Santeri Nousiainen

# Interaktiivisen puhelinvalikkosovelluksen suunnittelu ja toteutus tietoliikennealan vritvkselle.

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Tietotekniikka

Insinöörityö

2.1.2015

Metropolia

| | |
|---|---|
| Tekijä<br>Otsikko<br><br>Sivumäärä<br>Aika | Santeri Nousiainen<br>Interaktiivisen puhelinvalikkosovelluksen suunnittelu ja toteutus tietoliikennealan yritykselle.<br><br>39 sivua + 1 liite<br>2.1.2015 |
| Tutkinto | insinööri (AMK) |
| Koulutusohjelma | Tietotekniikka |
| Suuntautumisvaihtoehto | Ohjelmistokehitys |
| Ohjaajat | Järjestelmäasiantuntija Pekka Poutanen<br>Yliopettaja Tarmo Anttalainen |

Tässä insinöörityössä toteutettiin interaktiivinen puhelinvalikko sovellus softswitch-järjestelmälle. Sovelluksen softswitch-alustana toimii ohjelma nimeltä FreeSwitch. Insinöörityön tavoitteena oli korvata AinaComin nykyinen interaktiivinen puhelinvalikkojärjestelmä.

Sovelluksen ohjelmointi tehtiin Python-ohjelmointikielellä ja ohjelma kirjoitettiin käyttämällä Notepad++-sovellusta. Ohjelman kääntäjänä toimi FreeSwitchin oma Python-kääntäjä. Versionhallinnassa ja varmuuskopioiden tekemisessä käytettiin GitHub-versionhallinta-järjestelmää. Ohjelmoinnin lisäksi työssä kerrotaan yleisesti puhelinvalikoista ja niiden käytöstä sekä FreeSwitch-alustasta.

Työn tuloksena on toimiva sovellus, joka on parhaillaan testauksen alla. Sovellukselle on määritetty kymmenen puhelinnumeroa, joihin voi soittaa. Puhelinnumeroissa on erilaisia IVR-palveluita esimerkiksi puhelinvalikko, tiedotteen toisto ja soitonsiirto.

| | |
|---|---|
| Avainsanat | IVR, sovellus, softswitch, Python |

| | |
|---|---|
| Author(s)<br>Title<br><br>Number of Pages<br>Date | Santeri Nousiainen<br>Designing and making of interactive voice response software for a ICT-Company<br><br>39 pages + 1 appendice<br>2 January 2015 |
| Degree | Bachelor of Engineering |
| Degree Programme | Information Technology |
| Specialisation option | Software development |
| Instructor(s) | Pekka Poutanen, System Specialist<br>Tarmo Anttalainen, Senior Lecturer |

The goal of this thesis is to describe how an interactive voice response system was developed for the Softswitch platform. The Softswitch platform described in this thesis is called FreeSwitch. The main goal of the thesis is to document how the Ainacom's current interactive voice response solution was replaced with newer technology.

An IVR solution was programmed by using the Python programming language and written with the Notepad++ text-editor. The Python source code was compiled in FreeSwitch's own Python compiler. The GitHub version control software was used to make backups and to control different stages of development. Overall, this thesis gives an overview of interactive voice response solutions and of the FreeSwitch platform.

As a result, stable software, which is under testing at the moment, was created. The software has now its own number pool including ten separate phone numbers which are all valid. In each phone number there is a different IVR service like a menu or call transferring.

In the project all the main goals were achieved on time and the software is under testing. In addition, Ainacom obtained a new solution an interactive voice response system which will replace the current solution after the testing phase. In conclusion, the thesis gives an overview of the workflow, current state of software and some ideas for future development

| | |
|---|---|
| Keywords | IVR, Software, Software switch, Python |

## Sisällys

## Lyhenteet

| | |
|---|---|
| API | Application programming interface. Ohjelmointirajapinta, jonka kautta voidaan tehdä, vaihtaa tai vastaanottaa komentoja ohjelmien välillä. |
| Ext | Ulkoinen hallintalogiikka. Järjestelmäarkkitehtuuri, johon voidaan määrittää tulevan puhelun logiikka. |
| IP | Internet Protocol. Protokolla, joka vastaa tietoliikennepakettien toimittamisesta perille tietoliikenneverkoissa. |
| ISDN | Integrated Services Digital Network. Piirikytkentäinen puhelinverkkojärjestelmä, joka on suunniteltu puheen ja datan siirtoon tavallisissa puhelinlinjoissa. |
| IVR | Interactive Voice Response. Teknologia, joka mahdollistaa vuorovaikutuksen ihmisen ja tietokoneen välillä. |
| JSON | JavaScrip Object Notation. Tiedostomuoto, jota voidaan käyttää tiedon lähetykseen tai tiedon tallentamiseen. |
| MSC | Mobile Switching Center. Palvelin, joka käsittelee puhelinverkossa liikkuvia puheluita |
| PBX | Private Branch Exchange. Puhelinjärjestelmä, joka hallitsee yrityksen sisällä käyttäjien puheluita. |
| SIP-PAI | SIP P-Asserted Identity. Identiteetti, jossa näkyy SIP ID ja valinnainen lisä nimi, esimerkiksi: ”Santeri Nousiainen <sip:santeri@metropolia.fi>”. |
| SIP | Session Initiation Protocol. Protokolla, jota käytetään luomaan, hallitsemaan ja lopettamaan IP-pohjaisia istuntoja. |
| TTS | Text To Speech. Teknologia, jolla robotti pystyy lukemaan ennaltamääritetyn tekstin. |

| | |
|---|---|
| UML | Unified Modeling Language. Mallinnuskieli, joka on tarkoitettu järjestelmien visualisointiin. |
| VoIP | Voice over IP. Tekniikka, jonka avulla ääntä voidaan siirtää reealiaikaisesti Internetin välityksellä. |
| XML | Extensible Markup Language. Kuvauskieli, jolla voidaan kuvata tietoa sekä käyttää formaattina tiedonvälityksessä. |

# 1 Johdanto

Tausta

AinaCom Oy, joka on suomalainen viestintäjärjestelmien uudistamiseen keskittynyt yritys, tarjoaa asiakkailleen viestintäratkaisuja. Niihin kuuluvat muun muassa Microsoft Lync-palvelun integrointi yrityksen viestintäjärjestelmään sekä erilaiset puhelinvaihdejärjestelmät asiakkaan tarpeiden mukaan. Tämä opinnäytetyö tehdään AinaCom Oy:lle ja sen tavoitteena on korvata nykyinen puhelinvalikkoratkaisu (Interactive Voice Response, IVR) sekä tehdä uusi joustavampi, helpommin integroitava ja nykyistä kasvua vastaava IVR-ratkaisu.

Puhelinvalikot ja puhelunhallinta ovat nykyään arkipäivää; monesta yrityksestä ja laitoksesta (esimerkiksi sairaalat) löytyy automaattinen puhelinvastaaja, joka antaa soittajalle eri vaihtoehtoja miten toimia. Interaktiivinen puhelinvalikko on yleensä osana yrityksen tai laitoksen puhelinjärjestelmää. Automaattisella valikolla voidaan ohjata puheluita ihmisiä tehokkaammin, koska automaatti voi vastaanottaa useamman puhelun kerralla ja se antaa valmiit vaihtoehdot, joista asiakas valitsee, mitä hän haluaa tehdä tai kenelle hän haluaa puhua. AinaCom myy asiakkailleen puhelinvaihdejärjestelmiä, joihin sisältyy automaattinen puhelinvalikko, jonka ominaisuudet räätälöidään asiakkaiden tarpeiden mukaan. Nykyinen puhelinvalikko on luotava manuaalisesti uudelle tilaajalle, koska puhelinvaihdejärjestelmien kysyntä kasvaa, myös työmäärä kasvaa puhelinvaihdetta tarjoavalla yrityksellä.

Ainacomin nykyinen IVR-puhelinvalikkojärjestelmä on palvelimella ajettava sovellus, joka koostuu web-pohjaisesta graafisesta käyttöliittymästä sekä tietokannasta. Graafisesta käyttöliittymästä voidaan määritellä erikseen puhelinvalikko, joka tallennetaan tietokantaan. Sovellus vaatii puhelun kierrättämisen erillisen palvelimen kautta, jonka jälkeen se ohjataan piirikytkentäiseen puhelinverkkojärjestelmään (ISDN), sekä palvelimelta vaaditaan erillistä vastaanotinta, jotta se pystyy käsittelemään puheluita.

Insinöörityön tavoite on luoda uusi ratkaisu, joka käyttää Internet Protocol (IP) pohjaista Session Initation Protocol (SIP) -signalointia. SIP-signalointi tuo kapasiteettia lisää sekä

vähentää tarvetta hankkia fyysisiä laitteita, koska SIP-puheluita voidaan ohjata Internetissä esimerkiksi Linux-palvelimen välityksellä. IVR koostuu erilaisista osista, esimerkiksi: ”soita tiedote”, ”ohjaa puhelu” tai ”lopeta puhelu”, jolloin ratkaisuun saadaan joustavuutta ja monipuolisuutta. Erilaiset osat tuovat myös lisää hallittavuutta, toisin kuin normaalissa puhelinvalikossa komentojen järjestys ei ole vakio vaan erilaisia komentoja yhdistellään toimimaan asiakkaan haluamassa järjestyksessä tai ihan vain yksittäin. IVR rakennetaan monesta erilaisesta osasta myös sen takia, että tarvittaessa yksittäisiä osia on helpompi lisätä, poistaa tai muokata verrattuna ison kokonaisuuden muokkaamiseen.

Tavoite

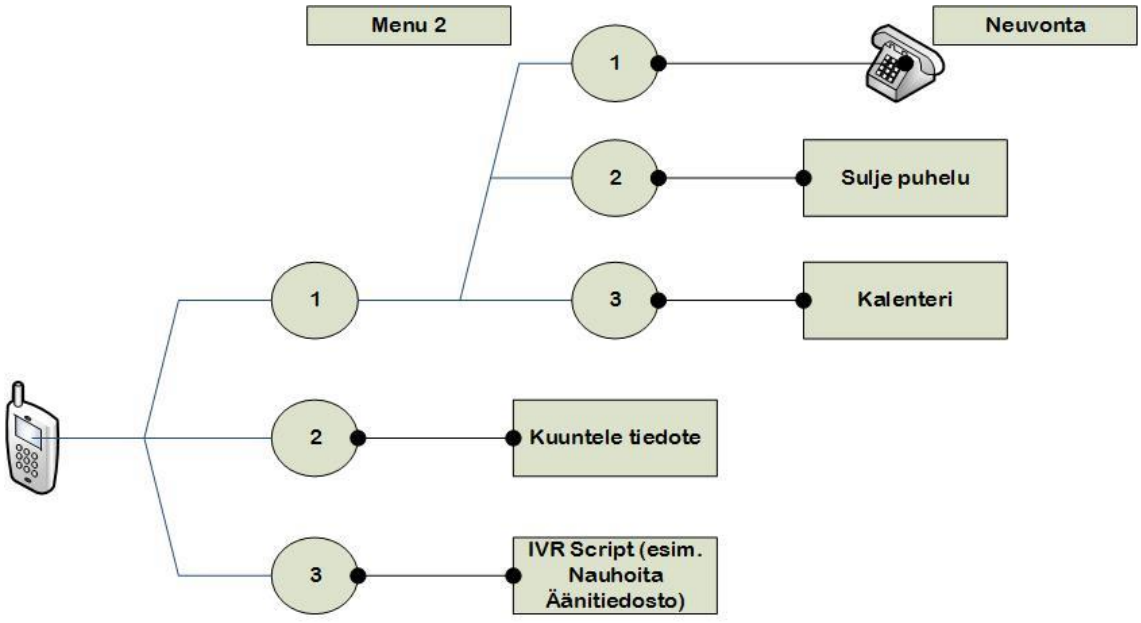
IVR tehdään omalle palvelimelle palvelinsovellukseksi, ja se toimii FreeSwitch alustan päällä. Sovellukselle annetaan oma puhelinnumero, johon soittamalla puhelu yhdistyy IVR:lle ja yhdistämisen jälkeen puhelu käsitellään puhelinvalikossa. FreeSwitch on puhelujen reititykseen ja puhelunhallintaan suunniteltu avoimen lähdekoodin alusta, johon kehittäjä voi tehdä omia työkaluja sekä käyttää sitä niin Windows- kuin Linux-palvelimella. Sovellus toteutetaan Python-ohjelmointikielellä ja XML (Extensible Markup Language) -merkkauskielellä. IVR käy komentodialoginsa käyttäen hypertekstin siirtoprotokollaa (http), ja komennot ovat JSON (JavaScript Object Notation)-formaatissa.Työssä käydään läpi ohjelman luontiprosessia, luontiprosessin ongelmia ja sitä miksi FreeSwitch on valittu alustaksi muiden vastaavien alustojen sijasta.

Insinöörityön tavoitteena on kehittää uusi IVR-järjestelmä ja käydä läpi kehittämisen eri vaiheita, erilaisia ongelma sekä vaihtoehtoisia ratkaisuja ohjelmoijan näkökulmasta. työssä ei kuitenkaan käydä läpi ulkoisen palvelulogiikan rakentamista IVR :lle eikä siitä, miten dialogi ulkoisen palvelulogiikan kanssa rakennetaan. Sovellusta ei julkaista kaupoissa, mutta sen pystyy lataamaan GitHub-palvelusta.

# 2 Puhelinvalikko

## 2.1 Interaktiivinen puhelinvalikko käsitteenä

Interaktiivisella puhelinvalikolla tarkoitetaan teknologiaa, jossa tietokone keskustelee ihmisen kanssa [10]. Keskustelu voidaan käydä joko äänen tai DTMF (Dual-Tone multi-frequency)-taajuuden tunnistamisen avulla. Yleisin kommunikointitapa järjestelmissä on DTMF. IVR-teknologiaa käytetään palvelemaan suuria määriä puheluita, joissa soittajan ei tarvitse asiantuntijan apua tai erikoisosaamista kuten, esimerkiksi ajanvaraus lääkärille. Teknologian ansiosta yrityksen tai laitoksen ei välttämättä tarvitse palkata henkilökuntaa yksinkertaisimpien asioiden hoitamiseen. Asiakaspalvelu voidaan keskittää kysymyksiin, joihin tarvitaan asiantuntijaa tai niitä ei voida hoitaa yksiselitteisesti.



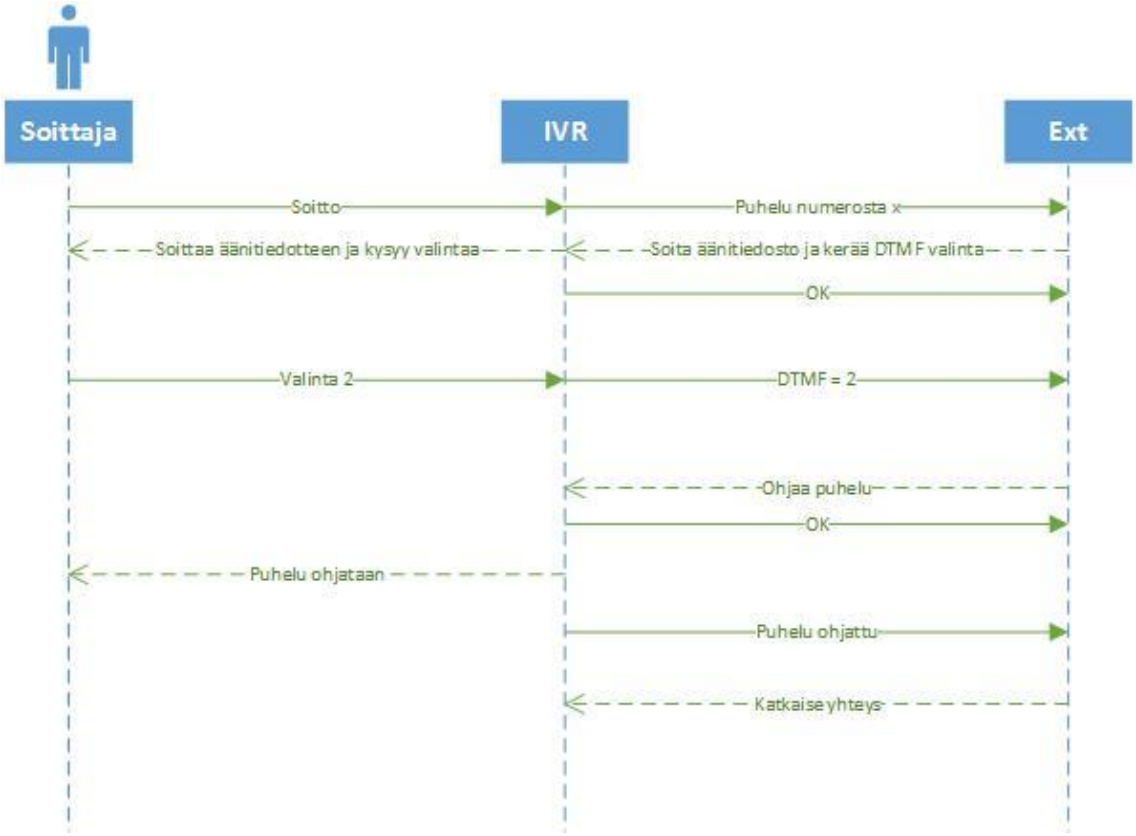

**Kuva 1. IVR järjestelmä yleisesti.**

Kuvasta 1 voidaan nähdä esimerkki yksinkertaisesta IVR-toiminnallisuudesta. Ensiksi puhelimella soitetaan ja saadaan kolme vaihtoehtoa: 1. Siirry toiseen valikkoon, 2. Kuuntele tiedote ja 3. Nauhoita äänitiedosto. IVR-skriptillä vaihtoehdossa 3 voi olla monia muitakin erilaisia toimintoja, joita halutaan ohjelmoida palveluun.

Järjestelmään voidaan myös skriptien avulla tehdä hienostuneempia versioita normaaleista toiminnoista, kuten puhelinvastaajasta. Normaali puhelinvastaaja tallentaa viestin ja viestinjättäjän tiedot, jotka voi jälkeenpäin kuunnella soittamalla tiettyyn numeroon. IVR teknologialla sen sijaan voi tehdä vastaajan, joka kysyy käyttäjältä, haluaako hän kuunnella, muokata, lähettää edelleen vai poistaa viestejä vastaajasta.

## 2.2 Toiminta

IVR-järjestelmän toiminta tapahtuu palvelimella, jossa on internetyhteys, puhelinverkkoyhteys tai molemmat. Insinöörityöntyön IVR palvelinta ei kytketä puhelinverkkoon, koska tarkoituksena on toimia kokonaan IP-pohjalla VoIP (Voice over IP)-verkossa. VoIP-verkko toimii internetissä, joten palvelimeen ei silloin tarvitse lisätä fyysistä komponenttia puhelinverkkoihin yhdistämistä varten. Tämä säästää rahaa ja on nopeampaa kuin puhelinverkossa toimiminen. Haittapuolena ovat Internetin tuomat ongelmat, kuten se, että DTMF-taajuus voi sekoittua verkkoliikenteessä, sekä tietoturvariskit.




**Kuva 2. IVR:n toiminta esimerkki.**

Kuvassa 2 nähdään esimerkki IVR:n toiminnasta puhelun tullessa. Asiakas soittaa IVR-järjestelmään, joka välittää puhelun tiedot, kuten numeron, eteenpäin ulkoiselle hallintalogiikalle (Ext). Tämän jälkeen tulee tieto siitä mitä palveluita tälle numerolle tarjotaan. Kuvan 2 esimerkissä asiakas ohjataan valikkoon ja häneltä kerätään valinta. Valinnan jälkeen suoritetaan asiakkaan valitsema komento ja puhelu katkaistaan. Insinöörityön ratkaisu käyttää valintojen keräämiseen ainoastaan DTMF-tekniikkaa.

Dual-tone multi-frequency on puhelinlaitteissa yleisesti käytetty äänitaajuusvalintatapa, jolla voidaan tunnistaa valittu numero. Toinen tapa on kuunnella ja tulkita soittajan äänestä komennot, mikä vaatii erikoistuneempaa tekniikkaa kuin taajuuksien kuuntelu. Puhelimessa eri numeroita vastaavat omat taajuudet, jolloin IVR tunnistaa, mitä numeroa käyttäjä on painanut. Numeroiden määrää, joita voi näppäillä, ei ole rajoitettu. Tarvittaessa voidaan pyytää käyttäjältä perinteisen 1 - 9 näppäimien sijasta myös numerot 1 - 19, 1 - 29, ja niin edelleen.

**Taulukko 1. DTMF-taajuudet (2).**

| | 1209 Hz | 1336 Hz | 1477 Hz |
|---|---|---|---|
| 697 Hz | 1 | ABC 2 | DEF 3 |
| 770 Hz | GHI 4 | JKL 5 | MNO 6 |
| 852 Hz | PRS 7 | TUV 8 | WXY 9 |
| 941 Hz | * | 0 | # |

Taulukosta 1 voidaan nähdä, millä taajuuksilla eri painallukset toimivat. Jokainen taajuuspari sisältää yhden taajuuden alhaisten taajuuksien ryhmästä (697 Hz, 770 Hz, 852 Hz, 941 Hz) sekä yhden taajuuden ylätaajuuksien ryhmästä (1209 Hz, 1336 Hz, 1477 Hz). Tällä tavalla muodostuu yhteensä 12 numeroparia.

## 2.3 Käyttöönotto

Interaktiivisia puhelinvalikoita on erilaisia ja niiden käyttöönotot tapahtuvat eri tavoilla eri yrityksissä. Tämän insinöörityön yksi tavoite oli tehdä IVR-järjestelmästä helpommin integroitava, jotta asiakkaan olisi mahdollisimman helppo ottaa se käyttöön. Järjestelmä ei vaadi minkäänlaista fyysistä palvelinta asiakkaan tiloihin, eikä sovelluksien asentamista. Asiakkaalle annetaan puhelinnumero, jossa palvelu toimii ja se riittää. Kuitenkin, jos on tarvetta luoda omia äänitiedostoja, asiakas joutuu käyttöönoton yhteydessä tekemään ja lataamaan ne itse palvelimelle.

### IVR:n käyttöönoton edut

Helpon käyttöönoton lisäksi voidaan luetella muutamia yleisiä etuja, joita IVR tuo asiakkaalle [4]:

- rahan säästäminen

- ajan säästäminen
- usean puhelun samanaikainen hallinta
- nopea vastausaika
- järjestelmällisyys.

Ajan ja rahan säästäminen on suurin etu IVR-järjestelmän käyttöönotossa niin suurelle kuin pienellekin yritykselle [4]. Puhelimeen vastaaminen vie aikaa asiakaspalvelijalta riippumatta siitä, onko kysymys yksinkertainen vai vaikea, joten työajan säästämiseksi yksinkertaiset asiat voidaan siirtää IVR:n hallintaan. Esimerkkejä yksinkertaisista asioista ovat eniten kysytyt kysymykset, kuten ajanvaraus sekä soitonsiirto tietylle henkilölle. IVR kuitenkin pystyy hoitamaan myös vaativampiakin tehtäviä, mutta kääntöpuolena on, että ohjelmoija tai muu erityisosaaja joutuu käyttämään paljon aikaa tämänlaisen ohjelman ohjelmointiin.

Asiakaspalvelijalle voi olla myös haastavaa palvella useampaa soittajaa samanaikaisesti, kun taas IVR pystyy vastaanottamaan puheluja laitteiden kapasiteettien perusteella useita kymmeniäkin samanaikaisesti. Normaalissa tilanteessa asiakaspalvelijan puhuessa puhelimessa asiakas joutuu jonottamaan, kunnes saa yhteyden palvelijaan. IVR-järjestelmä vastaa automaattisesti asiakkaalle, joten asiakas pysyy tyytyväisempänä, koska hän saa nopeaa palvelua.

Tekemällä järjestelmään yksinkertaiset ”polut”, joita pitkin soittaja voi edetä, luodaan yrityksen organisaatioon järjestelmällisyyttä. Esimerkkinä tästä voisi mainita, että puheluita ei yhdistetä väärälle henkilölle.

IVR:n käyttöönoton haitat

Järjestelmän mukana tulee myös etujen lisäksi muutamia mahdollisia haittoja [4]:

- Asiakasta ei palvele ihminen.
- Järjestelmä on mahdollisesti hidas.

- Hyvän IVR suunnitteleminen ja toteutus vaatii ammattilaisen.
- IVR voi olla liian monimutkainen.
- IVR voi olla epäselvä.

Monelle vanhemmalle ihmiselle voi tuottaa ongelmia kuunnella tai puhua robotin kanssa. [4.] Jos vanhus ei kuule, mitä IVR sanoo, niin hän ei voi kysyä uudelleen, mitä ääni sanoi, vaan suurimmassa osassa laitteissa pitää kuunnella nauhoitus loppuun ennen kuin voi aloittaa kuuntelemisen alusta. Robottien äänenlaatu vaihtelee hyvinkin paljon riippuen siitä, onko järjestelmästä maksettu vähän vai paljon. Asiakas voi myös nauhoittaa omat äänitiedotteet ja yhdistellä niitä, mutta tämä vaatii nauhoituslaitteet. Liian pitkät valikot ovat myös pitkäkestoisia. Jos asiakas haluaa vain tehdä ajanvarauksen, häntä ei kiinnosta kuunnella kaikkia muita mahdollisuuksia, mitä voi tehdä.

Hyvän ja yksinkertaisen puhelinvalikon suunnittelu vaatii aikaa. Suunnitellussa pitää huomioida, kuinka paljon halutaan antaa vaihtoehtoja asiakkaalle, jotta vastaajasta ei tulisi liian hidas. Valikosta voi myös tulla epäselvä riippuen eri valintojen määrästä. Ongelmaksi voi muodostua myös se, että valikon vaihtoehtoa yhdeksän kuunnellessaan asiakas on jo unohtanut, mikä vaihtoehto kaksi oli.

## 2.4 Puhelin vai puhelinvalikko?

Niin pienelle kuin suurelle yritykselle puhelinvalikko on hyvä ratkaisu asiakkaiden puheluiden hoitamiseen. Suuret yritykset saavat päivässä paljon puheluita, joissa kaikissa ei tarvita erikoisasiantuntijaa. Tämänlaisten puhelujen hallintaan IVR on oikea ratkaisu. Toisaalta puhelut, jotka koskevat monimutkaisempia ongelmia, IVR voi ohjata ihmiselle, jonka kanssa asiakas ratkaisee, miten edetään. Asiakaspalvelijoista ei siis kannata kokonaan luopua.

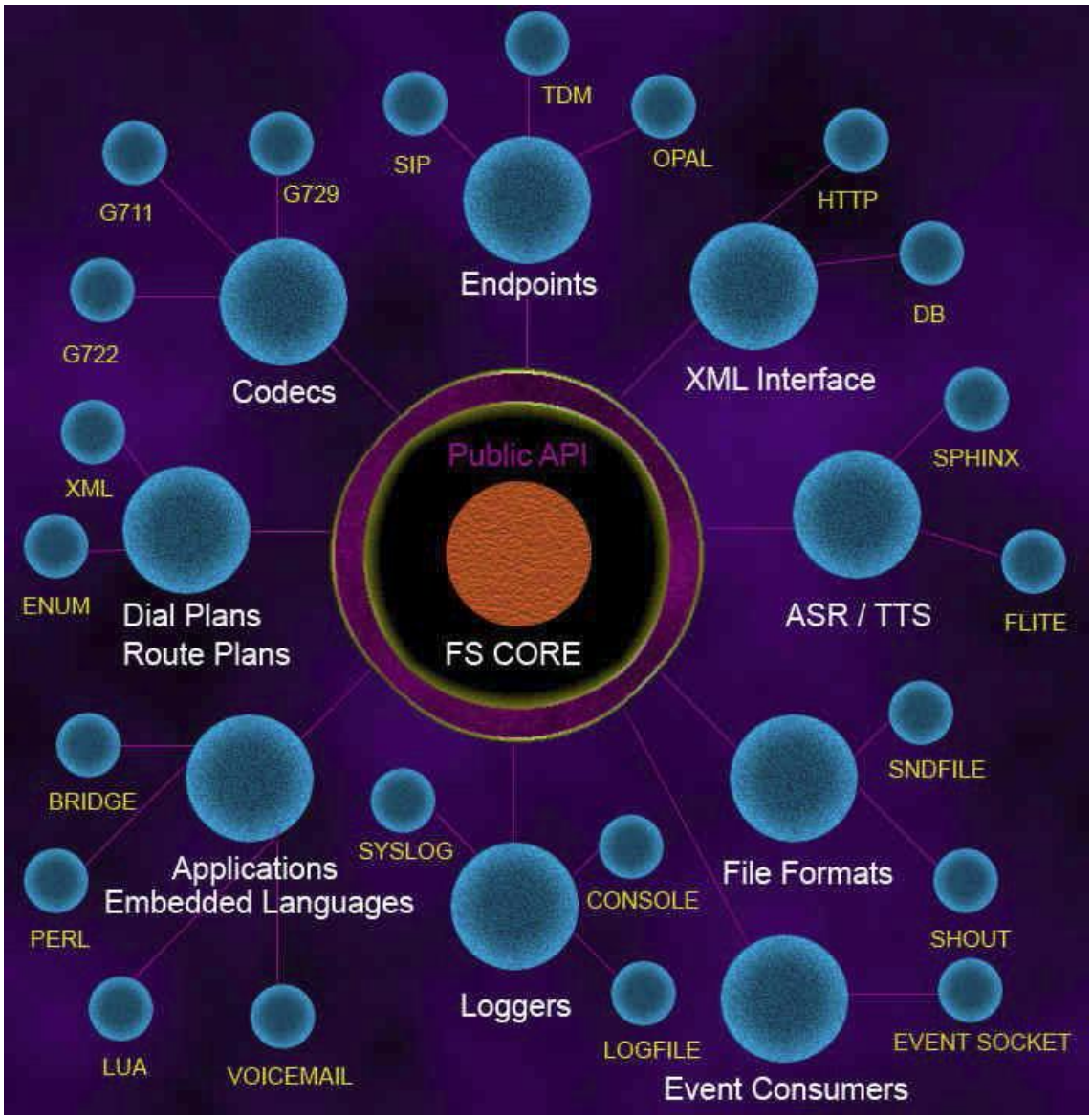
Pienille yrityksille IVR-järjestelmä voi luoda kuvan isommasta yrityksestä. IVR säästää myös henkilöstökuluissa eikä pienen yrityksen tarvitse palkata välttämättä asiakaspalvelijaa puheluiden vastaamiseen. Näissä tapauksissa IVR voi esimerkiksi

suoraan ohjata puhelut henkilöille sellaisina ajankohtina, kun henkilö voi puheluihin vastata.

## 3 FreeSwitch-alusta

Software Switch (softswitch) on ohjelma, joka toimii PC:llä puhelinkeskuksena eli yhdistää puhelimella soitettuja puheluita toiselle henkilölle joko Internetin tai puhelinlinjojen välityksellä. Pääosin softswitchejä käytetään puheluihin, jotka tapahtuvat internetissä, esimerkiksi Skype-puhelut. IP-verkossa soitetut puhelut toimivat VoIP-teknologian (Voice over IP) avulla, ja näitä puheluita ohjataan esimerkiksi softswitch järjestelmillä. Softswitch-sovelluksen tavoite on yhdistää puhelinlinjoja verkosta toiseen verkkoon. Tällaisia järjestelmiä on markkinoilla sekä maksullisia että maksuttomia. Avoimen lähdekoodin tunnetuimpia alustoja ovat Asterisk ja FreeSwitch [5].

FreeSwitch on avoimeen lähdekoodiin perustuva ohjelmistopohjainen (softswitch) alusta, joka on kehitetty vuonna 2006 korvaamaan maksullisia softswitch-alustoja. Sen on alun perin suunnitellut Anthony Minessale, Brian West ja Michael Jerris, jotka kaikki kolme ovat tunnetun Asterisk-järjestelmän entisiä kehittäjiä. Nykyään sitä kehittää moni ohjelmistoalan ammattilainen ja uusia työkaluja syntyy lähes päivittäin. FreeSwitchin kehittäjien tavoite on tarjota modulaarinen, skaalautuva ja vakaa ydin tietoliikennealan kehittäjille [1].




**Kuva 3. Hahmotelma FreeSwitchin modulaarisuudesta (1).**

Modulaarisuutta voidaan havainnollistaa kuvan 3 avulla. Keskellä kuvaa on FreeSwitchin ydin, ”FS CORE”, jota pääsee kehittämään julkisen rajapinnan kautta (Public API). Ytimeen liittyy kahdeksan eri moduulia, joilla jokaisella on oma tehtävänsä ja joihin myös ohjelmoija voi tuoda omia ominaisuuksia. Insinöörityössä keskityn muokkaamaan Dial Plan -moduulia sekä hyväksikäytän ohjelmaan sisäänrakennettua Python ohjelmointikielen kääntäjää.

## 3.1 FreeSwitch vs Asterisk

Asterisk on avoimeen lähdekoodiin pohjautuva PBX-alusta sovellusten rakentamiseen Linuxille. Se on myös otettu käyttöön yli 170 maassa, ja tällä hetkellä on yli miljoona Asterisk-pohjaista systeemiä käytössä. Asterisk-projekti alkoi vuonna 1999, kun Mark Spencer julkaisia lähdekoodin GL open source lisenssin alla. Sen jälkeen sitä on testattu tuhansien käyttäjien toimesta [6]. PBX-alustalla tarkoitetaan puhelinjärjestelmää, joka toimii yrityksen sisällä ja tarjoaa yrityksen työntekijöille puhepalveluita ja sellaisia lisäpalveluita, kuten puheluiden yhdistämistä, konferenssipalveluita ja puhelinvastaajan. PBX-sovelluksen päätarkoitus on auttaa puhelimia löytämään toisiaan ja kommunikoimaan näiden löydettyjen puhelimien kesken, kun taas softswitch auttaa verkkoja kommunikoimaan keskenään. Asterisk on PBX-järjestelmä, joka on pääosin suunniteltu yrityksen sisäisten puhepalveluiden hyödyntämiseen, jonka takia se ei sovellu mielestäni insinöörityön tarkoitusperiin.

Valitsin insinöörityön alustaksi FreeSwitchin, koska toteutettava IVR toimii muuallakin kuin yrityksen sisällä. FreeSwitchin valintaan vaikutti myös se, että se tukee sekä Windowsia että Linuxia. Tämä tuo ohjelmaan lisää joustavuutta, koska se ei enää ole käyttöjärjestelmästä riippuvainen. FreeSwitch on myös ilmainen käyttää ja ladata, mikä oli valitsemisen yksi syy.

## 3.2 Asennus

FreeSwitch on alustasta riippumaton ohjelma eli sitä voi käyttää niin Linux- kuin Windows-käyttöjärjestelmässä. Asennuspaketin voi ladata GitHub palvelusta. Ennen FreeSwitchin asennusta Linux-palvelimella on oltava seuraavat ohjelmat ja kirjastot valmiina:

- Git
- Autoconf
- Automake

- GCC-C++
- LIBJPEG-DEVEL
- LIBTOOL
- MAKE
- NCURSES-DEVEL

Asennuksen jälkeen FreeSwitchin konfiguraatiotiedostosta pitää käydä valitsemassa mitä ohjelmointikieltä käyttää ohjelmoinnissa.

```
<!-- Languages -->
<!-- <load module="mod_spidermonkey"/> -->
<!--load module="mod_v8"/>
<!-- <load module="mod_perl"/> -->
<load module="mod_python"/>
<!-- <load module="mod_java"/> -->
<!--load module="mod_lua"/>
```

**Kuva 4. Tuetut ohjelmointikielet.**

Kuvassa 4 on konfiguraatiotiedosto, jossa ohjelmointikielen valitseminen tapahtuu. Tuetut kielet ovat Spidermonkey, v8, Perl, Python, Java ja LUA. Tämän jälkeen FreeSwitch osaa tulkita Python-koodia ja voidaan aloittaa oman ohjelman tai skpritin tekeminen sovellukseen. Myös muut konfiguraatiotiedostot on kirjoitettu XML-merkkauskielellä.

## 3.3 FreeSwitchin tärkeät moduulit

### Soittosuunnitelma

Soittosuunnitelma (Dial Plan) on FreeSwitchin sisällä oleva moduuli, joka hallinnoi puhelun kulkua, tapahtumia puhelun aikana, puhelun siirtoa ja lopettamista. Jokaiselle numerolle voidaan tehdä oma soittosuunnitelma. Soittosuunnitelmia ja laajennuksia soittosuunnitelmiin tehdään XML-merkkauskielellä.

```
<extension name="Time of day, day of week setup" continue="true">
  <condition field="destination_number" expression="^0937897960$" wday="2-6" hour="8-21" break="never">
    <action application="set" data="office_status=open" inline="true"/>
    <action application="python" data="IVR"/>
  </condition>
  <condition hour="0-11" break="never">
    <action application="set" data="day_part=morning" inline="true"/>
  </condition>
  <condition>
    <action application="set" data="day_part=afternoon" inline="true"/>
  </condition>
  <condition hour="18-23" break="never">
    <action application="set" data="day_part=evening" inline="true"/>
  </condition>
</extension>
```

**Kuva 5. Soittosuunnitelma.**

Kuvassa 5 nähdään soittosuunnitelman yksi laajennus (extension). Kokonainen soittosuunnitelma rakentuu monesta laajennuksesta. Esimerkki alkaa siitä, että luodaan ”Time of day, day of week setup” -niminen laajennus, jossa ”continue = true” tarkoittaa, että vaikka löydetään tämän niminen laajennus, niin jatketaan vielä toisien laajennuksien etsimistä. Extension-viittauksen jälkeen siirrytään ehto-osioon (condition). Ensimmäinen attribuutti on ”field”, jolla viitataan soitettuun numeroon ”destination_number”, jossa expression tarkoittaa field-attribuutin arvoa. Tässä esimerkissä expression arvoksi on valittu puhelinnumero, johon soittamalla FreeSwitch alkaa toteuttaa kyseistä laajennusta. Attribuutti ”wday” tarkottaa viikonpäivää, ”hour” tarkoittaa kellonaikaa ja ”break” katkaisua. Jos nämä kaikki ehdot ”condition” tagin sisällä täyttyvät, niin laitetaan arvon open muuttujalle ”office_status” ja siirrytään Python-ohjelmaan nimeltä IVR.

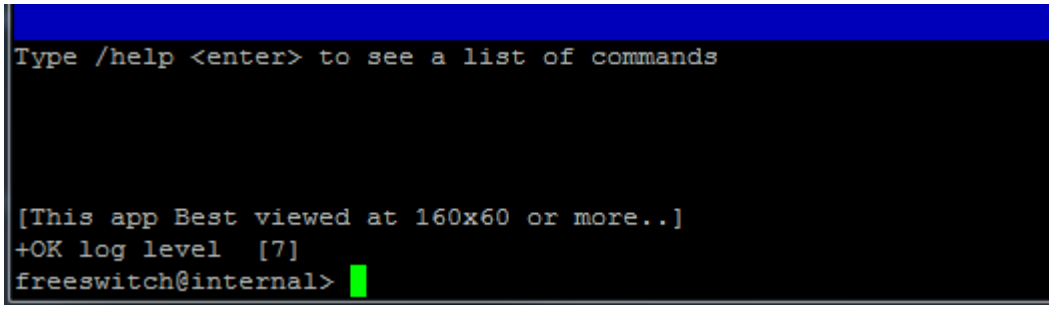
Laajennuksessa on myös lisäehtoja, joiden avulla voidaan määrittää, onko tällä hetkellä menossa aamu, iltapäivä vai ilta. Näiden avulla IVR valikosta voi tehdä ”viisaamman” laittamalla eri tiedotteen eri vuorokaudenajoille. Kellonaikojen avulla voidaan myös päättää, milloin IVR vastaanottaa puheluita.

### Endpoint

Soittosuunnitelmien lisäksi Endpoint-moduuli on ohjelmoinnin ja toiminnan kannalta tärkeä [1]. Moduulin päärooli on tehdä yleisistä kommunikaatioteknologioista ilmentymä, jota kutsutaan sessioksi. Sessiolla kuvataan yhteyttä FreeSwitchin ja jonkin toisen tietyn protokollan välillä [1]. Insinöörityön kannalta tärkein protokolla on SIP. SIP-istunto syntyy silloin kun puhelu on yhdistetty Internetissä yli FreeSwitchistä toiseen SIP-laitteeseen ja sessio on aktiivinen niin kauan kuin puhelu on päällä.

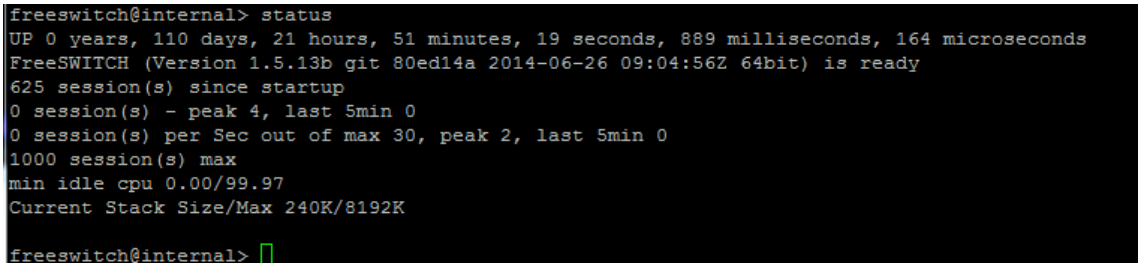
### 3.4 FreeSwitchin ohjelmointi

FreeSwitchin valintaan vaikutti myös se, että alustaan voi tehdä monimutkaisiakin ominaisuuksia yksinkertaisella ohjelmoinnilla. Alustan omat kirjastot ja funktiot ovat kattavia, ja niitä on helppo käyttää omissa sovelluksissa. Omia sovelluksia alustalle voi tehdä joko XML-merkkauskielellä tai FreeSwitchin tukemilla ohjelmointikielillä, jotka ovat mainittu kohdassa 3.2. Ohjelmoijalle tärkeä moduuli on FreeSwitchin tarjoama rajapinta (FSAPI), joka toimii komentoriviperiaatteella, eli siihen voi syöttää yksittäisiä tekstijono-arvoja ja se palauttaa myös tekstijono-arvoja. FSAPI saadaan auki kirjoittamalla Linux-komento `./fs_cli` jonka jälkeen komentoikkuna avautuu.




**Kuva 6. FSAPI:n perusnäkymä.**

Kuvassa 5 on aloitusnäkymä tilanteesta, kun FSAPI on käynnistetty. Tässä tilanteessa voidaan syöttää komentoja FreeSwitchille.




**Kuva 7. Esimerkkikomento.**

Kuvassa 6 rajapintaan syötettiin komento `status`, johon saatiin vastaus. Ensimmäisellä rivillä kerrotaan aika, kuinka pitkään FreeSwitch on ollut päällä ja tämän jälkeen saadaan nykyinen status seuraavalla rivillä. Statuksen jälkeen tulee yleistä tietoa FreeSwitchin kuormituksesta sinä aikana, kun se on ollut päällä. Komentoikkuna on

tärkeä työkalu ohjelmoinnin aikana, koska se toimii myös virheenhallintatyökaluna. Kaikki virheet, jotka puhelun aikana tai ohjelman ajon aikana tapahtuvat, kirjataan ja virhettä etsiessä voidaan käyttää rajapinnan komentoikkunaa apuna ja löytää virhe nopeammin. Rajapintaan voi lähettää esimerkiksi status-komennon `http://esimerkki.freeswitch.fi:4000/api/status` selaimen kautta, jolloin voidaan testata yhteyksiä palvelimien välillä. Tämä mahdollistaa myös web-applikaation, joka keskustelee FreeSwitchin kanssa http-komentojen avulla.

# 4 Toteutus

## 4.1 AinaComin tarpeet

Insinöörityön tavoitteena on korvata nykyiset IVR-palvelut sekä numerotiedotteet, esimerkiksi tiedonannot. Vaadittuja ominaisuuksia ovat

- tiedotteen toisto
- TTS (Text to Speech)
- DTMF-valintojen kerääminen
- puhelun katkaiseminen
- puhelun uudelleen ohjaus
- tallenteen äänitys
- tallenteen lataus alustalle
- virheenhallinta
- virheistä palautuminen.

Ominaisuuksien lisäksi jokaisella sessiolla tulee olla oma istuntotunnus ja ohjelman täytyy olla tietoinen soittajan puhelinnumerosta, vastaajan puhelinnumerosta. Nämä tiedot täytyy kerätä, jotta voidaan yksilöidä SIP-istunnot. Soittajan puhelinnumero tarvitaan myös sitä varten, että järjestelmä osaa soittaa oikean valikon ja/tai tiedotteet. IVR käyttää ennalta määriteltyjä komentoja dialogeihinsa, joita se käy läpi puhelun aikana. Seuraavat pseudokomennot ovat määritelty ennen työn aloittamista:

- CALL_START. Puhelun tullessa palveluun, IVR ilmoittaa, että puhelu on vastaanotettu ja SIP-istunto muodostettu.

- ANNOUCEMENT_COMPLETE. Onnistuneen tiedonannon tai TTS:n toiston jälkeen ilmoitetaan dialogin toiselle osapuolelle.

- DTMF_RESULT. Kerätään käyttäjän DTMF-valinnat ja kuittaus että valinnat on kerätty onnistuneesti. Virheelliset valinnat kirjataan myös.

- CALL_END. Puhelu on loppunut ja komentoja ei enää oteta.

- REDIRECT_RESULT. Puhelu on ohjattu eteenpäin onnistuneesti tai epäonnistuneesti.

- PLAY_ANNOUNCEMENT. Soita tiedonanto.

- PLAY_TEXT. Toista TTS.

- COLLECT_DTMF. Kerää DTMF-valinnat.

- RECORD_CALL. Nauhoittaa ääntä siihen asti, kun painetaan tiettyä numeroa.

- END_CALL. Lopeta puhelu.

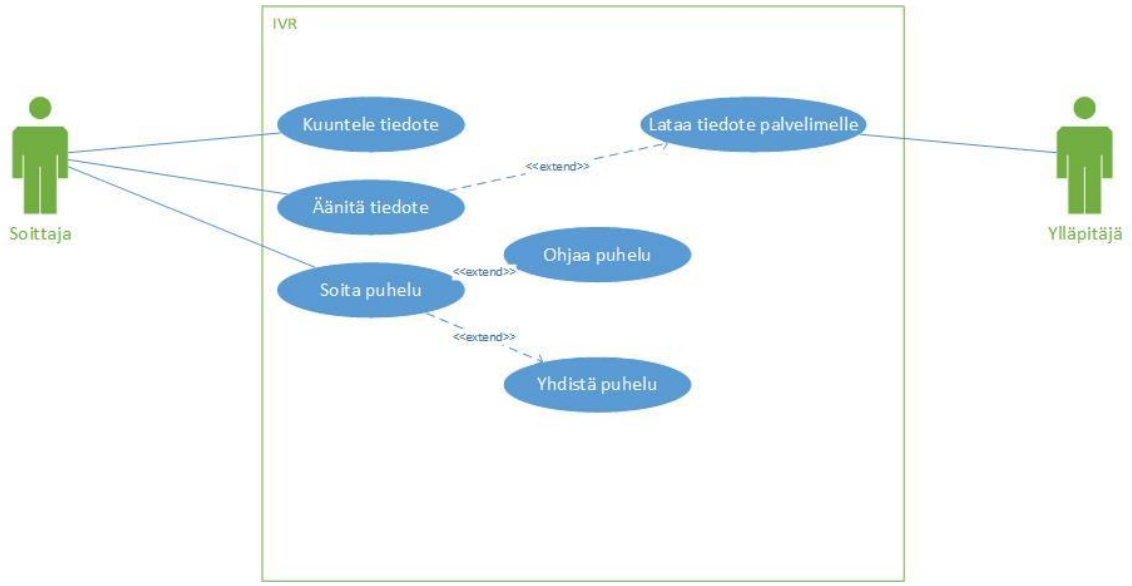
- REDIRECT_CALL. Ohjaa puhelu.

Vähimmäisvaatimus DTMF-keruussa on se, että pystytään soittamaan tiedonantoa samaan aikaan kuin keräämään valintoja. Tavoitteena on kuitenkin, että voidaan soittaa useampi tiedonanto samalla.

## 4.2 Suunnittelu ja aikataulu

Ainacomin nykyinen järjestelmä on toimiva, joten aikaa suunnitteluun, toteutukseen ja testaamiseen varattiin reilusti. Tavoitteena oli saada kolmen kuukauden aikana testattava sovellus, joka voidaan integroida järjestelmään. Sovelluksen teon aikana dokumentoidaan, mitä on tehty, mitä tehdään ja mitä tullaan tekemään.

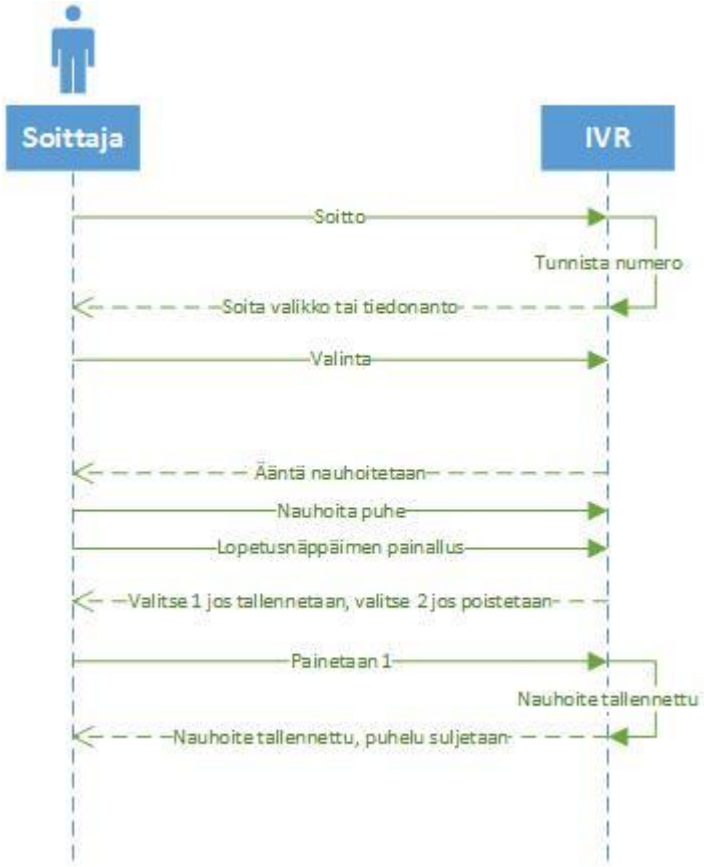
Suunnitelman ensimmäinen vaihe oli määritellä, mitkä ominaisuudet halutaan IVR järjestelmään ja mitä komentoja sen on osattava käyttää.




**Kuva 8. IVR:n Käyttötapauskaavio.**

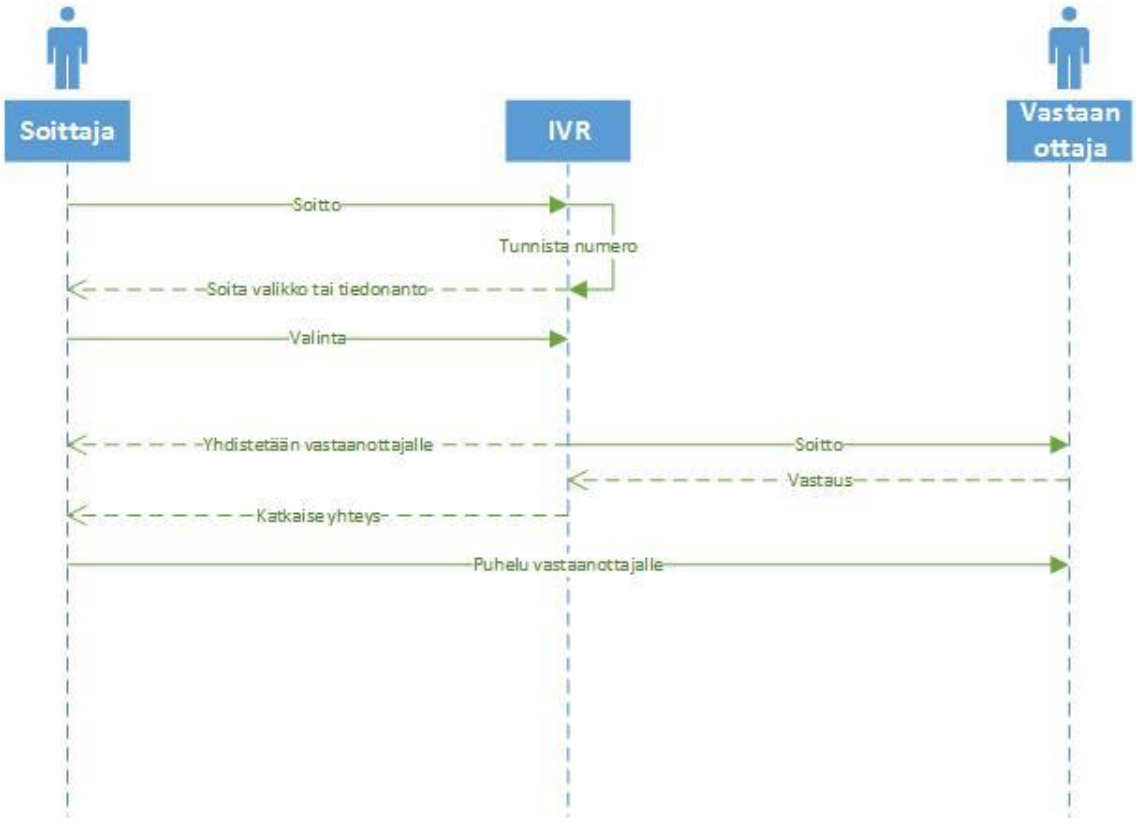
Kuvassa 8 on käyttötapauskaavio ominaisuuksista, jotka näkyvät soittajalle. Kaikki ominaisuudet voivat olla suoraan käytettävissä tai sitten niihin voidaan siirtyä valikosta painamalla valintanumeroa. Soittaja voi kuunnella tiedotteen, eli soittaessaan palveluun hänelle voidaan kertoa esimerkiksi ”Sinulle ei ole yhtään soittopyyntöä” tai hän voi äänittää itse tiedotteen, joka ladataan palvelimelle. Palvelimelle voidaan ladata ylläpitäjän tai soittajan toimesta ääninauhoite. Soittaja voi myös puhua normaalin

puhelun ja pyytää että puhelu ohjataan tietylle vastaanottajalle, tai puhelu voidaan myös yhdistää suoraan. Yhdistetyssä puhelussa yhteys FreeSwitchiin pidetään päällä, mutta ohjatussa puhelussa yhteys katkaistaan. IVR-palvelimelle on myös mahdollista lisätä tiedonantoja, jotka eivät ole puhetta, esimerkiksi musiikkia. Tämä tosin vaatii palvelimelle manuaalisen latauksen, jonka suorittaa ylläpitäjä.




**Kuva 9. Sekvenssikaavio nauhoittamisesta.**

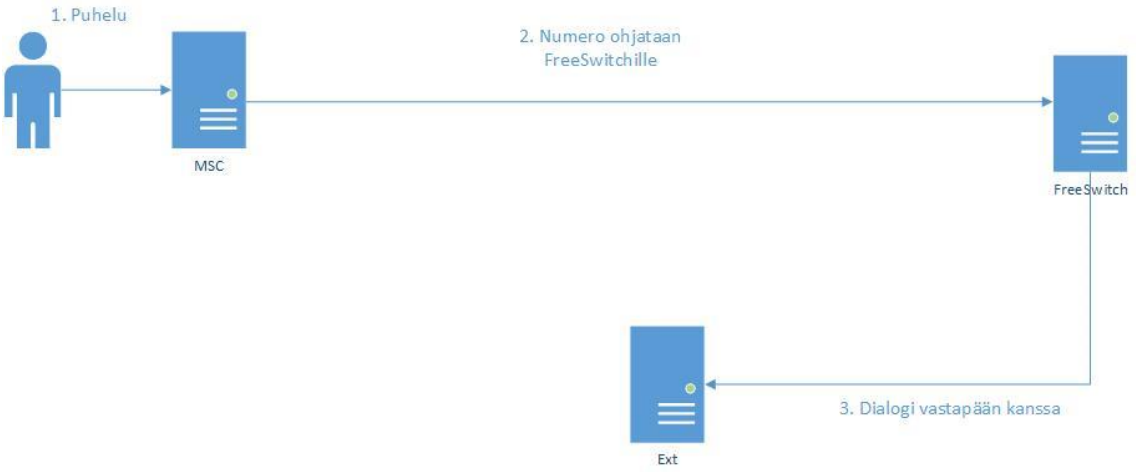
Kuvan 9 sekvenssikaavio havainnoi soittajan ja IVR:n välistä dialogia, kun halutaan nauhoittaa äänitiedote palvelimelle. Soittaja soittaa ensin palveluun ja valitsee valikosta vaihtoehdon ”Nauhoita tiedote”. Hänelle soitetaan ennen nauhoitusta tiedote ”Voit alkaa nauhoittamaan äänimerkin jälkeen, kun olet valmis paina numeroa yhdeksän”. Painalluksen jälkeen kysytään, onko soittaja tyytyväinen nauhoitukseensa vai halutaanko se poistaa. Jos soittaja on epävarma niin hän voi halutessaan kuunnella nauhoituksen uudestaan. Jos nauhoite halutaan tallentaa, niin IVR luo palvelimelle tiedoston ja ilmoittaa onnistuneesta tallennuksesta soittajalle.




**Kuva 10. Sekvenssikaavio puhelun ohjaamisesta.**

Kuvan 10 soittaja valitsee puhelun ohjaamisen ennalta määritettyyn numeroon, esimerkiksi asiakaspalveluun. IVR ilmoittaa vastaanottajalle tiedotteena, että yhdistää puhelun ja kun yhteys on saavutettu, IVR katkaisee yhteyden soittajaan, jolloin ainoa yhteys joka jää voimaan, on soittajan ja vastaanottajan välinen. IVR pystyy myös yhdistämään puhelun ja toimimaan itse ”siltana”, jolloin soittajan ja IVR:n välinen yhteys säilyy puhelun aikana.

Ominaisuuksien ja toiminnallisuuden suunnittelun jälkeen on seuraava vaihe kehitysalustan valitseminen. Alustaksi valitsin Linux-palvelimen, koska se on nopea pystyttää testausta varten, eikä Linuxissa ole lisenssimaksuja. Palvelimella ei ole muuta asennettuna kuin FreeSwitchin vaatimat kirjastot sekä Python-kirjasto ohjelmointia varten. Python toimii ohjelmointikielenä, koska FreeSwitch tukee sitä natiivisti. Pythonin asennuksen mukana tulevat kirjastot tukevat myös työtä paremmin (esimerkiksi JSON-kirjasto) kuin muut FreeSwitchin natiivikielet.




**Kuva 11. Palvelintopologia.**

Kuvassa 11 ovat palvelimet, jotka keskustelevat keskenään puhelun aikana. Kun puhelu tulee MSC:lle (Mobile Switching Centre) se ohjaa puhelun FreeSwitchille määritettyyn numeroon. FreeSwitch ottaa sen vastaan ja ilmoittaa myös, että puhelu on vastaanotettu ja on valmis käymään dialogin ulkoisen palvelimen kanssa.

## 4.3 Ohjelmallinen toteutus

FreeSwitch ohjelmoinnin aloitus.

Ohjelman UML (Unified Modeling Language) -kaaviot luotiin Microsoft Visio 2013-kaavionluontisovelluksella. Itse ohjelma tullaan kirjoittamaan Notepad++-kirjoitusohjelmalla ja Python-kääntäjänä toimii FreeSwitchin FSAPI- ja Python-moduuli.

Ohjelmointia aloitettaessa FreeSwitchin Python pohja tarjoaa kuusi funktiota, joilla jokaisella on oma tehtävänsä toiminnan aikana. Funktiot ovat

- handler
- hangup_hook
- input_callback

- fsapi
- runtime
- xml_fetch.

Handler-funktio suoritetaan ensimmäisenä. Funktio suoritetaan joka kerta, kun puhelu tulee. Session muuttujan sisällä on ID, jolla voidaan viitata käynnissä olevaan SIP-istuntoon. Handler-funktiossa alustetaan kaikki muuttujat, joita tarvitsemme puhelun aikana. Jos muuttuja alustetaan Handler-funktion ulkopuolella, kyseinen muuttuja on voimassa kaikille puheluille, joita hallinnoidaan samaan aikaan.

```
def handler(session, args):

    session.answer()
    session.setHangupHook(hangup_hook)
    session.setInputCallback(input_callback)

    ivrMenu(session)
```

**Kuva 12. Handler-funktio.**

Kuvassa 12 on handler-funktio, jossa vastataan session.answer()-funktiolla tulevaan SIP-puheluun. Tämän jälkeen alustetaan hangup_hook- ja input_callback-funktiot. Alustusten jälkeen suoritetaan funktio ivrMenu. Kuvassa 12 oleva funktio ”ivrMenu” on IVR-valikko, joka on toteutettu Python-ohjelmointikielellä [Liite 1].

```
def hangup_hook(session, what, args=''):

    freeswitch.consoleLog("info", "hangup hook for '%s'\n" % what)
```

**Kuva 13. Hangup_hook.**

Hangup_hook-funktio suoritetaan silloin kun, puhelu suljetaan, ohjataan tai yhdistetään muualle. What muuttuja kertoo, mikä on katkaissut puhelun. Esimerkkikoodi on hangup_hook funktio, joka tulostaa FSAPI komentoikkunaan syyn minkä takia puhelu on katkennut.

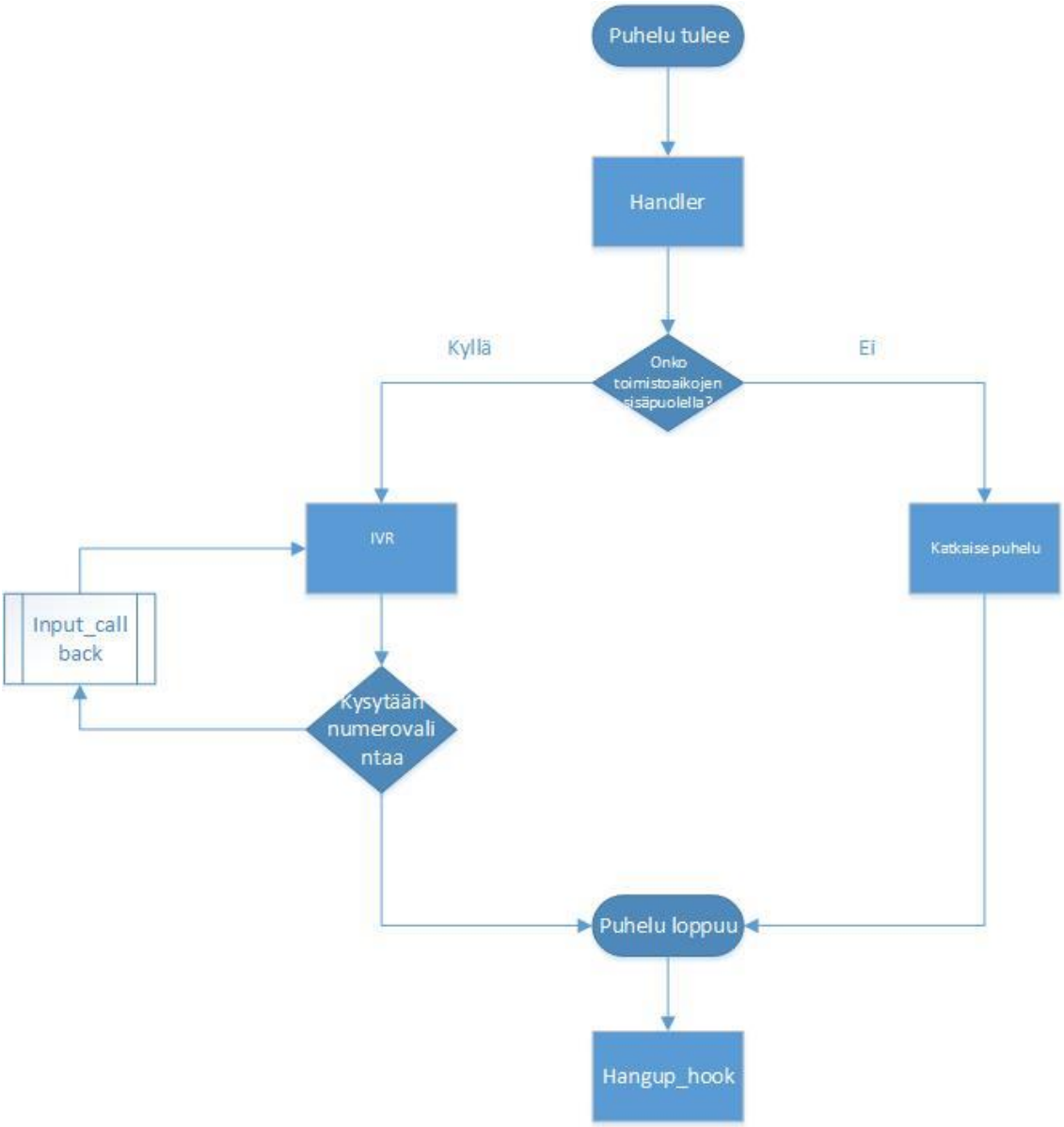
```
def input_callback(session, what, obj, args=''):

    if (what == "dtmf"):
        freeswitch.consoleLog("info", what + " " + obj.digit + "\n")
    else:
        freeswitch.consoleLog("info", what + " " + obj.serialize() + "\n")
        return "pause"
```

Kuva 14. input_callback.

Input_callback -funktio suoritetaan aina kun soittaja antaa puhelimella jonkin valinnan, esimerkiksi DTMF-painallus johtaa funktion suorittamiseen. Esimerkkikoodin parametreina ovat session-, what-,obj- ja args-muuttujat. Muuttuja ”what” kertoo, minkä syötteen käyttäjä on antanut, ja muuttujan ”obj” sisällä on tämän syötteen arvo. Funktio kertoo aina syötteen tullen, minkälainen syöte on kyseessä ja mikä sen arvo on.

Insinöörityön IVR ei käytä xml_fetch-, runtime- tai fsapi-funktioita, koska ne ovat tarkoitettu puheluiden ulkopuoliseen ohjaamiseen joko XML-merkkauskielen tai http:n avulla.




**Kuva 15. Vuokaavio ohjelman toiminnan vaiheista.**

Kuvassa 15 on vuokaavio ohjelman perustoiminnallisuudesta. Ensiksi handler-funktio, joka tarkastaa vuorokaudenajan ja jos aika on toimistoaikojen sisäpuolella, niin IVR-valikko suoritetaan. IVR valikon aikana numerovalinnat kerätään ja tallennetaan input_callback-funktiossa. Puhelun päättyessä viimeiseksi suoritetaan hangup_hook, riippumatta siitä, minkä takia puhelu päättyy.

JSON:n käskytys

IVR tottelee JSON-tiedostomuodossa olevia käskyjä, jotka määriteltiin suunnitteluvaiheessa. Esimerkkikoodi 1 on JSON-tiedoston rakenne. Komentoja ovat “CALL_START”, “GATHER_DIGITS”, “RECORD_AUDIO”, “CALL” sekä “HANGUP”. Kaikki komennot ovat alustettu “False”-arvoilla. Näin ollen jos jokin komento saa arvon True, niin IVR lähtee toteuttamaan kyseistä komentoa soittajalle.

```
menu:{
”CALL_START”:False,
”CLIENT”:{
”sipCallID”: session.getVariable('sip_call_id'),
"sipFromUser":checkPhoneNumber(session.getVariable('sip_from_user')),
"sipToUser":session.getVariable('sip_to_user'),
"sipPAI":session.getVariable('sip_P-Asserted-Identity')},
"PLAY_ANNOUNCEMENT":False,
"PLAY_FILE":{
            "SRC":"announcement_dir",
            "NAME":"announcement_X" },
            “RESULT” : “OK”,
            “GATHER_DIGITS”: False,
            “DIGIT” : 0
            “RECORD_AUDIO”:False,
            “CALL”: False,
            “HANGUP”: False,
            “CALL_INFO”:{
            “FORWARD”:”forwardNumber”,
            “HANGUP_CAUSE”:hangup_hook }
            }
 }
```

**Esimerkkikoodi 1. JSON:n tiedostorakenne.**

”CLIENT”-, ”PLAY_FILE”- ja ”CALL_INFO”-muuttujiin tallennetaan tietoja, joita kyseisessä istunnossa tarvitaan. Tällä tavalla saadaan pidettyä kaikki tarvittavat komennot ja tiedot yhdessä JSON-tiedostossa, mutta kuitenkin komennot ja tiedot ovat erillään. Result muuttuja on virheenhallintaa varten, jos se saa minkä tahansa muun arvon kuin ”OK”. Tällöin ohjelma siirtyy virhetilaan ja ryhtyy käsittelemään virhettä. Digit-muuttuja sisältää soittajan viimeisimmän valinnan.

Client-muuttuja psisältää ennalta määritetyt tiedot, jotka kerätään asiakkaasta ja sessiosta. PLAY_FILE- ja CALL_INFO-muuttujat taas sisältävät tietoa, jota tarvitaan ohjelman suorittamiseen. Näiden kahden tiedot voivat kuitenkin muuttua ohjelman suorituksen aikana.

IVR-Valikko

Ohjelman valikko koostuu monesta erillisestä ominaisuudesta, joita yhdistelemällä saadaan luotua erilaisia valikkoja asiakkaiden toiveiden mukaisesti.

```
if(menuJSON['CALL']==True):
            session.execute("bridge", forwNumber)
```

**Esimerkkikoodi 2. Soitonsiirto.**

Esimerkkikoodi 2 on IVR-valikon ensimmäinen osuus eli soitonsiirto. Ehtolauseessa tarkistetaan, onko komento ”CALL” true, ja jos niin on, ohjelma suorittaa käynnissä olevalle istunnolle soitonsiirron metodilla execute. Execute on metodi, jolla voidaan suorittaa erilaisia toimintoja ohjelman sisällä. Metodin sisällä bridge on execute-metodin parametri, jolla suoritetaan puhelun yhdistäminen. Parametri forwNumber sisältää numeron, johon puhelu halutaan yhdistää. Sen voi joko pyytää käyttäjältä, tai se voidaan tallentaa ohjelmaan valmiiksi.

Toinen ominaisuus on tiedotteen soittaminen. Tiedotteen soittaminen toimii samalla periaatteella kuin soiton yhdistäminen. Ohjelma tarkistaa, onko muuttuja ”PLAY_ANNOUNCEMENT” tosi, ja jos se on tosi, lähdetään toistamaan tiedotetta. Tiedotteen toistossa ensin luodaan tiedostokahva, joka tarkistaa että tiedote on olemassa.

```
f = open(soundsDirectory+"/"+fileToPlay".wav")
```

Kahvan luomisen jälkeen voidaan soittaa tiedote session.streamFile()-metodilla.

```
session.streamFile(soundsDirectory+"/"+fileToPlay+".wav")
```

Kun äänitiedosto on toistettu loppuun, siirrytään koodissa seuraavaan toimintoon. Tiedotteen soittamisen aikana voidaan myös kerätä DTMF-valintoja, mutta siihen käytetään session luokan eri metodia nimeltä playAndGetDigits. DTMF-valintojen keräystä varten on luotava oma muuttuja, johon ne tallennetaan.

```
digits=session.playAndGetDigits(1,5,2,4000,”#”,”menu.wav”,invali
d,”[123]”)
```

PlayAndGetDigits-metodin ensimmäinen parametri tarkoittaa kuinka monta numeroa minimissään on annettava, ja toinen parametri tarkoittaa enimmäismäärää jonka voi antaa. Metodin kolmas parametri kertoo, kuinka monta kertaa soittaja voi antaa väärän valinnan, ennen kuin puhelu katkaistaan. Neljännessä parametrissa määritellään aikakatkaisu millisekunneissa ja viidennessä lopetusmerkki. Lopetusmerkkiä painettaessa metodi tietää, että soittaja on tehnyt valinnan ja voi siirtyä seuraavaan toimintoon. Kuudes ja seitsemäs parametri ovat palvelimella olevia äänitiedostoja, joista ”menu.wav” toistetaan soittajalle onnistuneen valinnan jälkeen, kun taas ”invalid.wav” silloin kun soittaja tekee virheellisen valinnan. Viimeinen parametri kertoo sallitut numerot, jotka soittaja voi antaa.

## 4.4 Työn kulku

Työskentely alkoi FreeSwitchin opettelemisesta. Opetteluun käytin apuna FreeSwitchistä kirjoitettua kirjaa nimeltä FreeSwitch 1.2. kirja on kirjoitettu yhteistyössä FreeSwitchin suunnittelijoiden kanssa [1]. Kirjassa luodaan esimerkkitoimintoja ja kerrotaan FreeSwitchin eri moduuleista, joista ohjelmoijan on hyvä olla tietoinen.

Ohjelmoinnin aloittamisen ja eri metodien opettelemisen jälkeen kehitys aloitettiin yksi ominaisuus kerrallaan. Ominaisuuksien valmistuttua niitä testattiin soittamalla FreeSwitchille määritettyyn numeroon. Jos ominaisuus läpäisi yksinkertaisen yksikkötestin, niin seuraavan ominaisuuden kehittäminen voitiin aloittaa. Ohjelman laadun ylläpitämiseksi ja yllättävien tilanteiden välttämiseksi testausta pyrittiin tekemään mahdollisimman paljon kehityksen rinnalla.

# 5 Ohjelman tarkastus

Valmistunut ohjelma

Ohjelman saatiin valmiiksi testausta varten aikataulun mukaisesti ja ohjelmalta vaaditut ominisuudet täytettiin. Valmistunutta ohjelmaa testataan erilaisilla testausmenetelmillä, ennen kuin se otetaan tuotantoon ja käyttöön. Kehittämisen aikana ohjelmalle ja sen funktioille on tehty pienimuotoista yksikkötestausta. Ohjelmaa on myös yritetty kaataa puhelinsoitoilla, mutta sovellus ei ole kaatunut. Koska ohjelman ominaisuudet itsessään ovat yksinkertaisia eikä tietoa tarvitse liikutella isoja määriä, ohjelma on vakaampi kuin Ainacomin entinen puhelinvalikkosovellus.

Valmistuneeseen ohjelmaan voidaan myös tuoda lisäominaisuuksia ilman, että palvelua tarvitsee käynnistää uudelleen.

Ohjelmoinnissa ilmenneet ongelmat

Ongelmatilanteet ohjelmoinnin aikana liittyivät pääosin DTMF:n valintojen keräämiseen. FreeSwitch tunnisti satunnaisesti numeroita väärin tai ei tunnistanut, että numero annettiin. FreeSwitchin oma moduuli kuitenkin sisältää suhteellisen hyvän virhehallinnan, esimerkiksi playAndGetDigits-funktio koettaa niin monta kertaa uudestaan kerätä valintaa kunnes se on annettu. Menetelmä toimii tilanteissa joissa ei tunnisteta koko valintaa. Jos yksi kerta jää tunnistamatta, niin voidaan määrittää, että kokeillaan neljä kertaa uudestaan. Ohjelman testausta jatketaan lisäämällä käyttäjämäärää vähän kerrallaan, jotta voidaan huomata ilmeneekö ongelmia DTMF-valintojen kohdalla lisää.

DTMF-valintojen virheellinen tunnistaminen tuo myös ongelmia tiedostontallentamisessa. Soittaja voi haluta tallentaa tiedoston palvelimelle suoraan puhelimella, ja tiedostonimi oli suunnitelmien mukaan tarkoitus tallettaa numerovalinnoilla. Virheellisistä ja puutteellisista DTMF-valinnoista johtuen päädyttiin ratkaisuun tallentaa tiedostot puhelinnumeron perusteella.

TTS:n toistamisen kanssa ongelmana oli tuen puuttuminen suomen kielelle. Tuen puuttumisen vuoksi jäädään odottamaan FreeSwitch-alustan kehitystä.

Ohjelman kehittämismahdollisuudet

Ohjelmaa tullaan jatkossa kehittämään vielä lisää asiakkaiden tarpeiden mukaan. Sanojen ja äänentunnistus on yksi ominaisuus, jota lähdetään kehittämään kun IVR-sovelluksen kysyntä kasvaa. Toinen kehittämiskohde on sähköpostin ja IVR:n yhdistäminen, esimerkiksi Outlookin tilamuunnokset DTMF-valintojen perusteella tai lukemattomien sähköpostien tilatiedot ja tieto, keneltä kyseiset tiedot ovat.

FreeSwitch on jatkuvan kehityksen kohteena itsessään, ja kun alustaan tuodaan uusia ominaisuuksia, myös sovelluksessa voidaan hyödyntää kehitystä.

# 6 Yhteenveto

Insinöörityössä suunniteltiin ja toteutettiin toimiva ohjelma, joka osaa siirtää puhelun, katkaista puhelun, kerätä käyttäjän antamia valintoja, soittaa tiedotteen ja nauhoittaa puhelun. Sovelluksen tavoite on korvata tuotannossa oleva IVR-järjestelmä hyväksyntätestaamisen jälkeen.

Sovellus kirjoitettiin kokonaan käyttämällä Notepad++-tekstieditoria ja käyttämällä FreeSwitchin omaa ohjelmointikielen kääntäjää. Puheluiden testaamiseen käytettiin sekä SIP-protokollaa käyttävää lankapuhelinta, että mobiiliverkossa toimivaa Nokia Lumia 925-älypuhelinta.

Insinöörityön tekeminen oli haasteellinen, mutta opettava prosessi, jonka aikana tuli opittua Internetissä toimivista puheluista ja Internet-puheluiden protokollista. Lisäksi puhelinvalikoiden yleistyminen ja niiden toteutus kiinnittivät huomioni, koska hyvin usea laitos ja yritys kuitenkin on automatisoinut asiakaspalvelunsa IVR-järjestelmillä. Työtä tehdessäni näiden järjestelmien takana toimivat tekniikat selvisivät.

## Lähteet


1 Anthony Minessale, Michael S Collins, Darren Schreiber & Raymond Chandler 2013 FreeSWITCH 1.2.

2 Dual-Tone Multi-Frequency (DTMF) Signal Detection. 2004. Verkkodokumentti. Mathworks. http://se.mathworks.com/products/demos/signaltlbx/dtmf/dtmfdemo.html. Luettu 17.12.2014.

3 K.Banerjee. SIP (Session Initiation Protocol) Introduction. 2005. Verkkodokumentti. http://www.siptutorial.net/SIP. Luettu 18.12.2014.

4 How Interactive Voice Response (IVR) Works. Verkkodokumentti. HowStuffWorks. http://electronics.howstuffworks.com/interactive-voice-response1.html. Luettu 21.12.2014.

5 softswitch. Verkkodokumentti. Voip-info. http://www.voip-info.org/wiki/view/softswitch. Luettu 21.12.2014.

6 DTMF Issues & VoIP. Verkkodokumentti. VoipMechanic. http://www.voipmechanic.com/dtmf-issues.htm. Luettu 21.12.2014.

7 What is Asterisk. 2014. Verkkodokumentti. Asterisk. http://www.asterisk.org/get-started. Luettu 25.12.2014.

8 Mod Python. 2013. Verkkodokumentti. FreeSwitch. https://wiki.freeswitch.org/wiki/Mod_python. Luettu 27.12.2014.

9 Performance testing and configurations. 2014. Verkkodokumentti. FreeSwitch. https://wiki.freeswitch.org/wiki/Performance_testing_and_configurations. Luettu 28.12.2014.

10 Margaret Rouse. Interactive Voice Response (IVR). 2008. Verkkodokumentti. http://searchcrm.techtarget.com/definition/Interactive-Voice-Response. Luettu 16.12.2014

**Liitteet**

Liite 1 sisältää ohjelman käyttämän IVRMenu funktion lähdekoodin.

```
def IVRMenu(session, menuJSON):
    callerNum = menuJSON[”client”][”sipFromUser”] #caller
phonenum
    invalid = ”ivr/ivr_that_was_invalid_entry.wav”
    bongSound="tone_stream://v=-
;%(100,0,941.0,1477.0);v=7;>=2;+=.1;%(1000, 0, 640)"
# define sound files
while(session.ready()): # when session is ready
    if(menuJSON[“call”] == True): #callforward method
        try:
            menuJSON[“call”] = False
            forwNumber = menuJSON[“callInfo”][“forward”]
          session.execute(“bridge”,”sofia/internal/”+for-
wNumber+”@”+server)
            menuJSON[“result”]=”OK” # all ok flag for external
system
        except:
            menuJSON[“result”]=”NOK” # error flag for external
system
            payload = json.dumps(menuJSON)
            headers = {‘content-type’:’application-json’}
            r=requests.post(url,payload,headers)
    if(menuJSON[“play_announcement”] == True): #announcement
method
```

```
            try:
                menuJSON[“play_announcement”] = False
                fileToPlay = menuJSON[“play_file”][“name”]
                soundsDird = menuJSON[“play_file”][“src”]
                f = open(soundsDir+”/”+fileToPlay+”.wav”)
                session.stream-
File(soundsDird+”/”+fileToPlay+”.wav”)
               f.close()
            except:
                menuJSON[“result”]=”NOK”
                payload = json.dumps(menuJSON)
                headers = {‘content-type’:’application-json’}
                r = requests.post(url,payload,headers)
                session.streamFile(“file_not_found.wav”)
if(menuJSON[“gather_digits”] == True) # gather DTMF method
            try:
                menuJSON[“gather_digits”] = False
                fileToPlay = menuJSON[“play_file”][“name”]
                digits = session.playAndGetDig-
its(1,5,3,7000,”#”,”sounds_dir/main_menu.wav”, invalid,
“[1234567890]”)
                menuJSON[“digits”] = digits
                payload = json.dumps(menuJSON)
                headers = {‘content-type’:’application-json’}
                r = requests.post(url,payload,headers)
                recvData = r.text
                recvData = recvData.split(“[!]”)
                httpResponse = recvData[1]
                menuJSON = json.loads(recvData[0])
            except:
                menuJSON[“result”] = “NOK”
                payload = json.dumps(menuJSON)
                headers = {‘content-type’:’application-json’}
                r = requests.post(url,payload,headers)
if(menuJSON['MENU']['RECORD_AUDIO'] == True): #Record audio
method
```

```
    try:
        menuJSON['MENU']['RECORD_AUDIO'] = False
        isAudioAccepted = False
        while(session.ready() and not isAudioAccepted):
          session.streamFile("phrase:voicemail_record_message")
          session.streamFile(bong_sound)
          filename=sesssion.getVaria-
ble('sounds_dir')+"/"+callerNum+".wav"
         filename = str(filename)
         session.recordFile(filename,300,100,10)
         listen = True
         while(session.ready() and listen):
           session.streamFile(filename)
         recDigits = session.playAndGetDigits(1, 5, 2,
4000,"#", "phrase:voicemail_record_file_check:1:2:3", inva-
lid,"[1234567890]")
         if (recDigits == "1"):
           listen = True
           isAudioAccepted = False
           session.execute("sleep","500")
        elif (recDigits == "2"):
           listen = False
           isAudioAccepted = True
           session.streamFile("voicemail/vm-message.wav")
           session.execute("sleep","100")
           session.streamFile("voicemail/vm-saved.wav")
           session.execute("sleep","1500")
       elif (recDigits == "3"):
           listen = False
           isAudioAccepted = False
           session.execute("sleep","500")
           payload = json.dumps(menuJSON)
           headers = {'content-type':'application/json'}
           r = requests.post(url,data=payload,headers=headers)
   except:
       menuJSON['MENU']['RESULT'] = "NOK"
```

```
        payload = json.dumps(menuJSON)
        headers = {'content-type':'application/json'}
        r = requests.post(url,data=payload,headers=headers)

if(menuJSON['MENU']['HANGUP'] == True): #end call method
   menuJSON['MENU']['HANGUP'] = False
   session.hangup()
```