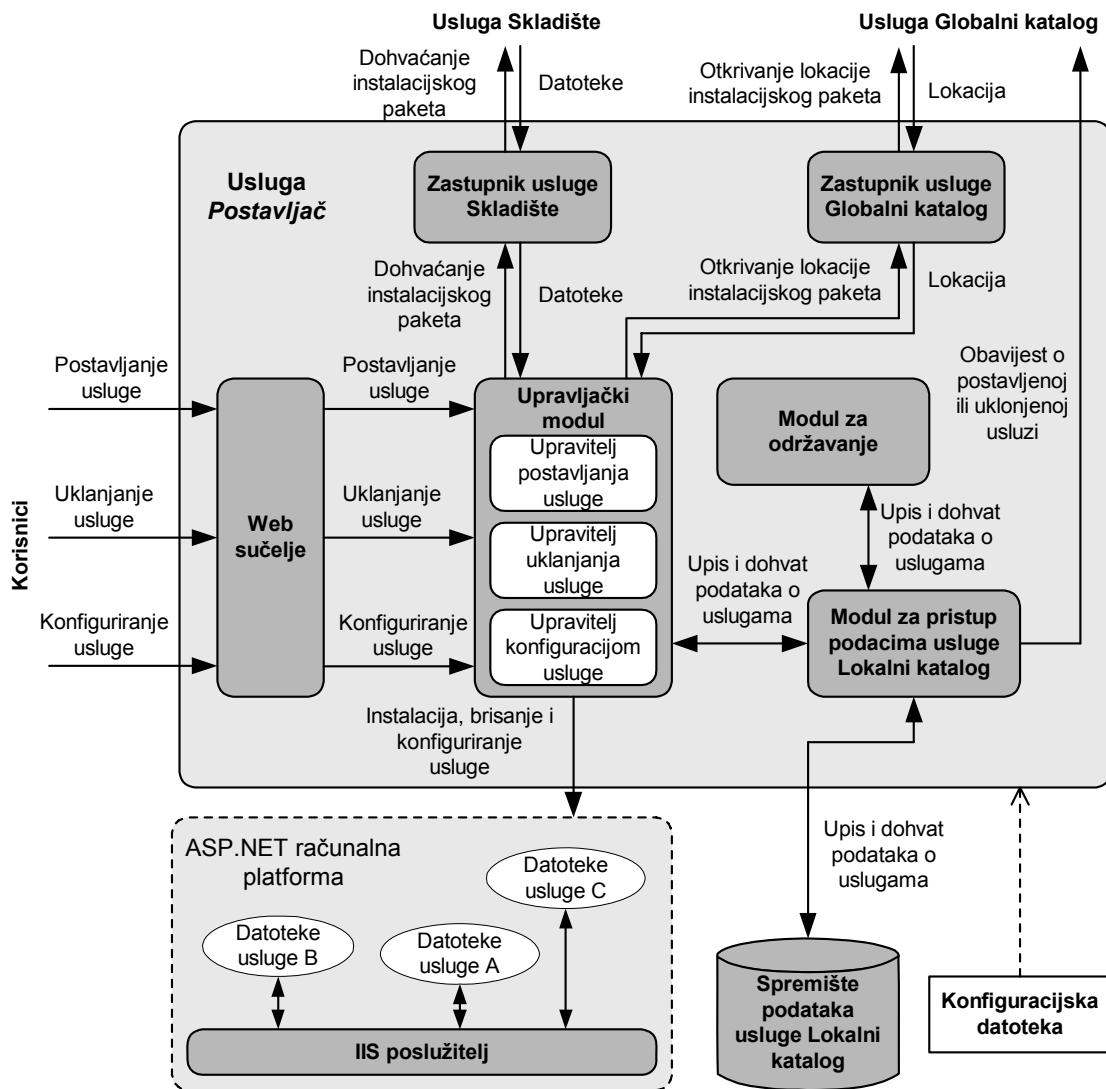
Slika 7-21 prikazuje arhitekturu programskog ostvarenja usluge *Postavljač*. Programska arhitektura usluga *Postavljač* sastoji se od sljedećih dijelova: *Web sučelje*, *Upravljački modul*, *Modul za pristup podacima usluge Lokalni katalog*, *Modul za održavanje*, *Zastupnik usluge Skladište* i *Zastupnik usluge Globalni katalog*.

Web sučelje prihvata SOAP poruke korisničkih zahtjeva. Usluga *Postavljač* prihvata korisničke zahtjeve za postavljanjem, uklanjanjem i konfiguriranjem aplikacijskih usluga. Na osnovi SOAP poruka korisničkih zahtjeva, *Web sučelje* poziva odgovarajuće metode *Upravljačkog modula* koji izvodi korisničke zahtjeve.

Upravljački modul se sastoji od *Upravitelja postavljanja usluge*, *Upravitelja uklanjanja usluge* i *Upravitelja konfiguracijom usluge*. *Upravitelj postavljanja usluge* izvodi postupak postavljanja aplikacijske usluge na lokalni čvor. *Upravitelj uklanjanja usluge* izvodi postupak uklanjanja usluge s lokalnog čvora. *Upravitelj konfiguracijom usluge* izvodi postupak promjene konfiguracijskih parametara postavljenih aplikacijskih usluga. Programska ostvarenja navedenih upravitelja prilagođena su potrebama aplikacijskih usluga ostvarenih .NET radnim okvirom prema programskoj arhitekturi opisanoj u odjeljku 7.2. Aplikacijske usluge ostvarene .NET radnim okvirom zahtijevaju za svoje izvođenje potporu IIS poslužitelja. Uloga *Upravljačkog modula* je, između ostalog, povezati izvršne datoteke aplikacijske usluge s IIS poslužiteljom.

Modul za pristup podacima usluge Lokalni katalog ostvaruje izravan pristup spremištu podataka usluge *Lokalni katalog*. Za uslugu *Postavljač* nužno je održavanje informacija o postavljenim aplikacijskim uslugama. Na primjer, *Upravljačkom modulu* je prilikom konfiguriranja aplikacijske usluge potrebna informacija o lokaciji konfiguracijske datoteke aplikacijske usluge. Pošto usluge *Postavljač* i *Lokalni katalog* dolaze u paru na čvorovima *VDE* računalne okoline, *Postavljač* sprema informacije o postavljenim aplikacijskim uslugama unutar spremišta podataka *Lokalnog kataloga*. Ovakav pristup omogućuje jednostavnije programsko ostvarenje usluge *Postavljač*, jer se izbjegava izgradnja posebnog spremišta podataka usluge *Postavljač*. Nadalje, prilikom registracije postavljene usluge izbjegnuta je spora komunikacija SOAP porukama između usluga *Postavljač* i *Lokalni katalog*, jer usluga *Postavljač* izravno

pristupa spremištu podataka usluge *Lokalni katalog* te upisuje podatke o postavljenoj usluzi. *Modul za pristup podacima usluge Lokalni katalog* sastoji se od programskih modula usluge *Lokalni katalog* opisanih u odjeljku 7.4.1. Jedino *Web sučelje* usluge *Lokalni katalog* nije dio *Modula za pristup podacima usluge Lokalni katalog*. Nadalje, ovaj modul obavještava uslugu *Globalni katalog* o postavljenim ili uklonjenim uslugama lokalnog čvora *VDE* računalne okoline.



Slika 7-21: Programska arhitektura usluge Postavljač

Modul za održavanje pokreće se tijekom inicijalizacije usluge *Postavljač*. Ovaj modul u pravilnim vremenskim razmacima provjerava aktivnost dodatnih procesa postavljenih aplikacijskih usluga. Popis dodatnih procesa postavljene usluge *Modul za održavanje* dohvata putem *Modula za pristup podacima usluge Lokalni katalog* iz spremišta podataka usluge *Lokalni katalog*. Ako neki od procesa nije aktivan, onda *Modul za održavanje* pokreće njegovu pripadnu

izvršnu datoteku. Nadalje, *Modul za održavanje* pokušava obrisati zaostale datoteka. Zaostale datoteke su datoteke koje iz nekog razloga prilikom uklanjanja usluge nisu uspješno obrisane. Popis zaostalih datoteka *Modul za održavanje* prikuplja putem *Modula za pristup podacima usluge Lokalni katalog* iz spremišta podataka usluge *Lokalni katalog*. Ako uspije obrisati zaostalu datoteku, *Modul za održavanje* izbriše njenu lokaciju iz popisa zaostalih datoteka u spremištu podataka usluge *Lokalni katalog*.

Zastupnik usluge Skladište i *Zastupnik usluge Globalni katalog* su posrednici u komunikaciji usluge *Postavljač* s uslugama *Skladište* i *Globalni katalog*. Zastupnici prevode međuobjektni poziv u prikladnu SOAP poruku zahtjeva koju šalju udaljenoj usluzi, a nakon toga primaju SOAP poruku odgovora koju pretvaraju u međuobjektnu poruku odgovora.

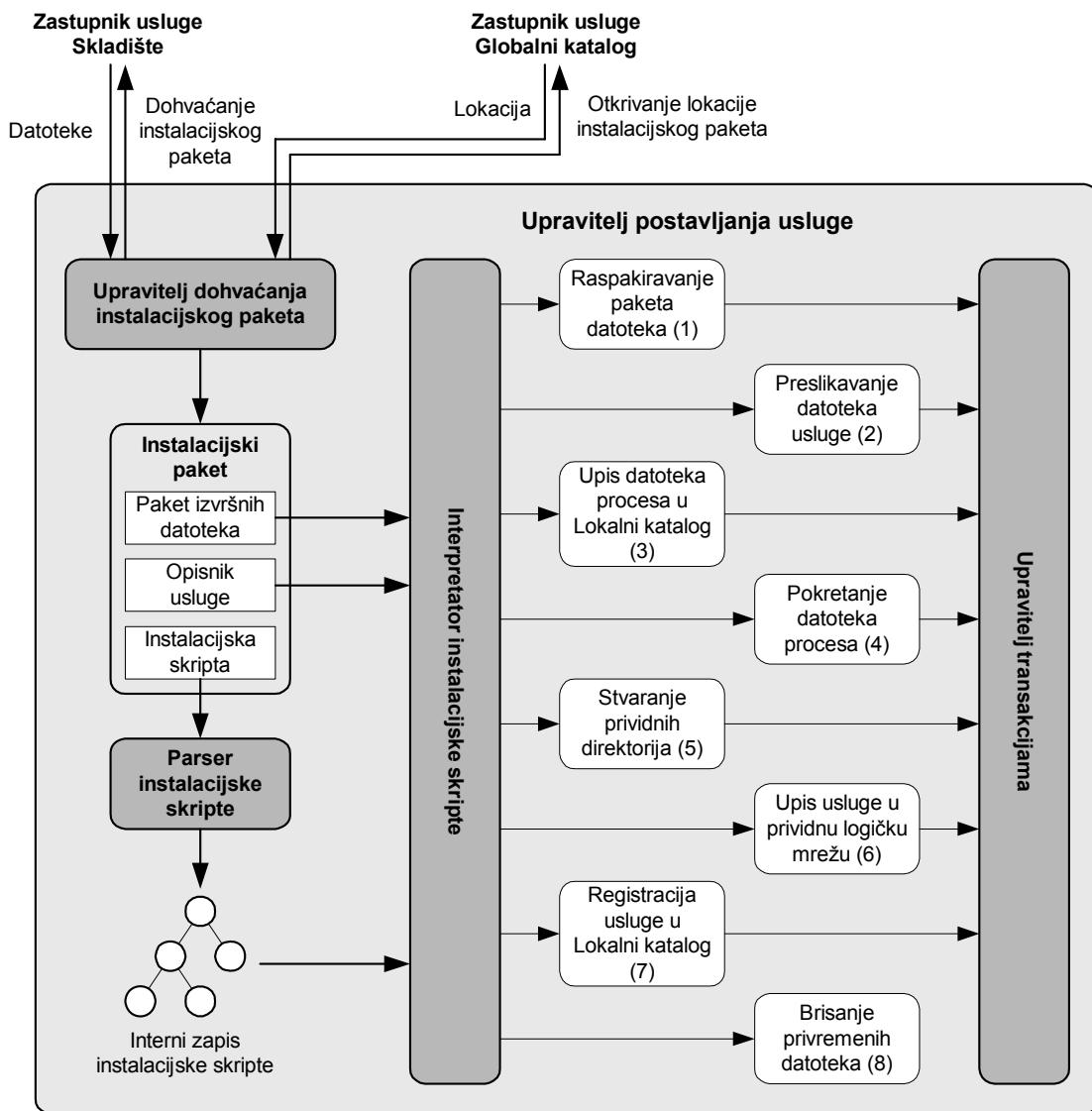
7.5.2 Postupak postavljanja aplikacijske usluge

Postupak postavljanja aplikacijske usluge najsloženiji je postupak u *Sustavu za postavljanje i otkrivanje usluga*. Postavljanje usluge izvodi *Upravitelj postavljanja usluge* koji je sastavni dio *Upravljačkog modula* usluge *Postavljač*. Slika 7-22 prikazuje programsku arhitekturu *Upravitelja postavljanja usluge*. Programska arhitektura *Upravitelja postavljanja usluge* sastoji se od sljedećih dijelova: *Upravitelj dohvaćanja instalacijskog paketa*, *Parsera instalacijske skripte*, *Interpretatora instalacijske skripte* i *Upravitelja transakcijama*.

Upravitelj dohvaćanja instalacijskog paketa izvodi dohvaćanje instalacijskog paketa aplikacijske usluge. Dohvaćanje instalacijskog paketa provodi se u dva koraka. U prvom koraku se putem usluge *Globalni katalog* otkrivaju lokacije svih preslika usluge *Skladište* u kojima je spremljen instalacijski paket usluge. Od skupa mogućih lokacija *Upravitelj dohvaćanja instalacijskog paketa* odabire jednu slučajnim odabirom. Slučajnim odabirom iz skupa mogućih lokacija osigurava se ravnomjerno korištenje preslika usluge *Skladište* (engl. load balancing). U drugom koraku *Upravitelj postavljanja usluge* komunicira s odabranom uslugom *Skladište* i dohvača instalacijski paket koji se sastoji od paketa izvršnih datoteka, opisnika usluge i instalacijske skripte.

Parser instalacijske skripte učitava instalacijsku skriptu u tekstualnom obliku i pretvara ju u prikladnu strukturu podataka. Podaci unutar instalacijske skripte organizirani su hijerarhijski što je posljedica XML formata zapisa podataka. Zbog hijerarhijske organizacije instalacijske skripte, *Parser instalacijske skripte* podatke pretvara u stablastu strukturu podataka. Čvorovi izgrađenog stabla su XML elementi i atributi instalacijske skripte. Programsko ostvarenje *Parsera instalacijske skripte* zasniva se na standardnim razredima .NET radnog okvira za obradu

XML dokumenata. Stablo instalacijske skripte također se izgrađuje uporabom standardnih razreda .NET radnog okvira.



Slika 7-22: Arhitektura upravitelja postavljanja usluge

Interpretator instalacijske skripte preuzima paket izvršnih datoteka, opisnik usluge i izgrađeno stablo instalacijske skripte te izvodi instalaciju aplikacijske usluge. Postupak instalacije izvodi se u osam koraka prikazanih na slici. Koraci postupka instalacije aplikacijske usluge uglavnom su upravljeni informacijama iz instalacijske skripte. Izgrađeno stablo instalacijske skripte omogućuje jednostavno dohvaćanje spremljenih informacija. Dohvaćanje informacija iz izgrađenog stabla izvodi se na osnovi imena traženog XML elementa.

Prvi korak postupka instalacije je raspakiravanje sažetog paketa datoteka aplikacijske usluge u privremeno stvoreni direktorij (1). Za svaki postupak postavljanja aplikacijske usluge

stvara se novi privremeni direktorij. U sljedećem koraku instalacije, datoteke nastale raspakiravanjem paketa datoteka preslikavaju se u direktorij aplikacijske usluge (2). Direktorij aplikacijske usluge naziva se istim imenom kao i aplikacijska usluga. U ovom koraku instalacije usluge se iz izgrađenog stabla instalacijske skripte dohvaćaju elementi s imenom *<File>* i *<Dir>*. Navedeni elementi definiraju imena datoteka i direktorija koji se iz privremenog direktorija preslikavaju u direktorij aplikacijske usluge.

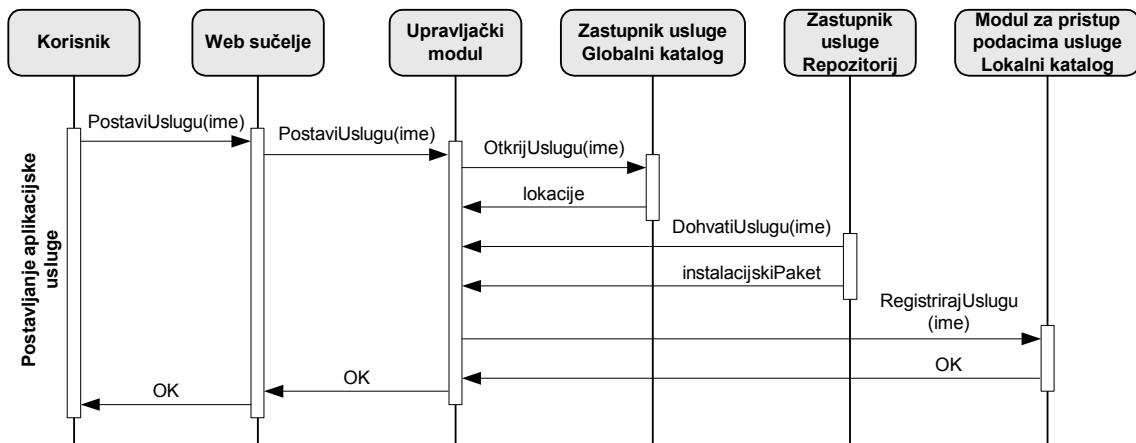
Nakon uspješnog preslikavanja datoteka slijedi upis datoteka procesa aplikacijske usluge u spremište podataka *Lokalnog kataloga* (3). U ovom koraku instalacije se iz stabla instalacijske skripte dohvaćaju elementi s imenom *<Executable>* koji sadrže informacije o datotekama dodatnih procesa usluge. *Interpretator instalacijske skripte* putem *Modula za pristup podacima usluge Lokalni katalog* zapisuje u spremište podataka *Lokalnog kataloga* lokacije izvršnih datoteka aplikacijske usluge. Nakon uspješnog upisa informacija u spremište podataka *Lokalnog kataloga* pokreću se datoteke dodatnih procesa aplikacijske usluge (4). Spremljene lokacije datoteka procesa čitaju se tijekom inicijalizacije usluge *Postavljač* kako bi se osiguralo ispravno izvođenje aplikacijskih usluga. Za pokretanje datoteka procesa zadužen je *Modul za održavanje usluge Postavljač*.

Sljedeći korak u postupku postavljanja aplikacijske usluge je definiranje prividnih direktorija (5). Definiranjem prividnih direktorija se aplikacijska usluga povezuje s IIS poslužiteljom putem kojeg pruža svoje funkcionalnosti. Prividni direktorij je zapis u internoj strukturi podataka IIS poslužitelja koji određuje prividnu i fizičku lokaciju usluge. Korisnički zahtjevi koji dolaze na IIS poslužitelj sadrže identifikator usluge za koju su namijenjeni. Dio tog identifikatora je prividna lokacija usluge na računalu. Putem te prividne lokacije i zapisa o prividnom direktoriju, IIS poslužitelj određuje fizičku lokaciju usluge na koju se zahtjev odnosi, pokreće uslugu ako već nije pokrenuta i prosljeđuje joj korisnički zahtjev. *Interpretator instalacijske skripte* u ovom koraku iz stabla instalacijske skripte dohvaća elemente s imenom *<VirDir>*. Navedeni elementi sadrže informacije o imenu prividnog direktorija te relativnu lokaciju fizičkog direktorija.

Nakon definiranja prividnih direktorija izvodi se registracija aplikacijske usluge u *Prividnu logičku mrežu* (6). *Prividna logička mreža* ima potrebnu infrastrukturu na svakom čvoru VDE računalne okoline. Uloga *Prividne logičke mreže* je posredovanje u komunikaciji između aplikacijskih usluga. Registracijom postavljene usluge omogućuje se pristupanje usluzi putem *Prividne logičke mreže*. Detalji o postupku registracije usluge u *Prividnu logičku mrežu* nalaze se u radu [93].

Sljedeći korak postupka instalacije je registracija aplikacijske usluge u *Lokalni katalog* (7). Registracija se ostvaruje putem *Modula za pristup podacima usluge Lokalni katalog* koji izravno upisuje podatke o usluzi u spremište podataka usluge *Lokalni katalog*. Tijekom više istovremenih pristupa spremištu podataka koriste se sinkronizacijski mehanizmi koji osiguravaju pravilan pristup spremištu. Ako je spremište podataka relacijska baza podataka, onda se upotrebljavaju sinkronizacijski mehanizmi pripadnog RDBMS sustava. Ako se pak pristupa XML spremištu podataka, onda se sinkronizacijskim mehanizmima .NET radnog okvira pristup XML spremištu podataka ograničava na samo jednu dretvu istovremeno. Nadalje, *Modul za pristup podacima usluge Lokalni katalog* obavještava uslugu *Globalni katalog* o novo postavljenoj aplikacijskoj usluzi. Na kraju ove akcije aplikacijska usluga je postavljena na računalo i spremna za korištenje.

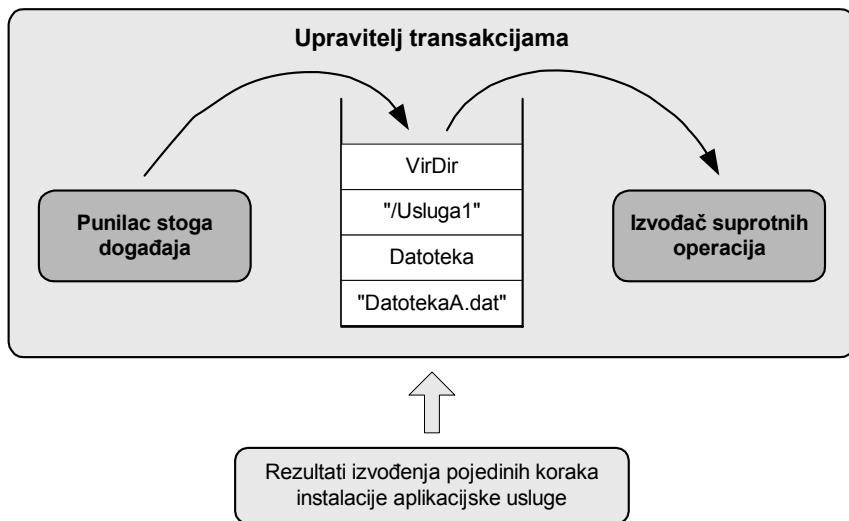
Posljednji korak u postupku instalacije je brisanje privremenih datoteka nastalih od pakiravanjem paketa datoteka (8). Slika 7-23 prikazuje na vremenskom dijagramu međudjelovanje pojedinih dijelova usluge *Postavljač* tijekom postavljanja aplikacijske usluge.



Slika 7-23: Vremenski dijagram postupka postavljanja usluge

Postupak postavljanja usluge ostvaren je u obliku transakcije, odnosno postupak se provodi u potpunosti ili nikako. U slučaju pogreške u samo jednom koraku postavljanja, poništavaju se svi prethodni koraci i računalo se dovodi u stanje u kojem je bilo prije početka postavljanja usluge. Postupak postavljanja nadgleda *Upravitelj transakcijama* koji je sastavni dio *Upravitelja postavljanja usluge*. Tijekom svakog koraka instalacije usluge obavještava se *Upravitelja transakcijama* o uspješno i neuspješno izvedenim akcijama. Jedino se tijekom brisanja privremenih datoteka ne obavještava *Upravitelja transakcijama* o uspješnosti izvedenih operacija jer u tom trenutku aplikacijska usluga je u potpunosti postavljena na korisničko računalo i spremna za izvođenje. Ako brisanje privremenih datoteka iz nekog razloga ne uspije, onda se lokacije neuspješno obrisanih datoteka putem *Modula za pristup podacima usluge*

Lokalni katalog zapisuju u spremište podataka usluge *Lokalni katalog. Modul za održavanje* naknadno pokušava obrisati zaostale datoteke. Slika 7-24 prikazuje arhitekturu *Upravitelja transakcijama*.



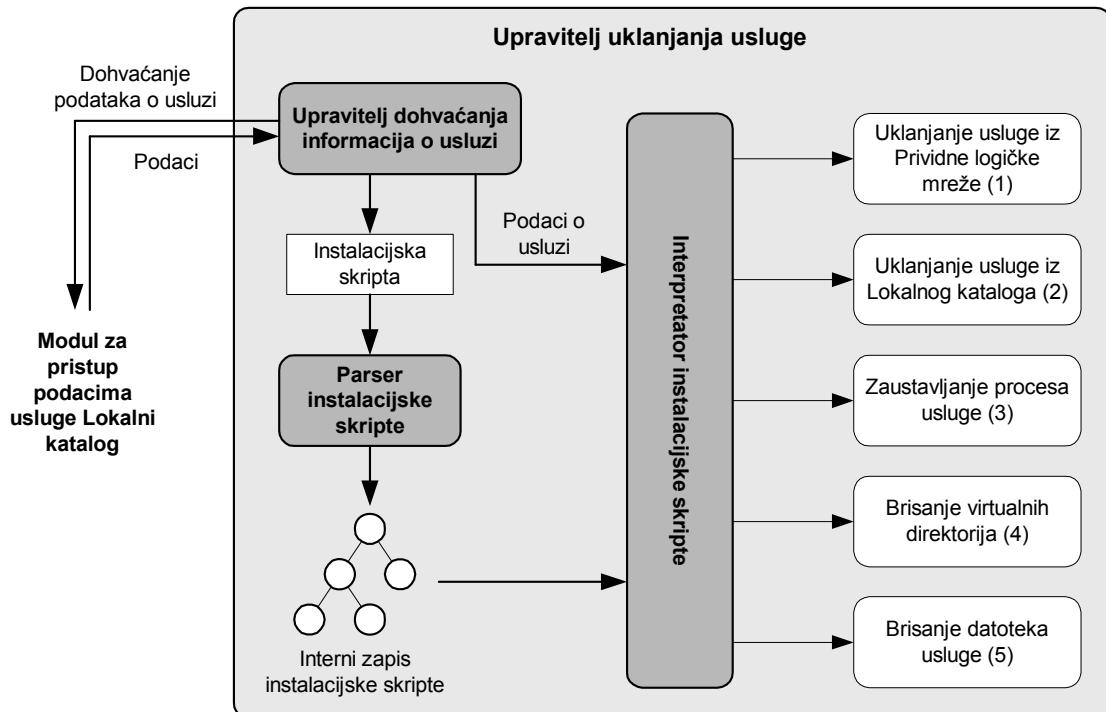
Slika 7-24: Arhitektura *Upravitelja transakcijama*

Punitelj stoga događaja prihvata informacije o uspješno izvedenim akcijama instalacije aplikacijske usluge te ih stavlja na stog događaja. Za jednu uspješno izvedenu akciju obično se na stog događaja stavlja više podataka, pri čemu se prvo stavljuju parametri akcije, a tek na kraju oznaka akcije. Na primjer, ako je uspješno preslikana datoteka "DatotekaA.dat", onda *Punilac stoga događaja* stavlja na stog dva podatka: "DatotekaA.dat" i "Datoteka". Podatak "Datoteka" označava uspješno obavljenu akciju preslikavanja datoteke, dok podatak "DatotekaA.dat" identificira parametar akcije.

Informacije o pojavi pogreške tijekom postavljanja aplikacijske usluge šalju se *Izvođaču suprotnih operacija*. Uloga *Izvođača suprotnih operacija* je vratiti računalo u stanje u kojem je bilo prije početka postavljanja usluge. *Izvođač suprotnih operacija* uzima podatke sa stoga događaja i izvodi suprotne operacije. Na primjer, ako je na vrhu stoga događaja podatak "VirDir", onda *Izvođač suprotnih operacija* zna da mora izvesti operaciju brisanja prividnog direktorija. Operacija brisanja prividnog direktorija zahtijeva dodatni parametar koji određuje ime prividnog direktorija. Potrebnu vrijednost parametra *Izvođač suprotnih operacija* saznaće oduzimanjem još jednog elementa s vrha stoga događaja. U primjeru sa slike ime prividnog direktorija je "/Usluga1". Postupak poništavanja neuspjelog postupka instalacije aplikacijske usluge zaustavlja se kada *Izvođač suprotnih operacija* dosegne dno stoga događaja.

7.5.3 Postupak uklanjanja aplikacijske usluge

Uklanjanje usluge izvodi se unutar *Upravitelja uklanjanja usluge* koji je sastavni dio *Upravljačkog modula usluge Postavljač*. Slika 7-25 prikazuje arhitekturu *Upravitelja uklanjanja usluge*. Programska arhitektura *Upravitelja uklanjanja usluge* sastoji se od sljedećih dijelova: *Upravitelj dohvaćanja informacija o usluzi*, *Parser instalacijske skripte* i *Interpretator instalacijske skripte*. *Upravitelj dohvaćanja informacija o usluzi* putem *Modula za pristup podacima usluge Lokalni katalog* dohvaća detaljne informacije o postavljenoj usluzi spremljene u spremištu podataka *Lokalnog kataloga*. Iz dohvaćenih podataka *Upravitelj dohvaćanja informacija o usluzi* sazna lokalnu lokaciju usluge i spremljene instalacijske skripte. Upravitelj učitava instalacijsku skriptu te ju proslijeđuje *Parseru instalacijske skripte*. *Parser instalacijske skripte* pretvara tekstualni zapis instalacijske skripte u stablastu strukturu podataka. Izgrađeno stablo instalacijske skripte zajedno s podacima o usluzi proslijeđuje se *Interpretatoru instalacijske skripte* koji provodi postupak uklanjanja postavljene aplikacijske usluge. *Interpretator instalacijske skripte* izvodi suprotne korake onima iz procesa postavljanja aplikacijske usluge.



Slika 7-25: Arhitektura *Upravitelja uklanjanja usluge*

Interpretator instalacijske skripte najprije briše zapis o usluzi iz *Prividne logičke mreže* (1). Brisanjem zapisa o aplikacijskoj usluzi onemogućava se pristup usluzi pomoću komunikacijskih mehanizama *Prividne logičke mreže*. Nakon toga slijedi brisanje zapisa o usluzi iz spremišta podataka usluge *Lokalni katalog* (2). Brisanje podataka o usluzi provodi se putem

Modula za pristup podacima usluge Lokalni katalog koji pristupa spremištu podataka i briše odgovarajući zapis. Unutar ove akcije ujedno se obavještava usluga *Globalni katalog* o uklonjenoj aplikacijskoj usluzi. Ovom akcijom aplikacijska usluga formalno prestaje postojati unutar *VDE* računalne okoline. Nakon ove akcije slijedi fizičko brisanje aplikacijske usluge sa računala. Na osnovi instalacijske skripte utvrđuju se imena dodatnih procesa usluge te se prekida njihovo izvođenje (3). Nadalje, dohvaćanjem XML elemenata s imenom *<VirDir>* određuju se imena prividnih direktorija postavljene usluge. Prividni direktoriji se brišu iz interne strukture podataka IIS poslužitelja (4). Ovom akcijom prekida se veza aplikacijske usluge i IIS poslužitelja. Nakon toga slijedi brisanje datoteka aplikacijske usluge (5). Briše se fizički direktorij aplikacijske usluge sa svim pripadnim datotekama i ugnježdenim poddirektorijima. Ako *Upravitelj uklanjanja usluge* ne uspije iz nekog razloga obrisati pojedine datoteke aplikacijske usluge, onda njihove lokacije putem *Modula za pristup podacima usluge Lokalni katalog* zapisuje u spremište podataka usluge *Lokalni katalog*. *Modul za održavanje* naknadno će pokušati obrisati zaostale datoteke.

7.5.4 Postupak konfiguriranja aplikacijske usluge

Konfiguriranje aplikacijske usluge izvodi *Upravitelj konfiguracijom usluge* koji je sastavni dio *Upravljačkog modula usluge Postavljač*. Za konfiguriranje aplikacijskih usluga nužan je jedinstveni format konfiguracijskih datoteka. Zbog jednostavnosti je iskorišten format konfiguracijskih datoteka definiran unutar .NET radnog okvira. *Sustav za postavljanje i otkrivanje usluga* zahtijeva od aplikacijskih usluga da konfiguracijske datoteke ostvare prema formatu .NET radnog okvira.

```
<configuration>
  ...
  <appSettings>
    ...
      <add key="parametar1" value="vrijednost1"/>
      <add key="parametar2" value="vrijednost2"/>
  </appSettings>
</configuration>
```

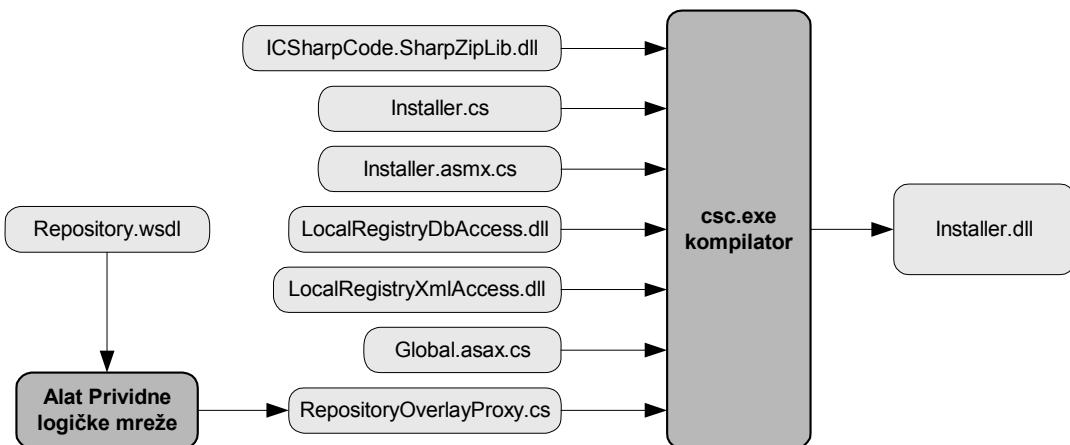
Slika 7-26: Primjer konfiguracijske datoteke aplikacijske usluge

Slika 7-26 prikazuje dio konfiguracijske datoteke aplikacijske usluge. Parametri aplikacijske usluge nalaze se unutar elementa *<appSettings>*. Svaki parametar izražen je elementom *<add>*, gdje atribut *key* sadrži ime parametra, a atribut *value* sadrži vrijednost parametra. Na primjer, prema slici parametar *parametar1* ima vrijednost *vrijednost1*. Za promjenu vrijednosti parametra konfiguracije potrebno je *Upravitelju konfiguracijom usluge* zadati uređeni par (*ime parametra, nova vrijednost*). Na osnovi *imena parametra* upravitelj

pronalazi element `<add>` s podacima o parametru te promijeni njegovu vrijednost na *nova vrijednost*.

7.5.5 Ostvarenje

Usluga *Postavljač* ostvarena je tehnologijama .NET radnog okvira. Izvorni kôd je u potpunosti napisan u C# programskom jeziku. Slika 7-27 prikazuje datoteke izvornog kôda usluge *Postavljač* u postupku prevodenja. Unutar datoteke *Installer.asmx.cs* smješten je izvorni kôd *Web sučelja* usluge *Postavljač*. U datoteci *Installer.cs* smješten je izvorni kôd *Upravljačkog modula*. Datoteka *Global.asax.cs* sadrži izvorni kôd *Modula za održavanje*. Unutar datoteke *Global.asax.cs* zadano je da se *Modul za održavanje* pokrene prilikom inicijalizacije usluge *Postavljač*.



Slika 7-27: Prevodenje izvornog kôda usluge *Postavljač*

Modul za pristup podacima usluge Lokalni katalog sastoji se od programskih knjižnica za pristup podacima usluge *Lokalni katalog*. Programska knjižnica *LocalRegistryDbAccess.dll* ostvaruje pristup bazi podataka usluge *Lokalni katalog*, dok programska knjižnica *LocalRegistryXmlAccess.dll* ostvaruje pristup XML spremištu podataka. Ovisno o konfiguraciji, prilikom izvođenja usluge *Postavljač* koristi se samo jedna od ovih programskih knjižnica. Usluga *Postavljač* komunicira sa ostalim uslugama *Sustava za postavljanje i otkrivanje usluga* putem njihovih zastupnika. Zastupnici se automatski stvaraju alatom *Prividne logičke mreže* na osnovi WSDL opisnika pripadne usluge. Na osnovi WSDL datoteke *Repository.wsdl* stvara se datoteka *RepositoryOverlayProxy.cs* u kojoj se nalazi izvorni kôd *Zastupnika usluge Skladište*. *Zastupnik usluge Globalni katalog* nalazi se unutar programskih knjižnica *LocalRegistryDbAccess.dll* i *LocalRegistryXmlAccess.dll*. Mehanizmi rukovanja ZIP datotekama ostvareni su unutar programske knjižnice *ICSharpCode.SharpZipLib.dll*. Navedena programska

knjižnica preuzeta je s Internet adrese navedene u [114]. Prevođenjem izvornog kôda usluge *Postavljač* nastaje datoteka *Installer.dll* koja se prilikom izvođenja usluge *Postavljač* ugrađuje u IIS poslužitelj.

7.6 Praćenje pogrešaka

Usluge *Sustava za postavljanje i otkrivanje usluga* izvode različite oblike složenih međudjelovanja. Osim međusobno, usluge *Sustava za postavljanje i otkrivanje usluga* ostvaruju suradnju i s ostalim podsustavima *VDE* računalne okoline. U takvim složenim komunikacijskim modelima teško je odrediti izvore pogrešaka. Na primjer, neka korisnik šalje zahtjev usluzi *A* koja tijekom obrade dodatno komunicira s uslugama *B* i *C*. Pretpostavimo da se pogreška dogodila u usluzi *B*. Usluga *A* u tom slučaju vraća korisniku poruku o pogrešci. Ako poruka o pogrešci ne donosi dovoljno informacija, onda korisnik može pogrešno zaključiti da je usluga *A* izvor problema.

U razvoju *Sustava za postavljanje i otkrivanje usluga* posebna pažnja je posvećena potpori otkrivanju i dijagnosticiranju pogrešaka koja se sastoji od dva koraka: *presretanje iznimaka* i *vođenje bilježaka o korištenju usluga*. Za presretanje iznimaka u *Sustavu za postavljanje i otkrivanje usluga* se koristi ugradena potpora .NET radnog okvira. Presretanje iznimaka omogućuje oporavak izvođenja usluge iz neregularnog stanja. Ako uslugu nije moguće vratiti u regularno stanje, onda se izvođenje usluge prekida, a korisniku se šalje poruka s detaljnim opisom pogreške.

```
11/17/2005 11:20:25 AM: Installer : Facility 'ProgramTranslator' has been successfully installed.  
11/17/2005 11:20:26 AM: Installer : Facility 'SSCLTranslator' has been successfully installed.  
11/17/2005 11:20:48 AM: Installer : Facility 'ProgramInterpreter' has been successfully installed.  
11/17/2005 11:20:50 AM: Installer : Facility 'ProgramInstaller' has been successfully installed.  
11/17/2005 11:26:11 AM: Installer : Facility 'SimpleCalculator' has been successfully installed.  
11/29/2005 9:53:23 AM: Installer : Installation process failed.  
System.Web.Services.Protocols.SoapException: There is no repository which contains 'MailBx'.  
    at Microsoft.Web.Services2.Messaging.SoapClient.SendRequestResponse(String methodname,  
SoapEnvelope envelope)  
    at Overlay.OverlayProxyContainer.GlobalRegistryService.GetFacilityStores(SoapEnvelope  
requestMessage){}  
11/29/2005 9:53:35 AM: Installer : Facility 'MailBox' has been successfully installed.  
11/29/2005 9:53:39 AM: Installer : Installation process failed. Facility is allready installed.{}
```

Slika 7-28: Dnevnik korištenja usluge *Postavljač*

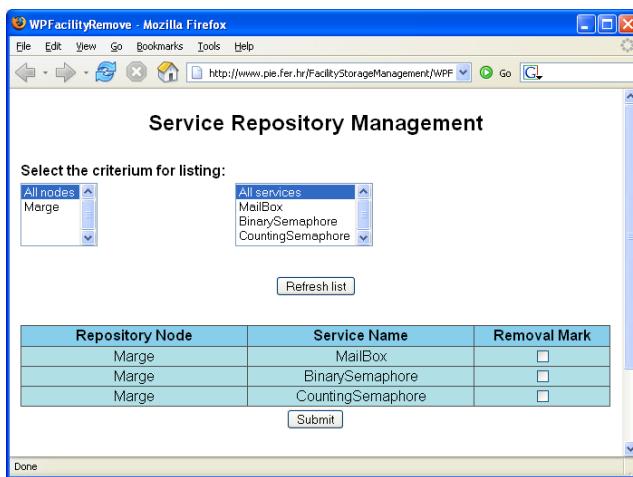
Vođenje bilježaka o korištenju usluge omogućuje pružatelju usluge otkrivanje i dijagnosticiranje pogreške u radu usluge. Informacije o korištenju usluge spremaju se u dnevnik korištenja usluge koji je u slučaju *Sustava za postavljanje i otkrivanje usluga* ostvaren kao datoteka. Svaka usluga sustava upisuje informacije o korištenju u vlastiti dnevnik. Usluge u dnevnik korištenja upisuju informacije o uspješnim i neuspješnim obradama korisničkih zahtjeva. Lokacija dnevnika korištenja navodi se unutar konfiguracijske datoteke usluge. Slika 7-

28 prikazuje primjer dnevnika korištenja usluge *Postavljač*. Svaki zapis u dnevniku korištenja sastoji se od oznake vremena, imena usluge i kratke poruke.

7.7 Web korisničko sučelje

Za uporabu *Sustava za postavljanje i otkrivanje usluga* razvijeno je Web korisničko sučelje. Web korisničko sučelje ostvareno je ASP.NET tehnologijom. Korisnik pristupa Web sučelju putem Web preglednika, kao što je Internet Explorer i Mozilla Firefox. Razvijeno Web sučelje korisniku pruža nadzor nad *Sustavom za postavljanje i otkrivanje usluga*. Korisničko sučelje sastoji se od tri Web stranice putem kojih je moguće vidjeti popis spremljenih usluga u pojedinim uslugama *Skladište*, spremiti instalacijski paket usluge u *Skladište* te postaviti ili ukloniti usluge s čvorova *VDE* računalne okoline.

Slika 7-29 prikazuje primjer stranice s pregledom spremljenih usluga. Na stranici se može filtrirati prikaz po pojedinim uslugama *Skladište*. U svakom retku tablice navodi se čvor na kojem se nalazi usluga *Skladište*, ime spremljene usluge te kontrola za odabir postupka uklanjanja usluge iz pripadnog spremišta. Odabirom gumba *Submit* šalje se zahtjev za uklanjanjem odabranih aplikacijskih usluga iz odgovarajuće usluge *Skladište*.



Slika 7-29: Web korisničko sučelje za pregled spremljenih usluga

Slika 7-30 prikazuje stranicu za postavljanje usluge u *Skladište*. Korisnik na stranici navodi lokacije datoteka instalacijskog paketa koje se nalaze na njegovom lokalnom računalu. Nadalje, korisnik odabire presliku usluge *Skladište* i zadaje ime usluge. Odabirom gumba *Submit* odabrane datoteke se šalju u pripadni *Skladište*.

Slika 7-30: Web korisničko sučelje za postavljanje usluga u *Skladište*

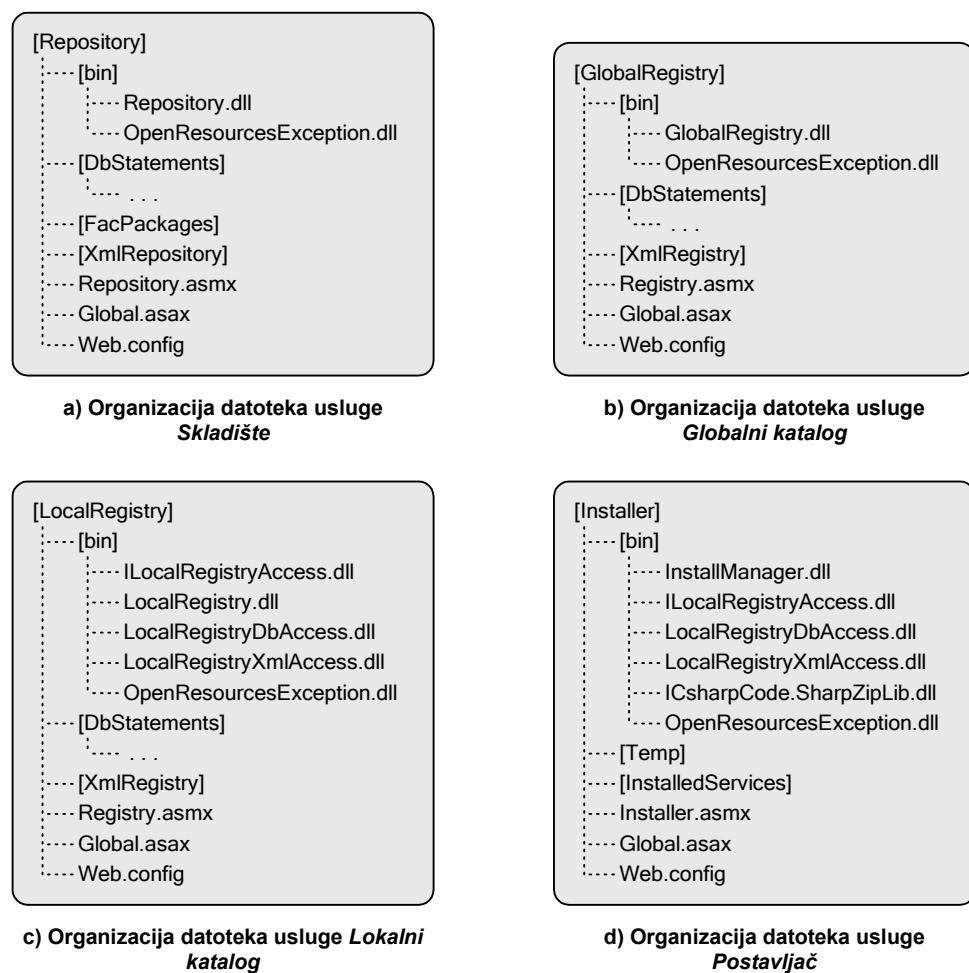
Slika 7-31 prikazuje stranicu za postavljanje usluga na čvorove *VDE* računalne okoline. Stranica prikazuje tablicu s Kartezijevim produkтом svih čvorova *VDE* računalne okoline i svih usluga koje se mogu postavljati. Odabirom filtriranja po čvoru ili usluzi tablica se može smanjiti. Svaki redak tablice sadrži ime čvora, ime usluge i oznaku za postavljanje ili uklanjanje usluge. Odabirom gumba *Submit* odabrane usluge se postavljaju ili uklanjaju s pripadnih čvorova.

Logical Node	Service Name	Installed Status
Merge	MailBox	<input type="checkbox"/>
Merge	BinarySemaphore	<input checked="" type="checkbox"/>
Merge	CountingSemaphore	<input checked="" type="checkbox"/>
Bart	MailBox	<input type="checkbox"/>
Bart	BinarySemaphore	<input type="checkbox"/>
Bart	CountingSemaphore	<input checked="" type="checkbox"/>

Slika 7-31: Web korisničko sučelje za postavljanje usluge na čvorove *VDE* računalne okoline

7.8 Instalacija Sustava za postavljanje i otkrivanje usluga

Instalacija *Sustava za postavljanje i otkrivanje usluga* zahtijeva postojanje .NET radnog okvira (inačica 1.1), Windows operacijskog sustava (Windows XP, Windows 2000, Windows 2003) te IIS poslužitelja (inačica 5.0) na korisničkom računalu. Nakon zadovoljenja navedenih preduvjeta slijedi postupak instalacije koji se sastoji od sljedećih koraka: preslikavanje datoteka sustava na korisničko računalo, definiranje prvidnih direktorija, postavljanje relacijske baze podataka te konfiguriranje usluga.

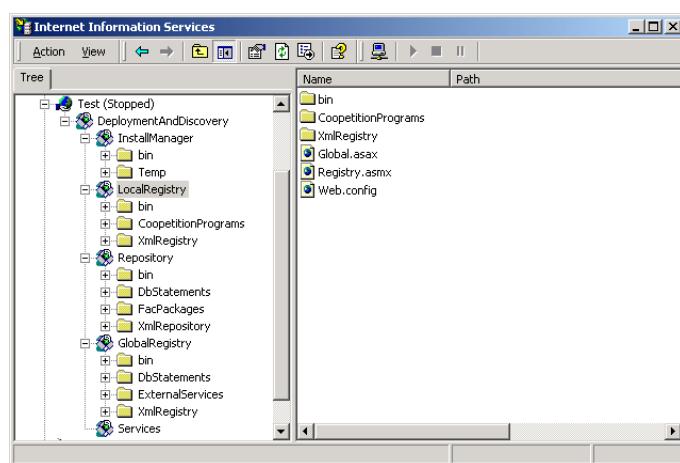


Slika 7-32: Organizacija datoteka usluga

Preslikavanje datoteka na korisničko računalo obuhvaća stvaranje fizičkih direktorija na diskovnom spremniku računala te smještaj datoteka na prikladna mesta. Slika 7-32 prikazuje razmještaj datoteka za usluge sustava. Datoteke izvršnog programskog kôda smještaju se unutar *bin* direktorija. Unutar direktorija *DbStatements* nalaze se definicije SQL upita koji se koriste za upisivanje i dohvatanje podataka iz relacijske baze podataka. Definicije SQL upita prilagođene su specifičnostima *Microsoft SQL Server* sustava relacijskih baza podataka. Direktoriji

XmlRegistry sadrže XML datoteke u koje se spremaju informacije potrebne uslugama *Lokalni* i *Globalni katalog*. Unutar direktorija *XmlRepository* usluge *Skladište* spremaju se XML datoteke koje sadrže informacije o lokacijama spremljenih instalacijskih paketa aplikacijskih usluga. Direktoriji *XmlRegistry* i *XmlRepository* se tijekom instalacije sustava ostavljaju praznima. Usluge same prilikom izvođenja stvaraju potrebne datoteke. Unutar direktorija *FacPackages* usluge *Skladište* spremaju se instalacijski paketi aplikacijskih usluga. Direktorij *Temp* usluge *Postavljač* koristi se za privremeno od pakiravanje paketa datoteka aplikacijske usluge tijekom postupka njenog postavljanja, dok se direktorij *InstalledServices* koristi za spremanje datoteka aplikacijskih usluga. Oba direktorija se tijekom instalacije usluge *Postavljač* ostavljaju praznima. Svaka usluga u svom korijenskom direktoriju sadrži jednu datoteku s nastavkom *.asmx*. Takve datoteke su pristupne točke putem kojih korisnik pristupa funkcionalnostima usluge sustava. Datoteka *Global.asax* definira operacije koje se izvode kao posljedica događaja nastalih tijekom izvođenja usluge. Konfiguracijske postavke usluga navode se u datoteci *Web.config*.

Definiranje prividnih direktorija sljedeći je korak u postupku instalacije sustava. Postupkom definiranja prividnih direktorija povezuju se usluge sustava s IIS poslužiteljom. Za svaku uslugu sustava potrebno je definirati po jedan prividni direktorij tako da pokazuje na korijenski direktorij usluge. Također je potrebno definirati virtualni direktorij u koji će se smještati virtualni direktoriji postavljenih aplikacijskih usluga. Za definiranje prividnih direktorija koristi se alat *Internet Information Services*. Navedeni alat nalazi se unutar upravljačke ploče (engl. Control Panel) Windows operacijskog sustava. Slika 7-33 prikazuje sučelje *Internet Information Services* alata s pravilno definiranim prividnim direktorijima.



Slika 7-33: Korisničko sučelje alata *Internet Information Services*

U slučaju potrebe za spremanjem informacija u relacijske baze podataka potrebno je pripremiti relacijsku bazu podataka. Preduvjet za uporabu relacijskih baza podataka je postojanje *Microsoft SQL Server* sustava na korisničkom računalu. Priprema relacijske baze podataka svodi se na stvaranje pripadnih tablica. Pomoć pri stvaranju tablica relacijske baze podataka pružaju unaprijed pripremljene skripte.

Nakon pripreme relacijske baze podataka potrebno je podesiti konfiguracijske postavke usluga sustava. Konfiguracijske postavke definiraju se unutar *Web.config* datoteka. U dodatku B nalazi se popis s objašnjenjem raspoloživih konfiguracijskih parametara.

8 Zaključak

Računarstvo zasnovano na uslugama olakšava razvoj raspodijeljenih aplikacija. U računarstvu zasnovanom na uslugama raspodijeljene aplikacije se ostvaruju povezivanjem dostupnih usluga uporabom otvorenih komunikacijskih protokola. Uporaba otvorenih protokola omogućuje komunikaciju usluga ostvarenih na raznorodnim računalnim platformama i operacijskim sustavima. Uz osnovne komunikacijske protokole, za razvoj složenih raspodijeljenih aplikacija potrebni su i dodatni mehanizmi, poput mehanizama za otkrivanje i postavljanje usluga. Mehanizmi otkrivanja usluga omogućuju pronalazak informacija o dostupnim uslugama unutar računalne okoline, dok mehanizmi postavljanja usluga omogućuju instalaciju izvršnih oblika usluga na udaljena računala.

U magistarskom radu definirana je arhitektura, te je oblikovan i programski ostvaren *Sustav za postavljanje i otkrivanje usluga*. Razvijeni sustav ostvaruje mehanizme postavljanja i otkrivanja aplikacijskih usluga. Sustav se sastoji od slabo povezanih usluga *Skladište*, *Katalog* i *Postavljač*. *Skladište* je spremište instalacijskih paketa aplikacijskih usluga. Instalacijski paketi sastoje se od instalacijske skripte, paketa datoteka usluge i opisnika usluge. Instalacijska skripta definira postupak postavljanja aplikacijske usluge na čvor računalne okoline. Paket datoteka usluge sadrži sve datoteke potrebne prilikom postavljanja usluge, dok opisnik usluge definira programsko sučelje aplikacijske usluge.

Katalog ostvaruje mehanizme za otkrivanje aplikacijskih usluga. Unutar *Kataloga* spremaju se informacije o spremljениm instalacijskim paketima, informacije o postavljenim uslugama te informacije o aktivnim postavljenim uslugama. *Katalog* je oblikovan u dvorazinsku hijerarhiju. Donja razina hijerarhije sastoji se od niza usluga *Lokalni katalog* koje se postavljaju na svaki čvor računalne okoline. Uloga *Lokalnog kataloga* je održavanje detaljnih informacija o postavljenim uslugama lokalnog računala. Gornju razinu hijerarhije čini usluga *Globalni katalog* koji se postavlja na unaprijed određen čvor računalne okoline. Uloga *Globalnog kataloga* je održavanje osnovnih informacija o uslugama koje su prisutne unutar računalne okoline. Pažljiva raspodjela informacija između *Globalnog* i *Lokalnog kataloga* poboljšava radna svojstva cjelokupnog *Kataloga*. Informacije koje se često mijenjaju i koje zahtijevaju velik memorijski prostor spremaju se isključivo unutar *Lokalnih kataloga*.

Postavljač ostvaruje mehanizme za postavljanje aplikacijskih usluga. Ova usluga se postavlja na svako računalo unutar računalne okoline. Postupak postavljanja aplikacijskih usluga izvodi se na osnovi instalacijske skripte. Postavljanje aplikacijskih usluga ostvareno je kao

transakcija, odnosno u slučaju pogreške tijekom postavljanja usluge cijeli postupak se poništava te se računalo vraća u stanje prije početka postavljanja usluge. Nadalje, tijekom postavljanja aplikacijskih usluga ne prekida se izvođenje postojećih postavljenih usluga. Postupak postavljanja usluga ovisan je o računalnim platformama i operacijskim sustavima za koje su aplikacijske usluge programski ostvarene. Zbog toga trenutno programsko ostvarenje usluge *Postavljač* omogućuje postavljanje aplikacijskih usluga koje su ostvarene tehnologijama .NET radnog okvira, a izvode se na Windows operacijskom sustavu.

Na osnovi korištenja *Sustava za postavljanje i otkrivanje usluga* određene su smjernice budućih istraživanja i proširenja. Sustav trenutno ostvaruje osnovni oblik sigurnosti koji se zasniva na modelu pješčanika. Prema modelu pješčanika aplikacijskoj se usluzi tijekom izvođenja ograničava pristup kritičnim računalnim sredstvima kao što su datoteke operacijskog sustava. Ovakav sigurnosni model ne rješava u potpunosti problem sigurnosti. Potrebno je istražiti metode digitalnog potpisivanja izvršnog kôda aplikacijskih usluga kako bi se onemogućilo širenje zlonamjernih usluga računalnom okolinom. Nadalje, potrebno je istražiti mogućnosti proširenja sustava s ciljem omogućavanja postavljanja aplikacijskih usluga ostvarenih za različite računalne platforme i operacijske sustave.

9 Literatura

- [1] A. S. Tanenbaum, M. van Steen: "Distributed Systems: Principles and Paradigms", Prentice-Hall, U.S.A., 2002.
- [2] M. P. Singh, M. N. Huhns: "Service-Oriented Computing: Semantics, Processes, Agents", John Wiley & Sons, England, 2005.
- [3] M. K. Goff: "Network Distributed Computing: Fitscapes and Fallacies", Prentice Hall, Professional Technical Reference, U.S.A, 2004.
- [4] R. Session: "Fuzzy Boundaries: Objects, Components, and Web Services", ACM Queue, Vol. 2, Issue 9, December/January, 2004-2005, pp. 40-47.
- [5] M. Huhns, M. P. Singh: "Service-Oriented Computing: Key Concepts and Principles", IEEE Internet Computing, Vol. 9, No. 1, January/February 2005, pp. 75-81.
- [6] M. P. Papazoglou: "Extending the Service-Oriented Architecture", Business Integration Journal, February, 2005, pp. 18-21.
- [7] D. Talia: "The Open Grid Services Architecture: Where the Grid Meets the Web", IEEE Internet Computing, Vol. 6, No. 6, November/December 2002, pp. 67-71.
- [8] H. T. Gao, J. H. Hayes, H. Cai: "Integrating Biological Research through Web Services", IEEE Computer, Vol. 38, No. 3, pp. 26-31.
- [9] D. A. Menasce: "MOM vs. RPC: Communication Models for Distributed Applications", IEEE Internet Computing, Vol. 9, No. 2, March/April 2005, pp. 90-93.
- [10] W. Vogels: "Web Services Are Not Distributed Objects", IEEE Internet Computing, Vol. 7, No. 6, November/December 2003, pp. 59-66.
- [11] S. Vinoski: "WS-Nonexistent Standards", IEEE Internet Computing, Vol. 8, No. 6, November/December 2004, pp. 94-96.
- [12] F. Curbera, M. Duftler, R. Khalaf, W. Nagy, N. Mukhi, S. Weerawarana: "Unraveling the Web Services Web: An Introduction to SOAP, WSDL, and UDDI", IEEE Internet Computing, Vol. 6, No. 2, March/April 2002, pp. 86-93.
- [13] L. F. Cabrera, K. Christopher, D. Box: "An Introduction to the Web Services Architecture and Its Specifications, version 2.0", Microsoft, October 2004;
<http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnwebsrv/html/introwsa.asp>

- [14] I. Foster, C. Kesselman, J. M. Nick, S. Tuecke: "The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration", Open Grid Service Infrastructure Working Group, Global Grid Forum, June 22 2002, <http://www.globus.org/alliance/publications/papers/ogsa.pdf>
- [15] K. Czajkowski, D. F. Ferguson, I. Foster, J. Frey, S. Graham, I. Sedukhin, D. Snelling, S. Tuecke, W. Vambenepe: "The WS-Resource Framework", March 2004; <http://www.globus.org/wsrf/specs/ws-wsrf.pdf>
- [16] M. P. Papazoglou, D. Georgakopoulos: "Service-Oriented Computing", Communication of ACM, Vol. 46, No. 10, October 2003, pp. 25-28.
- [17] A. Elfatatty, P. Layzell: "Negotiating in Service-Oriented Environments", Communications of the ACM, Vol. 47, No. 8, pp. 103-108.
- [18] M. P. Papazoglou, J. Dubray: "A Survey of Web Services Technologies", Technical Report, University of Trento, Department of Information and Communication Technology, <http://www.dit.unitn.it>
- [19] C. Peltz: "Web Services Orchestration and Choreography", IEEE Computer, Vol 36, No. 10, October 2003, pp. 46-52.
- [20] A. Carzaniga, A. Fuggetta, R. S. Hall, D. Heimbigner, A. van der Hoek, A. L. Wolf: "A Characterization Framework for Software Deployment Technologies", Technical Report CU-CS-857-98, Department of Computer Science, University of Colorado, Boulder, April 1998.
- [21] D. L. McGuiness, R. Fikes, J. Hendler, L. A. Stein: "DAML+OIL: An Ontology Language for the Semantic Web", IEEE Intelligent Systems, Vol. 17, No. 5, September/October 2002, pp. 72-80.
- [22] J. Hendler, D. McGuinness: "The DARPA Agent Markup Language", IEEE Intelligent Systems, Vol. 15. No. 6, November/December 2000, pp. 72-73.
- [23] F. Manola, E. Miller: "RDF Primer", W3C Recommendation, 10 February 2004, <http://www.w3.org/TR/rdf-primer/>
- [24] D. Brickley, R. V. Guha: "Resource Description Framework (RDF) Schema Specification 1.0", W3C Candidate Recommendation, 27 March 2000, <http://www.w3.org/TR/2000/CR-rdf-schema-20000327/>
- [25] D. L. McGuinness, F. Harmelen: "OWL Web Ontology Language Overview", W3C Recommendation, 10 February 2004, <http://www.w3.org/TR/owl-features/>
- [26] D. Martin et al: "Bringing Semantics to Web Services: The OWL-S Approach", Proceedings of the First International Workshop on Semantic Web Services and Web

- Process Composition (SWSWPC 2004), USA, 2004, <http://www.daml.org/services/owl-s/coalition-pubs.html>
- [27] V. Talwar, D. Milojicic, Q. Wu, C. Pu, W. Yan, G. Jung: "Approaches for Service Deployment", IEEE Internet Computing, Vol. 9, No. 2, March/April 2005, pp. 70-80.
- [28] P. Anderson, P. Goldsack, J. Paterson: "SmartFrog meets LCFG: Autonomous Reconfiguration with Central Policy Control", Proceedings of the 17th Large Installation System Administration Conference (LISA'03), 2003, pp. 213-222.
- [29] P. Goldsack, J. Guijarro, A. Lain, G. Mecheneau, P. Murray, P. Toft: " SmartFrog: Configuration and Automatic Ignition of Distributed Applications", A technical overview paper from the 2003 HP Openview University Association conference, <http://www.hpl.hp.com/research/smartfrog/papers.htm>
- [30] A. Fugetta, G. P. Picco, G. Vigna: "Understanding Code Mobility", IEEE Transactions on Software Engineering, Vol. 24, No. 5, May 1998, pp. 342-361.
- [31] "Systems Management Server 2003 Concepts, Planning, and Deployment Guide", Microsoft, October 2003, <http://www.microsoft.com/downloads/details.aspx?FamilyId=784838B3-34E0-4122-B3E2-17C5B4EEF8F4&displaylang=en>.
- [32] "Systems Management Server 2003 Reviewer's Guide", Microsoft, October 2003, <http://www.microsoft.com/smsserver/evaluation/revguide/default.asp>
- [33] "Radia Essentials Guide: for the Unix and Windows operating systems", Product manual, Hewlett-Packard Development Company, August 2004, http://ovweb.external.hp.com/lpe/doc_serv/
- [34] J. Tilly, E. Burke: "Ant: The Definitive Guide", O'Reilly, May 2002.
- [35] W. Goscinski, D. Abramson: "Distributed Ant: A System to Support Application Deployment in the Grid", Proceedings of the Fifth IEEE/ACM International Workshop on Grid Computing (GRID'04), 2004, pp. 436-443.
- [36] B. Chun, D. Culler, T. Roscoe, A. Bavier, L. Peterson, M. Wawrzoniak, M. Bowman: "PlanetLab: An Overlay Testbed for Broad-Coverage Services", ACM SIGCOMM Computer Communication Review, Vol. 33, Issue 3, July 2003, pp. 3-12.
- [37] "The Nixes Tool Set", Northwestern University, <http://www.aqualab.cs.northwestern.edu/nixes.html>
- [38] "PlanetLab Slice Deploy Toolkit", Intel Oregon, <http://jabber.services.planet-lab.org/php/software/tools.php>

- [39] "PlanetLab Application Manager", UC Berkeley & Intel Research Berkeley, <http://appmanager.berkeley.intel-research.net>
- [40] "The GNU Bourne-Again SHell", <http://cnswww.cns.cwru.edu/~chet/bash/bashtop.html>
- [41] "RPM Package Manager (RPM)", <http://www.rpm.org/>
- [42] F. Xu, M. H. Eres, D. J. Baker, S. J. Cox: "Tools and Support for Deploying Applications on the Grid", Proceedings of the 20004 IEEE International Conference on Services Computing (SCC'04), 2004, pp. 281-287.
- [43] G. Huang, M. Wang, L. Ma, L. Lan, T. Liu, H. Mei: "Towards Architecture Model based Deployment for Dynamic Grid Services", Proceedings of the IEEE International Conference on E-Commerce Technology for Dynamic E-Business (CEC-East'04), 2004, pp. 14-21.
- [44] S. Graupner, V. Kotov, H. Trinks: "Resource-Sharing and Service Deployment in Virtual Data centers", Proceedings of the 22nd International Conference on Distributed Computing Systems Workshops (ICDCSW'02), 2002, pp. 666-671.
- [45] "Configuration Description, Deployment, and Lifecycle Management (CDDLM-WG)", Global Grid Forum Working Group, <https://forge.gridforum.org/projects/cddlm-wg>
- [46] "Installable Unit Deployment Descriptor", W3C draft specification, <http://www.w3.org/Submission/InstallableUnit-DD/>
- [47] N. P. Sudmann, D. Johansen: "Software Deployment using Mobile Agents", Proceedings of the IFIP/ACM Working Conference on Component Deployment, 2002, pp. 97-107.
- [48] D. Johansen, R. van Renesse, F. B. Schneider: "Operating System Support for Mobile Agents", Proceedings of the Fifth Workshop on Hot Topics in Operating Systems (HotOS-V), May 1995, pp. 42-45.
- [49] M. Kusek, G. Jezic, I. Ljubi, K. Mlinaric, I. Lovrek, S. Desic, O. Labor, A. Caric, D. Huljenic: "Mobile Agent Based Software Operation and Maintenance", Proceedings of 7th International Conference on Telecommunications (ConTEL 2003), 2003, pp. 601-608.
- [50] T. Friese, M. Smith, B. Freisleben: "Hot Service Deployment in an Ad Hoc Grid Environment", Proceedings of the 2nd international conference on Service oriented computing, 2004, pp. 75-83.
- [51] F. Emekci, O. D. Sahin, D. Agrawal, A. El Abbadi: "A Peer-to-Peer Framework for Web Service Discovery with Ranking", Proceedings of the IEEE International Conference on Web Services (ICWS'04), 2004, pp. 192-199.

- [52] J. Gray, P. Helland, P. O'Neil, D. Shasha: "The Dangers of Replication and Solution", Proceedings of ACM-SIGMOD 1996 International Conference on Management of Data, Montreal, Quebec, 1996. pp. 173-182.
- [53] D. Xu, K. Nahrstedt, D. Wichadakul: "QoS-Aware Discovery of Wide-Area Distributed Services", Proceedings of the 1st International Symposium on Cluster Computing and the Grid (CCGRID '01), 2001, pp. 92-99.
- [54] C. Sun, Y. Lin, B. Kemme: "Comparison of UDDI Registry Replication Strategies", Proceedings of the IEEE International Conference on Web Services (ICWS'04), 2004, pp. 218-225.
- [55] K. Sivashanmugam, K. Verma, A. Sheth: "Discovery of Web Services in a Federated Registry Environment", Proceedings of the IEEE International Conference on Web Services (ICWS'04), 2004, pp. 270-277.
- [56] S. Vinoski: "Service Discovery 101", IEEE Internet Computing, Vol. 7, No. 1, January/February 2003, pp. 69-71.
- [57] J. H. Hausmann, R. Heckel, M. Lohmann: "Model-based Discovery of Web Services", Proceedings of the IEEE International Conference on Web Services (ICWS'04), 2004, pp. 324-331.
- [58] J. Colgrave, R. Akkiraju, R. Goodwin: "External Matching in UDDI", Proceedings of the IEEE International Conference on Web Services (ICWS'04), 2004, pp. 226-323.
- [59] H. Luo, M. Barbeau: "Performance Evaluation of Service Discovery Strategies in Ad Hoc Networks", Proceedings of the Second Annual Conference on Communication Networks and Service research (CNSR'04), 2004, pp. 61-68.
- [60] Y. Wang, E. Stroulia: "Flexible Interface Matching for Web-Service Discovery", Proceedings of the Fourth International Conference on Web Information Systems Engineering (WISE'03), 2003, pp. 147-156.
- [61] L. Aversano, G. Canforam A. Ciampi: "An Algorithm for Web Service Discovery through Their Composition", Proceedings of the IEEE International Conference on Web Services (ICWS'04), 2004, pp. 332-339.
- [62] A. Soydan Bilgin, M. P. Singh: "A DAML-Based Repository for QoS-Aware Semantic Web Service Selection", Proceedings of the IEEE International Conference on Web Services (ICWS'04), 2004, pp. 368-375.
- [63] S. A. McIlraith, T. C. Son, H. Zeng: "Semantic Web Services", IEEE Intelligent Systems, Vol. 16, No. 2, March/April 2001, pp. 46-53.

- [64] W3C: "QoS for Web Services: Requirements and Possible Approaches", W3C Working Group Note, November, 2003, <http://www.w3c.or.kr/kr-office/TR/2003/ws-qos/>
- [65] S. Kalepu, S. Krishnaswamy, S. W. Loke: "Reputation = f(User Ranking, Compliance, Verity)", Proceedings of the IEEE International Conference on Web Services (ICWS'04), 2004, pp. 200-207.
- [66] E. M. Maximilien, M. P. Singh: "Conceptual Model of Web Service Reputation", ACM SIGMOD Record, Vol. 31, Issue 4, December 2002, pp. 36-41.
- [67] S. A. McIlraith, D. L. Martin: "Bringing Semantics to Web Services", IEEE Intelligent Systems, Vol. 18, No. 1, January/February 2003, pp. 90-93.
- [68] "Mac OS X Tiger Technical Specifications: Bonjour", White Paper, Apple Computer, http://images.apple.com/macosx/pdf/MacOSX_Bonjour_TB.pdf
- [69] "JINI Architecture Overview", Technical White Paper, Sun Microsystems, January 1999, <http://www.sun.com/software/jini/whitepapers/architecture.pdf>
- [70] K. Czajkowski, S. Fitzgerald, I. Foster, C. Kesselman: "Grid Information Services for Distributed Resource Sharing", Proceedings of the 10th IEEE International Symposium on High-Performance Distributed Computing (HPDC-10'01), 2001, pp. 181-194.
- [71] M. Paolucci, K. Sycara, T. Nishimura, N. Srinivasan: "Using DAML-S for P2P Discovery", Proceedings of the First International Conference on Web Services (ICWS'03), June 2003, pp. 203-207.
- [72] Clip2: "Gnutella Protocol Specification v0.4", http://www9.limewire.com/developer/gnutella_protocol_0.4.pdf
- [73] T. Kawamura, J. De Blasio, T. Hasegawa, M. Paolucci, K. Sycara: "Preliminary Report of Public Experiment of Semantic Service Matchmaker with UDDI Business Registry", 1st International Conference on Service Oriented Computing (ICSOC 2003), Trento, Italy, December 2003.
- [74] "Globus Toolkit", <http://www.globus.org/toolkit/>
- [75] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, T. Berners-Lee: "Hypertext Transfer Protocol -- HTTP/1.1", Network Working Group RFC 2616, June 1999, <http://www.w3.org/Protocols/rfc2616/rfc2616.html>
- [76] J. Klensin: "Simple Mail transfer Protocol", Network Working Group RCF 2821, April 2001, <http://www.ietf.org/rfc/rfc2821.txt>
- [77] J. Postel, et al: "Transmission Control Protocol", DARPA Internet Program, RFC 793, September 1981, <http://www.ietf.org/rfc/rfc793.txt>

- [78] D. Raggett, A. Le Hors, I. Jacobs: "HTML 4.01 Specification", W3C Recommendation, 24 December 1999, <http://www.w3.org/TR/REC-html40/>
- [79] M. Gudgin, M. Hadley, N. Mendelsohn, J. Moreau, H. F. Nielsen: "SOAP Version 1.2 Part 1: Messaging Framework", W3C Recommendation, 24 June 2003, <http://www.w3.org/TR/2003/REC-soap12-part1-20030624/>
- [80] E. Christensen, F. Curbera, G. Meredith, S. Weerawarana: "Web Services Description Language (WSDL) 1.1", W3C Note, 15 March 2001, <http://www.w3.org/TR/wsdl>
- [81] T. Bellwood, et al: "UDDI Version 3.0.2", UDDI Spec Technical Committee Draft, 19 October 2004, <http://www.oasis-open.org/committees/uddi-spec/doc/spec/v3/uddi-v3.0.2-20041019.htm>
- [82] A. O. Freier, P. Karlton, P. C. Kocher: "The SSL Protocol Version 3.0", Transport Layer Security Working Group, 18 November, 1996, <http://wp.netscape.com/eng/ssl3/draft302.txt>
- [83] T. Dierks, C. Allen: "The TLS Protocol Version 1.0", Network Working Group RFC 2246, January 1999, <http://www.ietf.org/rfc/rfc2246.txt>
- [84] A. Nadalin, C. Kaler, P. Hallam-Baker, R. Monzillo: "Web Services Security: SOAP Message Security 1.0 (WS-Security 2004)", OASIS Standard 200401, March 2004, <http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf>
- [85] S. Anderson, et al: "Web Services Secure Conversation Language (WS-SecureConversation)", February 2005, <http://msdn.microsoft.com/library/en-us/dnglobspec/html/WS-secureconversation.pdf>
- [86] S. Anderson, et al: "Web Services Trust Language (WS-Trust)", February 2005, <http://msdn.microsoft.com/library/en-us/dnglobspec/html/WS-trust.pdf>
- [87] S. Bajaj, et al: "Web Services Federation Language (WS-Federation)", July 8 2003, <http://msdn.microsoft.com/webservices/webservices/understanding/advancedwebservices/default.aspx?pull=/library/en-us/dnglobspec/html/ws-federation.asp>
- [88] S. Bajaj, et al: "Web Services Policy Framework (WS-Policy)", September 2004, <http://msdn.microsoft.com/webservices/default.aspx?pull=/library/en-us/dnglobspec/html/ws-policy.asp>
- [89] T. Andrews, et al: "Business Process Execution Language for Web Services Version 1.1", February 2005, <ftp://www6.software.ibm.com/software/developer/library/ws-bpel.pdf>
- [90] S. Tuecke, et al: "Open Grid Services Infrastructure (OGSI) Version 1.0", June 27 2003, http://www-unix.globus.org/toolkit/draft-ggf-ogsi-gridservice-33_2003-06-27.pdf

- [91] S. Vinoski: "CORBA: Integrating Diverse Applications Within Distributed Heterogeneous Environments", IEEE Communications Magazine, Vol. 35, No. 2, February 1997, pp. 46-55.
- [92] D. Box: "Essential COM", Addison-Wesley Professional, December 22, 1997.
- [93] D. Škvorc: "Prividna mreža računalnih sustava zasnovanih na uslugama", magistarski rad u izradi, Fakultet elektrotehnike i računarstva, Zagreb, 2005.
- [94] D. Skrobo: "Raspodijeljeno usporedno interpretiranje programa u arhitekturama zasnovanim na uslugama", magistarski rad u izradi, Fakultet elektrotehnike i računarstva, Zagreb, 2005.
- [95] I. Gavran: "Korisnički jezik programskog modela zasnovanog na uslugama", magistarski rad u izradi, Fakultet elektrotehnike i računarstva, Zagreb, 2005.
- [96] M. Popović: "Nadziranje pristupa računalnim sustavima zasnovanim na uslugama", magistarski rad u izradi, Fakultet elektrotehnike i računarstva, Zagreb, 2005.
- [97] A. Milanović: "Programski model zasnovan na uslugama", doktorska disertacija u izradi, Fakultet elektrotehnike i računarstva, Zagreb, 2005.
- [98] M. Popović, I. Benc, S. Srbljić: "Access Control in Hierarchies of Protected Cells", Proceedings of the 9th World Multi-Conference on Systemics, Cybernetics and Informatics (WMSCI'05), Orlando, Florida, U.S.A., 2005, Vol. 3, pp. 356-361.
- [99] D. Skrobo, A. Milanović, S. Srbljić: "Distributed Program Interpretation in Service-Oriented Architectures", Proceedings of the 9th World Multi-Conference on Systemics, Cybernetics and Informatics (WMSCI'05), Orlando, Florida, U.S.A., 2005, Vol. 4, pp. 193-197.
- [100] M. Podravec, I. Skuliber, S. Srbljić: "Service Discovery and Deployment in Service-Oriented Computing Environment", Proceedings of the 9th World Multi-Conference on Systemics, Cybernetics and Informatics (WMSCI'05), Orlando, Florida, U.S.A., 2005, Vol. 3, pp. 5-10.
- [101] A. Milanović, S. Srbljić, D. Skrobo, D. Čapalija, S. Rešković: "Coopetition Mechanisms for Service-Oriented Distributed Systems", Proceedings of the 3th International Conference on Computing, Communications and Control Technologies (CCCT'05), Austin, Texas, U.S.A., 2005, pp. 118-123.
- [102] J. Richter: "Applied Microsoft .NET Framework Programming", Microsoft Press, Redmond, Washington, United States of America, 2002.
- [103] R. V. der Lans: "Introduction to SQL", Addison-Wesley, 1999.

- [104] M. Powell: "Web Services, Opaque Data, and the Attachments Problem", June 2004, <http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnwebsrv/html/opaquedata.asp>
- [105] J. J. Barton, S. Thatte, H. F. Nielsen: " SOAP Messages with Attachments", W3C Note, 11 December 2000, <http://www.w3.org/TR/SOAP-attachments>
- [106] H. F. Nielsen, E. Christensen, J. Farrell: "WS-Attachments", 17 June 2002, <http://msdn.microsoft.com/library/en-us/dnglobspec/html/draft-nielsen-dime-soap-01.txt>
- [107] N. Freed, N. Borenstein: "Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies", Network Working Group RFC 2045, November 1996, <http://www.ietf.org/rfc/rfc2045.txt>
- [108] M. Gudgin, N. Mendelsohn, M. Nottingham, H. Ruellan: " SOAP Message Transmission Optimization Mechanism", W3C Recommendation, 25 January 2005, <http://www.w3.org/TR/soap12-mtom/>
- [109] A. D. Rubin, D. E. Geer: "Mobile Code Security", IEEE Internet Computing, Vol. 2, No. 6, November/December 1998, pp. 30-34.
- [110] R. R. Brooks: "Mobile Code Paradigms and Security Issues", IEEE Internet Computing, Vol. 8, No. 3, May/June 2004, pp. 54-59.
- [111] D. Žubrinić: "Diskretna matematika", Element, Zagreb, 2002.
- [112] J. Clark, S. DeRose: "XML Path Language (XPath) Version 1.0", W3C Recommendation, 16 November 1999, <http://www.w3.org/TR/xpath>
- [113] K. Geiger: "Inside ODBC", Microsoft Press, 1995.
- [114] M. Krueger: "The Zip, GZip, BZip2 and Tar Implementation For .NET", <http://www.icsharpcode.net/OpenSource/SharpZipLib/Default.aspx>
- [115] ".ZIP File Format Specification", Version 6.2.1, PKWARE, April 1, 2005, http://www.pkware.com/business_and_developers/developer/appnote/

10 Životopis

Rođen sam 12. veljače 1980. godine u Koprivnici. Osnovnu školu završio sam u Virju, a gimnaziju u Koprivnici. Nakon što sam 1998. godine maturirao, upisao sam dodiplomski studij na Fakultetu elektrotehnike i računarstva u Zagrebu. Tijekom studija primio sam priznanje "Josip Lončar" za primjeren uspjeh na 4. godini studija. Na završnim godinama studija bio sam demonstrator iz sljedećih predmeta dodiplomske nastave: "Automati, formalni jezici i jezični procesori I" i "Automati, formalni jezici i jezični procesori II". Tijekom ljeta 2001. i 2002. godine sudjelovao sam na ljetnim radionicama u organizaciji tvrtke Ericsson Nikola Tesla d.d. i Fakulteta elektrotehnike i računarstva. Diplomirao sam 2003. godine pod vodstvom prof. dr. sc. Siniše Srbljića diplomskim radom naslova "Mjerenje i nadgledanje svojstava posredničkog sustava".

Nakon završenog studija upisao sam u studenom 2003. godine poslijediplomski studij na Fakultetu elektrotehnike i računarstva, smjer Jezgra računarstva. U siječnju 2004. godine zaposlio sam se kao znanstveni suradnik na Zavodu za elektroniku, mikroelektroniku, računalne i inteligentne sustava Fakulteta elektrotehnike i računarstva. Od rujna 2004. godine radim kao znanstveni novak na projektu 0036051 "Raspodijeljeni ugrađeni računalni sustavi" čiji je voditelj akademik Leo Budin. Pod vodstvom prof. dr. sc. Siniše Srbljića sudjelujem na tehnološkom projektu TP-01/036-29 "Okrilje posrednika javnog informacijskog sustava". U okviru nastavnih aktivnosti, aktivno sudjelujem u provedbi laboratorijskih vježbi iz kolegija "Automati, formalni jezici i jezični procesori I", "Automati, formalni jezici i jezični procesori II", "Inteligentni sustavi", "Operacijski sustavi 2" i "Analiza i projektiranje računalom". U sklopu suradnje Fakulteta elektrotehnike i računarstva i tvrtke Ericsson Nikola Tesla d. d. sudjelovao sam u organizaciji i provedbi ljetnih radionica 2004. i 2005. godine kao voditelj studenata dodiplomske nastave. Područja interesa su mi raspodijeljeno računarstvo, računarstvo zasnovano na uslugama, te tehnologije Web usluga.

Sudjelovao sam na međunarodnim konferencijama s jednim člankom u području računarstva zasnovanog na uslugama.

11 Sažetak

Računarstvo zasnovano na uslugama postaje sve važnija paradigma u oblikovanju, izgradnji i održavanju raspodijeljenih aplikacija. Specijalizirani mehanizmi, poput otkrivanja i postavljanja usluga, olakšavaju razvoj i održavanje raspodijeljenih aplikacija. Magistarski rad opisuje arhitekturu i programsko ostvarenje *Sustava za postavljanje i otkrivanje usluga*. Razvijeni sustav ostvaruje mehanizme za otkrivanje i postavljanje usluga. *Sustav za postavljanje i otkrivanje usluga* ostvaren je kao raspodijeljen sustav zasnovan na slabo povezanim uslugama *Skladište*, *Katalog* i *Postavljač*. *Skladište* je spremište instalacijskih paketa usluga. Instalacijski paket je skup datoteka usluge koji je posebno prilagođen automatiziranom postupku postavljanja usluge. *Katalog* omogućuje spremanje i pretraživanje informacija o dostupnim uslugama. *Postavljač* ostvaruje mehanizme za udaljeno postavljanje aplikacijskih usluga. Programsko ostvarenje sustava prilagođeno je zahtjevima računarstva zasnovanog na uslugama. Usluge sustava pružaju vlastite funkcionalnosti putem otvorenih i standardiziranih sučelja, dok se za komunikaciju koriste otvoreni protokoli. U sklopu rada definiran je jezik za opisivanje instalacijskog postupka koji je posebno prilagođen postavljanju usluga ostvarenih tehnologijama .NET radnog okvira.

12 Summary

"Service discovery and deployment in service-oriented systems"

Service-oriented computing is becoming one of the most important paradigms for design, development and maintenance of distributed applications. Specific mechanisms, such as mechanisms for service discovery and deployment, facilitate development and maintenance of distributed applications. The master thesis describes an architecture and implementation of the *Service discovery and deployment system*. Implemented system provides mechanisms for service discovery and deployment. *Service discovery and deployment system* is realized as a service-oriented distributed system, which consists of loosely coupled services: *Repository*, *Registry* and *Installer*. *Repository* provides storage for service installation packages. Installation package is a set of service files that are needed for automation of service deployment process. *Registry* provides functionalities for storing and searching information about available services. *Installer* provides mechanisms for remote service deployment. System implementation is compliant with Web services standards. Functionalities of system services are exposed through open interfaces and are accessible through open communication protocols. Also, special language for describing service installation process is proposed. Proposed language is suitable for deployment of services that are implemented with *.NET Framework* technologies.

13 Ključne riječi

Otkrivanje usluga, postavljanje usluga, računarstvo zasnovano na uslugama, arhitekture zasnovane na uslugama, raspodijeljeni sustavi

Service discovery, service deployment, service-oriented computing, service-oriented architectures, distributed systems

14 Dodatak A

Tablica 14-1: Metode sučelja usluge *Skladište*

Usluga <i>Skladište</i>
Spremanje aplikacijske usluge
FacilityUpload Sprema aplikacijsku uslugu. Paket treba biti binarna datoteka u ZIP formatu. Ulazni parametri <ul style="list-style-type: none">• <i>string facilityID</i> – ime usluge• <i>string script</i> – instalacijska skripta• <i>byte[] package</i> – paket datoteka• <i>string wsdl</i> – opisnik usluge
Dohvaćanje instalacijskog paketa
GetAvailableFacilities Vraća polje imena spremljenih usluga unutar usluge <i>Skladište</i> . Izlazna vrijednost <ul style="list-style-type: none">• <i>string []</i> – polje imena spremljenih usluga
GetFacilityInstallScript Vraća instalacijsku skripte spremljene usluge. Ulazni parametri <ul style="list-style-type: none">• <i>string facilityID</i> – ime usluge Izlazna vrijednost <ul style="list-style-type: none">• <i>string</i> – sadržaj instalacijske skripte
GetFacilityMethodWsdl Vraća WSDL opisnik metode spremljene usluge. Ulazni parametri <ul style="list-style-type: none">• <i>string facilityID</i> – ime usluge• <i>string methodName</i> – ime metode Izlazna vrijednost <ul style="list-style-type: none">• <i>string</i> – sadržaj opisnika usluge
GetFacilityPackage Vraća paket datoteka spremljene usluge. Ulazni parametri <ul style="list-style-type: none">• <i>string facilityID</i> – ime usluge Izlazna vrijednost <ul style="list-style-type: none">• <i>byte[]</i> – polje okteta sa sadržajem paketa datoteka

GetFacilityPackagePart

Vraća dio paketa datoteke spremljene usluge određen rednim brojem bloka. Veličina bloka specificira se unutar konfiguracijske datoteke *Skladišta*. Parametar *nPart* je ulazno/izlazni parametar metode. Parametar *nPart* u izlaznoj poruci sadrži ulaznu vrijednost parametra povećanu za jedan. Time se korisniku označava postojanje sljedećeg bloka paketa datoteke. Ako sljedeći blok paketa datoteke ne postoji, onda *nPart* u izlaznoj poruci ima vrijednost –1.

Ulagani parametri

- *string facilityID* – ime usluge
- *ref int nPart* – redni broj bloka

Izlazna vrijednost

- *byte[]* – polje okteta sa sadržajem traženog bloka paketa datoteke

GetFacilityWsdl

Vraća WSDL opisnik spremljene usluge.

Ulagani parametri

- *string facilityID* – ime usluge

Izlazna vrijednost

- *string* – sadržaj opisnika usluge

Uklanjanje instalacijskog paketa**RemoveFacility**

Uklanja spremljenu uslugu iz skladišta.

Ulagani parametri

- *string facilityID* – ime usluge

Tablica 14-2: Metode sučelja usluge *Globalni katalog*

Usluga <i>Globalni katalog</i>
Upis informacija
AddCompiler
Registracija informacija o <i>SSCL</i> kompilatoru.
Ulagani parametri
<ul style="list-style-type: none"> • <i>string compilerNodeName</i> – ime logičkog čvora na kojem je postavljen <i>SSCL kompilator</i> • <i>string jobQueueNodeName</i> – ime logičkog čvora s redom zadataka • <i>string jobQueueInstanceName</i> – ime instance reda zadataka • <i>string programQueueNodeName</i> – ime logičkog čvora s redom programa • <i>string programQueueInstanceName</i> – ime instance reda programa
AddEnvironmentNode
Registracija logičkog čvora.
Ulagani parametri
<ul style="list-style-type: none"> • <i>string nodeName</i> – ime logičkog čvora • <i>string nodePhysicalAddress</i> – fizička adresa računala na kojem se nalazi logički čvor

AddExternalService

Registracija vanjske usluge. Vanjske usluge nalaze se izvan granica *VDE* računalne okoline pa im se ne može pristupiti putem prividne logičke mreže.

Ulazni parametri

- *string serviceURL* – fizička adresa usluge
- *string serviceDescription* – kratki opis usluge
- *string wsdl* – WSDL opisnik usluge

AddFacility

Registracija aplikacijske usluge. Upotrebom ovog poziva samo je najavljena uporaba navedene aplikacijske usluge. Za registraciju spremljenih ili postavljenih oblika aplikacijske potrebno je pozvati pripadne metode usluge *Globalni katalog*.

Ulazni parametri

- *string facilityID* – ime usluge
- *string facDescription* – kratki opis usluge
- *string facType* – vrsta usluge. Dopusene vrijednosti su: *SystemFacility*, *BasicFacility* i *CoopetitionProgram*. Vrijednost *SystemFacility* upotrebljava se prilikom registracije specifičnih usluga računalne okoline. Vrijednost *CoopetitionProgram* upotrebljava se prilikom registracije raspodijeljenih programa. Vrijednost *BasicFacility* upotrebljava se u preostalim slučajevima.

AddInstalledFacility

Registracija postavljene usluge.

Ulazni parametri

- *string nodeName* – ime logičkog čvora na kojem je usluga postavljena
- *string facilityID* – ime usluge

AddInterpreter

Registracija informacija o *CL* interpretatoru.

Ulazni parametri

- *string interpreterNodeName* – ime logičkog čvora na kojem je postavljen *CL interpretator*
- *string jobQueueNodeName* – ime logičkog čvora s redom zadataka
- *string jobQueueInstanceName* – ime instance reda zadataka

AddRepository

Registracija usluge *Skladište*.

Ulazni parametri

- *string repAddress* – ime čvora na kojem se nalazi usluga *Skladište*
- *string repDescription* – kratki opis usluge *Skladište*

AddStoredFacility

Registracija spremljene aplikacijske usluge.

Ulazni parametri

- *string facilityID* – ime usluge
- *string repAddress* – ime logičkog čvora u koji je spremljena usluga

Dohvaćanje informacija

GetCompilers

Vraća informacije o aktivnim *SSCL kompilatorima* unutar *VDE* računalne okoline. Svaki *SSCL kompilator* određen je sljedećim informacijama: ime logičkog čvora na kojem je postavljen *SSCL kompilator*, ime logičkog čvora na kojem se nalazi red programa, ime instance reda programa, ime logičkog čvora na kojem se nalazi red zadatka i ime instance reda zadatka.

Izlazna vrijednost

- *CompilerInfo[]* – polje informacija o *SSCL kompilatorima*

GetEnvironmentNodes

Vraća informacije o logičkim čvorovima *VDE* računalne okoline. Svaki logički čvor određen je sljedećim informacijama: ime logičkog čvora i fizičkom adresom računala.

Izlazna vrijednost

- *NodeInfo[]* – polje informacija o logičkim čvorovima

GetExternalServices

Vraća osnovne informacije o registriranim vanjskim uslugama. Svaka vanjska usluga određena je fizičkom adresom i kratkim opisom.

Izlazna vrijednost

- *ExternalServiceInfo[]* – polje informacija o vanjskim uslugama

GetExternalServiceWsdl

Vraća WSDL opisnik vanjske usluge.

Ulagani parametri

- *string serviceURL* – fizička adresa usluge

Izlazna vrijednost

- *string* – WSDL opisnik usluge

GetFacilityInfo

Vraća osnovne informacije o registriranoj usluzi. Informacije o usluzi obuhvaćaju: ime usluge, vrstu usluge i kratak opis usluge.

Ulagani parametri

- *string facilityID* – ime usluge

Izlazna vrijednost

- *FacilityInfo* – informacije o usluzi

GetInstalledFacilities

Vraća informacije o postavljenim uslugama. Svaka postavljena usluga određena je sljedećim informacijama: ime usluga i ime logičkog čvora na kojem je usluga postavljena.

Izlazna vrijednost

- *InstalledFacilityInfo[]* – polje informacija o postavljenim uslugama

GetInstalledFacilityNodes

Vraća popis imena logičkih čvorova na kojima je postavljena zadana usluga.

Ulagani parametri

- *string facilityID* – ime usluge

Izlazna vrijednost

- *string[]* – polje imena logičkih čvorova na kojima je usluga postavljena

GetInterpreters

Vraća informacije o aktivnim *CL interpreterima*. Svaki *CL interpreter* određen je sljedećim informacijama: ime logičkog čvora na kojem je postavljen *CL interpreter*, ime logičkog čvora na kojem se nalazi red zadataka i ime instance reda zadataka.

Izlazna vrijednost

- *InterpreterInfo[]* – polje informacija o *CL interpreterima*

GetRepositories

Vraća polje imena logičkih čvorova na kojima je postavljena usluga *Skladište*.

Izlazna vrijednost

- *string[]* – polje imena logičkih čvorova

GetResources

Vraća polje imena instanci usluge koja je postavljena na zadatom logičkom čvoru.

Ulagni parametri

- *string nodeName* – ime logičkog čvora
- *string facilityID* – ime usluge

Izlazna vrijednost

- *string[]* – polje imena instanci usluge

GetStoredFacilities

Vraća informacije o spremlijenim uslugama. Svaka spremljena usluga određena je sljedećim informacijama: ime usluga i ime logičkog čvora na kojem se nalazi usluga *Skladište*.

Izlazna vrijednost

- *StoredFacilityInfo[]* – polje informacija o spremlijenim uslugama

Provjeravanje informacija**IsFacilityInstalled**

Vraća logičku vrijednost kojom se potvrđuje ili opovrgava postojanje usluge na zadatom logičkom čvoru.

Ulagni parametri

- *string nodeName* – ime logičkog čvora na kojem je postavljena usluga
- *string facilityID* – ime usluge

Izlazna vrijednost

- *bool* – *True* ako je usluga postavljena na zadani logički čvor, inače *False*

IsFacilityRegistered

Vraća logičku vrijednost kojom se potvrđuje ili opovrgava postojanje zapisa o usluzi unutar *Globalnog kataloga*.

Ulagni parametri

- *string facilityID* – ime usluge

Izlazna vrijednost

- *bool* – *True* ako postoji zapis o usluzi, inače *False*

IsFacilityStored

Vraća logičku vrijednost kojom se potvrđuje ili opovrgava postojanje zapisa o spremljenoj usluzi unutar *Globalnog kataloga*.

Ulazni parametri

- *string facilityID* – ime usluge

Izlazna vrijednost

- *bool* – *True* ako postoji zapis o spremljenoj usluzi, inače *False*

IsResourceActive

Vraća logičku vrijednost kojom se potvrđuje ili opovrgava postojanje zapisa o instanci usluge na zadanom logičkom čvoru.

Ulazni parametri

- *string nodeName* – ime logičkog čvora
- *string facilityID* – ime usluge
- *string resourceId* – ime instance usluge

Izlazna vrijednost

- *bool* – *True* ako postoji zapis o instanci usluge na zadanom logičkom čvoru, inače *False*

Uklanjanje informacija**RemoveCompiler**

Uklanjanje informacija o *SSCL kompilatoru*.

Ulazni parametri

- *string compilerNodeName* – ime logičkog čvora na kojem se nalazi *SSCL kompilator*

RemoveEnvironmentNode

Uklanjanje informacija o logičkom čvoru.

Ulazni parametri

- *string nodeName* – ime logičkog čvora

RemoveExternalService

Uklanjanje informacija o vanjskoj usluzi.

Ulazni parametri

- *string serviceURL* – fizička adresa usluge

RemoveInstalledFacility

Uklanjanje informacija o postavljenoj usluzi.

Ulazni parametri

- *string nodeName* – ime logičkog čvora na kojem je postavljena usluga
- *string facilityID* – ime usluge

RemoveInterpreter

Uklanjanje informacija o *CL interpretatoru*.

Ulazni parametri

- *string interpreterNodeName* – ime logičkog čvora na kojem se nalazi *CL interpretator*

RemoveRepository

Uklanjanje informacija o usluzi *Skladište*.

Ulazni parametri

- *string repAddress* – ime logičkog čvora na kojem se nalazi usluga *Skladište*

RemoveStoredFacility

Uklanjanje informacija o spremljenoj usluzi.

Ulazni parametri

- *string facilityID* – ime usluge
- *string repAddress* – ime logičkog čvora na kojem je usluga spremljena

Tablica 14-3: Metode sučelja usluge *Lokalni katalog*

Usluga <i>Lokalni katalog</i>
Upisivanje informacija
AddCoopetitionProgram Registracija raspodijeljenog programa. Ulazni parametri <ul style="list-style-type: none"> • <i>string facilityID</i> – ime raspodijeljenog programa • <i>string sDesc</i> – WSDL opisnik raspodijeljenog programa
AddResource Registracija instance usluge. Ulazni parametri <ul style="list-style-type: none"> • <i>string facilityID</i> – ime usluge • <i>string resourceId</i> – ime instance usluge
Dohvaćanje informacija
GetFacilities Vraća informacije o postavljenim uslugama. Svaka postavljena usluge određena je sljedećim informacijama: ime usluge, vrsta usluge i kratki opis usluge. Izlazna vrijednost <ul style="list-style-type: none"> • <i>FacilityInfoLight[]</i> – polje informacija o postavljenim uslugama
GetFacilityMethodWsdl Vraća WSDL opisnik metode spremljene usluge. Ulazni parametri <ul style="list-style-type: none"> • <i>string facilityID</i> – ime usluge • <i>string methodName</i> – ime metode Izlazna vrijednost <ul style="list-style-type: none"> • <i>string</i> – sadržaj opisnika usluge
GetFacilityWsdl Vraća WSDL opisnik spremljene usluge. Ulazni parametri <ul style="list-style-type: none"> • <i>string facilityID</i> – ime usluge Izlazna vrijednost <ul style="list-style-type: none"> • <i>string</i> – sadržaj opisnika usluge

GetResources

Vraća informacije oinstancama usluge. Svaka instanca usluge određena je sljedećim informacijama: ime usluge i ime instance.

Ulazni parametri

- *string facilityID* – ime usluge

Izlazna vrijednost

- *ResourceData[]* – polje informacija oinstancama usluge

Provjeravanje informacija**IsFacilityInstalled**

Vraća logičku vrijednost kojom se potvrđuje ili opovrgava postojanje zapisa o postavljenoj usluzi unutar *Lokalnog kataloga*.

Ulazni parametri

- *string facilityID* – ime usluge

Izlazna vrijednost

- *bool* – *True* ako postoji zapis o postavljenoj usluzi, inače *False*

IsResourceActive

Vraća logičku vrijednost kojom se potvrđuje ili opovrgava postojanje zapisa oinstanci usluge.

Ulazni parametri

- *string facilityID* – ime usluge
- *string resourceId* – ime instance usluge

Izlazna vrijednost

- *bool* – *True* ako postoji zapis oinstanci usluge, inače *False*

RemoveCooperationProgram

Uklanjanje informacija o rasподijeljenom programu.

Ulazni parametri

- *string facilityID* – ime raspodijeljenog programa

RemoveResource

Uklanjanje informacija oinstanci usluge.

Ulazni parametri

- *string facilityID* – ime usluge
- *string resourceId* – ime instance usluge

Tablica 14-4: Metode sučelja usluge *Postavljač*

Usluga <i>Postavljač</i>
Konfiguriranje usluge <i>Postavljač</i>
<p>GetConfigSettings</p> <p>Vraća trenutne vrijednosti konfiguracijskih parametara postavljene usluge. Za svaki konfiguracijski parametar vraća se ime ključa i pripadna vrijednost.</p> <p>Ulazni parametri</p> <ul style="list-style-type: none"> • <i>string facilityID</i> – ime usluge <p>Izlazna vrijednost</p> <ul style="list-style-type: none"> • <i>ConfigSetting[]</i> – vrijednosti konfiguracijskih parametara usluge
Postavljanje aplikacijskih usluga
<p>Install</p> <p>Postavlja zadanu uslugu na lokalno računalo.</p> <p>Ulazni parametri</p> <ul style="list-style-type: none"> • <i>string facilityID</i> – ime usluge
<p>Uninstall</p> <p>Uklanja zadanu uslugu s lokalnog računala. Ako je ulazni parametar <i>forced</i> postavljen u <i>True</i>, onda se nasilno predika izvođenje aktivnih instanci usluge. Ako je ulazni parametar <i>forced</i> postavljen u <i>False</i>, onda se uklanjanje ne izvodi ako usluga posjeduje aktivne instance.</p> <p>Ulazni parametri</p> <ul style="list-style-type: none"> • <i>string facilityID</i> – ime usluge • <i>bool forced</i> – određuje nasilno brisanje aktivnih instanci usluge
Konfiguriranje aplikacijskih usluga
<p>SetConfigSetting</p> <p>Postavlja novu vrijednost konfiguracijskog parametra postavljene usluge.</p> <p>Ulazni parametri</p> <ul style="list-style-type: none"> • <i>string facilityID</i> – ime usluge • <i>string key</i> – ključ konfiguracijskog parametra • <i>string value</i> – nova vrijednost konfiguracijskog parametra
<p>SetDispatcherEnvironmentParameter</p> <p>Sučelje za podešavanje konfiguracijskih parametara usluge <i>Postavljač</i>. Parametar <i>key</i> može poprimiti vrijednosti "<i>LocalNodeName</i>" ili "<i>GlobalRegistryAddress</i>". Ako se ulazni parametar <i>key</i> postavi u "<i>LocalNodeName</i>", onda se definira ime logičkog čvora na čkojem se nalazi usluga <i>Postavljač</i>. Ako se ulazni parametar <i>key</i> postavi u "<i>GlobalRegistryAddress</i>", onda se usluži <i>Postavljač</i> definira ime logičkog čvora na kojem se nalazi usluga <i>Globalni katalog</i>.</p> <p>Ulazni parametri</p> <ul style="list-style-type: none"> • <i>string key</i> – ključ konfiguracijskog parametra • <i>string value</i> – nova vrijednost konfiguracijskog parametra

15 Dodatak B

Tablica 15-1: Konfiguracijski parametri usluge *Skladište*

Usluga <i>Skladište</i>		
Ključ	Podrazumijevana vrijednost	Opis
DataMode	'Xml'	Oznaka načina spremanja informacija. Moguće vrijednosti su: <i>Xml</i> i <i>Database</i> . Vrijednost <i>XML</i> označava spremanje informacija u XML datoteke, a vrijednost <i>Database</i> označava spremanje informacija u relacijsku bazu podataka.
Provider	'SqlClient'	Oznaka vrste pristupa relacijskoj bazi podataka. Moguće vrijednosti su: <i>SqlClient</i> , <i>OleDb</i> i <i>ODBC</i> . Vrijednost <i>SqlClient</i> označava pristup SQL Server sustavu. Vrijednost <i>OleDb</i> označava OLE pristup relacijskom sustavu. Vrijednost <i>ODBC</i> označava pristup bazi podataka putem ODBC sučelja.
ConnectionString	-	Niz znakova za otvaranje veze s relacijskom bazom podataka. Ovaj niz ovisi o specifičnostima RDBMS sustava te je obavezan ako je parametar DataMode postavljen u <i>Database</i> .
StatementsPath	'.\DbStatements'	Relativna staza direktorija s datotekama SQL upita. Početna točka za određivanje apsolutne staze je lokacija <i>bin</i> direktorija usluge.
PackagesPath	'.\FacPackages'	Relativna staza direktorija u koji se spremaju instalacijski paketi usluge. Početna točka za određivanje apsolutne staze je lokacija <i>bin</i> direktorija usluge.
TraceLogFile	'.\Logs\FacilityRepository.log'	Relativna staza datoteke dnevnika korištenja usluge. Početna točka za određivanje apsolutne staze je lokacija <i>bin</i> direktorija usluge.
LocalNodeName	-	Ime lokalnog logičkog čvora.
GlobalRegistryAddress	-	Ime logičkog čvora na kojem se nalazi usluga <i>Globalni katalog</i> .
UseOverlay	'True'	Naznačuje komunikaciju putem prividne logičke mreže. Moguće vrijednosti su <i>True</i> i <i>False</i> .
localRTUrl	-	Fizička adresa lokalnog usmjeravajućeg modula logičke mreže.
BlockSize	'1048576'	Veličina blokova kojima se prenose dijelovi paketa datoteka.

Tablica 15-2: Konfiguracijski parametri usluge *Globalni katalog*

Usluga <i>Globalni katalog</i>		
Ključ	Podrazumijevana vrijednost	Opis
DataMode	'Xml'	Oznaka načina spremanja informacija. Moguće vrijednosti su: <i>Xml</i> i <i>Database</i> . Vrijednost <i>XML</i> označava spremanje informacija u XML datoteke, a vrijednost <i>Database</i> označava spremanje informacija u relacijsku bazu podataka.
Provider	'SqlClient'	Oznaka vrste pristupa relacijskoj bazi podataka. Moguće vrijednosti su: <i>SqlClient</i> , <i>OleDb</i> i <i>ODBC</i> . Vrijednost <i>SqlClient</i> označava pristup SQL Server sustavu. Vrijednost <i>OleDb</i> označava OLE pristup relacijskom sustavu. Vrijednost <i>ODBC</i> označava pristup bazi podataka putem ODBC sučelja.
ConnectionString	-	Niz znakova za otvaranje veze s relacijskom bazom podataka. Ovaj niz ovisi o specifičnostima RDBMS sustava te je obavezan ako je parametar DataMode postavljen u <i>Database</i> .
StatementsPath	'..\DbStatements\SqlServer'	Relativna staza direktorija s datotekama SQL upita. Početna točka za određivanje apsolutne staze je lokacija <i>bin</i> direktorija usluge.
XmlRegistryPath	'..\XmlRegistry'	Relativna staza direktorija s XML datotekama u kojima se spremaju informacije <i>Globalnog kataloga</i> . Početna točka za određivanje apsolutne staze je lokacija <i>bin</i> direktorija usluge.
TraceLogFile	'..\Logs\GlobalRegistry.log'	Relativna staza datoteke dnevnika korištenja usluge. Početna točka za određivanje apsolutne staze je lokacija <i>bin</i> direktorija usluge.
UseOverlay	'True'	Naznačuje komunikaciju putem prividne logičke mreže. Moguće vrijednosti su <i>True</i> i <i>False</i> .

Tablica 15-3: Konfiguracijski parametri usluge *Lokalni katalog*

Usluga <i>Lokalni katalog</i>		
Ključ	Podrazumijevana vrijednost	Opis
DataMode	'Xml'	Oznaka načina spremanja informacija. Moguće vrijednosti su: <i>Xml</i> i <i>Database</i> . Vrijednost <i>XML</i> označava spremanje informacija u XML datoteke, a vrijednost <i>Database</i> označava spremanje informacija u relacijsku bazu podataka.
Provider	'SqlClient'	Oznaka vrste pristupa relacijskoj bazi podataka. Moguće vrijednosti su: <i>SqlClient</i> , <i>OleDb</i> i <i>ODBC</i> . Vrijednost <i>SqlClient</i> označava pristup SQL Server sustavu. Vrijednost <i>OleDb</i> označava OLE pristup relacijskom sustavu. Vrijednost <i>ODBC</i> označava pristup bazi podataka putem ODBC sučelja.

ConnectionString	-	Niz znakova za otvaranje veze s relacijskom bazom podataka. Ovaj niz ovisi o specifičnostima RDBMS sustava te je obavezan ako je parametar DataMode postavljen u <i>Database</i> .
StatementsPath	'..\DbStatements\SqlServer'	Relativna staza direktorija s datotekama SQL upita. Početna točka za određivanje apsolutne staze je lokacija <i>bin</i> direktorija usluge.
XmlRegistryPath	'..\XmlRegistry'	Relativna staza direktorija s XML datotekama u kojima se spremaju informacije Lokalnog kataloga. Početna točka za određivanje apsolutne staze je lokacija <i>bin</i> direktorija usluge.
GlobalRegistryAddress	-	Ime logičkog čvora na kojem se nalazi usluga Globalni katalog.
LocalNodeName	-	Ime lokalnog logičkog čvora.
CoopetitionProgramsPath	'..\CoopetitionPrograms'	Relativna staza direktorija u koji se spremaju informacije o registriranim raspodijeljenim programima. Početna točka za određivanje apsolutne staze je lokacija <i>bin</i> direktorija usluge.
TraceLogFile	'..\Logs\LocalRegistry.log'	Relativna staza datoteke dnevnika korištenja usluge. Početna točka za određivanje apsolutne staze je lokacija <i>bin</i> direktorija usluge.
UseOverlay	'True'	Naznačuje komunikaciju putem prividne logičke mreže. Moguće vrijednosti su <i>True</i> i <i>False</i> .
LocalRTUrl	-	Fizička adresa lokalnog usmjeravajućeg modula logičke mreže.

Tablica 15-4: Konfiguracijski parametri usluge *Postavljac*

Usluga <i>Postavljac</i>		
Ključ	Podrazumijevana vrijednost	Opis
DataMode	'Xml'	Oznaka načina spremanja informacija. Moguće vrijednosti su: <i>Xml</i> i <i>Database</i> . Vrijednost <i>XML</i> označava spremanje informacija u XML datoteke, a vrijednost <i>Database</i> označava spremanje informacija u relacijsku bazu podataka.
Provider	'SqlClient'	Oznaka vrste pristupa relacijskoj bazi podataka. Moguće vrijednosti su: <i>SqlClient</i> , <i>OleDb</i> i <i>ODBC</i> . Vrijednost <i>SqlClient</i> označava pristup SQL Server sustavu. Vrijednost <i>OleDb</i> označava OLE pristup relacijskom sustavu. Vrijednost <i>ODBC</i> označava pristup bazi podataka putem ODBC sučelja.

ConnectionString	-	Niz znakova za otvaranje veze s relacijskom bazom podataka. Ovaj niz ovisi o specifičnostima RDBMS sustava te je obavezan ako je parametar DataMode postavljen u <i>Database</i> .
StatementsPath	'..\..\LocalRegistry\DbStatements'	Relativna staza direktorija s datotekama SQL upita. Početna točka za određivanje apsolutne staze je lokacija <i>bin</i> direktorija usluge.
XmlRegistryPath	'..\..\LocalRegistry\XmlRegistry'	Relativna staza direktorija s XML datotekama u kojima se spremaju informacije <i>Lokalnog kataloga</i> . Početna točka za određivanje apsolutne staze je lokacija <i>bin</i> direktorija usluge.
FacilitiesFilesPath	'..\InstalledFacilities'	Relativna staza direktorija u koji se postavljaju aplikacijske usluge. Početna točka za određivanje apsolutne staze je lokacija <i>bin</i> direktorija usluge.
TemporarayFilePath	'..\Temp'	Relativna staza direktorija u koji se privremeno od pakirava paket datoteka aplikacijske usluge. Početna točka za određivanje apsolutne staze je lokacija <i>bin</i> direktorija usluge.
TraceLogFile	'..\Logs\Dispatcher.log'	Relativna staza datoteke dnevnika korištenja usluge. Početna točka za određivanje apsolutne staze je lokacija <i>bin</i> direktorija usluge.
FacilitiesRootVirDir	'/Services'	Apsolutna staza prividnog direktorija u koji se ugnježđuju prividni direktoriji postavljenih aplikacijskih usluga.
IisPort	'80'	Vrijednost pristupa (engl. port) IIS poslužitelja.
GlobalRegistryAddress	-	Ime logičkog čvora na kojem se nalazi usluga <i>Globalni katalog</i> .
UseOverlay	'True'	Naznačuje komunikaciju putem prividne logičke mreže. Moguće vrijednosti su <i>True</i> i <i>False</i> .
LocalNodeName	-	Ime lokalnog logičkog čvora.
localRTUrl	-	Fizička adresa lokalnog usmjeravajućeg modula logičke mreže.
OverlayInterface	-	Fizička adresa lokalnog sučelja logičke mreže.
AdminUserName	-	Vrijednost imena administratorskog korisničkog računa.
AdminPassword	-	Vrijednost lozinke administratorskog korisničkog računa.