UNIVERZITET U NOVOM SADU
FAKULTET TEHNIČKIH NAUKA
Odsek za elektrotehniku i računarstvo
Institut za računarstvo i automatiku
Katedra za računarsku tehniku i računarske komunikacije

Izdvajanje IVR komponente virtuelnog centra za obradu poziva u poseban IVR server

- Diplomski rad -

Mentor:
Prof. Dr. Miroslav Popović

Kandidat:
Đorđe Miljković, E8409

Novi Sad, Jul 2007.

UNIVERZITET U NOVOM SADU
FAKULTET TEHNIČKIH NAUKA
KLJUČNA DOKUMENTACIJSKA INFORMACIJA

Redni broj: RBR	
Identifikacioni broj: IBR	
Tip dokumentacije: TD	*Monografski rad*
Tip zapisa: TZ	*Štampa*
Autor: AU	*Đorđe Miljković*
Mentor/Komentor MN	*prof. Dr Miroslav Popović*
Naslov rada: NR	*Izdvajanje IVR komponente virtuelnog centra za obradu poziva u poseban IVR server*
Jezik publikacije: JP	*srpski*
Jezik izvoda: JI	*sprski*
Zemlja publikovanja: ZP	*Srbija*
Uže geografsko područje: UGP	*Vojvodina*
Godina: GO	*2007*
Izdavač: IZ	*Fakultet Tehničkih Nauka*
Mesto i adresa: MA	*21000 Novi Sad, Trg Dositeja Obradovića 5*

Fizički opis rada:
FO

Naučna oblast: *Elektrotehnika*
NO

Naučna disciplina: *Računarska tehnika i računarske komunikacije*
ND

Predmetna odrednica/ključne reči: *Računarstvo, Virtualna obrada poziva, IVR komponenta,*

PO
UDK

Čuva se: *U biblioteci Fakulteta Tehničkih Nauka*
ČU

Važna napomena:
VN

Izvod: *U radu je izvršeno izdvajanje IVR komponente iz virtuelnog centra za obradu poziva u poseban IVR server*

IA

Datum prihvatanja teme
Od strane NN veća:
DP

Datum odbrane:
DO

Članovi komisije:
KO

Predsednik:
Član:
Član:

SKRAĆENICE

ABA - *AB Analysis*, AB analiza
ACD - *Automatic Call Distribution*, Automatska distribucija poziva
AG - *Agent*, Operater
API - *Aplication Programming Interface*, Programsko-korisnička sprega
CCR - *Customer Controlled Routing*, Korisničko kontrolisano preusmeravanje poziva
CD - *Call Distribution*, Distribucija poziva
CORBA -*The Common Object Request Broker Architecture*, Arhitektura razrešivača
CTI - *Computer Telephony Integration*, Objedinjenje računara i telefonije
DBA - *Data Base Agent*, Agent baze podataka
FSM - *Finite State Machine*, Automat sa konačnim brojem stanja.
IVR - *Interactive Voice Response*, Emitovanje govornih poruka
MSC - *Message Sequence Chart*, Dijagram sekvence poruka.
SDL - *Specification and Description Language*, Jezik za specifikaciju i opis.
SG - *Service Group*, Služba
SIP - *Session Initiation Protocol*, Protokol za iniciranje sesije
PSTN - *Public Switched Telephone Network*, Javna telefonska mreža
RTP - *Real-time Transfer Protocol*, Protokol za razmenu medija tokova u realnom vremenu
RTCP - *Real-time Transfer Control Protocol*, Protokol za kontrolu razmene medija tokova u realnom vremenu
TCP - *Transmission Control Protocol*, Protokol upravljanja prenosom podataka
UDP - *User Datagram Protocol*, Protokol korisničkih datagrama
VCC - *Virtual Call Center*, Virtualni centar za obradu poziva
VM - *Voice Mail*, Služba govorne pošte

SADRŽAJ

1. UVOD

U ovom radu je izvršeno je izdvajanje automata za emitovanje govornih poruka (eng. *Interactive Voice Response - IVR*) iz VCCWindows servera, koji predstavlja jedan od servera virtuelnog centra za obradu poziva. Objašnjena je njegova uloga, struktura, veza sa okruženjem i realizacija. Kao rezultat izdvajanja ove komponente nastao je VCCIVRServer koji je realizovan u programskom jeziku C++. Korišćeno je *Microsoft Visual C++* radno okruženje, dok je odredišna platforma PC računar sa *Microsoft Windows 2000* operativnim sistemom.

Polaznu tačku u radu predstavlja virtuelni centar za obradu poziva razvijen na Katedri za Računarsku Tehniku i Računarske Komunikacije, Fakulteta Tehničkih Nauka u Novom Sadu. Iz prvobitnog sistema izdvojen je IVR automat sa pratećim modulima i od tako izdvojenih elemenata je stvoren novi server - IVR server.

Rad sadrži šest poglavlja. U drugom poglavlju je dat uvod u VCC (eng. *Virtual Call Center - VCC*) sistem. Uopšteno su objašnjeni sistemi za masovno usluživanje. Prikazana je arhitektura virtuelnog centra za obradu poziva. Objašnjena je funkcionalnost IVR automata i protokola za prenos podataka u realnom vremenu (eng. *Real-time Transport Protocol - RTP*). Navedene su osnovne klase jezgra komunikacione programske podrške, sa elementima programske izvedbe.

Treće poglavlje predstavlja opis IVR servera i organizovano je u sedam podpoglavlja. U prvom potpoglavlju naveden je zadatak rada, nakon čega je u drugom izvršena analiza problema i navedeni razlozi izdvajanja IVR automata. Treće potpoglavlje opisuje postupak izdvajanja IVR automata i objašnjene su inicijalizacione datoteke *log.ini* i *Distribution.ini*. U četvrtom potpoglavlju dat je opis realizacije IVR automata, gde su detaljno opisane klase potrebne za funkcionisanje automata. Redosled akcija koje se odvijaju prilikom emitovanja govorne poruke dat je u petom potpoglavlju, dok su u šestom i sedmom potpoglavlju predstavljeni SDL i MSC dijagrami.

U četvrtom poglavlju prikazani su testovi kojim je izvršeno ispitivanje funkcionalnosti realizovanog IVR servera. Ispitivanje je sprovedeno korištenjem *CppUnit* radnog okruženja.

U petom poglavlju je iznet kratak zaključak. Šesto poglavlje sadrži spisak korišćene literature.

2. UVOD U VCC SISTEM

2.1 Sistemi za masovno usluživanje

Osnovni koncept virtuelnog centra za obradu poziva, kao sistema za masovno usluživanje (engl. *Call Center*) je automatska distribucija poziva (eng. *Automatic Call Distribution - ACD*). Ovaj koncept je vrlo sličan konceptu reda na jednom šalteru u banci. U banci korisnici formiraju jedan red ispred šaltera. Kada službenik na šalteru postane slobodan, osoba iz reda koja je najbliža šalteru se pomera napred i biva uslužena.

Slično ovome, kod automatske distribucije poziva, pozivi su usluženi na principu prvi pristigli - prvi usluženi. Sistem automatski odgovara na poziv i, ako je to potrebno, smešta poziv u red čekanja sve dok se ne prosledi nekom operateru (eng. *Agent*). Čim operater postane slobodan, on uslužuje prvi poziv u redu.

Automatska distribucija poziva uglavnom nema tako jednostavnu ulogu, kao što je serijska obrada poziva. Sistem obično nudi mogućnosti za postavljanje različitih tretmana za različite korisnike. Na primer, ako se radi o međugradskom pozivu, on može dobiti veći prioritet od ostalih ili se, na primer, mogu različito tretirati korisnici koji u nekom sistemu naručuju proizvode od onih koji imaju potrebu za tehničkom podrškom. Ovi problemi se mogu rešiti podelom sistema na službe i određivanjem prioriteta date službe u odnosu na ostale službe. Obično se korisnicima koji su stavljeni na čekanje pušta muzika i slično. Pri projektovanju i konfigurisanju sistema se teži minimizaciji vremena čekanja korisnika i obezbeđivanju načina za prosleđivanje poziva u različite redove čekanja.

Ceo proces obično nadgledaju nadzornici, koji mogu da ocene efikasnost različitih redova čekanja i da uoče mesta na kojima se mogu pojaviti potencijalni problemi. Bolji sistemi čak mogu da pokažu prosečno vreme čekanja i broj neusluženih poziva. Da bi sistem prilagodili promenama u gustini saobraćaja poziva, nadzornici mogu vršiti fina podešavanja rasporeda zaposlenih operatera ili čak i izvršiti potupunu rekonfiguraciju sistema u realnom vremenu.

Jedan ovakav sistem za masovno usluživanje, realizovan je sistemom za virtualnu obradu poziva.

Sistem za virtualnu obradu poziva predstavlja programsko rešenje koncepta sistema za masovno usluživanje. On eliminiše potrebu za korišćenjem skupih telefonskih centrala.

2.2 Virtualni centar za obradu poziva - VCC

Virtualni centar za obradu poziva je složen komunikacioni sistem, koji objedinjuje elemente računarske i telekomunikacione opreme. VCC je sistem za masovno usluživanje i omogućava opsluživanje poziva u slučajevima kada se očekuje pojava velikog broja poziva u određenom vremenskom intervalu. Njegov osnovni koncept je automatska distribucija poziva. VCC treba da obezbedi usluge raznim korisnicima, kako telefonskim tako i Internet pretplatnicima. Baziran je na internet/intranet mrežama.

Razvijan je u C++ programskom jeziku, Microsoft Visual C++ 6.0 razvojnom okruženju. Baziran je na sistemskoj programskoj podršci za rad sa konačnim automatima (Kernel). Zbog svoje distribuirane prirode koristi CORBA (eng. *The Common Object Request Broker Architecture*) model. Kao baza podataka koristi se *Sybase* ASE v12.5.

Postoje sledeće kategorije krajnjih korisnika VCC sistema:

- Analogni pretplatnici: Poziv se obavlja klasičnim telefonskim aparatom, priključenim na javnu telefonsku mrežu (*eng. Public Switched Telephone Network - PSTN*);
- Digitalni pretplatnici: Poziv se obavlja preko digitalnog terminala (*eng. Integrated Services Digital Network - ISDN*), koji je priključen na PSTN, odnosno globalnu telefonsku mrežu (*eng. Global Switched Telephone Network-GSTN*), preko osnovnog odnosno primarnog pristupa (*eng. Basic Rate Interface/Primary Rate Interface - BRI/PRI*);
- Internet pretplatnici: Koriste pretraživač (*eng. browser*) za pristup centru za obradu poziva;
- Internet/Intranet pretplatnici: Koriste H.323 terminal (PC sa mrežnom karticom, *sound blaster*, *hands-free* slušalice), i imaju pristup bazi podataka, kojoj upućuju upite (*eng. queries*) za pristup centru za obradu poziva.

Kategorije agenata virtuelnog centra za obradu poziva su sledeće:

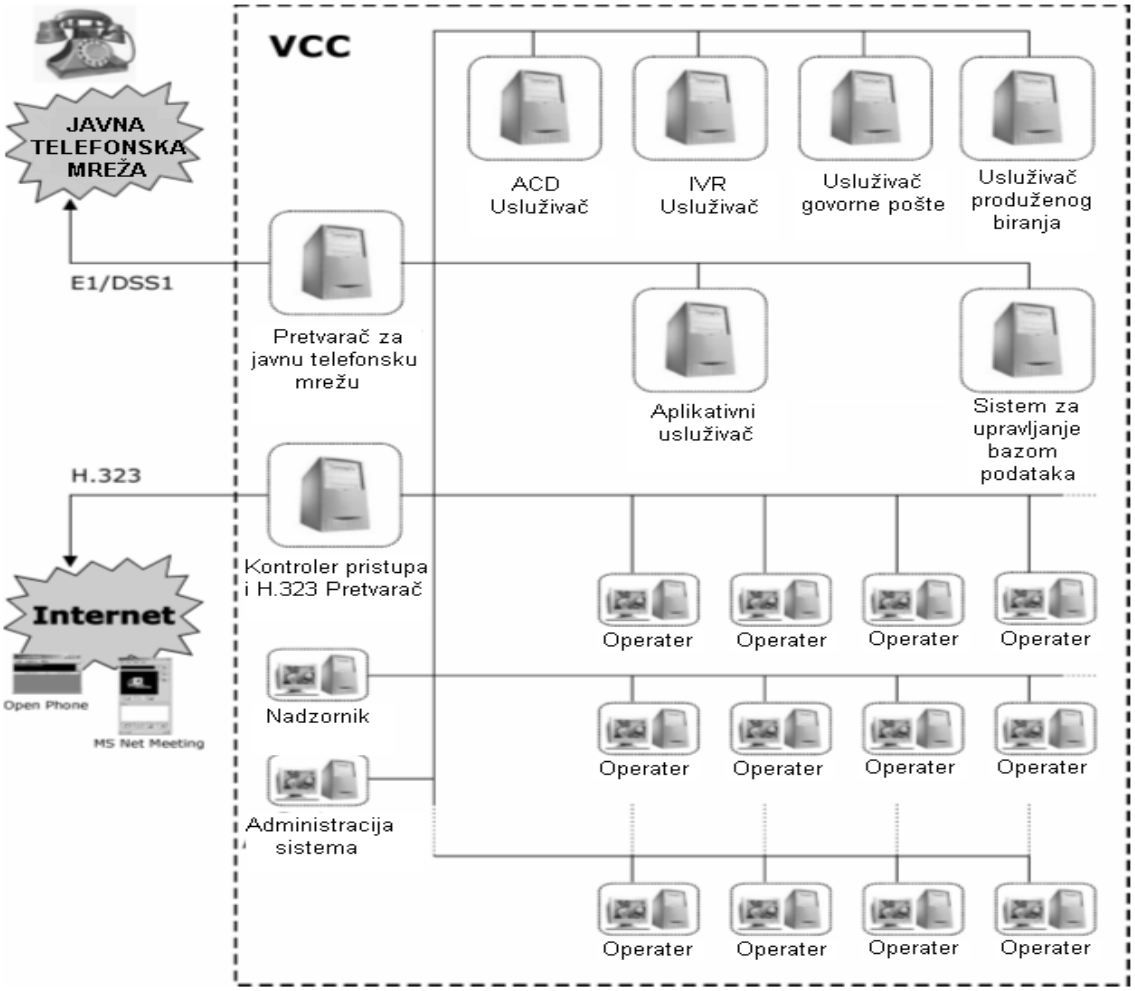
- Operater koji odgovara na pozive prosleđene od strane servera za objedinjenje računara i telefonije (*eng. Computer Telephony Integration - CTI*), koristeći klasičan telefonski aparat;
- Operater koji koristi H.323 terminal za komunikaciju sa korisnikom;
- Operater koji odgovara na telefonske pozive i *e-mail* poruke, koje uključuju tekst, sliku i video zapise;
- Drugi tipovi agenata, koji mogu biti naknadno uvedeni. Na primer operater koji pomoću H.323 opreme pruža usluge iz svoje kuće (*eng. remote agent*).

VCC usluge se obezbeđuju korišćenjem različitih kategorija servera, uključujući sledeće:

- *Domain Name Server (DNS*) i *Proxy* server;

- Server elektronske pošte (*Mail Server*);
- *World Wide W*eb (WWW) server;
- H.323 *Gatekeeper*;
- H.323 MCU *(Multipoint Control Unit)* koji se koristi za H.323 miltimedijalne konferencije;
- Nadzor, održavanje i operativno vođenje (eng. *Operation, Administration & Maintenance - OAM*);
- Servera za objedinjenje računara i telefonije (eng. *Computer Telephony Integration - CTI*);
- *Interactive Voice Response (IVR*) Server;
- Aplikacioni server;
- Server baza podataka.

Postojeća arhitektura VCC-a sadrži različite servere programske podrške (logičke servere). Svaki se može izvršavati na posebnom računaru ili više njih može da se pokreće na istom računaru (Slika 2.1).



Slika 2.1 Osnovna arhitektura VCC-a

Mogućnosti ovog VCC sistema su sledeće:

- prenos i parkiranje/aktiviranje dolaznih poziva;
- mogućnost definisanja individualnih servisa;
- agent je nezavisan od radnog mesta ili radnog PC-a;
- isti agent može da radi sa jednim ili sa više servisa;
- nadgledanje, statistika i nedeljna raspodela rada agenata;
- lokalni pozivi između agenata – PBX funkcionalnost;
- odlazni/dolazni pozivi ka/od PSTN mreže kroz PSTN kapiju (pretvaranje DSS1 u internu Q.71 baziranu signalizaciju i obrnuto);
- odlazni/dolazni pozivi ka/od interneta i intraneta (bazirano na H.323) kroz H.323 kapiju (pretvaranje H.323 u internu Q.71 baziranu signalizaciju i obrnuto).

Proširivanje sistema novim službama je potpuno nezavisno od obrade poziva. Obrada poziva koja je realizovana u VCC sistemu je bazirana na ITU-T Q.71 modelu obrade poziva.

2.2.1. Funkcionalni delovi VCC-a

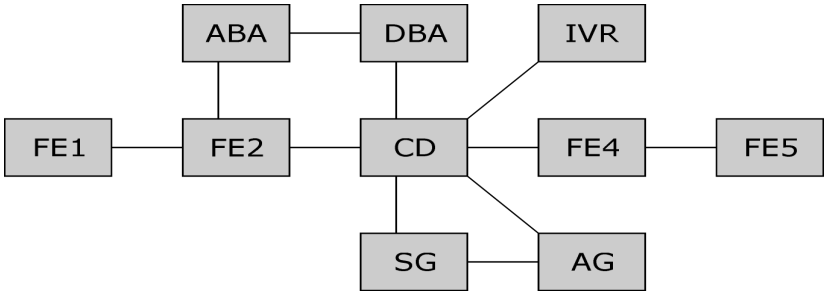
Osnovni funkcionalni delovi sistema za virtualnu obradu poziva (Slika 2.2) su:

- Q.71 model obrade poziva - FE1, FE2, FE4 i FE5 (*eng. Functional Entity "n"*);
- Raspodela poziva (*eng. Call Distributor - CD*);
- Služba (*eng. Service Group - SG*);
- Operater (*eng. Agent- AG*).

Ostali funkcionalni delovi koji se koriste u radu sistema su:

- AB analiza (*eng. AB Analysis - ABA*);
- Operater baze podataka (*eng. Data Base Agent - DBA*);
- Emitovanje govornih poruka (*eng. Interactive Voice Response - IVR*).

Funkcionalni objekat za distribuciju poziva je uveden kao veza Q.71 modela obrade poziva i AG, SG i IVR objekata. Funcionalni objekat operater predstavlja radno mesto operatera centra za obradu poziva. SG predstavlja grupu agenata i obezbeđuje određenu uslugu korisnicima. Osnovna funkcija IVR objekta je emitovanje govornih poruka korisniku.



Slika 2.2 Funkcionalni delovi VCC sistema

Procedura raspoređivanja poziva započinje kada korinik odabere broj neke od službi centra za obradu poziva. Ukoliko se radi o lokalnom pozivu njegova obrada se vrši preko FE2 objekta. Obrada poziva se nastavlja AB analizom radi utvrđivanja B korisnika. Ukoliko se radi o ACD pozivu, poziv se prosleđuje na CD i koji pri tome biva obavešten o rezultatima AB analize.

Ukoliko služba nije aktivna, poziv se preusmerava na IVR pomoću koga se korisniku emituke odgobarajuća poruka, u zavisnosti da li je služba aktivna ili ne. Ako je odgovarajuća služba aktivna, ispituje da li postoji operater prijavljen na tu službu, spreman da prihvati poziv. Ukoliko takav operater postoji, poziv mu se prosleđuje. Operater može da prihvati ili odbije ponuđen poziv. Ako operater prihvati poziv, poziv je raspoređen.

2.3 IVR automat

Jedna od mogućnosti VCC-a, koja povećava efikasnost sistema, je automatizacija rutinskih aktivnosti koje se ponavljaju. Ovo se postiže korišćenjem automata za emitovanje govornih poruka (eng. *IVR - Interactive Voice Response*). IVR predstavlja deo VCC-a koji omogućava prenos poruka korisniku u vidu zvuka i/ili slike. IVR omogućava korisnicima da koriste telefon sa tonskim biranjem praktično kao kompjuterski terminal. Ovim se ispunjenje rutinskih zahteva korisnika automatizuje i izbegava se dodavanje novih operatera.

Najsavremenije IVR tehnologije obuhvataju:

- automatsko prepoznavanje govora – omogućava korisnicima pogodnost korišćenja govornih komandi umesto korišćenja telefonske tastature. Nezavisno od boje glasa korisnika, ovakvi sistemi su u mogućnosti da prepoznaju jednostavne reči kao što su cifre 0-9, kao i reči “da”, “ne”, itd. i to na različitim jezicima;
- konverzija teksta u govor – mogućnost konverzije tekstualne datoteke u govor koji se pušta korisniku.

U virtuelnom centru za obradu poziva IVR predstavlja jednu od usluga servera koji objedinjuje računare i telefoniju (eng. *Computer Telephony Integration – CTI* server). IVR automat se može posmatrati i kao specijalna vrsta operatera. Interakcija korisnika sa automatom ostvarena je preko cifara korisničke tastature. Te cifre automat prihvata kao poruke i izvršava određene akcije po njihovom prijemu. Akcije koje automat izvršava su uglavnom akcije emitovanja i snimanja poruka. Taj skup akcija može biti proširen akcijama preusmeravanja poziva, slanja poruka elektronske pošte, itd.

Jedna od prednosti IVR tehnologije je podrška korišćenju više jezika, odnosno emitovane poruke mogu biti na različitim jezicima (eng. *Multilingual support*).

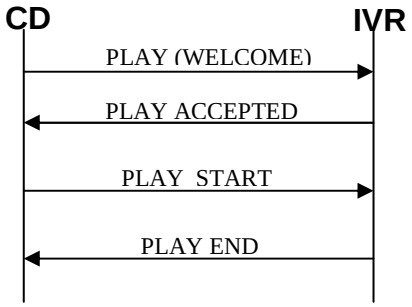
Mogućnosti IVR automata mogu se iskoristiti kada se poziv nalazi u redu čekanja na uslugu. Tako, korisnik za vreme čekanja na slobodnog operatera može stupiti u komunikaciju sa ovim automatom. Komunikacija može biti prekidna ili neprekidna. Na osnovu CCR (eng. *Customer Controlled Routing*) tabele se definiše da li se komunikacija raskida u slučaju pojave slobodnog operatera ili ne.

Kada primi zahtev za pružanje određene usluge korisniku IVR automat konsultuje IVR tabelu. Izgled IVR tabele prikazan je na slici 2.3.

IVR_ID	Identifikator na osnovu kog se bira jedna n-torka iz IVR tabele.
IVR_SCRIPT	Sadrži niz cifara nakon čijeg prijema automat treba da emituje određenu govornu poruku.
IVR_DATA	Naziv datoteke koja sadrži govornu poruku koju treba emitovati korisniku.
IVR_TYPE	Određuje da li govornu poruku treba neprekidno ponavljati ili ne. Takođe određuje da li treba pružiti korisniku dodatnu uslugu prihvatanjem cifara koje korisnik šalje preko telefonske tastature.

Slika 2.3. Izgled IVR tabele

IVR komunicira sa CD funkcionalnim objektom. Način njihove komunikacije prikazan je MSC dijagramima na slici 2.4.



Slika 2.4. MSC dijagram komunikacije IVR i CD funkcionalnih objekata u slučaju da postoji slobodan IVR automat.

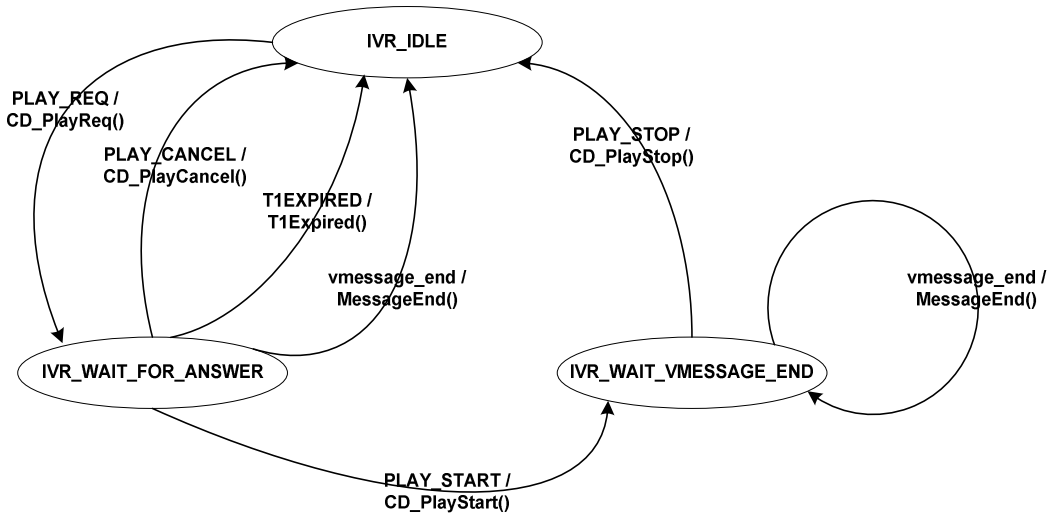
Komunikacija između ova dva automata je ista bez obzira da li se radi o lokalnom ili udaljenom pozivu. Poruke koje CD automat šalje IVR automatu su: ***PLAY_REQ***, ***PLAY_START***, ***PLAY_CANCEL***, ***PLAY_*STOP** i ***DIGIT***. Poruke koje IVR automat šalje CD-u su: ***PLAY_ACCEPTED*** i ***PLAY_END***.

Stanja u kojima može da se nađe IVR automat data su u tabeli na slici 2.5., dok je na slici 2.6. dat dijagram prelaza stanja IVR automata u zavisnosti od primljene poruke u trenutnom stanju.

r.b	*Naziv stanja*	*Opis stanja*
1.	**IVR_IDLE**	Automat je slobodan.
2.	**IVR_WAIT_FOR_ANSWER**	Stanje čekanja odgovora od strane CD automata

3.	**IVR_WAIT_VMESSAGE_END**	Stanje čekanja kraja govorne poruke

Slika 2.5. Stanja u kojima se može naći IVR automat



Slika 2.6. Dijagram prelaza stanja IVR automata

2.3.1 Standardne procedure IVR automata

Prihvatanje zahteva za uslugu (poruka: *PLAY_REQ*, parametar: *REQUEST_ID*): IVR prihvata ovaj zahtev samo ako se nalazi u stanju slobodan. U suprotnom, ova poruka se ignoriše. Porukom ***PLAY_REQ*** CD automat ispituje da li postoji slobodan IVR automat. U slučaju da je slobodan, IVR šalje poruku ***PLAY_ACCEPTED*** CD automatu i prelazi u stanje čekanja na odgovor CD automata. Istovremeno se pokreće vremenska kontrola i ukoliko IVR ne primi novu poruku u određenom vremenskom periodu prelazi u stanje slobodan.

Početak pružanja usluge korisniku (poruka: *PLAY_START*, parametri: *IVR_SCRIPT_ID, IP*): Ukoliko se IVR nalazi u stanju čekanja na odgovor CD automata i prihvati poruku ***PLAY_START*** IVR započinje emitovanje govorne poruke korisniku. Emitovana poruka je definisana parametrom ***IVR_SCRIPT_ID***. Pre početka emitovanja poruke IVR konsultuje gore prikazanu IVR tabelu. Na osnovu nje IVR saznaje kakvu uslugu treba da pruži korisniku. Parametar ***IP*** sadrži IP adresu korisnika.

Oslobađanje IVR automata (poruka: *PLAY_CANCEL*, nema parametara): Kada se nalazi u stanju čekanja na odgovor CD automata IVR može da primi ***PLAY_CANCEL*** poruku. U tom slučaju IVR automat prelazi u početno slobodno stanje.

Kraj pružanja usluge korisniku (poruka: *PLAY_STOP*, nema parametara): U toku emitovanja poruke korisniku IVR automat može da dobije poruku ***PLAY_STOP*** tada se emitovanje poruke raskida i automat prelazi u početno, slobodno stanje. Ukoliko se govorna poruka završi, a IVR ne treba da pruži korisniku dodatnu uslugu, niti da ponavlja

poruku što je definisano IVR tabelom, IVR automat prelazi u početno, slobodno stanje. Ukoliko treba da odgovori na neku konbinaciju cifara koje korisnik šalje automat prelazi u stanje čekanja cifara. Iz tog stanja IVR prelazi u početno po prijemu poruke ***PLAY_STOP***.

2.4 Protokol za prenos podataka u realnom vremenu (RTP)

RTP (eng. *Real-Time Transport Protocol*) definiše standardizovani format paketa za prenos audio i video sadržaja, kao i podataka, preko Interneta. U konkretnom slučaju, u VCC sistemu, RTP se koristi za prenos audio i video signala korisniku od strane IVR servera.

RTP ne poseduje standardni TCP (eng. *Transmission Control Protocol*) ili UDP (eng. *User Datagram Protocol*) mrežni prolaz preko koga komunicira. Standard jedino zahteva da se UDP komunikacija odvija preko parnog mrežnog prolaza a sledeći viši neparni mrežni prolaz se koristi za RTCP (eng. *Real-Time Transport Control Protocol*) komunikaciju. RTP se konfiguriše tako da koristi mrežne prolaze u opsegu od 16384 do 32767, iako standard to ne zahteva.

RTP može prenositi bilo koje podatke sa karakteristikama realnog vremena, kao što je interaktivna audio i video komunikacija. Uspostava i raskid veze se obično ostvaruju SIP (eng. *Session Initiation Protocol*) ili H.323 protokolom. RTP i RTCP su razvijeni tako da budu nezavisni od transportnog i mrežnog nivoa. Aplikacije koje koriste RTP su manje osetljive na gubitak paketa ali su obično veoma osetljive na njihovo kašnjenje. Zato je za takve aplikacije UDP bolji izbor od TCP.

Protokol nema mehanizam za obezbeđenje blagovremene isporuke paketa i ne daje QoS (eng. *Quality of Service*) garancije. Takođe je moguća isporuka paketa u neodređenom redosledu. Ovi problemi se rešavaju na nivou aplikacije tako što ona utvrđuje raspored paketa na osnovu primljenih RTP paketa, dok RTCP paketi obezbeđuju informaciju o kvalitetu prijema, što se takođe može iskoristiti za lokalno podešavanje.

RTCP (*engl. Real-Time Transport Control Protocol*) se koristi za nadgledanje mreže u toku veze i komunikaciju krajnjih tačaka izvan opsega. Npr. u slučaju zagušenja mreže, moguće je preći na kodiranje sa većim faktorom kompresije, ili npr. prijemnik može da menja veličinu reprodukcionog bafera u slucaju promene mrežnog kašnjenja.Mehanizam komunikacije izvan opsega omogućava formiranje dodatnog paralelnog toka podataka (npr. tekst koji prati video tok – “titl”). Pošto su RTCP poruke kratke, moguce je preneti više njih u jednom UDP paketu.

Predajnik povremeno šalje ***izveštaj predajnika*** koji sadrži apsolutni vremenski pečat. Napominje se da je to jedini mehanizam koji omogućava prijemniku da sinhronizuje više tokova.

Prijemnik povremeno šalje ***izveštaj prijemnika*** koji izveštava predajnika o uslovima prijema:

- Predajnik saznaje uslove prijema kod svih prijemnika;
- Prijemnici mogu da prilagode učestanost izveštavanja i tako ograničavaju opterećivanje mreže i predajnika.

2.5 Jezgro komunikacione programske podrške (Kernel)

Kernel je okruženje u kome je realizovan virtualni centar za obradu poziva. Sadrži klase generalizovane sistemske programske podrške koje realizuju mehanizam automata. Osnovna funkcija Kernel-a je prelazak automata iz tekućeg stanja u naredno stanje, pri čemu se obavljaju potrebne radnje (određivanje stanja i funkcije obrade). Takođe omogućavaju sadejstvo proizvoljnog broja automata. Time je pojednostavljena realizacija konačnog determinističkog automata, a njegova realizacija je svedena na definisanje funkcija prelaza. Generalizovana sistemska podrška se sastoji iz sledećih celina:

- Zajedničke sistemske rutine jezgra dostupne korisničkom programu (automatu);
- Osnovne klase za realizaciju konačnih deterministčkih automata, delova komunikacionog protokola;
- Osnovne klase sistema konačnih deterministčkih automata.

Iako se podrška može podeliti na navedene celine, one su međusobno povezane kako bi se ostvarilo kvalitetno funkcionisanje grupe automata.

2.5.1 Zajedničke sistemske rutine jezgra

Jezgro se sastoji iz tri dela:

- Rukovalac memorijom;
- Rukovalac porukama;
- Rukovalac vremenskim kontrolama.

Sva tri segmenta jezgra su celine koje obavljaju nezavisne funkcije ali su povezane funkcionisanjem sistema. Pri projektovanju svake celine posebno, uzete su u obzir i ostale celine, čime je realizacija pojednostavljena, ali zato postoji međusobna logička povezanost svih klasa jezgra.

Jezgro u osnovi omogućava razmenu poruka, što se svodi na prelančavanje skladišta iz liste u listu. Prilikom prelančavanja skladišta menja se njegov sadržaj. Da bi se pojednostavila realizacija jezgra, ubrzao rad sistema i pružila podrška sistemu automata definisano je zaglavlje zajedničko za sve poruke koje će kasnije biti opisano detaljnije.

2.5.2 Osnovna klasa za realizaciju konačnih deterministčkih automata

Klasa za realizaciju konačnih deterministčkih automata definiše mehanizam samog automata, kao i mehanizme za rukovanje porukama, vremenskim kontrolama, sadržajem poruka i memorijom. Za funkcionisanje sistema neophodno je jezgro u realnom vremenu. Na katedri za računarsku tehniku, razvijeno je upravo jedno ovakvo

jezgro. Podrška realizaciji automata definiše spregu (*eng. interface*) ka svim funkcijama jezgra, rukovanje sadržajem poruka i praćenje (*eng. logging*) događaja u sistemu.

Klasa za realizaciju konačnih determinističkih automata se sastoji iz četiri celine koje su modelovane sledećim klasama:

- *MessageHandler* – za rukovanje sadržajem poruka;
- *ProgrammingInterface* – za prilagođenje sistema jezgru;
- *LogAutomate* – za praćenje događaja u sistemu (logovanje);
- *FiniteStateMachine* – osnovna klasa za realizaciju automata.

Pored klasa potrebnih za realizaciju konačnih determinističkih automata, postoje i dve apstraktne klase *LogInterface* i *MessageInterface* koje definišu spregu koju sistem determinističkih automata koristi za praćenje događaja u sistemu, odnosno rukovanje sadržajem poruka. Sprega klasa za rukovanje sadržajem poruka omogućava razdvajanje realizacije automata od oblika poruka.

Realizovan je i jedinstven sistem za praćenje događaja u sistemu automata komunikacionih protokola. On omogućava i olakšava traženje grešaka i testiranje sistema, kao i kasnije praćenje rada uz minimalno opterećenje sistema.

2.5.3 Rukovanje sistemskim funkcijama jezgra

Jezgro u realnom vremenu je neophodan deo sistema automata. U slučaju promene platforme, obično se jezgro prilagodi novoj platformi, pa nema potrebe za promenom jezgra. Iako sistem automata funkcioniše nezavisno od jezgra, on je usko povezan sa jezgrom koje mu pruža sistemske funkcije za rad sa resursima sistema:

- rukovanje memorijom;
- rukovanje vremenskim kontrolama;
- razmena poruka.

Potreba da se funkcionisanje automata odvoji od konkretnog jezgra, kako bi se omogućila jednostavna promena jezgra, dovela je do definisanja sprežne klase automata i jezgra. Sprežna klasa (KernelAPI) definiše jedan generalizovani skup funkcija za rad sa jezgrom, koji sakriva realizacijuaciju jezgra od automata.

Za razliku od klasa za rukovanje sadržajem poruka i logovanjem, nije definisana sprega za rukovanje jezgrom, nego je sve realizovano u jednoj klasi, u kojoj se nalaze i objekti samog jezgra. Rukovanje jezgrom nije realizovano mehanizmom sprega (*interface*), koji je mnogo fleksibilniji u slučaju promene jezgra, jer se samo jezgro jako retko menja. Time je sistem pojednostavljen, ali je u slučaju izmene jezgra potrebno prekodirati klasu KernelAPI da podržava novo jezgro.

Funkcije klase KernelAPI se mogu podeliti u sledeće grupe:

- funkcije inicijalizacije;
- funkcije za rad sa memorijom;
- funkcije za rukovanje porukama;
- funkcije vremenkih kontrola.

Funkcije definisane u klasi KernelAPI, nisu uvek i funkcije koje će se koristiti pri realizacijiaciji automata, jer neke od njih sistem automata enkapsulira kako bi se pojednostavio rad sistema (npr. funkcije vremenskih kontrola).

2.5.4 Klasa konačnog determinističkog automata

Klasa FiniteStateMachine je osnovna klasa svakog konačnog determinističkog automata u sistemu. Ona u sebi sadrži i sve potrebne mehanizme i podatke za inicijalizaciju i funkcionisanje automata.

Radi jednostavnijeg rukovanja porukama i jednostavnije komunikacije među automatima, uvode se još tri strukture podataka. Strukture podataka koje opisuju automat su:

- *struktura za opis stanja automata:* Svaki automat je definisan stanjima u kojima može da se nalazi. Svako stanje je definisano svojim funkcijama prelaza tj. porukama na koje reaguje. Struktura SState u sebi sadrži sledeće podatke potrebne za definisanje jednog stanja automata:
  - broj grana u stanju (dimenziju niza grana automata) - u slučaju da nije definisana grana koja obrađuje neku poruku, funkcija *UnexpectedEventProcPtr* će biti izvršena;
  - skup poruka dozvoljenih u datom stanju;
  - funkcije koje će se izvršiti radi obrade date poruke.
- *struktura za opis grane automata*: Struktura grane je uređeni par čiji su elementi kôd poruke (događaja) i funkcija obrade događaja – funkcija prelaza.
- *struktura za jednostavnije rukovanje vremenskim kontrolama:* Struktura *TimerBlock* nije direktno vezana za funkcionisanje automata, ali mnogo olakšava rukovanje vremenskim kontrolama. U svakom automatu postoji konačan broj vremenskih kontrola čiji su parametri opisani strukturom TimerBlock.

Pored pokazivača na niz sa opisom stanja automata i niz sa opisom vremenskih kontrola automata, klasa FiniteStateMachine sadrži i podatke potrebne za funkcionisanje automata, praćenje rada automata kao i za jednostavniju realizaciju komunikacije između automata.

Funkcije samog mehanizma automata se mogu podeliti u dve grupe:

- funkcije inicijalizacije automata;
- funkcije kontrole automata.

Funkcije inicijalizacije i kontrole su:

- *InitEventProc* – funkcija za inicijalizaciju. Inicijalizuje niz struktura grana i stanja automata. Za svaku granu, funkcija se poziva jednom. Strukture podataka o granama i stanja automata se redom popunjavaju. Grane bi trebalo inicijalizovati po učestanosti njihovog pozivanja, pošto se zbog algoritma pretraživanja niza brže nalazi grana koja je pre inicijalizovana.

- *InitUnexpectedEventProc* – funkcija za inicijalizaciju. Definiše funkciju koja će biti pozvana u slučaju poruke za koju nije definisana grana. Za svako stanje u kome postoji ovakva funkcija, funkcija inicijalizacije mora biti pozvana.
- *FiniteStateMachine* – konstruktor klase u kom korisnik definiše parametre potrebne za definisanje automata. Podaci definisani u konstruktoru su potrebni za zauzimanje memorije potrebne za smeštanje strukture automata, a oni su:
    - *numOfState* - broj stanja automata;
    - *maxNumOfProceduresPerState* – maksimalni broj grana po stanju;
    - *numOfTimers* – broj vremenskih kontrola automata.
- *Initialize* – funkcija koja se poziva u konstruktoru klase. Definisana je od strane korisnika sistema i predviđena da u sebi sadrži i celu inicijalizaciju specifičnog automata, od definisanja automata pozivima funkcija *InitEventProc* i *InitUnexpectedEventProc* do inicijalizacije podataka specifičnih za automat i vremenskih kontrola.
- *GetProcedure* – funkcija za kontrolu mehanizma automata. Koristi se za određivanje funkcije obrade poruke.
- *NoFreeInstances* – funkcija za kontrolu mehanizma automata. Izvršava se kada nema više objekata datog tipa, a stigao je zahtev za novom obradom (vezom).
- *Process* – funkcija za kontrolu mehanizma automata. Preuzima poruku, određuje funkciju obrade poruke i proziva je.
- *FreeFSM* – funkcija za kontrolu mehanizma automata. U slučaju korišćenja mahanizma za slanje poruke nepoznatom automatu, potrebno je nakon raskida komunikacije automat prijaviti u listu slobodnih automata.

2.5.5 Praćenje događaja u sistemu

U cilju jedinstvenog automatskog sistema praćenja događaja bez obzira na mesto zapisa događaja, razvijena je klasa *LogAutomate*. Klasa *LogAutomate* definiše skup funkcija za registrovanje događaja u sistemu, koje su na raspolaganju automatima.

Za sam zapis događaja koriste se objekti izvedeni iz apstraktne klase LogInterface. Razdvajanjem funkcionalnosti zapisa podataka na dve klase omogućena je promena načina zapisa bez izmene koda automata.

Osnovna klasa definiše generičke funkcije za rad sa objektom klase zapisa događaja, gde se on tretira kao tok (*eng*. *stream*). Takođe, definisane su apstraktne funkcije u klasi LogAutomate, koje moraju da se realizuju u svakom tipu automata kako bi se događaji zapisivali uniformno.

Zahvaljujući informacijama sadržanim u zaglavlju poruka i objektu klase automata, zabeleženi podaci se mogu lako pretražiti i analizirati kako bi se otkrile greške u sistemu.

U klasi FiniteStateMachine u svim relevantnim tačkama izvršavanja rada automata, pozivaju se odgovarajuće funkcije zapisa događaja, čime je postignut automatizam praćenja rada sistema.

2.5.6 Sistem konačnih determinističkih automata

Sistem konačnih determinističkih automata je podeljen u dve celine:

- realizacija konačnog determinističkog automata;
- kontrola sistema konačnih determisnitičkih automata.

Funkcionalnost automata je ostvarena u klasi *FiniteStateMachine* iz koje se izvodi klasa svakog tipa automata u sistemu. Klasa FSMSystem u sebi sadrži funkcionalnost kontrole i funkcionisanja sistema konačnih determinističkih automata. Ona ima za zadatak da objedini automate u sistem automata. Obavlja sledeće zadatke:

- vodi računa o tipovima automata u sistemu;
- određuje instancu automata koji treba da obradi poruku;
- realizuje prioritete redova poruka;
- vodi računa o slobodnim instancama automata;
- realizuje mehanizam slanja nepoznatom automatu;
- inicijalizuje instance automata.

Pri inicijalizaciji, svaki objekat svakog tipa automata se registruje u sistemu čime je omogućeno da se poruke raspoređuju odgovarajućem objektu na obradu.

Sistem ima zadatak da vodi brigu o slobodnim automatima, u slučaju da je za dati tip automata definisana mogućnost slanja nepoznatom automatu, što se definiše prilikom inicijalizacije instanci automata tj. njihovog prijavljivanja sistemu. Ako je dati tip automata označen da koristi mehanizam slanja nepoznatom automatu, svaka njegova instanca se ulančava u listu slobodnih automata datog tipa.

Kada sistem preuzme poruku za dati tip automata ona u zaglavlju ima postavljen podatak o instanci automata na UNKNOWN_AUTOMATE, što inicira mehanizam određivanja slobodnog automata. Tada se iz liste uzima prvi automat i poruka mu se prosleđuje na obradu. Ukoliko nema slobodnih automata, sistem proziva funkciju NoFreeInstances, bilo koje instance datog automata.

Funkcija Start klase FSMSystem, redom prolazi kroz redove sa porukama, pomoću funkcije GetNewMessage uzima poruke i određuje automat koji treba da obradi poruke. Ukoliko se želi realizovati neki specifični algoritam prioriteta redova, treba izvesti novu klasu iz klase FSMSytem i realizovati funkciju GetNextMessage tako da uzima poruke iz redova po željenom algoritmu.

2.5.7 Korišćenje sistema konačnih determinističkih automata

Upotreba sistemskih podloga za razvoj konačnih determinističkih automata zahteva da se izvede klasa specifičnog automata iz klase *FiniteStateMachine* tj. realizuju konkretni tipovi raznih automata. Zatim je potrebno instancirati objekat klase FSMSystem, pomoću koga se:

- inicijalizuje jezgro u realnom vremenu;
- postave parametri sistema;
- prijave svi objekti sistemu pri čemu se, za pojedine tipove automata, specificira da li da podrže mehanizam slanja nepoznatom automatu ili ne.

Realizacija konkretnog tipa automata zahteva da se u izvedenoj klasi definišu sledeće funkcije:

- *GetMessageInterface* – vraća pokazivač na objekat koji ume da radi sa tekućom porukom. Ako se pojavi poruka u nepoznatom formatu, treba da obradi grešku (obično generiše izuzetak (eng. *exception*));
- *SetDefaultHeader* – funkcija postavlja parametre unutar zaglavlja poruke na vrednosti koje su uobičajene za dati tip automata. Parametri zaglavlja postavljeni u ovoj funkciji mogu, a ne moraju, da obuhvataju podatak o automatu kome se poruka šalje;
- *GetMbxId* – vraća kôd reda u koji se uobičajeno ulančavaju poruke za tip automata koji se razvija;
- *GetAutomate* – vraca kôd tipa automata koji se razvija i obično se koristi za postavljanje podataka u zaglavlju poruke;
- *SetDefaultFSMData* – poziva se kada treba postaviti podatke instance automata na standardne vrednosti;
- *NoFreeInstances* - funkcija koja treba da se realizuje u slučaju da automat podržava mehanizam slanja nepoznatom automatu. Ova funkcija će biti pozvana kada nema više slobodne instance automata za novu obradu;
- *Initialize* – funkcija koja inicijalizuje automat. Ona treba da inicijalizuje sve strukture potrebne za funkcionisanje automata, kao i podatke specifične za dati tip automata.

Za svako stanje automata potrebno je realizovati funkcije prelaza, kao članice specijalizovane klase. Svaki objekat svakog automata se prijavljuje objektu klase FSMSystem, kako bi bio inicijalizovan i prijavljen sistemu.

3. IVR SERVER

3.1 Zadatak

Cilj rada je da se u *Microsoft Visual C++ 6.0* razvojnom okruženju realizuje izdvajanje IVR komponente virtuelnog centra za obradu poziva iz *VCCWindowsServer* komponente VCC sistema u poseban IVR server. IVR (*Interactive Voice Response*) komponenta se koristi za emitovanje glasovnih poruka korisnicima sistema i realizovana je kao jedan automat sistema jezgra komunikacione programske podrške. Potrebno je:

- Upoznati se sa strukturom i osnovnom funkcionalnošću VCC sistema;
- Izmestiti odgovarajuće automate iz postojećeg sistema automata u novi sistem unutar IVR servera;
- Konfigurisati novonastali distribuirani sistem servera;
- Analizirati funkcionalnost IVR servera i po potrebi je unaprediti.

3.2 Analiza problema

Virtuelni centar za obradu poziva (eng. *Virtual Call Center - VCC*) je složen komunikacioni sistem, koji objedinjuje elemente računarske i telekomunikacione opreme. Omogućuje opsluživanje poziva u slučajevima u kojima se očekuje pojava velikog broja poziva u određenom vremenskom intervalu.

Primer primene su službe pružanja informacija, kao i sve službe kod kojih je zastupljena obimna komunikacija sa korisnicima usluga.

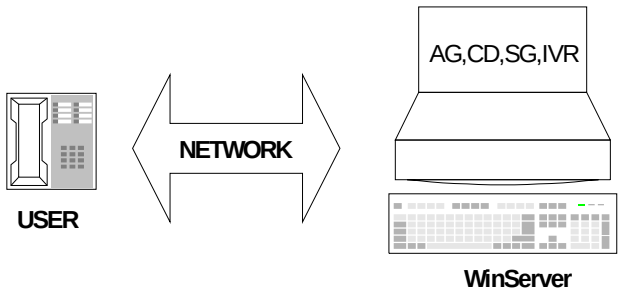
Virtuelni centar za obradu poziva je zasnovan na internet/intranet mreži i decentralizovan je (otud i naziv virtuelni). VCC je decentralizovan na više servera i to:

- **Aplikativni server** (*VCCAppServer*): predstavlja programsku komponentu kojom je realizovan aplikativni usluživač virtuelnog centra za obradu poziva. Baziran je na *CORBA* tehnologiji jer su komponente VCC sistema distribuirane u sistemu. Tekuće stanje podrazumeva realizovane CORBA komponente potrebne za rad VCCAdministrator-a, VCCAgent-a, VCCWinServer-a i VCCSupervizor-a. Bitno je naglasiti da je ova komponeta ključna kod nadogradnje postojećeg rešenja za konkretnu namenu (banke, bolnice, info centri) samo u slučaju da klijent razvija svoje okruženje na CORBA tehnologiji, primenom istog ORB-a (omniORB v4.0.0).
- **Windows server** (*VCCWinServer*): predstevlja centralni deo sistema. To je jezgro celog sistema. U okviru njega su opisani osnovni delovi za funkcionisanje sistema tj. osnovne klase potrebne za njegov rad. Usluživački program nema nikakvih posebnih komandi koje su dostupne korisniku. Pri podizanju usluživačkog

korisničkog programa, uspostavlja se veza sa usluživačkim korisničkim programom zasnovanom na *CORBA*-i, zbog uzimanja informacija o svim službama i operaterima koji mogu postojati u sistemu.

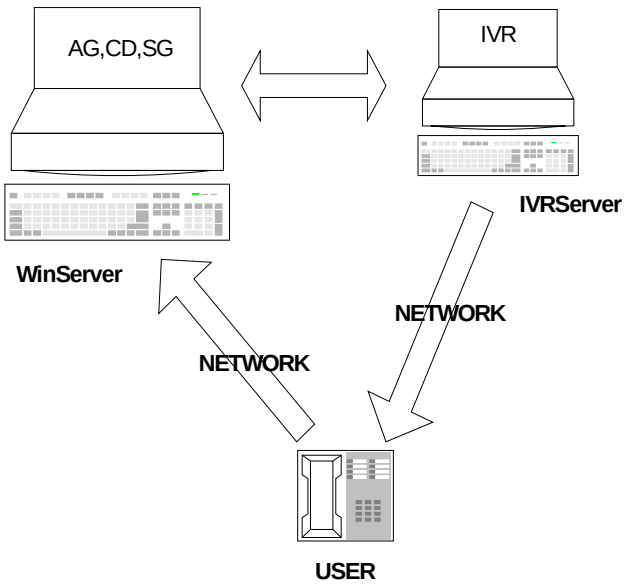
- **Linux server** (*VCCLinuxServer*): programska komponenta sadrži:
  - realizovan DSS1 protokol stek preko L3DSS1, DSS1Dispatcher, LAPD automata;
  - programsku podršku za upravljanje fizičkom arhitekturom kartice VoIPE1;
  - RTP modul za prenos govora sa ISDN na paketsku mrežu i obrnuto.
- **Server govorne pošte** (*VCCVoiceMailServer*): korisnički program koji obezbeđuje mogućnost prijema i čuvanja govornih poruka kada operater iz nekog razloga nije dostupan ili ne može da odgovori na dolazni poziv. Datoj poruci se može pristupiti od strane servisa/agenta.
- **Server za produženo biranje** (*VCCExtendDialingServer*): programska komponenta predstavlja usluživač koji obezbeđuje službu produženog biranja. Omogućava korisniku da u zavisnosti od naknadno odabranih brojeva na tastaturi dobije određene servisne usluge i informacije ili određenog operatera koji će ga opslužiti.

Zbog kompleksnosti VCCWindowsServera, definisane količinom automata angažovanih u obradi poziva (Slika 3.1), IVR funkcionalnost je potrebno izdvojiti u vidu posebne aplikacije i na taj način od nje napraviti posebnu komponentu, čime bi se uz jasno definisanu spregu sa istom omogućila komunikacija sa ostalim komponentama. U skladu s tim, sama realizacija novih zahteva, kako nad samom komponentom (VCCIvrServer) tako i celog sistema, ne bi se odražavala na druge delove sistema.



Slika 3.1. VCCWindowsServer pre izdvajanja IVR automata

Nakon razdvajanja dobijamo još jedan server, VCCIvrServer, koji emituje govorne poruke korisniku (Slika 3.2.). Izdvajanjem IVRServera iz VCCWindowsServera dobijena je nova komponenta u sistemu, kompleksnost VCCWindowsServera je smanjena a stvorena je mogućnost boljeg povezivanja ostalih komponenti VCC sistema sa IVR komponentom.



Slika 3.2. VCCWindowsServer i VCCIvrServer nakon izdvajanja IVR automata

3.3 Postupak izdvajanja IVR automata

Pre početka izdvajanja IVR automata iz VCCWinServera bilo je potrebno izvršiti detaljnu analizu komponenti koje će biti potrebne novom serveru da nesmetano vrši usluživanje korisnika u skladu sa njegovim zahtevima. Pri tome se moralo voditi računa o komunikaciji između novog servera (VCCIvrServer) i VCCWindowsServera, da bi se očuvala izvorna funkcionalnost sistema.

Analizom IVR automata vidi se da je njegova funkcionalnost ostvarena preko nekoliko klasa.

- **Klasa Ivr** – ova klasa obezbeđuje funkcionalnost govornog menija. Opcije menija se biraju korišćenjem cifara telefonske tastature. U sistemu se instancira onoliko IVR automata koliko istovremenih poziva sistem može da prihvati. Klasa *Ivr* nasleđuje klasu *CIVRAutomat*.
- **Klasa CVoiceMenu** - omogućava konfigurisanje hijerarhije menija. Da bi se ono omogućilo korišćena je dinamička struktura,slična strukturi uopštenog stabla.
- **Klasa CIVRAutomat** – u okviru ove klase se vrši inicijalizacija protokola za prenos podataka u realnom vremenu, kao i emitovanje govorne poruke korisniku.
- **Klasa RTP** - RTP se koristi za prenos audio i video podataka korisniku od strane VCCIvrServera.
- **Klasa CdbaData** – pristup bazi podataka iz koje se dobijaju skriptovi na osnovu koji se pravi glasovni meni.

Realizacija ovih klasa biće kasnije detaljno objašnjena.

Pre izdvajanja IVR komponenti proverena je funkcionalnost sistema i u datoteci za praćenje događaja u sistemu (*.*log* datoteka), koja će biti kasnije objašnjena, zabeleženi su događaji dati na Slici 3.3.

```
Wed Apr 11 10:32:39 2007
Msg To:       CD_FSM (0x08),                              Automate ID: 0x00000000
MsgFrom:      SG_FSM (0x09),                              Automate ID: 0x00000001
Received Msg:T_RINGING_ACCEPTED (0x00ae),                 Length: 5  Coding type: 0
09 01 08 ff | ae 00 01 00 | 00 00 00 00 | 00 00 ff ff | ff ff 00 05 | 00
ac 02 00 00 | 00
State: 2 -> 3
_____________________________________________________________________________
Wed Apr 11 10:32:39 2007
Msg To:       IVR_FSM (0x0a),                             Automate ID: 0x00000000
MsgFrom:      CD_FSM (0x08),                              Automate ID: 0x00000000
Received Msg:PLAY_REQ (0x00d9),                           Length: 7  Coding type: 0
08 01 0a ff | d9 00 00 00 | 00 00 ff ff | ff ff cd cd | cd cd 00 07 | 00
ab 04 02 00 | 00 00 00
Start Timer:  (0)
State: 0 -> 1
_____________________________________________________________________________
Wed Apr 11 10:32:39 2007
Msg To:       CD_FSM (0x08),                              Automate ID: 0x00000000
MsgFrom:      IVR_FSM (0x0a),                             Automate ID: 0x00000000
Received Msg:PLAY_ACCEPTED (0x00b0),                      Length: 5  Coding type: 0
0a 01 08 01 | b0 00 00 00 | 00 00 00 00 | 00 00 ff ff | ff ff 00 05 | 00
ab 02 02 00 | 00
State: 3 -> 4
_____________________________________________________________________________
Wed Apr 11 10:32:39 2007
Msg To:       IVR_FSM (0x0a),                             Automate ID: 0x00000000
MsgFrom:      CD_FSM (0x08),                              Automate ID: 0x00000000
Received Msg:PLAY_START (0x00c0),                         Length: 29  Coding type: 0
08 01 0a 01 | c0 00 00 00 | 00 00 00 00 | 00 00 cd cd | cd cd 00 1d | 00
ad 02 01 00 | b0 04 01 00 | 00 00 df 04 | 33 31 39 30 | e4 04 c0 a8 | 03 d9 e5 04 |
39 82 00 00 | 00
Stop Timer:  (0)
State: 1 -> 3
_____________________________________________________________________________
Wed Apr 11 10:32:44 2007
Msg To:       IVR_FSM (0x0a),                             Automate ID: 0x00000000
MsgFrom:      IVR_FSM (0x0a),                             Automate ID: 0x00000000
Received Msg:vmessage_end (0x0307),                       Length: 0  Coding type: 0
0a 01 0a ff | 07 03 00 00 | 00 00 00 00 | 00 00 cd cd | cd cd 00 00 | 00
State: 3 -> 0
_____________________________________________________________________________
Wed Apr 11 10:32:44 2007
Msg To:       CD_FSM (0x08),                              Automate ID: 0x00000000
MsgFrom:      IVR_FSM (0x0a),                             Automate ID: 0x00000000
Received Msg:PLAY_END (0x00b4),                           Length: 5  Coding type: 0
0a 01 08 01 | b4 00 00 00 | 00 00 00 00 | 00 00 ff ff | ff ff 00 05 | 00
ab 02 02 00 | 00
State: 4 -> 3
```

Slika 3.3. Log datoteka VccWindows servera pre izdvajanja IVR automata

Iz ove datoteke može se videti da IVR automat od svih automata u sistemu komunicira samo sa automatom za raspoređivanje poziva (eng. *Call Distributor – CD*). Redosled događaja je onakav kakav smo očekivali, a na korisničkoj strani čuje se pozdravna poruka.

Izdvajanje je izvršeno tako što su sve neophodne komponente za pravilno funkcionisanje IVR automata izdvojene iz VCCWindowsServer aplikacije i na osnovu njih je formirana VCCIvrServer aplikacija. Nakon toga je podešena inicijalna datoteka

Distribution.ini u kojoj je definisan novi IVR server sa automatima za emitovanje govorne poruke (IVR automati). Podešavanje je izvršeno tako da se i VCCWindowsServer i VCCIvrServer pokreću na istom računaru.

Da bi izvršili dodavanje novog servera (VCCIvrServer) u inicijalnoj datoteci *Distribution.ini* originalnog slučaja, u polje *SERVERS* dodat je novi server. Kako je potrebno i VCCWindows i VCCIvr server pokrenuti na istom računaru, njihove IP adrese će biti jednake a razlikovaće se broj mrežnog prolaza. Međutim, pri inicijalizaciji objekta za distribuiranje komponenti sistema uočena je greška. Analizom izvornog koda došlo se do zaključka da je potrebno promeniti nazive servera u konfiguracionoj datoteci za distribuiranje objekata i poređenje u okviru inicijalizacije izvršiti na bazi ovih naziva a ne na bazi IP adresa. Problem je u tome što se prilikom pokretanja VCCWindowsServera nakon poređenja IP adresa inicijalizuje objekat distribucije za datu IP adresu, pa prilikom pokretanja VCCIvrServera koji je na istoj IP adresi neće biti moguća inicijalizacija objekta distribucije tj. sistem javlja grešku da je računar sa datom IP adresom u funkciji.

Da bi se ovaj problem rešio, a pritom i omogućilo pokretanje VCCWindows i VCCIvr servera na istom računara, potrebno je izvršiti izmene u inicijalnoj datoteci i u izvornom kodu sistema. U inicijalnoj datoteci *Distribution.ini* serverima u sistemu su dodeljena konkretna imena i to:

- LinServer – za VCCLinux server;
- WinServer – za VCCWindows server;
- IVRServer – za VCCIVR server.

Ove izmene pregledno su date na slici 3.7. gde su upoređene inicijalne datoteke originalnog slučaja i nakon izvršenih izmena.

Na osnovu ovih izmena potrebno je bilo i u izvornom kodu izmeniti upit prilikom inicijalizacije objekta distribucije, pa se umesto poređenja IP adresa porede konkretna imena servera.Poređenje imena servera se vrši tako što će se ime servera koje bude pročitano iz inicijalne datoteke upoređivati sa imenom koje je unapred poznato i direktno upisano u kod. Kada uslov bude ispunjen (pročitano ime iz datoteke ekvivalentno zadatom), izvršiće se stvaranje novog objekta distribucije. Na ovaj način omogućeno je pokretanje oba servera sa istog računara.

IVR server je pokrenut u VCC sistemu i ispraćeni su događaji koji se upisuju u log datoteci VCCWindows i VCCIvr servera. Redosled događaja koji je zabeležen u saglasnosti je sa redosledom događaja u originalnom sistemu. Na strani korisnika je evidentiran prijem govorne poruke. Log datoteke VCCWindows i VCCIvr servera date su na slikama 3.4 i 3.5.

```
Tue Apr 10 11:03:21 2007
Msg To:       CD_FSM (0x08),                              Automate ID: 0x00000000
MsgFrom:      SG_FSM (0x09),                              Automate ID: 0x00000001
Received Msg:T_RINGING_ACCEPTED (0x00ae),                 Length: 5  Coding type: 0
09 01 08 ff | ae 00 01 00 | 00 00 00 00 | 00 00 ff ff | ff ff 00 05 | 00
ac 02 00 00 | 00
State: 2 -> 3
_____________________________________________________________________________
Tue Apr 10 11:03:21 2007
Msg To:       CD_FSM (0x08),                              Automate ID: 0x00000000
MsgFrom:      IVR_FSM (0x0a),                             Automate ID: 0x00000000
Received Msg:PLAY_ACCEPTED (0x00b0),                      Length: 5  Coding type: 0
```

```
0a 02 08 01 | b0 00 00 00 | 00 00 00 00 | 00 00 ff ff | ff ff 00 05 | 00
ab 02 02 00 | 00
State: 3 -> 4
_____________________________________________________________________________
Tue Apr 10 11:03:26 2007
Msg To:       CD_FSM (0x08),                              Automate ID: 0x00000000
MsgFrom:      IVR_FSM (0x0a),                             Automate ID: 0x00000000
Received Msg:PLAY_END (0x00b4),                           Length: 5  Coding type: 0
0a 02 08 01 | b4 00 00 00 | 00 00 00 00 | 00 00 ff ff | ff ff 00 05 | 00
ab 02 02 00 | 00
State: 4 -> 3
```

Slika 3.4. Log datoteka VCCWindowsServera nakon izdvajanja IVR automata

```
Tue Apr 10 11:03:21 2007
Msg To:       IVR_FSM (0x0a),                             Automate ID: 0x00000000
MsgFrom:      CD_FSM (0x08),                              Automate ID: 0x00000000
Received Msg:PLAY_REQ (0x00d9),                           Length: 7  Coding type: 0
08 01 0a ff | d9 00 00 00 | 00 00 ff ff | ff ff cd cd | cd cd 00 07 | 00
ab 04 02 00 | 00 00 00
Start Timer:  (0)
State: 0 -> 1
_____________________________________________________________________________
Tue Apr 10 11:03:21 2007
Msg To:       IVR_FSM (0x0a),                             Automate ID: 0x00000000
MsgFrom:      CD_FSM (0x08),                              Automate ID: 0x00000000
Received Msg:PLAY_START (0x00c0),                         Length: 29  Coding type: 0
08 01 0a 02 | c0 00 00 00 | 00 00 00 00 | 00 00 cd cd | cd cd 00 1d | 00
ad 02 01 00 | b0 04 01 00 | 00 00 df 04 | 33 31 39 30 | e4 04 c0 a8 | 03 d9 e5 04 |
39 82 00 00 | 00
Stop Timer:  (0)
State: 1 -> 3
_____________________________________________________________________________
Tue Apr 10 11:03:26 2007
Msg To:       IVR_FSM (0x0a),                             Automate ID: 0x00000000
MsgFrom:      IVR_FSM (0x0a),                             Automate ID: 0x00000000
Received Msg:vmessage_end (0x0307),                       Length: 0  Coding type: 0
0a 02 0a ff | 07 03 00 00 | 00 00 00 00 | 00 00 cd cd | cd cd 00 00 | 00
State: 3 -> 0
```

Slika 3.5. Log datoteka VCCIvrServera

Dolazi se do zaključka da izdvajanjem automata za emitovanje govornih poruka na poseban server nije narušena funkcionalnost sistema, jer je u potpunosti očuvan redosled poruka u sistemu, a na korisničkoj strani je emitovana očekivana poruka.

3.3.1 Podešavanje Log datoteke

U cilju realizacije praćenja događaja bez obzira na mesto zapisa događaja, razvijena je klasa *LogInterface*, koja definiše skup funkcija za registrovanje događaja. Zahvaljujući informaciji sadržanoj u zaglavlju poruka i objektu klase automata, zabeleženi podaci se mogu lako pretražiti i analizirati kako bi se otkrile greške.

U klasi *FiniteStateMachine* u svim relevantnim tačkama izvršavanja rada automata, pozivaju se odgovarajuće funkcije zapisa dogadjaja. Iz klase *LogInterface* su izvedene dve klase *UdpLogInterface* i *LogFile*. U navedenim klasama su realizovane funkcije deklarisane u klasi *LogInterface*. Funkcije *LogFile* klase su realizovane tako da se svi događaji upisuju u tekstualnu datoteku (*log* datoteka). U datoteku se upisuju sledeći podaci:

- Tip automata kome se šalje poruka;
- Identifikator automata kome se šalje poruka;
- Stanje automata pre obrade poruke;
- Tip automata koji je poslao poruku;
- Identifikator automata koji je poslao poruku;
- Kod poruke;
- Sadržaj poruke;
- Stanje automata posle obrade poruke.

Podaci kao što su: tip automata, stanje automata, kod poruke, se predstavljaju određenim brojnim vrednostima. Ukoliko bi *log* datoteka sadržala samo takve brojne vrednosti, na osnovu nje bi bilo vrlo teško pratiti nastale promene. Zbog toga, klasa *LogFile* koristi inicijalizacionu datoteku (*log.ini*) za prevođenje brojnih vrednosti u odgovarajuće tekstualne poruke i takvu informaciju upisuje u *log* datoteku.

Zbog obimnosti, nije prikazana inicijalizaciona datoteka za konkretan slučaj VCC sistema, već je na slici 3.6. dat izgled inicijalizacione datoteke nekog proizvoljnog primera.

```
[PROJECT]
ProjectName=VCC server
numOfAutoamtes=50
[AUTOMATES]
;kod automata = ime
0= TIMER SYSTEM
1= FE1 & FE5
2= FE2
3= FE4
4= FE5
[MESSAGES]
;redni broj sloga= kod poruke, ime poruke, kod kodiranja poruke, (kod
automata,…1)
0=0x00cc, AGENT SET STATE, 0
1=0x1301, L3_ALERTING, 1
2=0x1302, L3_CALL_PROCEEDING, 1
3=0x1307, L3_CONNECT, 1
4=0x130f, L3_CONNECT_ACKNOWLEDGE, 1
[TIMERS]
;redni broj sloga= kod automata, kod vremenske kontrole, automat
0= 14, 0, Lapd_T200
1= 14, 1, Lapd_T203
2= 15, 0, L3_T302
```

Slika 3.6. Primer *log.ini* datoteke

3.3.2 Distribuiranje automata u mreži

U cilju pojednostavljenja saradnje između automata na udaljenim serverima, razvijena je klasa *Distribution.* Ona poseduje funkcionalnost uspostave veze i slanja poruka između dva automata koji se nalaze na različitim serverima u mreži. Konfiguracija veza se vrši na bazi informacija sadržanih u datoteci *Distribution.ini.*

Datoteka *Distribution.ini* se mora nalaziti u C:\WINNT direktorijumu i sadrži podatke o sistemu, odnosno:

- podatke o serverima i
- podatke o automatima.

Na ovaj način se pravi jedna vrsta distribuirane mreže automata. Tako da će po prijemu poruke svaki automat znati sa kog servera je stigla poruka i tip automata koji je poslao.

Prilikom pokretanja sistema veza između servera se ostvaruje tako što svaki server pokušava da uspostavi vezu sa svim ostalim serverima čije su adrese postavljene u konfiguracionoj datoteci. Ukoliko je neki server inicijalizovan sa njim se može uspostaviti veza. Ukoliko nije inicijalizovan, sa ostalim serverima će se povezati tek kada se inicijalizuje.

Na slici 3.7. upoređene su inicijalizacione datoteke *Distribution.ini* za originalni sistem i sistem sa VCCIvrServerom. Sistem VCC servera je konfigurisan tako da se VCCWinServer i VCCIvrServer pokreću na istom računaru u mreži.

[IVR] PacketsToWaitVoiceMail=37 **[PSTN_GW]** gwAddress=192.168.0.50 **[H323_GW]** gwAddress=192.168.3.217 **[PROXY]** proxyOUT=192.168.3.217 proxyIN=192.168.3.217 **[SERVERS]** *;ukupan broj servera* NumberOfServers = 2 *;broj servera= ip adresa, port i naziv servera* 0=192.168.0.50,8888,Server0, 1=192.168.3.217,8889,Server1, **[AUTOMATES]** *;aType= srvNum, startId,lastId,compId ,#,* ;aType - tip automata ;srvNum - ukupan broj servera na kojima se nalaze instance datog automata ;startId - startni id automata na datom serveru ;lastId - zadnji id automata na datom serveru ;compId - id servera (iz [SERVERS]) 1= 1, 0,100,1,#, 2= 1, 0,100,1,#, 3= 1, 0,100,1,#, 4= 1, 0,100,1,#, 5= 1, 0,100,1,#, 6= 1, 0,100,1,#, 7= 1, 0,100,1,#, 8= 1, 0,100,1,#, 9= 1, 0,100,1,#, 10= 1, 0,100,1,#, 11= 1, 0,100,1,#, 12= 1, 0,100,1,#, 14= 1, 0,100,1,#, 15= 1, 0,100,0,#, 16= 1, 0,100,1,#, 20= 1, 0,100,1,#, 22= 1, 0,100,1,#, 23= 1, 0,100,1,#, 24= 1, 0,100,1,#, 25= 1, 0,100,1,#, 28= 1, 0,100,0,#,	**[IVR]** PacketsToWaitVoiceMail=37 **[PSTN_GW]** gwAddress=192.168.0.50 **[H323_GW]** gwAddress=192.168.3.54 **[PROXY]** proxyOUT=192.168.3.54 proxyIN=192.168.3.54 **[SERVERS]** *;ukupan broj servera* NumberOfServers = 3 *;broj servera= ip adresa, port i naziv servera* 0=192.168.0.50,8888,LinServer, 1=192.168.3.54,8889,WinServer, ← ***Izmena*** 2=192.168.3.54,8990,IVRServer, ***naziva servera*** **[AUTOMATES]** ;aType srvNum startId,lastId,compId,compId,#, ;aType - tip automata ;srvNum - ukupan broj servera na kojima se nalaze instance datog automata ;startId - startni id automata na datom serveru ;lastId - zadnji id automata na datom serveru ;compId - id servera (iz [SERVERS]) 1= 1, 0,100,1,#, 2= 1, 0,100,1,#, 3= 1, 0,100,1,#, 4= 1, 0,100,1,#, 5= 1, 0,100,1,#, 6= 1, 0,100,1,#, 7= 1, 0,100,1,#, 8= 1, 0,100,1,#, 9= 1, 0,100,1,#, 10= 1, 0,100,2,#, ← ***IVR automati instancirani na IVR serveru*** 11= 1, 0,100,1,#, 12= 1, 0,100,1,#, 14= 1, 0,100,1,#, 15= 1, 0,100,0,#, 16= 1, 0,100,1,#, 20= 1, 0,100,1,#, 22= 1, 0,100,1,#, 23= 1, 0,100,1,#, 24= 1, 0,100,1,#, 25= 1, 0,100,1,#,

29= 1, 0,100,1,#, 30= 1, 0,100,1,#, 33= 1, 0,100,1,#, 38= 1, 0,100,1,#,	28= 1, 0,100,0,#, 29= 1, 0,100,1,#, 30= 1, 0,100,1,#, 33= 1, 0,100,1,#, 38= 1, 0,100,1,#,

Slika 3.7. Upoređivanje *Distribution.ini* datoteka za slučajeve pre i posle izdvajanja IVR automata

U originalnom sistemu u polju *SERVERS* definisana su dva servera:

- Server0 – VCC Linux Server;
- Server1 – VCC Windows Server;

i u okviru njih se rukuje svim automatima u sistemu.

U sistemu sa VCCIvr serverom u polju *SERVERS* postoje tri definisana servera:

- LinServer – VCC Linux server;
- WinServer – VCC Windows server;
- IVRServer – VCC Ivr server;

Windows server i Ivr server su postavljeni i konfigurisani da rade na istom računaru ali na različitim mrežnim prolazima.

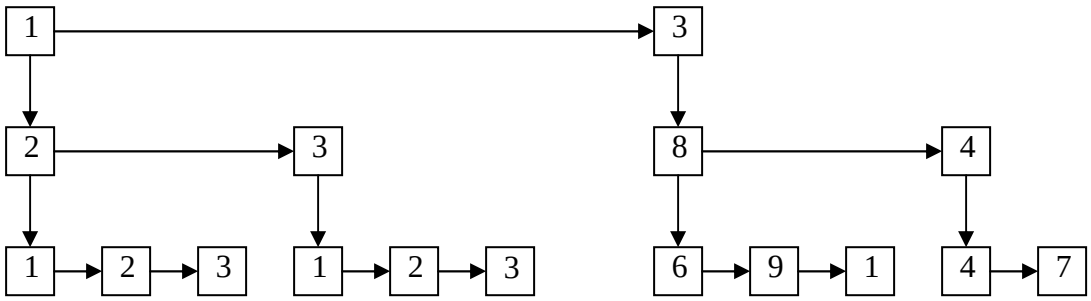
U polju *AUTOMATES* ove konfiguracione datoteke predstavljeni su svi raspoloživi automati i njihov položaj u sistemu. IVR automati (*aType=10*) su konfigurisani tako da rade u okviru VCCIVRServera, a svi ostali automati u okviru VCCWinowsServer-a i VCCLinuxServer-a.

Podešavanje automata za određeni server vrši se tako što se u polje *compId* određenog automata upisuje broj servera, definisanog u polju *SERVERS* ove datoteke, na kome su postavljeni dati automati.

3.4 Opis realizacije IVR automata

3.4.1 Klasa CVoiceMenu

Ova klasa omogućava konfigurisanje hijerarhije menija (Slika 3.8). U tu svrhu korišćena je dinamička struktura, slična strukturi uopštenog stabla.



Slika 3.8 Prikaz hijerarhije menija

Po prijemu svake cifre sa telefonske tastature pristupa se ovoj strukturi i proverava da li se u trenutnom stanju očekuje prihvaćena cifra. Pored informacije o cifri svaki element strukture sadrži polje m_bPlay tipa BOOL koje govori da li posle prijema očekivane cifre treba reprodukovati sadržaj *wave* datoteke ili treba preći u stanje čekanja naredne cifre. Svakom elementu koji nije na dnu hijerarhije (m_bPlay=FALSE) pridružuje se podlista pomoću koje definišemo koje cifre mogu da slede. Polje pExe je pokazivač na *String* u kom se nalazi informacija o *wave* datoteci čiji sadržaj treba da se reprodukuje. Ukoliko je m_bPlay=FALSE, polje pExe ima vrednost NULL, a polje pNextLevel sadrži pokazivač na glavu liste u kojoj treba proveravati da li je naredna prihvaćena cifra očekivana.

Atributi klase CVoiceMenu:

- **PointerListNode*** *HeadOfAllList* - pokazivač na početak liste pokazivača;
- **PointerListNode*** *TailOfAllList* - pokazivač na kraj liste pokazivača;
- **ListNode*** *MainHead* - pokazivač na početak prve liste strukture;
- **ListNode*** *CurrHead* - pokazivač na početak tekuće liste strukture;
- **Bool** *m_bPlay* - da li posle prijema očekivane cifre treba reprodukovati sadržaj wave datoteke ili treba preći u stanje čekanja naredne cifre.

Značajnije strukture u klasi CVoiceMenu:

- **ListNode:**
  - **int** *Digit* - cifra za koju će se vršiti reprodukcija govorne poruke iz određene wave datoteke;
  - **char** *pExe[255]* - pokazivač na string u kom se nalazi informacija o wave datoteci čiji sadržaj treba da se reprodukuje;
  - **struct ListNode*** *pNextLevel* - ovo polje sadrži pokazivač na glavu liste u kojoj treba proveravati da li je naredna prihvaćena cifra očekivana;
  - **struct ListNode*** *pNext* - pokazivač na sledeći element u tekućoj listi;
- **PointerListNode:**
  - **struct ListNode*** *Pointer* - pokazivač na početak tekuće liste;
  - **struct PointerListNode*** *pNext* - pokazivač na sledeći element liste pokazivača;

Metode klase CvoiceMenu:

- ***ListNode* FindDigitInList(int Digit, ListNode* Head)***: Pronalazi element čije polje *Digit* ima zadatu vrednost, u listi čiji je početak dat kao parametar. Povratna vrednost ove funkcije je pokazivač na taj element ili *NULL* ako takvog elementa nema.
- ***ListNode* AddInList(int Digit, ListNode **pHead)***: Ova metoda dodaje element u listu čiji su elementi tipa *ListNode*. Ukoliko je lista u koju se dodaje element prazna (tj. dodati element je jedini u listi), element se smešta u listu pokazivača, a ako nije element se ulančava u tekuću listu. Funkcija vraća pokazivač na dodati element.

- ***void AddInPointerList(ListNode *newPointer)***: Funkcija dodaje elemente u listu pokazivača. Ukoliko je lista prazna, postavlja pokazivač na početak liste pokazivača, a ako nije vrši ulančavanje pokazivača u listu.
- ***void UserAction(int Digit)***: Definiše stanje instance klase. Poziva se po prijemu cifre. Ukoliko se prihvaćena cifra ne očekuje atributu *CurrHead* pokazuje na početak prve liste strukture. U suprotnom pokazuje na početak liste pridružene toj cifri.
- ***void AddMenuOption(void* Pointer, ...)***: Ova funkcija ima promenjiv broj parametara. Prvi parametar je pokazivač na *String* koji sadrži naziv audio datoteke čiji sadržaj treba da se reprodukuje nakon prijema određene kombinacije tastera sa telefonske tastature. Kombinacija cifara koja treba da se prihvati da bi se emitovao zadati zvuk definiše se preko opcionih parametara. Poslednji parametar mora biti –1 da bi se mogao utvrditi tačan broj argumenata funkcije. Prva cifra se postavlja na početak prve liste strukture, a svaka sledeća se postavlja po hijerarhiji u sledeću koja je na nižem nivou od tekuće. Kada se dođe do kraja u strukturu koja je definisana za zadnju cifru u polje *pExe* se kopira pokazivač na *String* koji sadrži naziv datoteke čiji sadržaj treba da se reprodukuje.
- ***void DeallocateList(ListNode *Head)***: Izlančava i oslobađa elemente iz tekuće liste.
- ***void DeallocatePointerList()***: Prazni listu čiji su elementi tipa *PointerListNode*.
- ***void DeallocateLists()***: Oslobađa svu dinamičku memoriju zauzetu u toku izvršavanja programa.
- ***void InitCurrHead()***: Dodeljuje atributu *CurrHead* inicijalnu vrednost.

3.4.2 Klasa IVR

IVR automat obezbeđuje emitovanje govorne poruke korisniku. Pored toga on obezbeđuje funkcionalnost govornog menija. Opcije menija se biraju korišćenjem cifara telefonske tastature. U sistemu se instancira onoliko IVR automata koliko istovremenih poziva sistem može da prihvati. Klasa Ivr nasleđuje klasu CIVRAutomat.

Stanja u kojima je moguće da se nađe IVR automat prikazana su na slici 3.9.

Stanje	**Opis**
S0: IVR_IDLE	Stanje po inicijalizaciji programskog objekta. Automat je slobodan.
S1: IVR_WAIT_FOR_ANSWER	Automat je zauzet od strane nekog CD automata.
S2: IVR_ACTIVE	Automat je aktivan, zauzet.
S3: IVR_WAIT_VMESSAGE_END	Čekanje kraja govorne poruke.
S4: IVR_WAIT_FOR_DIGIT	Čekanje cifara sa korisničke tastature.

Slika 3.9. Stanja IVR automata

Atributi klase IVR:

- **bool** *DigitExcepted* – atribut koji čuva informaciju da li korisniku treba pružiti uslugu pri njegovom izboru neke kombinacije cifara sa korisničke tastature;
- **CVoiceMenu** *Menu* – klasa koja predstavlja strukturu menija;
- **int8** *Message[128]* - naziv datoteke čiji sadržaj treba reprodukovati korisniku;
- **DWORD** *request_id* – atribut koji čuva parametar REQUEST_ID koji se koristi u komunikaciji sa CD automatom;

Strukture u klasi IVR:

- **struct sIVR:**
  - **int** *ivrID* - identifikator na osnovu kog se bira jedna n-torka iz IVR tabele;
  - **char** *ivrScript[10]* - sadrži niz cifara nakon čijeg prijema automat treba da emituje određenu govornu poruku;
  - **char** *ivrType[10]* - određuje da li govornu poruku treba neprekidno ponavljati ili ne;
  - **char** *ivrData[255]* - naziv datoteke koja sadrži govornu poruku koju treba emitovati korisniku;

Metode klase IVR:

- ***int ConvToInt(char Ch)***: Prevodi karakter koji je zadat sa tastature u integer (*ASCII kod*). Ukoliko nije pritisnut taster od 0 do 9 vraća –1, a ako jeste vraća cifru.
- ***void AssignLeftDataFromMsg()***: Postavlja podatke o levom automatu u komunikaciji, u zavisnosti da li se radi o CD, SG ili IVR automatu.
- ***void UpdateMenu(int32 scriptID)***: Kreira meni za određeni *ID skript* koji mu je prosleđen. Meni se pravi za određenu kombinaciju cifara koja je definisana za dati *ID skript* u bazi podataka..
- ***void OnEndFileProc()***: Pravi poruku za kraj komunikacije *vmessage_end*. Zatvara vezu za audio i za video podatke, a zatim zatvara audio i video datoteke iz kojih se čita poruka. Potom se pripremljena poruka šalje IVR automatu koji, nakon prijema poruke, prosleđuje CD automatu poruku da je korisnik uslužen.

Metode za obradu primljenih poruka:

- ***void CD_PlayReq()***: Ova metoda se poziva kada od CD automata stigne poruka PLAY_REQ. Priprema i vraća poruku PLAY_ACCEPTED kojom obaveštava CD automat da je spreman da usluži korisnika.
- ***void CD_PlayStart()***: Ova metoda se poziva kada od CD automata stigne poruka PLAY_START. Iz pristigle poruke uzima *ID skript* na osnovu koga određuje koju poruku treba emitovati korisniku. Postavlja putanje za audio i video datoteku čiji sadržaj treba emitovati korisniku i podešava potrebne parametre za ostvarivanje komunikacije sa korisnikom (IP adresu udaljenog učesnika u komunikaciji, broj audio i video mrežnog prolaza). Uspostavlja vezu sa korisnikom i emituje mu govornu poruku.
- ***void CD_PlayCancel()***: Ova metoda se poziva kada od CD automata stigne poruka PLAY_CANCEL. Postavlja automat u slobodno stanje i briše podatke u automatu sa kojim je komunicirao.

- ***void CDPlayStop()***: Ova metoda se poziva kada od CD automata stigne poruka PLAY_STOP. Prekida emitovanje govorne poruke, zatvara veze za audio i za video podatke, a zatim zatvara audio i video datoteke iz kojih se čita poruka. Priprema poruku PLAY_END koja se šalje CD automatu, postavlja automat u slobodno stanje i briše podatke u automatu sa kojim je komunicirao.
- ***void T1Expired()***: Ova metoda se poziva kada istekne vremenska kontrola (IVR_TIMER1) a ni jedna poruka nije stigla IVR automatu. Postavlja automat u slobodno stanje i briše podatke u automatu sa kojim je komunicirao.
- ***void T2Expired()***: Ova metoda se poziva kada istekne vremenska kontrola (IVR_TIMER2) a ni jedna poruka nije stigla IVR automatu. Postavlja automat u slobodno stanje i briše podatke u automatu sa kojim je komunicirao.
- ***void UserDigit()***: Ova metoda se poziva nakon što je sa korisničke tastature zadata neka cifra. Proverava da li je zadata cifra definisana i ako jeste pušta poruku i postavlja se u stanje čekanja kraja poruke.
- ***void MessageEnd()***: Ova metoda se poziva nakon što je primljena poruka *vmessage_end*. Ukoliko korisniku treba pružiti uslugu, čeka da se sa korisničke strane zada cifra. Ukoliko se korisniku ne pruža usluga, priprema poruku PLAY_END šalje je CD automatu i prelazi u slobodno stanje.

3.4.3 Klasa CIVRAutomat

U ovoj klasi vrši se inicijalizacija protokola za prenos podataka u realnom vremenu, kao i emitovanje govorne poruke korisniku.

Atributi klase CIVRAutomat:

- **string** *m_audioFileName* - sadrži putanju na disku audio datoteke čiji sadržaj treba reprodukovati;
- **string** *m_videoFileName* - sadrži putanju na disku video datoteke čiji sadržaj treba reprodukovati;
- **FILE*** *m_audioFile* - pokazivač na audio datoteku koji sadrži informacije koje se puštaju korisniku;
- **FILE*** *m_videoFile* - pokazivač na video datoteku koji sadrži informacije koje se puštaju korisniku;
- **IVR_AUDIOHEADER** *m_audioFileHeader* -zaglavlje audio poruke koja se šalje;
- **IVR_VIDEOHEADER** *m_videoFileHeader* - zaglavlje video poruke koja se šalje;
- **DWORD** *m_AudioCounter* - brojač audio paketa;
- **DWORD** *m_VideoCounter* - brojač video paketa;
- **int** *m_portA* - audio mrežni prolaz;
- **int** *m_portV* - video mrežni prolaz;
- **string** *m_IPaddress* - IP adresa udaljenog učesnika u komunikaciji;
- **BOOL** *m_bisConnected* - daje informaciju da li je izvršeno povezivanje sa korisnikom;

Strukture klase CIVRAutomat:

- **IVR_AUDIOHEADER_t:**
  - **char** *code[7]*; -- kod koji označava da se radi o audio IVR poruki;
  - **DWORD** *packetSize*; -- element koji daje informaciju o veličini audio paketa;
  - **DWORD** *repeatOffset*; -- element koji daje informaciju da li će poruka biti ponovljena korisniku;
  - **DWORD** *repeatStart*; -- početak poruke;
  - **DWORD** *repeatEnd*; -- kraj poruke;
- **IVR_VIDEOHEADER_t:**
  - **char** *code[7]*; -- kod koji označava da se radi o video IVR poruki;
  - **DWORD** *packetSize*; -- element koji daje informaciju o veličini video paketa;
  - **DWORD** *repeatOffset*; -- element koji daje informaciju da li će poruka biti ponovljena korisniku;
  - **DWORD** *repeatStart*; -- početak poruke;
  - **DWORD** *repeatEnd*; -- kraj poruke;

Metode klase CIVRAutomat:

- ***BOOL _CheckAudioFile(int packet_size)***: Proverava audio datoteku za zadatu veličinu paketa pre slanja poruke i da li je veličina bafera audio signala odgovarajuća veličini paketa. Ukoliko nije, podešava vrednost bafera u zavisnosti od veličine paketa.
- ***BOOL _CheckVideoFile()***: Proverava video datoteku za zadatu veličinu paketa pre slanja poruke. Ukoliko je video datoteka prazna ili je zadati kod različit od koda video signala ne izvršava se slanje poruke.
- ***int _StartAudioSesion()***: Početak slanja paketa korisniku. Ova metoda vrši inicijalizaciju RTP protokola i otvara vezu za slanje paketa korisniku. Stvaraju se utičnice za slanje paketa, RTP i RTCP, podešavaju se baferi za prijem i slanje, određuje se broj mrežnog prolaza i lokalna adresa i nakon toga se kreiraju niti za prijem paketa (*RTPThread*) i prijem RTCP paketa (*RTCPThread*), kao i dve vremenske kontrole, vremenska kontrola za slanje RTCP paketa (*RTCPTimerProc*) i vremenska kontrola za kontrolu aktivnosti krajnje tačke u komunikaciji (*ControlTimer*). Takođe, metoda pokreće dve vremenske kontrole. U prvoj se vrši slanje, a u drugoj prihvatanje audio paketa.
- ***int StartVideoSesion()***: Početak slanja paketa korisniku. Ova metoda vrši inicijalizaciju RTP protokola i otvara vezu za slanje paketa korisniku. Kreiraju se utičnice za slanje paketa, RTP i RTCP, podešavaju se baferi za prijem i slanje, određuje se broj mrežnog prolaza i lokalna adresa i nakon toga se kreiraju niti za prijem paketa (*RTPThread*) i prijem RTCP paketa (*RTCPThread*), kao i dve vremenske kontrole, vremenska kontrola za slanje RTCP paketa (*RTCPTimerProc*) i vremenska kontrola za kontrolu aktivnosti krajnje tačke u komunikaciji (*ControlTimer*). Takođe metoda pokreće dve vremenske kontrole. U prvoj se vrši slanje, a u drugoj prihvatanje audio paketa.
- ***void _SendAudioData()***: Ova metoda se poziva iz vremenske kontrole (*IVR_audioSendTimerProc*) i vrši slanje RTP paketa periodično svakih 10ms.

Prilikom slanja svakog paketa brojač audio paketa se inkrementira. Kada se pošalje zadnji paket stvara se poruka za oznaku kraja i šalje IVR automatu.

- ***void _SendVideoData()***: Ova metoda se poziva iz vremenske kontrole (*IVR_videoSendTimerProc*) i vrši slanje RTP paketa periodično svakih 10ms. Prilikom slanja svakog paketa brojač video paketa se inkrementira.
- ***void _CollectAudioData()***: Ova metoda se poziva iz vremenske kontrole (*IVR_audioGetTimerProc*) i vrši prihvatanje RTP paketa periodično svakih 10ms. Prihvatanje audio paketa vrši se pozivanjem metode *GetPacket()* klaseRTP, koja vraća paket, njegovu dužinu i redni broj paketa**.**
- ***void _CollectVideoData()***: Ova metoda se poziva iz vremenske kontrole (*IVR_videoGetTimerProc*) i vrši prihvatanje RTP paketa periodično svakih 10ms.
- ***void _StopAudioSesion()***: Zaustavlja slanje audio paketa i raskida vezu sa korisnikom.
- ***void _StopVideoSesion()***:Zaustavlja se slanje video paketa i raskida vezu sa korisnikom.
- ***int Disconnect()***: Ova metoda raskida veze za audio i video komunikaciju, a zatim zatvara audio i video datoteke iz kojih se čita poruka.
- ***int Connect(TimerSystem* timerIVR, TimerSystem* timerRTP, int packet_size)***: Uspostavlja vezu sa korisnikom. Otvara i proverava audio i video datoteku i inicijalizuje brojače audio i video paketa. Aktivira RTP i IVR vremenske kontrole i startuje slanje audio i video poruka korisniku.

3.4.4 Klasa RTP

RTP klasa se koristi za prenos audio i video podataka korisniku od strane IVR automata.

Atributi klase RTP:

- **int** *numcalls* - služi za generisanje slučajnih brojeva;
- **int** *sendport* – mrežni prolaz za slanje;
- **int** *rtpport* – mrežni prolaz za prijem rtp paketa;
- **int** *rtcpport* - rtcp mrežni prolaz za prijem rtcp paketa;
- **int** *maxpacksize* - maksimalna veličina jednog paketa;
- **int** *numactive* - broj krajnjih tačaka koji su aktivni;
- **char*** *RTPPacketBuffer* - bafer u koji se preuzima RTP paket;
- **char*** *RTCPPacketBuffer* - bafer u koji se preuzima RTCP paket;
- **char*** *rtcppacket* - bafer iz kog se šalje RTCP paket;
- **unsigned char** *mediatype* - tip medija(video ili audio);
- **unsigned char** *numberssrc* - broj krajnjih tačaka koji učestvuju u sesiji;
- **unsigned short** *seqnumber* - redni broj paketa za slanje, uvećava se za jedan nakon svakog poslatog paketa;
- **unsigned short** *RTPpacketsnumber* - ukupan broj paketa koji se nalazi u listi;
- **unsigned long** *ssrc* - izvor sinhronizacije;
- **unsigned long** *localIP* - lokalna Ipadresa;

- **unsigned long** *RTCPtime* - vreme koje se dobije svakim novim pozivom RTCP vremenske kontrole;
- **unsigned int** *RTCPtimerID* - identifikator RTCP vremenske kontrole;
- **bool** *socketsopened* - otvorene mrežne utičnice;
- **SOCKET** *sendsocket* – mrežne utičnica za slanje;
- **SOCKET** *rtpsocket* – mrežna utičnica za prijem RTP paketa;
- **SOCKET** *rtcpsocket* – mrežna utičnica za prijem RTCP paketa;
- **DestinationList*** *destlist* - lista u kojoj se nalaze IP adrese destinacija;
- **DestinationList*** *ignorelist* - lista u kojoj se nalaze IP adrese čiji se paketi ne prihvataju;
- **DestinationList*** *acceptlist* - lista u kojoj se nalaze IP adrese čiji se paketi prihvataju;
- **RTPbuffer*** *RTPpacketlist* - pokazivač na prvi rtp paket u listi;
- **HANDLE** *sem* - semafor koji sprečava da se zavrsi aplikacija dok se ne pošalje BYE paket;
- **HANDLE** *EndingSem* - semafor koji sprečava da se završi aplikacija dok se ne završe niti (eng. *Thread*);
- **DestinationList*** *rtcpdestlist* - lista u kojoj se nalaze IP adrese destinacija svih učesnika sesije; ovo je neophodno zbog slanja RTCP paketa jer se oni šalju svim učesnicima sesije;

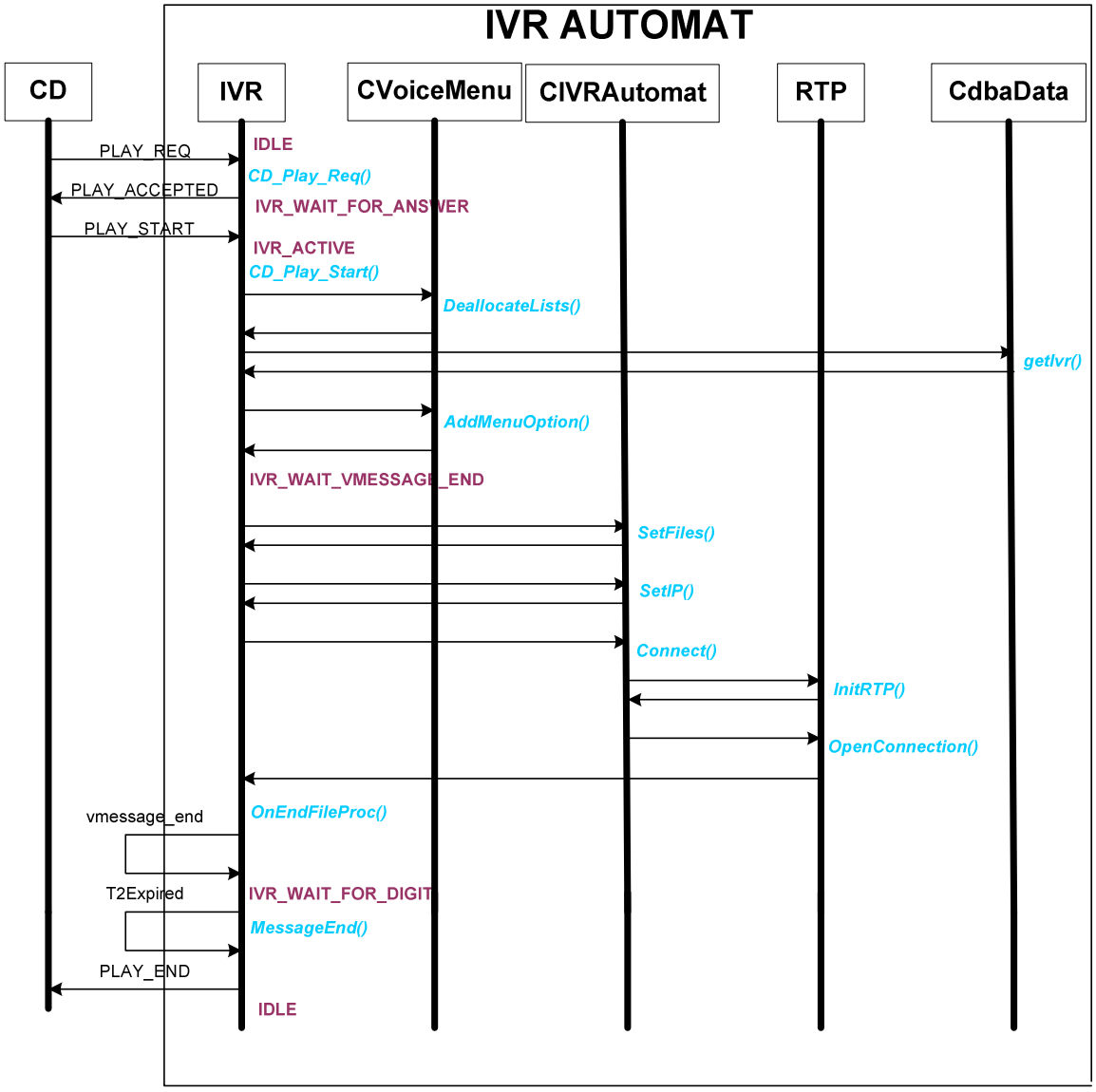
Strukture u klasi RTP:

- Struktura Element je struktura za čuvanje IP adresa destinacija. Polje next je pokazivac na sledeći element u jednostruko spregnutoj listi destinacija
  ***struct Element**
  - **unsigned long** *IPaddress*;
  - **Element*** *next*;
- Struktura List sadrži pokazivače na glavu i rep liste destinacija
  ***struct DestinationList**
  - **Element*** *head*;
  - **Element*** *tail*;
- Struktura koja sadrži podatke koji se odnose na slanje i prijem RTP paketa
  ***struct RTPControler**
  - **unsigned short** *read*; -indikator za čitanje buffera iz bazena buffera;
  - **unsigned long** *packetcount*; -ukupno poslatih paketa;
  - **unsigned long** *bytecount*; -ukupno poslatih bajta;
- ***struct RTPbuffer**
  - **char*** *RTPPacket*; -bafer u koji se smešta rtp paket;
  - **int** *packetlength*; -veličina paketa;
  - **unsigned short** *packetnumber*; -broj paketa;
  - **RTPbuffer*** *next*; -sledeći bafer u listi;

Važnije metode klase RTP:

- ***unsigned long CalcLocalIP()***: Funkcija koja kao rezultat vraća lokalnu IPadresu
- ***UINT ReceiveRTPPacket()***: Funkcija koja prihvata paket i šalje ga dalje na obradu. Ova funkcija predstavlja telo niti.
- ***int ProcessRTPPacket(char* RTPPacket, int length)***: Izdvaja iz paketa samo podatke i slaže ih u red odakle će biti preuzeti na daljnju obradu.
- ***int InitRTP(unsigned char medtype, const char* IP, int rtpID, int maxpcksize)***: Vrši inicjalizaciju RTP protokola za slanje audio i video signala.
- ***int OpenConnection()***: Stvara utičnice za slanje i prijem RTP i RTCP paketa i vrši povezivanje sa korisnikom.
- ***void DestroyConnection()***; Prekida vezu s korisnikom.
- ***bool GetPacket(char* buffer, int& length, int& number)***: Funkcija vraća paket, njegovu dužinu i redni broj.
- ***void AddToList(DestinationList* list, unsigned long newIP)***: Dodaje IP adresu novog korisnika u listu.
- ***bool DeleteFromList(DestinationList* list, unsigned long IP)***: Funkcija koja iz liste briše odredište sa IP adresom koja je prosleđena kao parametar prilikom poziva ove funkcija. Inače ova funkcija se poziva kada se primi BYE paket preko RTCP porta sa IP adrese. Ukoliko se IP adresa ne nalazi u tabeli ova funkcija vraća vrednost FALSE.
- ***int SendRTPPacket(unsigned char* rtppacket, unsigned char* message, unsigned short messagelength, unsigned long time_stamp, bool mark, bool padd, unsigned char paddinglength, bool ext, unsigned char* hdrextension)***: Funkcija SendRTPPacket šalje pakete klijentu. Kao ulaz mora dobiti poruku bez zaglavlja. Kao rezultat vraća kod greške ili 0 ukoliko je uspešno izvršena. Njeni parametri su pokazivač na bafer u koji se smešta rtppacket, pokazivač na bafer u kom se nalazi poruka, ali bez zaglavlja, dužina poruke (*messagelength*), trenutak odabiranja prvog bajta podataka iz tekućeg paketa (*time_stamp*), da li je poslednji paket okvira (eng. *frame*) ili ne (*mark*), ima li zaokruživanja veličine paketa na određeni broj bajta (*padd*) i ako ima koliko bajta dodajemo (*paddinglength*), ima li ekstenzije headera (*ext*), pokazivač na header ekstenziju.
- ***friend DWORD WINAPI RTPThread(LPVOID pParam)***: Programska nit (eng. *thread*) za prijem RTP paketa.
- ***friend void CALLBACK RTCPTimerProc(UINT uID, UINT uMsg, DWORD dwUser, DWORD dw1, DWORD dw2)***: Vremenska kontrola za slanje RTCP paketa.
- ***friend DWORD WINAPI RTCPThread(LPVOID pParam):*** Programska nit (eng. *thread*) za prijem RTCP paketa.

3.5 Redosled akcija prilikom reprodukovanja audio i video poruke korisniku



Slika 3.10. Redosled akcija prilikom reprodukovanja govorne poruke korisniku

Za opis akcija koje se odvijaju prilikom reprodukovanja govorne poruke korisniku korišćen je nešto drugačiji MSC (eng. *Message Sequence Chart*) dijagram, gde je sa

jedne strane prikazana komunikacija između CD i IVR automata dok je sa druge strane prikazano pozivanje metoda klasa koje sačinjavaju IVR automat.

IVR automat se sastoji od sledećih klasa:

- IVR - klasa preko koje IVR server komunicira sa CD automatom.
- CVoiceMenu - klasa koja predstavlja meni IVR servera. Njenom pretragom se dobija poruka koja treba da se emituje korisniku.
- CIVRAutomat - klasa koja komunicira sa RTP-om i priprema datoteku za slanje.
- RTP - klasa koja se koristi za emitovanje poruke korisniku.
- CdbaData - klasa preko koje se iz baze podataka uzimaju skriptovi koji daju informacije o poruci koja treba da se emituje.

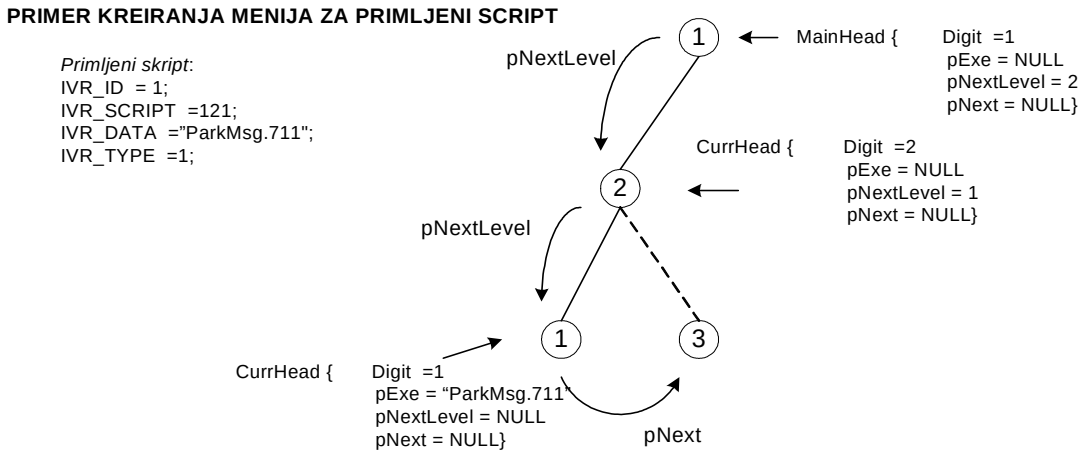
Kada korisnik uspostavi vezu se VCC serverom CD automat šalje IVR automatu poruku ***PLAY_REQ***, kojom proverava da li postoji slobodan automat koji će emitovati poruku korisniku. Po prijemu poruke poziva se metoda ***CD_PlayReq()*** IVR klase koja uzima podatke o automatu koji je poslao poruku, startuje vremensku kontrolu (***TIMER1***), vraća poruku CD automatu da je slobodan da korisniku emituje poruku (***PLAY_ACCEPTED***) i prelazi u stanje čekanja odgovora (***IVR_WAIT_FOR_ANSWER***).

Nakon primljene poruke CD automat šalje IVR automatu poruku ***PLAY_START (IVR_SCRIPT_ID, A_PORT, A_IP)***, kojom traži da se korisniku emituje poruka. Parametar ***IVR_SCRIPT_ID*** daje informaciju o vrsti poruke koju treba emitovati korisniku. Konkretno u ovom slučaju to je pozdravna poruka (***IVR_SCRIPT_ID =1***). Osim informacije o vrsti poruke koju treba emitovati, šalju se informacije o IP adresi i broju mrežnog prolaza **(*A_PORT, A_IP*)** korisnika. Kada primi poruku od CD-a IVR automat poziva metodu ***CD_PlayStart()*** klase IVR kojom započinje proces reprodukovanja poruke korisniku.

U metodi ***CD_PlayStart()*** ostvarena je kompletna funkcionalnost IVR automata koja se može predstaviti u nekoliko celina:

1. Uzima ***IVR_SCRIPT_ID*** koji je prosleđen kao parametar i daje informaciju koju poruku treba emitovati korisniku;
2. Poziva metodu ***UpdateMenu()*** koja pravi meni za prosleđeni skript na sledeći način:
    - Atribut koji čuva informaciju o toma da li korisniku treba pružiti uslugu (***DigitExcepted***) se postavlja na FALSE.
    - Poziva se metoda ***DeallocateLists()*** klase CVoiceMenu koja oslobađa svu dinamičku memoriju zauzetu u toku izvršavanja programa.
    - Poziva se metoda ***getIvr()*** klase CdbaData koja postavlja pokazivač na strukturu Ivr (***ivrStruck***) na početno mesto u IVR tabeli.
    - Proverava prvu cifru u skriptu. Ako je cifra u intervalu 0-9 kreira meni za sve skriptove u bazi podataka čiji je ID jednak identifikatoru skripta koji je prosleđen od strane CD automata kao parametar.
    - Poziva se metoda ***AddMenuOption()*** klase CVoiceMenu koja pravi meni za postojeći skript na sledeći način:
        - kako ova metoda ima promenljiv broj parametara, prvo se vrši njihova inicijalizacija .

- uzima se prvi parametar (pokazivač na String koji sadrži naziv datoteke čiji sadržaj treba da se emituje) i dodaje na početak prve liste.
- Poziva se metoda ***AddInList()*** koja prvo proverava da li je cifra već definisana u tekućoj listi. Ako nije definisana vraća NULL a ako jeste vraća pokazivač na strukturu gde je cifra definisana. Ukoliko vrati NULL, pravi strukturu ***ListNode*** i popunjava je za datu cifru. Ako je tekuća lista prazna, pokazivač na strukturu se dodaje listi pokazivača a ukoliko tekuća lista nije prazna, ulančava se u tekuću listu.
- zatim se ostali argumenti dodaju po hijerarhiji jedan ispod drugog. Na kraju, prilikom dodavanja poslednjeg argumenta u njegovo polje ***pExe*** kopira se pokazivač na string u kom se nalazi informacija o datoteci čiji sadržaj treba da se emituje korisniku a pokazivač se vraća na početak prve liste.



Slika 3.11. Primer stvaranja menija za primljeni skript

3. Određuju se IP adresa automata, broj automata, njegov broj mrežnog prolaza kao i ID službe. Zaustavlja se vremenska kontrola (***IVR_TIMER1***) i u zavisnosti od vrste poruke koja treba da bude emitovana korisniku u stringove ***a***, ***b*** i ***c*** se upisuje putanja na disku gde se nalaze željene poruke.
4. Poziva se metoda ***SetFiles()*** klase CIVRAutomat koja podešava putanje za audio i video datoteku za određeni događaj.
5. Poziva se metoda ***SetIP()*** klase CIVRAutomat koja podešava udaljenu IP adresu, mrežni prolazi za audio i video podatke.
6. Poziva se metoda ***Connect()*** klase CIVRAutomat koja otvara audio i video datoteku pomoću metoda ***CheckAudioFile()*** i ***CheckVideoFile()*** proverava navedene datoteke. Kada se završi provera datoteka inicijalizuju se brojači audio i video paketa i aktiviraju se vremenske kontrole (***RTPTimer***, ***IVRTimer***). Pozivanjem metoda ***StartAudioSesion()*** i ***StartVideoSesion()*** startuje se slanje poruka korisniku na sledeći način:
   - poziva se metoda ***InitRTP()*** klase RTP koja vrši inicijalizaciju RTP-a.

- pozivanjem metode ***OpenConnection()*** klase RTP kreiraju se mrežne utičnice za slanje paketa, za prijem RTP paketa i prijem RTCP paketa, dve vremenske kontrole (za slanje RTCP paketa (***RTCPTimerProc***) i za kontrolu aktivnosti krajnjih tačaka (***ControlTimer***)) i dve programske niti za prihvatanje paketa (***RTPThread*** i ***RTCPThread***).
- Pokreće se vremenska kontrola u kojoj se vrši slanje RTP paketa (***SendRTPPacket()***). Prilikom slanja RTP paketa audio i video brojači se uvećavaju za 1. Kada dođe do kraja datoteke kreira se programska nit ***OnEndFileProc()*** a atribut koji nosi informaciju da je emitovanje završeno **(*endOfWork)*** se postavlja na TRUE čime se završava vremenska kontrola za slanje.

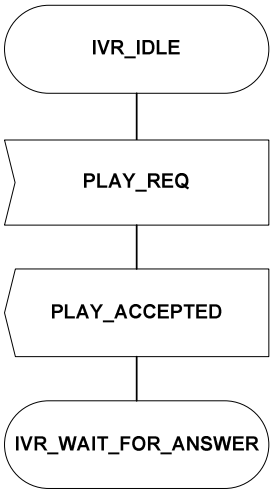
7. IVR server se postavlja u stanje čekanja kraja poruke (***IVR_WAIT_VMESSAGE_END***).

Poruka se emituje korisniku. Kada dodje do kraja poruke IVR automat pravi poruku koju šalje sam sebi (***vmessage_end***) kao znak da je puštanje poruke korisniku završeno. Nakon što primi poruku za kraj, IVR server poziva metodu ***MessageEnd()*** klase IVR koja startuje vremensku kontrolu (***IVR_TIMER2***) i prelazi u stanje čekanja cifre (***IVR_WAIT_FOR_DIGIT***).

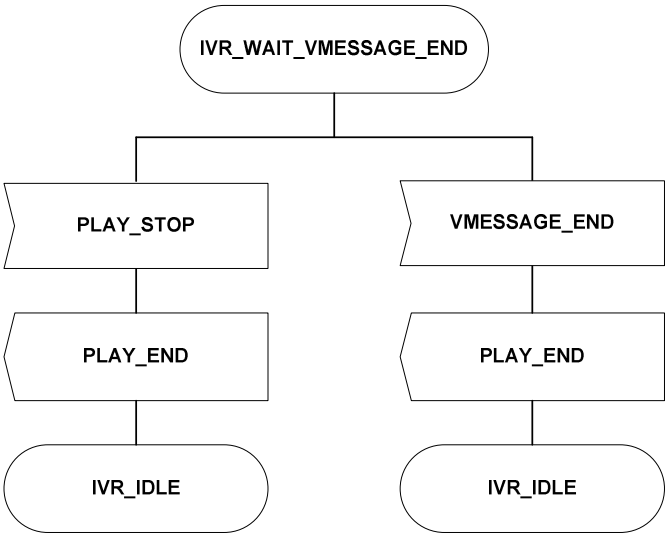
Nakon što primi poruku ***T2Expired*** IVR server zaustavlja vremensku kontrolu (***IVR_TIMER2***), pravi poruku ***PLAY_END*** koju šalje CD automatu i prelazi u stanje *IDLE*.

3.6 SDL Dijagrami

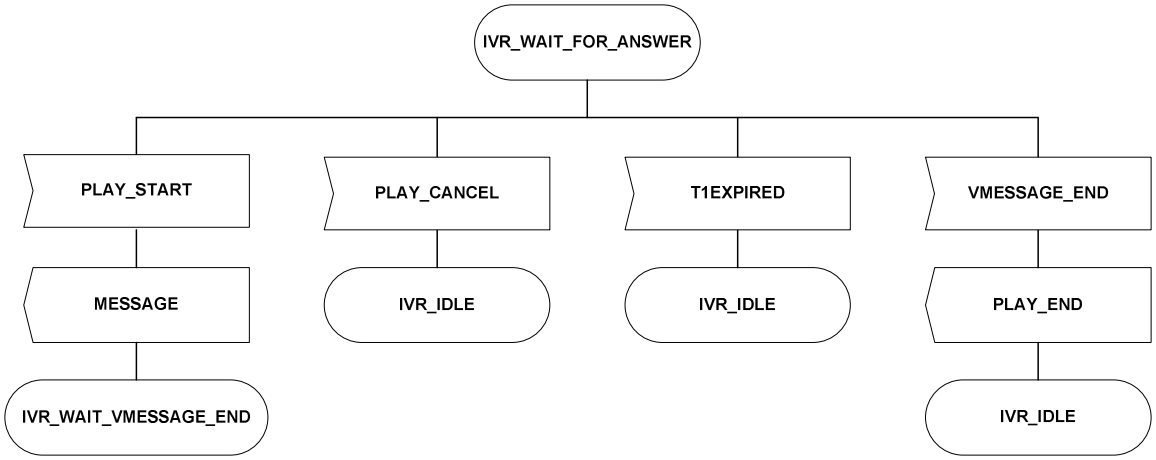
Na sledećim slikama su prikazani SDL dijagrami koji opisuju IVR automat.



Slika 3.12. IVR_IDLE stanje



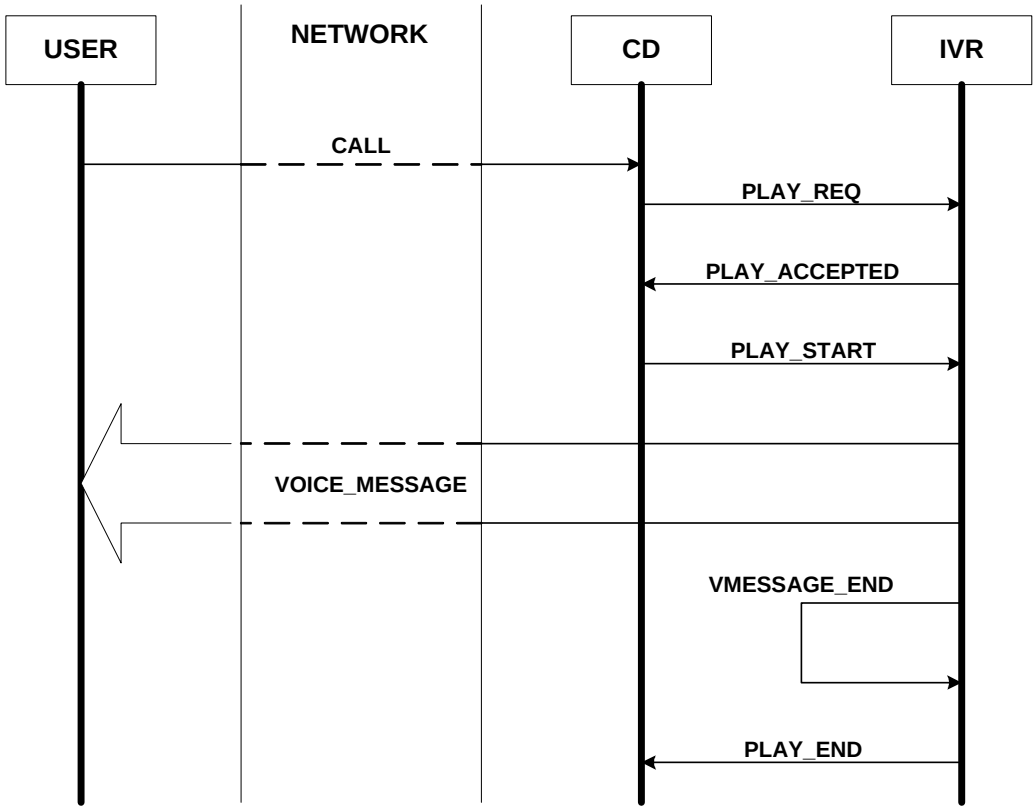
Slika 3.13. IVR_WAIT_VMESSAGE stanje



Slika 3.14. IVR_WAIT_FOR_ANSWER stanje

3.7 MSC Dijagram

Na slici 3.15. prikazan je primer emitovanja pozdravne poruke korisniku kada se povezuje na VCC server.



Slika 3.15. MSC Dijagram emitovanja pozdravne poruke korisniku

Korisnik se povezuje na VCC preko CD automata.

Po prijemu poruke da je korisnik povezan na VCC, CD automat šalje automatu za emitovanje govorne poruke poruku PLAY_REQ, kojom ispituje da li postoji slobodan IVR automat. Ukoliko postoji slobodan IVR automat, IVR automat vraća CD automatu poruku PLAY_ACCEPTED kojom potvrđuje da je spreman da emituje govornu poruku korisniku i prelazi u stanje čekanja odgovora od CD automata.

Nakon prijema poruke od IVR automata, CD automat porukom PLAY_START šalje zahtev za emitovanje pozdravne poruke korisniku. Kao parametri u poruci su navedeni:

- vrsta usluge koju treba da ispuni IVR automat (u ovom slučaju pozdravna poruka)
- IP adresa korisnika
- broj mrežnog prolaza.

Ove podatke IVR automat koristi da bi mogao da adresira korisnika prilikom emitovanja poruke.

Nakon primljene poruke IVR automat emituje glasovnu poruku korisniku i prelazi u stanje čekanja kraja poruke. Kada dođe do kraja poruke IVR automat stvara poruku VMESSAGE_END koju šalje samom sebi, a koja označava kraj emitovanja poruke.

Po završetku emitovanja poruke IVR automat šalje poruku PLAY_END CD automatu kojom ga obaveštava da je korisnik uslužen. Nakon obaveštavanja CD automata o završetku emitovanja poruke korisniku, IVR automat prelazi u stanje slobodan.

4. ISPITIVANJE IVR AUTOMATA

Svi testovi su izvršeni u CppUnit radnom okruženju za testiranje modula programske podrške.

CppUnit je radno okruženje za testiranje modula C++ izvornog koda. Nastao je kao prilagođenje JUnit radnog okruženja C++ programskom jeziku. Princip ispitivanja se zasniva na simulaciji rada jezgra programske podrške (Kernela).

CppUnit radno okruženje se nadograđuje sa dve klase: *TestCase* i *MessageFactory*. U klasi *TestCase* vrši se inicijalizacija i deinicijalizacija sistema i definišu se testovi koji su osmišljeni za ispitivanje automata, dok se sa klasom ExampleMessageFactory simulira rad ostalih automata u sistemu i njihova komunikacija sa automatima koji se ispituju.

Opis formiranja jednog testnog slučaja:

- U klasi *MessageFactory* napravi se poruka i šalje u sanduče automata koji se ispituje. Međutim kako kernel nije aktivan da bi poruka bila prosleđena automatu, potrebno je iz njegovog sandučeta uzeti poruku i direktno je obraditi *Process* metodom automata koji ispitujemo. Na taj način se simulira aktivnost kernela.
- Ispitivanje automata vrši se poređenjem vrednosti dobijene nakon obrade prosleđene poruke sa vrednostima koje se očekuju. Konkretno za slučaj *VCCIvrServera* upoređivana su dobijena stanja i poruke.
- Upoređivanje dobijenih vrednosti se vrši pomoću makroa CPPUNIT_ASSERT_EQUAL (*param1, param2*) koji poredi vrednosti *param1* i *param2*. Ukoliko su parametri indentični test je uspešan, a ako su različiti test je neuspešan i ponašanje ispitivanog automata nije očekivano.

Prilikom ispitivanja IVR automata osmišljeno je 24 testa, koji su pregledno dati i opisani u daljem tekstu. Ispitivani su prelazi stanja i poruke koje vraća IVR automat nakon dobijenih poruka od CD automata (*MessageFactory*). Kako bi u potpunosti ispitali ponašanje IVR automata prosleđivane su i očekivane i neočekivane poruke u pojedinim stanjima u kojima može da se nađe ispitivani automat.

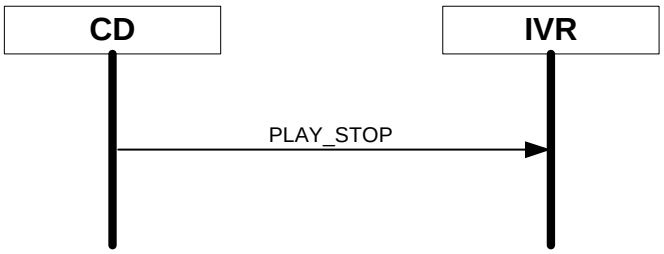
Test1: Nakon primljene poruke PLAY_STOP u IVR_IDLE stanju, IVR server ne menja stanje.

Naziv testa: IVR_TC1

Preduslov: IVR server se nalazi u stanju IVR_IDLE i čeka da mu CD automat pošalje zahtev za emitovanje poruke korisniku.

Test: Ako je IVR server u stanju IVR_IDLE i primi poruku PLAY_STOP od CD automata, treba da ostane u istom stanju, jer data poruka nije definisana za stanje u kom se trenutno nalazi IVR server.

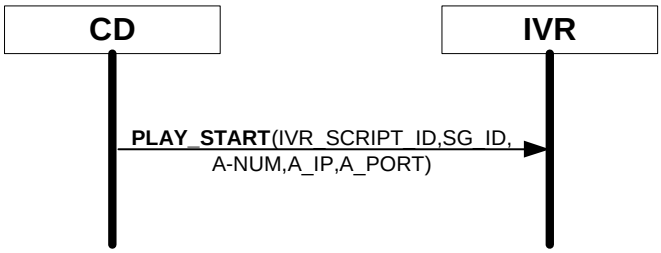
Rezultat: Test je uspešan jer je stanje IVR servera ostalo nepromenjeno nakon prijema zadate poruke.



Slika 4.1. MSC dijagram - testni slučaj IVR_TC1

Test2: Nakon primljene poruke PLAY_START u IVR_IDLE stanju, IVR server ne menja stanje.

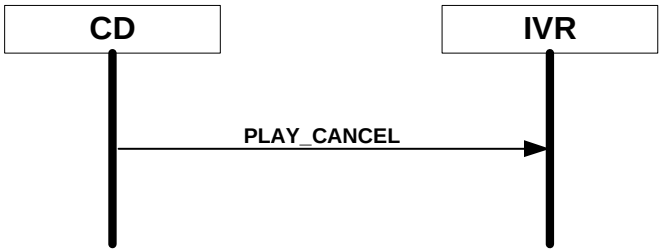
Naziv testa: IVR_TC2
Preduslov: IVR server se nalazi u stanju IVR_IDLE i čeka da mu CD automat pošalje zahtev za emitovanje poruke korisniku.
Test: Ako je IVR server u stanju IVR_IDLE i primi poruku PLAY_START od CD automata, treba da ostane u istom stanju, jer data poruka nije definisana za stanje u kom se trenutno nalazi IVR server.
Rezultat: Test je uspešan jer je stanje IVR servera ostalo nepromenjeno nakon prijema zadate poruke.



Slika 4.2. MSC dijagram - testni slučaj IVR_TC2

Test3: Nakon primljene poruke PLAY_CANCEL u IVR_IDLE stanju, IVR server ne menja stanje.

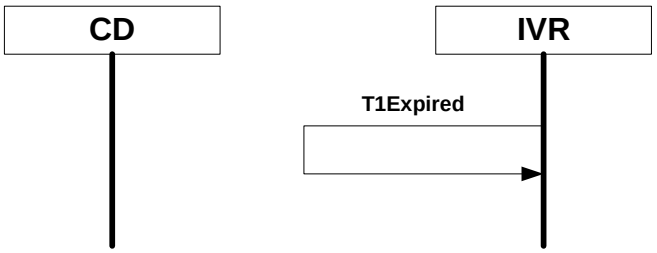
Naziv testa: IVR_TC3
Preduslov: IVR server se nalazi u stanju IVR_IDLE i čeka da mu CD automat pošalje zahtev za emitovanje poruke korisniku.
Test: Ako je IVR server u stanju IVR_IDLE i primi poruku PLAY_CANCEL od CD automata, treba da ostane u istom stanju, jer data poruka nije definisana za stanje u kom se trenutno nalazi IVR server.
Rezultat: Test je uspešan jer je stanje IVR servera ostalo nepromenjeno nakon prijema zadate poruke.



Slika 4.3. MSC dijagram - testni slučaj IVR_TC3

Test4: IVR server u IVR_IDLE stanju dobija poruku T1Expired i njegovo stanje ostaje nepromenjeno.

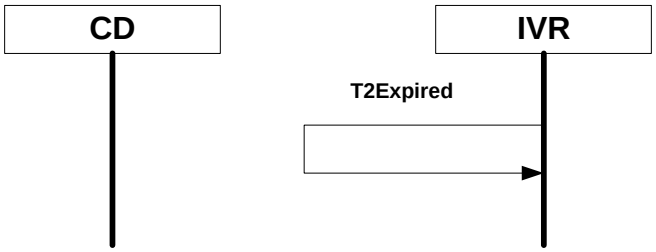
Naziv testa: IVR_TC4
Preduslov: IVR server se nalazi u stanju IVR_IDLE i čeka da mu CD automat pošalje zahtev za emitovanje poruke korisniku.
Test: Ako je IVR server u stanju IVR_IDLE i primi poruku IvrT1Expired, treba da ostane u istom stanju, jer data poruka nije definisana za stanje u kom se trenutno nalazi IVR server.
Rezultat: Test je uspešan jer je stanje IVR servera ostalo nepromenjeno nakon prijema zadate poruke.



Slika 4.4. MSC dijagram - testni slučaj IVR_TC4

Test5: IVR server u IVR_IDLE stanju dobija poruku T2Expired i njegovo stanje ostaje nepromenjeno.

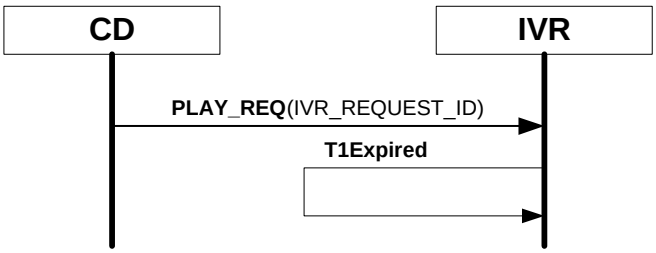
Naziv testa: IVR_TC5
Preduslov: IVR server se nalazi u stanju IVR_IDLE i čeka da mu CD automat pošalje zahtev za emitovanje poruke korisniku.
Test: Ako je IVR server u stanju IVR_IDLE i primi poruku IvrT2Expired, treba da ostane u istom stanju,jer data poruka nije definisana za stanje u kom se trenutno nalazi IVR server.
Rezultat: Test je uspešan jer je stanje IVR servera ostalo nepromenjeno nakon prijema zadate poruke.



Slika 4.5. MSC dijagram - testni slučaj IVR_TC5

Test6: IVR server koji se nalazi u stanju čekanja odgovora, nakon isteka vremenske kontrole prelazi u stanje IVR_IDLE.

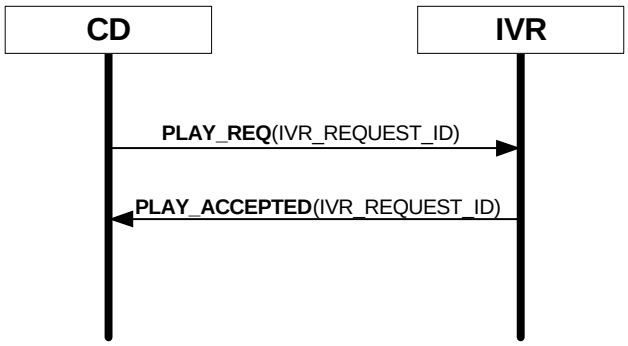
Naziv testa: IVR_TC6
Preduslov: IVR server se nalazi u stanju čekanja odgovora od CD automata (IVR_WAIT_FOR_ANSWER).
Test: IVR server je u stanju IVR_WAIT_FOR_ANSWER, međutim njemu ne stiže nikakva poruka. Nakon isteka vremenske kontrole IVR server treba da pređe u stanje IVR_IDLE.
Rezultat: Test je uspešan jer je nakon isteka tajmera IVR server prešao u očekivano stanje.



Slika 4.6. MSC dijagram - testni slučaj IVR_TC6

Test7: Zahtev za pružanje usluge korisniku i odgovor IVR servera.

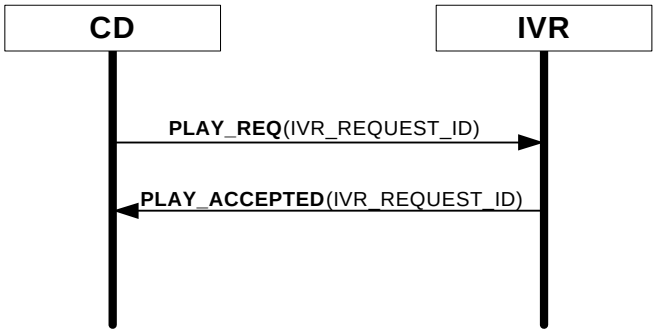
Naziv testa: IVR_TC7
Preduslov: IVR server se nalazi u stanju IVR_IDLE i čeka da mu CD automat pošalje zahtev za emitovanje poruke korisniku.
Test: Kada korisnik uspostavi vezu sa VCC serverom CD automat šalje ka IVR serveru poruku PLAY_REQ sa parametrom IVR_REQUEST_ID.Nakon što prihvati poruku, IVR server treba da je obradi i vrati poruku PLAY_ACCEPTED automatu od koga je dobijen zahtev za puštanje poruke korisniku.
Rezultat: Test je uspešan i IVR server je vratio očekivanu poruku.



Slika 4.7. MSC dijagram - testni slučaj IVR_TC7

Test8: Prelazak u sledeće stanje nakon zahteva za emitovanje poruke korisniku.

Naziv testa: IVR_TC8
Preduslov: IVR server se nalazi u stanju IVR_IDLE i čeka da mu CD automat pošalje zahtev za emitovanje poruke korisniku.
Test: Kada korisnik uspostavi vezu sa VCC serverom CD automat šalje ka IVR serveru poruku PLAY_REQ sa parametrom IVR_REQUEST_ID.Kada prihvati poruku IVR server je obradi, vraća poruku PLAY_ACCEPTED automatu od koga je dobijen zahtev za puštanje poruke korisniku i treba da pređe u stanje čekanja odgovora od CD automata (IVR_WAIT_FOR_ANSWER).
Rezultat: Test je uspešan jer je IVR server nakon vraćene poruke IVR server je prešao u očekivano stanje.



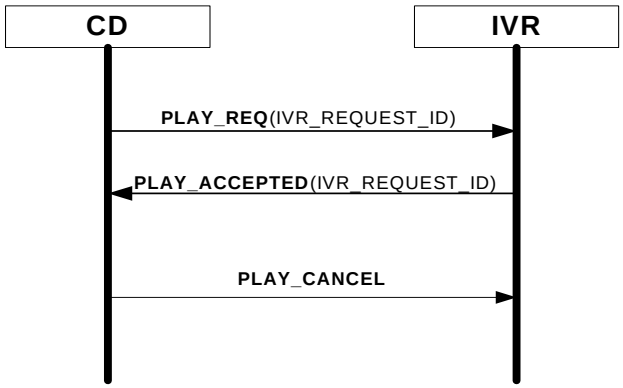
Slika 4.8. MSC dijagram - testni slučaj IVR_TC8

Test9: U stanju IVR_WAIT_FOR_ANSWER,od CD automata dobija poruku PLAY_CANCEL i prelazi u stanje IVR_IDLE.

Naziv testa: IVR_TC9
Preduslov: IVR server se nalazi u stanju čekanja odgovora od CD automata (IVR_WAIT_FOR_ANSWER).

Test: Ukoliko u datom stanju IVR serveru od CD automata stigne poruka PLAY_CANCEL, IVR server treba da pređe u stanje IVR_IDLE.

Rezultat: Test je uspešan jer je,nakon što je primio poruku od CD automata,IVR server prešao u očekivano stanje.



Slika 4.9. MSC dijagram - testni slučaj IVR_TC9

Test10: U stanju IVR_WAIT_FOR_ANSWER, primi poruku PLAY_STOP i ne menja stanje.

Naziv testa: IVR_TC10

Preduslov: IVR server se nalazi u stanju čekanja odgovora od CD automata (IVR_WAIT_FOR_ANSWER).

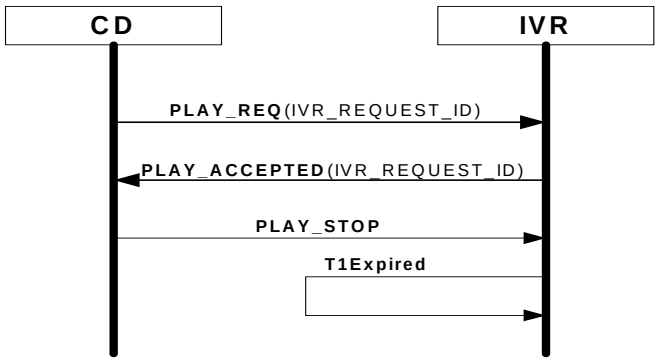
Test: Ukoliko u datom stanju IVR serveru od CD automata stigne poruka PLAY_STOP, IVR server treba da ostane u istom stanju,jer data poruka nije definisana za stanje u kom se trenutno nalazi IVR server.

Rezultat: Test je uspešan jer je stanje IVR servera ostalo nepromenjeno nakon prijema zadate poruke.

Slika 4.10. MSC dijagram - testni slučaj IVR_TC10

Test11: U stanju IVR_WAIT_FOR_ANSWER, primi poruku PLAY_STOP i ne menja stanje. Nakon isteka vremenske kontrole prelazi u stanje IVR_IDLE.

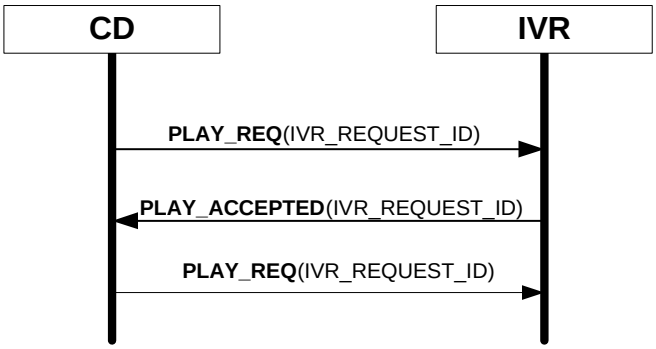
Naziv testa: IVR_TC11
Preduslov: IVR server se nalazi u stanju čekanja odgovora od CD automata (IVR_WAIT_FOR_ANSWER).
Test: IVR serveru u stanju IVR_WAIT_FOR_ANSWER, od CD automata stigne poruka PLAY_STOP. Kako ova poruka nije definisana za stanje u kome se trenutno nalazi, IVR server ostaje u istom stanju. Nakon što istekne vremenska kontrola (IVR_TIMER1) IVR server treba da pređe u stanje IVR_IDLE.
Rezultat: Test je uspešan jer je nakon isteka tajmera IVR server prešao u očekivano stanje.



Slika 4.11. MSC dijagram - testni slučaj IVR_TC11

Test12: U stanju čekanja odgovora, primi poruku PLAY_REQ i ne menja stanje.

Naziv testa: IVR_TC12
Preduslov: IVR server se nalazi u stanju čekanja odgovora od CD automata (IVR_WAIT_FOR_ANSWER).
Test: Ukoliko u datom stanju IVR serveru od CD automata stigne poruka PLAY_REQ sa parametrom IVR_REQUEST_ID, IVR server treba da ostane u istom stanju,jer data poruka nije definisana za stanje u kom se trenutno nalazi IVR server.
Rezultat: Test je uspešan jer je stanje IVR servera ostalo nepromenjeno nakon prijema zadate poruke.



Slika 4.12. MSC dijagram - testni slučaj IVR_TC12

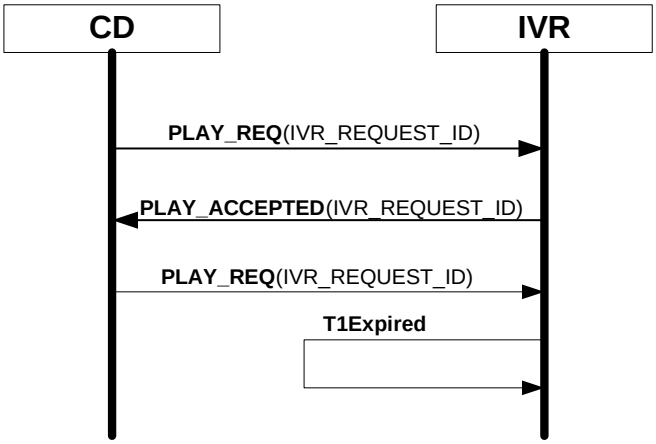
<u>Test13:</u> U stanju čekanja odgovora,primi poruku PLAY_REQ i ne menja stanje. Nakon isteka tajmera prelazi u stanje IDLE.

Naziv testa: IVR_TC13

Preduslov: IVR server se nalazi u stanju čekanja odgovora od CD automata (IVR_WAIT_FOR_ANSWER).

Test: IVR serveru u stanju IVR_WAIT_FOR_ANSWER, od CD automata stigne poruka PLAY_REQ sa parametrom IVR_REQUEST_ID. Kako ova poruka nije definisana za stanje u kome se trenutno nalazi, IVR server ostaje u istom stanju. Nakon što istekne tajmer (IVR_TIMER1) IVR server treba da pređe u stanje IVR_IDLE.

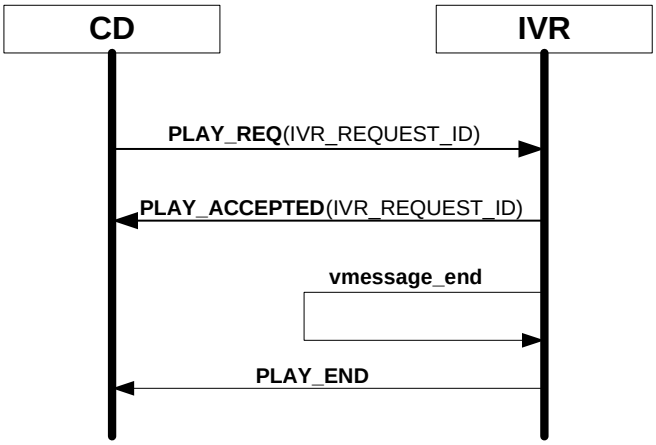
Rezultat: Test je uspešan jer je nakon isteka tajmera IVR server prešao u očekivano stanje.



Slika 4.13. MSC dijagram - testni slučaj IVR_TC13

Test14: U stanju čekanja odgovora, dobija poruku vmessage_end, kreira i šalje CD automatu poruku PLAY_END.

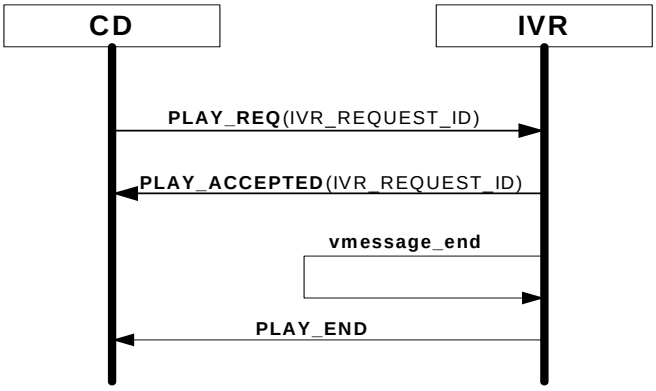
Naziv testa: IVR_TC14
Preduslov: IVR server se nalazi u stanju čekanja odgovora od CD automata (IVR_WAIT_FOR_ANSWER).
Test: IVR serveru u stanju IVR_WAIT_FOR_ANSWER,od stigne poruka vmessage_end. Nakon prijema ove poruke, IVR server, reba da kreira poruku PLAY_END i da je pošalje CD automatu.
Rezultat: Test je uspešan jer je nakon primljene poruke kreirana i poslata CD automatu očekivana poruka.



Slika 4.14. MSC dijagram - testni slučaj IVR_TC14

Test15: Kada u stanju čekanja odgovora dobije poruku vmessage_end, prelazi u stanje IVR_IDLE.

Naziv testa: IVR_TC15
Preduslov: IVR server se nalazi u stanju čekanja odgovora od CD automata (IVR_WAIT_FOR_ANSWER).
Test: IVR serveru u stanju IVR_WAIT_FOR_ANSWER, stigne poruka vmessage_end. Nakon prijema ove poruke, IVR server, treba da pređe u stanje IVR_IDLE.
Rezultat: Test je uspešan jer je nakon primljene poruke IVR server je prešao u očekivano stanje.



Slika 4.15. MSC dijagram - testni slučaj IVR_TC15

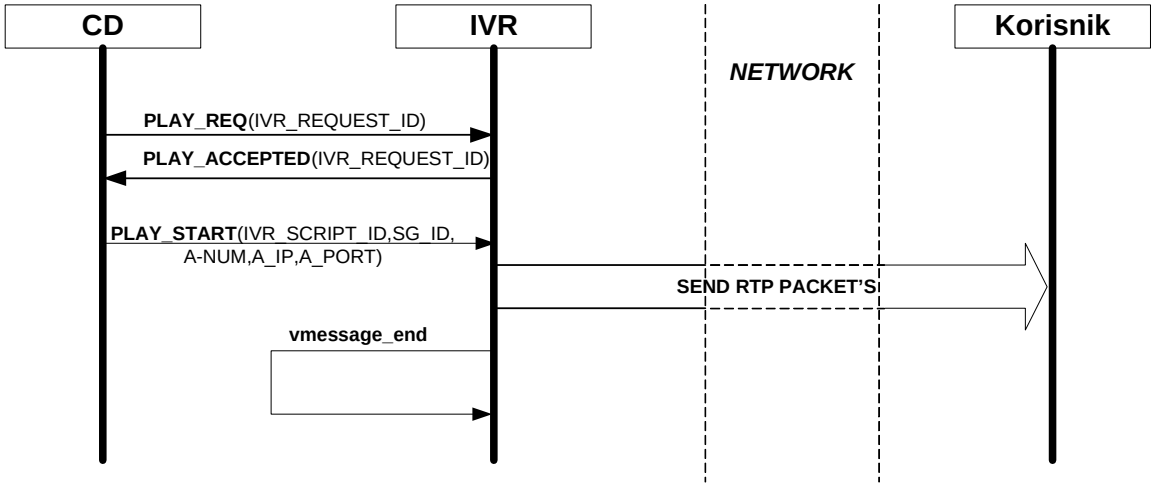
Test16: Emitovanje poruke korisniku i kreiranje poruke za kraj.

Naziv testa: IVR_TC16

Preduslov: IVR server se nalazi u stanju čekanja odgovora od CD automata (IVR_WAIT_FOR_ANSWER).

Test: Ukoliko u datom stanju IVR serveru od CD automata stigne poruka PLAY_START, sa parametrima koji nose informaciju o vrsti poruke koja treba da se emituje korisniku (IVR_SCRIPT_ID), identifikacionom broju službe VCC-a (SG_ID), broju automata od kog je stigla poruka (A_NUM), adresi korisnika (A_IP) i broju mrežnog prolaza preko koga se emituje poruka (A_PORT), IVR server treba da emituje poruku korisniku. Kada dođe do kraja poruke potrebno je da napravi poruku vmessage_end koju šalje u samom sebi, koja označava kraj poruke.

Rezultat: Test je uspešan i kreirana je očekivana poruka, koju je IVR server vratio u svoje sanduče. Takođe pomoću programa "Ethereal" na korisničkoj strani registrovan je prijem RTP paketa, poslatih od strane IVR servera.



Slika 4.16. MSC dijagram - testni slučaj IVR_TC16

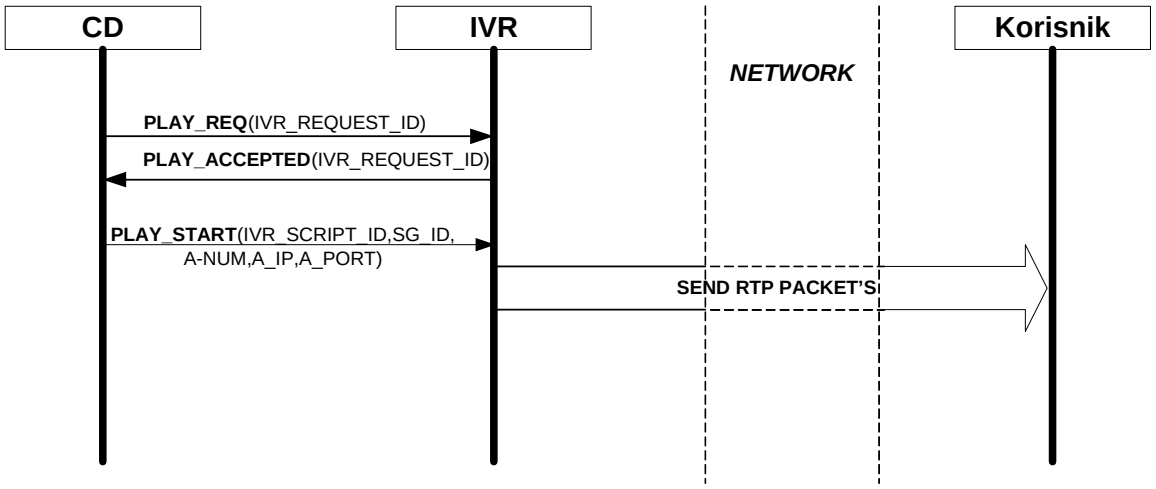
Test17: Prelazak u sledeće stanje nakon emitovanja poruke korisniku.

Naziv testa: IVR_TC17

Preduslov: IVR server se nalazi u stanju čekanja odgovora od CD automata (IVR_WAIT_FOR_ANSWER).

Test: Ukoliko u datom stanju IVR serveru od CD automata stigne poruka PLAY_START, sa parametrima koji nose informaciju o vrsti poruke koja treba da se emituje korisniku (IVR_SCRIPT_ID), identifikacionom broju službe VCC-a (SG_ID), broju automata od kog je stigla poruka (A_NUM), adresi korisnika (A_IP) i broju mrežnog prolaza preko koga se emituje poruka (A_PORT), IVR server počinje emitovanje poruke korisniku i treba da pređe u stanje čekanja kraja poruke (IVR_VMESSAGE_END).

Rezultat: Test je uspešan jer je nakon početka emitovanja poruke korisniku IVR server prešao u očekivano stanje.



Slika 4.17. MSC dijagram - testni slučaj IVR_TC17

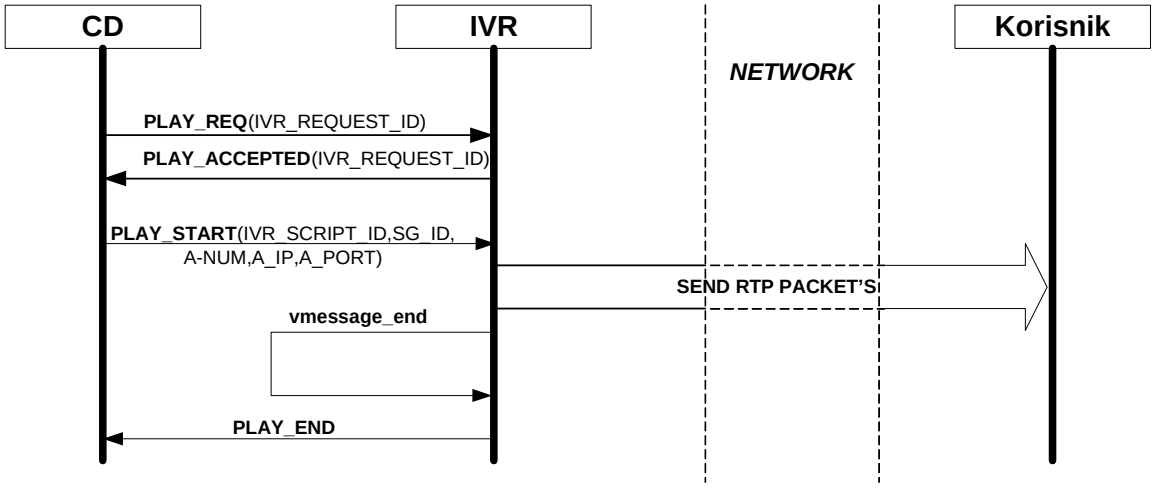
Test18: Kreiranje i slanje poruke koja obaveštava CD automat da je korisnik uslužen.

Naziv testa: IVR_TC18

Preduslov: IVR server se nalazi u stanju čekanja kraja poruke koja se emituje korisniku (IVR_WAIT_VMESSAGE_END).

Test: Ako se IVR server nalazi u stanju čekanja poruke koja se emituje korisniku i stigne mu poruka vmessage_end, koja označava da je završeno emitovanje poruke, treba da kreira poruku PLAY_END koju šalje CD automatu, da ga obavesti da je korisnik uslužen.

Rezultat: Test je uspešan jer je nakon primljene poruke da je emitovanje poruke korisniku završeno,kreirana i poslata očekivana poruka.



Slika 4.18. MSC dijagram - testni slučaj IVR_TC18

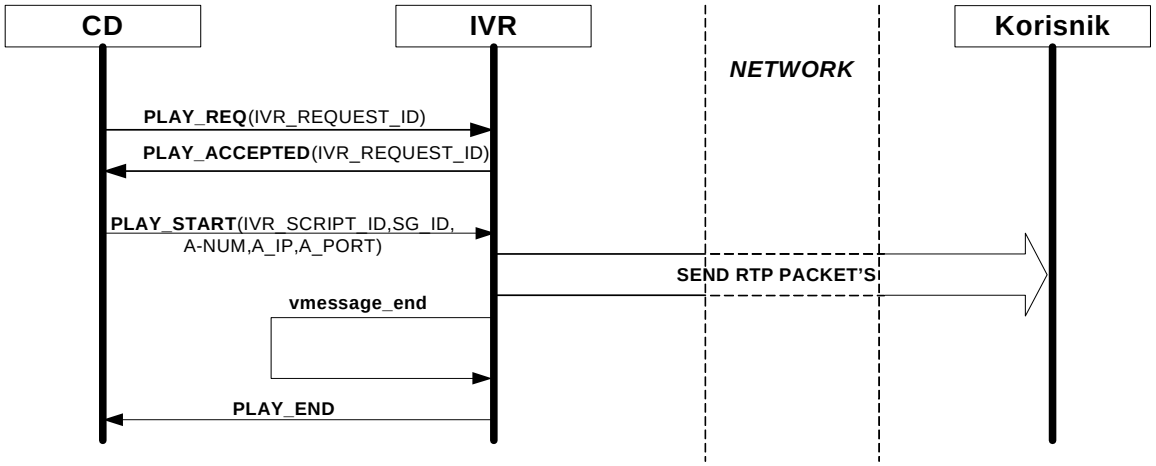
Test19: Prelazak u stanje IVR_IDLE nakon primljene poruke vmessage_end.

Naziv testa: IVR_TC19

Preduslov: IVR server se nalazi u stanju čekanja kraja poruke koja se emituje korisniku (IVR_WAIT_VMESSAGE_END).

Test: IVR server nakon što primi poruku vmessage_end, koja označava da je završeno emitovanje poruke i kreira poruku PLAY_END koju šalje CD automatu, da ga obavesti da je korisnik uslužen, treba da pređe u stanje IVR_IDLE.

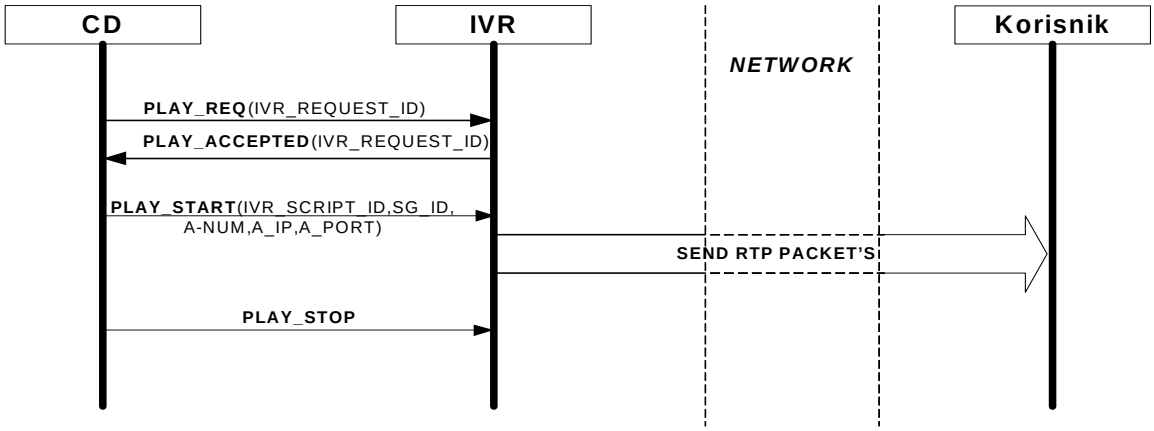
Rezultat: Test je uspešan jer je nakon primljene poruke, da je emitovanje poruke korisniku završeno i kreiranja poruke koja je poslata CD automatu, IVR server prešao u očekivano stanje.



Slika 4.19. MSC dijagram - testni slučaj IVR_TC19

Test20: Za vreme emitovanje poruke korisniku, IVR server dobija poruku PLAY_STOP i prelazi u stanje IVR_IDLE.

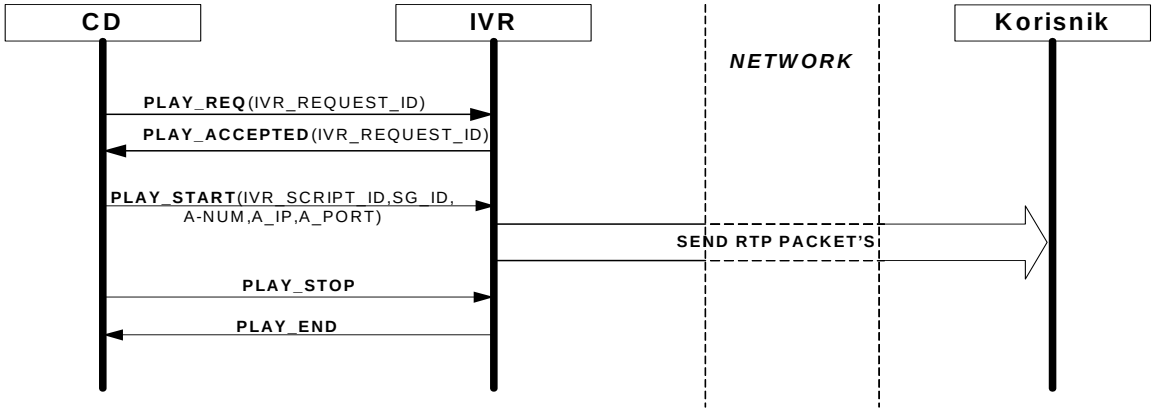
Naziv testa: IVR_TC20
Preduslov: IVR server se nalazi u stanju čekanja kraja poruke koja se emituje korisniku (IVR_WAIT_VMESSAGE_END).
Test: Ako se IVR server nalazi u stanju čekanja kraja poruke koja se pušta korisniku i primi poruku PLAY_STOP od CD automata,treba da pređe u stanje IVR_IDLE.
Rezultat: Test je uspešan jer je nakon primljene poruke IVR server prešao u očekivano stanje.



Slika 4.20. MSC dijagram - testni slučaj IVR_TC20

Test21: Za vreme emitovanje poruke korisniku, IVR server dobija poruku PLAY_STOP i kreira i šalje CD automatu poruku PLAY_END.

Naziv testa: IVR_TC21
Preduslov: IVR server se nalazi u stanju čekanja kraja poruke koja se emituje korisniku (IVR_WAIT_VMESSAGE_END).
Test: Ako se IVR server nalazi u stanju čekanja kraja poruke koja se pušta korisniku i primi poruku PLAY_STOP od CD automata, treba da kreira poruku PLAY_END i pošalje je CD automatu.
Rezultat: Test je uspešan jer je nakon primljene poruke kreirao očekivanu poruku i poslao je CD automatu.



Slika 4.21. MSC dijagram - testni slučaj IVR_TC21

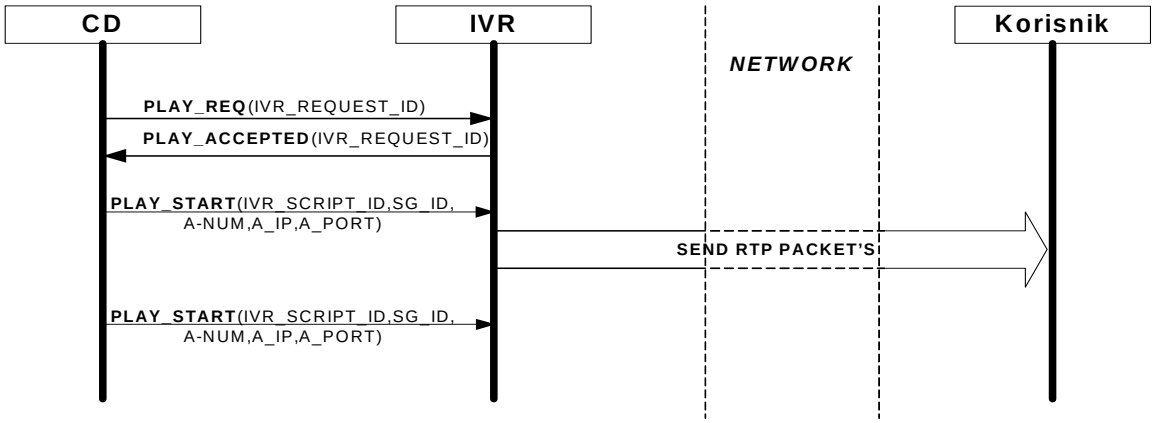
Test22 – u toku emitovanja poruke korisniku,dobija poruku PLAY_START i stanje IVR servera ostaje ne promenjeno.

Naziv testa: IVR_TC22

Preduslov: IVR server se nalazi u stanju čekanja kraja poruke koja se emituje korisniku (IVR_WAIT_VMESSAGE_END).

Test: Kada se IVR server nalazi u stanju čekanja kraja poruke koja se pušta korisniku i primi poruku PLAY_START od CD automata, kako ova poruka nije definisana za IVR server koji se nalazi u datom stanju, njegovo stanje treba da ostane nepromenjeno.

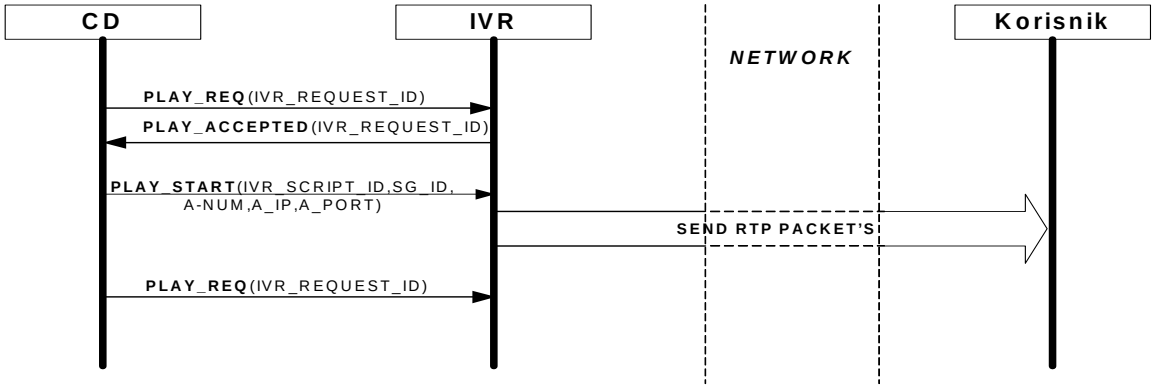
Rezultat: Test je uspešan jer je nakon primljene poruke stanje IVR servera ostalo nepromenjeno.



Slika 4.22. MSC dijagram - testni slučaj IVR_TC22

Test23: U toku emitovanja poruke korisniku dobija poruku PLAY_REQ i stanje IVR servera ostaje ne promenjeno.

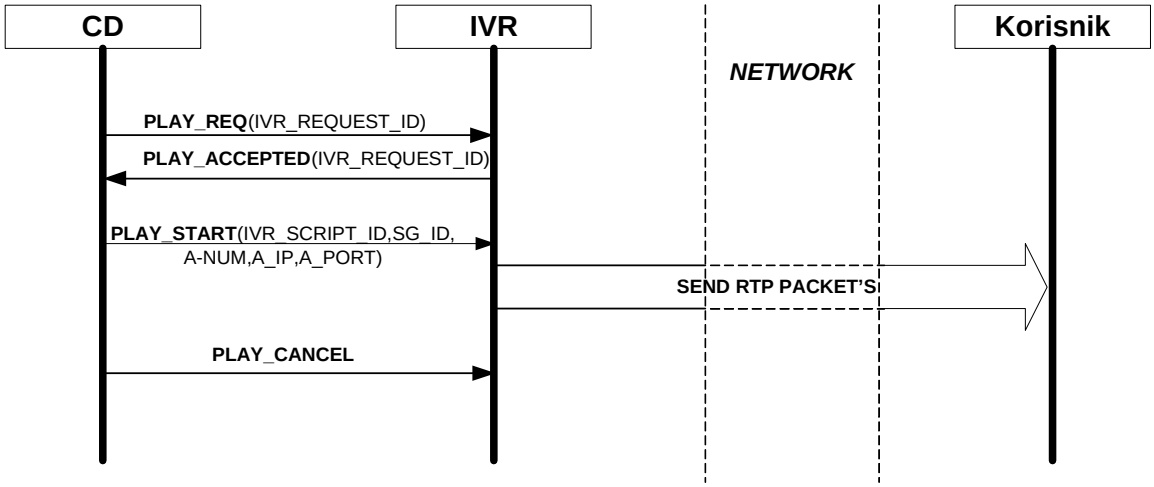
Naziv testa: IVR_TC23
Preduslov: IVR server se nalazi u stanju čekanja kraja poruke koja se emituje korisniku (IVR_WAIT_VMESSAGE_END).
Test: Kada se IVR server nalazi u stanju čekanja kraja poruke koja se pušta korisniku i primi poruku PLAY_REQ od CD automata, kako ova poruka nije definisana za IVR server koji se nalazi u datom stanju, njegovo stanje treba da ostane nepromenjeno.
Rezultat: Test je uspešan jer je nakon primljene poruke stanje IVR servera ostalo nepromenjeno.



Slika 4.23. MSC dijagram - testni slučaj IVR_TC23

Test24: U toku emitovanja poruke korisniku dobija poruku PLAY_CANCEL i stanje IVR servera ostaje ne promenjeno.

Naziv testa: IVR_TC24
Preduslov: IVR server se nalazi u stanju čekanja kraja poruke koja se emituje korisniku (IVR_WAIT_VMESSAGE_END).
Test: Kada se IVR server nalazi u stanju čekanja kraja poruke koja se pušta korisniku i primi poruku PLAY_CANCEL od CD automata, kako ova poruka nije definisana za IVR server koji se nalazi u datom stanju, njegovo stanje treba da ostane nepromenjeno.
Rezultat: Test je uspešan jer je nakon primljene poruke stanje IVR servera ostalo nepromenjeno.



Slika 4.24. MSC dijagram - testni slučaj IVR_TC24

5. ZAKLJUČAK

Osnovni cilj ovog rada je izdvajanje IVR komponente virtualnog centra za obradu poziva iz VCC serverske aplikacije na Windows operativnom sistemu i realizacija novog IVR servera koji će sadržati izdvojene komponente, kao i upoznavanje sa virtuelnim centrom za obradu poziva.

U toku rada sprovedena je detaljna analiza svih elemenata IVR komponente u cilju jednostavnije buduće nadogradnje, a poseban akcenat je stavljen na analizu mogućnosti elementa za rad sa glasovnim menijima. Na osnovu znanja o virtualnom centru za obradu poziva i zaključaka donetih na osnovu sprovedene analize IVR komponente izvršeno je uspešno izdvajanje potrebnih automata. Novi IVR server je uspešno postavljen, podešen i pokrenut u VCC okruženju.

Nakon izvršenog ispitivanja IVR automata, sprovedenog u CppUnit okruženju za ispitivanje, i postavljanja kompletnog IVR servera u adekvatno okruženje može se zaključiti da je postavljeni problem realizacije IVR servera uspešno rešen. Očuvana je početna funkcionalnost sistema, VCC server je oslobođen nepotrebne složenosti i stvorena je mogućnost za bolje povezivanje ostalih komponenti VCC sistema sa IVR komponentom.

S obzirom na veliku složenost VCC sistema postoji i velika mogućnost u pogledu unapređenja njegove funkcionalnosti pa tako i samog IVR servera. Funkcije koje je moguće u daljem radu implementirati radi potpunije funkcionalnosti komunikacije sa korisnikom su:

- funkcija prepoznavanja govora koja se može dodati IVR automatu i
- realizacija funkcije govornog menija.

Za sada je razmatran slučaj kada se podrazumeva da je operater čovek. U daljoj implementaciji trebalo bi razmotriti mogućnost da IVR automat u komunikaciji sa bazom podataka uspešno realizuje neke funkcije koje sada obavlja operater, čime bi se povećala ekonomičnost virtualnog centra za obradu poziva.

6. LITERATURA

[1] Ivan Velikić – Jedno rešenje sistemskih podloga programske podrške centra za distribuciju poziva , magistarski rad, Fakultet tehničkih nauka, Novi Sad 2001.

[2] Milan Savić – Jedno rešenje CTI servera u virtualnom centru za obradu poziva, diplomski rad, Fakultet tehničkih nauka, Novi Sad 2001.

[3] Branislav Zuković- Jedno rešenje sinhronizacije potiskivača odjeka u virtualnom sistemu za obradu poziva, diplomski rad, Fakultet tehničkih nauka, Novi Sad 2006.

[4] Augie Hansen – Potpuni vodič za programski jezik C, Mikro knjiga Beograd 1991.

[5] Laslo Kraus - Programski jezik C++ sa rešenim zadacima, Mikro knjiga Beograd 1994.

[6] MSDN Library Visual Studio 6.0

[7] ***http://en.wikipedia.org/wiki/Interactive_voice_response*** - Emitovanje govornih poruka

[8] ***http://searchcrm.techtarget.com/sDefinition/0,,sid11_gci848284,00.html*** – Virtualni centar za obradu poziva

[9] ***http://www.krt.neobee.net*** – Katedra za računarsku tehniku i računarske komunikacije, skripte za predavanja, Međuračunarske komunikacije i računarske mreže II, VCC, RTP i RTCP

[10] ***http://en.wikipedia.org/wiki/Dynamic_link_library*** - Datoteke sa dinamičkim povezivanjem

[11] ***http://www.telfor.org.yu/telfor2005/radovi/TM-2.12.pdf*** - Jedan pristup integracije virtualnog centra za obradu poziva u sistem VoIP telefonije,TELFOR 2005 Beograd

[12] ***http://users.teol.net/~mvlado/sockets/*** - Beej-ov vodič za mrežno programiranje, Korišćenje Internet socket-a, Brian “Beej” Hall

[13] ***http://cppunit.sourceforge.net/doc/lastest/cppunit_cookbook.html*** - CppUnit - The Unit Testing Library