UNIVERZA V LJUBLJANI

FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Dejan Strmljan

Preusmeritev klicev v IP-telefoniji s pomočjo SOAP HTTP tehnologije

DIPLOMSKO DELO NA UNIVERZITETNEM ŠTUDIJU

Mentor: doc. dr. Mojca Ciglarič
Ljubljana, 2009

Univerza
v Ljubljani

Fakulteta za računalništvo
in informatiko

Tržaška 25
1000 Ljubljana, Slovenija
telefon: 01 476 84 11
faks: 01 426 46 47
www.fri.uni-lj.si
e-mail: dekanat@fri.uni-lj.si

Št. naloge: 01546/2009
Datum: 15.02.2009

Univerza v Ljubljani, Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Kandidat: **DEJAN STRMLJAN**

Naslov: **PREUSMERITEV KLICEV V IP-TELEFONIJI S POMOČJO SOAP HTTP TEHNOLOGIJE**

CALL REDIRECTION IN IP TELEPHONY USING SOAP HTTP TECHNOLOGY

Vrsta naloge: Diplomsko delo univerzitetnega študija

Tematika naloge:

Preučite tehnologije XML in protokol SOAP ter preučite njune možnosti za uporabo v IP telefoniji. Predvsem se osredotočite na možnosti oddaljene administracije v povezavi z aktivnim imenikom, ki ga danes srečamo že v skoraj vsakem podjetju. Strežnik za upravljanje IP telefonije naj ima možnost z uporabo omenjenih standardnih tehnologij dostopati do podatkov v aktivnem imeniku. Predstavljeno problematiko implementirajte in preizkusite v resničnem sistemu. Navedite svoje izkušnje, praktično uporabnost kritično ovrednotite ter navedite možnosti za nadaljnje delo.

Mentor:

M. Ciglarič

doc. dr. Mojca Ciglarič

Fakulteta za računalništvo in informatiko
2
UNIVERZA V LJUBLJANI

Dekan:

prof. dr. Franc Solina

I Z J A V A O A V T O R S T V U

diplomskega dela

Spodaj podpisani Dejan Strmljan,

z vpisno številko 63020145,

sem avtor diplomskega dela z naslovom:

Preusmeritev klicev v IP-telefoniji s pomočjo SOAP HTTP tehnologije

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom

  doc. dr. Mojca Ciglarič

- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela

- soglašam z javno objavo elektronske oblike diplomskega dela v zbirki »Dela FRI«.

V Ljubljani, dne _____ Podpis avtorja: _____

ZAHVALA

Za izkazano zaupanje, za mentorstvo, pomoč, vodenje in svetovanje pri pripravi diplomskega dela, se najlepše zahvaljujem doc. dr. Mojci Ciglarič.

Diplomsko nalogo sem pripravljal v podjetju NIL Podatkovne komunikacije d.o.o., ki mi je med drugim omogočilo uporabo njihove programske in strojne opreme, za kar se jim iskreno zahvaljujem.

Posebno bi se rad zahvalil Matjažu Črvu, dipl. ing. rač., za korekten in strokoven pregled diplomskega dela, za prijetno diskutiranje ter dobronamerne kritike. Zahvala gre tudi sodelavcem Marku Tišlerji, Mihailu Guguvcevskemu, dipl. ing. rač., in Janu Bervarju ter vsem ostalim, ki so mi kakorkoli pomagali širiti moje »računalniško« obzorje.

Za lektoriranje in izbiro pravih besed se zahvaljujem Urški Tomec Kosec, dr. med.

Zahvalil bi se rad tudi moji Aniti, ki mi je z nasveti, potrpežljivostjo in priganjanjem pomagala pri doseganju želenega cilja.

Mami in oči – hvala, ker sta me požrtvovalno podpirala in mi stala ob strani. Hvala, ker sem si lahko sam izbral svojo poklicno pot. Če pogledam nazaj, vem da sem se odločil pravilno. Hvala za vse – diplomsko delo posvečam vama.

KAZALO VSEBINE

KAZALO VSEBINE I
SEZNAM SLIK II
SEZNAM TABEL II
SEZNAM UPORABLJENIH KRATIC IN SIMBOLOV III
POVZETEK V
ABSTRACT VI
1 UVOD 1
1.1 Osnovni problem in cilj 1
2 TEORETIČNE OSNOVE ZA REŠITEV PROBLEMA 4
2.1 Tehnologija XML 4
2.1.1 Sintaksa dokumenta XML 4
2.1.2 Imenski prostor 5
2.1.3 Spletna storitev 5
2.1.4 SOAP 6
2.1.5 WSDL 7
2.2 LDAP aktivni imenik 8
2.2.1 LDAP operacije 8
2.2.2 Struktura imenika 9
2.3 IP-telefonija 11
3 NADZORNIK KOMUNICIRANJA ZA RAZVIJALCE 13
3.1 Vmesniki na nadzorniku komuniciranja 13
3.2 Vmesnik AXL 14
3.3 Formalna definicija dokumenta XML za preusmeritev klica 16
4 OPIS REŠITVE 19
4.1 Diagram poteka 20
4.2 Zanimivi deli kode v programskem jeziku Java 21
4.2.1 Knjižnica LDAP 21
4.2.2 Knjižnica CM 23
4.2.3 Glavni program 24
5 MOŽNOSTI ZA DELOVANJE APLIKACIJE V REALNEM ČASU 25
5.1 Uporaba programskih knjižnic 25
5.2 Razvojno okolje podjetja Cisco 25
6 OBSTOJEČE REŠITVE 27
6.1 Nastavitev preusmeritve preko vmesnika na telefonu 27
7 SKLEP 28
7.1 Predlog izboljšav aktivnega imenika v podjetju 28
7.2 Možne izboljšave oziroma nadgradnja programa 28
7.2.1 Avtomatska gradnja programske kode iz datoteke WSDL 29
7.3 Ali je smiselno aplikacijo poganjati v realnem času? 29
8 PRILOGE 31
9 LITERATURA 32

SEZNAM SLIK

Slika 1.1: Potek telefonskega klica, ki pride v podjetje 2
Slika 1.2: Grafični prikaz cilja diplomske naloge 3
Slika 2.1: SOAP ovojnica 6
Slika 2.2: Pošiljanje sporočil SOAP na spletno storitev 7
Slika 2.3: LDAP strežnik 9
Slika 2.4: Drevesna struktura aktivnega imenika 10
Slika 2.5: Primer uporabe različnih naprav v omrežju IP-telefonije 11
Slika 3.1: Pošiljanje sporočil SOAP na vmesnik AXL 14
Slika 3.2: Potek HTTPS komunikacije 15
Slika 3.3: Grafični prikaz XML za posodobitev linije 17
Slika 3.4: Grafični prikaz XML-ja za preusmeritev vseh klicev 17
Slika 4.1: Rešitev problema v sliki 19
Slika 4.2: Diagram poteka 20
Slika 5.1: Primer razvojnega orodja v CUAE 26

SEZNAM TABEL

Tabela 2.1: Razlaga okrajšav v atributu 10
Tabela 3.1: Tabela različnih elementov v grafični predstavitvi dokumenta XML 16
Tabela 7.1: Definicija vnosa v aktivnem imeniku podjetja 28

SEZNAM UPORABLJENIH KRATIC IN SIMBOLOV

AD	Active directory (aktivni imenik)
API	Application Program Interface (programski vmesnik, ki zagotavlja, da ima računalniški program na razpolago funkcije operacijskega sistema ali drugega računalniškega programa)
ATA	Analogue Terminal Adapter (analogni vmesnik za povezavo analognega aparata v VoIP omrežje)
AXL	Administrative XML (administrativni XML)
AXL API	Administrative XML Layer Application API (vmesnik, ki omogoča vstavljanje, branje, spreminjanje in brisanje podatkov iz baze nadzornika komuniciranja)
C	Oznaka za programski jezik C
CM	Ime knjižnice, ki smo jo napisali pri razvoju programa
CN	Common Name (v aktivnem imeniku oznaka predstavlja polno ime)
CTI	Computer Telephony Integration (tehnologija, ki omogoča integracijo telefona na računalniku)
ctx	Ime spremenljivke, ki je omenjena v programu
CUCM	Cisco Unified Communications Manager (strežnik za upravljanje IP-telefonije - nadzornik komuniciranja)
DC	Domain Component (v aktivnem imeniku oznaka predstavlja domeno)
DLL	Dynamically Linked Library (v okolju Windows kratica za gonilnike)
DN	Distinguished Name (v aktivnem imeniku oznaka predstavlja unikatno ime)
EMAIL	Electronic Mail (elektronska pošta)
HTTP	Hypertext Transfer Protocol (protokol za izmenjavo nadbesedil ter grafičnih, zvočnih in drugih večpredstavnostnih vsebin na spletu)
HTTPS	Hypertext Transport Protocol Secure sockets (protokol, ki omogoča varno internetno povezavo)
IP	Internet Protocol (internetni protokol)
ISDN	Integrated Services Digital Network (digitalno omrežje z integriranimi storitvami)
ISO	International Standards Organization (organizacija za standardizacijo)
ISP	Internet Service Provider (ponudnik internetnih storitev)
JTAPI	Java Telephony Application Programming Interface (vmesnik, ki omogoča povezovanje računalnika s telefonskimi storitvami)
LDAP	Leightweight Directory Access Protocol (protokol, ki določa dostop do aktivnega imenika)
LDIF	LDAP Data Interchange Format (predpisani tekstovni format za opis informacije v aktivnem imeniku)
MGCP	Media Gateway Control Protocol (protokol za klicno signalizacijo v IP-telefoniji)
MIB	Management Information Base (skladišče upravljavskih informacij, ki jih v upravljavskem sistemu uporablja na primer SNMP)
MIME	Multipurpose Internet Mail Extension (standard za pošiljanje in sprejemanje pošte)
OSI	Open Systems Interconnection (model za medsebojno povezovanje odprtih standardov)

OU	Organizational Unit (v aktivnem imeniku oznaka predstavlja organizacijsko enoto)
PBX	Private Branch Exchange (digitalna ali analogna telefonska centrala, ki povezuje javno in privatno telefonsko omrežje in je v privatni lasti, locirana blizu naročnika)
PSTN	Public Switched Telephone Network (javno komutirano telefonsko omrežje)
RDN	Relative Distinguished Name (v aktivnem imeniku oznaka predstavlja relativno unikatno ime)
RFC	Requests for comments (zahtevek za spremembo)
RPC	Remote Procedure Call (klic za oddaljen postopek)
SASL	Simple Authentication and Security Layer (programsko ogrodje za integracijo varnosti v internetnem protokolu)
SCCP	Skinny Call Control Protocol (protokol za klicno signalizacijo v IP-telefonij)
SIP	Session Initiation Protocol (protokol, ki specificira povezovanje omrežja IP in telefonskega omrežja)
SN	Surname (v aktivnem imeniku oznaka predstavlja priimek)
SNMP	Simple Network Management Protocol (preprost protokol za upravljanje omrežja)
SOAP	Simple Object Access Protocol (standard za spletne storitve, ki temelji na XML)
SQL	Structured Query Language (strukturiran povpraševalni jezik za delo s podatkovnimi bazami)
SSL	Secure Sockets Layer (protokol, ki omogoča varno povezavo preko TCP/IP-omrežja)
TAPI	Telephony Application Programming Interface (vmesnik, ki omogoča povezovanje računalnika s telefonskimi storitvami)
TCP	Transmission Control Protocol (protokol za nadzor transporta)
TLS	Transport Layer Security (protokol, ki omogoča varno povezavo preko TCP/IP omrežja)
UDDI	Universal Description Discovery and Integration (javni register strukturiranih informacij o podjetjih in njihovih spletnih storitvah)
UDP	User Data Protocol (nepovezani protokol transportnega sloja v transportnem skladu TCP/IP)
URI	Uniform Resource Identificator (enotni označevalnik vira)
URL	Uniform Resource Locator (internetni naslov, na katerem se nahaja vsebina)
W3C	World Wide Web Consortium (mednarodna organizacija za standardizacijo)
WSDL	Web Services Description Language (jezik za opisovanje spletnih storitev)
XML	Extensible Markup Language (samo-opisni jezik, ki z uporabo oznak pojasnjuje pomen podatkov v dokumentu)

POVZETEK

Diplomsko delo je namenjeno pregledu možnosti praktične uporabe dokumentov XML in aktivnih imenikov pri upravljanju IP-telefonije v sodobnem podjetju. IP-telefonija predstavlja možnost vzpostavitve telefonskih pogovorov preko paketnega omrežja. Zaradi vse večje razširjenosti te tehnologije, ki ponuja razvijalcem in tudi podjetjem razvoj novih storitev, bomo v tem delu predstavili primer programa, ki izvaja oddaljeno administracijo naprav v IP-telefoniji.

Oddaljena administracija se izvaja s sestavljanjem in pošiljanjem sporočil SOAP, ki imajo obliko dokumenta XML, na strežnik za upravljanje IP-telefonije. Spletna storitev, ki teče na omenjenem strežniku, posluša in vsako prejeto sporočilo ustrezno interpretira ter izvede zahtevo, ki je lahko uspešno ali neuspešno izvedena. Zahteva lahko predstavlja dodajanje, spreminjanje ali brisanje objekta.

Verjetno danes skoraj ni podjetja, ki ne bi imelo delujočega aktivnega imenika ali vsaj razmišljajo o njegovi uvedbi, kjer so na enem mestu zbrane informacije o različnih uporabnikih, aplikacijah, datotekah, tiskalnikih in drugih zmogljivostih. Kot praktična uporaba te tehnologije je v tem delu predstavljeno branje podatkov iz imenika. Glede na vrednost določenega atributa (namestnik) v vnosu pa se izvede ustrezna posodobitev v omrežju IP-telefonije.

Cilj diplomske naloge je teoretična predstavitev in praktična uporabnost tehnologij XML in aktivnih imenikov pri upravljanju in nadzoru naprav v omrežju IP-telefonije.

KLJUČNE BESEDE:
Aktivni imenik, HTTP, IP-telefonija, Java, LDAP, preusmeritev klica, SOAP, XML

ABSTRACT

This Bachelor thesis provides an overview of practical usage of XML and active directories in management of IP telephony in a modern enterprise. IP telephony presents an option of carrying voice calls over IP network. Driven by increasingly wider use of this technology offering a developer and enterprises a development of new services, we will try to show an example of the program which does a remote administration of devices in IP telephony.

Today there is no enterprise without a working implementation of active directory or at least a plan about this solution which can provide all the information on users, applications, files, printers and other resources in one place. As a case of practical use we will present querying the directory. Taking into account the attribute value of one single directory entry we will make a decision on updating the IP telephony network.

The goal of this thesis is both a theoretical presentation and practical usage of XML technologies and active directories in administration and management of IP telephony.

KEYWORDS:
Active directory, call forwarding, HTTP, IP telephony, Java, LDAP, SOAP, XML

1 UVOD

Trenutni razvoj telekomunikacijske industrije bistveno spreminja uporabo telekomunikacijskih omrežij. Ljudje vse več uporabljamo telefonske zveze za prenos podatkov namesto za navadne telefonske klice.

Razumljivo je, da imajo te spremembe velik vpliv na omrežje in operaterje. To v zadnjem času zelo intenzivno povzroča konvergenco telekomunikacijskih omrežij s svojim skladom tehnologij in podatkovnih komunikacijskih omrežij, ki večinoma temeljijo na IP-tehnologiji, v eno skupno omrežje, kar posledično pomeni, da se sedaj tudi telefonski pogovori prenašajo preko IP-omrežij.

Pri IP-telefoniji se telefonski pogovori in faks klici prenašajo preko IP-omrežja, kot je internet, ali preko lokalnega omrežja, namesto preko vsakdanjega javnega telefonskega omrežja (PSTN, ISDN). Kot posledica uvajanja te nove tehnologije prihaja do bistveno nižjih cen govornih klicev, predvsem mednarodnih, saj je internet danes dosegljiv skoraj povsod.

Že nekaj časa se pri slovenskih ponudnikih dostopa do interneta (ISP) pojavlja na spisku njihovih storitev tudi IP-telefonija. Uporabniki se za to storitev čedalje pogosteje odločajo, saj med drugim omogoča tudi bistveno cenejše telefoniranje v domačem omrežju.

Za upravljanje vseh naprav (telefoni, prehodi, mostovi, faksi) skrbi centralni strežnik za upravljanje IP-telefonije, ki ga pri podjetju Cisco imenujejo Cisco Unified Communications Manager (CUCM – nadzornik komuniciranja). Seveda obstaja še nekaj konkurenčnih nadzornikov komuniciranja, ki so bili razviti pri drugih podjetjih, vendar se bomo osredotočili le na Cisco-v produkt (diplomska naloga je nastajala v podjetju, kjer se večinoma ukvarjajo s tehnologijami podjetja Cisco). V okviru diplomskega dela bomo napisali in predstavili program s katerim bomo izvajali oddaljeno administracijo naprav na omenjenem nadzorniku komuniciranja.

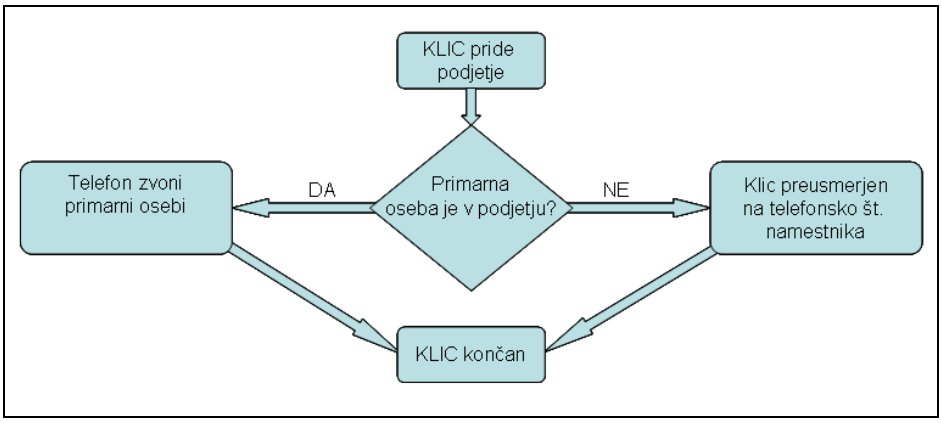
Delo je razčlenjeno po smiselnih enotah – poglavjih, tako da to poglavje vsebuje kratek opis problema in cilj naloge, v drugem poglavju se nahaja teoretična razlaga pojmov in tehnologij, ki jih mora bralec poznati, da bo razumel delovanje programa. V tretjem poglavju je predstavljen nadzornik komuniciranja kot ga vidi razvijalec, v četrtem pa je predstavljena rešitev problema v programskem jeziku Java. V petem poglavju bomo raziskali možnost delovanja aplikacije v realnem času, v šestem pa si bomo ogledali, ali morda že obstaja kje kaka podobna rešitev. Na koncu pa sledijo še sklepne ugotovitve.

1.1 Osnovni problem in cilj

V podjetju je zaposlena primarna oseba, ki obravnava telefonske klice na dodeljeni telefonski številki. V primeru odsotnosti te osebe je določen njen namestnik, ki v tem primeru prevzame vse njene telefonske klice. Naročnik, ki je uvedel IP-telefonijo, ima podatke o namestnikih shranjene v aktivnem imeniku in želi, da sistem IP-telefonije uporablja obstoječo bazo tudi pri nastavljanju preusmeritev klicev. Sistem se mora torej sam »zavedati« dnevnih sprememb v podjetju, zato mora znati podatke pravilno uporabljati. S to rešitvijo bodo v podjetju dosegli večjo dosegljivost in zadovoljstvo strank, saj v primeru, da zaposleni pozabi ročno nastaviti

preusmeritev vseh klicev na telefonsko številko namestnika, za to poskrbi sistem avtomatsko. Brez uvedbe te rešitve pa telefon primarne osebe lahko zvoni tudi v prazno.

Potek in obravnava telefonskega klica, ki pride v podjetje, je predstavljen na naslednji sliki.



Slika 1.1: Potek telefonskega klica, ki pride v podjetje

Naročnik ima v aktivnem imeniku poleg ostalih informacij o osebi shranjen tudi podatek o namestniku. V primeru odsotnosti primarne osebe informacijski sistem podjetja posodobi vnos, tako da atributu *namestnik* priredi novo vrednost, ki je enaka polnemu imenu in priimku osebe, ki nadomešča odsotnega. Takšen način označevanja namestnikov ni dober, ker se ime in priimek zaposlenega lahko v podjetju podvoji, pri čemer pride do napačnega delovanja programa. Žal ima podjetje tako organizacijo aktivnega imenika, kar je bilo potrebno pri samem razvoju aplikacije tudi upoštevati.

V aktivnem imeniku so za posamezno osebo shranjeni naslednji podatki:

- ime
- priimek
- telefonska številka
- namestnik - ime in priimek osebe, na katero želimo preusmeriti vse klice

Osnovni cilj tega diplomskega dela je teoretična predstavitev in praktična uporabnost tehnologij XML in aktivnih imenikov pri upravljanju IP-telefonije v sodobnem podjetju.

V diplomskem delu je predstavljena spletna storitev, ki teče na nadzorniku komuniciranja in omogoča aplikacijam oddaljeno administracijo naprav v IP-telefoniji. Oddaljena administracija se izvaja s pomočjo aplikacije, ki je predstavljena v tem delu in teče na samostojnem strežniku. Aplikacija sestavi XML, nato pa ga pošlje na omenjeno spletno storitev, ki glede na vsebino prejetega sporočila izvede ustrezno akcijo.

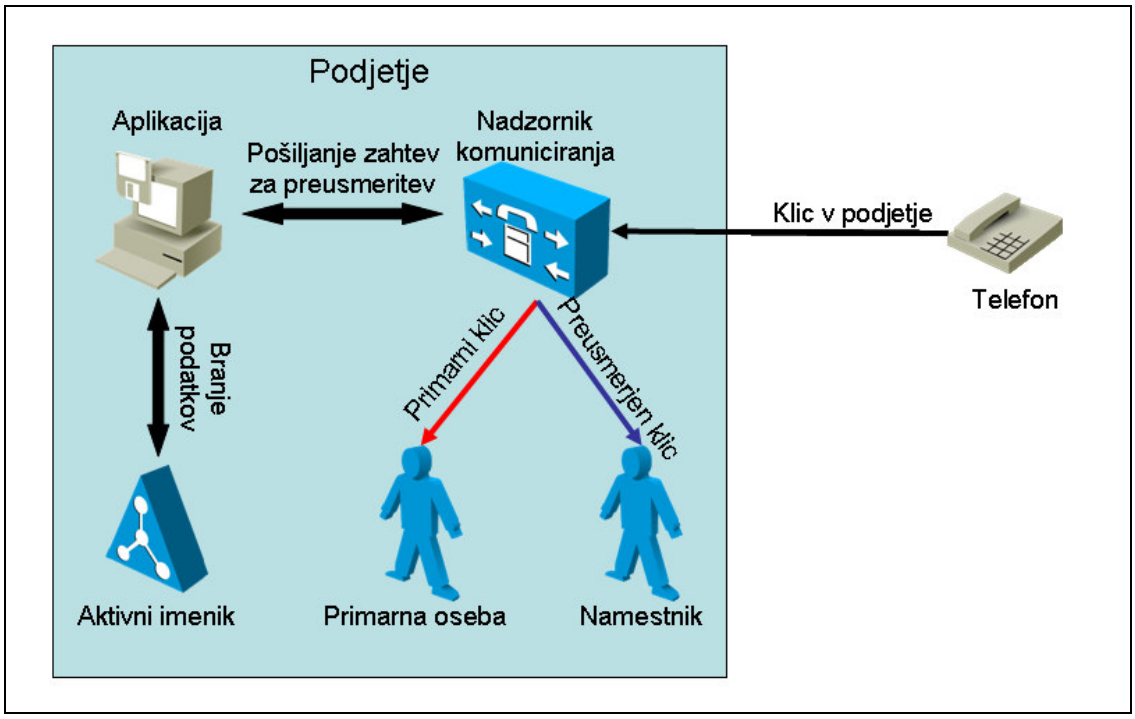
Aktivni imenik služi kot baza, iz katere aplikacija zajema podatke ter glede na vsebino izvede ustrezno akcijo. Predstavili bomo osnovne pojme aktivnih imenikov in podali nekaj osnovnih funkcij za delo z njimi.

Program, izdelan in predstavljen v okviru te diplomske naloge, se enkrat dnevno »sprehodi« čez aktivni imenik in preveri za vsako osebo (vnos) posebej, ali je potrebno preusmeriti vse telefonske klice prebranega vnosa na telefonsko številko namestnika. V primeru, da je

vrednost atributa *namestnik* v vnosu prazna, se zahteva za preusmeritev prekliče, v nasprotnem primeru pa se poišče telefonsko številko namestnika in izvede preusmeritev vseh klicev na poiskano številko. Preusmeritev klica se doseže s pravilno sestavljenim dokumentom XML, ki ga pošljemo nadzorniku komuniciranja.

Na spodnji sliki je predstavljen cilj diplomske naloge. Za boljše razumevanje se lahko delovanje programa razdeli v dve točki:

- iskanje potrebnih podatkov po aktivnem imeniku
- pošiljanje dokumenta XML nadzorniku komunikacij



Slika 1.2: Grafični prikaz cilja diplomske naloge

2 TEORETIČNE OSNOVE ZA REŠITEV PROBLEMA

Za uspešno implementacijo rešitve, ki združuje tri zelo pomembne tehnologije (XML, aktivni imeniki in IP-telefonija), je potrebno vsaj osnovno poznavanje vsake od njih. Omenjene tehnologije pokrivajo zelo široko področje znanja in z vsakega področja bi lahko nastalo več diplomskih nalog. Naša naloga je, da poskušamo te tri tehnologije uporabiti v praktični aplikaciji. Za razvoj takšne aplikacije je na začetku potrebno prebrati veliko dokumentacije, kar je izredno zamudno. Namen tega poglavja je, da bi bodoči razvijalec pridobil osnovna znanja iz omenjenih tehnologij na enem samem mestu.

V nadaljevanju tega poglavja bodo bralcu razloženi osnovni pojmi, ki jih mora poznati, da bo lahko razumel način delovanja programa. Najprej so podane osnove tehnologije XML, kjer med drugim bralec spozna sintakso in najpomembnejše pojme, ki jih ta tehnologija zajema. V nadaljevanju sledi predstavitev aktivnih imenikov ter operacij za delo z njimi. V zaključku poglavja so razloženi osnovni pojmi IP-telefonije, predstavljeno pa je tudi tipično omrežje IP-telefonije.

2.1 Tehnologija XML

Tehnologija eXtensible Markup Language (XML) temelji na uporabi specifikacij XML. Je format zapisa poljubne hierarhične globine na uporabniku prijazen način. Danes je njegova uporaba zelo široka, zato ni čudno, da se v zadnjem času ogromno podjetij odloča za uvedbo te tehnologije. XML je zelo nazorno predstavljen v literaturi [1], zato je nadaljevanje tega podpoglavja povzeto po njeni vsebini.

V tem poglavju si bomo podrobneje ogledali:

- sintakso dokumenta XML
- imenski prostor XML
- spletne storitve
- SOAP
- WSDL

2.1.1 Sintaksa dokumenta XML

Vsak podatek je shranjen med etiketama, ki označuje začetek in konec. Vsaka etiketa ima lahko tudi atribute. Etikete lahko gnezdimo, atributov pa ne moremo.

Primer dokumenta XML brez atributov:

```
<Oseba>
        <Ime> Anita </Ime>
        <Telefon> 01 3456789 </Telefon>
        <Eposta> anita.pestotnik@email.si </Eposta>
        <Starost> 29 </Starost>
</Oseba>
```

Primer dokumenta XML z uporabo atributov

<Oseba ime="Anita" telefon="01 3456789" Eposta ="anita.pestotnik@email.si" />

Kot je iz obeh primerov razvidno, lahko XML shranjuje podatke kot elemente, ali pa kot vrednosti atributov. Elemente in atribute poimenujemo glede na pomen, kar ljudem omogoča dokaj enostavno razumevanje podatkov, ki so shranjeni v dokumentih XML. V le-teh je označen začetek in konec elementa.

2.1.2 Imenski prostor

XML imenski prostori (XML Namespaces) določajo sintakso za povezovanje oznak XML in atributov z ustreznimi besednjaki. Če se združi dva dokumenta XML v enega in če se v obeh uporabi isto oznako XML, se mora določiti kategorije oziroma imenska področja za vsako od njih (npr.: <zaposleni:ime> in <računalnik:ime>). Imena imenskih področij (predponi *zaposleni* in *računalnik*) se mora povezati z Uniform Resource Identificator (URI) kot v naslednjem primeru:

```
<zaposleni:oseba  xmlns:zaposleni="http://....">
      <zaposleni:ime>Anita</zaposleni:ime>
      <zaposleni:priimek>Pestotnik
      </zaposleni:priimek>
</zaposleni:oseba>
```

Imenske prostore se določi po URI naslovih. V praksi URI ni potreben, da obstaja, saj služi le kot pripomoček za ločevanje imenskih prostorov med seboj. Če se v pod-elementu ne uporabi imenskega prostora (na primer namesto seznam:Ime pišemo le Ime), procesor to obravnava, kot da ni v tem imenskem prostoru.

2.1.3 Spletna storitev

Spletne storitve (Web services) omogočajo tujim aplikacijam povezovanje in povpraševanje po določenih informacijah. Storitve so namenjene predvsem razvijalcem novih aplikacij, ki lahko to storitev uporabijo.

Če želimo svetu ponuditi neko svojo komponento kot storitev, jo moramo opisati z jezikom za opisovanje. Lahko je registrirana tudi v kakšnem od namenskih registrov (UDDI), kjer jo lahko potem zainteresirani razvijalci tudi hitro najdejo. V zadnjem času so najbolj znane in najbolj zanimive tiste spletne storitve, ki ponujajo podatke o najnovejših tečajih delnic, izvajajo različne konverzije (med merskimi enotami, valutami itd.), ponujajo najnovejše podatke o vremenu itd.

Na področju spletnih storitev sta za nas zelo pomembna naslednja besednjaka XML:

- SOAP (Simple Objects Access Protokol)
- WSDL (Web Services Description Language)

2.1.4 SOAP

SOAP je protokol XML za izmenjavo podatkov med aplikacijami na svetovnem spletu. Aplikacijam omogoča dostop do oddaljenih objektov in prenos strukturiranih podatkov XML po omrežju. SOAP za prenos podatkov uporablja Hypertext Transfer Protocol (HTTP), ki je bil v osnovi namenjen prenosu nadbesedil, v praksi pa se uporablja še za vrsto drugih podatkovnih prenosov. V času, ko je nastajal SOAP, še ni obstajal standard za predstavitev različnih podatkovnih tipov v XML, zaradi česar prva različica tega protokola ni bila tako preprosta, kot so si to zamislili avtorji [2].

SOAP-sporočila se pošiljajo tako v smeri strežnik-odjemalec kot tudi obratno. Sporočilo vsebuje povpraševanje po določeni akciji ali odgovor na povpraševanje. SOAP ovojnica je dejansko dokument XML, zato je enostavno berljiva. Sporočilo SOAP je sestavljeno iz treh delov:

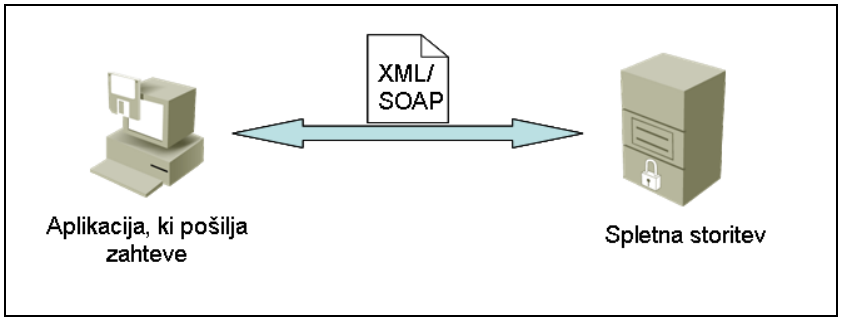
- XML ovojnica (Envelope) za prenos imen metod in parametrov do oddaljenih objektov
- pravila za kodiranje aplikacijskih tipov objektov, ki se prenašajo kot argumenti
- sintaksa za oblikovanje zahtev po izvajanju metod pri objektih in vrnitev rezultata

Slika 2.1: SOAP ovojnica

Na sliki 2.1 je prikazana struktura sporočila, ki vsebuje naslednje elemente:

- **ovojnica –** korenski element
- **glava –** izbirno dodamo dodatne napotke (npr. podpora transakcijam, varnost itn.)
- **telo –** vsebuje podatke v obliki XML. V telesu sporočila je lahko izbirno tudi napaka (SOAP Fault), s katerim ponudnik storitve (SOAP prejemnik) sporoča, da je prišlo do napake.

SOAP omogoča odjemalcem klicanje oddaljenih procedur (RPC). Uporabniki izvedejo zahtevo s pošiljanjem sporočila SOAP na spletno storitev. Strežnik se na zahtevo odzove z odgovorom, ki je prav tako SOAP-sporočilo.



Slika 2.2: Pošiljanje sporočil SOAP na spletno storitev

V nadaljevanju predstavljamo dva različna primera SOAP-sporočila, kjer prvi predstavlja poslano zahtevo, drugi pa odgovor na zahtevo:

Primer sporočila, ki predstavlja zahtevo SOAP (npr. Koliko je vredna delnica Petrola?):

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
        SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
        <SOAP-ENV:Body>
                <m:ZadnjaProdajnaCena
                        xmlns:m="http:/www.borza.si">
                        <symbol>Petrol</symbol>
                </m:ZadnjaProdajnaCena>
         </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Primer sporočila, ki predstavlja odgovor SOAP na zgornjo zahtevo:

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
        SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
        <SOAP-ENV:Body>
                <m:ZadnjaProdajnaCenaResponse
                        xmlns:m="http://www.borza.si">
                        <Price>455€</Price>
                </m:ZadnjaProdajnaCenaResponse>
        </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

2.1.5 WSDL

WSDL je jezik za opisovanje spletnih storitev in deluje kot vmesnik med odjemalcem in strežnikom. Protokol SOAP definira dostop do oddaljenih objektov in procedur, imena teh objektov in njihovi tipi pa so definirani v datotekah WSDL. Datoteke WSDL so v obliki XML in so praviloma shranjene na strežniku, ki ponuja določeno storitev. Ti opisi so abstraktni in

se pri dostopu do neke spletne storitve preslikajo v točno določen mrežni protokol (SOAP 1.1, HTTP GET/POST, MIME) in v določena sporočila, ki se prenašajo po tem protokolu. Na ta način WSDL omogoča lažjo komunikacijo med ponudniki spletnih storitev in njihovimi uporabniki.

V programskem jeziku Java lahko uporabnik s pomočjo namenskega programa (npr. grafičnega orodja Eclipse z dodatkom Axis za spletne storitve) iz WSDL datoteke avtomatsko zgradi vse potrebne objekte in funkcije, ki so potrebne za uspešno komunikacijo s spletno storitvijo. WSDL datoteke programerju na ta način zelo olajšajo delo, saj za uspešno vključitev spletne storitve v svojo aplikacijo ne potrebuje skoraj nobenega znanja o tehnologiji XML in SOAP,

2.2 LDAP aktivni imenik

Lightweight Directory Access Protocol (LDAP) je protokol za dostop do imenikov na osnovi odjemalec-strežnik, kjer odjemalec preko TCP protokola vzpostavi sejo z strežnikom LDAP in preko njega pošilja zahtevke ter sprejema odgovore. Protokol LDAP je nazorno opisan na spletni strani [3], zato so v nadaljevanju osnovni podatki o tem protokolu povzeti po omenjeni literaturi.

Imenik je množica objektov s podobnimi atributi, ki so organizirani logično in hierarhično. Največkrat podajamo kot primer telefonski imenik, ki vsebuje imena oseb ali organizacij, razvrščene po abecednem redu, kjer je zraven vsakega imena shranjen naslov in telefonska številka.

LDAP aktivni imenik (AD) je specializirana baza podatkov, ki je optimizirana za veliko število branj ter iskanj. Občasno se v bazo tudi piše oziroma posodablja določena polja. V aktivni imenik shranjeni podatki se ne spreminjajo zelo pogosto, saj v njem praviloma shranjujemo informacije o zaposlenih, pravice oseb za dostop do določenih virov v omrežju podjetja itd.

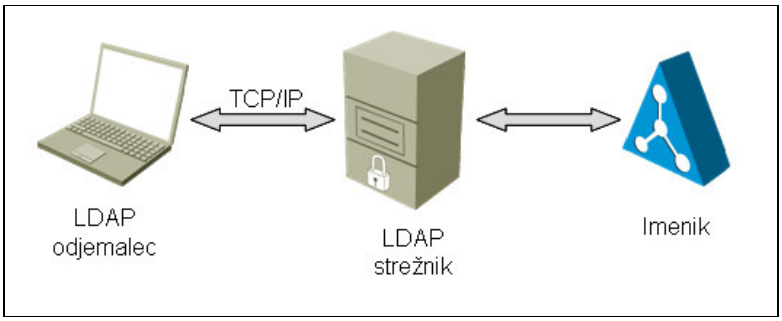
LDAP določa aplikacije s standardnimi metodami za dostop in morebitno spreminjanje shranjenih informacij v imeniku. Ta struktura omogoča podjetju, da centralizira informacije vseh uporabnikov na enem samem mestu, do katerega pa lahko dostopa več aplikacij. Rezultat je občutno zmanjšanje stroškov vzdrževanja podatkov.

V nadaljevanju tega poglavja si bomo podrobneje ogledali še:

- LDAP operacije in
- strukturo aktivnega imenika

2.2.1 LDAP operacije

Odjemalec začne sejo LDAP s povezavo na strežnik LDAP na privzeta TCP vrata 389 (slika 2.3). Odjemalec nato pošlje zahtevo na strežnik, ki ustrezno vrne odgovor. Z določenimi izjemami odjemalcu ni potrebno čakati na odgovor strežnika, preden pošlje naslednjo zahtevo, strežnik pa lahko odgovarja na zahteve v poljubnem vrstnem redu.



Slika 2.3: LDAP strežnik

Spodaj so naštete osnovne operacije, ki jih lahko odjemalec od strežnika zahteva:

- **start TLS** – uporabi LDAP verzijo 3 Transport Layer Security (TLS) za varno povezavo
- **bind** – s to operacijo se odjemalec poveže na strežnik in se mu predstavi. Navesti mora ustrezno unikatno ime, ki določa, kje v drevesu želi vstopiti v imenik, ter parametre za preverjanje istovetnosti.
- **iskanje** posameznega vnosa v imeniku
- **dodaj** nov vnos
- **izbriši** vnos
- **spremeni** oziroma **premakni** vnos

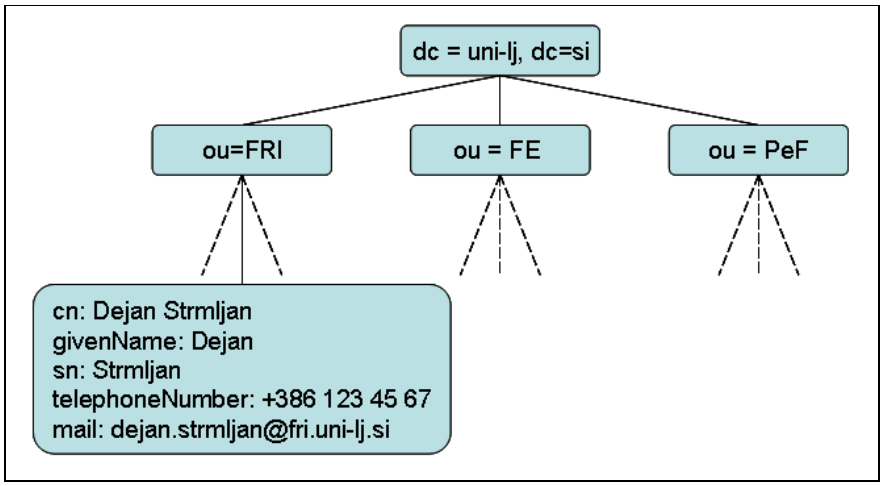
Če želi uporabnik uspešno izvajati določene operacije, ki so naštete v prejšnjem odstavku, mora imeti nastavljene ustrezne pravice. Večina uporabnikov ima zgolj bralne pravice, pisalne pa imajo le tisti, ki jih dejansko potrebujejo.

2.2.2 Struktura imenika

Osnovna enota informacije v imeniku je vnos, ki predstavlja zbirko informacij o nekem objektu, osebi, ustanovi, strojni opremi itn. Vsak vnos vsebuje skupino atributov, kjer vsak opisuje določeno podrobnost objekta. Vsak atribut ima svoje ime (tip ali opis) in eno ali več vrednosti. Tip pove, kakšna vrsta informacije je vsebovana v atributu, vrednost pa določa dejansko informacijo [4]. Če atribut, ki ga želimo dodati v aktivni imenik, ni definiran, potem pride do napake.

Aktivni imenik je organiziran v drevesno strukturo (slika 2.4), ki je zelo podobna organizaciji datotek in imenikov na trdem disku tako v okolju Linux kot okolju Windows. Za razliko od organizacije na disku pri aktivnih imenikih vsak vozel v drevesu vsebuje podatke in ima lahko hkrati tudi potomce [4].

Aktivni imenik vsebuje vnose v poddrevesih. Koren drevesa je prvi vnos, ki ponavadi predstavlja domeno. Kot primer je lahko domena *dc=uni-lj.si, dc=si*, ki globlje v notranjosti vsebuje nove vnose. Otroci pripadajo določenemu objektu. Strežniki lahko vsebujejo tudi reference na ostale strežnike. Tako bi lahko na primer poskus dostopa na *ou=fri,dc=uni-lj.si, dc=si* sprožil povezavo na drugi strežnik, ki vsebujejo podatke o tem poddrevesu. Odjemalec se nato poveže na drugi strežnik in tam poišče zahtevane podatke. Nekateri strežniki pa podpirajo tako imenovano verigo, kjer strežnik sam poišče zahtevane podatke pri drugih strežnikih in jih vrne odjemalcu.



Slika 2.4: Drevesna struktura aktivnega imenika

Vsak vnos ima unikatno ime, ki ga v angleščini imenujejo Distinguished Name (DN). Unikatno ime je sestavljeno iz relativnega unikatnega imena (Relative DN) in očetovega unikatnega imena. Unikatno ime si lahko predstavljamo kot absolutno pot do datoteke, relativno unikatno ime pa kot relativno pot do datoteke v mapi. Unikatno ime se lahko skozi življenjski čas vnosa spreminja, saj se lahko posamezni vnosi premikajo po drevesu. Vnose predstavljamo v posebnem formatu, ki ga v angleščini imenujejo LDAP Data Interchange Format (LDIF). Spodaj je podan primer takega zapisa.

dn: cn=Dejan Strmljan , ou=fri, dc=uni-lj ,dc=si
cn: Dejan Strmljan
givenName: Dejan
sn: Strmljan
telephoneNumber: +386 123 45 67
mail: dejan.strmljan@fri.uni-lj.si
objectClass: inetOrgPerson
objectClass: organizationalPerson
objectClass: person
objectClass: top

V zgornjem primeru se v prvi vrstici nahaja oznaka DN, ki predstavlja unikatno ime vnosa, ki ni atribut, niti ni del vnosa. Relativno unikatno ime predstavlja *cn=Dejan Strmljan,* očetovo unikatno ime pa predstavlja *ou=fri,dc=uni-lj.si dc=si*. Pomen ostalih okrajšanih imen atributov v zgornjem primeru je razložen v naslednji tabeli.

Okrajšava	Ime atributa	Slovenski opis
CN	Common Name	Polno ime
DC	Domain Component	Komponenta domene
OU	Organizational Unit	Organizacijska enota
Mail	Email	Elektronska pošta
SN	Surname	Priimek
ObjectClass	Object class	Vrednost objektnega razreda definira dovoljene atribute v vnosu

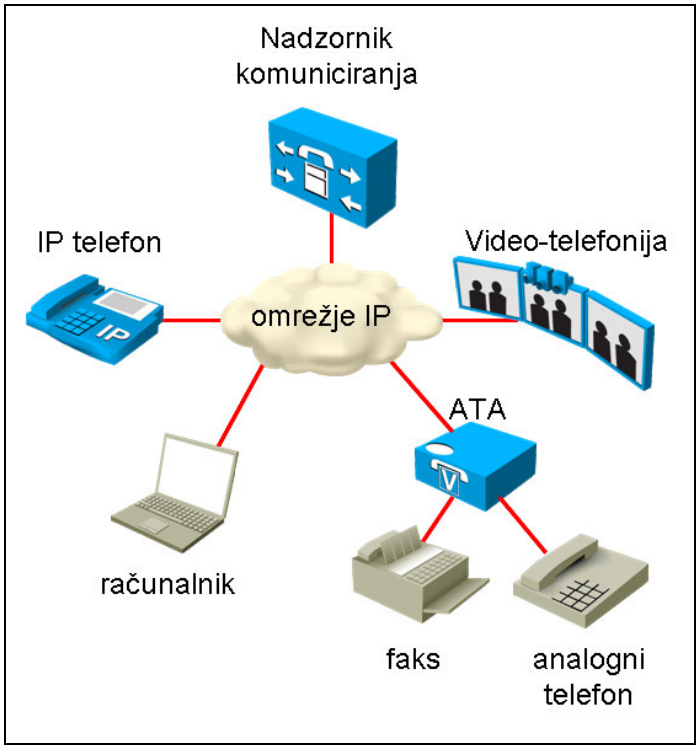
Tabela 2.1: Razlaga okrajšav v atributu

2.3 IP-telefonija

IP-telefonija ali prenos govora preko paketnega omrežja IP je telefonija preko internetnega protokola (IP). IP-telefonija omogoča prenos komunikacijskih podatkov (govor in faks) preko internetnega omrežja. V nadaljevanju so osnovne informacije o tej tehnologiji povzete po študijskem priročniku [5], bralec pa si lahko dodatne informacije pridobi tudi iz drugih virov, kot na primer [6].

IP-telefonija je tehnologija, ki omogoča, da se omrežje IP poleg prenosa podatkov uporablja še za zvokovne aplikacije (telefonija, tele-konferenca, video-telefonija itn.). IP-telefonija določa način prenosa klicev preko omrežja IP, vključno z načinom digitaliziranja zvoka. IP-telefon (ali kaka druga naprava) pretvori analogni govorni signal v digitalni format, ki se shrani v IP-paket. Le-ta je nato primeren za prenos preko obstoječega omrežja. Če poteka klic iz omrežja IP na klasično telefonsko številko, se mora signal, preden doseže cilj, pretvoriti v klasičen analogni telefonski signal. IP-telefonija omogoča izvajanje klicev direktno iz računalnika s posebnim IP-telefonom ali s tradicionalnim analognim telefonom, ki pa mora biti priključen na poseben analogni vmesnik (ATA).

Za vzpostavitev IP-telefonije torej potrebujemo le delujoče omrežje IP in naprave, ki so sposobne medsebojno komunicirati z uporabo standardiziranih pravil in po vnaprej dogovorjenih pravilih, postopkih in aplikacijah.



Slika 2.5: Primer uporabe različnih naprav v omrežju IP-telefonije

Za nadzor in upravljanje IP-telefonije skrbi programska oprema, imenovana nadzornik komuniciranja, ki teče na posebnem strežniku z Linux operacijskim sistemom. Nadzornik komuniciranja nadzira vse aktivne naprave v omrežju IP-telefonije. Primerjali bi ga lahko s centralo Private Branch Exchange (PBX) v klasični Public Switched Telephone Network

(PSTN) telefoniji, saj omogoča enake funkcije, le da je namenjen uporabi v omrežju IP. Njegova velika prednost v primerjavi s klasično centralo je v tem, da je lokacijsko neodvisen in ne potrebuje fizične povezave z vsakim IP-telefonom, zadostuje namreč že IP-povezljivost. Zelo uspešno ga lahko uporabimo kot zamenjavo PBX-a, saj omogoča celotno klicno signalizacijo tudi za klasične telefone.

Klicna signalizacija je zmožnost ustvarjanja in izmenjavanja kontrolnih informacij pri vzpostavljanju, nadziranju in prekinjanju povezave med dvema končnima točkama. IP-telefonija omogoča več različnih načinov signalizacije:

- H.323,
- Media Gateway Control Protocol (MGCP) ali
- Session Initiation Protocol (SIP) ali
- Skinny Call Control Protocol (SCCP)

Signalizacijske protokole lahko razdelimo v dve skupini, in sicer v točka-točka ali odjemalec-strežnik arhitekturo. SIP in H.323 sta primera točka-točka signalizacijskih protokolov, kjer končne naprave vsebujejo inteligenco za vzpostavljanje in prekinjanje klicev, znajo pa tudi interpretirati kontrolna sporočila. SCCP in MGCP sta primera tipa odjemalec-strežnik protokolov, kjer končne točke ne vsebujejo inteligence za kontrolo klicev, ampak samo pošiljajo in sprejemajo kontrolna sporočila od nadzornika komuniciranja. Primer SCCP klicne signalizacije je dogodek, ko oseba dvigne slušalko na IP-telefonu, telefon pošlje sporočilo nadzorniku komuniciranja, da je slušalka dvignjena, nadzornik pa nato telefonu pošlje klicni ton.

Ker tema diplomske naloge ne obsega podrobne razlage omenjenih protokolov, lahko bralec podrobnejše informacije o le-teh lahko v literaturi [5], [8], [9], [10] in [11].

3 NADZORNIK KOMUNICIRANJA ZA RAZVIJALCE

Že od samega začetka je bila IP-telefonija pri podjetju Cisco zasnovana tako, da omogoča razvijalcem široko paleto možnosti. Z odločitvijo, da bodo uporabili format XML, ki se je takrat šele dobro začel razvijati in je bil v marsičem še nedorečen, so naredili eno izmed najboljših odločitev.

Zaradi velike razširjenosti različnih tehnologij tega podjetja po vsem svetu, najdemo na spletu veliko različnih podjetij, ki so razvili različne aplikacije za lažje delo s tehnologijami omenjenega proizvajalca. Te rešitve so praviloma zelo drage in zato za manjše podjetje (70 zaposlenih), ki preprodaja Cisco opremo, lahko predstavljajo velik strošek in nekonkurenčnost na trgu. Zato so mala podjetja največkrat prisiljena sama razvijati aplikacije različnih funkcionalnosti, ki izražajo naročnikove želje. Na srečo podjetje Cisco za svoje produkte ponuja veliko informacij na svoji spletni strani [12], ki jih razvijalec lahko uporabi pri razvoju nove storitve. Večina podatkov (programska koda, postopki, definicije dokumentov XML itn.), ki jih potrebujemo za izdelavo programa, predstavljenega v tem delu, je tako pridobljena iz omenjenega vira.

V nadaljevanju tega poglavju so na kratko predstavljeni različni vmesniki na nadzorniku komuniciranja, ki jih ima razvijalec na razpolago. Vsak vmesnik ponuja točno določene funkcionalnosti. Glede na želje naročnika, se mora razvijalec odločiti, s katerim vmesnikom bo najhitreje razvil želeno aplikacijo. Vmesniki večinoma uporabljajo za komunikacijo tehnologijo XML. Program, ki je predstavljen v tem delu, za svoje delovanje uporablja vmesnik AXL, ki ga bomo obravnavali posebej. Na koncu je predstavljen še primer definicije dokumenta XML, ki ga razume vmesnik AXL. Ti dokumenti imajo vnaprej določeno strukturo, ki je definirana s strani proizvajalca. Prenašajo se v telesu sporočila SOAP in omogočajo oddaljeno administracijo naprav v omrežju IP-telefonije..

3.1 Vmesniki na nadzorniku komuniciranja

Nadzornik komuniciranja razvijalcem nudi vrsto različnih vmesnikov, ki jih njihove aplikacije lahko uporabljajo. Vmesnike lahko razdelimo v več skupin, ki jih bomo v nadaljevanju tudi predstavili.

V administrativne namene se uporabljata naslednja dva vmesnika:

- Administrativni XML (AXL) vmesnik – omogoča oddaljeno administracijo objektov na nadzorniku komuniciranja.
- Extension Mobility Servis API – storitev, temelječa na XML in HTTP, ki avtomatično nastavi uporabniški profil (telefonska številka, preusmeritve, zvonjenje itn.) na telefonu. Omenjeni servis bi lahko uporabili v hotelski aplikaciji pri prijavljanju in odjavljanju gostov v hotelskih sobah.

Vmesnike za upravljanje klicev delimo na:

- Cisco Telephony Application Programming Interface (TAPI) in Cisco Wave Driver – ponuja knjižnice za pisanje v programskem jeziku C ali C++.
- Cisco Java TAPI (JTAPI) – ponuja knjižnice za pisanje v programskem jeziku Java.
- Cisco WebDialer – spletni vmesnik, ki omogoča izvajanje klicev.

Vmesnika TAPI in JTAPI ponujata zelo dobro kontrolo klicev s pomočjo integracije računalnika in telefonije (Computer/Telephony Integration – CTI). Aplikacija se z uporabo knjižnic lahko registrira na nadzorniku komuniciranja kot pravi telefon. Prevzame lahko polno kontrolo nad klicem, ki ga lahko vzpostavi, sprejme, preusmeri in prekine. Aplikacija pa lahko tudi nadzira ali kontrolira naprave v omrežju. Z vmesnikom WebDialer pa lahko izvajamo le klice in smo tako za razliko od prejšnjih dveh vmesnikov veliko bolj omejeni.

Vmesnike, ki ponujajo informacije v realnem času, razdelimo na:

- XML storitve – v realnem času z uporabo sporočil SOAP dobimo nekatere podatke iz nadzornika komuniciranja.
- Simple Network Management Protocol/Management Information Base (SNMP/MIB) - s pomočjo tega preprostega protokola za upravljanje omrežja lahko v realnem času ugotovimo stanje omrežja.

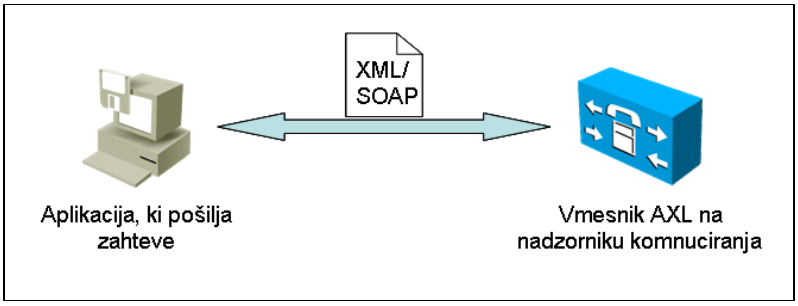
Aplikacija, predstavljena v tem diplomskem delu, pri svojem delovanju uporablja vmesnik AXL, zato je v naslednjem podpoglavju le-ta podrobneje predstavljen. Če bralca zanimajo podrobnejši opisi katere druge storitve, le-te najde v literaturi [12].

3.2 Vmesnik AXL

Vmesnik AXL (AXL API) omogoča aplikacijam oddaljen dostop do shranjenih konfiguracij na nadzorniku komuniciranja z uporabo XML SOAP protokola. Vmesnik na strežniku teče kot spletna storitev, ki sprejema zahteve SOAP. Skoraj vsaka stvar je v bazi nadzornika komuniciranja predstavljena kot objekt z atributi. Vmesnik AXL ponuja mehanizem za kreiranje, branje, posodabljanje in brisanje teh objektov. Posamezen objekt lahko predstavlja uporabnika, napravo, linijo itn.

Vmesnik AXL omogoča piscu programa, da dostopa do podatkov, shranjenih na nadzorniku komuniciranja, s pošiljanjem ter sprejemanjem podatkov v obliki dokumentov XML, namesto da za to uporablja težko berljive binarne knjižnice ali Dynamically Linked Library (DLL).

Metode vmesnika AXL, ki jih lahko drugače imenujemo tudi zahteve, se izvajajo z uporabo protokola SOAP prek HTTP.



Slika 3.1: Pošiljanje sporočil SOAP na vmesnik AXL

S pomočjo vmesnika AXL lahko izvajamo direktne poizvedbe SQL, kjer je stavek SQL podan v telesu sporočila SOAP. Vmesnik razume tudi SOAP metode, s katerimi dodajamo, spreminjamo ali brišemo objekte. SOAP metoda je dejansko dokument XML v telesu sporočila SOAP, ki ima v naprej določeno strukturo in bo predstavljen v naslednjem poglavju.

Storitev je dosegljiva na naslovu https://<NadzornikKomunciranja>:8443/axl/.

Odjemalci lahko tečejo na različnih operacijskih sistemih in komunicirajo s strežnikom z odprtim standardnim protokolom SOAP. Strežnik sprejme sporočilo SOAP in izvede zahtevo, ki se nahaja v telesu. V primeru uspešno izvedene zahteve sistem vrne ustrezen odgovor, ki je poimenovan natanko tako, kot je bila poimenovana poslana zahteva, razlika je le v tem, da je pri odgovoru dodana beseda Response.

Na primer: odgovor XML na poslano zahtevo addPhone se poimenuje addPhoneResponse.

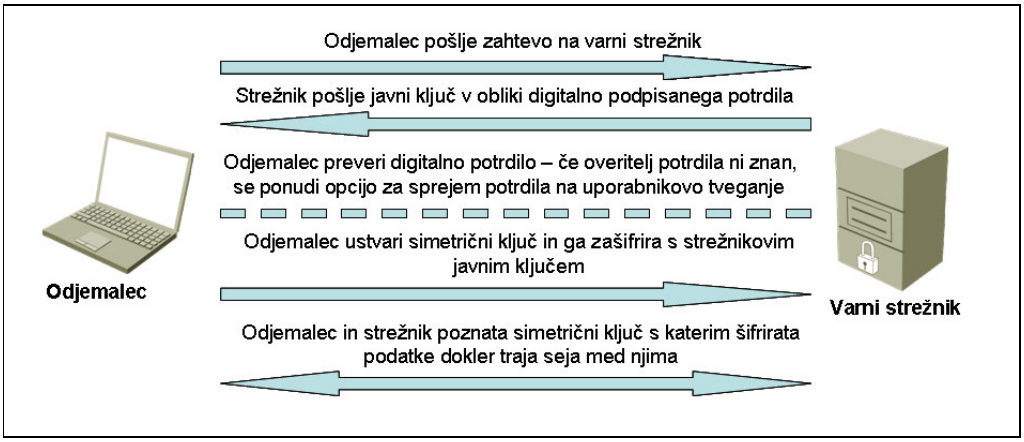
Če pride do izjeme na strežniku ali do kakršnekoli druge napake med procesiranjem zahteve, strežnik vrne napako, ki je vsebovana v telesu SAOP-sporočila (SOAP Fault). Vzrok napake je lahko ukaz, ki ga strežnik ne pozna, lahko pa je razlog za napako tudi napačno sestavljen dokument XML. V odgovoru je izbirno lahko še dodatno razložen vzrok napake.

Pošiljanje zahtev je popolnoma neodvisno od računalniškega okolja, na katerem teče odjemalec. Nadzornik komuniciranja in vsi njegovi vmesniki so dostopni preko varne Hypertext transfer Protocol over secure socket Layer (HTTPS) povezave, ki omogoča kriptografsko zaščito podatkov.

HTTPS je vlaganje protokola HTTP v zaščitni sloj SSL ali TLS, ki varuje promet nad četrtim slojem Open Systems Interconnection (OSI) referenčnega modela (nad TCP ali UDP). SSL in TLS nudita kriptografsko zaščito prenosa podatkov, ki vsebuje:

- šifriranje podatkov, tako da so med prenosom neberljivi za nepooblaščene osebe
- zagotavljanje celovitosti podatkov, s katero lahko ugotovimo nepooblaščene spremembe, ki so se zgodile med prenosom
- zagotavljanje izvora podatkov, s katerim lahko prepoznamo pooblaščenega pošiljatelja podatkov
- zagotavljanje identitet (overjanje) partnerjev v seji SSL ali TLS, kjer odjemalec prepozna (overi) strežnik ter strežnik (opcijsko) overi odjemalca

Komunikacija po varni povezavi je predstavljena na naslednji sliki.



Slika 3.2: Potek HTTPS komunikacije

V postopku, ki je prikazan na zgornji sliki, se odjemalec in strežnik dogovorita, kateri simetrični ključ bosta uporabljala v nadaljevanju seje. Šifriranje podatkov s simetričnim

ključem je veliko hitrejše in procesorsko manj zahtevno, kar je za uporabnika najbolj pomembno. Pri sprejemu digitalnega potrdila je najboljši način, da preverimo ali je izdan s strani zaupanja vredne agencije in da govorimo s pravim partnerjem (t.j. preverimo njegovo ime v potrdilu). Vse to je lahko seveda avtomatsko, stvar pa se podre, če je potrdilo izdano s strani nezaupanja vredne agencije, če je potrdilo poteklo, ali če se ime partnerja ne sklada s pričakovanim imenom. V primeru, da tak certifikat sprejmeš, se ti podre vsa zaščita SSL ali TLS [13].

Podrobnejšo razlago omenjenih kriptografskih postopkov si lahko bralec prebere v literaturi [2] in [13]. Če želimo, da naša aplikacija uspešno pošlje zahtevo SOAP nadzorniku komuniciranja preko HTTP POST metode, mora znati izvesti postopek, ki je grafično predstavljen na sliki 3.2. Aplikacijo lahko napišemo tako, da za sprejem potrdila ne potrebujemo interakcije uporabnika, ampak lahko kar privzamemo, da je strežnik varen, tako da se sprejem digitalnega potrdila izvede avtomatsko.

Trenutno je dolžina sporočila SOAP, ki ga lahko pošljemo nadzorniku komuniciranja, omejena na 40 KB. Če je dolžina daljša, potem odjemalec dobi odgovor z vsebino o napaki (SOAP Fault).

3.3 Formalna definicija dokumenta XML za preusmeritev klica

Za upravljanje nadzornika komuniciranja preko vmesnika AXL obstaja veliko različnih definicij dokumentov XML, ki so v grafični obliki dostopni na spletni strani proizvajalca [12]. V strokovni literaturi se je za takšne vrste dokumentov XML, s katerim izvajamo oddaljeno administracijo, uveljavil izraz administrativni XML (AXL). Ti dokumenti XML se tako prenašajo v telesu SOAP-sporočila.

V grafični predstavitvi dokumenta XML so vsebovani tako enostavni, kot tudi kompleksni elementi. V tabeli 3.1 je predstavljen pomen posameznih grafičnih simbolov.

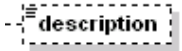
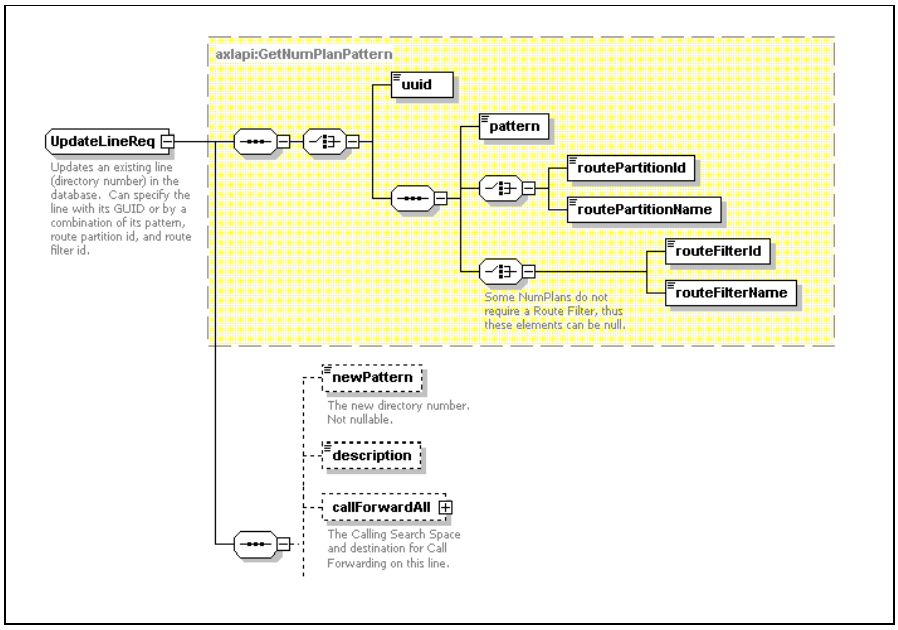
	Ime elementa	**Opis**
	Kompleksni element	Lahko vsebuje otroke in atribute
	Enostaven element	Nima otrok, ima samo atribute
	Opcijski element	Vsak tip elementa je izbirni
	Zaporedje	Vsi otroci si morajo slediti zaporedno
	Izbira	Samo en otrok je lahko izbran

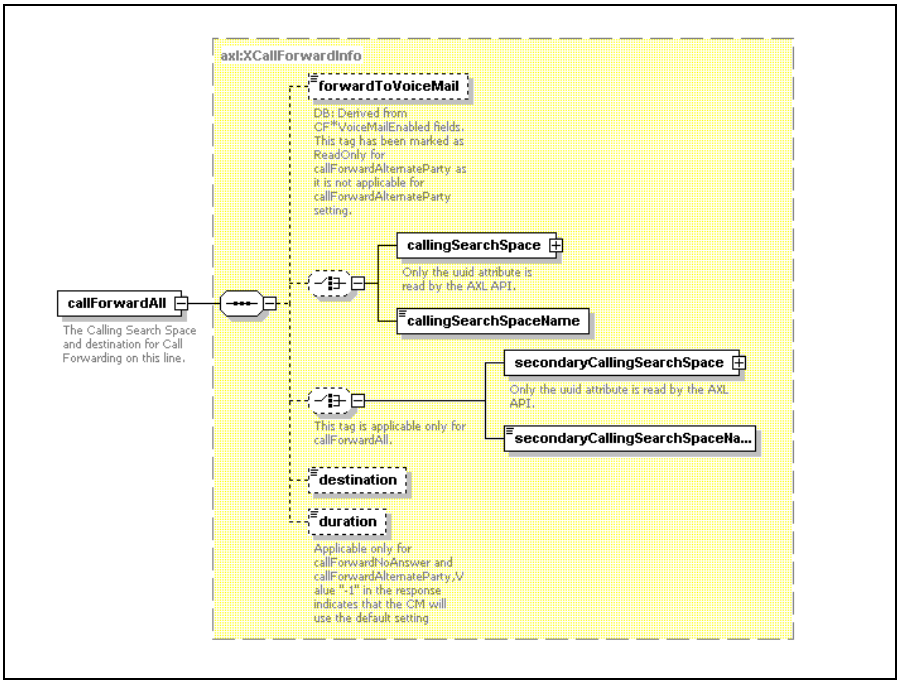
Tabela 3.1: Tabela različnih elementov v grafični predstavitvi dokumenta XML

Cilj diplomske naloge je izvesti preusmeritev klica z uporabo SOAP HTTP tehnologije, zato je na naslednjih dveh slikah predstavljen grafični opis dokumenta XML za posodobitev linije. Na podoben način je definiranih še nešteto drugih dokumentov XML, vsak pa izvede točno določeno akcijo. Z dokumentom XML, ki je grafično predstavljen na slikah 3.3 in 3.4, lahko torej izvedemo preusmeritev klica na točno določeni telefonski številki.



Slika 3.3: Grafični prikaz XML za posodobitev linije

Na sliki 3.3 manjka večina parametrov, ki se jih da nastavljati z zahtevo *UpdateLineReq*, saj bi v tem primeru slika obsegala skoraj dve strani in za samo razumevanje delovanja programa niso pomembni. Bistvena dva parametra, to sta *pattern* in *callForwardAll,* pa sta predstavljena. Z atributom *pattern* enolično določimo telefonsko številko, s parametrom *callForwardAll* pa tej telefonski številki nastavimo preusmeritev klicev na želeno telefonsko številko.



Slika 3.4: Grafični prikaz XML-ja za preusmeritev vseh klicev

Element *CallForwardAll* je kompleksen in zato lahko vsebuje otroke in dodatne atribute (glej tabelo 3.1). Na sliki 3.4 je prikazana razširitev prejšnjega atributa z izbirnimi parametri. Če ponovno pogledamo v tabelo 3.1 lahko opazimo, da lahko za potrebe našega programa dejansko nastavimo zgolj parameter *destination*, ostale pa ignoriramo.

Glede na predstavljeno grafično definicijo dokumenta XML (sliki 3.3 in 3.4) lahko sedaj podamo primer XML SOAP sporočila, ki ga moramo poslati na vmesnik AXL, če želimo na primer vse klice na številko 911 preusmeriti na številko 113.

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
<SOAP-ENV:Header />
        <SOAP-ENV:Body>
                <axl:updateLine xmlns:axl="http://www.cisco.com/AXL/API/1.0">
                        <UpdateLineReq>
                                <pattern>911</pattern>
                                <callForwardAll>
                                        <destination>113< /destination>
                                </callForwardAll>
                        </UpdateLineReq>
                </axl:updateLine>
        </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

V zgornjem primeru lahko bralec iz celotne strukture vidi, kako enostavno je sestaviti sporočilo SOAP, ki ga bo vmesnik razumel. Začetek in konec sporočila, ki sta predstavljena z ovojnico, glavo in telesom, sta vedno enaka. Spreminja se le vsebina XML v telesu sporočila. Vse kar mora torej razvijalec narediti je, da na spletni strani proizvajalca poišče ustrezen dokument XML, ki ga nato vstavi v telo SOAP-sporočila in pošlje na vmesnik. Na podoben način je nastala tudi aplikacija, ki je podrobno predstavljena v naslednjem poglavju.

4 OPIS REŠITVE

Programska rešitev, ki je predstavljena v tem poglavju, je napisana v programskem jeziku Java. Java se v zadnjem času zelo pogosto uporablja pri pisanju spletnih aplikacij, saj vsebuje veliko vgrajenih knjižnic, ki programerju olajšajo delo in posledično tudi skrajšajo razvoj nove storitve. V našem primeru so najbolj uporabne knjižnice za delo z dokumenti XML in SOAP. Zelo poenostavljena je tudi implementacija SSL, ki je potrebna pri pošiljanju zahtev na vmesnik AXL, saj je le-ta dosegljiv preko protokola HTTPS. Zelo hitro pa lahko z uporabo knjižnic implementiramo branje iz aktivnega imenika preko protokola LDAP.

Slika 4.1: Rešitev problema v sliki

Na zgornji sliki je grafično predstavljeno delovanje aplikacije, kar lahko z besedami povzamemo v dveh točkah:

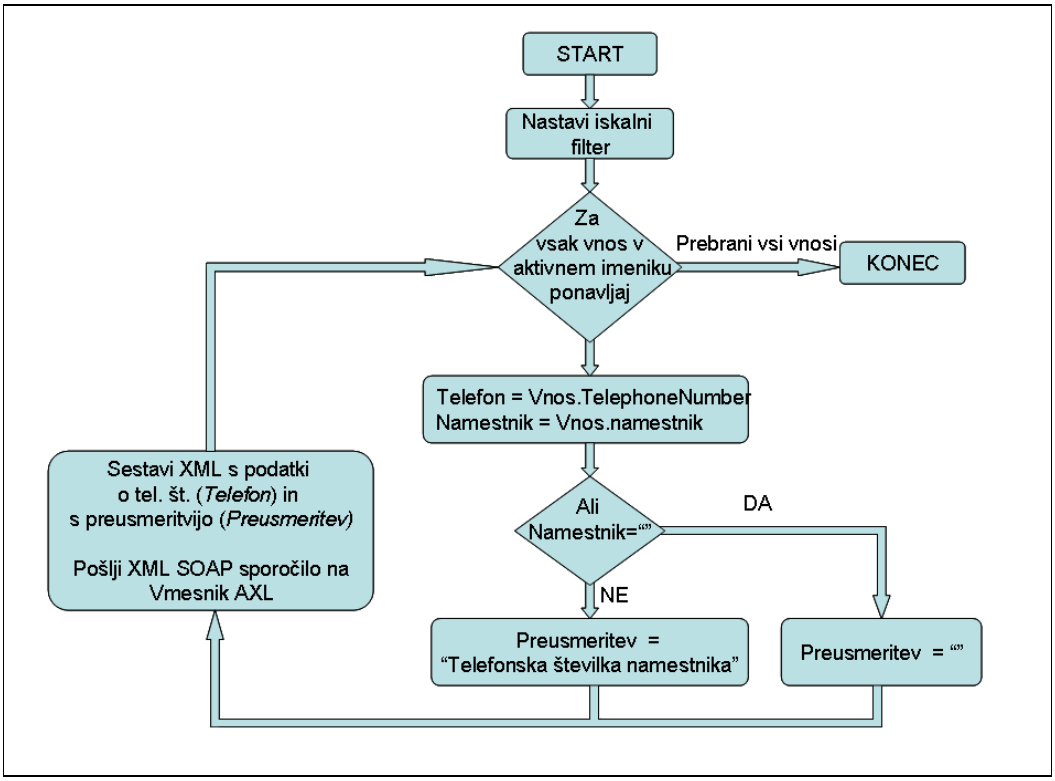
- V aktivnem imeniku se za vsako osebo posebej preveri, ali je potrebno klice na njeno telefonsko številko preusmeriti na drugo osebo, in če da, potem se v imeniku poišče telefonsko številko te druge osebe.
- Sestaviti je potrebno pravilen dokument XML, ga zapakirati v sporočilo SOAP in ga s pomočjo HTTP POST metode poslati nadzorniku komuniciranja. V dokumentu XML sta najbolj pomembna dva podatka, in sicer telefonska številka osebe ter telefonska številka osebe, na katero je potrebno preusmeriti vse klice.

Pri razvoju aplikacije, ki pošilja zahteve na vmesnik AXL, si razvijalec lahko pomaga z nazorno dokumentacija v [14], kjer lahko najde tudi programsko kodo za različne programske jezike. Na spletu obstaja tudi nekaj primerov za delo z aktivnimi imeniki v programskem jeziku Java, največ splošnih informacij pa je dostopnih na [15].

V nadaljevanju tega poglavja bo najprej predstavljen diagram poteka. Bralec si bo tako ob sliki lažje predstavljal posamezne funkcije in delovanje programa. Na koncu si bomo ogledali še najbolj zanimive delčke programske kode, ki je skopirana iz delujočega programa. Celotna programska koda je diplomski nalogi priložena na zgoščenki, tako da si jo bralec lahko tudi ogleda in požene.

4.1 Diagram poteka

Diagram poteka je diagram za prikaz možnih poti podatkov skozi sistem. V računalništvu ga uporabljamo za opis poteka operacij določenega računalniškega programa. Diagram poteka, ki je prikazan na naslednji sliki, prikazuje natančno zaporedje operacij, ki jih program pri obdelavi podatkov izvede. Različni grafični simboli predstavljajo vnos in izpis podatkov, odločitve, razvejitve, podprograme.



Slika 4.2: Diagram poteka

Diagram poteka je opisan zelo splošno in predstavlja časovni potek operacij, ki jih aplikacija v svojem življenjskem času izvede. Ob zagonu programa se najprej nastavi iskalni filter, ki določa unikatno ime poddrevesa v aktivnem imenika, kjer se nahajajo primarne osebe. V zanki nato za vsako primarno osebo program preveri ali je njen namestnik nastavljen. V primeru, da je vrednost prirejene spremenljivke *namestnik* enaka praznemu nizu znakov, se nastavi spremenljivko *preusmeritev* s praznim nizom, kar določa, da je primarna oseba v podjetju in da je potrebno morebitno aktivno preusmeritev preklicati. V nasprotnem primeru, se v aktivnem imeniku poišče telefonsko številko namestnika in se jo priredi spremenljivki *preusmeritev*. Po zaključku omenjenih operacij imamo v spremenljivki *telefon* shranjeno telefonsko številko primarne osebe, v spremenljivki *preusmeritev* pa telefonsko številko sekundarne osebe ali prazen niz. Obe spremenljivki podamo kot parameter metodi, ki sestavi sporočilo SOAP. Celotno sestavljeno sporočilo pa nato prejme metoda, ki poskrbi za pošiljanje sporočila na vmesnik, ter čaka na sporočilo o uspešni oziroma neuspešni izvedbi zahtevane akcije. Program se zaključi, ko je obravnaval vse vnose v poddrevesu.

V nadaljevanju bomo nekatere pomembnejše funkcije, ki so v tem diagramu zgolj omenjene, predstavili podrobneje. Bralec pa si bo lahko ogledal njihovo strukturo tudi v programskem jeziku Java.

4.2 Zanimivi deli kode v programskem jeziku Java

Program, ki pošilja zahteve na vmesnik AXL, smo napisali v programskem jeziku Java. Za ta programski jezik smo se odločili med drugim tudi zato, ker nadzornik komuniciranja teče na Tomcat strežniku, na katerem tečejo spletne strani, napisane v Javi. V primeru, da bi bilo kasneje potrebno določene stvari uporabiti pri morebitni izdelavi vmesnika na spletni strani, bi bile potrebne minimalne spremembe obstoječega programa.

Program je razdeljen na tri dele, na dve knjižnici in glavni program. Prva knjižnica, imenovana LDAP, skrbi za povezovanje in branje iz aktivnega imenika, druga, poimenovana CM, skrbi za sestavljanje in pošiljanje ustreznih sporočil SOAP na vmesnik AXL. Naloga glavnega programa je ustrezno klicanje konstruktorjev in metod iz prej omenjenih knjižnic. V nadaljevanju tega poglavja bomo na kratko poskušali predstaviti delovanje teh knjižnic. Kot zanimiv dodatek bo pri nekaterih metodah navedena tudi programska koda.

4.2.1 Knjižnica LDAP

Knjižnica ima definiran en razred z imenom LDAPConnect, kjer so definirane metode in funkcije, ki jih bomo v naslednjih nekaj odstavkih podrobneje predstavili.

Konstruktor *LDAPconnect* poskrbi za pravilno nastavljanje povezovalnih parametrov, kot na primer uporabniško ime, geslo, verodostojnost. Za preverjanje verodostojnosti se uporabljajo trije načini:

- Anonymous
- Simple
- SASL (Simple Authentication and Security Layer)

Na koncu konstruktorja se v globalno spremenljivko *ctx* shrani povezava na aktivni imenik. V metodah razreda se nato stalno(iskanje, morebitno pisanje in brisanje) sklicujemo na to dodeljeno spremenljivko.

```
public LDAPconnect(String username,String password,String auth,String server) throws
NamingException
{
      env = new Hashtable<String,String>();
      env.put("com.sun.jndi.ldap.connect.pool", "true");
      env.put("java.naming.security.principal", username);
      env.put("java.naming.security.credentials", password);

    #SASL, Anonymous, Simple
      env.put("com.sun.jndi.ldap.connect.pool.authentication", auth);
      env.put(Context.PROVIDER_URL, server);
      env.put(Context.INITIAL_CONTEXT_FACTORY, "com.sun.jndi.ldap.LdapCtxFactory");
      ctx = new InitialDirContext(env);
}
```

V razredu *LDAPconnect* sta definirani še dve metodi *search*, ki ju kličemo z različnimi parametri. S prvo metodo iščemo vnose v poddrevesu aktivnega imenika, ki je določen z nizom *domain*. Metoda vrne vse vnose, kjer ima iskani atribut točno določeno vrednost. To metodo uporabljamo pri iskanju telefonske številke namestnika, saj poznamo dva atributa (ime, priimek). Podjetje zagotavlja, da do podvojenih atributov (ime, priimek) ne more priti, tako da s to metodo v trenutku dobimo iskani vnos. Od tu naprej je naloga glavnega programa, da iz vnosa prebere telefonsko številko, kar pa je trivialno.

```
public NamingEnumeration search(String domain, Attributes matchAttrs)
{
        Try
        {
                NamingEnumeration answer = ctx.search(domain, matchAttrs);
                return answer;
        } catch (NamingException e)
        {
                System.out.println("Neuspeh pri branju!!!");
                e.printStackTrace();
                return null;
        }
}
```

Z drugo metodo *search* iščemo vnose prav tako v poddrevesu aktivnega imenika, ki je določen z nizom *domain*. Ta metoda se od prve razlikuje v tem, da tu preverjamo vrednost relativnega unikatnega imena in nas atributi v vnosu ne zanimajo. V spremenljivki filter določimo iskano vrednost. Določimo lahko točno določeno vrednost, na primer filter=Studenti, ali bolj splošno, kot je na primer filter=*, kar pomeni da želimo dobiti vse vnose v poddrevesu. Filtre lahko kombiniramo z ALI (|) ter IN (&). Več o iskanju po aktivnem imeniku s pomočjo filtrov si lahko bralec prebere v [4] ali v RFC2254 [16].

```
public NamingEnumeration<SearchResult> search(String domain, String
filter,SearchControls ctls)
{
        try {
                NamingEnumeration<SearchResult> answer = ctx.search(domain, filter, ctls);
                return answer;
        } catch (NamingException e)
        {
                System.out.println("Neuspeh pri branju!!!");
                e.printStackTrace();
                return null;
        }
}
```

Z spremenljivko *domain* v obeh metodah določimo iskanje po poddrevesu z globino ena. Če želimo iskanje razširiti še v globino je potrebno za to poskrbeti v glavnem programu, ali pa spremeniti omenjeni metodi, tako da bi znali delati tudi to. To lahko dosežemo z dodajanjem zank ali pa rekurzivno.

4.2.2 Knjižnica CM

Podobno kot pri komunikaciji z aktivnim imenikom je tudi za komunikacijo z vmesnikom AXL napisana posebna knjižnica, imenovana CM, v kateri je razred *AxlToolkit,* kjer so definirane pomembne metode, ki so opisane v nadaljevanju.

Konstruktor *AxlToolkit* poskrbi za nastavitev parametrov, kot so strežnik, uporabniško ime, geslo in priklop. Ob koncu konstruktorja se kliče še metoda *init*, ki poskrbi za registracijo odjemalca s strežnikovim digitalnim potrdilom in obratno, saj je vmesnik AXL dosegljiv preko varne HTTPS povezave.

```
public AxlToolkit(String username,String password,String host,String port)
{
        this.host=host;
        this.username=username;
        this.password=password;
        this.port=port;
        this.init();
}
```

Prva pomembna metoda razreda se imenuje *createCallForwardAllMessage*, ki glede na dva vhodna parametra sestavi sporočilo SOAP. Prvi vhodni parameter *number* določa telefonsko številko, drugi - *destination* pa določa številko, na katero želimo prevezati vse klice. Če je niz v parametru *destination* prazen, pomeni da se morebitna aktivna preusmeritev prekliče. Za sestavljanje SOAP-sporočila uporablja metoda vgrajene knjižnice programskega jezika Java. Alternativno bi lahko sestavili SOAP-sporočilo tudi kot niz znakov. Kar se funkcionalnosti metode tiče, je popolnoma vseeno, kako sestavimo sporočilo. Zaradi lepšega programiranja pa smo se rajši odločili za uporabo vgrajenih knjižnic.

```
public SOAPMessage cretaeCallForwardAllMessage(String number, String destination)
throws Exception
 {
     MessageFactory mf = MessageFactory.newInstance();
     SOAPMessage soapMessage = mf.createMessage();
     SOAPEnvelope envelope = soapMessage.getSOAPPart().getEnvelope();
     SOAPBody bdy = envelope.getBody();

     SOAPBodyElement bodyElement =
                    bdy.addBodyElement(envelope.createName("axl:updateLine"));
     bodyElement.addNamespaceDeclaration("axl", "http://www.cisco.com/AXL/API/1.0");
     bodyElement.addAttribute(envelope.createName("sequence"),
                    String.valueOf(System.currentTimeMillis()));
     SOAPElement el = bodyElement.addChildElement("UpdateLineReq");
     el.addChildElement("pattern").addTextNode(number);
     el.addChildElement("callForwardAll").addChildElement("destination")
                    .addTextNode( destination);

     return soapMessage;
   }
```

Metoda *sendMessage* pošilja SOAP sporočila. Kot argument prejme SOAP sporočilo, ki ga pošlje nadzorniku komuniciranja na vmesnik AXL, od katerega nato prejme odgovor. Odgovor zapakiran v sporočilu SOAP je lahko:

- Sporočilo je bilo uspešno poslano. Ob določenih zahtevah se vrnejo tudi podatki.
- Pri pošiljanju sporočila je prišlo do napake. Kot dodaten opis lahko sporočilo SOAP vsebuje še vzrok napake.

```
public SOAPMessage sendMessage(SOAPMessage requestMessage) throws Exception
{
      SOAPMessage reply = null;
      try {
         reply = con.call(requestMessage, getUrlEndpoint());

         if (reply != null) {
           //Preverimo ce vsebuje sporocilo  soap fault
           SOAPPart replySP = reply.getSOAPPart();
           SOAPEnvelope replySE = replySP.getEnvelope();
           SOAPBody replySB = replySE.getBody();

         if (replySB.hasFault()) {
               System.out.println("ERROR: " + replySB.getFault().getFaultString());
           }
         else {
                  if (debug)
                  System.out.println("Prejeli smo pozitivni odgovor!");
                  }
         }
         else {
                  System.out.println("Nismo prejeli nobenega odgovora!");
         }
      }  catch (Exception e) {
         e.printStackTrace();
         throw e;
      }
      return reply;
}
```

4.2.3 Glavni program

Glavni program je v svojem razredu. V programski kodi ni nobenih posebnosti, saj je večinoma vse potrebno za delovanje implementirano v prej opisanih knjižnicah. Glavni program v *main* metodi poskrbi za inicializacijo aktivnega imenika. V programskih zankah poišče ustrezne podatke iz imenika, poskrbi za sestavljanje ustreznega sporočila SOAP in ga pošlje z uporabo metode HTTP POST na vmesnik AXL.

Program se zaključi, ko prebere vse vnose v poddrevesu, ki smo jih ob zagonu programa z uporabo filtra nastavili.

5 MOŽNOSTI ZA DELOVANJE APLIKACIJE V REALNEM ČASU

V prejšnjem poglavju je podrobneje razloženo in opisano delovanje programa, ki se izvede enkrat dnevno in z uporabo SOAP HTTP tehnologije pošilja zahteve nadzorniku komuniciranja. V našem primeru smo nastavljali preusmeritev telefonskih klicev enkrat dnevno, podobno funkcionalnost pa bi lahko dosegli tudi v realnem času.

V tem poglavju si bomo ogledali in na kratko predstavili dva možna načina, ki omogočata takšno storitev. Prvi način je z uporabo programskih knjižnic, ki omogočajo integracijo računalnika in telefonije, in je z vidika razvijalca veliko bolj zahteven. Drugi, pa je uporaba namenskega razvojnega okolja CUAE, ki ga je podjetje Cisco razvilo z namenom hitrejšega in enostavnejšega razvoja novih aplikacij in storitev v IP-telefoniji. Podrobnejše informacije o obeh tehnologijah si bralec lahko poišče na spletni strani [12].

5.1 Uporaba programskih knjižnic

Z uporabo programskih knjižnic TAPI, napisanimi za programski jezik C ali JTAPI, ki so namenjene za programiranje v Javi, lahko spišemo aplikacijo, ki bo registrirala posamezne klicne dogodke na nadzornik komuniciranja in v primeru sproženega dogodka izvedla ustrezno akcijo. Aplikacija na ta način pridobi popolni nadzor nad napravami in lahko vzpostavlja, sprejema, prevzema in končuje klice.

V našem primeru bi lahko napisali aplikacijo s pomočjo enega izmed obeh programskih jezikov in z ustreznimi ukazi registrirali dogodek, ki bi ob vsakem aktivnem klicu sprožil nov dogodek, aplikacija pa bi nato poskrbela za iskanje po aktivnem imeniku ter v primeru potrebe po preusmeritvi zahtevala preusmeritev klica na drugo telefonsko številko.

Pozitivna stran te rešitve je, da se preusmeritev klica izvede v realnem času. Slabost pa je ta, da se ob vsakem aktivnem klicu izvaja določen algoritem, ki se lahko po nepotrebnem izvede tudi večkrat dnevno.

Zgornji primer bi lahko izboljšali tako, da že preusmerjenih številk ne bi ponovno preverjali, vendar je praviloma preusmerjenih številk zelo malo, tako da s tem verjetno ne bi pridobili prav veliko časa.

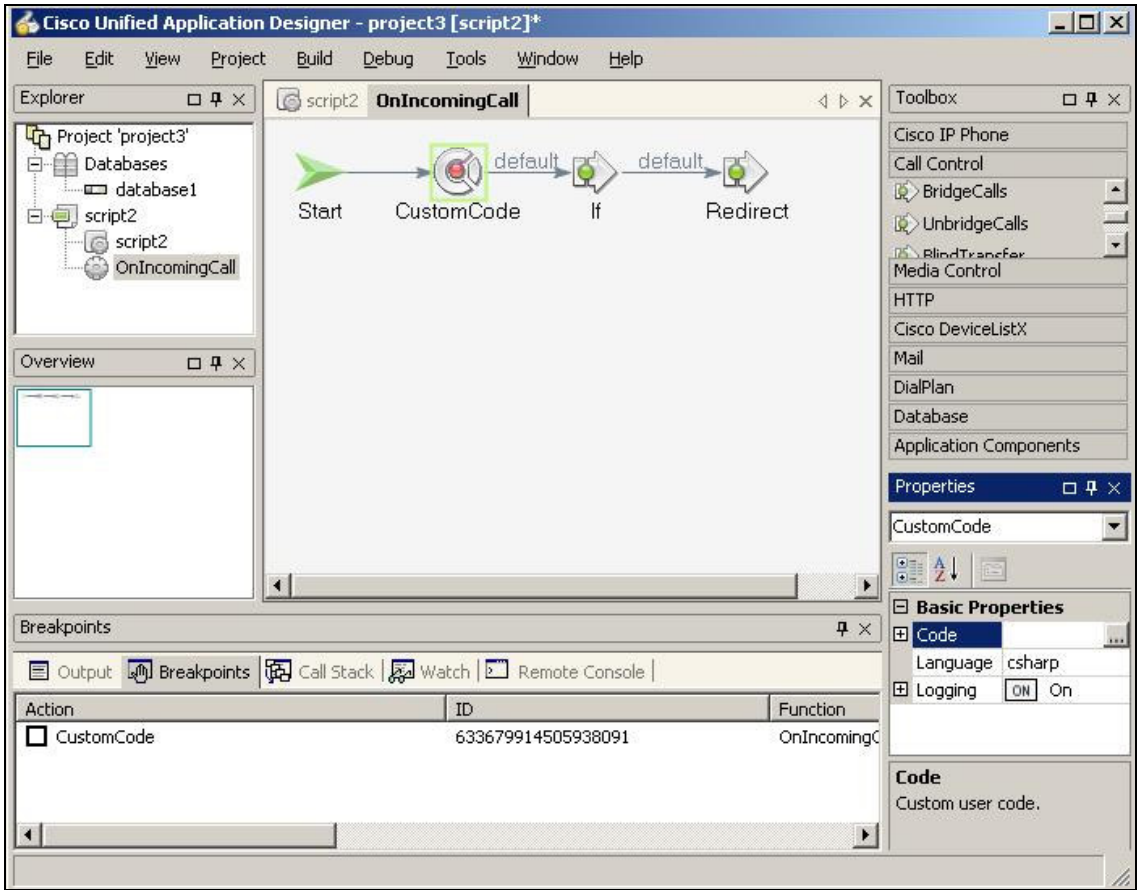
5.2 Razvojno okolje podjetja Cisco

Cisco Unified Application Environment (CUAE) je zbirka storitev in orodij za razvijalce, ki omogočajo zelo hiter razvoj, zanesljivo izvajanje in avtomatizacijo upravljanja komunikacijskih aplikacij. Okolje je namenjeno predvsem razvijalcem, ki niso vešči programiranja. Trenutno ima to okolje še veliko programskih hroščev, zato ni priporočljivo za uporabo v produkcijske namene.

Pozitivna stran razvojnega okolja je ta, da omogoča zelo hiter razvoj želenih funkcionalnosti. Razvijalec dejansko z miško in klikanjem doseže, kar želi, in mu ni potrebno vedeti, kaj se skriva za vsem skupaj. Ob zaključku se vsa koda zapiše v neberljivo binarno datoteko.

Negativna stran pa je, da moramo vse klice »speljati« skozi ta naš samostojni strežnik, ki se lahko zaradi preobremenjenosti tudi preneha odzivati, kar ima lahko za posledico nedelovanje sistema IP-telefonije.

Na sliki 5.1 je primer razvojnega orodja, v katerem je približno narisan potek razvoja programa za preusmeritev klicev. Vse vhodne klice preusmerimo v naš program, kjer najprej v CustomCode definiramo kodo za iskanje po aktivnem imeniku, v if delu preverimo pogoj za preusmeritev, v zadnjem delu pa izvedemo še preusmeritev. Na koncu sledi zaključek klica.



Slika 5.1: Primer razvojnega orodja v CUAE

6 OBSTOJEČE REŠITVE

Sodobni razvijalec, ki želi napisati določeno aplikacijo, najprej preveri, ali je nekdo kaj podobnega že napisal. Dejstvo je, da na spletu lahko najdemo veliko koristnih podatkov ali celo delčke programske kode, ki zna narediti natanko tisto, kar mi zahtevamo od naše aplikacije.

Na spletu obstaja nekaj primerov za delo z aktivnimi imeniki, napisanih v programskem jeziku Java, največ splošnih informacij za delo z njimi pa je na voljo na spletni strani [15], ki nam lahko pri razvoju osnovnih funkcionalnosti zelo pomaga. Pri razvoju aplikacij za oddaljeno administracijo nadzornika komuniciranja obstaja zelo dobra dokumentacija na [14], kjer najdemo tudi splošno programsko kodo za sestavljanje in pošiljanje dokumentov XML na strežnik. Verjetno obstaja veliko podobnih ali bolj razvitih aplikacij od te, ki je predstavljena v tem delu, vendar zaradi skrivanja programske kode pred konkurenco žal ni dostopna ostalim razvijalcem, tako da je zelo težko najti obstoječe rešitve. Razvijalec, ki pozna osnove tehnologij SOAP in XML, lahko s pomočjo dokumentacije, dostopne na spletni strani proizvajalca Cisco, dokaj hitro razvije svojo aplikacijo. V nadaljevanju tega poglavja bomo predstavili eno izmed obstoječih rešitev, ki teče kot servis za IP-telefone in ponuja storitev za preusmeritev telefonskih klicev na drugo številko.

6.1 Nastavitev preusmeritve preko vmesnika na telefonu

IP-telefoni proizvajalca Cisco ponujajo paleto različnih možnosti za razvijalce. Telefon se lahko poveže na določen spletni naslov in od tam prebere podatke, ki so v obliki dokumenta XML. Glede na prebrano vsebino telefon ustrezno prikaže podatke na zaslonu. Telefon lahko prikazuje tekst, menije, slike itn. Vrste dokumentov XML, ki jih pozna IP-telefon in njihov način prikazovanja, so nazorno opisani v [17].

Na vsakem telefonu lahko ročno nastavimo preusmeritev vseh klicev na ustrezno telefonsko številko s pritiskom na posebno funkcijsko tipko *Prevezi Vse*. Ob pritisku na to tipko se telefon poveže na spletni naslov, ki je definiran v trenutno na ekranu prikazanem dokumentu XML. Na spletnem naslovu teče storitev, ki zna narediti isto stvar kot naša aplikacija, in je predstavljena v tem delu.

Po raziskovanju programske kode, ki teče na strežniku in je napisana v programskem jeziku Java, pridemo do ugotovitve, da se preusmeritev klicev izvaja na zelo podoben način, kot to počne naš program. Aplikacija teče na spletnem strežniku in sprejema zahteve preko metode HTTP GET. Ob zahtevi za preusmeritev klicev se najprej preverijo uporabniške pravice, nato pa se pošlje sporočilo SOAP na vmesnik AXL s podatki o preusmeritvi, ki jih je uporabnik vnesel v obrazec na telefonu. Obrazec predstavlja v bistvu XML dokument, ki s pritiskom na tipko *Potrdi* pošlje podatke o preusmeritvi na spletni strežnik.

Ta rešitev je zelo podobna programu opisanemu v tej diplomski nalogi, razlika je le v tem, od kod aplikacija dobi informacije o nastavljanju preusmeritev.

7 SKLEP

S programom, predstavljenim v tej diplomski nalogi, je podjetje pridobilo funkcionalnost, ki si jo je po uvedbi IP-telefonije želelo. Program se izvede enkrat dnevno, se sprehodi čez vnose aktivnega imenika in ustrezno, glede na vrednost atributa *namestnik* v vnosu, preusmeri vse telefonske klice iz določene številke na neko tretjo telefonsko številko, ki je v lasti osebe, zapisane v omenjenem atributu.

Med samim razvojem ali v času, ko neka aplikacija že deluje v produkcijskem okolju, se porodijo razne ideje o izboljšanju in nadgradnji. V nadaljevanju si bomo zato ogledali še nekaj predlogov za nadaljnje delo in razvoj:

- predlog izboljšav aktivnega imenika v podjetju
- možne izboljšave oziroma nadgradnja programa
- ali je smiselno aplikacijo poganjati v realnem času

7.1 Predlog izboljšav aktivnega imenika v podjetju

Največji problem v tem podjetju je organizacija aktivnega imenika, predvsem definiranje osebe, na katero je potrebno narediti preusmeritev vseh klicev. Spomnimo se, da je vnos za posamezno osebo videti takole:

Ime	Priimek	Telefonska številka	Namestnik (ime in priimek osebe, na katero naredimo preusmeritev)

Tabela 7.1: Definicija vnosa v aktivnem imeniku podjetja

Dejstvo je, da lahko v podjetju obstajata dva Janeza Novak, zaradi česar program ne deluje več pravilno. V podjetju sicer zagotavljajo, da se to ne more zgoditi. Vsakemu pa je jasno, da se lahko že jutri v podjetju zaposli človek z istim imenom in priimkom, kot ga ima že nekdo od zaposlenih, in takrat bo prišlo do nepravilnega delovanja aplikacije.

Podjetju bi lahko predlagali dve možni rešitvi. Namesto zadnjega polja naj:

- uporabijo kar številko, na katero je potrebno narediti preusmeritev, ali
- v zadnje polje vnesejo edinstveni atribut osebe na, katero je potrebno narediti preusmeritev.

7.2 Možne izboljšave oziroma nadgradnja programa

Program, opisan v tej diplomski nalogi, predstavlja osnovo, ki jo lahko na dokaj enostaven način razširimo z dodajanjem novih metod, predvsem v knjižnici za delo z vmesnikom AXL. V dokumentaciji, ki jo najdemo na [12], so na podoben način, kot je to predstavljeno na slikah v podpoglavju 3.3, definirani dokumenti XML še za veliko drugih ukazov. V primeru, da želimo določen postopek avtomatizirati, moramo najprej poiskati definicijo dokumenta XML, ki ga je potrebno sestaviti, da se bo izvedla želena akcija (dodajanje, brisanje, spreminjanje objekta), nato pa dodamo novo metodo, ki bo sestavila želeni dokument XML, v našo

knjižnico. Ko je nova metoda napisana, je naša naloga zgolj klicanje te metode iz glavnega programa.

V nadaljevanju si bomo ogledali še možnost avtomatske gradnje programske kode iz datoteke WSDL, ki lahko razvijalcu prihrani ogromno časa in energije.

7.2.1 Avtomatska gradnja programske kode iz datoteke WSDL

Na nadzorniku komuniciranja obstaja dodatek, kjer najdemo datoteke WSDL, za različne verzije nadzornikov komuniciranja. WSDL je jezik za opisovanje spletne storitve. V datoteki so opisane vse metode, ki jih vmesnik AXL omogoča. Pri vsaki verziji prihaja do določenih sprememb v shemi, zato so te datoteke dodane kot dodatek in jih ni mogoče potegniti direktno iz strežnika. Razvijalec jih lahko po želji vključi v svoj program, s tem pa mu je olajšan marsikateri problem.

Velika prednost tega načina razvijanja aplikacij je v tem, da razvijalcu ni potrebno vedeti, kako se sestavi sporočilo SOAP in v kakšni obliki ga je potrebno poslati na vmesnik. Z uporabo namenskega programa (primer Axis), se iz datoteke WSDL avtomatsko zgradi programska koda, ki jo nato razvijalec pri pisanju aplikacije uporabi. Za branje, pisanje, spreminjanje določenih objektov mora tako razvijalec samo klicati ustrezno avtomatsko zgrajeno metodo s pravilnimi atributi.

Tovrstni način omogoča veliko možnosti pri izboljšanju našega programa. Žal pri avtomatsko zgrajeni kodi prihaja do določene programske zanke, ki je razlog za veliko porabo pomnilnika in koda zaenkrat še ni uporabna. Verjetno bi se splačalo poglobiti v to težavo in ugotoviti, v čem je problem ter ga odpraviti, saj bi v tem primeru aplikacija postala univerzalna in bi z njo lahko zelo dobro upravljali in nadzirali sistem IP-telefonij in to z zelo malo truda in znanja. Predvsem bi v tem primeru razvijalec prihranil ogromno časa, saj bi mu bilo prihranjeno brskanje po dokumentaciji dokumentov XML na spletni strani proizvajalca.

7.3 Ali je smiselno aplikacijo poganjati v realnem času?

Glede na vse, kar smo do sedaj predstavili, lahko ugotovimo, da nima smisla izvajati preusmeritve klicev v realnem času. Preusmeritev klica se v podjetju največkrat nastavi, ko gre nekdo na dopust in je tako odsoten vsaj en dan. Če oseba le za kratek čas odide iz svojega delovnega mesta, podjetje te spremembe ne bo vnašalo v aktivni imenik, ampak bo, če bo to potrebno, ročno nastavila preusmeritev na samem telefonu.

Zakaj naročnik potrebuje program, ki je predstavljen v tem diplomskem delu, če že obstaja storitev na telefonu, ki omogoča ročno nastavljanje preusmeritve telefonskih klicev?
V današnjem času je zadovoljstvo strank za podjetje izrednega pomena. Če želimo zmanjšati človeški faktor zmotljivosti (pozabljivost, neorganiziranost itn), je potrebno uvesti čim več avtomatike. Ta program je posebnega pomena v podjetjih z velikim številom zaposlenih.

Odločitev za predstavljeno rešitev je tako še vedno najbolj smiselna, če pa se bo kasneje v praksi izkazalo, da bi bilo potrebno vse to izvajati v realnem času, se bomo verjetno odločil za JTAPI. Verjetno pa ima razvojno okolje CUAE še dobre možnosti, da v nekaterih aplikacijah

nadomesti JTAPI, predvsem zaradi dejstva, da omogoča zelo hiter razvoj želenih funkcionalnosti.

8 PRILOGE

CD s programsko kodo

9 LITERATURA

[1] Andrej Krajnc, Gordana Budimir: Nekateri vidiki razvoja tehnologije XML, vir:http://home.izum.si/cobiss/cobiss_obvestila/2001_4/Html/clanek_03.html, dostopno januarja 2009

[2] Vidmar Tone, Informacijsko-komunikacijski sistem, Pasadena, Ljubljana 2002

[3] Lightweight Directory Access Protocol, vir: http://en.wikipedia.org/wiki/Lightweight_Directory_Access_Protocol, dostopno januarja 2009

[4] Jaka Sodnik, Preprost protokol za dostop do imenika LDAP,diplomsko delo na FE, Ljubljana 2002

[5] Cisco Voice over IP, Volume 1, Student Guide¸ 17.3.2008

[6] Voice over Internet Protocol, vir: http://en.wikipedia.org/wiki/VoIP , dostopno januarja 2009

[7] vir: www.cisco.com, dostopno januarja 2009

[8] Media Gateway Control Protocol, vir:http://en.wikipedia.org/wiki/Media_Gateway_Control_Protocol, dostopno januarja 2009

[9] H.323, vir:http://en.wikipedia.org/wiki/H.323, dostopno januarja 2009

[10] SIP, vir:http://en.wikipedia.org/wiki/SIP, dostopno januarja 2009

[11] Skinny Call Control Protocol, vir:http://en.wikipedia.org/wiki/Skinny_Client_Control_Protocol, dostopno januarja 2009

[12] Cisco Developer Community, vir:http://developer.cisco.com, dostopno januarja 2009

[13] Jan Bervar, PKI Hell, vir: http://blogs.nil.com/blog/author/jan-bervar/, dostopno februar 2009

[14] AXL Configuration Programming Vir: http://www.cisco.com/en/US/docs/voice_ip_comm/cucm/devguide/6_0_1/ccmdvCh1.html, dostopno januarja 2009

[15] Tips for LDAP Users,vir: http://java.sun.com/products/jndi/tutorial/ldap/index.html

[16] T. Howes, RFC2254 - The String Representation of LDAP Search Filters, vir:http://www.faqs.org/rfcs/rfc2254.html, dostopno januarja 2009

[17] Darric Deel, Mark Nelson, Anne Smith, Developing Cisco IP Phone Services, Cisco Press, Indianapolis, Februar 2002