



OULUN YLIOPISTO  
UNIVERSITY of OULU

SÄHKÖ- JA TIETOTEKNIIKAN OSASTO  
TIETOTEKNIIKAN KOULUTUSOHJELMA

## **ELEOHJAUS MATKAPUHELIMESSA**

Työn tekijä \_\_\_\_\_  
Inkeroinen Tuomas

Työn valvoja \_\_\_\_\_  
Riekki Jukka

Hyväksytty \_\_\_\_\_/\_\_\_\_\_ 2008

Arvosana \_\_\_\_\_

**Inkeroinen T. (2007) Eleohjaus matkapuhelimessa.** Oulun yliopisto, sähkö- ja tietotekniikan osasto. Diplomityö, 46 s.

## **TIIVISTELMÄ**

**Tässä työssä tutkitaan matkapuhelinten erilaisia käyttöliittymiä. Käsiteltäviä käyttöliittymiä ovat perinteinen käyttöliittymä, puheohjaus sekä fyysiset käyttöliittymät. Tutkittavia fyysisiä käyttöliittymiä ovat kosketusnäyttö, fyysinen valinta ja eleohjaus. Työssä toteutetaan yleiskäyttöinen eleohjausjärjestelmä, jota voi käyttää matkapuhelimen sisäisten sovellusten sekä ulkopuolisten HTTP-pyyntöillä ohjattavien sovellusten hallintaan. Työssä tutkitaan myös eri ohjelmointikielillä ohjelmoitujen sovellusten välisen kommunikoinnin toteuttamista. Sovellusten välinen kommunikointi toteutetaan TCP- ja UDP-pistokkeiden avulla. Kehitettävää järjestelmää voi käyttää millä tahansa ohjelmointikielellä toteutettu sovellus. Lisäksi työssä toteutetaan kaksi elehallintajärjestelmää käyttävää sovellusta: puhelimen elehallintasovellus sekä Internet-sovellusta ohjaava elehallintasovellus. Kehitettävien sovellusten osalta pääpaino on puhelimen elehallintasovelluksessa, jolla ohjataan puhelimen sisäistä toimintaa. Tälle sovellukselle tehdään käytettävyydestä.**

**Avainsanat: käyttöliittymä, fyysinen käyttöliittymä, sovellusten välinen kommunikointi, käytettävyys**

**Inkeroinen T. (2007) Gesture control on a mobile device.** University of Oulu, Department of Electrical and Information Engineering. Master's Thesis, 466 p.

## **ABSTRACT**

**This thesis researches different user interfaces of mobile phones. Handled user interfaces are traditional user interface, speech control and physical user interfaces. Discussed physical user interfaces are touch screen, physical selection, and gesture control. Universal gesture control system is developed in this work. This system can be used to control internal applications of mobile phone and remote applications controlled by HTTP-requests. Also inter process communication between applications implemented by different programming languages are studied. Inter process communication is implemented with TCP and UDP sockets. Universal gesture control system can be utilized by application implemented by any programming language. In addition two applications that utilize gesture control system are developed: phone gesture control application and gesture application which controls Internet-application. The main focus is in phone gesture control application which manages internal functionality of mobile phone. Usability testing is being done for this application.**

**Key words: gesture control, user interface, inter process communication, usability.**

# SISÄLLYSLUETTELO

TIIVISTELMÄ

ABSTRACT

SISÄLLYSLUETTELO

ALKULAUSE

LYHENTEIDEN JA MERKKIEN SELITYKSET

1.	JOHDANTO.....	7
2.	MATKAPUHELIMEN KÄYTTÖLIITTYMÄ .....	8
2.1.	Perinteinen käyttöliittymä .....	8
2.2.	Puheohjaus.....	10
2.3.	Fyysiset käyttöliittymät .....	11
2.3.1.	Kosketusnäyttö .....	11
2.3.2.	Fyysinen valinta .....	12
2.3.3.	Eleohjaus .....	13
3.	ELEHALLINTAJÄRJESTELMÄ .....	15
3.1.	Toteutettavuusanalyysi .....	15
3.1.1.	Eri kielten soveltuvuus toteutukseen.....	15
3.1.2.	Ohjelmien välinen kommunikointi.....	16
3.1.3.	Puhelimen hallinta.....	17
3.1.4.	Kohdelaitteiden ominaisuudet.....	17
3.2.	Vaatimusmäärittely .....	18
3.3.	Suunnittelu ja toteutus .....	21
3.3.1.	Arkkitehtuuri .....	21
3.3.2.	Sääntöryhmät.....	22
3.3.3.	Järjestelmän käyttäytyminen .....	25
3.3.4.	Luokkakaaviot.....	28
3.3.5.	Toteutus .....	31
4.	TESTAUS .....	37
4.1.	Yksikkö- ja komponenttitestaus .....	37
4.2.	Käytettävyydestaus .....	38
5.	POHDINTA.....	41
6.	YHTEENVETO .....	43
7.	LÄHTEET .....	44

## ALKULAUSE

Tämä lopputyö on tehty Tietokonetekniikan laboratoriolle UbiLife-projektissa. Kiitokset kaikille projektin jäsenille hyvästä yhteistyöstä. Erityisesti kiitos eleentunnistusjärjestelmän kehittäneelle Mikko Kauppilalle sekä työn valvojalle Jukka Riekille. Lisäksi kiitokset myös työn toiselle tarkastajalle Tapio Seppäselle.

Omistan tämän työn rakkaalle vaimolleni sekä maailman parhaalle Tapio-pojalle, jonka syntyminen samoihin aikoihin työn aloittamisen kanssa ei ainakaan edistänyt työn valmistumista.

Oulussa, toukokuun 9. päivänä 2008

Tuomas Inkeroinen

## LYHENTEIDEN JA MERKKIEN SELITYKSET

API	Application programming interface, ohjelmointirajapinta
ARM9	Symbian-laitteissa käytettävä prosessori
ARM11	Symbian-laitteissa käytettävä prosessori
CLDC	Connected, Limited Device Configuration, J2ME-laitekonfiguraatio
HMM	Hidden Markov model, tilastollinen luokittelumenetelmä
HTTP	Hypertext Transfer Protocol, hypertekstin siirtoprotokolla
J2ME	Java 2 Platform Micro Edition, rajoitettujen laitteiden Java-sovellusympäristö
MB	megatavu
MHz	megahertsi
MIDP	J2ME-laiteprofiili
NFC	Near Field Communication, radiotaajuinen etätunnistustekniikka
OS	Operating System, käyttöjärjestelmä
PC	Personal Computer, henkilökohtainen tietokone
QWERTY	yleisin kirjoituskonetyyppinen näppäimistöasettelu
RAM	Random access memory, käyttömuisti, keskusmuisti
RF	Radio Frequency, radiotaajuus
RFID	Radio frequency identification, radiotaajuinen etätunnistus
S60	Nokian kehittämä matkapuhelinohjelmistoalusta
TCP	Transmission Control Protocol, luotettava, yhteydellinen tietoliikenneprotokolla
T9	Text on 9 keys, numeronäppäimillä toimiva ennakoiva tekstinsyöttöjärjestelmä
UDP	User Datagram Protocol, epäluotettava, yhteydetön tietoliikenneprotokolla
URL	Uniform Resource Locator, merkkijono, joka kertoo tiedon sijainnin
WWW	World Wide Web, internetin hypertekstijärjestelmä

# 1. JOHDANTO

Matkapuhelinvalmistajat ovat kehittäneet markkinoille yhä pienempiä laitteita, joissa on entistä monipuolisemmin ominaisuuksia ja sovelluksia. Monilta ominaisuuksiltaan matkapuhelin muistuttaa jo pöytäkoneita. Tällaisten laitteiden käyttäminen alkaa olla hidasta perinteisesti numeronäppäinten ja valikkojen avulla. Käyttöliittymien kehityksellä on ratkaiseva rooli monipuolisten toimintojen mielekkäässä ja sujuvassa käyttämisessä. Perinteisten käyttöliittymien lisäksi matkapuhelimissa käytetäänkin puheohjausta ja fyysisiä käyttöliittymiä, kuten kosketusnäyttöjä ja RFID-tunnistimiin perustuvaa fyysistä valintaa. Koska elekieltä käytetään jatkuvasti päivittäisessä elämässä, eleisiin perustuva vuorovaikutus voisi olla käyttäjille mieluinen ja intuitiivinen tapa hallita myös mobiililaitettaan. [1, 2]

UbiLife-projektissa, jonka osana tämä työ on tehty, kehitetään teknologiaa helpottamaan liikkuvan käyttäjän tilanteen tunnistamista, tilanteeseen sopeutuvien ja helppokäyttöisten sovellusten rakentamista sekä älykkään ympäristön resurssien hallintaa. Muita aiheita ovat palveluiden haku, paikkatiedon hallinta, komponenttipohjaiset välitason ohjelmistot, sisältöpohjainen reititys ja fyysiset käyttöliittymät. [3]

Tässä diplomityössä tutkitaan mobiililaitteen erilaisia käyttöliittymiä. Työn pääpaino on kiihtyvyydataan perustuvassa eleohjauksessa. Tässä menetelmässä puhelimen liikkeitä tunnistetaan laitteen kiihtyvyyssanturien tuottamasta datasta. Tiettyä laitteen liikerataa vastaamaan sovitaan tietty toiminto. Esimerkiksi kun käyttäjä piirtää laiteella viivan alaspäin, laite avaa kalenterin nykyisen päivän näkymään. Kun käyttäjä piirtää ympyrän, laite voi esimerkiksi näyttää ”asetta herätys”-dialogin käyttäjälle.

Tässä diplomityössä on tavoitteena kehittää puhelimeen yleiskäyttöinen elehallintapalvelu. Vaatimuksena on toteuttaa palvelu, jota voi helposti käyttää mikä tahansa sovellus. Lisäksi toteutetaan kaksi sovellusta, jotka käyttävät tätä palvelua. Toisella sovelluksella ohjataan puhelimen sisäistä toimintaa ja toisella ulkoista HTTP-pyyntöillä ohjattavaa internetsovellusta. Toteutus tehdään Nokia Sport 5500- ja Nokia N95-matkapuhelimille.

Puhelimen sisäistä toteutusta ohjaavalle sovellukselle tehdään käytettävyydestä testauksia. Testaustuloksista pyritään selvittämään parantaako kehitetty uudenlainen käyttöliittymä puhelimen käytettävyyttä. Testaus aloitetaan mahdollisimman varhain ja testauksen tuloksia pyritään hyödyntämään myös sovelluksen kehitystyössä.

Mikko Kauppila on toteuttanut algoritmit, joilla eleet tunnistetaan [4]. Algoritmit on toteutettu Javalla, mutta mahdollisesti osa toiminnallisuudesta on helpompi toteuttaa Pythonilla. Joidenkin asioiden toteuttamiseen tarvitaan todennäköisesti myös Symbian C++-ohjelmointia. Työssä tullaan arvioimaan, mitkä asiat kannattaa toteuttaa milläkin kielellä. Työssä tutkitaan myös eri kielillä toteutettujen komponenttien välisen kommunikoinnin toteuttamista S60-matkapuhelimessa.

## 2. MATKAPUHELIMEN KÄYTTÖLIITTYMÄ

Matkapuhelin muistuttaa nykyään monessa suhteessa ominaisuuksiltaan pöytäkoneetta. Merkittävimmät eroavaisuudet ovat laitteiden fyysinen koko, tiedon syöttämisen ja lukemisen tehokkuus sekä muistin määrä ja suoritusteho. Pöytäkoneeseen kuuluu useimmiten kohtalaisen suuri näyttö sekä täysikokoinen näppäimistö ja hiiri. Matkapuhelimessa taas näytön fyysinen koko on pieni ja näppäimistö koostuu useimmiten numeronäppäimistä sekä muutamasta erikoisnäppäimestä.

Myös laitteiden käyttötilanteet poikkeavat toisistaan. Pöytäkoneetta käytetään yleensä paikoillaan ja tutussa paikassa, kun taas matkapuhelinta voidaan käyttää missä ja milloin tahansa. Matkapuhelinta käytetään usein liikkeessa ja tilanteissa, joissa tehdään myös muita asioita yhtä aikaa. Niinpä matkapuhelimen käyttöliittymän tulisi vaatia käyttäjältä mahdollisimman vähän huomiota. [5, 6]

Käyttöliittymiä suunniteltaessa tärkeimpiä asioita ovat helppokäyttöisyys ja oppimisen helppous. Helppokäyttöisyys tarkoittaa nopeutta ja tehokkuutta, jolla tehtävästä suoriudutaan. Oppimisen helppous taas tarkoittaa sitä, kuinka vaistonvarainen ja looginen menetelmä on. Näppäinyhdistelmiin sidotut pikavalinnat ovat esimerkki helppokäyttöisestä, mutta vaikeasti opittavasta menetelmästä. Perinteinen tapa tehdä sama asia valikkojen kautta on esimerkki vaikeasti käytettävästä, mutta helposti opittavasta asiasta. [6]

Mobiililaitteen käytön tulee olla intuitiivista. Tällöin käyttäjän ei tarvitse miettiä, mitä hän on tekemässä tai muistaa käyttämiään komentoja [7]. Käytön on oltava helposti opittavaa uudelle käyttäjälle sekä tehokasta käyttää pitkään puhelinta käyttäneelle. Tärkeintä olisi keskittyä siihen, että useimmiten käytettävien toimintojen käytettävyyden on mahdollisimman hyvä. Nykyaikaisessa puhelimessa on niin paljon ominaisuuksia, että kaikkien käyttöä ei voida optimoida.

### 2.1. Perinteinen käyttöliittymä

Perinteisesti matkapuhelinta hallitaan näppäimistöllä, jossa on numeronäppäimet ja muutamia erikoisnäppäimiä. Useimmat toiminnot tehdään navigointinäppäinten ja kahden valintänäppäimen avulla. Kirjaimet syötetään numeronäppäimillä, joko ennakoivan tekstinsyötön avulla tai kirjoittaen numeroihin liitettyjä kirjaimia useiden näppäinpainallusten avulla.

Navigointinäppäimet ovat nykyisessä puhelimissa lähes poikkeuksetta nelisuuntaisia. Navigointinäppäimillä voidaan vaihtaa valittuna olevaa kohdetta työpöydällä ja erilaisissa valikoissa. Yleensä navigointi toteutetaan yhden kelluvan näppäimen avulla, mutta erityisesti varhaisemmissa malleissa käytettiin erillisiä näppäimiä.

Navigointia ja valikkorakennetta kehittämällä on saatu parannettua matkapuhelinten käytettävyyttä. Muutoin valikkorakenteista olisi tullut entistä pidempiä ja syvempiä matkapuhelinten toimintojen jatkuvasti lisääntyessä. Aikaisemmin matkapuhelimissa oli tyypillisesti pieni tekstipohjainen näyttö, jolle mahtui kerrallaan vain yksi tai muutamia rivejä tekstiä. Näissä puhelimissa

valikkorakenne oli kaksisuuntainen. Nykyään valikot ovat yleensä kaksiulotteisia ja niissä voidaan yhdellä sivulla esittää vähintään yhdeksän sovelluskuvaketta sekä mahdolliset niihin liittyvät tekstit. Nykyään näytöt mahdollistavat grafiikan käytön ja ne ovat poikkeuksetta värillisiä, lisäksi näyttöjen koot sekä erityisesti resoluutiot ovat huomattavasti kasvaneet. Perustilassa puhelimesta on usein myös eräänlainen työpöytä, josta voi käynnistää pikavalinnoiksi määritettyjä sovelluksia, ja jossa jotkin sovellukset näyttävät tietojaan.

Valintanäppäimet sijaitsevat välittömästi näytön alapuolella tai vieressä ja niiden toiminnallisuus riippuu aina puhelimen tilasta. Valintanäppäimen toiminto on ilmaistu näytön valintanäppäinalueella. Kuvassa 1 toiminnot ovat ”Valinnat” ja ”Poistu”. Vakiintuneen käytännön mukaisesti vasemman puoleinen valintanäppäin on ensisijainen toimintonäppäin ja sillä yleensä annetaan hyväksyvä vastaus dialogiin. Oikean puoleisella valintanäppäimellä useimmiten poistutaan sovelluksesta tai annetaan hylkäävä vastaus. [6]

Muita usein käytettyjä näppäimistön painikkeita ovat puhelunhallintapainikkeet, äänenvoimakkuuden säätöpainikkeet sekä virtakytkin. Toimintojen käyttäminen vaatii usein monta peräkkäistä valintaa, kuten näppäinlukon avaaminen, sovelluksen hakeminen puhelimen valikoista ja halutun toiminnon valitseminen sovelluksen valikoista.



Kuva 1. Perinteinen käyttöliittymä.

Tekstin syöttäminen mobiililaitteessa jää aina jälkeen pöytäkoneen tiedonsyöttökyvykkydestä. Perinteinen kirjoitustapa numeronäppäimillä on ”triple tap”-syöttötapa. Tässä menetelmässä vuorotellaan useilla painalluksilla näppäimeen yhdistettyjen kirjainten välillä. Viimeisin valinta näytetään näytöllä. Tämä tekstinsyöttötapa on helppo oppia, mutta hidasta. Ennakoiva tekstinsyöttö T9 ennustaa tulevaa sanaa jatkuvasti. Tässä yhtä kirjainta varten tarvitaan keskimäärin yksi näppäimen painallus. T9 vaatii jonkin verran enemmän opettelua, mutta sen avulla päästään huomattavasti nopeampaan tekstin syöttämiseen. T9 onkin tullut erittäin suosituksi menetelmäksi. Vielä tehokkaampaan tekstin syöttämiseen päästään

QWERTY-näppäimistöllä, jossa jokaiselle kirjaimelle on oma näppäimensä. Pienelläkin QWERTY-näppäimistöllä kirjoittaminen on selvästi nopeampaa kuin numeronäppäimillä. Kirjoittamisen hitaudesta johtuen matkapuhelimen käyttöliittymässä olisi pyrittävä siihen, että asiat voisi mieluummin valita kuin kirjoittaa. On tärkeää, että näppäimet on toteutettu siten, että käyttäjä saa palautteen näppäimen painalluksesta. Useimpien laitteiden näppäimistöä käytettäessä näppäimen painalluksen tuntee ja kuulee. [6]

## 2.2. Puheohjaus

Nykyään useissa puhelimissa on puheohjaustoiminnallisuus. Puheohjaus soveltuu matkapuhelimen käyttöön erityisen hyvin, koska matkapuhelinta käytetään usein tilanteissa, joissa puhelinta ei voida katsoa eikä koskea. Tällainen tilanne on esimerkiksi matkapuhelimen käyttäminen ajon aikana.

Tyypillisesti puheohjauksella voi käyttää toimintoa, jolla voi soittaa sanomalla vastaanottajan nimen. Tämä toiminto edellyttää kunnolla toimiakseen pitkien nimien, sekä etu- että sukunimen käyttöä. Toinen usein käytetty toiminto, missä matkapuhelimessa voi hyödyntää puheohjausta on sovellusten käynnistäminen äänikomennoilla.

Eräänä syynä siihen, että puheohjausta ei käytetä nykyistä enemmän on puheentunnistuksessa tapahtuvat virheet. Tätä pidetään jopa tärkeimpänä syynä puheohjauksen vähäiseen käyttöön. Puheentunnistuksen tehokkuuteen vaikuttavat esimerkiksi käyttäjän ominaisuudet, tilanne ja ympäristö. Huonosti toimiva puheentunnistus voi vaatia jopa enemmän käyttäjän keskittymistä kuin perinteinen käyttötapa. [8] Automaattinen puheentunnistus on erittäin vaativa menetelmä ja se perustuu yleensä Markovin piilomalleihin (HMM) [9].

Toinen merkittävä syy siihen, että puheohjausta ei käytetä nykyistä enempää on puheohjauksen sosiaalinen hyväksyttävyyys. Ihmiset ovat haluttomia puhumaan koneelle erityisesti yleisellä paikalla. Erilaisten laitteiden kanssa puhumisesta saattaa ajan myötä tulla tavallista ja siten myös yleisesti hyväksyttyä. [10]

Puheen hyödyntäminen palautteen antamisessa on erittäin käyttökelpoinen tapa käytettäessä fyysistä käyttöliittymää (kts. seuraava luku). Perinteisessä käyttöliittymässä käyttäjän on yleensä helppoa havaita, milloin esimerkiksi valikon valinta on tehty onnistuneesti. Myös fyysistä käyttöliittymää käytettäessä tarvitaan palautetta, milloin tehty toiminto on hyväksytty tai hylätty. Puhetta käyttämällä voidaan helposti antaa yksikäsitteinen palaute suoritetusta toiminnosta. Toisaalta synteettisesti tuotettua puhetta pidetään usein ärsyttävänä [8].

### 2.3. Fyysiset käyttöliittymät

Uudentyyppinen käyttöliittymä antaa lisäarvoa ihmisen ja koneen väliseen vuorovaikutukseen ainoastaan, jos se parantaa käytettävyyttä perinteiseen käyttöliittymään verrattuna. Käyttöliittymä mielletään sitä intuitiivisemmaksi, mitä paremmin käyttäjä pystyy ohjaamaan ja hallitsemaan laitetta inhimillisillä perustoiminnoilla, kuten koskettamalla tai kättä ojentamalla. [7] Fyysisten käyttöliittymien avulla on mahdollista tehdä matkapuhelimen käytöstä mahdollisimman intuitiivisia.

#### 2.3.1. Kosketusnäyttö

Koskettaminen on luonnollinen tapa käyttää tietokoneita. Kosketusnäyttöjä onkin hyödynnetty erityisesti julkisten tilojen tietokoneissa, kuten lipunmyyntiautomaateissa ja infopisteissä. Viime aikoina kosketusnäytöistä on tullut erittäin suosittuja myös matkapuhelinten käyttöliittymissä. Matkapuhelimessa kosketusnäyttöä käytetään joko osoitinkynän tai sormen kosketuksella.

Varsinaisena kosketusnäyttöjen läpimurtona voi pitää Applen iPhone-puhelinmallin julkistamista vuonna 2007. Kosketusnäytöllisiä puhelimia on markkinoilla kuitenkin ollut aikaisemminkin kuten jo vuonna 1993 BellSouthin ja IBM:n julkaisema Simon tai 2004 julkistettu Nokian 7710, mutta aikaisemmin niiden suosio on jäänyt vaatimattomaksi. iPhoneen synnyttämän ylimainonnan myötä kosketusnäyttötekniikan käyttö matkapuhelimeissa on lähtenyt voimakkaaseen kasvuun. Vuonna 2012 kosketusnäyttöjä arvioidaan olevan jopa 40 prosentissa matkapuhelimeissa. [11]

Kosketusnäyttö soveltuu erityisen hyvin pieniin mobiililaitteisiin, koska sen avulla voidaan toteuttaa myös näppäimistö. Tällaisella ratkaisulla voidaan pienikokoiseen matkapuhelimeen saada kohtalaisen suurikokoinen näyttö. Kosketusnäytölle voidaan toteuttaa helposti myös QWERTY-näppäimistö.

Kosketusnäyttöjen pinnalla esiintyvä pöly, tahrat ja naarmut voivat aiheuttaa ongelmia kosketuksen havaitsemisessa. Kosketusnäytöt ovat myös huomattavasti kalliimpia ja kuluttavat enemmän energiaa kuin tavalliset näytöt. Lisäksi, toisin kuin tavallista näppäimistöä käytettäessä, kosketusnäytöllä käyttäjä ei tunne alas painuvaa näppäintä. Kosketusnäyttöä käytettäessä onkin tärkeää tarjota käyttäjälle jonkinlainen palaute hyväksytystä syötteestä. Annettu palaute voi myös olla asiayhteydestä riippuvaa, esimerkiksi numeronäppäimen painaminen voisi tuntua erilaiselta kuin äänenvoimakkuuden säätö. Erilaisilta tuntuvia palautteita saadaan aikaan tuottamalla värinä tai töytäisy tietyille näytön osalle. [11]

### 2.3.2. *Fyysinen valinta*

Fyysinen valinta perustuu tiedon liittämiseen ympäristön kohteisiin. Tieto voidaan liittää esimerkiksi visuaalisilla tai RFID-tunnisteilla. Fyysinen valinta mahdollistaa luontevan ja nopean tavan tehdä asioita pyyhkäisemällä, osoittamalla tai koskettamalla ympäristöstä löytyviä tunnisteita. Fyysiseen valintaan perustuvat käyttöliittymät voidaan luokitella ScanMe-, PointMe- tai TouchMe-paradigmoilla. [12]

Pyyhkäisyyn perustuvaa fyysistä valintaa kutsutaan ScanMe-paradigmaksi. Tässä menetelmässä fyysinen kohde valitaan matkapuhelimen näytöltä. Kun käyttäjä tulee kiinnostavaan ympäristöön, hän voi matkapuhelimellaan selailta ympäristöstä löytyviä tunnisteita. Esitettyjen tietojen tulee olla sellaisia, että käyttäjä voi ymmärtää, mihin fyysiseen kohteeseen näytöllä esitetty kohde liittyy. Tämän tyyppinen käyttöliittymä edellyttää tekniikka, joka mahdollistaa kommunikoinnin suunnasta riippumatta, kuten RF-tekniikat. ScanMe-menetelmässä on aina oltava vahvistus käyttäjältä, vaikka kohteita olisikin vain yksi, koska käyttäjä ei ole vielä valinnut kiinnostuksen kohdettaan. [12]

PointMe-paradigmalla tarkoitetaan osoittamiseen perustuvaa valintaa. Osoittaminen on luonnollinen tapa valita lähistöllä olevia näkyviä kohteita. Osoitusparadigman käyttö edellyttää, että matkapuhelimessa on jokin osoittamisväline, kuten infrapunälähetin tai kamera. Osoittaminen vaatii näköyhteyden kohteeseen. Tärkeää on myös jollakin tavalla ilmaista käyttäjälle, mitä kohdetta osoitetaan. [12]

TouchMe-menetelmässä tunnistetta kosketetaan virtuaalisesti lukijalla. Virtuaalinen koskettaminen tarkoittaa sitä, että lukija tuodaan tunnisteeseen lähelle, mutta fyysistä kosketusta ei vaadita. Koskettaminen vaatii, että käyttäjä tunnistaa tunnisteeseen sijainnin, mutta suoraa näköyhteyttä ei tarvita. [12]

Near Field Communication eli NFC on teknologia lyhyen kantaman langattomalle tiedonvälitykselle. NFC:n avulla voidaan toteuttaa TouchMe-paradigma. NFC toteutetaan yleensä induktiivisilla RFID-tunnisteilla tai UHF RFID-tunnisteilla, joissa lukijan välimatka on rajoitettu muutamiin senttimetreihin. Koskettaminen on yksiselitteinen tapa valita oikea tunniste ja kohde. Usein tämä on käyttökelpoinen menetelmä, kun kohteita on hyvin paljon. Koskettaminen mahdollistaa nopean vuorovaikutustavan ja siinä ei välttämättä vaadita hyväksyntää käyttäjältä. Joissakin uusimmissa matkapuhelin malleissa on sisäänrakennettu NFC-lukija, kuten Nokian puhelinmallissa 6131 NFC [13].

TouchMe-paradigmaa käytettäessä tunnisteeseen liitetään usein visuaalinen symboli eli ikoni. Ikoni ilmaisee käyttäjälle tunnisteeseen sijainnin sekä tunnisteeseen liittyvän palvelun. Ikonien tulee olla sellaisia, että käyttäjä ymmärtää niiden kuuluvan käyttöliittymään. Käyttäjän tulee myös ymmärtää tunnisteeseen liittyvä palvelu ikonin ulkonäön ja sijainnin perusteella. Ikoniin voidaan liittää myös useita palveluita, jolloin käyttäjä valitsee haluamansa palvelun puhelimesta näytettävältä listalta. [14, 15, 16]

Tunniste on hyvä sijoittaa suoraan kohteeseen, johon palvelu liittyy. Esimerkiksi tulostimeen liitetystä tulostusikonista voidaan päätellä, että tunniste käynnistää tulostuksen juuri tällä tulostimella. Tunniste voidaan sijoittaa myös palvelua kuvaavaan dokumenttiin kuten karttaan tai julisteeseen. Joskus tunniste voidaan sijoittaa myös siten, että sijainti ei liity millään tavalla palveluun, kuten tyhjälle

seinälle. Tällöin sijainnista voidaan päätellä ainoastaan, että palvelu voidaan käynnistää tästä pisteestä. Eräs mahdollisuus on sijoittaa tunniste sulautetusti ympäristöön, ilman tunnistetta ilmaisevaa ikonia. Tällöin on jollakin tavalla kerrottava käyttäjälle, mitä kohteita voidaan koskettaa ja mitä palveluja kohteisiin liittyy. [14, 15, 16]

NFC-teknologian ennakoitaan yleistyvän matkapuhelimissa voimakkaasti. Vuonna 2006 arvioitiin vuoteen 2010 mennessä 50 prosentissa puhelimia olevan NFC-ominaisuus [17]. Todellisuudessa yleistyminen näyttää tapahtuvan huomattavasti hitaammin. Tulevaisuuden visiona on, että NFC-tunnisteita alkaisi olla kaikkialla kaupunkiympäristöissä. Niiden avulla matkapuhelimeen voisi saada monenlaista tietoa kohteesta jota tunniste kuvaa. Matkapuhelimeen on kehitetty monenlaisia NFC:tä hyödyntäviä toimintoja, kuten erilaisten palveluiden käynnistäminen, tietojen vaihto tai vaikkapa elektroninen maksaminen. [18]

### 2.3.3. Eleohjaus

Myös eleohjaukseen perustuva käyttöliittymä on fyysinen. Matkapuhelimen eleohjaus perustuu joko matkapuhelimen liikuttamiseen tai matkapuhelimen havaitsemaan ulkoiseen liikkeeseen. Liikkeen havaitseminen voi perustua näköön, kosketukseen tai kiihtyvyydataan. Nykyään markkinoilla on joitakin puhelimia, joissa on jollakin tavoin hyödynnetty eleohjausta. Tällaiset puhelimet löytyvät useimmilta matkapuhelinvalmistajilta, ainakin Sony-Ericssonilta [19], Applelta [20], Nokialta [21] ja Samsungilta [22].

Konenäköön perustuva eleohjaus soveltuu havaitsemaan sekä staattisia että dynaamisia eleitä. Staattiset eleet tarkoittavat asentoja ja dynaamiset liikkeitä. Tällainen eleohjaus perustuu puhelimen kameran käyttöön: kameran kuvasta tunnistetaan esimerkiksi käyttäjän käsillä tekemiä liikkeitä tai vaikkapa käden asento. Tämä ratkaisu soveltuu huonosti matkapuhelimeen, koska luonteva käyttötapa matkapuhelimelle on pitää sitä kädessä. [2] Kameran havaitseman liikkeen käyttö vaatii puhelimen sijoittamista paikkaan, josta esim. käden liike voidaan havaita. Näköön perustuva eleohjaus on käytössä esimerkiksi Sony Ericssonin puhelinmallissa W380i [19].

Kosketukseen perustuvassa eleohjauksessa havaitaan 2-ulotteiselle tasolle piirrettävä liikerata. Tällaista tekniikkaa käytetään yleisimmin kosketusnäytöissä. Kosketuksen havaitsemista voidaan käyttää myös esimerkiksi tunnistamaan, milloin puhelin on taskussa tai korvalla.

Kiihtyvyyssantureihin perustuva eleohjaus voidaan toteuttaa havaitsemaan staattista asentoa tai dynaamista liikettä. Staattisen asennon havaitseminen on yksinkertaisin tapa ja se voidaan suoraan laskea painovoimakiihtyvyydestä. Tällainen eleohjaus soveltuu hyvin vaikkapa valikoiden selaamiseen. Ravistelun tunnistaminen on myös kohtalaisen suoraviivaista. Kun ravistelun voimakkuus ylittää määritetyn kynnsarvon, se tulkitaan komennoksi. Huomattavasti monimutkaisempaa on eleentunnistus, jossa käytetään puhelimen liikeratoja komentoina. Eleiden luokitteluun käytetään yleensä Markovin piilomalleja (Hidden Markov Model), tukivektorikoneita (Support Vector Machine) tai Bayes-verkkoja, (Bayesian Network). [2]

Kiihtyvyydataan perustuva eleohjaus hyödyntää laitteen sisäisiä kiihtyvyyssantureita. Tarkan tiedon laitteen asennosta saisi käyttämällä gyroskooppia. Ne ovat kuitenkin liian kalliita ja suuria matkapuhelimiin, joten niiden sijaan käytetään kiihtyvyyssanturin antamaa arviota laitteen asennosta. [2]

Menetelmä, jossa eleitä tunnistetaan laitteen liikkeistä jatkuvasti on erityisen haasteellinen. Tilanteita joissa tunnistetaan vääriä eleitä (false-positive) syntyy huomattavasti vähemmän, jos eleentunnistuksen alku ja loppu ilmaistaan esimerkiksi nappia painamalla.

Suurimpia eleentunnistukseen liittyviä haasteita on eleiden erilaisuus eri käyttäjien välillä. Toimivan eleohjauksen toteuttamiseksi eleiden tulisikin olla käyttäjän opetettavissa. Täsmällisen tuloksen saavuttamiseksi eleitä on toistettava riittävän usein opetusvaiheessa. Eleiden opettamisen tulee olla myös mahdollisimman nopeaa ja helppoa. Eleohjauksen tunnistustarkkuuden olisi oltava lähes 100 prosentista. Liian monet virheet eleiden tunnistuksessa saavat käyttäjän hylkäämään menetelmän. [23]

Myös yksittäisen käyttäjän antamissa eleissä nopeus ja mittakaava vaihtelevat eri toistokerroilla. Eleiden määrän tulisi myös olla riittävän pieni, jotta saavutettaisiin hyvä eleiden tunnistustarkkuus. Yksinkertaisten eleiden opetukseen riittää jo muutama toisto, kun taas monimutkaisempia eleitä opettaessa toistoja tarvitaan enemmän. Jo 10 toistokerralla on saavutettu hyvä tunnistettavuus myös monimutkaisemmilla eleillä. [24]

Ensimmäinen kaupallinen matkapuhelinmalli, jossa hyödynnetään kiihtyvyyssantureihin perustuvaa eleohjausta on vuonna 2005 julkistettu Samsungin SCH-S310. Tämä puhelin tunnistaa numeroeleet 1-9 sekä viisi ilmaan piirrettävää symbolielettä. Numeroeleillä on mahdollista soittaa puhelimen pikavalintanumeroihin. Nuolieleillä voidaan vaihtaa MP3-soittimen kappaleita. Viestin poistaminen tapahtuu ravistelemalla puhelinta pystysuunnassa kahdesti. Piirtämällä ilmaan kirjaimen O tai X, puhelimen saa toistamaan ennalta nauhoitetut äänet. [2]

### 3. ELEHALLINTAJÄRJESTELMÄ

Tässä kappaleessa kuvataan elehallintajärjestelmän suunnittelu ja toteutus. Koska tutkimusryhmässä ei ollut kokemusta vastaavanlaisten järjestelmien toteuttamisesta, päätettiin edetä useammassa vaiheessa. Koko järjestelmän yksityiskohtaisempi suunnittelu tehtiin sen jälkeen, kun oli ensin todettu tiettyjen ohjelmiston osien toteuttaminen mahdolliseksi. Alun perin tavoitteena oli elehallintasovelluksen kehittäminen, mutta myöhemmin päätettiin laajentaa järjestelmä yleiskäyttöiseksi eleohjaimeksi, jota voisi hyödyntää mikä tahansa sovellus.

Järjestelmä kehitettiin hyödyntämään Mikko Kauppilan toteuttamaa eleentunnistus- ja nauhoitusjärjestelmää [4]. Eleentunnistus- ja nauhoitusjärjestelmän kehitystyö oli osittain valmis tämän projektin alkaessa ja eteni osittain rinnakkain tämän projektin kanssa.

#### 3.1. Toteutettavuusanalyysi

Suunnittelun pohjana toimi toteutettavuusanalyysi. Toteutettavuusanalyysissä selvitettiin mahdollisuuksia toteuttaa eri osia ohjelmasta eri ohjelmointikielillä sekä näiden osien välisen kommunikoinnin toteuttamismahdollisuuksia. Tämän vaiheen tavoitteena oli myös tutkia, miten ja mitä puhelimen toiminnallisuutta voidaan ohjata ohjelmallisesti. Lisäksi toteutettavuusanalyysissä selvitettiin, miten toteuttaa järjestelmä, joka käynnistyy automaattisesti ja on aina taustalla käynnissä.

##### 3.1.1. Eri kielten soveltuvuus toteutukseen

Eleentunnistus- ja nauhoitusjärjestelmä on toteutettu Javalla. Niinpä eleentunnistuksen käyttöön tarvitaan Java-kielellä toteutettava Midlet-sovellus. MIDP Javan turvallisuusominaisuuksiin kuuluu ”hiekkalaatikko”, jossa sovellusta ajetaan. Tämä turvallisuusmalli rajoittaa sovelluksen pääsyä ulkopuolisiin resursseihin, joten puhelimen ominaisuuksien hallintaan tarvitaan Symbian C++-sovellus. Osa toiminnallisuudesta on mahdollista toteuttaa myös Pythonilla, jos tämä helpottaa toteutustyötä.

Toteutettavuusanalyysissä tutkittiin mahdollisuutta käynnistää ohjelma automaattisesti puhelimen käynnistyessä. Osoittautui, että automaattinen käynnistyminen ei ole mahdollista Java-sovellukselle. Niinpä tätä varten tarvitaan Symbian C++-sovellus, joka käynnistyy puhelimen käynnistyksen yhteydessä. Työssä toteutettiin sovellus, joka Symbian Startup List Management API:a käyttäen käynnistyi puhelimen käynnistyksen yhteydessä [25]. Java-sovelluksen käynnistäminen onnistuu push-mekanismiin avulla [26]. Java-midlet toteutettiin käynnistymään puhelimen käynnistyksen yhteydessä siten, että toteutettu automaattisesti käynnistytävä C++-sovellus luo TCP-yhteyden tai lähettää UDP-

paketin porttiin, jota midlet on rekisteröitynyt push-mekanismin avulla kuuntelemaan.

### 3.1.2. Ohjelmien välinen kommunikointi

Mobiili Java ei tue Java-natiivirajapintaa, eli sillä ei ole mahdollisuutta päästä suoraan käsiksi käyttöjärjestelmän kutsuihin. Java- ja Symbian-ohjelmien välinen kommunikointi on mahdollista toteuttaa pistokkeiden (engl. socket) avulla. TCP-pistokkeiden käytöstä löytyi esimerkki Symbian Developer Network-sivustolta [27]. Python-skriptien kutsuminen taas on mahdollista suoraan Symbian-sovelluksesta. Tästä löytyi esimerkki DZone Snippets-sivustolta, johon on kerätty S60 Python-koodiesimerkkejä [28]. Python-skriptien käyttäminen edellyttää Python S60-tulkin asentamista laitteeseen.

Toteutettavuusanalyysin aikana tutkittiin TCP- ja UDP-pistokkeiden käyttöä J2ME-midletin ja Symbian C++-sovelluksen välillä. Näissä vertailuissa TCP osoittautuikin yllättäen nopeammaksi. Molempia protokollia käyttäen lähetettiin 100 kpl lyhyitä merkkijonoja. UDP:llä ensimmäisen ja viimeisen viestin vastaanoton välinen aika oli 2-3 sekuntia, TCP:llä alle yksi sekunti. Kehitettävää sovellusta ajatellen näillä viiveillä ei ole käytännön merkitystä, sillä näiden testien perusteella yhden viestin lähetyksajaksi saadaan keskimäärin 30 ms. TCP:tä käytettäessä lähetyksen tapahtui niin nopeasti, että vastaanottopäässä lukukomento palautti useamman lähetetyn viestin kerralla. UDP:ta käytettäessä vastaanottopää ehti lukea kunkin viestin erikseen. Viestin lähetyksen toteutettiin Javalla UDP-paketteja käyttäen seuraavasti:

```
dg = udpConnection.newDatagram( msg.getBytes(), msg.length(),
                                url);
udpConnection.send(dg);
```

TCP-protokollaa käyttäessä luodaan ensin TCP-yhteys ja avataan lähtevä vuo (engl. stream):

```
sc = (StreamConnection)Connector.open(url);
os = sc.openDataOutputStream();
```

Viestin lähetyksen tapahtuu kirjoittamalla vuohon:

```
os.write(msg.getBytes(), 0, msg.length());
os.flush();
```

TCP on yhteydellinen protokolla ja siinä luodaan yhteys kahden pistokkeen välille, kun taas UDP-protokollaa käytettäessä kukin UDP-paketti sisältää tiedon kohdeosoitteesta. Nopeusero TCP-yhteyden hyväksi johtuu todennäköisesti siitä, että TCP-yhteydessä lähetetään viestit valmiiksi auki olevan yhteyden yli, kun UDP-paketteja käytettäessä kukin viesti joudutaan reitittämään erikseen. UDP:n etuna on, että viestit on helppo lukea yksi kerrallaan. Lisäksi UDP-yhteyttä käytettäessä paketteja voidaan lähettää monelta eri asiakkaalta samaan porttiin. Tiedonsiirron luotettavuus on riittävää puhelimen sisällä myös UDP-pistokkeita käyttäen. Ohjelmien välisen kommunikoinnin aiheuttamaa viivettä tullaan tutkimaan

tarkemmin, mikäli sovelluksen toiminta osoittautuu käytettävyydestänsä liian hitaaksi. Vertailua kokeiltiin myös Pythonin ja C++-sovelluksen välillä, mutta siinä UDP oli yhtä nopea kuin TCP. Javalla TCP-yhteyden toteutus eroaa siis jollain tavalla Pythonin vastaavasta toteutuksesta.

### ***3.1.3. Puhelimen hallinta***

Kehitettävällä sovelluksella tulee kyetä hallitsemaan tiettyjä matkapuhelimen toimintoja. Lopulliseen ohjelmistoon valitaan hyödyllisiksi arvioituja ja käytettävyydestänsä aikana hyödyllisiksi osoittautuvia ominaisuuksia. Toteutettavuusanalyysin aikana selvitettiin puhelimesta olemassa olevien toimintojen ja ohjelmien käyttöä Symbian C++-sovelluksesta.

Puhelimen natiivisovellusten käynnistäminen osoittautui varsin suoraviivaiseksi Symbian-sovelluksella. Tähän löytyi hyvä ohjeistus Forum Nokia-sivustolta [25]. Symbian-sovellus, jolla on graafinen käyttöliittymä, voi käynnistää tiettyjä muiden sovellusten näkymiä oman AppUi-luokkansa ActivateViewL-metodin avulla. Useimmat sovellukset on kuitenkin mahdollista käynnistää ainoastaan yhteen näkymään. Jotkut sovellukset voidaan käynnistää muutama eri näkymään. Esimerkiksi Viestit-sovellusta ei voida käynnistää ohjelmallisesti viestin kirjoitusnäkymään. Tällaisiin asioihin löytyy yleensä jokin muu Symbian C++-API. Erilaisten toimintojen toteutukseen löytyy runsaasti opastusta ja esimerkkejä eri keskustelufoorumeilta. Toteutettavuusanalyysin aikana tehtiin testisovellus, johon toteutettiin seuraavat toiminnot: viestien lukeminen ja kirjoittaminen, pikavalintoihin soittaminen, puhelimen gallerian kuvien selailu, selaimen avaaminen tiettyyn www-osoitteeseen, kalenterin avaaminen nykyisen päivän kohdalta, profiilin vaihtaminen, kellosovelluksen avaaminen sekä Python-skriptin suorittaminen.

### ***3.1.4. Kohdelaitteiden ominaisuudet***

Ohjelmistoa kehitetään Nokian puhelinmalleille 5500 Sport sekä N95. 5500 Sport puhelinmallissa on yksi 235 MHz:n ARM9-prosessori ja 32 MB RAM-muistia. Käyttöjärjestelmä on Symbian OS v9.1 ja kehitysalusta on S60 3rd Edition (initial release). Puhelin tukee Java MIDP 2.0 -profiilia ja CLDC 1.1-konfiguraatiota.

Puhelinmallissa N95 on 332MHz:n ARM11 Dual-prosessori, 3D-grafiikkakiihdytin sekä 256 MB RAM-muistia. Käyttöjärjestelmä on Symbian OS v9.2 ja kehitysalusta on S60 3rd Edition, Feature Pack 1. Java MIDP-profiili ja CLDC-konfiguraatio on sama kuin 5500 Sportissa.

CLDC-konfiguraatio määrittelee API:t, jotka kaikki saman konfiguraation laitteet tarjoavat. Saman CLDC-konfiguraation laitteissa on myös käytössä sama virtuaalikone. [29]

Merkittevä ero työn kannalta on eri S60-kehitysalustaversiot. Java-sovellusten pitäisi olla alustariippumattomia ja molemmissa puhelinmalleissa on sama MIDP ja

CLDC. Eli Javan osalta toiminnassa ei pitäisi laitteiden välillä olla mitään eroja. Kehitettävää sovellusta testataan molemmissa puhelimissa. Tarvittaessa C++-sovelluksen tietyt asiat tehdään erikseen molemmille puhelimille. Eri versiot erotetaan toisistaan käännöslippujen avulla. Kehitystyö tehdään pääosin 5500 Sport-puhelimella, koska tämä oli ainoa käytössä oleva puhelin projektin alkuvaiheissa. Se on puhelinmalleista suorituskyvyltään selvästi heikompi, joten suorituskykyongelmat tulevat sillä esille.

### 3.2. Vaatimusmäärittely

Vaatimusmäärittely tehtiin toteutettavuusanalyysin aikana saatujen kokemusten pohjalta. Työ eteni vaiheittain myös toteutettavuusanalyysin jälkeen. Aluksi toteutettiin nopeasti järjestelmä, jota voitiin testata. Saatujen kokemusten perusteella vaatimusmäärittelyä tarkennettiin ja seuraavassa vaiheessa toteutettiin lisää. Alkuperäinen tavoite oli toteuttaa puhelimen elehallintasovellus. Elehallintasovelluksen toiminnallisuutta kehitettiin aikaisempien vaiheiden ja käytettävyydestien perusteella. Lopulliseen versioon päätettiin lisäksi sisällyttää yleiskäyttöinen elehallintajärjestelmä sekä kaksi sitä käyttävää sovellusta: puhelimen elehallintasovellus sekä Internet-sovellusta käyttävä elehallintasovellus.

Tässä projektissa ei siis kehitetä eleentunnistusta, vaan käytetään aiemmin toteutettua eleentunnistusjärjestelmää. Aiemmin toteutettu järjestelmä vastaa myös eleiden nauhoituksesta. Aiemman järjestelmän toimintaa laajennetaan siten, että tarjotaan yleiskäyttöinen eleentunnistus, jota useat asiakassovellukset voivat käyttää. Järjestelmää käyttävä paikallinen sovellus voidaan toteuttaa millä tahansa ohjelmointikielellä, joka voi vastaanottaa tietoja TCP- tai UDP-protokollaa käyttäen. Ulkopuolinen sovellus voi olla mikä tahansa HTTP-pyynnöillä ohjattava sovellus. Yleiskäyttöiselle elehallintajärjestelmälle asetetut vaatimukset on lueteltu taulukossa 1.

Taulukko 1. Elehallintajärjestelmän vaatimukset

Vaatus	Kuvaus
Toteutetun eleentunnistusjärjestelmän käyttäminen	Järjestelmä tulee sisältämään Mikko Kauppilan toteuttaman eleentunnistusjärjestelmän, jonka lisäksi järjestelmään toteutetaan eleohjain. Eleentunnistusjärjestelmä tunnistaa eleitä puhelimen kiihtyvyysanturien tuottaman jatkuvan datan perusteella ja eleohjain vastaa tunnistettujen eleiden käsittelystä.
Yleiskäyttöinen elehallintajärjestelmä	Kehitettävän järjestelmän tulee olla yleiskäyttöinen siten, että sitä voidaan käyttää useiden sovellusten hallintaan. Elehallintaa tulee kyetä käyttämään puhelimen sisäisten sovellusten ohjaamiseen TCP-yhteydellä tai UDP-viesteillä sekä puhelimen ulkopuolisten sovellusten ohjaamiseen HTTP-pyynnöillä.
Aktiivisten eleiden määrän rajoittaminen	Järjestelmää käyttävillä sovelluksilla on oltava mahdollisuus asettaa haluamansa eleet aktiivisiksi siten, että ainoastaan aktiivisia eleitä tunnistetaan. Sovellukset voivat ilmoittaa aktivoitavat eleet TCP- tai UDP-protokollaa käyttäen. Tällaisella ratkaisulla voidaan aktiivisten eleiden määrä pitää kohtalaisen pienenä.
Eleiden yleiskäyttöisyys	Eleiden on oltava yleiskäyttöisiä siten, että mikä tahansa sovellus voi käyttää mitä tahansa ennalta nauhoitetuista eleistä. Yksittäinen sovelluksen tulee voida käyttää myös samoja eleitä usealle eri toiminnolle aktivoimalla eri tilanteissa tarvitsemansa eleet.

Myös eleiden on oltava yleiskäyttöisiä. Tällöin eleiden kokonaismäärä voidaan pitää mahdollisimman pienenä. Käyttäjän on vaikea muistaa kovin suurta määrää eleitä [30].

Sovellusten tulee kyetä määrittämään aktiivisten eleiden lukumäärä. Pienellä aktiivisten eleiden lukumäärällä saadaan väärin tunnistettujen eleiden mahdollisuus mahdollisimman pieneksi [2].

Puhelimen elehallintasovelluksen tulee kyetä hallitsemaan puhelimen sisäistä toimintaa eleitä käyttämällä. Sovelluksen avulla voidaan ohjata taulukossa 2 esitettyjä toimintoja. Puhelimen elehallintasovelluksesta toteutetaan kaksi erilaista versiota. Toisessa elehallinta aktivoidaan tietyn näppäimen painalluksella, toisessa taas aktivointi tapahtuu aktivoimalla puhelin. Puhelimen aktivointi tarkoittaa mitä tahansa toimenpidettä tai tapahtumaa, joka saa puhelimen taustavalon syttymään. Tällaisia ovat esimerkiksi näppäimen painallus, viestin saapuminen tai taustavalon syyttäminen eleellä.

Taulukko 2. Elehallintasovelluksen toiminnot

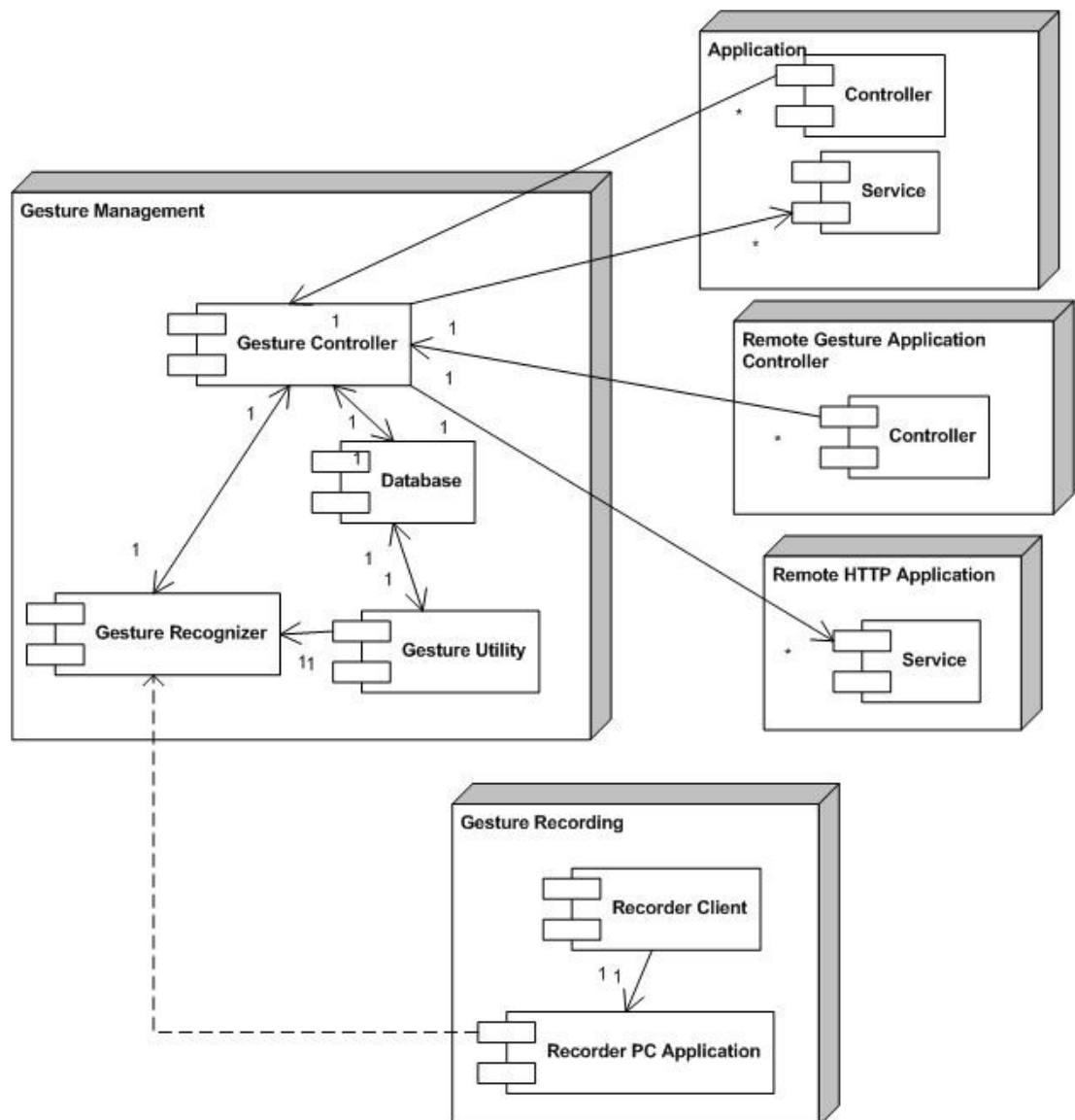
Toiminto	Kuvaus
Taustavalon sytyttäminen	Sytyttää taustavalon ja avaa näppäinlukon. Näppäinlukko palautetaan, jos puhelin on käyttämättä viisi sekuntia.
Pikavalintaan soittaminen	Soittaa puhelimen pikavalinnoissa määritettyihin numeroihin. Numeroa yksi vastaava ele soittaa vastaajaan, numeroita kahdesta yhdeksään vastaavat eleet soittavat niihin liitettyihin pikavalintanumeroihin.
Hälytyksen asettaminen	Näyttää käyttäjälle hälytyksen asetusdialogin. Tästä dialogista käyttäjä voi asettaa haluamansa kellonajan hälytysajaksi tai peruuttaa dialogin.
Käyttöprofiilin vaihtaminen	Käyttäjä voi vaihtaa puhelimen käyttöprofiilia profiilien kokous, yleinen ja äänetön välillä.
Tekstiviestin lukeminen	Avaa saapuneen lukemattoman tekstiviestin. Jos lukemattomia viestejä ei ole, niin toiminnolla avataan tekstiviestisovelluksen saapuneet-kansio.
Viestin lähetys	Avaa viestin kirjoitusikkunan. Käyttäjä voi valita vastaanottajan ja kirjoittaa viestin sekä lähettää viestin tai sulkea ikkunan.
Kalenteri	Avaa kalenteri sovelluksen. Sovellus avataan nykyisen päivän näkymään.

Internet-sovellusten hallintaa varten tarvitaan puhelimeen asennettava ohjainsovellus. Ohjainsovellus vastaa eleohjaimen konfiguroinnista siten, että elehallinnan avulla voidaan hallita HTTP-pyynnöillä internet-sovellusta.

### 3.3. Suunnittelu ja toteutus

#### 3.3.1. Arkkitehtuuri

Järjestelmän pääkomponentit on esitelty kuvassa 2. Pääkomponentit ovat eleiden nauhoitusjärjestelmä (Gesture Recording), eleiden hallintajärjestelmä (Gesture Management) sekä elehallintaa käyttävät sovellukset (Application, Remote HTTP Application, Remote Gesture Application Controller).



Kuva 2. Järjestelmän pääkomponentit.

Eleiden nauhoittamista varten on toteutettu sovellukset sekä PC-tietokoneeseen että puhelimeen. Puhelimessa oleva sovellus (Recorder Client) lähettää nauhoitetun kiihtyvyyssanturidatan bluetoothin yli PC:lle. PC:llä ajettava sovellus (Recorder PC

Application) tallentaa vastaanottamansa datan tiedostoon ja laskee datan perusteella parametrit Markovin piilomalleille. Lasketut parametrit tallennetaan tiedostoon, joka sisällytetään eleen tunnistimeen (Gesture Recognizer) käännoisaikana. Nauhoitusvaiheessa nauhoitetuille eleille annetaan myös yksikäsitteiset nimet, jotka toimivat eleiden tunnisteinä.

Eleohjain (Gesture Controller) sekä eleapuohjelma (Gesture Utility) toteutetaan samaan J2ME MIDlet-projektiin eleen tunnistimen kanssa. Eleohjain vastaa eleentunnistimen luomisesta. Samalla eleohjain asettaa itsensä eleentunnistimen kuuntelijaksi. Toteuttamansa GestureListener-rajapinnan kautta se saa tiedon kaikista tunnistetuista eleistä. Eleohjain toteuttaa myös UDP- ja TCP-palvelimen. Näin eleohjainta käyttävät sovellukset voivat lähettää sille pyyntöjä UDP-paketteina tai TCP-yhteyden kautta.

Taulukossa 3 on esitetty pyynnöt, joita sovellukset voivat tehdä eleohjaimelle. Sovellukset voivat luoda, aktivoida ja vapauttaa sääntöryhmiä. Sääntöryhmä (kts. seuraava kappale) koostuu säännöistä, joista jokainen määrittelee yhden eleen ja sitä vastaavan komennon.

Taulukko 3. Eleohjaimelle esitettävät pyynnöt

<b>Merkkijono</b>	<b>Merkitys</b>
set <group-name><gesture-name-1><URL-1> ... <gesture-name-N><URL-N>	Luo sääntöryhmän annetulla ryhmän nimellä ja lisää säännöt ryhmään.
activate <priority><group-name>	Aktivoi sääntöryhmän annetulla nimellä ja annetulla prioriteetillä.
deactivate <group-name>	Vapauttaa annetulla nimellä olevan sääntöryhmän.

Sääntöryhmät talletetaan tietokantaan (Database), joka toteutetaan MIDP RMS-järjestelmällä [31]. Tähän tietokantaan on mahdollista päästä käsiksi ainoastaan saman MIDP-projektin sovelluksista. Eleapuohjelman (Gesture Utility) avulla käyttäjä voi muokata ja poistaa tietokannassa olevia sääntöryhmiä tai luoda uusia.

Elehallintaa käyttäviä sovelluksia (Application) voidaan toteuttaa millä tahansa ohjelmointikielellä, joka tukee UDP- tai TCP-yhteyksiä. Elehallintaa varten on oltava ohjainosa (Controller), joka huolehtii eleohjaimen käytöstä sekä palveluosa (Service), joka kuuntelee eleitä vastaavia URL-osoitteita. Puhelimen sisäisiä sovelluksia käytettäessä ohjain ja palveluosa voivat hyvin olla samassa sovelluksessa, mutta ulkopuolisia palveluja käytettäessä tarvitaan erillinen ohjainsovellus puhelimeen (Remote Gesture Application Controller). Erillinen ohjainsovellus vaaditaan, koska puhelimeen ei voida luoda TCP- tai UDP-yhteyttä puhelimen ulkopuolelta, vaan pyynnöt eleohjaimelle on lähetettävä puhelimesta suoritettavalta sovellukselta.

### 3.3.2. Sääntöryhmät

Sääntöryhmillä saavutetaan eleiden yleiskäyttöisyys. (Ele, URL)-parit muodostavat sääntöjä, jotka ryhmitellään sääntöryhmiksi. Jokainen sääntö sisältää yhden eleen ja siihen liittyvän URL:n, joka määrittää eleeseen liittyvän toiminnon. Aktiivinen

sääntöryhmä määrittää eleet, joita eleentunnistusjärjestelmä tunnistaa sekä toiminnot, jotka eleohjain suorittaa, kun eleentunnistin tunnistaa eleitä.

Yksittäinen ele voi esiintyä usean ryhmän säännöissä, mutta kussakin ryhmässä vain kerran. Sovellukset käyttävät palvelua luomalla ja aktivoimalla sääntöryhmiä. Sovellukset voivat luoda sääntöjä olemassa olevaan ryhmään tai luoda uusia ryhmiä. Sovellus voi aktivoida kerrallaan yhden olemassa olevista ryhmistä. Kun sääntöryhmä on aktivoitu, niin ainoastaan kyseisen ryhmän säännöissä esiintyviä eleitä tunnistetaan.

Sääntöryhmä luodaan antamalla ryhmälle nimi sekä yksi tai useampia (ele, URL)-pareja. Jos annetulla nimellä on jo olemassa ryhmä, niin säännöt lisätään tähän ryhmään. Jos jonkin säännön URL tai ele esiintyy jossakin kyseisen ryhmän säännössä, niin tällainen sääntö jätetään luomatta.

Sääntöryhmä aktivoidaan kertomalla aktivoitavan ryhmän nimi sekä antamalla aktivoinnille prioriteetti. Prioriteetti voi olla joko yksi tai nolla, joista numero yksi tarkoittaa korkeampaa prioriteettia. Uusi sääntöryhmä aktivoituu vain, jos aktiivisena jo olevalla sääntöryhmällä on sama tai pienempi prioriteetti. Taustalla jatkuvasti aktiivisena oleva puhelimen elehallintasovellus käyttää prioriteettia nolla ja muut kuin taustasovellukset prioriteettia yksi.

Kun sääntöryhmä aktivoidaan, niin eleohjain kertoo ryhmässä olevat eleet tunnustusjärjestelmälle, joka tällöin tunnistaa ainoastaan kyseisiä eleitä. Sääntöryhmän aktivoineen sovelluksen on myös vapautettava aktivoimansa ryhmä, kun se ei enää tarvitse sitä. Kun sääntöryhmä vapautetaan, niin eleohjain aktivoi edellisen aktiivisen ryhmän tai ryhmän, joka on viimeksi yritetty aktivoida pienemmällä prioriteetilla.

Taulukossa 4 on esimerkkejä käytettävistä URL:eista sekä eleohjaimen toiminto tilanteessa, jossa säännön ele tunnistetaan. Jos säännön URL on UDP- tai TCP-osoite, niin kyseiseen osoitteeseen lähetetään URL:n toiminto-osa. Jos URL on HTTP-osoite, niin suoritetaan URL:n mukainen HTTP GET-pyyntö.

Taulukko 4. Esimerkkejä eleisiin liitettävistä URL:eista

URL	Toiminto
datagram://127.0.0.1:1235?calendar	Lähetää UDP-paketin, joka sisältää merkkijonon "calendar" oman koneen porttiin 1235.
socket://127.0.0.1:1235?clock	Avaa TCP-yhteyden oman koneen porttiin 1235 ja lähettää merkkijonon "clock".
http://gw16.rotuaari.net:8084/reaches/request?service=show_products&event=start&control=MIDlet	Tekee URL:n mukaisen HTTP-pyyntö.

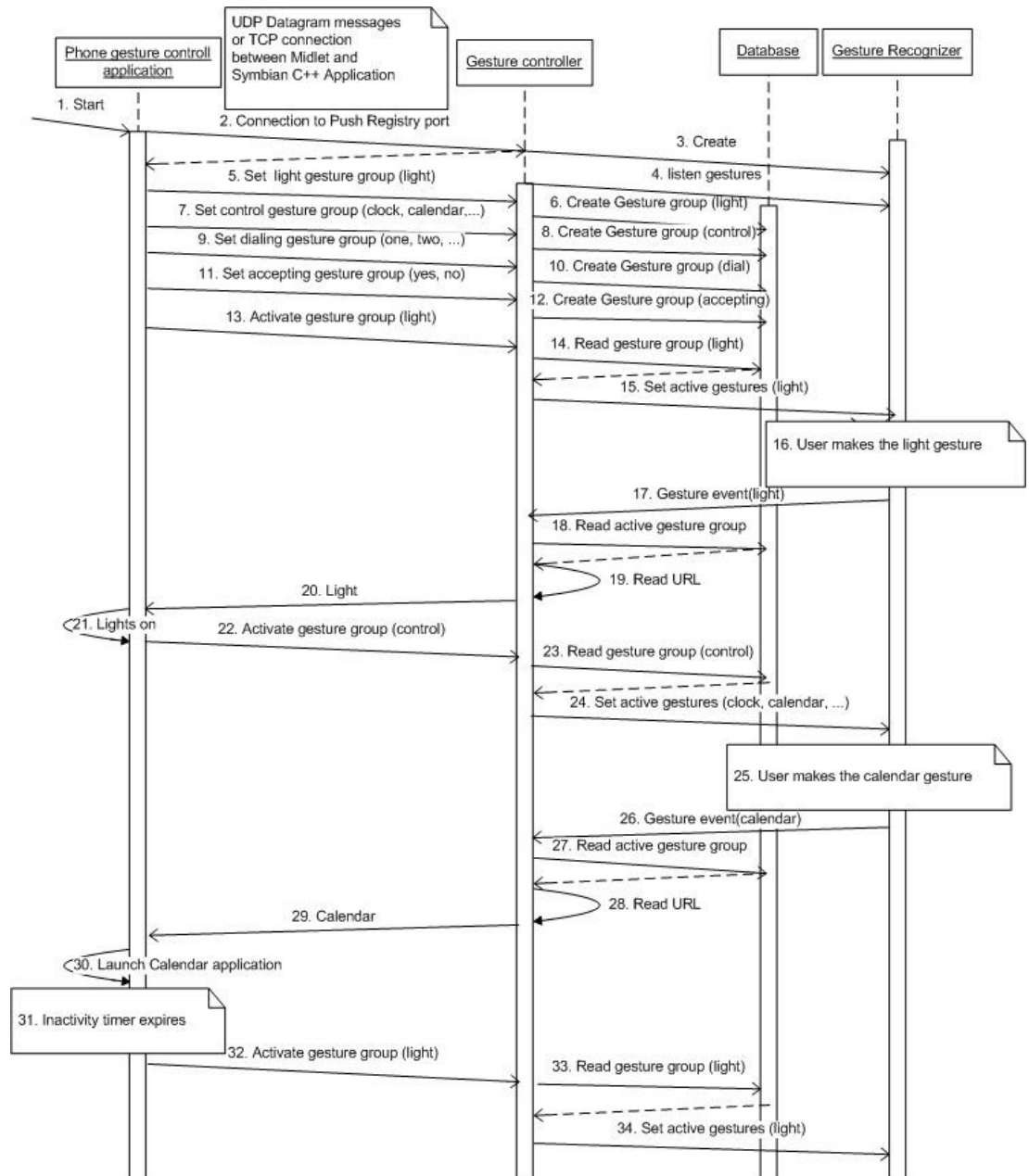
Taulukossa 5 on listattu kaikki tässä projektissa käytetyt sääntöryhmät. Elehallintasovellus luo ja käyttää ryhmiä Light, Control, SpeedDial ja Accepting. Internet-sovelluksen ohjainsovellus luo sääntöryhmän HTTP\_requests, jolla ohjataan REACHeS-järjestelmän valokuva-albumisovellusta[16].

Taulukko 5. Projektissa käytetyt sääntöryhmät

Ryhmä	Komentoele	URL
Light	light	socket://127.0.0.1:1237?light
Control	calendar	socket://127.0.0.1:1237?calendar
Control	clock	socket://127.0.0.1:1237?clock
Control	profile	socket://127.0.0.1:1237?profile
Control	msg	socket://127.0.0.1:1237?msg
Control	msg_write	socket://127.0.0.1:1237?msg_write
Control	speed_dial	socket://127.0.0.1:1237?speed_dial
SpeedDial	one	socket://127.0.0.1:1237?one
SpeedDial	two	socket://127.0.0.1:1237?two
SpeedDial	three	socket://127.0.0.1:1237?three
SpeedDial	four	socket://127.0.0.1:1237?four
SpeedDial	five	socket://127.0.0.1:1237?five
SpeedDial	six	socket://127.0.0.1:1237?six
SpeedDial	seven	socket://127.0.0.1:1237?seven
SpeedDial	eight	socket://127.0.0.1:1237?eight
SpeedDial	nine	socket://127.0.0.1:1237?nine
Accepting	yes	socket://127.0.0.1:1237?yes
Accepting	no	socket://127.0.0.1:1237?no
HTTP_requests	calendar	http://gw16.rotuaari.net:8084/reaches/request?service=show_products&event=start&control=MIDlet
HTTP_requests	clock	http://gw16.rotuaari.net:8084/reaches/request?service=show_products&event=first&control=MIDlet
HTTP_requests	profile	http://gw16.rotuaari.net:8084/reaches/request?service=show_products&event=previous&control=MIDlet
HTTP_requests	msg	http://gw16.rotuaari.net:8084/reaches/request?service=show_products&event=next&control=MIDlet
HTTP_requests	msg_write	http://gw16.rotuaari.net:8084/reaches/request?service=show_products&event=last&control=MIDlet
HTTP_requests	speed_dial	http://gw16.rotuaari.net:8084/reaches/request?service=show_products&event=stop&control=MIDlet

### 3.3.3. Järjestelmän käyttäytyminen

Kuvassa 3 kuvataan järjestelmän käynnistyminen sekä toimintaa eleellä aktivoitavassa sovelluksessa käyttäen TCP-yhteyttä. Eleohjaimen (Gesture Controller) käynnistämistä puhelimen käynnistyksen yhteydessä huolehtii puhelimen elehallintasovellus (Phone gesture control application). Puhelimen elehallintasovellus on esimerkki kuvan 2 elehallintaa käyttävästä sovelluksesta. Sovellus toteuttaa sekä ohjainosan että palveluosan.



Kuva 3. Järjestelmätason sekvenssikaavio.

Elehallintasovellus käynnistyy automaattisesti puhelimen käynnistyksen yhteydessä (viesti 1). Käynnistyttyään sovellus avaa TCP-yhteyden porttiin, jota eleentunnistusjärjestelmän eleohjain (Gesture controller) on rekisteröinyt kuuntelemaan käyttäen Midlet Push-rekisteriä (viesti 2). Eleohjain siis käynnistyy automaattisesti push-rekisterin avulla, kun jokin asiakassovellus luo TCP-yhteyden eleohjaimen käyttämään porttiin.

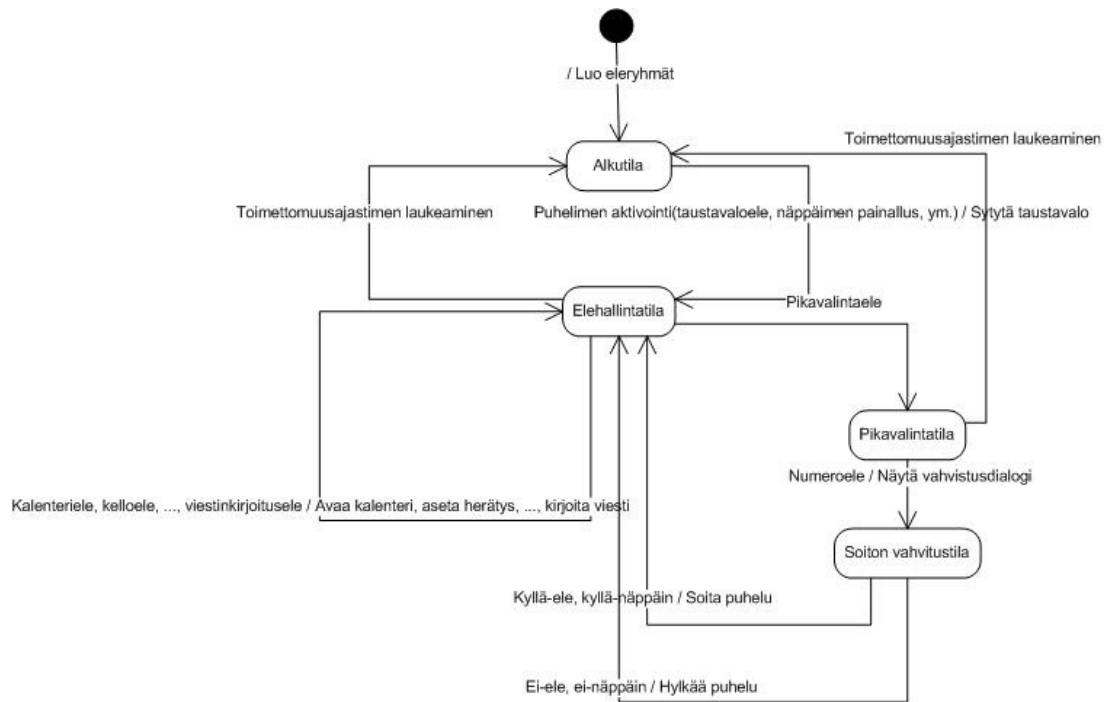
Elehallintasovellus luo käynnistyessään sääntöryhmät taustavalolle, hallintaeleille, numeroeleille ja hyväksymiseleille. Elehallintasovellus lähettää pyynnöt eleohjaimelle, joka tallettaa sääntöryhmät MIDP RMS-tietokantaan (Database) (viestit 5-12). Kussakin ryhmässä olevat URL:t liitetään oman koneen porttiin, jota sovellus kuuntelee. Elehallintasovellus aktivoi aluksi taustavaloryhmän (viesti 13). Eleohjain lukee aktivoitavan ryhmän säännöt tietokannasta ja kertoo aktivoitavat eleet eleentunnistimelle (Gesture Recognizer) (viestit 14-15).

Taustavaloryhmä sisältää yhden säännön. Kun tämän säännön ele tunnistetaan, niin eleentunnistin kertoo tunnistetun eleen eleohjaimelle (viestit 16-17). Eleohjain lukee eleryhmän tietokannasta ja hakee tunnistettua elettä vastaavan URL:n (viestit 18-19). Tässä URL on TCP-osoite, joten eleohjain lähettää URL:n toiminto-osan ”light” TCP-yhteyden yli kyseiseen osoitteeseen (viesti 20).

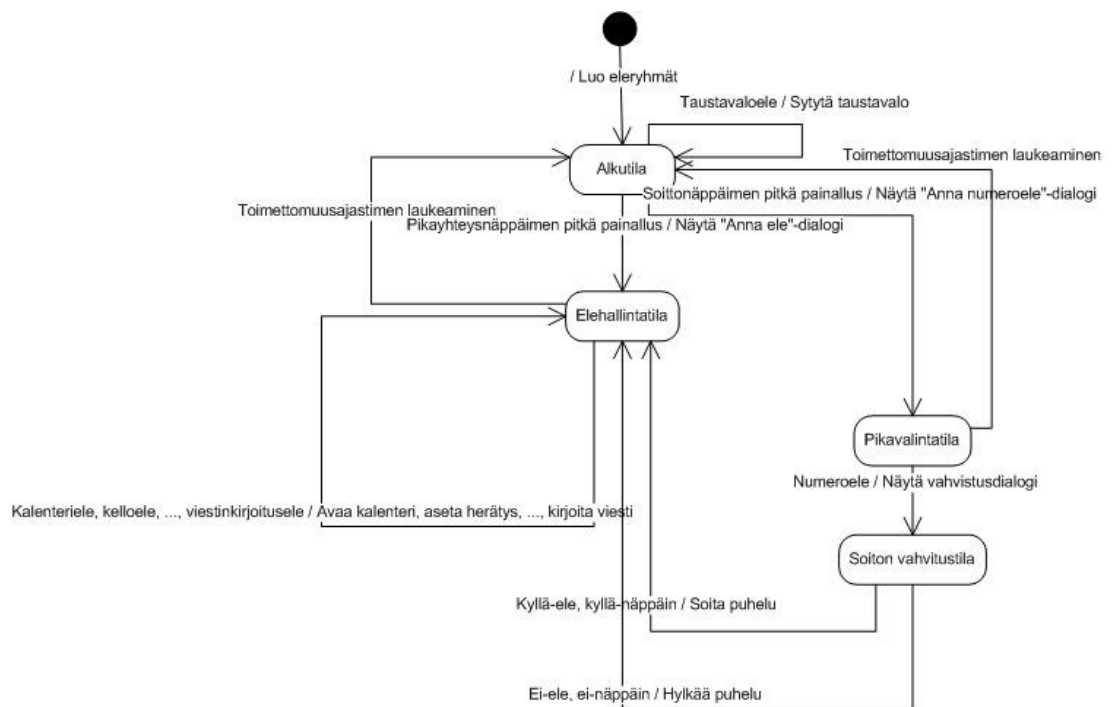
Elehallintasovellus kuuntelee tätä osoitetta ja vastaanotettuaan viestin laittaa puhelimen taustavalon päälle ja avaa näppäinlukon (viesti 21). Eleellä aktivoitavassa sovelluksessa aktivoidaan hallintaeleet. Hallintaeleet sääntöryhmän aktivointi (viestit 22-24) sekä tunnistetun kalenterieleen käsittely (viestit 26-29) tapahtuu samalla tavalla kuten aiemmin taustavaloryhmän aktivointi ja tunnistus.

Saatuana ”calendar”-viestin elehallintasovellus avaa kalenterisovelluksen tämän päivän näkymään (viesti 30). Kun puhelin on ollut käyttämättä viisi sekuntia, niin elehallintasovellus aktivoi taustavaloryhmän (viestit 31-32).

Kuvissa 4 ja 5 esitetään puhelimen elehallintasovelluksen toimintaa kuvaavat tilansiirtymäkaaviot Nokian 5500 Sport-puhelimelle toteutetussa sovelluksessa. Kuva 4 kuvaa tilansiirtymäkaavion sovelluksessa, joka aktivoituu aktivoitaessa puhelin. Kuvassa 5 taas kuvataan toimintaa näppäimellä aktivoitavassa sovelluksessa. (kts. 3.2. Vaatimusmäärittely)



Kuva 4. Puhelimen aktivoinnilla aktivoitavan sovelluksen tilakaavio.



Kuva 5. Näppäimellä aktivoitavan sovelluksen tilakaavio.

Kuvissa 4 ja 5 esitetään tilasiirtymät eri tavoilla aktivoitavissa sovelluksissa. Jokaisessa tilassa on yksi aktiivinen sääntöryhmä. Alkutilassa aktiivisena ovat taustavalo säännöt, elehallintatilassa hallintasäännöt, pikavalintatilassa numerosäännöt ja soiton vahvistustilassa hyväksymissäännöt. Lukuun ottamatta taustavaloa, kukin sääntöryhmä on aktiivisena viisi sekuntia aktivoinnin jälkeen.

Eleellä aktivoitavassa sovelluksessa (kuva 4) siirrytään elehallintatilaan aktivoimalla puhelin esimerkiksi taustavalo-oleella tai minkä tahansa näppäimen painalluksella. Näppäimellä aktivoitavassa sovelluksessa (kuva 5) sama tilasiirtymä tapahtuu pitkällä pikayhteysnäppäimen painalluksella. Pikavalintatilaan pääsee eleellä aktivoitavassa sovelluksessa elehallintatilasta toistamalla pikavalintaeleen. Näppäimellä aktivoitavassa sovelluksessa tähän tilaan pääsee suoraan alkutilasta pitkällä soittonäppäimen painalluksella.

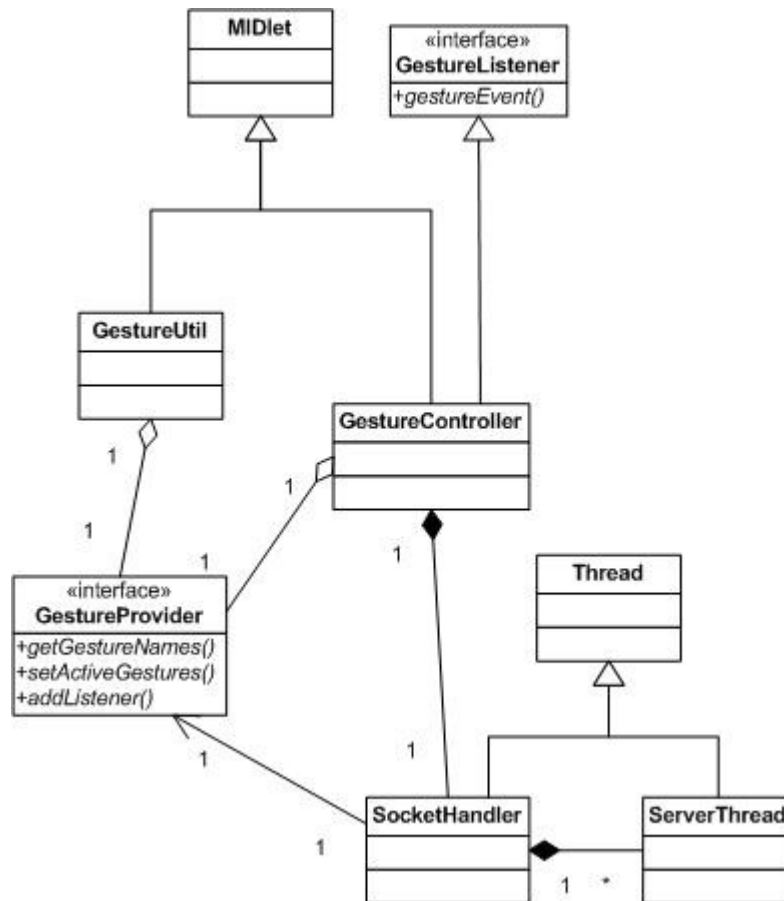
Pikavalintatilassa näytetään tilaan liittyvä dialogi käyttäjälle. Näppäimellä aktivoitavassa sovelluksessa vastaava dialogi näytetään myös elehallintatilassa. Tiloihin liittyvät dialogit on esitetty kuvassa 6.



Kuva 6. Puhelimen elehallintasovelluksen tiloihin liittyvät dialogit.

### 3.3.4. Luokkakaaviot

Kuvassa 7 on esitetty elehallintajärjestelmän (Gesture Management kts. kuva 2) merkittävimmät luokat. Eleentunnistusjärjestelmästä on kuvattu ainoastaan GestureListener-rajapinta ja GestureProvider-rajapinta.



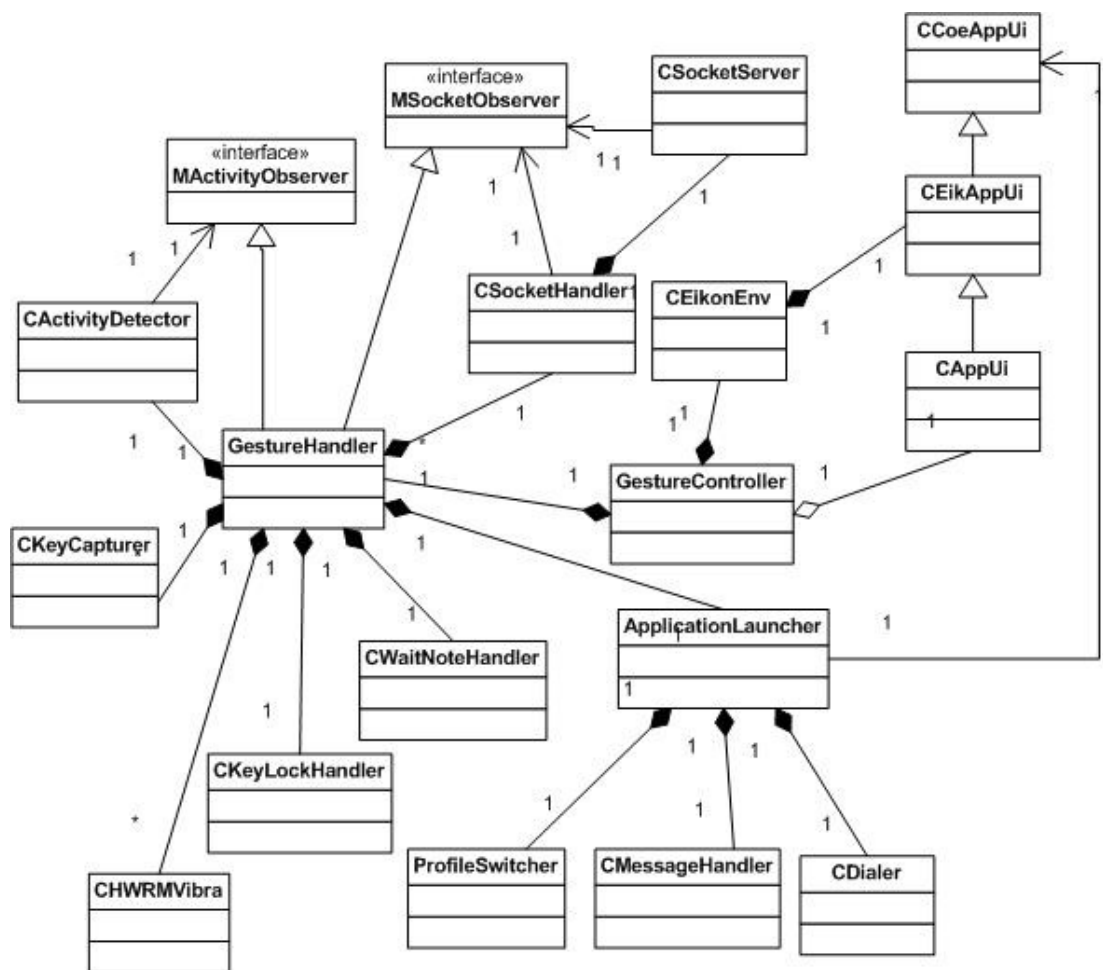
Kuva 7. elehallintajärjestelmän luokkakaavio.

Elehallinta-järjestelmän luokkien roolit on kuvattu taulukossa 6. GestureProvider on eleentunnistusjärjestelmän tarjoama rajapinta järjestelmää käyttäville sovelluksille. Tämän rajapinnan kautta sovellus voi kysyä listan kaikista nauhoitetuista eleistä (getGestureNames), asettaa aktiiviset eleet (setActiveGestures) tai rekisteröityä eleiden kuuntelijaksi (addListener). Eleiden kuuntelijaksi rekisteröityvän olion on toteutettava GestureListener-rajapinta. Tämän rajapinnan kautta olio saa tiedon tunnistetuista eleistä (gestureEvent). Jokaisella J2ME midlet-sovelluksella on oltava MIDlet-luokka, jonka avulla MIDP-järjestelmä huolehtii mm. sovelluksen käynnistymisestä ja sulkemisesta. GestureController on eleohjaimen MIDlet-luokka. GestureController omistaa SocketHandler-luokan, joka huolehtii elehallintajärjestelmän ja sovellusten välisestä kommunikoinnista. ServerThread-luokka toteuttaa TCP-palvelimen. Kutakin asiakassovellusta varten luodaan oma TCP-palvelin säie.

Taulukko 6. Elehallintajärjestelmän luokat

Luokka	Rooli
GestureListener	eleiden kuuntelurajapinta
GestureProvider	eleiden tunnistusrajapinta
GestureController	eleohjaimen MIDlet luokka
SocketHandler	elehallintajärjestelmän ja sovellusten välisen kommunikoinnin toteuttaja
ServerThread	TCP-palvelimen toteutus
GestureUtil	Eleapuohjelma

Puhelimen elehallintasovelluksen (Phone gesture controll application kts. kuva 3) tärkeimmät luokat on esitetty kuvan 8 luokkakaaviossa. Luokkien roolit on kuvattu taulukossa 7. Luokkakaaviosta on jätetty selkeyden vuoksi pois suurin osa perintähierarkiasta sekä muut vähemmän merkittävät luokat.



Kuva 8. Puhelimen elehallintasovelluksen luokkakaavio.

Taulukko 7. Puhelimen elehallintasovelluksen luokat

Luokka	Rooli
GestureController	sovelluksen käynnistäjä, luo CeikonEnv-ympäristön
GestureHandler	eleiden käsittelijä
MActivityObserver	puhelimen aktiivisuuden tarkkailurajapinta
CActivityDetector	puhelimen aktiivisuuden tarkkailija (käytetään vain eleellä aktivoitavassa sovelluksessa)
MsocketObserver	pistokeyhteyden tarkkailurajapinta
CsocketHandler	pistokeyhteyksien käsittelijä
CsocketServer	TCP-palvelin (käytetään vain käytettäessä TCP-protokollaa)
CKeyCaptorer	näppäintapahtumien kuuntelija (käytetään vain näppäimellä aktivoitavassa sovelluksessa)
CHWRMVibra	värinäpalautteen antaja, (järjestelmän luokka)
CKeyLockHandler	näppäinlukon käsittelijä
CWaitNoteHandler	eleidentunnistusdialogien näyttäjä (käytetään vain näppäimellä aktivoitavassa sovelluksessa)
ApplicationLauncher	puhelimen toimintojen ohjaaja
ProfileSwitcher	profiilin vaihtaja
CMessageHandler	viestien lähettäjä ja vastaanottaja
CDialer	pikavalintaan soittaja (N95:ssa erillinen sovellus)

### 3.3.5. Toteutus

Ohjelmiston toteutus on edennyt iteratiivisella mallilla, yhteistyössä eleentunnistus- ja nauhoitusjärjestelmän toteutuksen kanssa. Aluksi toteutettiin yksinkertainen sovellus, joka kuunteli tunnistettuja eleitä sekä tulosti eleet näytölle. Tässä sovelluksessa käytettiin hyväksi toteutettavuusanalyysivaiheessa tehtyä Java-Midletin ja Symbian C++-sovelluksen välisen kommunikoinnin testisovellusta.

Seuraavaksi sovellukseen integroitiin aiemmin toteutetusta testisovelluksesta puhelimen toimintojen hallinta. Alkuperäisen näytölle tulostamisen sijaan sovellus teki jonkin toteutettavuusanalyysissä toteutetuista toiminnoista. Nyt tietyllä eleellä pystyi siis tekemään tietyn toiminnon.

Java Midletistä sekä Symbian C++-sovelluksista muokattiin taustasovelluksia, joilla ei ole lainkaan näkyvää käyttöliittymää. Tästä seurasi puhelimen hallinnan osalta useita ongelmia, koska kaikkien Symbian C++ API:n käyttäminen ei ole mahdollista taustasovelluksesta, vaan niihin vaaditaan GUI-sovelluksen tarjoama ympäristö. Joihinkin tapauksiin tarvittiin vaihtoehtoinen API, jota voi käyttää taustasovelluksesta. Esimerkiksi viestin lähetyksen toteuttaminen ei onnistunut Send UI-API:a käyttämällä taustasovelluksesta, mutta sen sijaan viestin lähetyksen on mahdollista toteuttaa RsendAs-API:n avulla.

Joidenkin API:n käyttöön kuitenkin riittää se, että sovellukselle luodaan CeikonEnv-olio, jolle asetetaan CeikAppUi-luokasta peritty AppUi-olio. CeikonEnv-luokka toteuttaa ympäristön, jota tarvitaan käyttöliittymäkontrollien luontiin. Symbian C++-järjestelmä tarjoaa CeikonEnv-olion automaattisesti GUI-sovelluksille. CeikAppUi käsittelee sovelluksen käyttöliittymään liittyviä asioita kuten valikkoja ja ponnahdusikkunoita. [32] Näiden luominen ei tuo sovellusta näkyviin, mutta näin sovellus voi kuitenkin käyttää esimerkiksi CeikAppUi:n tarjoamaa ActivateViewL-metodia. Tällä metodilla voidaan aktivoida muiden sovellusten näkymiä.

Sovellusten on tarkoitus olla aina käynnissä, joten seuraavaksi elehallintasovellukseen lisättiin toiminto, jolla se käynnistyy puhelimen käynnistyksen aikana. Eleohjain muokattiin käyttämään Javan push-rekisteriä. Vastaava toiminnallisuus oli toteutettu ja testattu jo toteutettavuusanalyysivaiheessa. Koska sovellukset käynnistyvät automaattisesti ja ovat aina käynnissä, sovellusten käynnistysikonit ovat turhia. Symbian-sovelluksen ikonin poistaminen onnistui, mutta Java-Midletiltä sitä ei onnistuttu poistamaan.

Sovellusta testatessa pohdittiin useita eri tapoja aktivoida eleentunnistus. Tähän asti elehallinta oli ollut aina aktiivisena. Muita tapoja voisivat olla näppäimen painalluksella aktivoitava sekä eleellä aktivoitava elehallinta. Sovellus päätettiin toteuttaa kaikilla kolmella tavalla ja vertailla eri tavoilla toteutettujen sovellusten käytettävyyttä. Tässä vaiheessa toiminnallisuudeksi valittiin seuraavat: viestien lukeminen ja kirjoittaminen, pikavalintoihin soittaminen, kalenterin avaaminen nykyisen päivän kohdalta, profiilin vaihtaminen ja kello-sovelluksen avaaminen.

Tämän vaiheen jälkeen saatiin myös N95-laite käyttöön. Ensimmäisenä havaittu ero puhelinmallien välillä oli pikayhteysnäppäimen puuttuminen N95:sta. Näppäimellä aktivoitavassa sovelluksessa käytettävät näppäimet on esitetty kuvassa 9.



Kuva 9. Aktivointinäppäimet.

5500 Sport-puhelinmallissa käytetään pikayhteysnäppäintä hallintaeleiden aktivointiin. Pikayhteysnäppäin on hyvä sen takia, että se sijaitsee erillään muista näppäimistä puhelimen sivustalla. Tällaista näppäintä on helppo käyttää tarvitsematta edes katsoa puhelinta. Myös N95:ssa on näppäimiä puhelimen sivustalla, mutta niiden käyttö ei onnistunut ainakaan julkisten Symbian C++-APIen avulla. N95:ssa päädyttiin käyttämään oikean puoleista valintanäppäintä. Pikavalintaan soittamiseen käytetään molemmissa malleissa soitonäppäintä, joka on looginen näppäin puhelun soittamiseen. Näppäinten käyttö on toteutettu siten, että lyhyt näppäimen painallus toimii normaalisti, mutta pitkä painallus aktivoi elehallinta- tai pikavalintaan soittotilan. Sovelluksen aktivointi onnistuu myös silloin, kun puhelin on lepotilassa ja näppäimet ovat lukittuna.

Toinen puhelinmallien välinen ero liittyi pikavalintaan soittamiseen. Testattaessa huomattiin, että N95:lla pikavalintaan soittamistoiminto aiheutti Symbian C++-paniikin. Paniikki tarkoittaa virhetilannetta, joka johtaa ohjelman kaatumiseen. Yleensä paniikki ilmaisee ohjelmointivirhettä. Pikavalintoihin soittaminen oli toteutettu Ctelephony-APIa käyttämällä. Testatessa ilmeni, että tämän API:n käyttö N95:ssa oli mahdollista vain GUI-sovelluksesta. Ongelma ratkaistiin toteuttamalla erillinen GUI-sovellus, joka käynnistetään pikavalintaan soittotilanteessa. Jostain syystä tämä API taas ei toiminut 5500 Sport puhelimen kehitysalustalla GUI-sovelluksesta, joten oli tehtävä kaksi erilaista toteutusta, jotka erotetaan käännöslipuilla toisistaan.

Myös Java Midletit toimivat eri tavalla N95:ssa ja 5500 Sport:ssa. 5500 Sportissa UDP-yhteys toimi hyvin. N95:lla UDP-yhteyden käyttöön sisältyi merkittäviä ongelmia. Puhelin näyttää aina Javalla UDP-yhteyttä luotaessa ”Valitse yhteysosoite”-dialogin. Tällöin N95:ssa myös avataan verkkoyhteys. Verkkoyhteyden avaaminen ja yhteysosoitteen valinta tehdään myös puhelimen käynnistyksen tai sovelluksen asentamisen yhteydessä, jos käytetään UDP-porttia push-rekisterissä. N95:ssa push-rekisterin käyttö ei myöskään toiminut UDP-portin avulla käynnistyksen yhteydessä. Niinpä kommunikointi toteutettiin myös TCP-yhteyden avulla. TCP- ja UDP-protokollan käytön voi valita käännösaikana. Lisäksi N95:ssa näytetään sovelluksen käynnistymisen jälkeen tyhjä sovellusikkuna J2ME-taustasovelluksille. Käyttäjä joutuu häivyttämään tämän tyhjän sovellusikkunan menu- tai lopetusnäppäimen painalluksella. Tähän ei löytynyt mitään muuta ratkaisua, mutta tämä ei ole merkittävä ongelma.

Seuraavassa vaiheessa eleentunnistusjärjestelmää kehitettiin siten, että aktiivisena olevat eleet pystyy vaihtamaan suoritusajana. Eleet erotettiin erillisiksi ryhmiä siten, että taustavalo on omana ryhmänään ja muita ryhmiä ovat hallintaeleet, numeroeleet ja hyväksymiseleet (kts. taulukko 5). Tähän asti tunnistettavia eleitä oli kerrallaan 15, nyt enimmillään yhdeksän. Puhelimen ollessa alkutilassa (kts. kuvat 4 ja 5) aktiivisena on ainoastaan yksi ele, taustavalo. Tätä varten tarvittiin kommunikoinnin toteuttaminen molempiin suuntiin. Tähän asti oli riittänyt, että eleohjain lähetti viestejä tunnistetuista eleistä elehallintasovellukselle, mutta nyt elehallintasovelluksen oli kyettävä myös kertomaan aktivoitavat eleet eleohjaimelle. Niinpä toteutettiin eleohjaimen TCP- ja UDP-palvelimen toiminnallisuus. Tässä vaiheessa jätettiin aina aktiivisena oleva eleentunnistussovellus pois, koska sen oli käytettävyydestä huomattu tunnistavan runsaasti turhia eleitä (false-positive).

Seuraavaksi vaiheessa kehitettiin eleohjaimesta yleiskäyttöinen. Tätä varten kommunikointi oli toteutettava siten, että eleohjainta voi käyttää usea asiakassovellus (kts. taulukko 3). UDP-protokollaa käytettäessä tästä ei aiheutunut mitään lisätyötä.

Eleohjain yksinkertaisesti kuuntelee tiettyä UDP-porttia ja käsittelee siihen saapuvat paketit. Alla on esitetty eleohjaimen UDP-portin kuuntelun toteutus:

```
UDPDatagramConnection connection =
    (UDPDatagramConnection) Connector.open(url);
Datagram dg = connection.newDatagram(length);
connection.receive(dg);
```

Tämän jälkeen eleohjain odottaa kunnes UDP-porttiin tulee viesti. Saapuva viesti voi tulla miltä tahansa asiakassovellukselta. TCP-yhteydessä tarvitaan oma säie jokaista asiakasta varten:

```
ServerSocketConnection ssc =
    (ServerSocketConnection)Connector.open(url, mode);
while(true){
    SocketConnection socket =
        (SocketConnection) ssc.acceptAndOpen();
    ServerThread server = new ServerThread(this, s);
    server.start();
}
```

ServerSocketConnection-luokan acceptAndOpen-metodi odottaa kunnes TCP-porttiin avataan yhteys. Tämän jälkeen luodaan ServerThread-olio, jolle kutsutaan start-metodia, joka käynnistää säikeen. ServerThread säikeen toteutus on seuraavanlainen:

```
public void run(){
    is = sc.openDataInputStream();

    while (true) {
        boolean endMark = false;
        buf = new byte[maxLen];
        for(int i=0; i<buf.length && !endMark; i++){
            byte ch = (byte)is.read();
            if(ch!=(byte) '#'){
                buf[i] = ch;
            }else{
                endMark = true;
            }
        }
    } //while
    ...
} //run
```

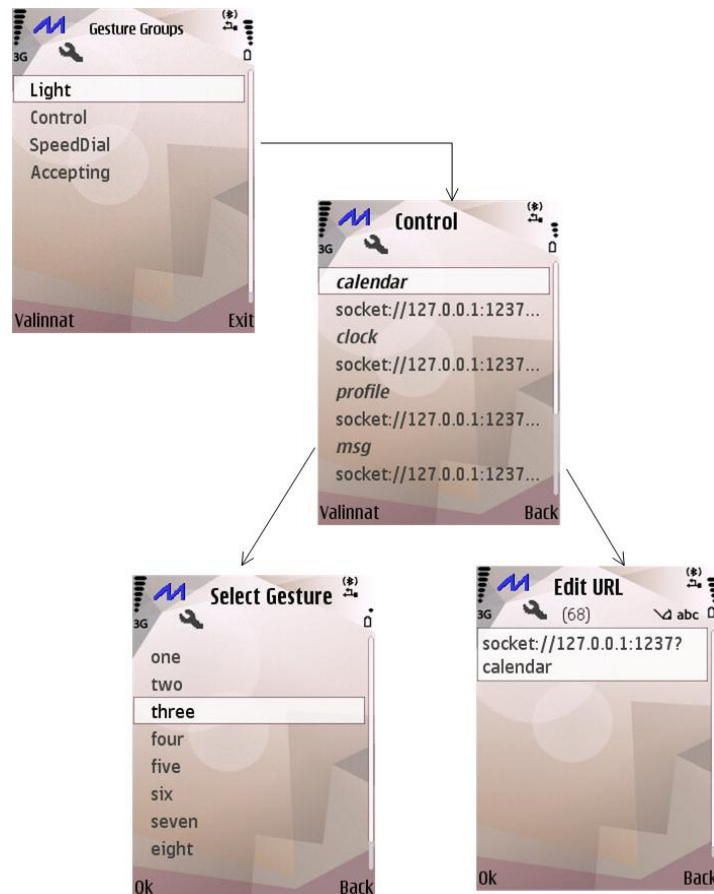
TCP-yhteydessä jokaisen pyynnön lopussa on lopetusmerkkinä '#'. Merkkejä luetaan kerrallaan aina lopetusmerkkiin asti. Vaihtoehtoinen ratkaisu olisi katkaista TCP-yhteys jokaisen pyynnön jälkeen ja luoda yhteys uudelleen seuraavaa pyyntöä varten.

Asiakassovelluksilta saapuneen pyynnön käsittely tapahtuu samalla tavalla sekä UDP- että TCP-toteutuksessa. Pyynnöt, joita asiakassovellukset voivat esittää sekä eleohjaimen toiminta kussakin tilanteessa on esitetty taulukossa 3. Jos pyyntö ei noudata taulukossa 3 esitettyä muotoa, niin se hylätään.

Tässä vaiheessa otettiin myös käyttöön (ele, URL)-säännöt, jotka eleohjain tallentaa Java RMS-tietokantaan. Sääntöryhmä toteutettiin yksinkertaisesti lisäämällä eleitä ja URL:ejä peräkkäin RecordStore-tietovarastoon. Eleohjain luo kullekin sääntöryhmälle oman RecordStore-tietovaraston, jonka tunnisteena toimii ryhmän nimi. Tietovarasto luodaan sääntöä luotaessa, mikäli sitä ei ole jo olemassa. Jos sääntöryhmä on olemassa, niin sääntö lisätään olemassa olevaan ryhmään. Eleohjain myös tarkistaa, että lisättävän säännön elettä tai URL:ia ei ole jo käytetty kyseisessä sääntöryhmässä (kts. 3.3.2. Sääntöryhmät).

Internet-sovelluksen hallintasovellus toteutettiin käyttämään REACHeS-järjestelmän valokuva-albumia, jolla voidaan näyttää kuvasarjoja WWW-selaimessa. REACHeS on järjestelmä, jota voidaan käyttää HTTP-pyyntöjen avulla [16]. Hallintasovellus on yksinkertainen Java Midlet, joka luo käynnistyessään sääntöryhmän, jossa palvelun hallintaan tarvittavat eleet liitetään HTTP-osoitteisiin (kts. taulukko 5). Käynnistettäessä sovellus aktivoi kyseisen ryhmän. Suljettaessa sovellus poistaa sääntöryhmän aktiivisuuden.

Kuvaruutukaappauksia eleapuohjelman toiminnasta on esitelty kuvassa 10. Eleapuohjelmalla on pääsy eleohjaimen luomaan tietokantaan, josta se lukee sääntöryhmät ja alkutilassa ohjelma näyttää listan kaikista sääntöryhmistä.



Kuva 10. Eleapuohjelma.

Toisessa näkymässä käyttäjä on avannut sääntöryhmän nimeltä Control. Tällöin eleapuohjelma lukee säännöt tietokannasta ryhmän nimen perusteella ja näyttää ryhmän säännöt. Käyttäjä voi muuttaa sovellusten luomia sääntöjä eleapuohjelman

avulla. Säännössä käytettävä ele voidaan vaihtaa avaamalla muokattava ele puhelimen valintanäppäimellä. Tällöin sovellus kysyy kaikki puhelimesta olevat eleet eleentunnistusjärjestelmältä ja näyttää ne käyttäjälle. Käyttäjä voi valita käytettävän eleen tästä listasta. Säännössä käytettävä URL:ia voidaan muokata avaamalla URL puhelimen valintanäppäimellä. Tällöin sovellus avaa näkymän, jossa käyttäjä voi muokata URL:ia. Valitsemalla ”Ok” tästä näkymästä, sovellus tallentaa URL:n kyseiselle säännölle.

Käyttäjä voi myös luoda uusia sääntöjä olemassa olevaan ryhmään tai luoda kokonaan uuden eleryhmän. Uutta sääntöä luotaessa näytetään ensin sama eleiden valintalista kuin muokatessa olemassa olevaa sääntöä. Kun käyttäjä on valinnut eleen, sovellus näyttää käyttäjälle tyhjän tekstikentän, johon käyttäjä voi syöttää URL:n.

## 4. TESTAUS

### 4.1. Yksikkö- ja komponenttitestaus

Yksikkötestausta suoritettiin jatkuvasti ohjelmistoa kehitettäessä. Toteutettavuusanalyysivaiheessa toteutetut pienemmät kokonaisuudet oli helpompi testata kattavasti kuin kokonainen järjestelmä. Pääpaino sovelluksia kehitettäessä oli käytettävyydestä ja muu testaus jäi kevyemmäksi. Tärkeintä on, että sovelluksissa ei ole käytettävyyttä merkitsevästi häiritseviä virheitä. Sovellusten välisen yhteyden testisovellus sekä sovellusten käynnistystestisovellus testattiin erillisinä komponentteina.

Virrankulutus osoittautui yhdeksi järjestelmän merkittävimmistä ongelmista. Virrankulutustestien tuloksia on lueteltu taulukossa 8. Nämä testit tehtiin yhteydettömässä tilassa siten, että puhelinta ei testin aikana käytetty. Normaalisissa käytössä ajat ovat vielä lyhyempiä, johtuen muusta virrankulutuksesta. Ensimmäinen eleentunnistimen versio pysyi täydellä akulla käynnissä vain 15 tuntia. Ilman järjestelmän käynnissä oloa, vastaava aika oli noin 19 vuorokautta. Mahdollisia syitä näin suureen virrankulutukseen ovat kiihtyvyyssanturin lukemisesta aiheutuva virrankulutus tai eleen tunnistusalgoritmien virrankulutus.

Taulukko 8. Virrankulutustestien tuloksia

Testi	Akun kesto
Ei elehallintaa	19 vuorokautta
Elehallintasovellus käynnissä, 16 elettä aktiivisena	15 tuntia
Elehallintasovellus käynnissä, 16 elettä aktiivisena, simuloitu data	23 tuntia
Ei elehallintaa, kiihtyvyyssanturin mittausten lukeminen	3 vuorokautta 20 tuntia
Elehallintasovellus käynnissä, yksi ele aktiivisena	27 tuntia
Elehallintasovellus käynnissä, ei yhtään elettä aktiivisena	28 tuntia
Elehallintasovellus käynnissä, yksi ele aktiivisena, simuloitu data	44 tuntia

Seuraavaksi testattiin pelkän kiihtyvyyssanturin lukemisen vaikutusta virrankulutukseen. Tällöin tulos oli 3 vuorokautta ja 20 tuntia. Tämän jälkeen testattiin tunnistusalgoritmien virrankulutusta simuloitulla datalla. Kiihtyvyyssantureita ei siis käytetty vaan algoritmille syötetään simuloitua dataa. Tällöin puhelin pysyi käynnissä 23 tuntia.

Näiden testien perusteella voidaan todeta, että merkittävin virrankulutus aiheutuu eleentunnistusalgoritmeista. Myös aktiivisten eleiden lukumäärän vaikutusta virrankulutukseen testattiin. Kun sovelluksen ensimmäisessä versiossa oli kerrallaan aktiivisena 16 elettä, puhelin pysyi käynnissä 15 tuntia. Kun ainoastaan taustavalo-ole oli aktiivisena, puhelin pysyi tällöin päällä 27 tuntia. Tilanteessa, jossa ei ollut yhtään aktiivista elettä, virran kulutus oli 28 tuntia. Tilanne parani siis huomattavasti, kun sääntöryhmät otettiin käyttöön, mutta virrankulutus on edelleen merkittävä ongelma. Myöskään aktiivisten eleiden vähentäminen nolnaan ei ratkaise tilannetta, koska virrankulutusta aiheuttaa myös kiihtyvyyssignaalin esiprosessointi.

## 4.2. Käytettävyystestaus

Käytettävyystestausta on Scott Weissin mukaan tehtävä mahdollisimman aikaisin ja riittävän usein [6]. Potentiaalisia käyttäjiä tälle sovellukselle ovat kaikki matkapuhelimen käyttäjät, joten testaajien valinta on yksinkertaista. Ainakin osan testaajista olisi hyvä olla henkilöitä, jotka eivät ole tekemisissä ohjelmistokehityksen kanssa.

Käytettävyystestausta tehtiin aluksi kolmella erilaisella sovelluksella. Testauksen aluksi testaajan kanssa nauhoitettiin kaikki sovelluksissa käytettävät eleet eli numeroeleet numeroille 1-9 sekä seitsemän hallintaelettä. Kutakin sovellusta testattiin ensin vuorokausi, jonka jälkeen testaajalle annettiin mahdollisuus nauhoittaa haluamansa eleet uudestaan. Uudelleennauhoituksen jälkeen kutakin sovellusta testattiin taas vuorokausi. Testauksen tavoitteena oli löytää eleiden tunnistukseen käytettävyydeltään paras menetelmä. Testattavat menetelmät olivat aina aktiivisena oleva eleentunnistus, tiettyä nappia painamalla aktivoitava eleentunnistus sekä millä tahansa eleellä aktivoitava eleentunnistus. Aktivoinnin vaativissa sovelluksissa eleentunnistus on aktiivisena 5 sekuntia, jonka jälkeen palataan alkutilaan.

Ensimmäisen vaiheen käytettävyystestaukseen osallistui viisi testihenkilöä. Merkittävimmiksi puutteiksi nousivat turhaan tunnistetut eleet sekä väärin tunnistetut eleet. Jokainen testaaja antoi ”ei”-vastauksen kysymykseen ”Oliko järjestelmän toiminta luotettavaa”. Tulos oli sama kaikilla kolmella sovelluksella. Kysymykseen ”Toimiko järjestelmä odotetusti” tuli yksi kyllä vastaus, joka koski näppäimellä aktivoitavaa sovellusta. Muuten sovellusten käyttö koettiin yleensä intuitiiviseksi, helpoksi ja sen koettiin nopeuttavan puhelimen käyttöä. Testauksen perusteella selvisi, että aina aktiivisena oleva elehallinta on käytettävyydeltään selkeästi huonoin. Myös millä tahansa käyttäjän eleellä aktivoitava elehallinta tuntui tekevän asioita itsekseen. Lisäksi jotkin toiminnot koettiin turhiksi tai niiden tekeminen koettiin helpommaksi perinteisellä menetelmällä. Tällaisia olivat erityisesti kellosovelluksen avaaminen sekä profiilin vaihto.

Näiden tulosten perusteella keskeytimme käytettävyystestauksen ja jatkoimme sovellusten kehittämistä. Aina aktiivisena olevan elehallintasovelluksen kehittäminen lopetettiin. Lisäksi eleellä aktivoitavaa sovellusta muokattiin siten, että elehallinta aktivoitui ainoastaan yhdellä eleellä tai aktivoimalla puhelin esimerkiksi minkä tahansa näppäimen painalluksella (kts. 3.3.3). Kellosovelluksen avaamisen sijaan elehallintasovellus muutettiin avaamaan kellosovelluksen aseta hälytys-dialogi. Profiilin vaihto tapahtui aiemmin kahdessa vaiheessa. Ensin annettiin profiilin vaihtoele, jonka jälkeen aktivoitava profiili valittiin listasta näppäimillä tai numeroeleellä. Toiminto muutettiin siten, että profiilieleellä vaihdetaan aktiivista profiilia yleinen-, äänetön- ja kokousprofiilin välillä. Myös pikavalintaan soittamiseen tehtiin muutos. Aikaisemmin käyttäjä joutui hyväksymään tai hylkäämään soitettavan puhelun näppäimillä. Sovellusta muokattiin siten, että käyttäjä voi vahvistaa puhelun näppäinten lisäksi myös eleillä.

Testausta muutettiin siten, että testaus aloitetaan kolmella eleellä, joilla testataan molempia sovelluksia yksi päivä. Tämän jälkeen nauhoitetaan loput hallintaeleet ja testataan taas päivä molempia sovelluksia. Viimeisessä vaiheessa nauhoitetaan numeroeleet ja hyväksyntäeleet. Tässä vaiheessa molempia sovelluksia testataan

kaksi päivää. Samalla kun nauhoitetaan uusia eleitä, uudelleennauhoitetaan huonosti tunnistuvat aiemmin nauhoitetut eleet.

Toisen vaiheen testeihin osallistui myös viisi testihenkilöä. Nyt käyttökokemukset olivat parempia kuin ensimmäisessä vaiheessa. Testaajat kokivat keskimäärin kahden eleen toimineen hyvin. Käyttökokemus yleensä parani selkeästi testauksen edetessä ja testaajan oppiessa toistamaan nauhoittamansa eleet tarkemmin. Tosin joillakin testaajilla osa eleistä lakkasi täysin tunnistumasta testauksen edetessä. Erään testaajan mukaan eleiden tunnistus onnistui parhaiten välittömästi nauhoituksen jälkeen. Yksi testaaja ei saanut ainoatakaan elettä toimimaan hyvin missään testauksen vaiheessa. Yhtä testaajaa lukuun ottamatta kaikilla ainakin osa eleistä tunnistui huonosti koko testauksen ajan tai sekoittui muiden eleiden kanssa. Eleiden tunnistus onnistui testaajilta parhaiten istuessa sekä paikallaan ollessa. Eleiden toistaminen koettiin lähes mahdottomaksi liikkeessa.

Suurin osa testaajista käyttäisi mieluummin sovellusta, jossa eleentunnistus aktivoidaan tietyn näppäimen painalluksella. Tätä sovellusta käytettäessä käyttäjä saa selkeän palautteen, milloin elehallinnan avulla voi ohjata puhelinta (kts. kuva 6). Dialogin näyttäminen antaa käyttäjälle myös varmuuden siitä, että sovellus todella on käynnissä. Molemmat sovellukset ilmaisivat värinällä käyttäjälle tunnistetun eleen. Tätä pidettiin yleisesti riittävänä palautteena. Useampi testaaja esitti ajatuksen, että tarvittaisiin palautetta myös tunnistamattomista eleistä. Epäonnistuneet yritykset toistaa eleitä, joista ei saanut mitään palautetta, koettiin turhauttaviksi. Käyttäjä ei voinut aina olla edes varma, että sovellus todella on käynnissä.

Kaikki testaajat olivat kiinnostuneita ottamaan käyttöönsä eleohjauksen myös omilla puhelimissaan. Vaatimuksena kuitenkin oli, että eleentunnistustarkkuus olisi parempi. Nykyisellään järjestelmä koettiin epäluotettavaksi ja eleiden toistamisen todettiin vaativan paljon harjoittelua. Muuten sovellus koettiin helppokäyttöiseksi. Nykyisen version ei koettu juurikaan nopeuttavan puhelimen käyttöä. Käytännöllisenä pidettiin sitä, että eleitä voitiin käyttää toimintojen pikavalintoina ilman, että vaadittiin valikoiden selaamista.

Osa testaajista olisi valmiita käyttämään elehallintaa myös julkisilla paikoilla. Julkisesti oltaisiin valmiita käyttämään pieniä ja huomaamattomia eleitä. Toisaalta juuri tällaiset eleet tunnistuvat helposti turhaan. Suurin osa testaajista muisti nauhoittamansa eleet hyvin. Eleiden muistamista helpotti, että testien aluksi testaajat kirjoittivat kuvauksen kustakin käytettävästä eleestä tai piirsivät eleen kuvan. Testien ajan testaajilla oli käytössään paperi, jolle eleet oli kuvattu.

Kaikki testaajat kokivat järjestelmän toiminnan riittävän nopeaksi. Mikään sovelluksen toiminnoista ei erottunut testien perusteella hyödyllisimmäksi tai hyödyttömimmäksi. Nykyisten toimintojen lisäksi sovellukselta toivottiin puhelun vastaamis- ja hylkäämistointoa sekä mahdollisuutta avata eri sovelluksia. Hyvä elehallintajärjestelmä olisi sellainen, joka tarjoaisi paljon toimintoja, joista käyttäjä voisi valita ne joita haluaa käyttää.

Eräs huomio testauksessa oli, että eleiden toistaminen vaatii runsaasti harjoittelua. Yleensä eleiden toiston voimakkuus kasvoi sitä mukaan, kun niiden käyttö tuli tutummaksi. Testaajia ohjeistettiin, että eleiden tulisi olla riittävän voimakkaita, mutta siitä huolimatta erityisesti ensimmäisellä kerralla nauhoitetut eleet olivat usein voimakkuudeltaan liian heikkoja. Tällaiset eleet sekoittuvat usein keskenään, ja tunnistuvat helposti itsekseen.

Eleiden toistamisen onnistumisessa oli merkittäviä eroja käyttäjien välillä. Joidenkin käyttäjien kanssa kaikki eleet jouduttiin opettamaan useaan kertaan, ennen

kuin eleentunnistin tunnisti ne. Toisille käyttäjille taas riitti, että eleet nauhoitettiin kerran. Yhdeksän erilaista elettä, jotka olivat yhtä aikaa aktiivisena, vaikutti olevan liikaa jokaiselle testaukseen osallistuneelle. Osalla käyttäjistä otettiin käyttöön vain pieni osa eleistä. Joillakin jo neljän toimivan eleen nauhoittaminen oli haastavaa.

## 5. POHDINTA

Tässä työssä toteutettu järjestelmä täyttää asetetut vaatimukset. Kuitenkin puhelimen sisäisten toimintojen hallinnasta saadut käyttökokemukset olivat enimmäkseen huonoja. Huonot käyttökokemukset johtuivat lähinnä siitä, että järjestelmä tunnisti käyttäjien eleet liian huonosti. Toisin sanoen tekniikka saatiin toimimaan, mutta eleiden toistaminen riittävän samanlaisena osoittautui haasteelliseksi. Jonkin verran tunnistustarkkuutta paransi aktiivisten eleiden ryhmittely. Käytettävyydestä osoitti myös, että eleiden käyttäminen vaatii runsaasti harjoittelua. Korkea oppimiskynnys saa todennäköisesti suuren osan käyttäjistä hylkäämään menetelmän.

Huonosti toimiva elehallinta vaatii käyttäjältä enemmän huomiota kuin perinteinen menetelmä. Tilanne on siis sama kuin puheohjauksen kanssa [8]. Lisäksi, kuten aiemmin todettiin (kts. 2.2.3.), eleentunnistustarkkuuden olisi oltava lähes 100 prosenttista, jotta käyttäjät kokisivat sen luotettavaksi. Toteutetussa järjestelmässä eleiden kokonaismäärä on liian suuri. Mitä enemmän puhelimesta on eleitä sitä vaikeampaa on valita selkeästi toisistaan erottuvat ja helposti toistettavat eleet. Käyttäjän on myös vaikeampi muistaa ja oppia toistamaan suuri määrä eleitä. Puhelimen elehallintasovellusta kehitettäessä oli ajatuksena, että käyttäjät voisivat valita kuhunkin toimintoon luonnollisesti liittyvän eleen. Tällainen ele olisi helppo muistaa. Yleensä käyttäjät kuitenkin valitsivat mieluummin eleitä, jotka ovat helppoja toistaa ja erottuvat hyvin muista saman ryhmän eleistä. Toimintoihin luontevasti liittyviä eleitä on vaikea keksiä.

Myös elehallintajärjestelmän virrankulutus osoittautui merkittäväksi ongelmaksi. Merkittävin virrankulutus aiheutui eleentunnistukseen liittyvästä prosessoinnista. Aina aktiivisena ole eleentunnistus ei siis ole toimiva ratkaisu sen aiheuttaman virrankulutuksen vuoksi.

Elehallinta on nykyään harvinainen käyttöliittymä matkapuhelimesta. Niinpä elehallinnan käyttöä ei ainakaan vielä koeta yleisesti hyväksyttäväksi. Sosiaalinen hyväksyttävyys kuitenkin tulee lisääntymään sitä mukaan, kun menetelmästä tulee yleisemmin käytetty.

Elehallintajärjestelmän kehittämisessä tärkeintä on parantaa eleentunnistuksen luotettavuutta ja ratkaista virrankulutusongelmat. Eleentunnistuksen luotettavuutta voidaan parantaa määräämällä eleiden kokonaismäärä riittävän pieneksi, kehittämällä eleiden uudelleen opettamista sekä toteuttamalla oppiva järjestelmä. Myös eleentunnistusalgoritmien kehittämistä tulee tutkia.

Eleiden määrän pienentäminen vaatii myös sovellusten kehittämistä. Sovelluksista tulee tehdä enemmän asiayhteydestä riippuvia, jotta kaikki halutut asiat voidaan tehdä. Asiayhteyden on oltava sellainen, että käyttäjä voi helposti ymmärtää, mitkä säännöt kulloinkin ovat aktiivisina. Aktiivinen sääntöryhmä voidaan ilmaista esimerkiksi asiayhteyteen liittyvällä dialogilla.

Puhelimen elehallintasovelluksen hallintaeleiden määrän voisi rajoittaa kolmeen eleeseen. Puhelimen ollessa perustilassa aktiivisena olisivat säännöt, joilla aktiivista sääntöryhmää vaihdetaan. Aktivoitavia sääntöryhmiä voisivat olla esimerkiksi profiilinvaihto-, ajanhallinta- sekä viestisääntöryhmä. Profiilinvaihtosääntöryhmä sisältäisi yhden eleen, jolla vaihdetaan profiilia. Ajanhallintaryhmässä olisi kaksi elettä, toinen kalenteria ja toinen kelloa varten. Viestisääntöryhmässä olisi myös kaksi elettä, yksi lukemista ja toinen kirjoittamista varten. Lisäksi puhelimesta on myös paljon tapahtumia, joihin voidaan liittää aktiivinen sääntöryhmä. Hälyttävän puhelun aikana voidaan aktivoida puhelunvastaamissääntöryhmä. Tässä ryhmässä

olisi kaksi elettä: toisella vastataan puheluun ja toisella hyljätään puhelu. Muita tapahtumia, joiden tapahtuessa voitaisiin toimia asiayhteydestä riippuvasti ovat esimerkiksi käynnissä oleva puhelu, uuden viestin saapuminen sekä herätyskellon tai kalenterin hälytykset.

Myös internet-valokuva-albumin toiminnot voidaan toteuttaa pienemmällä määrällä eleitä. Aluksi aktiivisena voisi olla ainoastaan aloitusele. Aloituseleen jälkeen voitaisiin aktivoida säännöt seuraavan kuvan näyttämiseksi sekä esityksen lopettamiseksi. Lisäksi tässä sääntöryhmässä voisi olla myös kolmas ele, jolla aktivoitaisiin sääntöryhmä, joka sisältää eleet edellisen, ensimmäisen ja viimeisen kuvan näyttämiseksi.

Eleiden opettamista tulee myös kehittää. Ihanteellinen tilanne olisi se, että käyttäjä voi helposti missä tahansa itse opettaa minkä tahansa käyttämänsä eleen. Tällöin huonosti tunnistuvat eleet voidaan uudelleennauhoittaa aina kun se on tarpeen. Opettaminen tulee toteuttaa siten, että se on mahdollista suoraan puhelimesta, ilman PC-sovellusta. Opettaminen tulisi myös kyetä tekemään useammassa vaiheessa. Tällöin eleiden opetuksessa käytetyt eleet eivät olisi liian samankaltaisia vaan mukaan saataisiin riittävästi hajontaa.

Eleentunnistusjärjestelmästä tulee kehittää oppiva järjestelmä. Tämä järjestelmä muuttaisi eleiden malleja eleiden toistotavan muuttuessa. Tällainen menetelmä voisi vähentää harjoittelun tarvetta. Jos puhelimen kiihtyvyyssanturidatasta voidaan päätellä, että on annettu ele, jota ei kuitenkaan tunnisteta, niin järjestelmä voisi kysyä käyttäjältä, mitä elettä hän tarkoitti. Sovellus voisi ehdottaa elettä, joka on lähimpänä luettua kiihtyvyyssdataa. Oppiva järjestelmä päivittäisi eleiden malleja käyttäjän valintojen mukaan.

Ratkaisu virrankulutusongelmiin olisi, että kiihtyvyyssdatan lukeminen lopetettaisiin kokonaan, kun aktiivisia eleitä ei ole. Tällöin puhelimen elehallintasovelluksesta voisi tehdä version, jossa ei käytettäisi lainkaan taustavaloellettä. Järjestelmä on suurimman osan ajasta tässä tilassa. Kiihtyvyyssdatan lukeminen aloitettaisiin vasta sitten, kun asetettaisiin aktiivisia eleitä.

Elehallintajärjestelmää voisi kehittää pelkän elehallinnan sijaan havaitsemaan myös ravistelua sekä staattista asentoa. Ravistelua voitaisiin käyttää eleentunnistuksen sijaan aina kun tarvitaan vain yhtä sääntöä. Ravistelun havaitsemisen toteuttaminen on huomattavasti eleentunnistusta yksinkertaisempaa. Tällöin myös virrankulutus saataisiin pienemmäksi. Staattisen asennon havaitseminen taas olisi tarpeellinen menetelmä esimerkiksi valikoiden tai listojen selailuun sekä elehallintapelien toteutukseen.

## 6. YHTEENVETO

Tässä työssä on tutkittu matkapuhelimen käyttöliittymiä. Työssä käsitellyjä käyttöliittymiä ovat perinteinen käyttöliittymä ja puheohjaus sekä fyysiset käyttöliittymät. Fyysisistä käyttöliittymistä käsiteltiin kosketusnäyttöä, fyysistä valintaa sekä elehallintaa.

Työssä toteutettiin yleiskäyttöinen elehallintajärjestelmä. Järjestelmä laajentaa aiemmin toteutettua eleentunnistusjärjestelmää ja tarjoaa sovelluksille yksinkertaisen ohjelmointikielestä riippumattoman tavan järjestelmän hyödyntämiseen.

Työn alussa tehtiin toteutettavuusanalyysiä eri ohjelmointikielten soveltuvuudesta toteutukseen, ohjelmien välisestä kommunikoinnista sekä puhelimen hallinnasta ohjelmallisesti. Toteutettavuusanalyysin pohjalta tehtiin järjestelmän vaatimusmäärittely. Työssä toteutettiin elehallintajärjestelmän lisäksi kaksi järjestelmää käyttävää sovellusta. Työ eteni iteratiivisesti ja puhelimen elehallinta sovellukselle tehtiin käytettävyydestä.

Eleentunnistuksen tunnistustarkkuutta on parannettu määrittämällä vain tarvittava osa eleistä aktiivisiksi. Kehitetyn järjestelmän avulla voidaan myös vähentää tarvittavien eleiden määrää, käyttämällä samoja eleitä useassa eri merkityksessä sääntöryhmien avulla. Sovellusten välinen kommunikointi toteutettiin TCP- ja UDP-pistokkeiden avulla. Tämä mahdollistaa järjestelmän käytön millä tahansa ohjelmointikielellä toteutetulla sovelluksella.

Käytännössä elehallinta osoittautui haasteelliseksi käyttää. Käyttäjän tulee kyetä toistamaan eleet riittävän samanlaisina, kuin ne on nauhoitusvaiheessa opetettu. Eleentunnistustarkkuutta tulee parantaa ja eleiden määrä tulee rajata riittävän pieneksi.

## 7. LÄHTEET

- [1] Subramanya S. & Byung K. (2006) User Interfaces for Mobile Content. IEEE Computer Society Press, osa 39, numero 4, s. 85-87
- [2] Choi E.S., Bang W.C., Cho S.J., Yang J., Kim D.Y. & Kim S.R. (2005) Beatbox Music Phone: Gesture-based Interactive Mobile Phone using a Tri-axis Accelerometer. Industrial Technology, 2005. ICIT 2005. IEEE International Conference 14.-17.12.2005, s. 97- 102.
- [3] Ubilife (luettu 28.11.2007), <http://www.ubilife.fi/>
- [4] Kauppila M, Pirttikangas S, Su X & Riekkii J. (2007) Accelerometer Based Gestural Control of Browser Applications. International Workshop on Real Field Identification (RFid2007), in Conjunction with UCS2007, Tokyo Denki University, Tokyo, Japan, 25.10.2007, s. 2-17.
- [5] Kiljander H. (2004) Evolution and usability of mobile phone interaction styles. Teknillisen korkeakoulun tietoliikenneohjelmistojen ja multimedian julkaisuja, Espoo.
- [6] Weiss S. (2002) Handheld Usability. Usable Products Company, John Wiley&Sons, LTD.
- [7] Kuivakari S, Huhtamo E, Kangas S & Olsson E. (1999) Keholliset käyttöliittymät. Tekes, Teknologian kehittämiskeskus. Paino-Center Oy, Sipoo.
- [8] Graham R & Carter C. (2000) Comparison of speech input and manual control of in-car devices while on the move. Personal and Ubiquitous Computing, Springer London, osa 4, numerot 2-3, s. 155-164.
- [9] Rabiner L.R. (1989) A tutorial on hidden Markov models and selected applications in speech recognition. Proceedings of the IEEE (1989), osa 77, numero 2, s. 257-286.
- [10] I. Kiss. (2001) On Speech Recognition in Mobile Communications, Väitöskirja, Tampereen teknillinen yliopisto, Tampere.
- [11] Vaughan-Nichols S. (2007) New Interfaces at the Touch of a Fingertip. IEEE Computer Society Press, osa 40, numero 8, s. 12 – 15.
- [12] Ailisto H, Pohjanheimo L, Väikkynen P, Strömmer E, Tuomisto T & Korhonen I. (2006) Bridging the physical and virtual worlds by local connectivity-based physical selection. Personal and Ubiquitous Computing, Springer London, osa 10, numero 6, s. 333-344.
- [13] <http://europe.nokia.com/A4307094>, (luettu 16.4.2008)

- [14] Riekkilä J, Salminen T & Alakärppä I (2006) Requesting Pervasive Services by Touching RFID Tags. *IEEE Pervasive Computing*, Jan.-Mar. 2006. 5(1)40-46.
- [15] Riekkilä, J. (2007) RFID and smart spaces, *Int. J. Internet Protocol Technology*, Vol. 2, Nos. 3/4, pp.143–152.
- [16] Riekkilä J., Sanchez I., Pyykkönen M. (2008) Universal Remote Control for the Smart World, *The 5th International Conference on Ubiquitous Intelligence and Computing (UIC08)*, Oslo, Norway, June 23-25.
- [17] Semico, Press Releases (luettu 16.4.2008), <http://www.semico.com/press/press.asp?id=177>
- [18] Ailisto H., Matinmikko T., Häikiö J., Ylisaukko-oja A., Strömmer E., Hillukkala M., Wallin A., Siira E, Pöyry A, Törmänen V., Huomo T., Tuikka T., Leskinen S. & Salonen J. (2007) Physical browsing with NFC technology. Espoo. VTT Tiedotteita. VTT Research Notes 2400.
- [19] Sony Ericsson, Phone portfolio, W380i (luettu 16.4.2008), <http://www.sonyericsson.com/cws/corporate/products/phoneportfolio/specification/w380i>
- [20] Apple, iPhone, Features, Accelerometer (luettu 16.4.2008) <http://www.apple.com/iphone/features/index.html#accelerometer>
- [21] Nokia Europe, Nokia 5500 Sport, Support (luettu 16.4.2008) <http://europe.nokia.com/A4252185>
- [22] Samsung's Digital World, Press Release (16.4.2008) [http://www.samsung.com/aboutsamsung/electronicsglobal/investorrelations/newspublicdisclosure/pressrelease/PressRelease.asp?seq=20050112\\_0000096403](http://www.samsung.com/aboutsamsung/electronicsglobal/investorrelations/newspublicdisclosure/pressrelease/PressRelease.asp?seq=20050112_0000096403)
- [23] Mäntyjärvi J, Kela J, Korpipää P & Kallio S. (2004) Enabling fast and effortless customisation in accelerometer based gesture interaction. *ACM International Conference Proceeding Series*, osa 83, s. 25-31.
- [24] Kallio S, Kela J, Mäntyjärvi J. (2003) Online Gesture Recognition System for Mobile Interaction. *VTT Elektronikka. Systems, Man and Cybernetics*, *IEEE International Conference on 5.-8.10. 2003* osa 3, s. 2070- 2076.
- [25] Forum Nokia, resources for mobile applications developers (luettu 8.4.2008), <http://www.forum.nokia.com>
- [26] The MIDP 2.0. Push Registry (luettu 16.4.2008), <http://developers.sun.com/mobility/midp/articles/pushreg/>
- [27] Symbian Developer Network, Java papers, J2ME MIDP (luettu 8.4.2008), [http://developer.symbian.com/main/oslibrary/java\\_papers/midp.jsp](http://developer.symbian.com/main/oslibrary/java_papers/midp.jsp)

- [28] series60 code (luettu 8.4.2008), <http://snippets.dzone.com/tags/series60>
- [29] Connected Limited Device Configuration (CLDC) (luettu 8.4.2008)  
<http://java.sun.com/products/cldc/>
- [30] Pylvänäinen T. (2005) Accelerometer Based Gesture Recognition Using Continuous HMMs. Springer Berlin / Heidelberg, osa 3522 / 2005, s. 639-646.
- [31] MIDP Database Programming Usign RMS: a Persisten Storage for MIDlets (luettu 17.4.2008),  
<http://developers.sun.com/mobility/midp/articles/persist/>
- [32] App framework UIKON in C++ component reference (luettu 17.4.2008),  
[http://www.symbian.com/developer/techlib/v9.1docs/doc\\_source/reference/reference-cpp/N101A2/index.html](http://www.symbian.com/developer/techlib/v9.1docs/doc_source/reference/reference-cpp/N101A2/index.html)