Erkki Mäkinen (toim.)

Pieniä tietojenkäsittelytieteellisiä tutkimuksia

Syksy 2004

TIETOJENKÄSITTELYTIETEIDEN LAITOS
TAMPEREEN YLIOPISTO

D-2004-2

TAMPERE 2004

TAMPEREEN YLIOPISTO
TIETOJENKÄSITTELYTIETEIDEN LAITOS
JULKAISUSARJA D – VERKKOJULKAISUT
D-2004-2, JOULUKUU 2004

Erkki Mäkinen (toim.)

Pieniä tietojenkäsittelytieteellisiä tutkimuksia

Syksy 2004

TIETOJENKÄSITTELYTIETEIDEN LAITOS
33014 TAMPEREEN YLIOPISTO

ISBN 951-44-6188-6
ISSN 1795-4274

Sisällysluettelo

Ohjelmistonhallinta ja ohjelmistopiratismi

Tiina Harra

Tiivistelmä.
Tutkimuksessa tarkastellaan tietokoneohjelmia tekijänoikeudellisesta näkökulmasta. Työssä pyritään selventämään tietokoneohjelmien käyttöoikeuksien eli lisenssien luonnetta. Työn keskeisin sisältö liittyy ohjelmistonhallintaan ja sen merkitykseen yrityksille ja muille organisaatioille sekä ohjelmistopiratismiin ja sen vaikutuksiin.

Tietokoneohjelmat on suojattu tekijänoikeuslailla ja ohjelmien käyttö ilman sen tekijänoikeuden haltijan lupaa on laitonta. Lisenssi eli käyttöoikeus on edellytys ohjelman lailliselle käytölle. Yrityksissä ja organisaatioissa ohjelmistolisenssien hallinta on tärkeää, eikä ainoastaan laillisuusnäkökulmasta. Laittomien ohjelmistojen käyttö tuo yrityksille ja organisaatioille monia negatiivisia seuraamuksia. Lisäksi ohjelmistopiratismin haittavaikutukset ovat merkittävät niin kansallisella kuin kansainväliselläkin tasolla.

Avainsanat ja -sanonnat: Käyttöoikeus, lisenssi, lisenssienhallinta, ohjelmistonhallinta, piratismi, tekijänoikeus, tietokoneohjelma.
CR-luokat: K.5

1. Johdanto

Tietokoneohjelmien käyttö on useimmille yrityksille arkipäivää ja ohjelmistojen käytöstä saatavat hyödyt ovat yrityksille moninaiset. Jotta ohjelmistoja pystyttäisiin yrityksissä täysin hyödyntämään, on ohjelmistojen käyttöä hallinnoitava yhtä hyvin kuin muitakin yrityksen menestymiseen vaikuttavia osatekijöitä.

Ohjelmistot muodostavat yrityksissä jopa 25 % yrityksen tietotekniikkaan varatusta budjetista [BSA, a]. Toisena tärkeänä näkökulmana on laillisuus. Tietokoneohjelmat on suojattu tekijänoikeuslailla ja niiden käyttö ilman tekijänoikeuden haltijan lupaa on laitonta. Laittomien ohjelmistojen käyttö johtaa ohjelmistopiratismiin, joka on maailmanlaajuisesti yhä suureneva ongelma.

Ohjelmistopiratismi sekä ohjelmistojen laillisen käytön mahdollistamat lisenssit eli käyttöoikeudet ovat hyvin ajankohtaista asiaa. Ohjelmistovalmistajien maailmanlaajuinen etujärjestö Business Software Alliance (BSA) pyrkii ohjelmistopiratismin vähentämiseen ja laillisen verkkoympäristön edistämiseen kouluttamalla tietokoneiden käyttäjiä ohjelmistojen tekijänoikeuksista sekä tiedottamalla piratismista. BSA on perustettu vuonna 1988 ja sillä on edustajia 60 maassa. Suomessa BSA Finland aloitti toimintansa vuonna 1994. [BSA, a]

BSA:n lisäksi esimerkiksi SIIA (The Software & Information Industry Association) pyrkii lisäämään tietoa ohjelmistoista, niiden laillisesta käytöstä sekä tukee piratisminvastaista toimintaa.

Tutkimuksessa tarkastellaan aluksi tietokoneohjelmia tekijänoikeuslain näkökulmasta. Työssä keskitytään tarkastelemaan lainsäädäntöä Suomessa. Luvussa 3 käydään läpi ohjelmistolisenssejä; lisenssien luonnetta ja eri lisenssityyppejä. Tämän jälkeen kerrotaan lyhyesti ohjelmiston elinkaaresta yrityksissä. Luvussa 5 tarkastellaan ohjelmistonhallintaa ja sen vaiheita. Luvun 6 aiheena ovat ohjelmistonhallinnan hyödyt yrityksille ja muille organisaatioille.

Ohjelmistonhallintaosuuden jälkeen otetaan esille ohjelmistopiratismi. Tutkimuksessa on ensin esitelty eri piratismityyppejä, jonka jälkeen kerrotaan syksyllä 2004 valmistuneesta ohjelmistopiratismitutkimuksesta. Lopuksi pohditaan ohjelmistopiratismin vaikutuksia muun muassa maailman taloudelle.

2. Tietokoneohjelmat ja tekijänoikeus Suomessa

Tietokoneohjelmia koskevat nimenomaiset säännökset otettiin mukaan tekijänoikeuslakiin vuonna 1991 (11.1.1991/34). Laissa ei ole määritelty tarkasti mikä on tietokoneohjelma. Hallituksen esityksen 161/1990 mukaan tietokoneohjelma on joukko käskyjä, jotka saavat tietokoneen suorittamaan halutun toiminnon. Tekijänoikeuslaki rinnastaa tietokoneohjelman kirjalliseen teokseen. Tekijänoikeuslain 1 §:ssä sanotaankin:

> "Sillä, joka on luonut kirjallisen tai taiteellisen teoksen, on tekijänoikeus teokseen, olkoonpa se kaunokirjallinen tahi selittävä kirjallinen tai suullinen esitys, sävellys- tai näyttämöteos, elokuvateos, valokuvateos tai muu kuvataiteen teos, rakennustaiteen, taidekäsityön tai taideteollisuuden tuote taikka ilmetköönpä se muulla tavalla.
>
> Kirjallisena teoksena pidetään myös karttaa sekä muuta selittävää piirustusta tai graafista taikka plastillisesti muotoiltua teosta sekä tietokoneohjelmaa."

Tekijänoikeus suojaa ilmaisumuotoa, ei ideaa. Tietokoneohjelmissa tekijänoikeussuojan kohde on käytännössä koodi. Ohjelman looginen rakenne, algoritmit, ideat ja periaatteet eivät kuulu tekijänoikeuden piiriin. [Hollmén, 1998; Takki, 2002]

Tekijänoikeussuojan kohteena on teos, kuten tietokoneohjelma, ja teoksen konkreettinen yksittäinen fyysinen tallenne on teoskappale, esimerkiksi siis CD-levylle tallennettu tietokoneohjelma. Tekijänoikeuslain 27 §:n 2 momentissa sanotaan, että tekijänoikeuden luovutus ei sisälly kappaleen luovutukseen. Tämä on se tärkein seikka, joka tulisi aina muistaa tietokoneohjelmien kohdalla.

Tekijänoikeus antaa sen haltijalle oikeuden määrätä teoksesta valmistamalla siitä kappaleita, saattamalla sen yleisön saataviin, muuttamattomana tai muutettuna. Tekijänoikeuden haltija voi kuitenkin antaa suostumuksensa teoksen kopioimiseen, levittämiseen sekä muuttamiseen sovittua korvausta vastaan sopimuksin rajoitetussa laajuudessa: tästä on lisensioimisessa kyse.

Tekijänoikeuslaki sisältää joitakin säännöksiä, jotka koskevat nimenomaan tietokoneohjelmia. Osa näistä säännöksistä on indispositiivisia eli pakottavia, jolloin sopimuksissa ei voida pätevästi rajoittaa niissä määriteltyjä käyttäjän oikeuksia. Osa säännöksistä on puolestaan dispositiivisia eli tahdonvaltaisia. Niitä noudatetaan vain, mikäli asianosaiset eivät ole muuta sopineet. [Takki, 2002]

Tekijänoikeuslain perusteella sillä, joka on laillisesti hankkinut tietokoneohjelman, on oikeus ottaa siitä varmuuskopio, tutkia ohjelman toimintaperiaatteita normaalikäytön yhteydessä sekä kääntää ohjelman koodi, mutta vain sikäli kun se ohjelmien yhteentoimivuuden saavuttamiseksi tarvittavien tietojen hankkimiseksi on välttämätöntä. Lisäksi, ellei sopimuksessa sitä ole kielletty, saa laillisesti tietokoneohjelman hankkinut muunnella ohjelmaa sovitun käyttötarkoituksen puitteissa ja korjata ohjelmassa esiintyviä virheitä. [TekL]

3. Lisenssi eli käyttöoikeus

Tietokoneohjelmien lisensioiminen perustuu siihen, että tietokoneohjelma on tekijänoikeuslain suojaama teos. Tekijänoikeuden omistajalla on oikeus määrätä teoksen levittämisestä, kopioimisesta ja muuttamisesta sekä asettaa näille ehtoja.

Tietokoneohjelmat luovutetaan asiakkaalle säännönmukaisesti siten, että toimittaja myöntää asiakkaalle lisenssin eli käyttöoikeuden ohjelmaan. Asiakas saa maksamaansa lisenssimaksua vastaan sopimuksessa määritellyn oikeuden käyttää ohjelmaa, mutta ohjelman tekijänoikeus jää toimittajalle tai kolmannelle osapuolelle. Asiakas ei siis omista ohjelmaa sinällään, vaan hän saa ainoastaan siihen käyttöoikeuden. [Takki, 2002]

3.1. Käyttöoikeuden laajuus

Sopimusehdoista tulisi aina käydä selkeästi ilmi, millaiseksi ajaksi ja minkä luonteisena käyttöoikeus on myönnetty. Käyttöoikeus voi olla myönnetty määräämättömäksi ajaksi (perpetual license), jolloin käyttöoikeus on ikuinen, ellei sopimusta rikota. Asiakas maksaa sopimuksen tullessa voimaan tietyn lisenssimaksun, eikä ole tämän jälkeen enää velvollinen maksamaan muita maksuja käyttöoikeuden jatkumisesta. Käyttöoikeus voi olla myös myönnetty toistaiseksi, jolloin käyttöoikeudesta maksetaan määräkausittaisia maksuja.

Lisenssi pysyy tällöin voimassa niin kauan kuin asiakas maksaa käyttöoikeusmaksunsa. Käyttöoikeus voi myös olla määräaikainen, jolloin asiakas saa käyttöoikeuden ohjelmistoon esimerkiksi kolmen vuoden ajaksi maksamalla sovitun lisenssimaksun. Määräajan jälkeen käyttöoikeudesta on sovittava erikseen. [Takki, 2002]

Käyttöoikeuden laajuuteen liittyy sekä kysymys organisaatiorajoista että organisaation sisällä olevista rajauksista. Käyttöoikeus voi rajoittua asiakasyrityksen juridisten rajojen sisäpuolelle, tai käyttöoikeutta voidaan hyödyntää myös asiakkaan mahdollisten konserniyhtiöiden tarpeissa. Organisaation sisällä käyttöoikeus voi olla rajattu monin eri tavoin. Laitekohtainen (designated equipment) lisenssi liittyy tiettyyn laiteyksilöön, johon ohjelmisto on asennettu. Lisenssin ollessa henkilökohtainen (named users), vain nimetyt käyttäjät saavat käyttää ohjelmistoa kulloinkin käytössään olevassa laitteistossa, mutta yleensä ainoastaan yhdessä laitteistossa kerrallaan. Lisenssi voi rajata sallitun käytön käyttäjien enimmäismäärään, joko absoluuttisesti tai yhtäaikaisesti (concurrent users). Ohjelman käyttö voi olla rajattu myös tietoteknisen käyttöympäristön perusteella, jolloin lisenssi voidaan kytkeä tiettyyn käyttöjärjestelmään tai sen tiettyyn versioon, tiettyyn tietokantaohjelmaan tai sen tiettyyn versioon, tai tiettyyn laitetyyppiin. Lähiverkkoympäristöön voidaan edellyttää hankittavaksi erilliset palvelin- eli serverilisenssit ja käyttäjälisenssit. [Takki, 2002]

3.2. Lisenssityypit

Ohjelmistovalmistajien ja erilaisten lisenssityyppien määrä on valtava, mutta lisensointitavat ovat kuitenkin suurimmaksi osaksi samankaltaisia. Pääsääntöisesti lisenssejä on kahta tyyppiä: yksittäisen käyttäjän ja usean käyttäjän lisenssit. Usean käyttäjän lisenssejä kutsutaan myös volyymilisensseiksi, koska ne sallivat useiden kopioiden käyttämisen samasta ohjelmistosta. [BSA, b]

Yksittäisiä lisenssejä kutsutaan usein loppukäyttäjän käyttöoikeussopimuksiksi (end user license agreement, EULA). Tämänkaltaisia lisenssejä ovat niin sanotut shrink wrap -lisenssit, jotka koskevat pakettituotteita, joita ostetaan suoraan liikkeiden hyllystä. Pakettituotteet sisältävät useimmiten suojamuoviin pakatun ohjelmiston, rekisteröintikortin ja käyttöoppaan. Avaamalla pakkauksen ostaja tulee sidotuksi valmistajan vakiosopimusehtoihin. Sopimusehdot voivat olla kiinnitettyjä pakkaukseen tai ne voivat ilmestyä tietokoneen näytölle, kun ohjelmaa asennetaan ensimmäistä kertaa, jolloin käyttäjän tulee hyväksyä sopimusehdot jotain näppäintä painamalla. Yksittäisiä lisenssejä voi myös tulla mukana tietokoneissa, joilla on esiasennettuja ohjelmia tai niitä voidaan hankkia myös verkkokauppojen kautta. [BSA, b; Rebeiro, 1997]

Volyymilisenssit eroavat yksittäisten käyttäjien käyttöoikeussopimuksista. Yhtäaikaiskäyttöä koskevat lisenssit ovat eräs volyymilisenssien muoto ja ne sallivat käyttäjän asentaa tietyn maksimimäärän ohjelmistokopioita tietylle määrälle tietokoneita, samalla rajoittaen tietyllä yksittäisellä hetkellä käytössä olevan ohjelman käyttäjien määrää. [BSA, b]

Per seat ja per server -lisenssejä tarvitaan useimmiten tietyntyyppisten palvelinohjelmistojen kanssa. Per server -lisenssi koskee yhtäaikaiskäyttöä ja se oikeuttaa tietyn maksimimäärän käyttäjiä olemaan yhteydessä palvelimeen millä tahansa yksittäisellä hetkellä. Per seat -lisenssi edellyttää pääsääntöisesti yhden lisenssin jokaiselle laitteelle, joka on yhteydessä palvelimeen. [BSA, b]

Käytönaikaisella eli run time -lisenssillä jotain tiettyä ohjelmistoa voidaan käyttää ainoastaan tietyn sovelluksen yhteydessä, sen käytön aikana, ei itsenäisesti. Käytönaikainen lisenssi liittyy esimerkiksi tietokantaohjelmiin. [Takki, 2002]

Erityisryhmille, kuten oppilaitoksille ja julkisyhteisöille, on lisäksi olemassa monia lisensointiohjelmia. Näiden lisenssien käyttöön liittyy yleensä aivan erityisiä ehtoja. [BSA, b]

4. Ohjelmiston elinkaari

Ennen varsinaisen ohjelmistonhallinnan käsittelyä on tärkeätä ymmärtää ohjelmiston elinkaari yrityksissä. Ohjelmistonhallinnan keinoin saadaan ohjelmistoista parempi hyöty sen eri elinkaaren vaiheissa. Tässä kuvataan ohjelmiston elinkaari ohjelmistonhallinnan näkökulmasta.

Ohjelmiston elinkaari alkaa yrityksessä havaitusta tarpeesta. Tarkempien tarvemäärittelyjen jälkeen yleisin tapa edetä hankinnassa on tarjouspyyntöjen jättäminen potentiaalisille ohjelmistotoimittajille. Tarjousten perusteella tehdyn toimittaja- ja ohjelmistovalinnan jälkeen ryhdytään neuvottelemaan sopimuksen ehdoista. Ehtojen sopimisen jälkeen tehdään tilaus. Erityisesti ohjelmistonhallinnan näkökulmasta on tärkeää varmistaa, että toimitus vastaa tilausta: alkuperäiset asennusmediat, ohjekirjat ja käyttöoppaat sekä kaikki lisenssiin liittyvät asiakirjat tulee olla mukana toimituksessa. [BSA, b; Peregrine, 2004b]

Jakelu- ja asennusvaiheen perustana on lisenssisopimuksen ehdot. Ne määrittelevät, ketkä ovat oikeutettuja käyttämään kyseistä ohjelmaa ja kuinka monelle ja mille laitteille ohjelma asennetaan. Jakelu voi tapahtua sähköisesti tai fyysisesti ja ohjelman asennus voidaan tehdä joko automaattisesti tai manuaalisesti. [Peregrine, 2004b]

Ohjelmien asennusta seuraa ylläpitovaihe, jossa tärkeintä on varmistua siitä, että lisenssisopimuksen ehtoja noudatetaan. Yrityksessä tapahtuvat muutokset heijastuvat usein myös ohjelmistojen käyttöön. [Peregrine, 2004b] Jonkin

tietyn ohjelman käyttäjä voi esimerkiksi vaihtua, jolloin on tärkeää pitää myös lisenssitiedot ajantasaisina. Mikäli ohjelman käyttäjä vaihtuu, tulee ohjelma poistaa vanhan käyttäjän koneelta ja asentaa uuden käyttäjän koneelle, ei vain kopioida ohjelmaa uudelle käyttäjälle. Säännölliset inventoinnit varmistavat, että koneille asennettuja ohjelmia on lisenssisopimuksia vastaava määrä ja ainoastaan niitä oikeutetut käyttäjät käyttävät ohjelmia.

Ohjelmiston elinkaari päättyy ohjelman poistoon. Yrityksessä tapahtuneiden muutosten myötä vanha ohjelma ei kenties enää vastaakaan yrityksen tarpeita. Elinkaaren seurannalla pystytään ennakoimaan ohjelmistoista sekä myös laitteista aiheutuvia päivitystarpeita ja kustannuksia, jotka eivät enää vastaa yrityksen tarpeita. [BSA, b; Efecte, 2004]

5. Ohjelmistonhallinnan vaiheet

Asianmukainen ohjelmistonhallinta jaetaan useimmiten joko kolmeen [Microsoft, 2004] tai neljään vaiheeseen [BSA, 2001] sen mukaan, pidetäänkö ohjelmistonhallintaan liittyvien yrityskohtaisten toimintaohjeiden laadintaa yhtenä vaiheena. Tässä tutkimuksessa noudatetaan BSA:n [2001] tapaa jakaa ohjelmistonhallinta neljään vaiheeseen.

5.1. Vaihe 1: toimintaohjeiden laadinta

Ensimmäisenä vaiheena yrityksen tulisi laatia ohjelmistonhallintaan liittyvät toimintaohjeet. Toimintaohjeista tulisi käydä ilmi ainakin ohjelmistojen hankintaa ja asennusta koskevat käytännöt sekä ainoastaan laillisten ohjelmien käytön tärkeys yrityksessä. Vielä suositeltavampaa olisi, että toimintaohjeet kattaisivat koko ohjelmiston elinkaaren [BSA, c]. Toimintaohjeet tulisi jakaa yrityksen jokaiselle työntekijälle, sekä saattaa vielä yleisesti nähtäville esimerkiksi yrityksen Intranetiin. [BSA, 2001; Peregrine, 2004a; Microsoft, 2002] Useat eri tahot suosittavat yrityksiä myös tekemään päätöksen ohjelmistohankintojen keskittämisestä sekä ostamaan ohjelmistoja ainoastaan tunnetuilta, valtuutetuilta jälleenmyyjiltä. Keskittämällä ohjelmistohankinnat paitsi helpotetaan ohjelmistonhallintaa, lisäksi mahdollistetaan esimerkiksi volyymialennusten saaminen. [BSA, 2001; Microsoft, 2002]

5.2. Vaihe 2: inventointi

Toimintaohjeiden laatimisen jälkeen toisena vaiheena on selvittää tarkalleen kaikki yrityksen koneille asennetut ohjelmat. Inventointi voidaan suorittaa joko manuaalisesti tai jonkin tähän tarkoitukseen tehdyn ohjelman avulla automaattisesti. Pienissä yrityksissä manuaalinen inventointi on hyvinkin mahdollinen, mutta jos koneita on paljon, on ymmärrettävä, että inventoinnin tekemiseen manuaalisesti kuluu paljon aikaa ja henkilöresursseja. Tästä syystä mm. BSA ja

monet ohjelmistotoimittajat suosittelevatkin yrityksille jonkin inventointityökalun käyttöönottoa [BSA, 2001; BSA, c; Peregrine, 2004].

Mikäli inventointi suoritetaan manuaalisesti, kaikista yrityksen koneille asennetuista ohjelmista tulee kirjata ylös vähintään ohjelman nimi, versionumero ja toimittaja. Kun jokaisen koneen sisältämien ohjelmien tiedot on kirjattu, laaditaan näistä yhteenveto, josta nähdään asennettujen ohjelmien määrät kokonaisuudessaan. [BSA, 2001; Microsoft, 2004] Monet inventointityökalut tallentavat hyvinkin runsaasti erilaista tietoa suoritettujen laiteskannausten pohjalta ja helpottavat huomattavasti ohjelmistonhallintaa yrityksissä. Niiden ominaisuuksiin kuuluu usein myös erilaisten raporttien tuottaminen, mikä on eduksi esimerkiksi uusia ohjelmistohankintoja tai päivitystarpeita perusteltaessa.

Inventointi tulee suorittaa myös ohjelmistojen oheismateriaalille. Kaikki ohjelmistojen oheismateriaali, kuten levykkeet, CD:t, alkuperäiset käsikirjat, lisensseihin liittyvät asiakirjat sekä laskut ja muu materiaali, jolla voidaan todistaa ohjelmien laillisuus, tulee kerätä yhteen. Näistä asiakirjoista kootaan tiedot voimassa olevista lisensseistä seuraavaksi tapahtuvaa vertailua varten. Inventoinnin jälkeen oheismateriaali on hyvä säilyttää yhdessä turvallisessa paikassa. [BSA, 2001; Microsoft, 2004]

5.3. Vaihe 3: vertailu

Kolmantena vaiheena on suorittaa vertailu inventoinnissa kerättyjen tietojen perusteella. Koneilta löytyneiden ohjelmien tietoja verrataan lisenssisopimuksista kerättyihin tietoihin ja näin ollen varmistutaan siitä, että yrityksen käytössä on ainoastaan lisensoituja ohjelmia. [BSA, 2001]

Ensimmäiseksi verrataan asennettujen ohjelmien määrää hankittujen lisenssien määrään ohjelmistoittain. Mikäli koneilta on löytynyt luvattomia ohjelmia, tulee ne poistaa välittömästi tai hankkia niiden käytön oikeuttavat lisenssit. Tässä vaiheessa voidaan yrityksessä myös havaita käyttämättömiä lisenssejä, jolloin kyseinen ohjelma voidaan vielä asentaa useammalle koneelle voimassa olevien lisenssiehtojen mukaisesti. [Bigler, 2003; Peregrine, 2004b; Microsoft, 2004; Microsoft, 2002]

Seuraavaksi verrataan asennettujen ohjelmien käyttäjien määriä lisenssisopimuksissa olevien käyttäjien määriin ohjelmistoittain. Näin varmistutaan siitä, että lisenssisopimuksissa olevia ehtoja noudatetaan käyttäjien määrien tai nimettyjen käyttäjien osalta. [Peregrine, 2004b]

Viimeisenä kannattaa vielä varmistua siitä, että kaikkia laillisesti käytössä olevia ohjelmia todella käytetään. Ohjelmien käyttöasteet on mahdollista selvittää tiettyjen ohjelmien avulla tai tähän tarkoitukseen voidaan käyttää

myös työntekijöille laadittuja kyselylomakkeita. [Peregrine, 2004b; Microsoft, 2004]

5.4. Vaihe 4: valvonta ja ylläpito

Neljäs vaihe on yrityksen toimintaohjeiden noudattamisen valvonta ja tietojen pitäminen ajantasaisina. Uusien ohjelmien hankinnat ja vanhojen poistot tulee aina kirjata ylös. Yrityksessä olisi hyvä nimetä tietojen ylläpidosta ja ohjelmistonhallinnasta vastuussa olevat henkilöt. Jatkossakin inventointi olisi hyvä suorittaa säännöllisesti, jotta yrityksessä voidaan olla varmoja siitä, ettei koneilla ole laittomia ohjelmia käytössä. [BSA, 2001]

6. Ohjelmistonhallinnan hyödyt

Tehokkaan ohjelmistonhallinnan hyödyt eivät rajoitu pelkästään ohjelmistoostojen keskittämisestä saataviin volyymialennuksiin, vaan yritys hyötyy monin eri tavoin. Ohjelmistonhallinnan avulla yrityksessä tiedetään, mitä lisenssejä tarvitaan ja mitkä ovat tarpeettomia. Rahallisia säästöjä saadaan siis paitsi volyymialennuksilla, myös sillä, että hankitaan ainoastaan lisenssejä, joita todella tarvitaan. Kun lisenssejä hallitaan keskitetysti, on ohjelmistoinvestointien takaisinmaksuajat helpompi nähdä. Ohjelmistonhallinnan avulla päästään myös hyötymään tehokkaasti ohjelmien päivitysmahdollisuuksista, mikä tulee huomattavasti edullisemmaksi kuin uusien ohjelmien ostaminen. [BSA, 2001; Microsoft, 2002]

Yrityksissä saattaa esiintyä usein yhteensopivuusongelmia tiedonvälityksessä, mikäli samasta ohjelmasta on käytössä monia eri ohjelmistoversioita. Tietämällä mitä ohjelmistoja ja mitä versioita yrityksessä on, voidaan helpommin ja nopeammin päästä eroon näistä ongelmista. Mahdollisimman pitkälle vakioidusta työympäristöstä hyötyvät lisäksi yrityksen teknisen tuen palvelut ja käyttäjäkoulutus. [Microsoft, 2002]

Myös yrityksen tietoturva paranee ohjelmistonhallinnan myötä. Ostamalla ohjelmistot ainoastaan valtuutetuilta jälleenmyyjiltä vähennetään laittomien kopioiden riskiä yrityksessä. Kun työntekijät tietävät, mistä ohjelmistonhallinnassa on kyse ja noudattavat yrityksen laatimia toimintaohjeita, vältytään laittomasti asennetuista ohjelmista ja vähennetään samalla virusten riskiä. [Microsoft, 2002]

Ohjelmistonhallinnasta saatavat rahalliset säästöt ja yrityksen tuottavuuden parantuminen ovat merkittäviä seikkoja ohjelmistonhallinnan tärkeyden ja kannattavuuden kannalta, mutta kaikkein tärkeimpänä on kuitenkin yrityksen toiminnan laillisuus. Jotta toiminta olisi laillista, tulee yrityksen käyttää ainoastaan laillisia ohjelmia. Lisäksi useat lisenssisopimukset velvoittavat

lisenssinsaajan pitämään rekisteriä lisenssien määrästä, niiden käyttäjistä sekä koneista, joille ohjelma on asennettu. Tämän lisäksi monissa lisenssiehdoissa on myös lauseke, joka mahdollistaa lisenssinmyöntäjän suorittamaan tarkastuksen lisenssien käytön oikeellisuuden varmistamiseksi lyhyelläkin varoitusajalla, kuten ilmoittamalla asiasta 30 päivää aikaisemmin. Monet tahot ovat sitä mieltä, että tällaiset tarkistukset tulevat lisääntymään huomattavasti lähivuosien aikana. [Bigler, 2003; O'Brien, 2004; Peregrine, 2004a]

7. Piratismityypit

Ohjelmistopiratismi on suuri ongelma maailmanlaajuisesti, kuten myöhemmin esiteltävä BSA:n teettämä tutkimuskin osoittaa. Seuraavaksi kuvataan ohjelmistopiratismin eri muotoja.

Ohjelmistopiratismista on tullut yleinen termi ohjelmistojen laittomalle kopioinnille. Koen ja Im [1997] jakavat piratismin kolmeen eri luokkaan: kaupallinen piratismi (commercial piracy), yhteisöissä tapahtuva piratismi (corporate piracy) ja ohjelmistovarkaus (softlifting). Kaupallisella piratismilla he tarkoittavat ohjelmistojen laitonta kopiointia myynti- ja levitystarkoitukseen. Yhteisöissä tapahtuva piratismi on tyypillisimmillään sitä, että esimerkiksi yrityksessä yksi ohjelma kopioidaan monen käyttäjän koneille tai asennetaan verkkoon monen käyttäjän ulottuville. Yhteisöissä tapahtuvan piratismin tarkoituksena on harvoin taloudellisten hyötyjen tavoittelu. Ohjelmistovarkaudesta on puolestaan kyse, kun esimerkiksi lainataan ystävältä jokin ohjelma ja kopioidaan se itselle tai tuodaan töistä jokin ohjelma myös kotikäyttöön. Ohjelmistovarkaus on näistä kolmesta kenties se harmittomimmaksi luultu piratismin muoto, jota monet jopa ovat ennen erehtyneet pitämään laillisena. [Koen and Im, 1997] Uskon kuitenkin, että nykyisin lähes kaikki ymmärtävät toiminnan laittomuuden, vaikka sitä harjoittavatkin. Syynä tähän on varmasti se, että yksityiseen käyttöön kopioiminen on vielä ollut lain ylläpitäjien ulottumattomuudessa.

BSA [BSA, 2001] jakaa ohjelmistopiratismin viiteen yleisimpään muotoon, jotka ovat loppukäyttäjäpiratismi (end user piracy), palvelinohjelmiston lisenssioikeudet ylittävä käyttö (client-server overuse), Internet-piratismi (Internet piracy), esiasennetut ohjelmistot (hard-disk loading) ja ohjelmistojen väärentäminen (software counterfeiting).

Loppukäyttäjäpiratismia on BSA:n [2001] mukaan esimerkiksi yhden lisensoidun ohjelman kopiointi useampiin koneisiin kuin ohjelman lisenssi sallii, sekä asennukseen ja jakeluun tarkoitetun ohjelmistolevykkeen tai CD:een kopioiminen. Tältä osin loppukäyttäjäpiratismi vastaa Koenin ja Imin [1997] yhteisöissä tapahtuvaa piratismia. Tämän lisäksi loppukäyttäjäpiratismiksi

luetaan myös päivityspalvelujen hyväksikäyttö laittomiin ohjelmaversioihin sekä lisenssiehtojen vastainen ohjelmien käyttö käyttörajoitettujen ohjelmien kohdalla, kuten esimerkiksi opiskelu- tai tutkimuskäyttöön tarkoitettujen ohjelmien käyttö niiden ehtojen vastaisesti. Verkon kautta käytössä olevan ohjelman sallitun käyttäjämäärän ylittyessä puhutaan palvelinohjelmiston lisenssioikeudet ylittävästä käytöstä. [BSA, 2001]

Internet-piratismiksi luetaan www-sivut, joilta voi ladata koneelleen ohjelmia ilmaiseksi, ja kaikenlaiset muut sivut, jotka tarjoavat laittomia ohjelmistoja. Internetistä onkin mahdollista löytää lähes mikä tahansa markkinoilla oleva ohjelmisto laittomana versiona. Internet-piratismi uhkaa lisäksi sähköistä kaupankäyntiä. [BSA, 2001]

Piratismityyppinä esiasennetuista ohjelmistoista on kyse silloin, kun tietokoneen kovalevylle on valmiiksi asennettu laittomia ohjelmistokopioita myyjän toimesta. Ohjelmistojen väärentäminen puolestaan tarkoittaa tekijänoikeuksilla suojatun ohjelmiston laitonta kopioimista. Kopiointi voi tapahtua kenen tahansa toimesta. [BSA, 2001] Monet eri tahot kehottavatkin kiinnittämään huomiota tiettyihin seikkoihin laittomien kopioiden tunnistamiseksi, kuten esimerkiksi pakkaukseen ja oheismateriaaliin, myyntihintaan ja –paikkaan, CD-levyn merkintöihin ja väriin [BSA, 2001; TTVK].

SIIA [2004] on jakanut piratismin kymmeneen eri esiintymismuotoon: ohjelmistovarkaus (softlifting), työasemien rajoittamaton pääsy palvelimella sijaitsevaan ohjelmistoon lisenssiehtojen vastaisesti (unrestricted client access), esiasennetut ohjelmistot (hard-disk loading), ainoastaan tiettyjen laitteistojen yhteydessä myytävien ohjelmistojen kopiointi ja myynti erikseen (OEM piracy / unbundling), ei-kaupallisen ohjelmiston kaupallinen käyttö (commercial use of noncommercial software), väärentäminen (counterfeiting), CD-R-piratismi (CD-R piracy), Internet-piratismi (Internet piracy), CD-valmistajien ottamat ylimääräiset kopiot ja niiden myynti (manufacturing plant sale of overruns and scraps) ja vuokraaminen (renting). Nämä eri muodot eivät kuitenkaan sulje toisiaan pois, vaan saattavat esiintyä yhdessä. Lähes kaikki näistä piratismimuodoista on jo tullut esille edellä esitetyissä BSA:n [2001] sekä Koenin ja Imin [1997] piratismiluokitteluissa, vaikka käytetyt termit vaihtelevatkin. SIIA [2004] on lähinnä tarkemmin erotellut piratismin eri esiintymismuodot toisistaan. Lisäksi jotkin näistä piratismimuodoista ovat lähinnä Amerikassa esiintyviä, kuten ohjelmistojen vuokraus elokuvien tapaan. [SIIA, 2004]

8. Piratismin laajuus ja vaikutukset

Edellisessä luvussa esitetyt ohjelmistopiratismin monet eri muodot antavat jo viitteen siitä, miten laajalle piratismi on levinnyt. BSA:n heinäkuussa 2004

julkistama kymmenes ohjelmistopiratismitutkimus antaa karuja lukemia laittomien ohjelmistojen määristä niin Suomessa kuin muuallakin maailmalla.

8.1. Piratismitutkimuksen tulokset

BSA:n International Data Corporationilla (IDC) teettämän ohjelmistopiratismitutkimuksen mukaan maailmanlaajuisen ohjelmistopiratismin taso oli 36 % vuonna 2003. Tutkimuksen mukaan Suomessa ohjelmistoista oli laittomia 31 %, kun taas EU-maiden laittomien ohjelmistojen määrä oli jopa 37 %. Taloudellisten tappioiden arvioidaan olevan Suomessa jopa 125 miljoonaa euroa ja EU:ssa yli 8 miljardia euroa. [BSA, a]

Verrattaessa Pohjoismaiden lukuja keskenään, todetaan että Suomea korkeampi piratismitaso oli tutkimuksen mukaan ainoastaan Norjassa, jossa se oli 32 %. Alhaisin piratismitaso Pohjoismaissa oli Tanskassa, 26 %, joka oli myös alhaisin taso koko EU:n alueella. EU:n korkein piratismitaso oli Kreikassa, jossa jopa 63 % ohjelmistoista oli tutkimuksen mukaan laittomia. Maailmanlaajuisesti kaikkein alhaisin piratismitaso oli USA:ssa, 22 %.[BSA, a]

Piratismitasojen alueellisiin eroihin voivat vaikuttaa monet eri seikat, kuten ohjelmistojen hinta, ihmisten tulot, lainsäädäntö, laittomien ohjelmistojen saatavuus sekä kulttuuriset erot. Piratismitutkimuksen tuloksia luettaessa kannattaa lisäksi huomioida, että piratismitaso vaihtelee maidenkin sisällä eri kaupungeissa ja eri toimialoilla. [BSA, 2004] Suomen BSA:n puheenjohtajan Jari Heikkisen mukaan suuryrityksissä ollaan kyllä tarkkoina lisenssiasioiden kanssa, mutta pienille ja keskisuurille yrityksille laittomien ohjelmien käyttö on ongelma. Heikkisen mukaan myös toimialojen välillä on selvästi havaittavia eroja laittomien ohjelmien käytössä. Pankeissa ja vakuutusalalla lisenssiasiat ovat kunnossa, mutta esimerkiksi insinööri- ja arkkitehtitoimistoissa on esiintynyt monia muita aloja enemmän ongelmia ohjelmien laillisuuden kanssa. [Aamulehti, 2004]

IDC:n tekemässä tutkimuksessa arvioitiin myynti- ja markkinatietoa 86 maassa kuudella markkina-alueella: Aasia, Itä-Eurooppa, Latinalainen Amerikka, Lähi-Itä ja Afrikka, USA ja Kanada sekä Länsi-Eurooppa. Vaikka tämä olikin jo kymmenes BSA:n piratismia koskeva tutkimus, eivät aikaisemmat tulokset ole suoraan verrattavissa tähän uusimpaan tutkimukseen tutkimusmenetelmien muuttumisen vuoksi. Varmaa kuitenkin on, että ohjelmistopiratismi on yhä kasvava ongelma. [BSA, 2004]

8.2. Piratismin vaikutukset

Ohjelmistopiratismilla on monia negatiivisia vaikutuksia paitsi laittomia ohjelmistoja käyttäville yrityksille ja organisaatioille, myös maailman taloudelle. Piratismin aiheuttamat menetykset vaikuttavat suoraan ohjelmistoteolli-

suuden kannattavuuteen. Piratismissa menetettyjen tuottojen vuoksi ohjelmistovalmistajilla on yhä vähemmän resursseja tutkimukseen ja uusien tuotteiden kehittämiseen. Ohjelmistovalmistajien on pakko siirtää kustannukset asiakkaille, ja tämä näkyy ohjelmistojen korkeina hintoina.

Paikallinen ohjelmistoteollisuus kärsii yrittäessään kilpailla ulkomailta tulevien laadukkaiden piraattituotteiden kanssa. IDC:n huhtikuussa 2003 tekemä tutkimus piratismin taloudellisista vaikutuksista osoittaa, että alentamalla piratismin tasoa voitaisiin saavuttaa merkittäviä taloudellisia hyötyjä niin uusien työpaikkojen kuin saatavien verotulojenkin muodossa. Tässä BSA:n toimesta tehdyssä tutkimuksessa huomioitiin ohjelmistopiratismin taloudelliset vaikutukset hyvin suppeasti. Tutkimuksessa piratismissa menetetyt tulot laskettiin käyttäen hyödyksi maan laillisten ohjelmistomarkkinoiden määrää sekä maan piratismitasoa. [BSA, 2004]

Samaiseen IDC:n tutkimukseen viitaten BSA Finlandin puheenjohtaja Jari Heikkinen on tuonut esille piratismin aiheuttamat menetykset ohjelmistovalmistajille. Heikkisen mukaan laskemalla Länsi-Euroopan nykyistä piratismitasoa 10 % vuoteen 2006 mennessä voitaisiin saada jopa 250000 uutta työpaikkaa ja 18 miljardia euroa verotuloja. Mikäli piratismitaso pysyy maailmanlaajuisesti samana, nousee laittomien ohjelmistojen arvo IDC:n arvion mukaan yli 30 miljardin euron seuraavan viiden vuoden aikana. [BSA, a]

Vaikka monet eri tahot taistelevatkin jatkuvasti ohjelmistopiratismia vastaan, on tekijöitä, jotka myös kasvattavat piratismia. Näinä tekijöinä on pidetty talouden hidasta kasvua, kehittyvien maiden uusia käyttäjiä sekä laittomien ohjelmistojen lisääntynyttä saatavuutta erityisesti Internetissä ja vertaisverkoissa. Ilman tehokkaita lakeja ja näiden valvontaa ohjelmistopiratismin pelätään kasvavan Internetinkin käytön kasvaessa. [BSA, 2004]

9. Yhteenveto

Jo pelkästään oikeudelliset syyt puoltavat ohjelmistolisenssien hallinnoinnin tärkeyttä yrityksissä ja organisaatioissa. Lisenssien puuttuminen tai lisenssiehtojen rikkominen voivat johtaa suuriin oikeudellisiin vastuisiin. Tästä syystä lisensoinnin ymmärtäminen ja ohjelmistonhallinnan toteuttaminen on yrityksissä ja muissa organisaatioissa tärkeää. Asianmukaisen ohjelmistonhallinnan avulla tiedetään, mitä lisenssejä omistetaan, ei makseta turhista tai käyttämättömistä lisensseistä, maksimoidaan kustannustehokkuus, varmistetaan ohjelmistojen tukipalvelujen saanti ja toimitaan lain puitteissa.

Laittomien ohjelmistojen käyttö johtaa ohjelmistopiratismiin, joka on yhä paheneva maailmanlaajuinen ongelma. Piratismi vahingoittaa paitsi ohjelmistoteollisuutta ja maailman taloutta, myös piraattituotteita käyttäviä yrityksiä ja

muita organisaatioita. Laittomissa ohjelmistoissa voi olla virheitä, ne voivat sisältää viruksia, tukipalvelut puuttuvat ja kokonaiskäyttökustannukset nousevat. Ohjelmistopiratismin tunnistaminen ja ymmärtäminen on tärkeää, vaikka sen hävittäminen kokonaan onkin hyvin vaikeaa. Yleisellä tiedottamisella ja koulutuksella voidaan kuitenkin vaikuttaa piratismin määrään. Tämä edellyttää ohjelmistovalmistajien, yritysten ja organisaatioiden sekä yksittäisten ihmisten yhteistyötä. Myös lakeja ja lakien valvontaa tarvitaan.

Mitä muuta voitaisiin tehdä, jotta ohjelmistopiratismi saataisiin vähenemään? Gopal ja Sanders [2000] ovat ehdottaneet globaalia hintaerottelua. He perustelevat ehdotustaan sillä, että esimerkiksi kehittyvissä maissa laillisen ohjelman ja piraattiversion hintaero on niin suuri, että se houkuttelee näitä pienituloisia maita ostamaan piraattiversioita. Heidän mukaansa ongelma perustuu siihen, että ohjelmistojen hinnat määräytyvät pitkälti USA:n mukaan, jossa ne ovat liian korkeat verrattuna siihen, mitä niistä monissa muissa maissa pystytään maksamaan. Gopal ja Sanders [2000] siis ehdottavat, että ohjelmistoja myytäisiin eri hintaan eri maissa. Koska heidän artikkelinsa on jo neljän vuoden takaa, olisi mielenkiintoista tietää, onko heidän ehdottamaansa hintaerottelua kokeiltu jossain. Tässä saattaisi myös piillä mahdollisuus jatkotutkimukselle.

Kaiken kaikkiaan tietokoneohjelmien oikeudellisen luonteen ja niitä suojaavan tekijänoikeuden ymmärtäminen on avainasemassa. Yrityksissä tulisi aina olla henkilö, joka tuntee ohjelmistolisenssisopimukset. Lisenssitodistuksiin ja ohjelmistojen oheismateriaaliin tulisi suhtautua samalla vakavuudella kuin esimerkiksi arvopapereihin. Jokaisen yrityksen ja organisaation täytyy olla tarvittaessa kykenevä osoittamaan ohjelmistojen käyttöoikeutensa. Yrityksiin kohdistuvat tarkastukset tulevat varmasti lisääntymään lähitulevaisuudessa, niin ohjelmistofirmojen kuin niiden etujärjestöjenkin toimesta. Palmerin [2004] mukaan yritykset voivat maksaa nyt tai he voivat maksaa myöhemmin - paljon enemmän - kiinni jäädessään.

Viiteluettelo

[Aamulehti, 2004] Ohjelmistoyritykset jahtaavat ahkerasti laittomia kopioita. *Aamulehti*, 14.8.2004, A12.

[Bigler, 2003] Mark Bigler, The high cost of software piracy. *Internal Auditor* (Jun. 2003), 64-65.

[BSA, a] Business Software Alliance. *http://www.bsa.org/finland/* [4.10.2004].

[BSA, b] Business Software Alliance, Miksi lisensointi on tärkeää? Saatavana *http://www.bsa.org/finland/antipiracy/Tools-Resources.cfm* [13.10.2004].

[BSA, c] Business Software Alliance, Guide to software management. Available as *http://www.bsa.org/finland/antipiracy/Tools-Resources.cfm* [13.10. 2004].

[BSA, 2001] Business Software Alliance, Ohjelmistonhallinnan opas. 2001. Saatavana *http://global.bsa.org/resources/2001-04-02.50.pdf* [11.10.2004].

[BSA, 2004] Business Software Alliance, First annual BSA and IDC global software piracy study. July 2004. Available as *http://www.bsa.org/ globalstudy/* [18.10.2004].

[Efecte, 2004] Efecte Oy. *http://www.efecte.fi/it-omaisuus/elinkaari.html* [13.10. 2004].

[Gopal and Sanders, 2000] Ram D. Gopal and Lawrence Sanders, Global software piracy: You can't get blood out of a turnip. *Communications of the ACM* **43**, 9 (Sep. 2000), 83-89.

[Hollmén, 1998] Sakari Hollmén, *Tietokoneohjelmien tekijänoikeudellisesta suojasta*. Turun yliopiston oikeustieteellisen tiedekunnan julkaisuja, Yksityisoikeudellinen julkaisusarja B:41, Turku, 1998.

[Koen and Im, 1997] Clifford M. Koen Jr. and Jin H. Im, Software piracy and its legal implications. *Information & Management* **31** (1997), 265-272.

[Microsoft, 2002] Microsoft Corporation, Käyttöoikeuksien hallintaopas. 2002. Saatavana *http://www.microsoft.com/finland/partner/msia/opas.pdf* [4.10.2004].

[Microsoft, 2004] Microsoft Software Asset Management. *http://www.microsoft. com/resources/sam/default.mspx* [4.10.2004].

[O'Brien, 2004] Frances O'Brien, Why IT asset management is important now. 10 June 2004, Gartner Research. Available as *http://www3.gartner.com/ resources/121300/121336/why_it_asset_ma.pdf* [13.10.2004]

[Palmer, 2004] Ian Palmer, Are you top of your software licensing? *Financial Executive* (Jun. 2004), 47-49.

[Peregrine, 2004a] Peregrine Systems, Justifying projects in software license compliance. White paper, January 2004. Available as *http://www. peregrine.com/asset-solutions/pdfs/SLC_WP.pdf* [11.10.2004].

[Peregrine, 2004b] Peregrine Systems, Managing your software assets. White paper, 2004. Available as *http://www.peregrine.com/fr/asset-solutions/ pdfs/SoftwareAssetMgmt_WP.pdf* [4.10.2004].

[Rebeiro, 1997] Michael Rebeiro, Software licensing - strategy, audit and negotiation. *Computer Audit Update*, (Nov. 1997), 9-18.

[SIIA, 2004] The Software & Information Industry Association. *http://www.siia. net/piracy/whatis.asp* [18.10.2004].

[Takki, 2002] Pekka Takki, *IT-sopimukset: käytännön käsikirja*. Gummerus, Jyväskylä, 2002.

[TekL] Tekijänoikeuslaki 8.7.1961/404.

[TTVK] Tekijänoikeuden tiedotus- ja valvontakeskus, Antipiracy.fi. *http://www.antipiracy.fi/tunnista/* [18.10.2004].

Scalable Vector Graphics Tiny profile

Vesa Huotari

Abstract.
Mobile devices have more processing capacity and better displays than before. This development together with higher bandwidth of mobile networks enables users to receive high-quality presentation and to use interactive services with their mobile device. Requirements for representing 2D vector graphics and creating user interfaces for these new devices and services have lead to development of vector graphics standard called Scalable Vector Graphics (SVG).

This work presents the SVG focusing on SVG Mobile Tiny profile, which is intended for the smallest category of mobile devices.

Keywords: Scalable Vector Graphics, SVG Tiny, W3C, Mobile Applications.
CR Classification: I.3.6.

1. Introduction

There is a wide variety of client devices from mobile phones with 96 x 65 pixels displays to 20" desktop displays with 1600*1200 pixels resolution and more. Different devices in terms of display resolution, power consumption, processor capabilities, human-device interface technologies and usage context bring challenges for displaying graphics effectively. Rendering traditional raster images to different devices is not straightforward operation. A large image can usually maintain good quality when scaled downwards, but this is not the case when a small image is scaled upwards. Raster images that are meant for portable devices with low-resolution display are not suitable for high-resolution displays. Scaling images to higher resolution makes small pixels to become large blocks. Good quality, high-resolution raster images may also be very large in terms of file size resulting long download times. On the contrary, vector graphics can be zoomed and scaled according to screen size without loss of quality. Also the file size does not increase when vector image is scaled upwards.

World Wide Web Consortium (W3C) leads the development of Web. Sir Tim Berners-Lee formed W3C in 1994 and now it has over 350 member organizations [W3C, 2004a]. Major part of W3C's work is producing Internet standards, which are referred to as 'recommendations'. For example, HTML (Hypertext Markup Language) and XML (eXtensible Markup Language) are W3C recommendations. XML is a meta-language that is used to describe other

languages like Scalable Vector Graphics (SVG). Scalable Vector Graphics is a recommendation for presenting 2-dimensional graphics in XML.

Figure 1 illustrates the SVG's relationship to other W3C specifications. All the specifications presented are not mature yet but under development. Lower blocks like URI, HTTP, XML, most of the interaction domain, and some other block s are ready [W3C, 2004a].

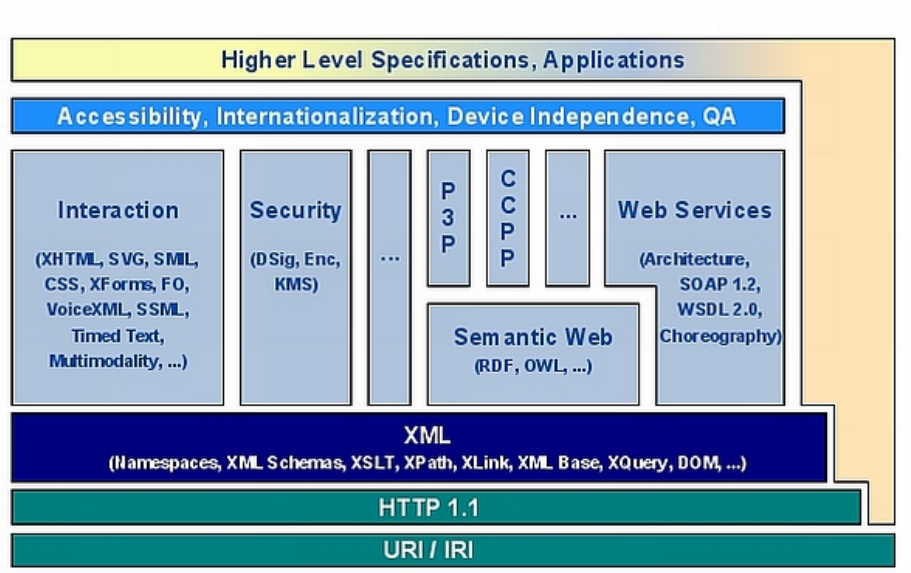


Figure 1. W3C specifications

Vector graphics in web environment has existed before SVG. Virtual Reality Modeling Language (VRML) was developed for describing virtual environments in Word Wide Web. VRML is standardized as ISO/IEC 14772. VRML is still used but it hasn't become popular. Building 3D graphics is more complex than making 2D graphics. 3D graphics is widely used by game industry in network games but VRML has no part in this since games are usually based on proprietary technology. The need for standardized 3D graphics exists and Web 3D consortium[1] continues the work done with VRML to develop X3D, which is XML-based language for making 3D graphics.

2. Overview of SVG characteristics and features

SVG is a fundamentally a mark-up language like xHTML or any other XML–based language and natively in human readable format and editable with text editors. XML–based files or documents are often called as structured documents. Complex presentations might be very difficult to edit and interpret

[1] http://www.web3d.org

in textual format; like web pages with embedded scripting and complex table structures. Vector graphics elements like curves and paths may require a graphical tool.

The image shown in Figure 2 is made with Sodipodi[2] tool. It contains few basic features of SVG: font, rectangle, star, colors and opacity.

Figure 3 contains a snapshot of actual content from previous SVG file. Docu-

Figure 2. Example SVG image

ment is in plain text format and like HTML, SVG uses different tags words bracketed by '<' and '>'.

Textual source code format may help diffusion and learning of SVG. Being

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.0//EN"
"http://www.w3.org/TR/2001/REC-SVG-20010904/DTD/svg10.dtd">
<!-- Created with Sodipodi ("http://www.sodipodi.com/") -->
<svg xmlns="http://www.w3.org/2000/svg"
xmlns:xlink="http://www.w3.org/1999/xlink" version="1.0" x="0" y="0" width="650"
height="200" id="svg602" xml:space="preserve">
<defs id="defs604" />
<text x="17.4646912" y="30.0937939" transform="scale(4.919548,4.919551)" style="font-
size:12;font-weight:normal;stroke-width:1;font-family:STLiti;" id="text608"
xml:space="preserve">
<tspan x="17.4646912" ….
```

Figure 3. A snapshot of a SVG file

somewhat similar to HTML, developers familiar with XML or HTML can learn the basics of SVG just examining the source code of a SVG file. Still the HTML developers examine the source code for learning new things and the same can be done with SVG.

---

[2] http://sourceforge.net/projects/sodipodi/

Scalability is one of the key features of vector graphics. Figure 4 illustrates the differences between raster image and SVG image. SVG images are scalable and can be zoomed without quality loss. [W3C, 2000]

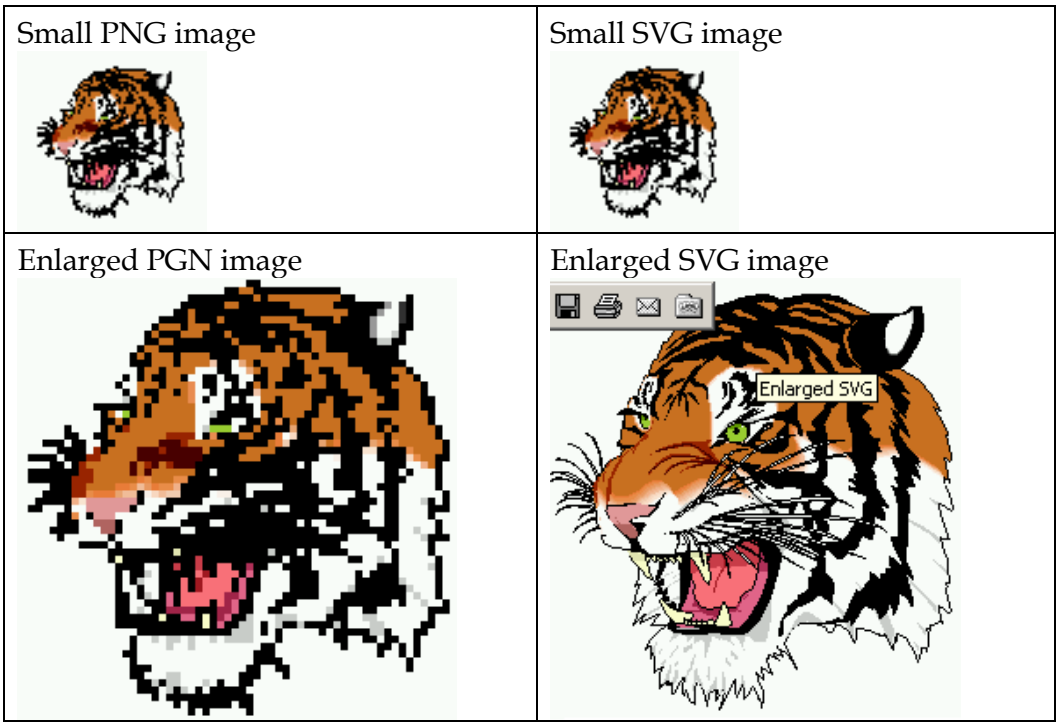


Figure 4. Example of vector graphics and raster graphics scalability

PNG images and other raster images however suffer from zooming. Other downside of raster images is that high-quality/large raster requires a lot of memory space. Compression decreases the file size but decreases the overall quality of image as well. Raster image formats are likely to remain for photographic representation, for which they are originally intended.

2.1. SVG profiles

SVG consists two parts, which are further divided into two main parts. First, full profile contains all the features of SVG. Secondly, there is a SVG Mobile part, which is divided into Basic profile and Tiny profile (see Figure 5). Basic profile is a subset of full profile and is meant for portable devices like PDA's and devices with more processing capabilities than for example regular mobile phones. Tiny profile, as a subset of Basic profile, is meant for the most resource limited devices like mobile phones.

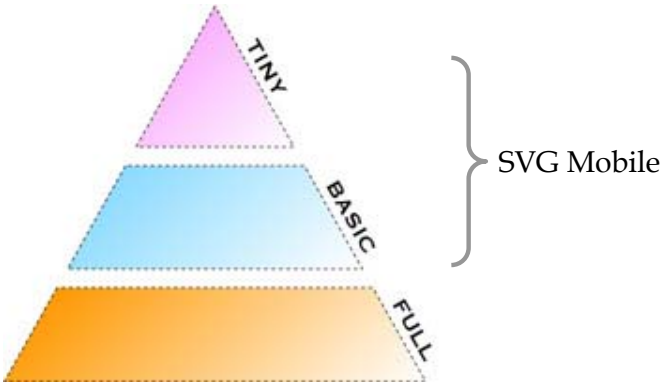


Figure 5. SVG profiles

SVG supports shapes, raster images and fonts. Shapes include rectangle, ellipses, circles, lines, polylines and polygons and range of raster effects like solid colors, gradient filling and, for example, opacity.

It's possible to embed non-scalable raster images like JPEG/JPG (Joint Photographic Experts Group), PNG (Portable Network Graphics) and GIF (Graphic Interchange Format) to a SVG document.

SVG supports also Document Object Model (DOM), which enables using different scripting-languages like JavaScript. This makes it possible to create interactive applications and dynamically manipulate the SVG presentation on the client side. Generating SVG on the server side is also possible using standard technologies like Perl Java or any other server side programming language.

SVG Basic profile contains most of the features of full SVG excluding some of the filtering features. Going trough all the differences of Tiny profile vs. Basic profile, would be a long story but to crystallize, SVG Tiny 1.1 includes following vector graphic and hypertext primitives:

- Rectangle, circle, ellipse, line, polygons, paths
- Animation features and some filtering effects
- Basic font features
- Basic hyperlinks.

Compared to full SVG, SVG Tiny profile contains only small pieces of SVG taking into account the limited capabilities of the target devices. A complete list of supported features of SVT and basic profiles is available in the mobile SVG specification[3]. As a subset of full SVG, SVG tiny is missing features like gradient, opacity and drop shadow [Mori and Ramanath, 2004], which could be useful in mobile phones as well. SVG is under development and SVG Mobile will change in the future. This will be discussed more in chapter 3.2.

---

[3] http://www.w3.org/TR/SVGMobile/#sec-eleind

2.2. Competing technologies

Macromedia Inc. has developed Flash vector graphics for Web and it has become the de facto technology for building visually attractive and interactive presentations. Flash has been around for several years and there are good tools, education and active ecosystem that support Flash developers. Flash has established itself but the problem with it is that Flash is a proprietary, closed technology owned by one company. Developing competing tools is difficult and the overall development and extendibility of Flash remains in the hand on Macromedia. SVG as a royalty-free and vendor-neutral [W3C, 2004b] technology has an advantage here.

Moreover, Flash does not fit into the W3C technology base. W3C technologies are based on human readable, structured documents like xHTML, XML, SVG and RDF (Resource Description Framework) and Flash itself remains a kind of 'black box'.

Macromedia has also developed Flash Lite version for small devices like mobile phones [Macromedia, 2004]. Flash Lite player also supports SVG Tiny, so Macromedia has activities on SVG side as well.

3. Standardisation and Industrial support for SVG Tiny

3.1. Standardisation

3GPP (3rd Generation Partnership Project) chose SVG Tiny as the mandatory vector graphics media format in MMS (Multimedia Messaging Service) [Quint, 2004]. SVG basic profile may be supported (optional) by MMS. Furthermore, 3GPP is considering whether SVG 1.2 Tiny should be adopted [3GPP, 2004].

Java community process (JCP) is developing API for supporting SVG tiny in Java2 Micro Edition (J2ME) [JSR 226, 2004]. This work is lead by Nokia with support from other mobile industry companies like Siemens and Motorola. Java applications and vector graphics in resource-limited devices are an interesting combination. Current mobile phones can be slow in with Java applications and vector graphics is processor intensive also.

The mobile industry is working on the standardizations side and on the device side trying to bring new devices to the market. MMS with SVG capabilities and SVG as promising description language for interfaces and new services work as a driver for the mobile industry. Vendors want to ensure that their ideas and visions are taken into account and they participate the standardization work.

3.2. Current state of SVG recommendations and the future plans of W3C

Original SVG 1.0 received recommendation status on 5th of September 2001. SVG 1.1 followed this on 14th of January 2003. The next version should be ready by January 2005 according to W3C roadmap[W3C, 2004f].

Document	WD1	Current WD	LC	Ends	CR	PR	REC
SVG 1.0	-	-	-	-	-	-	5 Sep 2001
SVG 1.1	-	-	-	-	-	11 Nov 2002	14 Jan 2003
SVG 1.2	11 Nov 2002	18 Mar 2004	[May 2003]	[Jun 2004]	[July 2004]	[Nov 2004]	[Jan 2005]
SVG Mobile Profiles	-	-	-	-	-	11 Nov 2002	14 Jan 2003
SVG Mobile1.2	9 Dec 2003	25 Mar 2004	-	-	-	-	-
SVG Print Requirements	18 Feb 2003	-	-	-	-	-	-
SVG Print	15 July 2003	-	-	-	-	-	-
Authoring Tool Guidelines	[Feb 2003]	-	-	-	-	-	-
Accessibility Techniques	[Mar 2003]	-	-	-	-	-	-

Table 1. W3C SVG roadmap

Legend: WD1 = first working draft; LC = last call for comments (i.e., last WD); Ends = deadline for LC comments; CR = Candidate Recommendation; PR = Proposed Recommendation; REC = W3C Recommendation[W3C, 2004f].

In Table 1 there are displayed also other SVG recommendations that are in working draft phase. SVG Print is considered as an alternative for postscript, which is page description language for printing documents.

Mobile SVG Profile: SVG Tiny, Version 1.2 is in working draft phase and under discussion. Some of the new features drafted are supported for streaming content, support for video and audio. Also animation, linking, interactivity features, text and painting features will get new features among other enhancement, according to Tiny 1.2 draft [W3C, 2004e]. Adding support for audio, video and streaming content makes it possible to have a much richer presentation than just vector graphics. It seems that SVG is moving towards complete multimedia environment. W3C has developed SMIL (Synchronized Multimedia Integration Language), which is used in MMS messaging. SMIL is meant for simple multimedia presentations. The relationship between SMIL and SVG standards may blur while SVG features are boosted.

3.3. Tools and Devices

Easy to use and effective tools are needed for an emerging technology to getwider use. There are native SVG editors available as well editors that export SVG [W3C, 2004c]. W3C maintains a list of available implementations[4].

[4] http://www.w3.org/Graphics/SVG/SVG-Implementations.htm8

Mobile SVG viewers are available for different operating systems including JavaME, Symbian, Pocket PC and PalmOS. A majority of these are commercial software [W3C, 2004c].

While device manufactures are working on the standardization side, they have also introduced client devices with SVG support. With SVG as mandatory part of MMS it is likely that more devices will support SVG. From the user 's point of view native support for SVG is more practical than downloading and installing a third party plugin. SVG Tiny is supported by few products from Siemens, Sony Ericsson and Nokia [W3C, 2004d].

4. Where to use SVG Tiny

4.1. Geographical information system applications

Zooming and rotating are important features for small device map applications [Kobayashi et al. 2003]. GIS specific description languages like OpenGIS[5] consortium's XML-based GML format (Geography Markup Language) can benefit from SVG because it's possible to transform GML to SVG [W3C, 2004b]. Wireless technologies make it possible to locate the client device. For example, it's possible to locate the mobile phone using GSM (Global System for Mobile Communications) or a laptop using WLAN (Wireless Local Area Network). The accuracy of location information depends on the technology used. The accuracy of local area network technologies like WLAN and Bluetooth is from 2-30 meters and the accuracy of GSM and GPRS (General Packet Radio Service) is from 100 meters to several kilometers [Rainio, 2002].

4.2. Messaging and Advertisement

Vector graphics offer opportunities to provide visually attractive messages and advertisements to user device. SMS has been adopted in marketing, but as text-only media with 160 characters/message it is very difficult to describe, for example, what some device or clothing looks like. However, SMS is cost-effective compared to MMS, which can carry SVG Tiny content.

Bluetooth and WLAN capable devices could be used on short distances for receiving messages and advertisements without operator fee. Vector graphics and embedded bitmap images provide new ways for marketing to advertise goods and services and to deliver these advertisements directly to user. Instead of receiving text-only messages, users could receive colorful, animated advertisements with interactive content.

---

[5] http://www.opengeospatial.org/

Users can decorate their mobile phones with screensavers and background images. It's also possible to send different type of messages to other users. SVG could enrich messaging and decorating. Sending and receiving animated messages instead of static images may attract users.

4.3. Other applications

Maintenance and engineering people could use SVG enabled devices to solve problems and get maps and engineering plans on demand. Games and cartoons can also be developed with SVG [W3C, 2001]. If SVG could be used more on device level to describe the actual user interface could ease design issues that relate to wide range of screen sizes of mobile devices. Users may want to have more control over the mobile device user interface. For example in some cases it would be preferable to have a possibility to rotate the interface 90 degrees to left or right. Series 60, which is the platform for mobile phones, build on top of Symbian has now SVGT fonts, icons and themes [Nokia, 2004]. Having a scalable user interface where controls and display primitives like fonts can be adjusted according to user preferences, could also solve some accessibility issues. For example, users having slightly impaired vision could zoom the SMS message, get large icons and zoom other interface elements large enough and suitable for them.

Scalable fonts and graphics in general could help users to read information with small devices in a more flexible way. For example, scalable graphics offer an accessible solution for user with different vision abilities: it's possible to get large icons and fonts

W3C has created a large family of recommendations that are based on XML. XML's advantage is its royalty freeness and vendor neutrality. It is also possible to transform one XML-document into another. For example, transform a single XML source could be transformed into different mediums like web

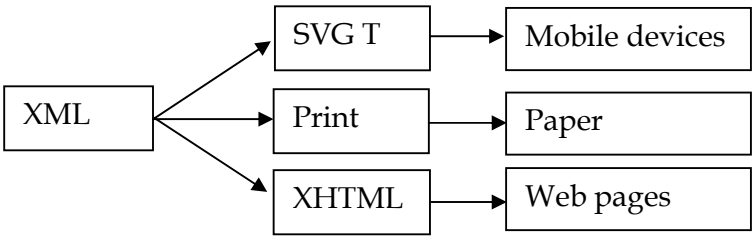


Figure 6. XML transformations

pages (xHTML), SVG (mobile phones) or printing it out to paper (SVG Print) as Figure 6 illustrates.

Having one base format for documents could bring benefits especially for publishing processes. W3C has developed technology for transforming purposes and there are proprietary and open source tools available for transforming XML-based documents and publishing purposes. For example, Apache foundations Cocoon project[6] has open source tools for transforming/publishing purposes.

5. Design issues with SVG content for small devices

SVGT makes it possible to have animations in small devices. One problem with SVG animation is that frame rate depends on the complexity of animation. Complex animation may be slow. Components like anti-aliasing and component inheritance are also time-consuming [Mori and Ramanath, 2004].

One solution to performance issues is hardware acceleration of SVG, which is developed by Khronos, a member-funded organization, whose member list includes companies from mobile phone industry, graphics and multimedia industry and other companies that have interest in graphics area. Having hardware acceleration does not reduce the variation between device capabilities but could enable more complex presentations in devices supporting hardware acceleration.

Some devices may one day support hardware acceleration and some may not. The problem with devices having different processor capabilities does not vanish. From author point of view it is time consuming to go trough all possible devices and test cases when designing SVG Tiny graphics.

6. Conclusions

The success of SVG in general will depend on how designers feel about the technology and on the availability of tools for creating content. Secondly, users must find the SVG content. Without applications or content that fulfills user needs, SVG is likely to fail. If there are user needs that SVG could solve, there is a market for content providers [Mori and Ramanath, 2004].

SVG can enrich the Web content and mobile content by enabling new services and enhancing existing ones. As a part of the W3C technology, SVG has synergy advantages compared to, say, Flash. Both SVG and Flash are not natively supported by Internet Explorer, Opera or Mozilla browsers, but demand a plugin. SVG has advantage since many standards developed by W3C are part of browsers all ready, SVG could be incorporated into browsers easily than Flash.

---

[6] http://cocoon.apache.org/

Mobile phones and messaging is one driver for SVG Tiny because it's mandatory technology for MMS. MMS messaging is still minor business for teleoperators but in general all new mobiles phone have MMS capabilities and it should teleoperators intend to support becoming more common by lowering the costs of sending MMS messages. Overall mobile industry is investing on SVG technology development heavily, which helps SVG becoming a mainstream technology.

SVG is becoming more a multimedia description language than only vector graphics description language, especially if the development of SVG 1.2 goes like it currently seems. This may be an advantage for SVG because developers and content providers can build better and media rich applications based on SVG, which they would be able to do without support for audio and video.

References

[3GPP, 2004] 3GPP TS 26.140: "Multimedia Messaging Service; Media formats and codecs". http://www.3gpp.org/ftp/Specs/archive/26_series/26.140/. 19.10.2001.

[JSR 226, 2004] JSR 226: Scalable 2D Vector Graphics API for J2METM http://jcp.org/en/jsr/detail?id=226 . 19.10.2004.

[Kobayashi et al. 2003] Arei Kobayashi, Satoru Takagi and Naomi Inoue. Extension of SVG for human navigation by cellular phone. In proceedings of the SIGGRAPH 2003 conference on Web graphics. 2003.

[Macromedia, 2004]. Macromedia Flash Lite. http://www.macromedia.com/software/devices/products/flashlite/. 19.10.2004.

[Mori and Ramanath, 2004] Koichi Mori and Sai P Ramanath. SVGT A study in implementation. In SVG Open 2004 Conference presentation slides.

[Nokia, 2004] Nokia. Introduction To The Series 60 Scalable UI v1.1. http://www.series60.com/file?id=247 . 2.12.2004.

[Quint, 2004a] Antoine Quint. Mobile SVG. 2004. http://www.xml.com/pub/a/2004/08/18/sacre.html. 19.10.2004.

[Quint, 2004b] Antoine Quint. Going Mobile With SVG: Standards. http://www.xml.com/pub/a/2004/06/16/mobilesvg.html. 19.10.2004

[Rainio, 2002] Antti Rainio. Mobiilipaikannus ja paikantavat päätelaitteet. Prosessori 11/2002, 84-87.

[W3C, 2004a] About the World Wide Web Consortium (W3C) http://w3.org/Consortium/. 10.10.2004.

[W3C, 2000]Accessibility Features of SVG http://www.w3.org/TR/SVG-access/. 11.10.2004.

[W3C, 2004b] About SVG 2d Graphics in XML. http://www.w3.org/Graphics/SVG/About. 20.10.2004.

[W3C, 2004c] SVG Implementations. http://www.w3.org/Graphics/SVG/SVG-Implementations.htm8. 21.10.2004.

[W3C, 2004d] W3C Scalable Vector Graphics (SVG) – History. http://www.w3.org/Graphics/SVG/History.htm8. 21.10.2004.

[W3C, 2004e] Mobile SVG Profile: SVG Tiny, Version 1.2 W3C Working Draft 13 August 2004. http://www.w3.org/TR/SVGMobile12/. 1.11.2004.

[W3C, 2004f] Scalable Vector Graphics Roadmap http://w3.org/Graphics/SVG/Roadmap. 10.10.2004.

[W3C, 2001] SVG Mobile Requirements. W3C Working Draft 3 August 2001 http://www.w3.org/TR/SVGMobileReqs. 8.11.2004.

[Web3D, 2004] Web 3D consortium. http://www.web3d.org. 11.11.2004.

Haittaohjelmien luonne ja niiltä suojautuminen

Lauri Hämäläinen

Tiivistelmä.
Tutkimuksessa esitetään, miten haittaohjelmia määritellään, ketkä niitä levittävät, miten ne havaitaan ja miten ne on mahdollista poistaa. Samalla pohditaan ongelman kasvun syitä. Lopuksi esitetään ratkaisukeinoja ongelman lieventämiseksi.

Avainsanat ja -sanonnat: Haittaohjelmat, virukset, madot.
CR-luokat: D.4.6, K.6.5

1. Johdanto

Haittaohjelmat ovat ohjelmia, jotka on suunniteltu aiheuttamaan haittaa tai tuottamaan vahinkoa. Viestintävirasto [2004] määrittelee haittaohjelmien tarkoittavan sellaisia ohjelmia, joiden tarkoituksena on aiheuttaa tietojärjestelmissä ei-toivottuja tapahtumia. Moni ohjelma voi myös olla haitallinen väärinkäytettynä. Kaikki haittaohjelmat eivät myöskään tuota suoraa vahinkoa, vaan kuluttavat järjestelmän resursseja tai häiritsevät käyttäjän ja muiden ohjelmien toimintaa.

Haittaohjelmat ovat yksi suurimmista uhkista tietokoneille, tietojärjestelmille ja tietoverkoille. Jonkin tietyn haittaohjelman räjähdysmäinen leviäminen voi pahimmillaan lamaannuttaa internetin suurimmaksi osaksi. Erilaisia haittaohjelmia on jo lukematon määrä ja niiden tiukka luokittelu on jo vaikeaa, sillä yhä useampi haittaohjelma on muuntautumiskykyinen ja osaa käyttää eri leviämismenetelmiä. Tietoverkkojen leviäminen ja tiedonsiirtonopeuksien kasvu ovat edesauttaneet myös haittaohjelmien leviämistä. Jatkuvasti tietoa tuottava ja kuluttava verkottunut yhteiskuntamme on houkutteleva kohde haittaohjelmille. Kerran läpipäässeen haittaohjelman on mahdollista levitä verkkojen sisällä aina heikoiten suojatun kohteen kautta.

Haittaohjelmia vastaan on kehitetty monipuolisia suojakeinoja. Heikoin linkki suojauksissa on kuitenkin yleensä tietokoneen käyttäjä itse. Tehokkainkaan suoja ei hyödytä mitään, jos sitä ei aktiivisesti käytä tai sen käyttöä ei ymmärretä oikein. Käyttäjä voi jo omalla toiminnallaan pienentää riskiä toimimalla turvallisesti ja olemalla tarkkaavainen tietokoneensa toimintaa kohtaan.

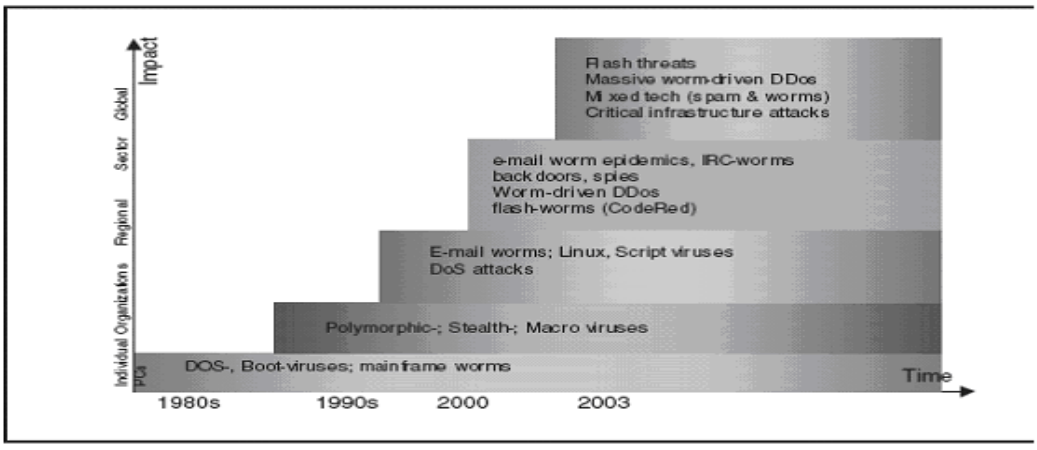
2. Haittaohjelmien kehitys

Tietojenkäsittelyn alkuaikoina tietokoneiden turvallisuus koostui pääasiassa fyysisistä seikoista. Tietokoneet olivat yleensä lukituissa tiloissa ja kulunvalvontaa tarkkailtiin. Berghel [2003] kertoo, kuinka tapaukset haittaohjelmista ovat kasvaneet vuoden 1988 kuudesta tapauksesta 82 094 tapaukseen vuonna 2002. Vuoden 2003 puoliväliin mennessä tapauksia oli jo 76404. Tästä on havaittavissa selvä noususuhdanne [Berghel, 2003].

Nykyään on melkein mahdotonta seurata uutisia kaikista uusista haittaohjelmista. Valtamedia esittää vain suurimpia uhkia ja alan oma tutkimus tuo esiin myös muita potentiaalisia uhkia. Treese [2004] pohtii internetin nykyistä tilannetta ja alati lisääntyviä haittoja. Haittaohjelmista on tullut jokapäiväinen ongelma. Yksi luotettavimmista lähteistä uusien uhkien tiedottamisessa ovat suojausohjelmien verkkosivut.

Levenhagen [2004] käsittelee haittaohjelmaongelma kiihtyvää kasvua vuonna 2003. Hän kuvaa, kuinka haittaohjelmat ylittivät uutiskynnyksen kyseisenä vuonna. Erityisesti massapostitusmadot aiheuttivat haittaa, syynä oli osaksi yritysten varautumattomuus roskapostiaaltoa vastaan.

Nikishin [2004]. käsittelee vuoden 2004 uhkia ja puolet esiintymistä on aiheuttanut eriasteisia epidemioita. Kirjoittaja arvelee tulevaisuuden olevan epävarmaa ja uudentyyppisten haittaohjelmien aiheuttavan lisää harmia. Kuvassa 1 on kuvattuna uhkia ja niiden vaikutuksia. Haittaohjelmien vaikutukset ovat levinneet globaalille tasolle.



Kuva 1. Uhat ja kohteet Nikishin [2004] mukaan.

Network Associaten virustietokirjasto (VIL) sisältää luokituksia madoista ja viime vuosien kehitystä kuvataan kuvassa 2. Tämän perustella voidaan myös todeta matojen määrän kasvu [Kienzle and Elder, 2003]. Sähköpostin ja

vertaisverkkojen käytön suuri suosio näkyy niitä varten suunniteltujen matojen määrässä.

Category of Worm	1998	1999	2000	2001	2002	2003*
Traditional	1	1	0	10	3	4
Windows File Sharing	0	7	14	20	28	80
E-mail	1	18	44	93	159	192
IRC	1	16	42	23	45	84
Peer-to-Peer	0	0	1	1	44	128

(2003 figures are projected from actual 1st quarter totals)*

Kuva 2. Viime vuosien VIL -merkinnät madoista kategorioittain [Kienzle and Elder, 2003].

Haittaohjelmia tutkitaan paljon, sekä kaupallisella sektorilla että tutkimuslaitoksissa. Käyttäjien, tutkijoiden ja ohjelmistotuottajien välinen yhteistyö hyödyttää kaikkia osapuolia ja näiden kolmen osapuolen tiedonvälitys on avuksi uusien haittaohjelmien tunnistamiseksi ja niiden hoitokeinojen kehittämiseksi. Mitä aikaisemmin tiedot uusista haittaohjelmista leviävät, sitä nopeammin niihin voidaan alkaa kehittää vastatoimenpiteitä ja suojauksia niitä vastaan.

Tulevaisuudessa haittaohjelmille on yhä useampia vaikutuskohteita. Yksi uusimmista uhkista ovat älypuhelimiin kohdistuvat haittaohjelmat. Älypuhelimet eli kämmentietokoneet ja matkapuhelimet ovat usein hyviä apuvälineitä työnteossa. Tästä johtuen ne saattavat sisältää arkaluonteista tietoa yrityksistä niiden avulla voidaan päästä laittomasti yritysten verkkoihin sisälle tai haittaohjelmat voivat pahimmillaan hajottaa laiteen. Haittaohjelmia siis kehitetään uusille järjestelmille sitä mukaan kun uusia teknologioita otetaan käyttöön [James, 2004]. Teknologisella kehityksellä on siis aina varjopuolensa, kun uudentyyppiset lieveilmiöt alkavat kehittyä.

3. Haittaohjelmien määrittely

3.1. Yleinen määrittely

Haittaohjelmat aiheuttavat siis haittoja, mutta ohjelmat ja niiden vaikutukset ne eivät aina ole suoraan näkyviä. Vaarallisimmat haittaohjelmat pysyvät piilossa ja käyttäjä voi olla täysin tietämätön niiden olemassaolosta. Näin ne voivat toimia salassa.

Nikishin [2004] esittää, että yleiskäsityksen saamiseksi moderneista viruksista, on määriteltävä modernin tietokoneiden ”eläinkunnan” viisi ominaisuutta: käyttäytyminen, esiintymisalue, penetraatiometodien erilaisuus, lukumäärän kasvu ja lisääntymisnopeus. Luokittelu osoittaa selvästi, että haittaohjelmien luokkien määrä on kasvanut vuodesta 1994, jolloin tunnettiin ainoastaan viruksia ja muutamia tuhoavia troijalaisia. Yksityiskohtiin asti luokittelu ei yllä, mutta se antaa selkeän yleiskuvauksen kohdealueensa objekteista ja erottelee ne kyllin selkeästi. Luokittelua vaikeuttavana piirteenä on uusien yhdistettyjen haittaohjelmien tulo. Tämä Dwanin [2004] esittämä luokka ’yhdistetyt haittaohjelmat’ on ajankohtainen lisäys haittaohjelmien luokitukseen ja selittää miten ominaisuuksien päällekkäisyys on mahdollista. Esitän tämän tyypin viimeisenä.

3.1.1. Käyttäytyminen

Ensimmäisenä luokkana ovat klassiset haittaohjelmat, jotka edustavat todellista uhkaa tietokonejärjestelmien toiminnalle. Se sisältää monia alaluokkia. Ensimmäisen luokkaan kuuluvat oikeat virukset, jotka tartuttavat suoritettavia tiedostoja ja kopioivat itseään. Virukset vaativat aktivoinnin toimiakseen. Ensimmäiseen luokkaan kuuluvat myös madot, jotka lisääntyvät leviämällä verkkojen kautta tietokoneesta toiseen. Matojen alaluokkia ovat esimerkiksi sähköpostimadot ja asiakassovellusmadot, Windowsin tiedostonjakomadot ja perinteiset madot [Kienzle and Elder, 2003]. Erona viruksiin on se, että madot eivät tarvitse isäntäohjelmaa ja osaavat levittää itse itseään. Troijalaiset kuuluvat myös ensimmäiseen luokkaan. Ne ovat usein naamioituneita hyödyllisiksi ohjelmiksi tai sellaisten osiksi. Ne voivat tehdä myös hyödyllisiä asioita, mutta samalla ne aiheuttavat kuitenkin haittoja. Alalajeina ovat tuhoavat troijalaiset, takaovitroijalaiset, troijalaiset varkaat ja vakoojat. Viimeisenä käsiteltyyn luokkaan kuuluvat hakkeroidut eli muunnellut verkkopaketit, joilla yritetään murtautua haluttuun kohdejärjestelmään.

Toisena luokkana ovat haittaohjelmat, jotka eivät aiheuta tietokoneelle suoranaista uhkaa. Ne saattavat kuitenkin kuluttaa tietokoneen resursseja, suorittaa epätoivottuja toimenpiteitä ja ärsyttää käyttäjää. Kyseisen luokka sisältää myös lukuisia alaluokkia. Automaattiset numeronvalitsinpalvelut (dialers) vaihtavat tietokoneen modeemiyhteyden ottamaan yhteyttä esimerkiksi maksullisiin pornosivustoihin. Käyttäjä ei välttämättä huomaa, että hän käyttää muualle ohjattua verkkoyhteyttä ja joutuu maksamaan erittäin korkea hintaa yhteydestä. Hieman vaarattomammat pilailuohjelmat voivat häiritä käyttäjää ja näyttää erilaisia viestejä, kuten ilmoituksia olemattomista viruksista. Usein ohjelmat eivät ole vaarallisia ja tekevät vain käytännön piloja.

Adware eli mainosohjelmat näyttävät ja lataavat mainoksia käyttäjälle. mainosten näyttäminen voi tapahtua käyttäjän luvalla tai ilman, usein ilmaisohjelmat sisältävät lisenssiehdoissa vaatimuksen mainosten vastaanottamiseen. Adware kuluttaa järjestelmän resursseja ja on usein päälle tunkevaa ja häiritsevää. Mainosohjelmien määrä on kasvussa ja yhä useampi haittaohjelma liittyy mainoksiin [Everett, 2004].

Spyware eli vakoiluohjelmat pyrkivät saamaan käyttäjän tietokoneelta tietoja ja samalla tutkivat tiedostoja. Tiedot lähetetään verkon välityksellä eteenpäin, jolloin niiden avulla voidaan saada tietoa kohteen ominaisuuksista esimerkiksi tietomurtoa varten, käyttäjien profiilia voidaan kartoittaa ja salasanoja anastaa. Spyware asennetaan yleensä käyttäjän tietämättä jonkin toisen ohjelman mukana tai ladataan käyttäjän koneelle salaa verkkosivulta. Spyware kuluttaa usein järjestelmän resursseja.

Tietojen kalastuksella (phishing) pyritään saamaan käyttäjältä henkilökohtaista tietoa, kuten salasanoja, pankkikortin tunnuksia tai sosiaaliturvatunnusta. Käyttäjää yritetään huijata tekaistuilla verkkosivuilla tai sähköpostilla luovuttamaan tietojaan. Tarkoituksena on saada käyttäjä luulemaan, että hän luovuttaa tietojaan luottamalleen tai viralliselle lähteelle. Hämäyksessä käytetään oikeiksi naamioituja sivuja ja virallisen näköistä sähköpostia. Toimivat menetelmät leviävät usein nopeasti, kun mahdollisimman moni yrittää hyötyä niistä [Furnell, 2004].

Kolmantena luokkana on riskware. Tämä riskiohjelmien luokka sisältää ohjelmia, jotka ovat mahdollisesti haitallisia käyttötarkoituksesta riippuen. Ohjelmia ovat esimerkiksi keskusteluihin käytettävä IRC (Intenet Relay Chat), etähallintaohjelmat, verkkoresurssien hallintatyökalut ja palvelut kuten ftp ja proxy. Käytännössä millä tahansa ohjelmalla voi suorittaa epätoivottuja toimenpiteitä, joten kaikkien eri ohjelmien listaaminen tähän osioon ei ole järkevää.

3.1.2. Esiintymisalue

Haittaohjelmien esiintymisalueet voidaan jakaa kahteen eri luokkaan. Ensimmäinen luokka sisältää käyttöjärjestelmät, kuten Windows ja Linux, toimisto-ohjelmat kuten MS Office ja erilaiset tietojen tallentamiseen käytettävät järjestelmät kuten SQL sisäänrakennettuine skriptikielineen.

Toinen luokka koostuu verkoista, jotka ovat tarpeeksi monimutkaisia virusten ja troijalaisten lisääntymisen mahdollistamiseksi. Tällaisia ovat sähköpostipostijärjestelmät ja internetselaimet, verkkopelit, verkottomat sovellukset, tukiohjelmat ja tarpeeksi monimutkaiset itsenäiset lisäohjelmat.

Esiintymisalueiden monimuotoisuus on hyvä esimerkki haittaohjelmien kehityksestä. Haittaohjelmat ovat levinneet todella laajalle alueelle ja kasvualustaksi sopii melkein mikä tahansa tietojärjestelmän osa-alue.

3.1.3. Penetraatiometodien erilaisuus

Penetraatiomenetelmät voidaan jakaa kolmeen luokkaan: virheisiin (haavoittuvuuksiin) ohjelmien suojausjärjestelmissä, dokumentoituihin vääristymiin ja inhimillisiin tekijöihin. Virheiden avulla voidaan ohjelmat saada pakotettua toimimaan halutulla tavalla tai niihin voidaan liittää haitallista ohjelmakoodia. Haavoittuvuuksia voidaan koettaa löytää kokeilemalla sokeasti eri hyökkäysmenetelmiä tai hyödyntämällä dokumentoituja haavoittuvuuksia ja ongelmia ohjelmissa.

Inhimillisillä tekijöillä tarkoitetaan käyttäjän toimintaa ja eri tapoja, joilla hänen toimintaansa yritetään vaikuttaa. Yksikään käyttäjä ei ole täysin samanlainen kuin muut ja sama käyttäjä voi toimia samankaltaisissa tilanteissa eri tavoin. Vaihteleva toiminta voi johtua vaikka stressistä tai kiireestä. Avaako käyttäjä tuntemattomia sähköpostiliitteitä tai hyväksyykö hän tuntemattomien ohjelmien pyynnöt päästä palomuurin läpi, riippuu monista eri vaikuttimista. Haittaohjelmien tekijät pyrkivät käyttämään hyväksi käyttäjien tietämättömyyttä tai herkkäuskoisuutta. Social engineering -termiä käytetään usein kun viitataan tämänkaltaiseen toimintaan.

3.1.4. Lukumäärän kasvu

Haittaohjelmien määrän kasvu selviää viime vuosien haittaohjelmien tilastoja tutkimalla. Erilaisia haittaohjelmia on jo monia, ja nykyään eri haittaohjelmien variaatioiden lisääntymisen myötä ei välttämättä kannata listata vain jo tunnettujen haittaohjelmien variaatioita. Perheyhteyksien löytämisellä voidaan saada uusia tuloksia paisuttamatta virustietokantojen kokoa liiaksi. Määrän kasvun syynä on haittaohjelmien kirjoittajien kasvanut lukumäärä ja käytettävissä on yhä enemmän työkaluja haittaohjelmien kirjoittamiseen. Koska haittaohjelmista on saatavilla yhä enemmän tietoa ja ohjeita niiden kirjoittamiseen, haitallisen informaation leviämistä ei voi enää estää tehokkaasti.

3.1.5. Leviämistavat

Haittaohjelmien lisääntymisnopeudet voidaan jaotella kolmeen luokkaan: perinteiset virukset, jotka leviävät tallennusvälineiden kautta, sähköposti- ja verkkomadot ja viimeisenä verkkomadot, jotka eivät vaadi ihmisen apua leviämisessään. Lisääntymisnopeus koskee ainoastaan viruksia ja matoja, sillä troijalaiset ja muut haittaohjelmat eivät pysty lisääntymään. Tämä ei kuitenkaan tarkoita sitä, ettei näitä mainittuja haittaohjelmia voisi levittää kerralla

suuria määriä. Levitys tapahtuu suuressa mittakaavassa esimerkiksi massasähköpostituksilla, jolloin saadaan kerralla lähetettyä suuria määriä ohjelmia.

3.1.6. Yhdistetyt uhat

Haittaohjelmien luokittelua vaikeuttaa uudentyyppisten haittaohjelmien kehittyminen. Yhdistetyt uhat (blended threats) eroavat perinteisistä viruksista ja madoista, sillä ne käyttävät monia eri hyökkäys- ja lisääntymismenetelmiä. Niillä on suurempi todennäköisyys onnistua aiheuttamaan vahinkoa, sillä ne yrittävät monia eri menetelmiä kerralla. Yksinkertainen suojaus ei toimi yhdistettyjä uhkia vastaan, sillä haittaohjelmat käyttävät eri lähestymistapoja kohdatessaan tietyn tyyppisiä suojauksia. Penetraatiota yritetään kunnes päästään läpi tai menetelmät loppuvat kesken. Tämä haittaohjelmatyyppi on erityisen vaarallinen, sillä läpi päästessään se voi vapauttaa lastinaan muita haittaohjelmia kohteeseensa. Tulevaisuudessa varmasti yhä useampi haittaohjelma on yhdistelmä aikaisimpien ohjelmien ominaisuuksia [Dwan, 2004].

4. Haittaohjelmien levittäjät ja levittämispaikat

4.1. Levittäjät

Haittaohjelmien levittäjinä on monen tyyppisiä henkilöitä ja tahoja. Karkeasti jaon voi tehdä tarkoituksellisesti levittäjiin ja tahattomiin levittäjiin ja yksilötasolta isompiin kokonaisuuksiin. Esitän luokittelun levittäjistä perustuen Gordonin [2000] ja Kemmererin [2003] artikkeleihin sekä omiin lisäyksiini.

4.1.1. Yksilöt

Haittaohjelmien levittäjien ja kirjoittajien motiivit voivat vaihdella todella paljon. Omien etujen tavoittelu, rahallinen hyöty, maine ja kunnia ovat osin motiiveina. Mahdollisuus toimia anonyymisti ja jäljittämisen vaikeus madaltavat helposti toiminnan aloituskynnystä. Esittelen seuraavaksi yksilötasolla haittaohjelmien levittäjiä ja kirjoittajia.

Skripti-ipanat [script kiddie] ovat hakkerikulttuurin alimmalla tasolla. He osaavat vain käyttää valmiita haittaohjelmia ohjeiden avulla. Heillä ei ole tietämystä tai taitoja itse ohjelmien rakenteista tai toiminnoista.

Parasiitit ovat yksilöitä, joilla ei ole varsinaisia hakkerointitaitoja eikä kysyä kirjoittaa replikoituvaa ohjelmakoodia. He hyötyvät muiden tuotoksista ja levittävät muiden työn tuloksia ja informaatiota.

Virusten kirjoittajia on vaikea tarkoin kategorisoida. Heidän ikänsä, koulutuksensa ja motiivinsa vaihtelevat laajalti. Itse viruksetkin vaihtelevat tekijöidensä mukaan. Tärkein erottava asia muista on heidän kykynsä kirjoittaa viruskoodia.

Hakkerit yrittävät todistaa muille pystyvänsä murtautumaan tiettyihin kohteisiin, kuten tiettyihin virastoihin tai yrityksiin. Hakkerit eivät aiheuta murtautumisen lisäksi muuta haittaa kohteelleen. Krakkerit murtautuvat myös tietokoneisiin ja aiheuttavat lisäksi tuhoja. Hakkerin ja krakkerin ero on ollut julkisuudessa varsin sekava ja termejä on käytetty eri merkityksissä.

Syyttömät levittävät tietämättään haittaohjelmia. Tietämättömyys tai välinpitämättömyys on syy siihen, että käyttäjät omalla toiminnallaan edistävät haittaohjelmien leviämistä. Käyttäjät voivat myös olla sitä mieltä, että heillä ei ole tietokoneillaan mitään suojeltavan arvoista, haittaohjelmat voivat kuitenkin vain käyttää kauttakulkupaikkoina ja apuvälineinä käyttäjien tietokoneita.

Sisäpiiriläiset ovat laillisia järjestelmien käyttäjiä, joilla saattaa olla henkilökohtaisia syitä olla vihamielisiä järjestelmää kohtaan. He voivat aiheuttaa suuria vahinkoja, koska he ovat jo järjestelmässä sisällä ja heillä on laillisia oikeuksia suorittaa toimintoja. Tyytymätön työntekijä voi olla esimerkki sisäpiiriläisestä.

4.1.2. Organisaatiotaso

Lailliset organisaatiot voivat toimia toisia organisaatioita vastaan haittaohjelmilla, käyttäen niitä yritysvakoiluun tai vahingoittamalla kilpailevien organisaatioiden toimintaa

Viime aikoina haittaohjelmien käyttö on yhdistetty organisoituun rikollisuuteen. Sähköiset verkot tarjoavat mahdollisuuden rikolliseen toimintaan ja suurten resurssien tukemana rikolliset voivat toteuttaa suuria vahinkoja. Erilaiset petokset ja tietomurrot ovat mahdollisia käyttötapauksia. Vaarallisimmat rikokset ovat, niitä joita ei havaita. Tällöin toimintaa voidaan jatkaa huomaamatta [Treese, 2004].

Valtiollisella tasolla haittaohjelmia voidaan käyttää toisten valtioiden vakoiluun ja vahingoittamiseen. Tällä tasolla käytössä on jo huomattavan suuret resurssit ja moraalisia ja eettisiä periaatteita voi olla jo vaikea arvioida.

4.2. Levittämispaikat

Virustenvaihtosähköpostijärjestelmät (Bulletin Board Systems, BBS) antavat mahdollisuuden ladata käyttäjille viruksia tai käyttäjä voi itse lisätä niitä järjestelmään. Järjestelmien luonne vaihtelee, osa on julkisia ja osa yksityisiä, mutta kaikille on yleistä virusten jakelu ja varastointi.

Virusten vaihdantaverkostot ovat kehittyneet Virus BBS -järjestelmistä. Aiempine järjestelmien käyttäjät ovat kehittäneet itselleen tiiviin verkoston, jossa tiedostojen vaihdanta on nopeampaa.

Virusten jakelusivustot ovat paikka tiedostojen vaihtoon. Sivustojen on vaihdettava paikkaa usein, jotta ne eivät paljastuisi. Jos sivustoilla annetaan

kaikille oikeus tarkastella sisältöä, myös viranomaiset pääsevät jäljittämään jakelijoita ja sivustojen ylläpitäjiä. Tällaiseen levitykseen on onneksi helppoa puuttua.

Virustenjakorobotit ja tiedostopalvelimet jakavat viruksia automaattisesti verkossa. Käyttäjät voivat toimia suhteellisen anonyymisti ladatessaan haittaohjelmia näiden automaattisten palveluiden avulla.

Virusohjekirjoissa annetaan ohjeita virusten kirjoittamiseen ja annetaan valmista lähdekoodia käytettäväksi omien virusten kirjoittamiseksi.

Kaupallisia viruksia on myös mahdollista ostaa. Tällaisia haittaohjelmia käytetään yleensä taloudellisen edun saamiseen. Virukset ovat usein tarkoin kohdennettuja tiettyihin toimintoihin, kuten yritystietojen hankintaan. Antivirusohjelmiston tekijät ja valtion osapuolet ovat myös ostaneet uutta viruskoodia, jotta ne voisivat testata suojautumiskykyänsä niitä vastaan.

5. Haittaohjelmien eettiset kysymykset

Tietokoneen käyttäjä ottaa tietoisen riskin käyttäessään tuntemattomia tai laittomia ohjelmia. Tällainen toiminta edesauttaa haittaohjelmien uhriksi joutumista. Ohjelmien tekijät kokoavat yleensä tärkeitä tietoja ohjelmasta tekstidokumenttiin, jonka käyttäjä on velvoitettu lukeman. Nämä lisenssiehdot ja sopimusehdot on syytä aina lukea läpi, sillä niissä saatetaan kertoa mahdollisista haittavaikutuksista, joihin käyttäjä suostuu hyväksyessään ehdot. Lainopillisesti käyttäjä näin hyväksyy seuraamukset. Osa ohjelmista ei kylläkään edes kysy käyttäjältä asennuslupaa eikä esitä minkäänlaisia käyttöehtoja. Ohjelman kehittäjän tarkoitusperät ovat tällaisten ohjelmien kohdalla varsin selvät; tarkoitus on suorittaa toimenpiteitä käyttäjältä salassa ja toivoa, että ei paljastu.

Haittaohjelmien kehittäminen vaatii taitoa. Taitotieto ohjelmakoodin kirjoittamiseen voi olla peräisin laillisista lähteistä ja vasta väärinkäytettynä vaarallista. Siksi on erittäin tärkeää, että tärkeiden ja salassa pidettävien järjestelmien kehittäjät ovat rehellisiä ja heidän moraalinsa on korkealla tasolla. Järjestelmissä ja ohjelmissa käytetty ohjelmakoodi ei saa levitä ulkopuolisille ja järjestelmien pitää olla vastustuskykyisiä mahdollisimman monella haittaohjelmatyypille. Neutraalin kolmannen osapuolen suorittama analyysi ohjelman turvallisuudesta ja ongelmakohdista olisi hyvä keino löytää mahdollisia tahallisia tai tahattomia haavoittuvuuksia ohjelmista. Käytännössä tällaisen toiminnan esteinä ovat liikesalaisuuksien säilyttäminen ja salassapitovelvollisuus.

Haittaohjelmien tutkijalla täytyy olla vastuu tutkimuksestaan ja tutkimustuloksistaan. Tutkijan täytyy käsitellä sekä haittaohjelmia, niiden vastaohjelmia että tavallisia ohjelmia. Löydettyjä heikkouksia tai ongelmakohtia ei saa

käyttää omiin tarkoituksiin, vaan tutkimustulosten avulla ongelmat pyritään korjaamaan. Haittaohjelmien tekijätkin voivat suorittaa omia tutkimuksiaan, mutta heidän motiivinsa on haittaohjelmien parantaminen. Tämä ei siis ole tieteellistä tutkimusta, vaan voidaan laskea ennemminkin omien etujen tavoittelemiseksi, sillä tutkimus ei millään tavalla täytä tieteelliselle tutkimukselle asetettuja vaatimuksia. Ohjelmistotuotannossa voidaan noudattaa monenlaisia sääntöjä, normeja ja standardeja sekä ylläpitää korkeaa ammattietiikkaa. Pakottaminen tietynlaiseen etiikkaan on varmasti vaikeaa ja viime kädessä jokainen itse vastaa omasta etiikastaan.

Haittaohjelmien levittäjät voivat hyödyntää eri maiden lainsäädäntöjen puutteita toiminnassaan. Toimiminen alueilla, joissa haittaohjelmien kirjoittamista ja levittämistä ei lasketa rikokseksi, on kyseisen maan lainsäädännön perusteella oikeutettua, mutta moraalisesti arveluttavaa. Haittaohjelmien tekijät yrittävät siis käyttää hyväksi kaikkia mahdollisia porsaanreikiä ohjelmiensa hyväksi.

6. Ohjelmistot suojautumista ja poistamista varten

Haittaohjelmia vastaan ei tarvitse olla voimaton, suojautumista ja poistamista varten kehitetty monia ilmaisia ja maksullisia ohjelmia. Luettelen muutamia tunnetuimpia ja samalla tehokkaimpia ohjelmia. Yksinään mikään näistä ei anna riittävää turvaa vaan jokaista kolmea tyyppiä on syytä käyttää, jotta suoja on tarpeeksi hyvä. Itse käytän monia eri suojaohjelmia, sillä yksikään ohjelma ei yksin löydä kaikkia haittaohjelmia. Kaikkia ohjelmia pitää myös päivittää, jotta ne olisivat ajan tasalla uusien uhkien suhteen.

Järjestelmän ja tiedostojen säännöllinen varmuuskopiointi on tarpeellinen tapauksia varten, joissa alkuperäistä järjestelmää tai tiedostoja ei voi enää palauttaa. Samalla taataan tietojen säilyminen suojattuna.

Käyttöjärjestelmän ja ohjelmien ajanmukaisilla päivityksillä voidaan korjata haavoittuvuuksia ja ongelmakohtia. Automaattisten päivitysten käyttö on suotavaa, sillä vain asennettu päivitys toimii. Käyttöjärjestelmän päivitysten asentaminen vaatii usein järjestelmänvalvojan oikeuksia, joten päivittämisen tulisi olla kyseisen henkilön vastuulla. On syytä myös varmistua, että päivitykset ovat aitoja.

Verkkoliikennettä on hyvä aika ajoin tarkkailla ja säilöä tapahtumia lokitiedostoihin. Poikkeava liikenne on yleensä merkki epätoivotuista vieraista. Jälkikäteen lokitiedostojen avulla voi selvittää, mitä on tapahtunut ja milloin. Tällainen valvonta voidaan suorittaa yleensä suojaohjelmien sisäänrakennetuilla seurantatyökaluilla.

Varsinaisia antivirusohjelmia on monia, esimerkiksi F-Securen tekemä Antivirus 2005 [F-secure, 2004]. Antivirusohjelmat tutkivat tietokoneen tiedostoja, sähköpostin liitteitä ja muita kohteita, joissa mahdolliset virukset ja madot saattavat piilotella. Eri ohjelmat eroavat siten, että ne etsivät haittaohjelmia eri algoritmeilla ja osaavat käsitellä ja poistaa löytyneitä haittaohjelmia eri tavoin.

Palomuuriohjelmisto voi olla esimerkiksi Zonelabsin Zonealarm Pro [Zonelabs, 2004]. Palomuuri estää useimpien haittaohjelmien latautumisen automaattisesti esimerkiksi verkkosivuja selattaessa ja samalla varoittaa epäilyttävästä verkkoliikenteestä. Palomuurin avulla käyttäjä voi täydellisesti hallita tietokoneensa tulevaa ja lähtevää verkkoliikennettä ja kontrolloida, miten eri ohjelmat käyttävät verkkoyhteyttä. Uusimmissa käyttöjärjestelmissä kuten Windows XP:ssä on sisäänrakennettuna palomuuri, mutta täysin erillinen palomuuri tarjoaa paljon enemmän säätömahdollisuuksia vaativalle käyttäjälle.

Muita haittaohjelmia kuten mainos- ja vakoiluohjelmistoja [Adware, Spyware] voi havaita esimerkiksi Lavasoftin Ad-Aware [Lavasoft, 2004] ja Spybot Search & Destroy [Spybot, 2004] -ohjelmilla. Nämä ohjelmat toimivat antivirusohjelmien tavoin: ne tutkivat tietokoneen kovalevyn sisältöä etsien pääosin mainosohjelmia ja vakoiluohjelmia. Nämä ohjelmat osaavat myös löytää matoja ja troijalaisia, mutta usein näiden poistossa tarvitaan apuna varsinaisia antivirusohjelmia.

Osa käyttöjärjestelmien automatisoiduista turvamenetelmistä on hyödyllisiä, mutta pahimmat haittaohjelmat osaavat hyödyntää myös niitä. Blaster-mato käytti hyväkseen Windowsin järjestelmänpalautustoimintoa, jossa muutokset palautuivat järjestelmän uudelleenkäynnistyksessä ja tällöin mato pystyi palauttaman itsensä. Käyttöjärjestelmän ja ohjelmien epätavalliseen toimintaan on siis kiinnitettävä huomiota, sillä se on usein merkki kutsumattomista vieraista.

Christodorescu ja Jhan [2004] esittävät uusia havaitsemistekniikoita antivirusohjelmille haittaohjelmien poistamiseksi. Artikkelissa testataan kolmea kaupallista antivirusohjelmaa ja niiden kykyä havaita obfuskaatiota. Obfuskaatiolla tarkoitetaan himmentämistä, jonka avulla haitallista ohjelmakoodia voidaan lisätä toiseen ohjelmaan vaikuttamatta isäntäohjelman toimintaan. Näin virus voi käyttää suojana tavallista ohjelmakoodia ja kuitenkin toimia haitallisesti suorittamalla omia toimintojaan. Testatut antivirusohjelmat eivät olleet kovinkaan vastustuskykyisiä tavallisille obfuskaatiovaihteluille ja tekijät jopa esittävät teorian puumerkinpoistoalgoritmista, joka hyödyntää antivirusohjelmien heikkouksia obfuskaatioiden havaitsemisessa.

Lisätietoja virusten ja matojen analyysista löytyy esimerkiksi Singhin ja Lakhotian [2002] bibliografiasta.

7. Ongelman hoitokeinot

Proaktiivisuus on tärkeä asenne tietoturvallisuudessa. Käyttäjä voi omalla toiminnallaan välttää suurimman osan haittaohjelmista. Käyttäjän vallassa on viimekädessä kaikki se mitä tietokoneella tehdään. Epäilyttäviä ja tuntemattomia verkkosivuja ja ohjelmia on syytä välttää ja omia tietoja kannattaa jakaa vain luotetuille tahoille. Myös epäilyttäviä sähköpostiviestejä ja niiden liitteitä ei ole tarpeen avata. Ohjelmistoja asennettaessa on tärkeää lukea lisenssi- ja sopimusehdot läpi, eikä vain hyväksyä niitä kritiikittömästi. Jotkin ohjelmat sisältävät käyttäjälle mahdollisesti haitallisia tai epämieluisia ominaisuuksia ja hyväksymällä ohjelmiston käyttöehdot käyttäjä hyväksyy samalla seuraukset. Jokin ohjelma voi esimerkiksi käyttää sisäänrakennettuja mainosohjelmia, joita ilman ohjelma ei toimi.

Haittaohjelmien poisto on luonnollisesti käyttäjän ensisijainen toiminto, kun hän haluaa niistä eroon. Haittaohjelmien poistossa voi olla kuitenkin poisto-ohjelmien kyvykkyydestä riippuen eriasteisia ongelmia. Haittaohjelma voi liittää itsensä järjestelmän kriittisiin osiin, jolloin poisto vaatii erikoistoimenpiteitä. Onnistunut poisto voi tehdä jopa järjestelmästä toimintakelvottoman, jos järjestelmän kriittisiä mutta saastuneita osia poistetaan samalla. Älykäs poisto-ohjelma osaa parantaa tiedostoja ja sisältää palautustoimintoja, joiden avulla voidaan palauttaa järjestelmään tehtyjä muutoksia. Tietokoneen puhdistusoperaatioiden jälkeen on aina syytä varmistaa lopuksi järjestelmä suorittamalla poisto-ohjelmilla täydelliset tarkistukset. Yksikin piiloon jäänyt haittaohjelma voi aiheuttaa sen, että jo poistetut haittaohjelmat latautuvat takaisin tietokoneelle.

Käyttäjien tietoturvakoulutuksella voidaan parantaa tietämystä yleisestä tietoturvallisuudesta ja oikeista toimintatavoista. Turvallisten käytäntöjen noudattamisella taataan se, että haittaohjelmille ei anneta mahdollisuutta toimia käyttäjän huolimattomuuden takia. Selkeät ohjeet menettelemisestä ongelmatapauksissa ja ohjeet käytettävien ohjelmien asetuksista edesauttavat käyttäjien kyvykkyyttä toimia oikein.

Palomuuri- ja antivirusohjelmistoja on tarjolla sekä ilmaisina että kaupallisina versioina, joten niiden hankkiminen ei varmastikaan ole taloudellinen kysymys. Vähimmäisvaatimuksena jokaisessa tietokoneessa tulisi nykyään olla nämä kaksi ohjelmistoa asennettuna ajanmukaisilla päivityksillä.

Vaihtoehtoisten ohjelmien käyttäminen valtaohjelmien sijaan voi auttaa haittaohjelmien torjumisessa yksinkertaisesti siksi, että haittaohjelmia ei ole harvinaisimmille ohjelmille paljoa kehitetty. Linuxin ja Macintoshin eri

käyttöjärjestelmäversiot ovat Windowsia turvallisempia. Aitojen ja laillisesti hankittujen ohjelmistojen käyttö on luotettavampaa kuin laittomasti hankittujen ja piraattiohjelmistojen, jälkimmäisten mukana saattaa tulla haittaohjelmia paljon todennäköisemmin.

Haittaohjelmiin kohdistuva lainsäädäntöä tulee kehittää niin, että tekijöitä voidaan asettaa vastuuseen entistä paremmin. Myös tekijöiden jäljittämisen tulisi olla helpompaa, tähän on apuna viranomaisten ja yritysten yhteistyö. Tapausten selvittämisessä toiminnan nopeus on erityisen tärkeää, sillä jäljet katoavat helposti verkossa. Haittaohjelmat voivat myös epäsuorasti käyttää hyväksi muita tietokoneita, jolloin hyökkäyksiä tekevä tietokone ei välttämättä ole itse syyllinen.

Ohjelmien laadun parantaminen on ensisijaisen tärkeää. Jo suunnitteluvaiheessa on syytä tarkastaa koodin tietoturvallisuus, riskit ja haavoittuvuudet. Toteutus- ja testausvaiheessa pitää myös huomioida turvallisuus, sillä jokainen tietoverkkojen ulottuvissa oleva tietokone ohjelmineen on uhattuna.

Saltzer ja Schroeder ovat kehittäneet 8 periaatetta turvallisen järjestelmän suunnitteluun[Kemmerer, 2003]. Nämä periaatteet ovat nykyäänkin yhä voimassa. Näiden periaatteiden toteuttamisella voidaan jo suunnitteluvaiheessa varautua ja välttyä useilta haittaohjelmilta ja niiden vaikutusyrityksiltä. Lyhyesti lueteltuna periaatteet ovat seuraavat: vähimmät oikeudet, mekanismien säästäväisyys, täydellinen välitys, avoin suunnittelu, käyttöoikeuksien erottelu, yleisten mekanismien välttäminen, tunnettu mekanismi, psykologinen hyväksyttävyys ja turvalliset perusasetukset. Kemmerer [2003] esittelee myös eri käyttöjärjestelmien haavoittuvuuksia tarkemmin.

Tevis ja Hamilton [2004] toteavat, että ohjelmistojen haavoittuvuuksia voidaan tehokkaasti löytää käyttämällä työkaluja, joilla voidaan etsiä lähdekoodista heikkouksia. Monien ohjelmointikielten imperatiivinen luonne saattaa olla perimmäinen syy niiden turvallisuuden heikkouksiin. He ehdottavat ratkaisuna siirtymistä imperatiivisesta ohjelmoinnista funktionaaliseen ohjelmointiin. Tällä tavalla voidaan siirtyä rakentamaan turvallisempia ohjelmia, jotka ovat paljon vaikeampia kohteita haittaohjelmille.

8. Yhteenveto

Kaikissa lähdeartikkeleissa on ilmaistu huoli tulevaisuuden kehityksestä ja uhkakuvien vakavuudesta. Haittaohjelmien määrän jatkuva kasvu antaa synkän kuvan tietoverkkojen tulevaisuudesta. On jopa arveltu internetin hajoamista, kun roskapostin ja haittaohjelmien määrä ylittää normaalin liikenteen. Voidaan myös kysyä, voidaanko ongelmaa korjata nykyisessä tilanteessa vai tarvitaanko tilalle uusi turvallisempi verkko.

Lisääntyviä ja voimistuvia haittaohjelmistoja vastaan toimittaessa on syytä siirtää painopiste reaktiivisuudesta proaktiivisuuteen. Pelkkä reagoiminen ja toimiminen vasta tartuntojen ja vahinkojen sattuessa on huono ratkaisu. Varovaisuus ja sopivan suojaohjelmistokokonaisuuden järjestelmällinen käyttäminen ovat jo suuri etu haittaohjelmistoja vastaan. Tämän lisäksi ohjelmistot on pidettävä ajan tasalla päivittämällä niitä ja käyttäjien on hyvä säännöllisesti seurata tiedotuksia uusista uhkista. Haittaohjelmat tulevat aina odottamatta ja ne määräävät suojausohjelmien kehityksen suunnan. Suojausohjelmat ovat aina kehityksessä hieman jäljessä.

Tietoyhteiskuntaan siirtymisen edellytyksiä ovat palveluiden turvallisuus ja luottamus tietojärjestelmiin. Vapauksien rajoittamista tulee myös tarkoin miettiä, turvallisuuden lisääminen pakotteiden ja määräysten käytöllä ei välttämättä ole paras keino. Tällä hetkellä internet on melko vapaa alue, jonka sisältöä on käytännössä mahdotonta valvoa, jolloin se on hyvä kasvualusta haittaohjelmille. Nykypäivänä tietoverkot ovat pääasiallisia haittaohjelmien leviämiskanavia.

Haittaohjelmien ei voida antaa välinpitämättömästi aiheuttaa tuhoa, sillä vaikutukset kertaantuvat yhä suurempina tietoverkkojen siirtyessä yhä suuremmaksi osaksi jokapäiväistä elämää. Erilaisten palveluiden, tietovirtojen ja toimintojen sähköistyminen asettaa ne myös haittaohjelmien kohteiksi. Haittaohjelmien tutkimuksella on siis tulevaisuudessa entistä suurempi kysyntä. Jos joku tietokoneenkäyttäjä kokee että ongelma ei kosketa häntä, mieli muuttuu varmasti kun hänen tietokoneensa saa ensimmäisen tartunnan. Haittaohjelmia riittää kyllä nykyisellä kasvuvauhdilla kaikille.

Viiteluettelo

[Berghel, 2003] Hal Berghel, Digital village: malware month. *Communications of the ACM* **46** , 12 (Dec. 2003) 15-19.

[Christodorescu and Jha, 2004] Mihai Christodorescu and Somesh Jha, Testing malware detectors. *ACM SIGSOFT Software Engineering Notes* **29**, 4 (July 2004) 34-44.

[Dwan, 2004] Berni Dwan,The malapropisms of malware. *Computer Fraud & Security* **3** (March 2004) 13-16.

[Everett, 2004] Catherine Everett, Upsurge in adware. *Computer Fraud & Security* **8** (August 2004) 2.

[F-secure, 2004] F-Secure Antivirus 2005 ohjelman kotisivut http://www.f-secure.fi/fin/ (5.12.2004)

[Furnell, 2004] Steven M. Furnell, Getting caught in the phishing net. *Network Security* **5** (May 2004) 14-18.

[Gordon, 2000] Sarah Gordon, Technologically enabled crime: Shifting paradigms for the Year 2000. *Computers & Security*, **14**, 5 (1995) 391-402.

[James, 2004] Gareth James, Malicious threats to Smartphones. *Network Security* **8,** (August 2004) 5-7.

[Kemmerer, 2003] Richard A. Kemmerer, Cybersecurity. In: *Proceedings of the 25th International Conference on Software Engineering*, May 2003, 705-715.

[Kienzle and Elder, 2003] Darrell M. Kienzle and Matthew C. Elder, Internet WORMS: past, present, and future: Recent worms: a survey and trends. In: Proceedings of the 2003 ACM Workshop on Rapid Malcode, October 2003, 1-10.

[Lavasoft, 2004] Lavasoft Ad-Aware homepages http://www.lavasoft.de/software/adaware/ (5.12.2004)

[Levenhagen, 2004] Roger Levenhagen, Trends, codes and virus attacks - 2003 year in review. *Network Security* **1**, (Jan. 2004) 13-15.

[Nikishin, 2004] Andrey Nikishin, Malicious software – past, present and future. *Information Security Technical Report* 9, 2 (April-June 2004) 6-18.

[Peslak, 2004]Alan R. Peslak, Improving Software Quality: An Ethics Based Approach. In: Proceedings of the 2004 SIGMIS Conference on Computer Personnel Research: Careers, Culture, and Ethics in a Networked Environment, April 2004, 144-149.

[Singh and Lakhotia, 2002] Prabhat K. Singh and Arun Lakhotia, Technical correspondence: Analysis and detection of computer viruses and worms: an annotated bibliography. *ACM SIGPLAN Notices* **37**, 2 (Feb. 2002) 29-35.

[Spybot, 2004] Spybot Seach & Destroy homepages http://www.safer-networking.org/ (5.12.2004)

[Tevis and Hamilton, 2004] Jay-Evan J. Tevis and John A. Hamilton, Methods for the prevention, detection and removal of software security vulnerabilities. In: Proceedings of the 42nd Annual Southeast Regional Conference, April 2004, 197-202.

[Treese, 2004] Win Treese, Putting it together: The state of security on the internet. *netWorker* **8**, 3 (Sept. 2004) 13-15.

[Viestintävirasto, 2004] Viestintäviraston kotisivut, haittaohjelmien määrittely http://www.ficora.fi/suomi/tietoturva/haittaohj.htm (5.12.2004)

[Zonelabs, 2004] Zonelabs Zone Alarm homepages http://www.zonelabs.com (5.12.2004)

Suomen kielen morfologia ja kaksitasomalli

Simo Härkönen

Tiivistelmä.
Morfologisessa jäsentämisessä taivutetusta sanasta analysoidaan vartalo ja taivutusmuoto. Vastakkaiseen suuntaan etenevää prosessia sanotaan tuottamiseksi. Morfologinen jäsentäminen on lukuisten kieliteknologisten sovellusten ensimmäinen askel.

Kaksitasomalli on parhaiten dokumentoitu tapa jäsentää suomen kielen sanoja. Kaksitasomallin mukainen jäsennin koostuu sanastokomponentista ja sääntökomponentista. Sanastokomponentti sisältää listan tunnetuista sanoista, sekä tiedon siitä, miten ne taipuvat.

Taivutetun sanan pintamuoto on se muoto, jonka näkee juoksevassa tekstissä. Syvämuoto on kielitieteellinen, abstrakti muoto. Syvämuoto on helppo tuottaa, ja siitä on helppo päätellä taivutus. Sääntökomponentti määrittelee pintamuodon ja syvämuodon välisen vastaavuuden. Säännöt ovat kielitieteellisiä uudelleenkirjoitussääntöjä. Kaksitasomalli käyttää samoja sääntöjä sekä jäsentämiseen että tuottamiseen.

Avainsanat ja -sanonnat: Kaksitasomalli, morfologinen jäsentäminen, äärellinen transduktori
CR-luokat: I.7

1. Johdanto

Kieliin, säännöllisiin lausekkeisiin ja transduktoreihin liittyvät matemaattiset merkinnät esitellään luvussa 2. Automaattiteorian uudelleenkirjoitussääntöjä en määrittele lainkaan, sillä kaksitasomallin säännöt ovat kielitieteellisiä.

Luvussa 3 kerron suomen kielen morfologiasta, jotta lukija ymmärtäisi, millaisia ongelmia morfologiseen jäsentämiseen liittyy. Tilan säästämiseksi rajoitun nominien taivutukseen. Samalla määrittelen nominien syvämuodon.

Luvussa 4 kerron kaksitasomallin säännöistä sekä niiden kääntämisestä äärellisiksi transduktoreiksi. Luvussa 5 sovellan sääntöjä tuottamiseen ja jäsentämiseen.

2. Merkintöjä

2.1. Aakkostot

Aakkosto Σ on äärellinen joukko *merkkejä*. Aakkostolle voidaan määritellä liitos-operaatio • niin, että (Σ,•) on Σ:n virittämä vapaa monoidi. Jos a, b ∈ Σ, niin niiden liitosta a • b merkitään ab. Yksittäisiä merkkejä a ∈ Σ merkitään pienillä kirjaimilla. Merkkijoukkoja A ⊆ Σ merkitään isoilla kirjaimilla. Komplementtia Σ \ A merkitään ~A.

Sana W muodostetaan liittämällä peräkkäin 0…n merkkiä aakkostosta Σ. Aakkoston Σ *kaikkien sanojen joukkoa* merkitään Σ*. *Kieli* L on joukko sanoja: L ⊆ Σ*.

Tyhjä sana λ muodostuu nollasta merkistä. Tyhjä kieli ϕ ei sisällä yhtään sanaa. Kieli L = { λ } sisältää yhden sanan. Sanajoukkoon Σ^{λ} kuuluvat kaikki nollan ja yhden mittaiset sanat.

Kielten L, L_1, $L_2 \subseteq \Sigma^*$ joukko-operaatioita merkitään seuraavasti:

- Unioni: $L_1 + L_2 = L_1 \cup L_2$
- Liitos: $L_1L_2 = \{ w_1w_2 \mid w_1 \in L_1, w_2 \in L_2 \}$
- Kleenen tähti: $L^* = \{ \lambda \cup w_1 \cup w_2w_3 \cup w_4w_5w_6 \cup \ldots \mid w_k \in L \}$
- Kleenen plus: $L^+ = LL^* = \{ w_1 \cup w_2w_3 \cup w_4w_5w_6 \cup \ldots \mid w_k \in L \}$.

2.2. Säännölliset lausekkeet ja säännölliset relaatiot

Säännölliset lausekkeet määritellään seuraavasti [Aho and Ullman 1972]:

(1) Tyhjä merkki ϕ on säännöllinen lauseke, jota vastaa kieli ϕ .

(2) Sana a on säännöllinen lauseke, kun a ∈ Σ^{λ}. Sitä vastaa kieli { a }.

(3) Jos p ja q ovat säännöllisiä lausekkeita, joita vastaavat kielet P ja Q, niin

(3a) unioni p + q on säännöllinen lauseke, jota vastaa kieli P + Q,

(3b) liitos pq on säännöllinen lauseke, jota vastaa kieli PQ, ja

(3c) Kleenen tähti p* on säännöllinen lauseke, jota vastaa kieli P*.

(4) Mikään muu ei ole säännöllinen lauseke.

Säännöllisten lausekkeet ovat sulkeutuvia seuraavien operaatioiden suhteen:

- Unioni, liitos ja Kleenen tähti (määritelmän mukaan).
- Komplementti: Jos R on säännöllinen lauseke, niin myös ~R on.
- Leikkaus: Jos R_1 ja R_2 ovat säännöllisiä lausekkeita, niin myös $R_1 \cap R_2$ on, sillä $R_1 \cap R_2 = \sim(\sim R_1 \cap \sim R_2)$.

Kaksitasomallissa säännöllisiä lausekkeita käytetään morfologisten reunaehtojen ilmaisemiseen, eli millaisessa ympäristössä morfologinen muutos voi tapahtua.

Säännöllisillä relaatioilla ilmaistaan vastaavuutta syvämuodon ja pintamuodon välillä. Säännöllisten relaatioiden rakennuspalikkoina ovat järjestetyt sanaparit [a, b] ∈ Σ* x Σ*. Tässä a on syvämuoto ja b on pintamuoto. Sanaparien liitos määritellään sanojen liitosoperaation avulla: [a, b] [c, d] = [ac, bd].

Merkintä [A, B], missä A, B ⊆ Σ tarkoittaa { [a, b] | a ∈ A, b ∈ B }. Yhtäsuuruusmerkki (=) sisältää koko aakkoston Σ. Esimerkiksi [=, b] sisältää ne parit, joissa mitä tahansa syvämuodon merkkiä vastaa pintamuodon b.

Kaplan ja Kay [1994] määrittelevät säännölliset relaatiot seuraavasti:

(1) Tyhjä joukko ϕ ja { [a, b] } ovat säännöllisiä relaatioita, kun a, b ∈ Σ^{λ}.

(2) Jos R_1 ja R_2 ovat säännöllisiä relaatioita, niin ovat myös

(2a) Liitos R_1R_2 = { [ac, bd] | [a, b] ∈ R1 ja [c, d] ∈ R2 },

(2b) Unioni $R_1 \cup R_2$, ja

(2c) Kleenen tähti $R^* = \{ \lambda \cup p \cup p_1p_2 \cup p_1p_2p_3 \cup \ldots \mid p_k \in R \}$.

Säännölliset relaatiot ovat suljettuja liitoksen, unionin ja Kleenen tähden suhteen (määritelmän mukaan).

Kun relaatiot kirjoitetaan omalle rivilleen, syvämuoto on ylhäällä ja pintamuoto alhaalla:

$$WQ = V\left(\frac{V \cup l}{=}\right)[CC^*].$$

Kaavan vasen puoli (WQ) on säännöllisen relaation nimi. Kaarisuluissa oleva termi sisältää parit { [x, y] | x ((V (l), y ((}. Kaarisulut tarkoittavat, että termi on valinnainen. Hakasulkuja käytetään ainoastaan ryhmittelyyn. ”Yksitasoinen” merkkijoukko C tarkoittaa parijoukkoa [C, C].

2.3. Äärelliset tilakoneet ja transduktorit

Järjestelmä M = (Q, Σ^{λ}, δ, q0, F) on äärellinen tilakone, kun

- Q on tilojen joukko
- Σ^{λ} on tyhjällä merkillä täydennetty syöteaakkosto
- q0 ∈ Q on alkutila
- F ⊆ Q on lopputilojen joukko
- δ ∈ Q x Σ^{λ} x Q on siirtymärelaatio.

Äärellinen tilakone aloittaa tilasta q_0 ja lukee nauhalta syötettä. Tilakone voi siirtyä tilasta q_1 tilaan q_2 lukemalla nauhalta merkin a, jos (q_1, a, q_2) ∈ δ. Lisäksi jos (q_1, λ, q_2) ∈ δ, on mahdollista tehdä tilasiirtymä lukematta nauhaa.

Tilakone *hyväksyy* syötteen w ∈ Σ*, jos on mahdollista siirtyä tilasta q_0 johonkin lopputilaan q ∈ F niin, että lopputilassa kaikki syötemerkit on luettu. Tilakonetta vastaava kieli L(M) on niiden syötteiden joukko, jotka tilakone hyväksyy.

Voidaan todistaa konstruktiivisesti, että mielivaltaiselle säännölliselle lausekkeelle voidaan rakentaa tilakone, joka hyväksyy vain kyseistä lauseketta vastaavat kielet.

Algoritmi: Tilakoneen rakentaminen säännöllisen lausekkeen perusteella
Syöte: Säännöllinen lauseke
Tulos: Tilakone, joka hyväksyy lauseketta vastaavan kielen.

1) Tyhjän kielen hyväksyy automaatti $M = (q_0, \Sigma^\lambda, \phi, q_0, \phi)$ eli automaatti, jossa ei ole yhtään siirtymää eikä lopputilaa.
2) Jos $a \in \Sigma^\lambda$, niin tilakone $M = (\{q_0, q_1\}, \Sigma^\lambda, \{(q_0, a, q_1)\}, q_0, q_1)$ hyväksyy kielen. Kyseisessä tilakoneessa on yksi siirtymä alkutilasta q_0 lopputilaan q_1.
3) Jos p ja q ovat säännöllisiä lausekkeita, ja M_p ja M_q kyseiset kielet hyväksyviä tilakoneita, niin
    a. Kielen p + q hyväksyvä tilakone M rakennetaan yhdistämällä tilakoneet M_p ja M_q. Koneen M tilajoukko on $Q_m = Q_p \cup Q_q \cup \{ q_0 \}$, missä q_0 on M:n alkutila. Alkutilasta on tyhjä siirtymä sekä M_p:n että M_q:n alkutiloihin.
    b. Kielen pq hyväksyvä tilakone M_{pq} rakennetaan niin, että M_p:n lopputiloista lisätään tyhjä siirtymä M_q:n alkutilaan. M_{pq}:n lopputilojen joukko on M_q:n lopputilojen joukko.
    c. Kielen p* hyväksyvä tilakone M rakennetaan M_p:stä kaksivaiheisesti. Ensin lisätään M_p:hen uusi lopputila. Vanhoista lopputiloista lisätään tyhjät siirtymät uuteen lopputilaan, ja uudesta lopputilasta tehdään koneen ainoa lopputila. Sitten uudesta lopputilasta lisätään tyhjä siirtymä alkutilaan.

Äärellisen transduktorin $T = (Q, \Sigma^\lambda, \delta, q_0, F)$ ero tilakoneeseen nähden on se, että siinä on kaksi nauhaa. Siirtymärelaatio asettaa kaksi ehtoa siirtymälle – yhden kummallekin nauhalle. Tällöin siirtymärelaatio $\delta \subseteq Q \times (\Sigma^\lambda, \Sigma^\lambda) \times Q$. Jos $(q_1, [a:b], q_2) \in \delta$, niin tilasta q_1 voidaan siirtyä tilaan q_2 lukemalla syöte a nauhalta 1 ja syöte b nauhalta 2. Siirtymäfunktio määritellään vastaavasti.

Transduktori hyväksyy sanaparin $[w_1, w_2]$, jos on mahdollista siirtyä tilasta q_0 johonkin lopputilaan $q \in F$ niin, että lopputilassa kaikki syötemerkit kummaltakin nauhalta on luettu. L(T) on niiden syöteparien joukko, jotka transduktori hyväksyy.

Transduktoria voidaan ajaa myös niin, että vain ensimmäisellä nauhalla on syötettä. Toinen nauha on tyhjä, ja sille kirjoitetaan ajon aikana. Jos transduktori siirtyy tilasta q_1 tilaan q_2 lukien merkin a, ja $(q_1, [a:b], q_2) \in \delta$, niin nauhalle 2 kirjoitetaan b.

On yleisesti tunnettua, että säännöllisen lausekkeen hyväksyvä tilakone voidaan *determinisoida* niin, että nykyinen tila ja nauhan syötemerkki määrittävät yksikäsitteisesti seuraavan tilan. Myös säännöllisen relaation hyväksyvä transduktori voidaan determinisoida, jos siinä ei ole muotoa [a:λ] tai [λ:a] olevia siirtymiä.

Valitettavasti determinisointi on mahdotonta silloin, kun transduktoria ajetaan yhdellä syötenauhalla. Suorituksen haarautuminen on väistämätöntä, ja se tulee ottaa huomioon toteutuksessa.

3. Suomen kielen nominien taivutus

Seuraava säännöllinen lauseke kuvaa abstraktilla tasolla suomen kielen nominien taivutusta:

nomini = vartalo [monikko] [sijamuoto] [omistusliite] [liitepartikkeli].

Käytännössä tilanne on monimutkaisempi. Monikon tunnus riippuu siitä, tuleeko sen jälkeen sijamuoto vai ei (talot – taloiksi). Taivutuspäätteissä tapahtuu kontekstista riippuvia muutoksia (talojen: monikon i muuttuu j:ksi). Lisäksi päätteissä on täysin säännöllistä vaihtelua, kuten vokaaliharmonia.

Kaikesta kontekstiriippuvaisuudesta huolimatta säännölliset menetelmät riittävät taivutuksen kuvailuun. Itse asiassa 90 % sanoista on mahdollista jäsentää pelkillä säännöllisillä lausekkeilla [Brodda and Karlsson 1981].

Kun kaksitasomallilla kuvataan taivutusta, ensimmäinen tehtävä on määritellä mahdollisimman yksinkertainen *syvämuoto*, joka kuvaa taivutusta korkealla tasolla. Sanan *pintamuoto* taas on se muoto, joka tekstissä esiintyy. Kaksitasomallin säännöt määrittelevät suhteen pintamuodon ja syvämuodon välillä.

3.1. Merkintätapoja vaihtelun kuvaamiseen

Arkkifoneemit ovat syvämuodon aakkosia, jotka voivat realisoitua pintamuodossa useammalla eri tavalla. Esimerkiksi essiivin päätettä -na, -nä voidaan kuvata syvämuodon päätteellä –nA, kun määritellään arkkifoneemi A → { a, ä }. Pintamuotoa tuotettaessa säännöt valitsevat, kumpi kirjain pintamuotoon tulee. Arkkifoneemeja voi esiintyä vain syvämuodossa.

Apumerkkejä käytetään ilmaisemaan, että jokin morfologinen muunnos on mahdollinen. Esimerkiksi kun monikon genetiiviin *talojen* liitetään omistusliite (*talojesi*), genetiivin tunnus n katoaa. Taulukko 1 antaa tästä esimerkin.

	talon + mon. genetiivi + yks.
Sijamuoto	2. persoonan omistusliite
Syvämuoto	talojen/si
Sääntö	n/ $\rightarrow \lambda$
Pintamuoto	talojesi

Taulukko 1. Apumerkin ja säännön avulla poistetaan omistusliitettä edeltävä n

Valintakriteereitä käytetään, kun päätteen valinta riippuu sanan aikaisemmista kirjaimista. Tällöin ei ole kysymys yksittäisen kirjaimen muuttamisesta, vaan suuremmista valinnoista. Esimerkiksi yksikön illatiivin pääte voi olla -:n (talo -> talo:n -> taloon) tai -h:n (maa -> maah:n -> maahan). On mahdollista ilmaista säännöllisillä relaatioilla, missä ympäristössä mikäkin pääte on mahdollinen. Tällöin päätteiden edelle laitetaan numerot (3+:n, 4+h:n) ja numerot yhdistetään valintakriteereihin.

3.2. Vartalo ja taivutus

Sanan *vartalo* on se merkkijono, johon taivutukset lisätään. Se ei ole välttämättä sanakirjan perusmuoto.

Koskenniemen väitöskirjassa [1983] suurin osa taivutukseen liittyvistä epäsäännöllisyyksistä kuvataan säännöillä. Kuitenkin joitakin vaihteluja on helpompi kuvata käyttämällä useampaa vartaloa.

Koskenniemi käyttää neljää vartaloa eri taivutusmuodoille. Monilla sanoilla kaikki vartalot ovat samanlaisia. Hyvin harvalla sanalla on neljä erilaista vartaloa.

- *Nominatiivivartaloa* käytetään yksikön nominatiivin muodostamiseen.
- *Yksikön taivutusvartaloa* käytetään niiden taivutusmuotojen muodostamiseen, joissa ei esiinny monikon i–tunnusta.
- *Monikon taivutusvartaloa* käytetään niiden taivutusmuotojen muodostamiseen, joissa monikon i–tunnus esiintyy.
- *Konsonanttivartaloa* käytetään yksikön partitiivin sekä monikon genetiivin muodostamiseen.

Esimerkiksi sanan saapas eräitä taivutusmuotoja ovat

saapas	yksikön nominatiivi,
saappaaseen	yksikön illatiivi,
saapasta	yksikön partitiivi,
saappaisiin	monikon illatiivi.

Tässä tapauksessa nominatiivivartalo on *saapas*, yksikön taivutusvartalo on *saappaa* (saappaaksi, saappaana). Myös monikon taivutusvartalo on *saappaa*, jolloin toinen a pudotetaan pois säännöillä. Konsonanttivartalo on *saapas*.

Erityisesti konsonanttivartalo on vain harvoilla sanoilla – muilla sanoilla vastaavat sijamuodot muodostetaan yksikkö- ja monikkotaivutusvartaloilla.

3.3. Jatkoluokat

Syvämuotoa muodostettaessa jokaisella vartalolla ja taivutuspäätteellä on luettelo *jatkoluokista*, jotka ovat sallittuja kyseisen liitteen jälkeen. Esimerkiksi konsonanttivartalon jälkeen tuleva jatkoluokka sisältää päätteet yksikön partitiiville sekä monikon genetiiville. Nimeän jatkoluokat PIENILLÄ ISOILLA KIRJAIMILLA. Jatkoluokka LOPPU merkitsee sanan loppua.

Tärkeimmät jatkoluokat – joiden sisältämät liitteet määritellään seuraavassa luvussa – ovat NOMINATIIVITAIVUTUS, YKSIKKÖTAIVUTUS, MONIKKOTAIVUTUS, OMISTUSLIITE sekä LIITEPARTIKKELI.

Monissa tilanteissa sanan perään voi vielä liittää päätteen, mutta se ei ole välttämätöntä (talo – talomme – talommekin). Tällaisia tilanteita varten otamme käyttöön merkinnät:

- OMISTUSLIITE? = { OMISTUSLIITE, LIITEPARTIKKELI, LOPPU }
- LIITEPARTIKKELI? = { LIITEPARTIKKELI, LOPPU }.

3.4. Nominien taivutuksen syvämuoto

Tässä luvussa kuvailen Koskennimen [1983] väitöskirjassa käytetyn nominien syvämuodon.

Taulukossa 2 on aluksi esitelty NOMINATIIVITAIVUTUS, joka voidaan liittää nominatiivivartaloon.

Tunnus	Jatkoluokka	Taivutusmuoto
	LIITEPARTIKKELI?	yks. nominatiivi

Taulukko 2. NOMINATIIVITAIVUTUS

Taulukossa 3 on esitelty jatkoluokka nimeltä YKSIKKÖTAIVUTUS. Siihen kuuluvat ne sijamuodot, joissa ei esiinny monikon i:tä.

Tunnus	Jatkoluokka	Taivutusmuoto
$+n	LIITEPARTIKKELI?	yks. genetiivi
$+n#	LOPPU	yks. genetiivi
+nA	OMISTUSLIITE?	yks. essiivi
!+A	OMISTUSLIITE?	yks. partitiivi
$+ksi	LIITEPARTIKKELI?	yks. translatiivi
$+kse	OMISTUSLIITE	yks.translatiivi
$+ssa	OMISTUSLIITE?	yks.inessiivi
$+stA	OMISTUSLIITE?	yks. elatiivi
3+:n	OMISTUSLIITE?	yks. illatiivi
4+h:n	OMISTUSLIITE?	yks. illatiivi
5+seen	OMISTUSLIITE?	yks. illatiivi
$+llA	OMISTUSLIITE?	yks. adessiivi
$+ltA	OMISTUSLIITE?	yks. ablatiivi
$+lle	OMISTUSLIITE?	yks. allatiivi
$+ttA	OMISTUSLIITE?	yks. abessiivi
$+t	LIITEPARTIKKELI?	mon. nominatiivi
%+en	OMISTUSLIITE?	mon. genetiivi

Taulukko 3. YKSIKKÖTAIVUTUS

Taulukossa 3 on käytetty seuraavia merkintöjä:

$ - Konsonantin astevaihtelun apumerkki, esimerkiksi katto -> katon.

+ - Apumerkki, joka merkitsee vartalon ja sijapäätteen väliä.

A - Arkkifoneemi, joka voi pintamuodossa realisoitua a:na tai ä:nä.

- Loppumerkki.

3, 4, 5, %, !- Numerot ovat valintakriteerien tunnuksia. Illatiivin päätteeseen vaikuttavat aikaisemmat merkit. Valintakriteereitä käsitellään seuraavassa luvussa.

Taulukossa 4 esitellään MONIKKOTAIVUTUS. Siihen kuuluvat ne sijapäätteet, joissa on monikon tunnus i. Merkit 2, 8, 9, & ovat valintakriteerien tunnuksia. Muuten merkinnät vastaavat edellisen taulukon merkintöjä.

Tunnus	Jatkoluokka	Taivutusmuoto
&+ien	OMISTUSLIITE?	mon. genetiivi
$6+iden	OMISTUSLIITE?	mon. genetiivi
$6+itten	OMISTUSLIITE?	mon. genetiivi
+inA	OMISTUSLIITE?	mon. essiivi
7+iA	OMISTUSLIITE?	mon. partitiivi
$6+itA	OMISTUSLIITE?	mon. partitiivi
$+iksi	LIITEPARTIKKELI?	mon. translatiivi
$+ikse	OMISTUSLIITE	mon. translatiivi
$+issA	OMISTUSLIITE?	mon. inessiivi
$+istA	OMISTUSLIITE?	mon. elatiivi
2+iin	OMISTUSLIITE?	mon. illatiivi
8+ihin	OMISTUSLIITE?	mon. illatiivi
9+isiin	OMISTUSLIITE?	mon. illatiivi
$+illA	OMISTUSLIITE?	mon. adessiivi
$+iltA	OMISTUSLIITE?	mon. ablatiivi
$+ille	OMISTUSLIITE?	mon. allatiivi
$+ittA	OMISTUSLIITE?	mon. abessiivi
+ine	OMISTUSLIITE?	komitatiivi
$+in	LIITEPARTIKKELI?	mon. instruktiivi

Taulukko 4. MONIKKOTAIVUTUS

Taulukko 5 määrittelee jatkoluokan OMISTUSLIITE. Kauttaviiva (/) on omistusliitteen tunnus. Kaksoispiste (:) merkitsee vokaalin kahdentumista. Syvämuodon arkkifoneemi A voi realisoitua pintamuodossa a:na tai ä:nä.

Tunnus	Jatkoluokka	Taivutusmuoto
/ni	LIITEPARTIKKELI?	Yks. 1. persoona
/si	LIITEPARTIKKELI?	Yks. 2. persoona
/nsA	LIITEPARTIKKELI?	Yks./mon. 3. persoona
/:n	LIITEPARTIKKELI?	Yks./mon. 3. persoona
/mme	LIITEPARTIKKELI?	mon. 1. persoona
/nne	LIITEPARTIKKELI?	mon. 2. persoona

Taulukko 5. OMISTUSLIITE

Taulukko 6 määrittelee viimeisen laajan jatkoluokan, LIITEPARTIKKELIN. Tässä alaviiva (_) on liitepartikkelin tunnus.

Tunnus	Jatkoluokka	Taivutusmuoto
_hAn	LOPPU	-han
_kA:n	LOPPU	-kaan
_kin	LOPPU	-kin
_kO	LOPPU	-ko
_pA	LOPPU	-pa
_kinkO	LOPPU	-kin ko
_kA:nkO	LOPPU	-kaan ko
_kOhAn	LOPPU	-ko han
_kOs	LOPPU	-ko s
_pAhAn	LOPPU	-pa han
_pAs	LOPPU	-pa s

Taulukko 6. LIITEPARTIKKELI

Taulukon 7 KONSONANTTITAIVUTUS tulee yleensä konsonanttivartalon jälkeen. Jotkut sanat sallivat konsonanttitaivutuksen (poljinta, lammasten), toiset eivät (hevonenta, kolmasten).

Tunnus	Jatkoluokka	Taivutusmuoto
1+tA	OMISTUSLIITE?	yks. partitiivi
2+ten	OMISTUSLIITE?	mon. genetiivi

Taulukko 7. KONSONANTTITAIVUTUS

Taulukon 8 TTATAIVUTUS on erityisluokka niille sanoille, joiden partitiivin tunnus on –ttA, kuten ”venettä”. Jos sana vaatii tta-taivutusta, vartalon jatkoluokkiin lisätään TTATAIVUTUS.

Tunnus	Jatkoluokka	Taivutusmuoto
+ttA	OMISTUSLIITE?	yks. partitiivi

Taulukko 8. TTATAIVUTUS

KOLMITAVUTAIVUTUS taulukossa 9 on erityisluokka niille kolmitavuisille sanoille, joissa astevaihtelu voi joko tapahtua tai jäädä tapahtumatta, kuten ”yksiköihin – yksikköihin”.

Tunnus	Jatkoluokka	Taivutusmuoto
$+inA	OMISTUSLIITE?	mon. essiivi
$8+ihin	OMISTUSLIITE?	mon. illatiivi
$+ine	OMISTUSLIITE?	komitatiivi

Taulukko 9. KOLMITAVUTAIVUTUS

3.5. Valintakriteerit

Valintakriteereitä käytetään valitsemaan sanalle sopiva pääte useasta mahdollisesta samaa taivutusmuotoa merkitsevästä päätteestä. Kullekin päätteelle kirjoitetaan säännöllinen relaatio. Pääte voidaan liittää sanaan, jos valintakriteerirelaatio hyväksyy sanan, ja jos päätteen luokka kuuluu sanan jatkoluokkiin.

Kaksitasomalli sisältää toisenkin tavan valita pääte useasta vaihtoehdosta. Sana voi jatkoluokilla ilmasta, mitä päätteitä sen jälkeen voi tulla. Yksikön partitiivin tta-pääte valitaan tällä menetelmällä.

Koskenniemi perustelee valintakriteerien olemassaolon sillä, että taivutusluokkien määrä kasvaisi valtavaksi. Hänen mukaansa Penttilän nykysuomen sanakirjassa tarvitaan yli 80 taivutusluokkaa.

Käytännössä valintakriteerit laskevat tavujen lukumäärää, sekä viimeisen tavun muotoa.

Koskenniemi käyttää valintakriteereitä määritellessä taulukon 10 kirjainluokkia.

Luokka	Sisältää	Selitys
V	aeiouyääAOÅ	pintamuodon vokaalit + arkkifoneemit
V:	aeiouyääAOÅ:	edellisten lisäksi kahdentumisen apumerkki
C	konsonantit	pintamuodon konsonatit + arkkikonsonantit
=	kaikki merkit	sekä kirjaimet, että apumerkit
Z	Tyhjä merkki	merkki, joka pintamuodossa katoaa

Taulukko 10. Tavujen laskennassa käytetyt merkkijoukot

Seuraavia säännöllisiä relaatioita käytetään apuna, kun määritellään tavuja laskevia säännöllisiä relaatioita:

$$Q = \left[\frac{=}{C} \cup \frac{C}{=} \cup \frac{.}{Z} \right]$$

$$W = \frac{V:}{V}$$

$$QW = Q * W$$

$$WQ = W\left(\frac{V:}{=} \right) QQ * .$$

Nämä vaatinevat selitystä. Ensinnäkin, nämä käyvät läpi *sekä syvämuodon että pintamuodon*. Siksi ne ovat säännöllisiä *relaatioita*, eivät säännöllisiä lausekkeita.

Luokaan Q kuuluvat sellaiset kirjainparit, joissa joko syvämuoto tai pintamuoto on konsonantti. On mahdollista, että syvämuodon vokaalia vastaa

pintamuodon konsonantti (talo&+ien - talo Z Z jen), joten kumpikin merkkipari on tarpeellinen. Lopuksi Q sisältää apumerkin ”.” realisoitumisen tyhjänä.

Luokka W sisältää vokaaliparit. Syvämuoto voi sisältää vokaalin kahdentumismerkin, mutta sen pitää pintamuodossa realisoitua aitona vokaalina.

Sen sijaan, että luettelisin kaikki valintakriteerit, esitän muutaman esimerkin. Koskenniemi [1983] ei implementoinut ohjelmaa, joka olisi kääntänyt säännölliset relaatiot transduktoreiksi. Sen sijaan hän rakensi transduktorit käsin ja syötti ne ohjelmaan, samalla yhdistäen monta eri kriteeriä samaan transduktorit.

Tavujen laskennassa käytetään taulukon 11 apurelaatioita. Näistä 2C tunnistaa kaksitavuiset, konsonanttiin päättyvät sanat. Relaatiot 2CV ja 3CV hyväksyvät vain vokaaliin loppuvia sanoja. 3CV hyväksyy myös yli 3 tavua pitkät sanat.

Nimi	Relaatio	Esimerkki	Syvämuoto
2C	QW (W) Q Q* [V : Z]	kieltä	kielE1+tA
2CV	2C W	taloa	talo!+a
3CV	2C (WQ)* V	asteikkoa	asteikko!+a

Taulukko 11. Tavujen laskennassa käytetyt apurelaatiot

Vastaavuusesimerkki 1 näyttää, miten 2C hyväksyy kaksitavuisen sanan vartalon. Valintakriteerien tarkistushan alkaa, kun vasemmalta oikealle etenevä jäsennys törmää valintakriteerin tunnukseen (1).

```
Q  W (W) Q  Q* [V:Z]
k  i  e  l     E   1  +  t  A
k  i  e  l     Z
```

Vastaavuusesimerkki 1. Relaation 2C evaluoiminen

YKSIKKÖTAIVUTUKSEN yksikön partitiivin pääte !+A on sallittu, jos sana päättyy yhteen vokaaliin ja se on vähintään kaksi tavua pitkä. Nyt tämä valintakriteeri voidaan ilmaista säännöllisellä relaatiolla

! = [2CV + 3CV].

4. Kaksitasomallin säännöt

Säännöt määrittelevät pintamuodon ja syvämuodon välisen suhteen. Säännöillä on kaksi esitystapaa – kielitieteelliset uudelleenkirjoitussäännöt käännetään suoritusta varten äärellisiksi transduktoreiksi. Säännöt voi ymmärtää suodattimina, jotka hylkäävät kieliopin vastaisen syötteen.

4.1. Uudelleenkirjoitussäännöt

Uudelleenkirjoitussäännöt koostuvat *vastaavuudesta*, *operaattorista*, sekä oikeasta ja vasemmasta *kontekstista*:

$$\frac{a}{B} \Leftrightarrow VK \_\_ OK .$$

Sääntö määrittelee, milloin syvämuodon a:ta vastaa pintamuodon b ∈ B. Vasen ja oikea konteksti ovat säännöllisiä relaatioita. Sääntö on relevantti, jos syvämuodossa on kirjain a, ja vasen konteksti ja oikea konteksti täyttyvät.

Kontekstirajoitussäännöt laittavat rajoituksia sille, millaisessa kontekstissa syvämuodon a:ta voi vastata pintamuodon b ∈ B. *Pintapakotussäännöt* asettavat rajoituksia sille, miten syvämuodon a voi realisoitua annetussa kontekstissa.

Vastaavuus on abstrakti pari [a:B], jossa a ∈ Σ ja B ⊆ Σ. Tässä a on yksittäinen syvämuodon merkki. Se voi olla arkkifoneemi.

Operaattoreita on kolme.

⇒: *Kontekstirajoitusoperaattori* kertoo, että syvämuodon a:ta voi vastata pintamuodon b ∈ B vain silloin, kun vasen ja oikea konteksti täyttyvät. Sääntö salli sen, että muutos ei tapahdu. Sääntö ei salli sitä, että muunnos tapahtuu joissain muussa kontekstissa.

⇐: *Pintapakotusoperaattori* kertoo, että vasemman ja oikean kontekstin ympäröimää syvämuodon a:ta vastaa aina pintamuodon b ∈ B. Sääntö ei ota kantaa siihen, voiko sama muunnos tapahtua myös muissa konteksteissa.

⇔: Näiden operaattorien yhdistelmä kertoo, että vasemman puolen muunnos tapahtuu aina annetussa kontekstissa ja vain annetussa kontekstissa.

/⇐: Myöhemmin lisättiin negatiivinen pintapakotusoperaattori. Tällöin syvämuodon a:ta ei saa vastata pintamuodon b ∈ B, jos vasen ja oikea konteksti täyttyvät. Negatiivisen pintapakotussäännön voi aina kirjoittaa tavallisena pintapakotussääntönä, jossa vastaavuus on [a : ~B].

4.2. Esimerkkejä säännöistä

Esimerkki 1. Omistusliitettä edeltävän n:n poistaminen

Seuraava sääntö kertoo, että omistusliitteen tunnusta (/) edeltävä n poistetaan:

$$\frac{n}{Z} \Leftarrow \_\_"/".$$

```
s a l i + n / m m e
s a l i Z Z Z m m e
```

Vastaavuusesimerkki 2. Omistusliitettä edeltävän n:n poistaminen.

Esimerkki 2. Konsonantin astevaihtelu

Astevaihtelussa kaksi peräkkäistä konsonanttia muuttuu yhdeksi, tai yksi konsonantti muuttuu pehmeäksi tai katoaa. Astevaihtelun kohteena olevia konsonantteja merkitään vartalossa arkkifoneemeilla K, P ja T, esimerkiksi tikKa, puKu, sanKo.

Astevaihtelun laukaisee apumerkki $, joka esiintyy mm. monissa sijapäätteissä. Arkkifoneemi ei voi realisoitua k:na, p:nä tai t:nä sen läsnäollessa:

$$\frac{K}{k},\frac{P}{p},\frac{T}{t} \Leftrightarrow \_\_ \sim \$.$$

Seuraavat säännöt määrittelevät P:n astevaihtelun:

$$\frac{P}{v} \Leftarrow V(l \cup r)\_\_$$

$$\frac{P}{m} \Leftarrow Vm\_\_$$

$$\frac{P}{Z} \Leftarrow \frac{=}{V}(l \cup r)p\_\_ .$$

Esimerkkeinä annettakoon huopa – huovan, kampa – kamman ja kurppa – kurpan.

Esimerkki 3. Vokaaliharmonia

Taulukko 12 sisältää vokaaliharmoniaa käsittelevissä säännöissä käytetyt merkkijoukot.

Luokka	Sisältää	Selitys
Vetu	Ääöy	Etuvokaalit
Vtaka	Aou	Takavokaalit
Neut	~(Vetu ∪ Vtaka)	vokaaliharmonian kannalta neutraalit kirjaimet

Taulukko 12. Vokaaliharmonian määrittelyssä käytetyt merkkijoukot

Seuraava sääntö kertoo, että A, O ja U muuttuvat takavokaaleiksi, jos edellinen ei-neutraali vokaali oli takavokaali:

$$\frac{A}{a},\frac{O}{o},\frac{U}{u} \Rightarrow \frac{=}{Vtaka} Neut * \_\_ .$$

Muussa tapauksessa arkkifoneemit muuttuvat etuvokaaleiksi:

$$\frac{A}{\ddot{a}},\frac{O}{\ddot{o}},\frac{U}{y} \Rightarrow \left[\# \cup \frac{=}{Vetu} \right] Neut * \_\_ .$$

4.3. Kaksitasosääntöjen kääntäminen transduktoreiksi

Koskenniemen [1983] väitöskirjassa säännöt käännettiin transduktoreiksi käsin. Myöhemmin Koskenniemi [1986] antaa hyvin yleisluontoisen kuvauksen sääntökääntäjästä.

Kayn ja Kaplanin [1994] kuvaus kääntäjästään on tarkempi. Heidän kehittämä kääntäjä on kuitenkin hyvin monimutkainen. Ongelmat ovat lähtöisin siitä, että transduktoria ajettaessa ei voida tietää, kuuluuko luettu merkki vasempaan kontekstiin, vastaavuuteen, vai oikeaan kontekstiin.

Hahmottelen tavan kääntää sääntöjä, joka ei ota tätä ongelmaa huomioon. Kiinnostuneet voivat katsoa Kayn ja Kaplanin artikkelia.

Kääntäminen alkaa sillä, että sääntö muutetaan säännölliseksi relaatioksi. Sen jälkeen säännöllinen relaatio muutetaan transduktoriksi. Käsittelen ensin jälkimmäisen vaiheen.

Säännöllisen relaation muuttaminen transduktoriksi

Säännöllisen relaation voi tulkita säännölliseksi lausekkeeksi, kun kirjainparit tulkitaan yksittäisiksi aakkosiksi. Jos säännöllisen relaation aakkosto on Σ^{λ}, niin sitä vastaavan säännöllisen ilmauksen aakkosto on $\{ [a:b] \mid [a:b] \in \Sigma^{\lambda} \times \Sigma^{\lambda} \}$.

Nyt voimme käyttää algoritmia, joka muuttaa säännölliset ilmaukset tilakoneiksi. Syntynyt tilakone voidaan tulkita transduktoriksi tekemällä aakkosto-muunnos toiseen suuntaan. Jos transduktori ei sisällä muotoa [a : λ] tai [λ : a] olevia siirtymiä, se voidaan myös determinisoida tilakoneanalogian mukaisesti.

Kontekstirajoitussääntöjen muuntaminen relaatioiksi

Oletetaan, että sääntö on muotoa

[a:B] => VK__OK.

Tällöin transduktorin tulee hyväksyä [a:b] oikeassa kontekstissa ja hylätä se muualla. Tämä voidaan ilmaista säännöllisellä relaatiolla

(~[a:B] + VK[a:B]OK)*.

Relaation ongelma on se, että jos pari [a:b] esiintyy sanassa useamman kerran, sama pari voi kuulua sekä ensimmäisen merkin oikeaan kontekstiin että toisen merkin vasempaan kontekstiin. Seuraava jako kahteen rinnakkaiseen filtteriin vähentää tätä ongelmaa:

(~[a:B] + VK[a:B])*

(~[a:B] + [a:B]OK)*.

Ongelma ei kuitenkaan poistu, sillä edelleen ensimmäinen [a:B] voi kuulua toisen [a:B]:n vasempaan kontekstiin.

Kay ja Kaplan ratkaisevat ongelman tekemällä kaksi kontekstitransduktoria, joista ensimmäinen merkitsee apumerkeillä kaikki kelvolliset vasemmat ja toinen oikeat kontekstit. Sen jälkeen vastaavuustransduktori tekee muunnoksensa merkeille, joita ympäröi oikean ja vasemman kontekstin apumerkki. Neljännessä vaiheessa oikean ja vasemman kontekstin apumerkit poistetaan.

Luvussa 5 annan esimerkin kontekstirajoitussäännöstä, jossa sama merkki kuuluu sekä vastaavuuteen että vasempaan kontekstiin.

4.3.1. Pintapakotussääntöjen muuttaminen relaatioiksi

Jos pintapakotussääntö on muotoa

[a:B] <= VK__OK,

niin syvämuodon a:ta pitää vastata pintamuodon b ∈ B annetussa kontekstissa. Sääntö ei hyväksy, että sanaparin osana on relaatio

VK[a:~B]OK.

Sääntö siis hylkää sanat, jotka alkavat relaation

[=:=]* + VK[a:~B]OK.

mukaisesti.

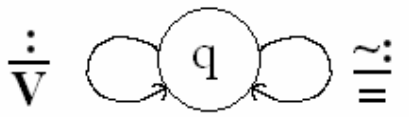
Olkoon M tästä säännöllisestä suhteesta rakennettu hyväksyvä deterministinen transduktori. Siitä voidaan tehdä hylkäävä poistamalla kaikki siirtymät $(q_1, [a:b], q_2) \in \delta$, joissa q_2 on lopputila, ja muuttamalla kaikki jäljelle jääneet tilat lopputiloiksi.

4.4. Esimerkkejä transduktoreiksi muunnetuista säännöistä

Esimerkkitransduktorit käsittelevät vokaalien kahdentumista. Tässä yhteydessä V = { a, e, i, o, u, y, ä, ö }. Ensimmäinen sääntö sanoo, että kaksoispiste (:) muuttuu pintamuodossa joko vokaaliksi tai häipyy:

$\frac{:}{V} \Leftarrow \_\_$ *(Sääntö 1).*

Sitä vastaava transduktori on kuvassa 1.



Kuva 1. Sääntöä 1 vastaava transduktori. Oikea siirtymä päästää suodattimesta läpi säännön kannalta epärelevantit merkit.

Tämän lisäksi tarvitaan sääntöjä, jotka kertovat, milloin kaksoispiste (:) muuttuu miksikin vokaaliksi. Sääntöjä on yksi kullekin vokaalille:

$\frac{:}{a} \Rightarrow \frac{=}{a}(h) \_\_$ *(Sääntö A),*

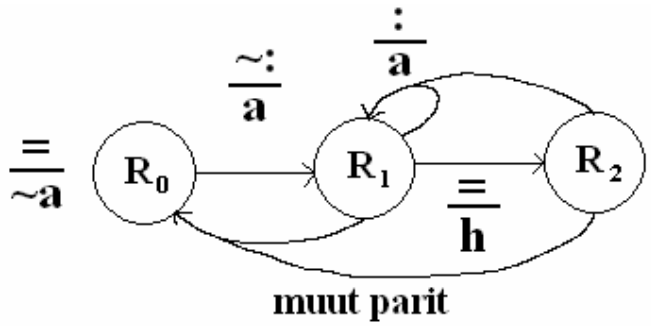
$\frac{:}{e} \Rightarrow \frac{=}{e}(h) \_\_$ *(Sääntö E),*

…

$\frac{:}{ö} \Rightarrow \frac{=}{ö}(h) \_\_ .$ *(Sääntö Ö).*

Säännöllisen relaation muodostaminen ei tälle säännölle onnistu, sillä esimerkiksi vastaavuudessa [ma:4+h:n, maahan] ensimmäinen kaksoispiste on toisen kaksoispisteen vasen konteksti.

Automaatin muodostamista tämä ei haittaa, vaan voidaan muodostaa kuvan 2 automaatti:



Kuva 2. Sääntöä A vastaava automaatti. Automaatti hyväksyy parin [":" : a] vain jos pintamuodon edeltävä konteksti on a(h).

5. Sääntöjen soveltaminen

5.1. Transduktorien ajaminen kahdella nauhalla

Säännöt voi ymmärtää rinnakkaisina suodattimina, jotka syötteen pitää läpäistä. Taulukko 13 antaa esimerkin siitä, miten säännöt hyväksyvät oikean syötteen. Taulukossa 14 säännöt hylkäävät väärän syötteen.

Syvämuoto Pintamuoto	m m	a a	: A
Sääntö 1 Sääntö a Sääntö o	q -> q r0 -> r0 r0 -> r0	q -> q r0 -> r1 r0 -> r0	q -> q r1 -> r0 r0 -> r0

Taulukko 13. Luvun 4.4 sääntöjen soveltaminen oikeaan syötteeseen

Kun sääntö 1 kohtaa kaksoispisteen, se asettaa rajoituksen pintamuodon vastaavalle merkille: $x \in V$. Sääntö A sallii kaksoispisteen muuntumisen a:ksi. Sääntö O ei salli kaksoispisteen muuntumista o:ksi, sillä $\frac{:}{o}$ ei kuulu kumpaankaan r_0:n tilasiirtymään. Tästä syystä a jää ainoaksi vaihtoehdoksi, kun kaikkien suodattimien asettamat reunaehdot otetaan huomioon.

Syvämuoto Pintamuoto	M M	a a	: o
Sääntö 1 Sääntö a	q -> q r0 -> r0	q -> q r0 -> r1	q -> q r1 -> r0

Sääntö o	r0 -> r0	r0 -> r0	pysähtyy

Taulukko 14. Luvun 4.4 sääntöjen soveltaminen väärään syötteeseen

Taulukossa 14 sääntöä O vastaava tilakone ei salli, että kaksoispistettä vastaa o, sillä [":" : o] ei kuulu kumpaankaan r_0:n tilasiirtymään. Näin kieliopin vastainen syöte hylätään.

Transduktoreja siis ajetaan merkki kerrallaan ja rinnakkain. Syötteen pitää läpäistä kaikki suodattimet.

5.2. Tyhjät merkit ja sallitut parit

Olemme nähneet, että syvämuoto ja sitä vastaava pintamuoto voivat olla eripituisia. Jotta tilakone voisi hyväksyä tällaisen kielen, siinä pitää olla muotoa [a : λ] ja [λ : a] olevia siirtymiä. Toisaalta, haluamme välttää tällaisia puolityhjiä siirtymiä kahdesta syystä.

Ensinnäkin, jos puolityhjiä siirtymiä ei ole, voimme determinisoida transduktorin tilakonetulkinnan avulla.

Toiseksi, säännöllisten relaatioiden joukko ei yleensä ole suljettu leikkauksen ja erotuksen suhteen. Kuitenkin samanpituisten säännöllisten relaatioiden joukko – joissa nauhan 1 sisältämät sanat ovat yhtä pitkiä kuin nauhan 2 sisältämät sanat – on suljettu leikkauksen ja erotuksen suhteen [Kay and Kaplan, 1994].

Jos transduktorien tuottama kieliperhe on suljettu leikkauksen suhteen, voimme yhdistää kaikki automaatit yhdeksi automaatiksi [Reape and Thompson, 1988].

Ratkaisuna on tulkita pintamuodon "tyhjät" merkit aakkoston merkeiksi. Tällä on kaksi seurausta. Ensinnäkin, tuottaessa pitää käsittelyyn lisätä kolmas vaihe, joka poistaa "tyhjät" Z–merkit. Toiseksi, jäsentäessä syötettä pitää muotoa [a : Z] olevia siirtymiä käsitellä ikään kuin ne olisivat tyhjiä siirtymiä. Tämä ei tee jäsentämisestä vaikeampaa, sillä ilman Z–merkkiä vastaavissa tilanteissa käytettäisiin puolityhjiä siirtymiä.

Suomen kielessä merkki b ei voi muuttua miksikään muuksi. Siksi säännöt eivät estä b:n muuttumista c:ksi (bertta – certta), koska säännöt eivät yksinkertaisesti sano mitään b:stä. Haluamme kuitenkin välttää tällaiset parit.

Ratkaisu on listata kaikki *sallitut parit*. Tämä tehdään kaksivaiheisesti. Ensin listataan oletusparit, jotka ovat tyypillisesti muotoa [n:n]. Sitä täydennetään lisäämällä joukkoon sääntöjen vastaavuuksissa esiintyvät parit.

5.3. Transduktorien ajaminen yhdellä nauhalla

Kerron ensin jäsentämisestä. Syvämuodon sanasto on tallennettu sanastokomponenttiin. Sanasto on tallennettu puunrakenteeseen niin, että puusta voi aina katsoa seuraavat mahdolliset kirjaimet, jotka ovat osa jotakin sanaa.

Kun sana loppuu, sanastokomponentti luettelee mahdolliset jatkoluokat. Myös jatkoluokat on järjestetty vastaaviksi puiksi.

Oletetaan, että sanasta on jäsennetty k-1 kirjainta, ja sääntöjä vastaavat transduktorit $M_1 \ldots M_n$ ovat tilassa $s_1 \ldots s_n$. Luetaan syötteestä merkki k.

Ensin katsotaan sanaston seuraavat merkit, eli minkälaiset seuraavat kirjaimet ovat osa jotakin sanaa. Jos merkki k-1 oli jonkin sanan viimeinen merkki, seuraava mahdollinen merkki otetaan sanan jatkoluokista.

Sitten katsotaan, mitkä merkit ovat sääntöjen sallimia. Vertailun lähtökohtana ovat tilojen $s_1 \ldots s_n$ ulospäin menevät siirtymät. Näiden perusteella voidaan muodostaa leikkaus niistä merkkipareista, joiden pintamuoto on k, ja joilla mikään tilakone ei pysähdy.

Jos kaikki transduktorit hyväksyvät myös muotoa [a: Z] olevan syötteen, pitää myös nämä parit lisätä mahdollisten parien joukkoon.

Lopullinen seuraavien mahdollisten parien joukko on siis

$P = \text{SanastoParit} \cap \text{SääntöParit} \cap \text{SallitutParit}$.

Tässä kohtaa suoritus voi haarautua, jos P sisältää useamman parin. Esimerkiksi kun pintamuodosta on luettu ”maah”, emme tiedä, vastaako sitä syvämuoto ”maahinen” vain ”ma:4+h:n”.

Jos P:ssä on p merkkiä, niin transduktorista otetaan p kopiota. Kutakin konfiguraatiota viedään eteenpäin yhdellä merkillä. Yllä mainittu prosessi toistetaan kaikille vaihtoehtoisille konfiguraatioille.

Transduktorit voidaan varastoida jonoon tai pinoon. Jos ne varastoidaan jonoon, niin kaikki transduktorit lukevat merkin k, ennen kuin mikään transduktori siirtyy merkkiin k+1, eli saadaan aikaan leveyssuuntainen haku. Jos haarautuvat transduktorit varastoidaan pinoon, saadaan syvyyssuuntainen haku.

Kun syöte loppuu, ja konfiguraation kaikki transduktorit ovat lopputilassa, olemme saaneet aikaan tuloksen. Koska useampi erilainen konfiguraatio voi päästä lopputilaan, on samasta sanasta mahdollista saada useampi oikea analyysi. Jos sana ei ole sanastossa, voi myös käydä niin, ettei mikään konfiguraatio tuota oikeaa analyysiä.

Sanan tuottaminen on olennaisesti samanlainen prosessi kuin jäsentäminen. Tuottaminen voidaan tehdä kahdella tapaa. Yksi mahdollisuus on antaa syötteeksi syvämuoto, johon kaikki taivutuspäätteet on jo liitetty. Toinen mahdollisuus on antaa pelkkä sana, sekä tieto siitä, miten sanaa tulee taivuttaa.

6. Yhteenveto

Kaksitasomalli mahdollistaa sanojen tehokkaan jäsentämisen ja tuottamisen. Kaksikymmentä vuotta sitten julkaistu väitöskirja sisälsi jo melkein kaikki olennaiset piirteet: kehitystä on tapahtunut pääasiassa sääntökääntäjissä ja toteutuksissa [Antworth, 2004].

Viiteluettelo

[Aho and Ullman, 1972] Alfred V. Aho and Jeffrey D. Ullman, *The Theory of Parsing, Translation and Compiling - Volume 1: Parsing*. Prentice Hall, 1972.

[Antworth, 1990] Evan L. Antworh, *PC-Kimmo, A Two-Level Processor for Morphological Analysis*. Summer Institute of Linguistics, 1990.

[Antworth, 2004] Evan L. Antworth, PC-Kimmo, a morphological parser. Available as http://www.sil.org/pckimmo/ (5.10.2004).

[Brodda and Karlsson, 1981] Ben Brodda and Fred Karlsson, *An Experiment with Automatic Morphological Analysis of Finnish*. University of Helsinki, 1981.

[Kaplan and Kay, 1994] Ronald M. Kaplan and Martin Kay, Regular models of phonological rule systems, In: *Computational Linguistics* ***20, 3*** (1994), 331 – 378.

[Koskenniemi, 1983] Kimmo Koskenniemi, Two-Level Morphology: A General Computational Model for Word-Form Recognition and Production. Helsingin yliopistopaino, 1983.

[Koskenniemi, 1985] Kimmo Koskenniemi, Compilation of automata from morphological two-level rules. In: *Papers of the Fifth Scandinavian Conference of Computational Linguistics* (1985), 143-149.

[Reape and Thompson, 1988] Mike Reape and Henry S. Thompson, Parallel intersection and serial composition of finite state transducers. *Proc. of the 12th International Conference on Computational Linguistics* **2** (1988), 535-539.

Kuvapuhelin konsultoinnin apuna

Heidi Laitinen

Tiivistelmä.
Kuvapuhelimen avulla välitetään liikkuvaa kuvaa ja ääntä toiseen kuvapuhelimeen. Se on tarkoitettu kahdenkeskiseen kommunikointiin tai pienryhmäneuvottelua varten. Kuvapuhelin on hyväksi esimerkiksi viittomakielisten henkilöiden kommunikoidessa. Tällöin fyysinen paikka ei ole esteenä ja viittovan henkilön suun sekä käsien liikkeet huomataan. Kuvapuhelinyhteyttä voidaan nykyään käyttää apuna myös kuurojen sosiaalisiin tuki- ja tulkkauspalveluihin, jolloin se on tärkeä tuki konsultoinnin apuna.

Avainsanat ja -sanonnat: Kuvapuhelin.
CR-luokat: H.1.2, H.5.2

1. Johdanto

Viimeaikoina on pyritty tuomaan tietotekniikka lähelle käyttäjiä, jotta he voisivat helposti käyttää tietoyhteiskunnan palveluja paremmin, helpommin ja tehokkaammin. On huomattu, että tekniikan avulla voidaan helposti auttaa rajoittuneina henkilöitä. Nyt tämä teknologia pyritään tuomaan käytettäväksi. Kuvapuhelin tarjoaakin kuulovammaisille ja maahanmuuttajille erinomaisen välineen kommunikointiin ja tietoyhteiskunnan palveluiden hyödyntämiseen. Kuvapuhelin eroaa tavallisesta tietokoneesta videokameran, mikrofonin sekä kuvapuhelinohjelman avulla. Fyysinen kontakti tapahtuu tällöin Internetin välityksellä.

Esimerkiksi viittomakielessä käsien ja sormien paikat, kasvojen sekä vartalon liikkeet, eleet ja ilmeet ovat tärkeitä. Kuurolle kaikki informaatio välittyy kuvan kautta, joten kuvaresoluution merkitys on suurempi kuin kuulevien kommunikoidessa. Tässä tapauksessa kuvapuhelin antaa kommunikoidessa suuren hyödyn.

Tampereella Ehyt-hanke tutkii kuvapuhelimenkäyttöä päivähoitoalalla. Siellä on kasvanut erilaiset ihmisryhmät, jotka tarvitsevat muun muassa tulkkauspalveluita. Tarvittavia tulkkeja on joskus erittäin vaikea saada, joten kuvapuhelin antaa hyvän mahdollisuuden saada yhteys tarvittavaan henkilöön ilman aika- ja paikkasitoumuksia. Ehyt-hankkeen käyttäjäryhmä koostuu kuulovammaisista, maahanmuuttajista sekä varhaiskasvatuksen alueelta olevista henkilöistä.

Seuraavissa luvuissa esitän kuvapuhelimen toimintaa, Ehyt-hanketta sekä sen käyttäjäryhmää ja heidän kuvapuhelinprojektiansa, tulkkipalvelua sekä kuvapuhelimen käytön yleistymistä Suomessa.

2. Ehyt-hanke

Ehyt-hankkeen tavoitteena on kehittää päivähoidon alueella yhteistyömahdollisuuksia kuvapuhelinta käyttäen päiväkotien ja tulkkikeskusten välisessä viestinnässä. Erityisenä käyttäjäryhmänä ovat viittomakieliset ja kuntouttavan varhaiskasvatuksen piirissä olevat perheet sekä maahanmuuttajaperheet. Kuvapuhelin toimii Tampereen kaupungin sisäverkon alueella.

Hankkeen toisena tavoitteena on myös luoda uusi palvelu Kuurosokeiden Toimintakeskukselle. Kuvapuhelin voisi tällöin olla asumispalvelun tukena.

Hankkeen kolmantena tavoitteena on selvittää miten kuuro voi henkilökohtaisella laitteellaan käyttää tulkkipalvelua julkisen WLAN-verkon kautta sekä arvioida henkilökohtaisesti käytettävän etätulkkipalvelun toimivuutta. Hankkeen toteutusaika on vuoden 2004 heinäkuusta seuraavan vuoden kesäkuuhun. [Media Tampere, 2004]

Ehyt-hankkeenkoordinaattorina toimii Oy Media Tampere Ltd. Yhteistyökumppaneina ovat Tampereen kaupunki/Sote/Päivähoito, Suomen Kuurosokeat ry/Kuurosokeiden Toimintakeskus, Tampereen Kuurojen yhdistys/ tulkkikeskus sekä Tampereen Tietotekniikkakeskus [Media Tampere, 2004].

Hanketta rahoittaa Tampereen kaupunki sekä Pirkanmaan liiton kautta Euroopan Yhteisö, Euroopan aluekehitysrahasto. Media Tampere vastaa hankkeen koordinoinnista ja kuvapuhelinten hankinnasta, käytön opastuksesta, kuvapuhelimen käyttöön liittyvästä teknillisestä tuesta sekä tiedottamisesta ja raportoinnista. [Media Tampere, 2004]

3. Kuvapuhelintekniikat

Kuvapuhelin voidaan asentaa hyvään, tehokkaaseen mikroon, joka varustetaan kameralla, mikrofonilla, kaiuttimilla, kuvapuhelinohjelmalla, video- ja ISDN-kortilla kuvankäsittelyä varten sekä äänikortti ääntä varten. Lisäksi tarvitaan puhelinlinja, joka muutetaan digitaaliseksi ja varustetaan ISDN-verkkopäätteellä [Hakulinen ja Savela, 2000]. Tieto kulkee kuvapuhelimesta toiseen Internetin kautta. Valittaessa kuvapuhelinta on hyvä kiinnittää huomiota tekniikkaan, kuvanlaatuun, nopeuteen, tehokkuuteen sekä äänen laatuun.

Kuva 1. Kuvapuhelin viittomakielisen käytössä.

Käyttäjän näytössä näkyy kuvapuhelinohjelma, jossa näkyy sekä keskustelukumppanin että käyttäjän oma kuva. Omasta kuvastaan hän voi tarkastaa, että kameran asento on oikea, jotta vastapuoli näkee hänet hyvin.

Keskusyksikköön liitetään kamera, mikrofoni ja Internet-yhteys. Näistä laitteista keskusyksikkö muuttaa tietoa lähetettäessä datan siirrettävään muotoon, ISDN:ään tai dataverkkoon sopivaksi. Dataa vastaanotettaessa se purkaa verkosta saadun datan käyttäjälle kuvaksi ja ääneksi. Tätä tapahtumaa kutsutaan koodekiksi. [Köyste et al., 2003]

Lähiverkossa olevat työasemat, jotka käyttävät IP-teknologiaa on mahdollista kytkeä ISDN-järjestelmiin siltalaitteen kautta. Käytettäväksi kaistanleveydeksi voidaan määritellä myös IP:tä käytettäessä esimerkiksi 384kb tai 768kb. Nämä kaistanleveydet riittävät kolmen ISDN-linjan videoneuvottelulle. [Köyste et al., 2003]

Lisäksi tarvitaan seuraavia tekniikoita. Videokamera voi olla pieni, zoomaava tai kääntyvä erikoiskamera. Mikrofonin olisi hyvä olla liikuteltava, sekä niitä voi tarvittaessa olla enemmän kuin yksi. Neuvottelusiltalaitteilla voidaan muodostaa yhteyksiä ISDN-, ATM- ja IP-verkoista ja millä voidaan muodostaa neuvotteluyhteys tarvittaessa useamman henkilön kanssa. Portinvartijan tehtävänä on reitittää neuvotteluyhteykset tietyille verkkolaitteille ja yhdyskäytävä

sovittaa eri siirtotekniikat toisiinsa. Palomuureilla määrätään, millaista liikennöintiä sallitaan kahden erilaisen verkon välillä. [Köyste et al., 2003]

Kuvapuhelintekniikoiden suurimpia haittoja käyttäjälle voivat olla Internet-yhteyden hitaus, kuvapuhelimen näytön pienuus, kuvan ja äänen säätämismahdollisuuksien puuttuminen ja kuvan ja äänen eriaikaisuus.

Ehyt-hankkeessa käytetään siirtotekniikkana Tampereen kaupungin sisäistä verkkoa. Tällä hetkellä he kilpailuttavat kuvapuhelinlaitteistoa, joten keskityn siirtotekniikkaan, jota useimmat kuvapuhelimet käyttävät.

Yleisin siirtotekniikka on ISDN. ISDN-verkko on osa yleistä puhelinverkkoa ja perinteisten operaattoreiden hallinnassa. Kuvapuhelinta käytettäessä olisi hyvä käyttää kahta yhteyttä, toista kuvan ja toista äänen siirtämiseen. Silti kuvanlaatu voi vaihtua hitaammin kuin ääni. Haluttaessa voidaan saada vielä parempaa kuvaa lisäämällä ISDN-linjojen määrää kolmeen. Tällöin on käytössä kuusi kappaletta 64 kb:n kanavaa. [Köyste et al., 2003]

Dataverkot käyttävät kaikkialla samaa tiedonsiirtokäytäntöä, IP-protokollaa. Videoneuvotteluille dataverkko antaa maailmanlaajuisen kattavuuden, mutta haittana siinä on vaihteleva tiedonsiirtokaista, mikä aiheuttaisi ongelmia kuvan ja äänen laadulle. [Köyste et al., 2003]

Paikallisverkoissa ATM on runkoverkkotekniikkaa, jossa tiedonsiirto tapahtuu esimerkiksi IP:tä käyttäen. Tämä tekniikka tarjoaa hyvän vaihtoehdon muille siirtotekniikoille, sillä tässä verkkotekniikassa voi varata halutun kapasiteetin kuvapuhelinta varten. Toinen runkoverkon hyvä ominaisuus on se, etteivät verkon ruuhka-ajat pääse haittaamaan neuvottelun videokuvan laatua. Ongelma ATM-verkossa on sen rajoittunut levinneisyys. [Köyste et al., 2003]

Viimeisimpänä tiedonsiirtoväylänä ovat langattomat siirtoväylät, jotka toimivat GSM- ja UMTS-verkostoissa. Ne pystyvät siirtämään kuvaa lyhyiden etäisyyksien päähän, mutta pitempien matkojen kuten satojen kilometrien matkalla kuvan ja äänen siirtäminen ei vielä onnistu tarpeeksi hyvin. Tulevaisuudessa tämä tulee parantumaan. [Köyste et al., 2003]

4. Käyttötapahtuma

Ehyt-hankkeessa neljään päiväkotiin hankitaan kuvapuhelimet. Nämä päiväkodit ovat Hippos, Kaleva, Kanjoni ja Kisapuisto. Päiväkodit on valittu siten, että kuvapuhelimen käyttäjät ovat viittomakieliset ja kuntouttavan varhaiskasvatuksen piirissä olevat perheet sekä maahanmuuttajaperheet. Perheiden kanssa tehtävät kuvapuhelinpalaverit toimivat päiväkodeissa. Siellä varataan aika perheiden sekä tulkkipalvelun kanssa. Tämän jälkeen päivähoitohenkilöt, jotka ovat erikoistuneet toimimaan kuvapuheliman kanssa, ottavat yhteyden tarvittavaan tulkkiin, jolloin he sekä perheet alkavat keskustelemaan ennalta

sovituista asioista. Asiat ovat henkilökohtaisia, joten tietoturvallisuudesta on huolehdittava. Koska kuvapuhelin toimii tällä hetkellä Tampereen kaupungin sisäisessä verkossa, isoa ongelmaa ei ole, mutta silloin kun kuvapuhelimet yleistyvät tai tarvitaan yhteyttä Tampereen kaupungin sisäisen verkon ulkopuolelle, tähän ongelmaan on kiinnitettävä huomiota. Näköpuhelimen käyttäjät toimivat näköpuhelimen kanssa yksityisessä, rauhallisessa tilassa, jossa he voivat toimia rauhassa luottamuksellisten asioiden kanssa.

4.1. Käyttäjäryhmät

Kuulovammaiseksi kutsutaan henkilöä, jolla on jonkinasteinen kuulovamma. Yleisempiä luokkia ovat normaalikuuloiset, huonokuuloiset (lievä, kohtalainen, keskivaikea, vaikea) varhaiskuurot, sekä kuuroutuneet.

Kuurot kommunikoivat useimmiten viittomakielellä. Viittomakieli tuotetaan käsien liikkeillä, ilmeillä ja suun ja vartalon liikkeillä. Kuurot ovat syntymäkuuroja tai kuuroutuneet ennen puheen oppimista [Kuurojen liitto, 2004].

Viittomakieli on kuurojen yhteisössä käytettävä luonnollinen kieli, joka on käsien ja vartalon liikkeistä, sekä ilmeistä koostuvien merkkien (viittomien) järjestelmä. Nämä otetaan vastaan näön avulla. Viittomakielet ovat yhdenveroisia kielellisiä järjestelmiä puhuttujen kielten kanssa. Viittomakieli muuttuu käyttötilanteen mukaan. Siinä on myös alueellisia ja sosiaalisia murteita. Viittoessa voi tuottaa ja nähdä yhtäaikaisia, kerrosteisiä viestejä, jolloin hyvään visuaaliseen palautteeseen on kiinnitettävä huomiota. Viittomakieli on ainoa kieli, jonka kuuro voi omaksua vaivatta ilman erityistä opetusta. [Kuurojen liitto, 2004]

Huonokuuloiset ovat myös puheella kommunikoivia kuulovammaisia. Kuulonvajaus vaihtelee lievästä vaikeaan. Ryhmään voidaan laskea myös henkilöt jotka ovat menettäneet kuulonsa puheen oppimisen jälkeen. Heidän apuvälineenään on yleensä kuulolaite, mutta se ei takaa normaalia kuuloa. Vaikeasti huonokuuloiset käyttävät myös huuliolukua tai viitottua puhetta kommunikaationsa tukena. [Kuurojen liitto, 2004]

Maahanmuuttajat ovat eri kansallisuuksista. Heidän tarpeensa ovat lisääntyneet vähitellen. Osa heistä kommunikoi omalla äidinkielellään ja jotta kaikki osapuolet tulevat ymmärretyksi, heidän kanssaan työskentelemiseen tarvitaan heidän kielensä tulkkia.

Kuntouttavaa varhaiskasvatusta tarvitseva lapsi päivähoitolain käsitteen mukaisesti on erityistuen tai tehostetun tuen tarpeessa oleva lapsi. Kuntouttavaa varhaiskasvatusta tarvitsevan lapsen tunnistaneet vanhemmat, kasvattajat ja terveydenhuollon asiantuntijat ovat huomioineet lapsen kehitykseen ja oppimisvaikeuksiin liittyviä riskitekijöitä.

Valtakunnallisten tutkimusten mukaan keskimäärin 10-15 prosenttia lapsista tarvitsee erityistä huomiota ja tukea kehitykseen ja oppimiseen liittyvissä vaikeuksissa. Yksi päivähoidon keskeisistä päämääristä on ”kuntouttava arki”, jossa lapset saavat kasvatusta ja opetusta muiden lasten kanssa. Tavoitteena on saada lapsi päivähoitoon kodin lähelle ja tarvittavat tukitoimet lapsen lähelle. [Tampereen kaupunki, 2003]

Kuva 2. Kuvapuhelin neuvottelutilanteessa.

4.2. Käyttäjien erityistarpeet

Kuvapuhelin laitteistolla täytyy huomioida tietyt asiat, jotta kommunikointi kaikkien osapuolien kanssa olisi mahdollisimman helppoa ja vaivatonta. Tämän takia suunnittelijoiden täytyy huomioida hyvä laitteisto, kuvanlaatu, audioliitännät, valaistus ja tilan tarvitsemat ominaisuudet.

Hyvä kuvanlaatu. Viittomakielen käyttö asettaa vaatimuksia kuvapuhelimen kautta välittyvälle videokuvalle. Kameroiden sijainti on suunniteltava hyvin ja olisi luotava mahdollisuus, että kameraa olisi helppo tarvittaessa siirrellä. Kuvan täytyy olla terävä ja kuvan rakeisuutta on pyrittävä poistamaan, jotta huulten liikkeet, näkyvät selkeästi. Monitorille ei saa tulla heijastuksia ja pintojen on oltava puhtaita. Taajuus on suositeltavaa olla 30 kuvaa/s, sillä tällöin se näkyy normaalina juoksevana kuvana [Köyste et al., 2003]. Videokuvan päivitys on oltava samanaikaista äänen kanssa. Jos äänen ja kuvan välillä on viivettä, puheen ymmärtäminen on vaikeaa sillä jos päivitys ei suju, on vaikeaa ymmärtää kahden eriaikaisen tiedon sekoitusta.

Köysteen ja muiden [2003] raportista käy ilmi, että huonokuuloiset voivat käyttää tietokoneen neuvottelulaitteiston kuulokeliitäntään liitettävää induktiosilmukkaa. Silmukan valinta riippuu käyttötilanteesta. Mikäli paikalla on useampi kuulolaitteen käyttäjiä, on suositeltavaa käyttää huoneen kiertävää silmukkaa, jolloin yksi silmukka on kaikkien läsnäolijoiden käytössä. Yksit-

täinen käyttäjä voi käyttää kaulasilmukkaa tai yhdistää FM-laitteensa sopivalla liitäntäjohdolla tietokoneeseen, jolloin langaton kuuntelu mahdollistuu.

Hyvä ja selkeä ääni. Hyvän kaksisuuntaisen äänentoiston aikaansaaminen osapuolten kesken vaatii mikrofonien ja kaiuttimien järjestämistä neuvotteluhuoneessa. Kaiunkumous ja sen laadukas toteuttaminen vaikuttavat äänen kuuluvuuteen. Neuvotteluhuoneessa olisi minimoitava kaikuminen, sillä ne haittaavat puheen ymmärtämistä. Nämä tekijät haittaavat puheen ymmärtämistä. Mikrofonin sijoitus on myös tärkeää. Jos mikrofonin sijoittaa liian kauaksi puhujasta, ääni hajoaa tai synnyttää kaikua. Liian läheinen mikrofonin sijoittelu voi äänittää myös henkäykset, jotka haittaavat puheen ymmärtämistä. Kuulolaitteen käyttäjille on myös huomioitava heidän istumapaikkansa, jotta kaiuttimista tuleva ääni saavuttaisi päämääränsä selkeästi ja jotta se ei kaikuisi.

Hyvä valaistus. Valaistuksen on neuvotteluhuoneessa oltava säädeltävissä yksilöllisiin tarpeisiin, jotta se ei olisi liian kirkas, eikä hämärä vaan jotta se antaisi mahdollisuuden selkeälle kommunikoimiselle. Viittoen kommunikoitaessa varjostuksen muodostuminen on estettävä kasvojen alueella, jotta huulien liikkeet sekä kädet näkyisivät selkeästi.

Sopiva neuvottelutila. Parhaimmillaan neuvottelutila on pienehkö huone. Tilan taustan tulee olla rauhallisen sävyinen, jotta kommunikointi näkyisi selkeästi, eikä neuvottelun häiritseviä tekijöitä olisi. Neuvottelevien henkilöiden vaatetus tulisi olla myös hillityn sävyinen. Päivähoitohenkilöiden mielestä kuvapuhelintilanteeseen sopii parhaiten tumma vaatetus, jotta viitottaessa vaaleiden käsien liikkeet näkyisivät kontrastia eli tummaa taustaa vasten paremmin. Tilan kaikuminen voi olla myös ongelmana. Sitä voi tarvittaessa vähentää erilaisilla akustointilevyillä ja vähäkaikuisella lattiamateriaalilla.

5. Tulkauspalvelu

Tulkkien ammattikunta on viime vuosikymmenten aikana yrittänyt antaa kielipalveluita maahanmuuttajille ja viranomaisille. Suomessa maahanmuuttajilla on lain turvaama oikeus saada asioida äidinkielellään viranomaisten kanssa esimerkiksi sosiaali-, terveystoimen sekä oikeuslaitoksen piirissä. [Tuhkanen, 2004]

Tulkkikeskukset tarjoavat maksuttomia tulkkaus- ja käännöspalveluita esimerkiksi vastaanottokeskuksille, kotouttamiseen ja turvapaikanhakuun liittyvissä asioissa sekä sosiaali- ja terveyden huoltopalveluissa.

Maahanmuuttajat lisääntyvät ja tulkkien palveluita tarvitaan yhä enemmän. Tulkkipalvelulla täytyy olla tulkkeja, jotka yhteensä osaavat kymmenen eri äidinkieltä. Tällä hetkellä Suomessa on seitsemän alueellista ja yksi yksityinen tulkkikeskus, jotka ovat vastanneet tulkkaustarpeeseen. [Tuhkanen, 2004]

Tulkki saattaa joutua matkustamaan jopa useita tunteja lähitulkkaustilanteeseen. Kuvapuhelimen avulla tulkki voi olla päivän aikana yhteydessä joka puolelle kaupunkia eikä työaika kulu matkoihin. [Tuhkanen, 2004]

Etätulkkaus on tulkkaustilanne, jossa tulkki ja asiakas eivät fyysisesti ole samassa paikassa. He ovat yhteydessä toisiinsa puhelimen, tietokoneen, kuvapuhelimen tai videoneuvottelulaitteiston välityksellä. Yleisin etätulkkauksen muoto on puhelinneuvottelut.

Kuvapuhelinyhteydessä tulkki voi katsoa yhtä tai useampaa näyttöä. Tällä tavoin hän pystyy pitämään tarvittaessa katsekontaktin henkilöön ja tarvittaessa etsimään tietoa toisesta tietokannasta esimerkiksi terveydenhuolto- tai maahanmuuttajatapauksessa.

Etätulkkauksen kustannukset ovat kohtuulliset. Laitehankintojen jälkeen kuvapuhelinneuvottelu ISDN-yhteydellä maksaa puhelinmaksun verran kutakin linjaa kohden. [Tuhkanen, 2004]

6. Ehyt-hanke nyt

Olen ollut tekemässä havaintoja Ehyt-hankkeen kahdessa neuvottelukokouksessa, joissa on ollut päiväkotien ja Media Tampereen henkilökuntaa. Tämä luku perustuu näihin havaintoihin.

Ehyt-hankkeessa on joulukuun 2004 ensimmäisellä viikolla hankittu kuvapuhelimet kahteen päiväkotiin, Kalevaan ja Kisapuistoon. Nämä kuvapuhelimet ovat vasta kokeiluvaiheessa, jossa henkilökunta näkee ja voi kokeilla laitetta ensimmäisiä kertoja. Media Tampere kilpailuttaa kuvapuhelintoimittajia saadakseen hinta-laatu -suhteeltaan hyvät kuvapuhelimet kaikille päiväkodeille heti vuoden 2005 alussa. Ensi vuoden alkupuolella on suunnitteilla ensimmäiset viralliset kuvapuhelinkokeilut neljän päiväkodin kanssa. Päivähoitohenkilökunta pitää myös yhteisiä tapaamisia, joissa he miettivät, missä tilanteissa kuvapuhelimet tulevat eniten hyödyksi.

6.1. Kuvapuhelimen toiminnalliset rajoitukset

Tällä hetkellä Kaleva ja Kisapuisto ovat kokeilleet puhelinyhteyttä toisilleen ja harjoitelleet sen käyttöä. He ovat myös panneet merkille mitkä asiat ovat tässä ensimmäisessä kokeilussa olleet huonoja ja tulevaisuudessa korjattavissa.

Ensimmäisenä teknillisenä harmina he ovat maininneet, että puhelimen soittoääni on liian hiljainen. Soittoääntä on vaikea kuulla, jos toimistolla tai kuvapuhelinhuoneessa ei ole ketään lähistöllä. Toisena teknillisenä haittana pidetään sitä, että näiden ensimmäisten kokeilujen yhteydessä kuva tulee puheeseen verrattuna viiveellä. Tämä johtuu osakseen siitä syystä, että kokeilut on tehty kello kolmen ja neljän aikaan iltapäivällä, jolloin Internet-yhteydet

ovat suurimmassa käytössä. Oikeassa käytössä tämä epäselventäisi kommunikointia viitottaessa. Kokeilijat ovat sitä mieltä, että ei ole välttämättä helppoa siirtää palavereja sellaiseen ajankohtaan, jolloin Internetiä ei käytettäisi paljon.

Kuvapuhelin on toiminut äänellisesti hyvin, paitsi hiljaisen hetken jälkeen aloitettaessa uudelleen puhumaan, se leikkaa puheen alun pois. Päivähoitohenkilöt ovat miettineet, että jos viivettä ei saada pois, niin se hidastaisi palavereja. Tällöin kuvapuhelin käyttö ei hyödynnyttäisi heitä ajallisesti. Neljäntenä teknillisenä haittapuolena on, jos kuvan suurentaa koko näytön laajuudelle, he eivät näe omaa pientä kuvaansa. Tällöin myös kuva huononee eikä ei pysy enää tarkkana.

Käyttäessään kuvapuhelinta päivähoitohenkilökunta on huomannut, että tilan tulee olla siisti ilman häiritseviä tekijöitä, sekä viitottaessa vaatteiden on hyvä olla tummat. He ovat huomioineet, että heidän täytyy myös tarkastaa missä he pitävät kameraa, jotta vastapuoli näkisi hyvin heidän ilmeensä sekä liikkeensä. Ongelmana he pitävät, että jos palaverissa on useampia henkilöitä, joihinka kamera pitää vuorotellen siirtää, niin silloin aikaa menee kameran paikan säätämiseen tai liikutteluun.

Kommunikoidessaan he ovat huomanneet, että heidän täytyy katsoa kameraan eikä näytölle, josta he näkevät myös oman kuvansa sekä sen, että heidän täytyy odottaa toisen puheenvuoron loppuun, jotta he eivät puhuisi päällekkäin. Vuoropuhelu voi olla välillä vaikeaa, elleivät käyttäjät ole sopineet tiettyjä sääntöjä.

6.2. Hyviä puolia

Päivähoitohenkilöstö on erittäin innostunut kuvapuhelinkokeilusta. He eivät vierasta tekniikkaa ollenkaan. Heillä on hyvät kuvapuhelintilat ja valmis koneisto kuvapuhelinohjelmaa varten. Jos kaikki saataisiin toimimaan, he uskovat, että kuvapuhelin tulisi tarpeeseen. Se auttaisi tulkkipalveluissa, sekä se voisi yhdistää päiväkoteja toimimaan yhdessä enemmän. He ovat keksineet paljon hyviä uusia ideoita, joilla voisi toimia tavallistenkin asioiden kanssa. He voisivat opettaa esikoululaisille mediakasvatusta, sekä heillä voisi olla kummiryhmä. Lapset voisivat tällöin kuvapuhelimen kanssa pitää yhteyttä kaupungin toisella puolen olevaan päiväkotiin tai jopa pitää pientä askartelukerhoa.

Kuvapuhelin toimisi hyvin myös, jos jossain päiväkodissa tarvittaisiin apua viitottaessa ja jolloin ei saisi yhteyttä tulkkipalveluun. Henkilöstö voisi kuvapuhelimen välityksellä opetella tai kerrata tiettyjä viittomakielen sanastoja. Fyysinen sijainti ei olisi tällöin esteenä. Päivähoitohenkilöstö haluaa hurvitella myös ajatuksella, että omalle ruokalan henkilöstölle voisi toinen ruokalan

emäntä antaa vinkkejä miten päiväkodin henkilöstö saisi maukkaita välipaloja aina silloin tällöin.

7. Kuvapuhelimen menneisyys, nykyisyys ja tulevaisuus

7.1. Historiaa

Seuraavat kolme tapausta olen valinnut näyttämään kuvapuhelimen kehitystä ja sen aikaisempia tapahtumia. Kuulovammaisten tulkkipalvelu tuli lakisääteiseksi vuonna 1979 osaksi invalidihuoltolain mukaista lääkintähuoltoa [Hakulinen ja Savela, 2000].

Tapaus 1. Vuonna 1996 työministeriö toteutti Vaasassa etätulkkauskokeilun. Hankkeeseen osallistui Vaasan läänin tulkkikeskus (nykyinen Pohjanmaan tulkkikeskus) vastaanottokeskus ja työvoimatoimisto. Seitsemän kuukauden aikana he yrittivät tutkia etätulkkauksen hyviä ja huonoja puolia sekä sitä, säästääkö etätulkkaus kustannuksia. Tulokset kertoivat, että puolen vuoden osalta etätulkkaus on sitä taloudellisempaa mitä kauempana asiakkaat ovat ja mitä useampia tulkkauksia tehdään. Tulkeilta työaikaa säästyi kuusi viikkoa.

Tapaus 2. Suomessa Kuurojen Liitto toteutti vuosina 1998-2000 multimediaprojektin, jossa tarjottiin viittomakielisiä palveluita kuvapuhelintekniikan ja puhelinverkon välityksellä. Projekti koostui kahdesta pilottihankkeesta, jotka toimivat Oulun ja Lapin läänissä sekä Pohjois-Karjalassa. Oulun ja Lapin läänissä toteutettiin Pohjois- ja Itä-Suomen sosiaalinen verkko-hanke ja Kuvapuhelin tulkkipalveluissa -hanke Pohjois-Karjalassa. [Tuhkanen, 2004]

Ensimmäisessä pilottihankkeessa pyrittiin tarjoamaan sosiaalisia palveluita kuvapuhelimen välityksellä sekä ennaltaehkäisemään syrjäytymistä. Jälkimmäisen hankkeen tavoitteena oli saada kuurot toimimaan etätulkkauspalvelun avulla itsenäisinä tietotekniikkayhteiskunnan jäseninä. Hankkeeseen osallistuneet viittomakieliset pitävät siitä, että palvelu paransi heidän elämän laatuaan sekä olivat tyytyväisiä palveluihin. [Tuhkanen, 2004]

Tapaus 3. Vuosina 2002-2004 toteutettiin Raha-automaattiyhdistyksen rahoittaman Esteetön etätulkkaus -projekti. Tämän projektin avulla pyrittiin parantamaan viittomakielisten tulkkien saatavuutta Internetin välityksellä. Samalla pyrittiin supistamaan kunnille koituvia tulkkauskuluja sekä vähentämällä tulkkien matkustamiseen kuluvaa aikaa. Projektissa pyrittiin luomaan ympärivuorokautinen etätulkkausmalli, jotka suunnattiin sekä yksityisille että julkisille organisaatioille, kuten sairaaloille, sosiaalikeskuksille ja poliisille. Samaan aikaan Helsinkiin perustettiin etätulkkauspiste, joka palveli kahdesta neljään tuntia päivässä neljänä kertana viikossa. Tulkkausta sai myös varaamalla sitä erikseen. Projekti sai vuoden 2004 TERVE-SOS-palkinnon.

Tämä palkinto myönnetään vuosittain tutkimus-, kehittämis- tai koulutushankkeelle, jonka toteutusta pidetään erityisen ansiokkaana. [Tuhkanen, 2004]

7.2. Nykyisyys

Suomen tulkkikeskuksissa etätulkkauksen käyttö on vähäistä lähi- tai puhelintulkkaukseen verrattuna. Pohjanmaan ja Turun seudun tulkkikeskukset ovat maan edellä kävijöitä tällä saralla. Pohjois-Suomen etätulkkaus on erittäin vähäistä. [Tuhkanen, 2004]

Vuonna 2003 Pohjois-Suomen tulkkikeskuksesta tehtiin 7389 tuntia sekä puhelin- että kuvapuhelintulkkausta noin 30 eri kuntaan ja 34 eri kielellä. Etätulkkauksen määrä tästä luvusta oli vain 23,5 tuntia. Etätulkkauksen käyttö on vielä erittäin vähäistä myös niissä toimipisteissä mihin laitteet on hankittu. Etätulkkauksen heikkoon käyttöprosenttiin on mahdollisesti se syy, että Pohjois-Suomen koulujen, poliisin, oikeuslaitoksen ja mielenterveys- palveluiden käytössä ei ole kuvapuhelinyhteyttä. [Tuhkanen, 2004]

Hakulinen ja Savela[2000] tutkivat ratkaisuvaihtoehtoja tulkkipalveluongelmiin. Heidän ensimmäisenä ratkaisuvaihtoehtonaan oli kuvapuhelintulkkipalvelujen järjestäminen täydentämään olemassa olevia tulkkipalveluja. Näitä palveluntuottajia olisivat enimmäkseen kuntien sosiaalihuolto tai yksityiset tulkkikeskukset, säätiöt, vakuutusyhtiöt, seurakunnat ja osuuskunnat.

Toisena heidän ratkaisunaan oli kuvapuhelimien saatavuuden nostaminen. Pääideana oli kuvapuhelimien järjestäminen julkisiin tiloihin. Tämä mahdollistaisi tasa-arvoisen aseman kommunikoinnissa kuulovammaisille.

Kolmanneksi tärkein toimintavaihtoehto liittyi kuvapuhelintulkki- palvelujen keskittämiseen. Tällöin palvelut voidaan tuottaa yhdestä tai vaihtoehtoisesti joistakin toimipisteistä käsin koko maahan. Tällöin koko maan kuvatulkkipalvelut yleistyisivät sekä olisivat samanarvoisia kaikille kuulovammaisille.

7.3. Tulevaisuus

Tulevaisuudessa viittomakielisten etätulkkauspalvelut tulevat kehittymään. Langaton tiedonsiirto on kehittynyt viime vuosina räjähdysmäisesti ja sen myötä myös mobiililaitteiden tiedonsiirto on saanut uuden käsitteen. Langaton tiedonsiirto mahdollistaa myös erilaisten ryhmätukisovellusten käytön.

Valtion teknillinen tutkimuskeskus aloittaa Kuurojen liiton kanssa yhteisen hankkeen, jossa viittomakielen tulkkausta tarvitseva henkilö voi tilata tulkin matkapuhelimen näytölle[Tuhkanen, 2004]. Näin viittomakielien henkilö ottaa puhelimen lääkärin vastaanotolle ja soittaa Tulkkikeskukseen. Tulkki saataisiin näin ollen kulkemaan viittomakielisen mukana, jolloin saataisiin etätulkkauksen fyysinen sijainti katoamaan sekä viittomakielinen saisi itsenäisyyttä omatoimiseen asioimiseen.

Ajan kuluessa kotikoneiden suorituskyky kasvaa ja samalla niiden että kuvapuhelinlaitteiden ja -ohjelmien hinnat laskevat. Internetin laajakaistat yleistyvät yksityisillä puolilla. Myös monet kaupungit voivat kasvattaa kuvapuhelimien määrää eri sosiaalialoilla ja nostaa etätulkkauksen henkilökuntaa. Tärkeää on kuvapuhelinten tarpeen tiedostaminen.

Kuvapuhelimet voisivat tällöin yleistyä koteihin, jolloin yksityiset henkilöt voivat pitää helposti yhteyttä sukulaisiinsa ja ystäviinsä sekä tukihenkilöihinsä tai etätulkkauspalveluun.

8. Lopuksi

Kuvapuhelimella on huomattu olevan hyötykäyttöä monissa eri ihmisryhmissä. Vaikka lainsäädäntö ei vielä tunnekaan kuvapuhelinta apuvälineenä, ovat monet kunnat valmiita tukemaan kuvapuhelimen käyttöä vammaislain nojalla. Kunnissa on myös pohdittu miten varat riittäisivät tähän palveluun. Toiminnan ylläpitämiseen ja sen laajentamisen ongelmana ovat suuret investointikustannukset. Tällä hetkellä ne ovat vielä liian kalliita kotitalouksiin hankittaviksi. Mikäli tekniikka ja laajakaistan käyttö tulisi edullisemmaksi, voitaisiin kuvapuhelimen käyttökustannukset saada normaalin puhelun tai atk-käyttöliikenteen tasalle. Kunnat pystyisivät tällöin saavuttamaan säästöjä etätulkkauksella, jolloin ajomatkoja voitaisiin korvata puhelinmenoilla. Kuvapuhelin auttaisi myös monia henkilöitä, jotka tarvitsevat tulkkia säännöllisesti sekä se auttaisi heitä itsenäistymään.

Kuvapuhelimet ovat sopineet uusiin ympäristöihinsä, mutta konkreettinen käyttö on jäänyt vielä vähäiseksi. Monet eri tahot ovat kannustamassa ihmisiä käyttämään kuvapuhelinta, jotta se levittäytyisi laajempaan käyttöön. Käytön levittäytymiseen on silti paneuduttava ja siitä tiedotettava.

Ihmisillä, jotka eivät ole tottuneet teknologiaan on luonnollinen pelko sitä kohtaan. Tämä tarkoittaa vain sitä, että heille on näytettävä ja annettava mahdollisuus käyttää kuvapuhelinta. Kuvapuhelimen tulevaisuus onkin sen jälkeen tekniikan kehityksessä ja sen hintojen laskusuhdanteessa. Mitä parempia ja yksinkertaisempia tekniikoita voidaan antaa henkilöille kotikäyttöön, sitä enemmän useampi ihminen saa mahdollisuuden helpompaan ja itsenäisempään omatoimiseen työskentelemiseen.

Viiteluettelo

[Brown, 1992] Carl Brown, Assistive technology computers and persons with disabilities. *Communications of the ACM* **35**, 5 (1992) 36-45.

[Caditz et al., 2004] JJ Caditz, Attila Narin, Gavin Jancke and Anoop Gupta, Exploring PC-telephone converce with the enhanced telephony prototype. In: *Proc. CHI04*, ACM Press, 2004, 215-222.

[Hakulinen ja Savela, 2000] Tuovo Hakulinen ja Annamari Savela, Kuvapuhelimen tulkkipalvelu tulevaisuudessa. Saatavilla: http://www.mol.fi/esf/ennakointi/raportit/kuvapuhe.pdf. Linkki tarkastettu 9.12.2004.

[Kuurojen liitto ry, 2004] Kuurojen liitto ry, Viittomakielistä kumppanuutta. Saatavilla: http://www.kl-deaf.fi/index.html. Linkki tarkastettu 9.12. 2004.

[Köyste et al., 2003] Anneli Köyste, Timo Mäkimattila, Sami Hautakangas ja Pasi Häkkinen, Saavutettavien kuvapuhelinratkaisujen teknisten järjestelmien ja laiteratkaisujen kartoitus. Saatavilla: http://wwwold.kuntaliitto.fi/tietot/kuvapuh/raportti20030316.pdf. Linkki tarkastettu 9.12.2004.

[Media Tampere, 2004] Media Tampere, Ehyt-Erityisryhmien hyvinvointipalvelut ja kuvapuhelin. Saatavilla: http://www.mediatampere.fi/projektit/ehyt. Linkki tarkastettu 9.12.2004.

[Tampereen kaupunki, 2003] Tampereen kaupunki, Sosiaali- ja terveystoimi, Kuntouttavan varhaiskasvatuksen ohjelma 2003-2006. Saatavilla: http://www.tampere.fi/tiedostot/4WF6TsYRz/kuntvko.pdf. Linkki tarkastettu 9.12.2004.

[Tuhkanen, 2004] Suvi Tuhkanen, Viranomaisten kokemuksia etätulkkauksessa. Saatavilla: http://www.tkukoulu.fi/koululaitos/virtuaalitulkki/gradu.pdf. Linkki tarkastettu 9.12.2004.

Osallistuvaa kuntapäätöksentekoa edistävät verkkosovellukset

Mikko K. Lammi

Tiivistelmä
Tutkielma käsittelee suoraa kuntapäätöksentekoa edistäviä verkko-sovelluksia. Verkko-osallistumista ja verkkodemokratiaa kehittämällä pyritään lisäämään kunnan asukkaiden ja kunnan päätöksentekijöiden välistä vuorovaikutusta. Tutkielmassa verkko-osallistumisen menetelmät jaotellaan ryhmiin ja kartoitetaan millaisilla tietoteknisillä välineillä menetelmiä on mahdollista toteuttaa.

Avainsanat ja -sanonnat: kunta, kuntalaiset, internet, päätöksenteko, sähköinen hallinto, verkkodemokratia, verkko-osallistuminen, verkkosovellus, verkkovaikuttaminen

CR-Luokat: K.4, H.4.3, J.1

1. Johdanto

Kunnat ovat hiljalleen kehittäneet sähköistä hallintoaan tarjoamalla kuntalaisille erilaisia palveluita internetin välityksellä. Kuntien hallintoa on muutenkin pyritty tekemään kuntalaisille helpommin lähestyttäväksi. Tavoitteena on kuntalaisten aikaisempaa aktiivisempi osallistuminen päätöksentekoon.

Internet mahdollistaa aivan uudenlaisia menetelmiä, joiden avulla kuntalaiset voivat suoraan osallistua ja vaikuttaa kunnan asioihin. On olemassa tarve hyödyntää kuntalaisten ainutlaatuista osaamista ja tuottaa siten aikaisempaa parempia päätöksiä. Toisaalta taustalla voidaan nähdä myös tavoite siirtyä edustuksellisesta demokratiasta kohti suoraa demokratiaa, jonka toteuttamisen uudet tietotekniset välineet mahdollistavat.

Tarkoituksena on selvittää millaisia internetiä hyödyntäviä, suoraa kuntapäätöksentekoa lisääviä ja osallistumiseen kannustavia menetelmiä on olemassa. Tarkoituksena ei kuitenkaan ole jäädä pohtimaan kuinka paljon näistä menetelmistä on todellisuudessa hyötyä itse päätöksenteossa, vaan ainoastaan tarkastella löydettyjä menetelmiä mahdollistavia välineitä ja niiden taustalla olevia tietoteknisiä ratkaisuja. Pääasiallinen tutkimusmenetelmä on kirjallisuuskartoituksen tekeminen, ja valittu näkökulma kohdistuu erityisesti kuntalaisia ja päätöksentekijöitä yhdistäviin menetelmiin ja välineisiin.

Tutkielmassa käydään aluksi läpi sähköisen hallinnon teoriaa kuntasektorin näkökulmasta ja esitellään lyhyesti yksinkertaistettu malli kunnan päätök-

sentekoprosessista. Lisäksi verkko-osallistumisen ja verkkodemokratian teorian avulla pyritään pohjustamaan osallistumisen merkitystä ja sijoittumista sähköisen hallinnon kokonaisuuteen. Tutkielman toisessa osuudessa tutkitaan osallistuvan kuntapäätöksenteon kannalta oleellisia verkko-osallistumismenetelmiä. Viimeiseksi käydään lävitse tietoteknisiä välineitä, joita hyödyntämällä verkko-osallistumismenetelmät voidaan käytännössä toteuttaa.

2. Kuntapäätöksenteko verkossa

2.1. Sähköinen hallinto ja kunta

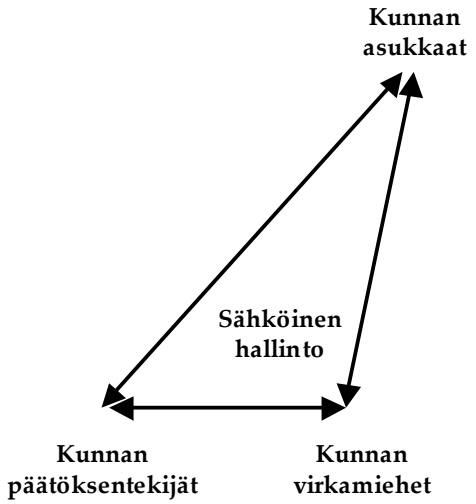
Kuntasektorin *sähköisellä hallinnolla* (e-government, electronic government, online government) tarkoitetaan tieto- ja viestintäteknologiaa aktiivisesti hyödyntäviä toimintoja ja prosesseja, joita kunta käyttää itse hyväkseen tai tarjoaa muille toimijoille. Tavoitteena on tietoteknisiä apuvälineitä hyödyntäen tehostaa ja uudistaa vanhoja toimintatapoja ja -prosesseja sekä parantaa tuottavuutta ja palvelun laatua. [Anttiroiko, 2002]

Grönlundin [2002, pp. 4-5] mukaan demokraattisen hallinnon peruselementit ovat kansalaisyhteiskunta, muodollinen politiikka ja hallinto, jotka ovat keskenään jatkuvassa vuorovaikutuksessa. Sähköisen hallinnon tavoitteena on parantaa näiden peruselementtien välistä vuorovaikutusta rakentamalla julkisyhteisöiden toimintaa uudella tavalla.

Grönlundin [2002, pp. 4-5] mallia mukaillen kunnan sähköisen hallinnon keskeisimmät toimijat ovat kunnan asukkaat, kunnan päätöksentekijät ja kunnan virkamiehet. Näiden toimijoiden välinen vuorovaikutus toimii hyvin vaihtelevasti ja monessa tapauksessa vain yksisuuntaisesti. Kunnan päätöksentekijät ja virkamiehet joutuvat väistämättä tekemään läheistä yhteistyötä, mutta kunnan asukkaat tuntevat usein jäävänsä liian kauaksi hallinnosta ja päätöksenteosta [Keskinen, 1995; Kivekäs et al., 2003]. Erityisesti kunnan päätöksentekijöiden ja asukkaiden välisessä vuorovaikutuksessa on paljon kehittämisen varaa, sillä asukkaita kohdellaan lähinnä tahdottomia tiedon omaksujina, joille riittää vain tehdyistä päätöksistä tiedottaminen [Kivekäs et al., 2003].

Tosiasiassa kuntalaiset tietävät monesta asiasta päättäjiä enemmän, joten vuorovaikutusta kehittämällä on mahdollista tehdä aikaisempaa parempia ja oikeudenmukaisempia päätöksiä [Keskinen, 1995]. Siitä missä määrin sähköinen hallinto parantaa päätöksenteon laatua ollaan eri mieltä, mutta tämä problematiikka ei kuulu tutkielman alueeseen. Sen sijaan siitä ollaan yhtä mieltä, että erilaisilla *verkko-osallistumista* (e-participation, electronic participation, online participation) ja *verkkodemokratiaa* (e-democracy, electronic democracy, online democracy) tukevilla välineillä on mahdollista aktivoida kuntalaisia mu-

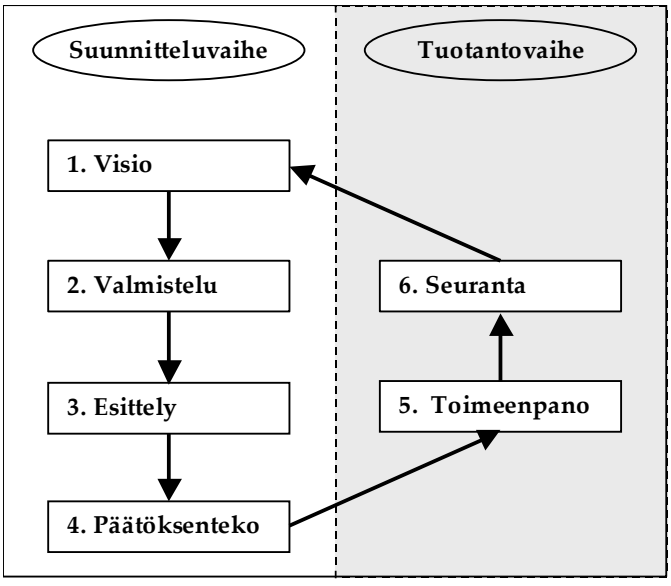
kaan päätöksentekoon, jolloin kuntalaisille tarjoutuu mahdollisuus tiedoillaan virallisia päätöksentekijöitä [UK cabinet office, 2002a; Keskinen, 1995].



Kuva 1. Kunnan sähköisen hallinnon tärkeimmät toimijat, niiden väliset vuorovaikutussuhteet ja vuorovaikutussuhteiden läheisyys [Muokattu: Grönlund, 2002, pp. 4-5]

2.2. Kunnan päätöksentekoprosessi

Kunnan monimutkaista päätöksentekoprosessia voidaan kuvata hieman yksinkertaistaen kuuden kohdan mallilla: 1. visio, 2. valmistelu, 3. esittely, 4. päätöksenteko, 5. toimeenpano ja 6. seuranta.



Kuva 2. Kunnan päätöksentekoprosessi vaiheittain kuvattuna [Muokattu: UK cabinet office, 2002b]

Päätöksentekoprosessi voidaan lisäksi jakaa kahteen suurempaan vaiheeseen, jotka ovat suunnitteluvaihe ja tuotantovaihe. Suunnitteluvaihe kuvaa poliittisten visioiden muotoutumista erilaisiksi toiminnoiksi ja ohjelmiksi, joiden tavoitteena on tuottaa halutut lopputulokset, esimerkiksi jokin julkinen palvelu. Tuotantovaihe käsittää tehtyjen päätösten toimeenpanon käytännössä ja näiden toimien vaikutusten arvioinnin. Toiminnan vaikutusten tai onnistumisen seuranta ja arviointi voi johtaa jälleen uusien visioiden syntymiseen, jolloin prosessi käynnistyy alusta udelleen. [UK cabinet office, 2002b]

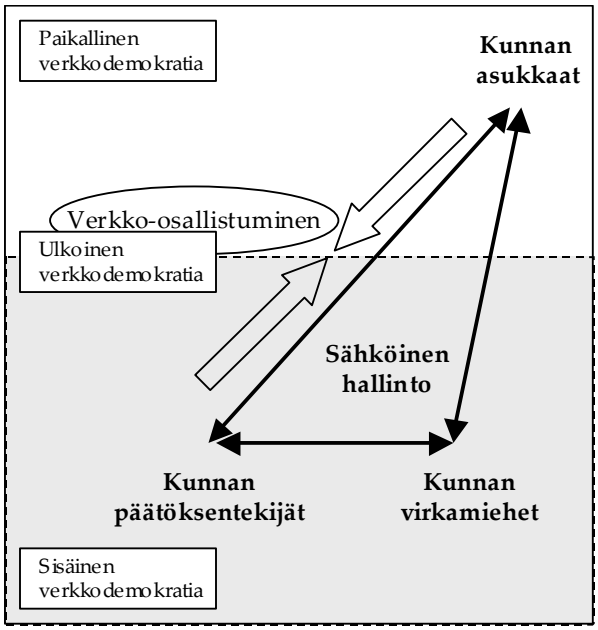
2.3. Verkko-osallistuminen ja verkkodemokratia

Verkko-osallistumisella pyritään kaventamaan kunnan asukkaiden ja kunnan päättäjien väliin jäävää kuilua. Osallistuminen korostaa kansalaisten mahdollisuutta olla aktiivisia vaikuttajia asioissa, jotka he kokevat tärkeiksi [Valtioneuvosto, 2002]. Kuntalaisten oikeudet osallistua ja vaikuttaa oman kuntansa asioihin on turvattu lainsäädännön kautta [Valtioneuvosto, 2002]. Lainsäädäntö ei kuitenkaan takaa aktiivista osallistumista, vaan kuntien on itse pidettävä huolta siitä, että kuntalaisilla on riittävät edellytykset osallistua ja vaikuttaa [Kivekäs et al., 2003].

Osallistumisen erilaiset menetelmät voidaan jaotella usealla eri tavalla. OECD:n raportti [OECD, 2001] arvioi osallistumista vuorovaikutuksen määrän perusteella, jolloin voidaan erottaa toisistaan tiedottava (yksisuuntainen vuorovaikutus), konsultoiva (neutraali, kaksisuuntainen vuorovaikutus) ja aidosti osallistuva (yhteistyöhän perustuva kaksisuuntainen vuorovaikutus) osallistuminen. Toinen vaihtoehto on käsitellä erilaisia osallistumisen menetelmiä päätöksentekoprosessin vaiheiden mukaisesti, jolloin päätöksentekoprosessin vaiheet määrittelevät myös osallistumismuodot. On kuitenkin huomattava, että eri osallistumismenetelmät ovat usein päällekkäisiä sekä vuorovaikutuksen määrän että päätöksentekoprosessin vaiheiden suhteen. Käytännössä verkko-osallistuminen voi muistuttaa mitä tahansa näistä osallistumisen menetelmistä tietoteknisessä muodossa toteutettuna.

Verkko-osallistumisen tavoitteena on verkkodemokratian toteuttaminen [Seppälä, 2000]. Verkkodemokratia kuuluu myös olennaisena osana sähköiseen hallintoon. Tarkemmin jaoteltuna verkkodemokratia voidaan jakaa paikalliseen, sisäiseen ja ulkoiseen verkkodemokratian toimijoista suhteista riippuen [Beaynon-Davies, 2004, p. 274]. Paikallinen verkkodemokratia on kansalaisten välistä vuoropuhelua, sisäinen verkkodemokratia vastaavasti on hallinnon välistä vuoropuhelua tietoteknisiä välineitä hyödyntäen. Tässä tutkielmassa keskitytään ulkoiseen verkkodemokratiaan, joka toimii rajapintana yhdistäen kunnan hallinnon ja kunnan asukkaat. Verkkodemokratian tavoitteena voidaan nähdä kaavoihin kangistuneen edustuksellisen demokratian elvyttäminen tai

vaihtoehtoisesti edustuksellisen demokratian muuntaminen kohti suoraa demokratiaa [Berg, 1995; Keskinen, 1995].



Kuva 3. Kunnan sähköinen hallinto jaoteltuna sisäiseen, ulkoiseen, ja paikalliseen verkkodemokratiaan

3. Verkko-osallistumisen menetelmät

Verkko-osallistumisen pohjana on aina hyvä tiedottaminen, sillä ilman toimivaa informaation välitystä kuntalaiset eivät välttämättä tiedä missä, miten ja mihin voi ylipäätään vaikuttaa. On myös tärkeää tiedottaa missä päätöksentekoprosessin vaiheessa tiedotettavat asiat ovat.

Verkossa tapahtuvaa, osallistuvaa kuntapäätöksentekoa voidaan toteuttaa monella tavalla. Rosén [2001, pp. 8-10] on jaotellut erilaiset menetelmät seitsemään kategoriaan, joita hieman muokkaamalla saadaan jaottelu: henkilökohtainen yhteydenotto, verkkojuttelu, konsultointi, sähköiset keskustelujärjestelmät, vuorovaikutussivut, kuntalaisaloitteet ja kansanäänestykset.

3.1. Henkilökohtainen yhteydenotto

Henkilökohtainen yhteydenotto on sähköinen vastine perinteisille kirjeelle tai puhelinsoitolle. Henkilökohtaisella yhteydenotolla pyritään vaikuttamaan suoraan johonkin tiettyyn päättäjään tai kysytään hänen mielipidettään jostakin asiasta. Sähköposti on käytännössä yleisin sähköinen henkilökohtaisen yhteydenoton väline.

3.2. Verkkojuttelu

Verkkojuttelu on vähän käytetty osallistumismuoto päätöksenteossa [Rosén, 2001]. Verkkojuttelu on suoraa reaaliaikaista vuoropuhelua kuntalaisten ja päättäjien välillä. Suomessa verkkojuttelua päättäjien kanssa on kokeillut valtionhallinnon kansalaisfoorumi otakantaa.fi, jossa kansalaisten kanssa ajankohtaisista asioista keskustelemassa on ollut mm. ministereitä [Otakantaa.fi, 2004]. Kuntasektorilla verkkojuttelua ei ole toistaiseksi juurikaan hyödynnetty.

3.3. Konsultointi

Konsultoinnilla voidaan tarkoittaa esimerkiksi verkossa toteutettavia kansalaisraateja, asiantuntijafoorumeita tai muita kuntalaisten ja päättäjien tiiviiseen vuorovaikutukseen perustuvia keinoja [UK cabinet office, 2002b]. Konsultoinnissa pyritään usein rajaamaan osallistujien joukkoa paremman vuorovaikutuksen takaamiseksi.

Kansalaisraadeissa tietyin kriteerein valittu tai vapaaehtoinen joukko kuntalaisia tutustuu päätöksenteon alla olevaan asiaan virallisen ja epävirallisen informaation avulla esimerkiksi sähköisiä dokumentteja lukemalla ja ennen kaikkea keskustelemalla keskenään internetin välityksellä. Keskustelu voi olla julkista, jolloin muutkin kuin konsultointiin osallistuvat voivat seurata keskustelua. Asiaan perehtyminen kestää tietyn ajan, jonka jälkeen kansalaisraadilta pyydetään heidän suosituksensa asiasta, joka viedään päättäjien tiedoksi ja julkaistaan kaikkien nähtäväksi kunnan internetsivuilla. Päättäjät voivat jatkaa vuoropuhelua kansalaisraadin kanssa antamalla raadin suosituksesta palautetta tai tekemällä jatkokysymyksiä.

Asiantuntijafoorumi muistuttaa kansalaisraatia sillä erotuksella, että siihen osallistuvat ovat asiantuntijoita, joilla voidaan olettaa olevan jo entuudestaan vahvat tiedot päätettävissä olevasta asiasta. [UK cabinet office, 2002b] Asiantuntijafoorumin tapauksessa on tärkeää, että kaikki halukkaat voivat seurata käytävää keskustelua koko konsultoinnin ajan.

Ruotsissa sijaitseva Kalixin kunta järjesti vuonna 2000 koko kunnan laajuisen konsultaation, johon kutsuttiin kaikki kunnan asukkaat keskustelemaan päättäjien ja toisten kuntalaisten kanssa kunnan keskustan uudistamisesta. Keskusteluun pystyi osallistumaan internetissä *keskustelufoorumilla* (bulletin board, online message board) tai perinteisemmillä menetelmillä kuten puhelimitse. Kunnan 15000 asukkaasta 1200 osallistui konsultaatioon. Osallistujista peräti 86% osallistui internetin välityksellä. Osallistumisen rajaus tehtiin antamalla jokaiselle kunnan rekisteröityneelle äänestäjälle henkilökohtainen salasana. Esimerkkejä pienemmän mittakaavan verkkokonsultoinneista on monia

muitakin, mutta Kalixin tapaus on yksi laajimmista. [Coleman and Gøtze, 2001 ja UK cabinet office, 2002a]

3.4. Sähköiset keskustelujärjestelmät

Sähköisten keskustelujärjestelmien, käytännössä yleensä keskustelufoorumien, välityksellä kuntalaiset ja päättäjät voivat vaihtaa mielipiteitä toistensa kanssa kirjallisesti ja lähes reaaliaikaisesti. Sähköiset keskustelujärjestelmät voidaan jakaa kansalaisjohtoisiin järjestelmiin ja hallintojohtoisiin järjestelmiin. [UK cabinet office, 2002b]

Kansalaisjohtoiset keskustelujärjestelmät ovat keskustelujärjestelmiä, joissa keskustellaan mistä tahansa kuntalaisia askarruttavista asioista. Kuntalaiset voivat itse määrätä keskustelunaiheista ja uudet kommentit keskusteluun tulevat heti näkyviin kaikille käyttäjille. Kunnan päätöksentekijät ovat mukana keskusteluissa, mutta heillä ei ole erityistä velvollisuutta vastata heille osoitettuihin kysymyksiin. [UK cabinet office, 2002b]

Hallintojohtoisissa keskustelujärjestelmissä keskustelu on paljon rajatumpaa. Keskustelu voidaan rajata koskemaan vain tiettyä aihetta, eikä kuntalaisilla ole välttämättä oikeutta vaikuttaa keskustelun aiheisiin. On myös mahdollista, että kuntalaisten kommentit menevät ensin *keskustelun ylläpitäjän* (moderator) hyväksyttäviksi, ja vasta sen jälkeen asiallisiksi varmistetut kommentit julkaistaan kaikkien nähtäville. [UK cabinet office, 2002b]

Suomalaisissa kunnissa on melko paljon käytössä erilaisia sähköisiä keskustelujärjestelmiä, joiden käyttö on hyvin vaihtelevaa. Jyväskylä ja Tampere tarjoavat suhteellisen vilkasta keskustelua erilaisilla periaatteilla. Jyväskylän Jyväskylätori-keskustelu (http://www.jyvaskyla.fi/keskustelu/) noudattaa vapaata kansalaisjohtoista käytäntöä, kun taas Tampereen keskustelupalsta (http://inter2.tampere.fi/osallistu/keskustelu/) on tiukasti hallintojohtoinen.

Kuva 4. Kansalaisjohtoista mielipiteiden vaihtoa Jyväskylätorilla

3.5. Vuorovaikutussivut

Vuorovaikutussivut tarkoittavat verkkosivuja, jotka toimivat jonkin merkittävän ajankohtaisen asian tiedotus- ja vuorovaikutuskanavana. Vuorovaikutussivut ovat useita erilaisia osallistumismenetelmiä yhdistäviä sivukokonaisuuksia, joilla voidaan jakaa virallista, päätöksentekoprosessissa tuotettua informaatiota, ja toisaalta samanaikaisesti ottaa kuntalaiset mukaan päätöksentekoon verkkokeskustelun, -kyselyiden, -pelien tai muiden suoraan osallisuuteen kannustavien välineiden avulla.

Jos asian päätöksentekoprosessissa ollaan edetty jo esittelyvaiheeseen, on mahdollista järjestää suunnitelmien virallinen kommentointi vuorovaikutussivujen yhteydessä. Tällöin suunnitelmia on mahdollista kommentoida, ja niihin on jopa mahdollisesti tehdä omia merkintöjä tai huomautuksia, joihin muut kuntalaiset voivat puolestaan vastata.

Usein isoista kunnallisista hankkeista on tapana perustaa oma projektisivu internetiin, jonka yhteyteen vuorovaikutteiset osallistumismahdollisuudet on loogista toteuttaa. Erittäin isoista hankkeista on mahdollista perustaa jopa kokonaan oma verkkopalvelu. Esimerkiksi Tampereella suunnitteilla olevan Vuoreksen kaupunginosan verkkopalvelu (http://www.vuores.fi/) tarjoaa monipuolisesti informaatiota ja yhdistää perinteistä osallistumista verkko-osallistumiseen tiedottamalla aktiivisesti mm. erilaisten ei-sähköisten workshop-tapahtumien tuloksista internetissä. Lisäksi Vuoreksen keskustan arkkitehtikilpailun ehdotukset julkaistiin verkossa, jossa ne olivat asetettavissa paremmuusjärjestykseen ja kommentoitavissa. [Vuores, 2004]

3.6. Kuntalaisaloitteet

Kuntalaisaloitteita voi tehdä monessa kunnassa sähköisesti internetsivujen kautta. Yksittäisen kuntalaisen tekemällä aloitteella ei kuitenkaan ole samaa painoarvoa kuin usean ihmisen yhdessä tuumin toimeenpanemalla aloitteella.

Virossa sähköistä aloitetta on kehitetty "Tana otsustan mina" ("Tänään minä päätän") -verkkopalvelulla, jossa on mahdollista esittää aloitteita muiden palvelun käyttäjien kommentoitavaksi. Näin tehtyä aloitetta saadaan kehitettyä yhteistyössä muiden palvelun käyttäjien kanssa, ja lopulta aloite voidaan lähettää eteenpäin päättäjille, jos se saa muilta käyttäjiltä yli 50% kannatuksen. [Coleman and Gøtze, 2001]

3.7. Kansanäänestykset

Edustukselliset verkkovaalit, eli *etä-äänestäminen* (televoting, e-voting, electronic voting), ei kuulu tämän tutkielman aihepiiriin, koska etä-äänestämisessä ei ole enää kyse suorasta osallistumisesta päätöksentekoon, vaan epäsuorasta osallistumisesta.

Verkkoa on kuitenkin mahdollista hyödyntää myös epävirallisempiin äänestyksiin, kuten erilaisiin neuvoa-antaviin äänestyksiin. Yksinkertaisimpia ovat kaikille avoimet sähköiset *mielipide- ja monivalintakyselyt* (polling, surveys), joita voidaan tukea antamalla lisäinformaatiota vastausvaihtoehdoista. Kyselyyn vastaajia ei välttämättä pyritä identifioimaan mitenkään, eikä vastaajien osallistumiskertojen määrää välttämättä pystytä rajoittamaan luotettavasti yhteen ilman raskaita teknisiä ratkaisuita. Monimutkaisemmat neuvoa-antavat äänestykset voivat olla osallistumisryhmältään tarkasti rajattuja, salasanoilla ja käyttäjätunnuksilla tai erilaisilla älykorteilla varmennettuja järjestelmiä. Esimerkiksi Kreikassa laajan neuvoa-antavan verkkoäänestysjärjestelmän suunnitelmat ovat edenneet jo pilottivaiheeseen. [Bouras et al., 2003]

4. Verkko-osallistumisen välineet

Verkko-osallistumisen menetelmiä voidaan toteuttaa käyttämällä erilaisia tietoteknisiä välineitä. Relevantteja välineitä ovat mm. sähköposti, reaaliaikainen keskustelu, keskustelufoorumit, kuvapohjainen kommentointi, pelit, kyselyt ja äänestykset sekä tiedottavat verkkosovellukset.

4.1. Sähköposti

Sähköposti on edelleen erittäin käytetty verkko-osallistumisen väline. Sen käytön helppous ja yleisyys on kuitenkin johtanut suuriin ongelmiin. Erilaisia sähköpostitiedusteluita, -ehdotuksia ja -huomautuksia tulee paljon enemmän kuin mihin päättäjät ehtivät vastata. Siksi henkilökohtaiselle sähköpostille on yritetty kehitellä korvaavia järjestelmiä, jotka helpottaisivat erityisesti postintulvan lajittelua.

Kollektiiviset sähköpostiosoitteet ovat yksi vaihtoehto, esimerkiksi poliittisen puolueen yhteinen sähköpostiosoite: puolue@puolue.fi. Niiden avulla monta ihmistä voi käyttää samaa sähköpostiositetta, jonka tarkoituksena on toimia keskitettynä kommunikointikanavana [Seppälä, 2003]. Toinen vaihtoehto ovat erilaiset *HTML-lomakkeisiin* (Hypertext Markup Language form, HTML form) perustuvat verkkoratkaisut, joiden avulla internetsivuille sijoitetulta HTML-lomakkeelta lähetetty viesti voidaan helpommin ohjata oikealle henkilölle. Lisäksi turhien sähköpostien korvaajaksi on kehitelty erilaisia kysymys-vastaus-palveluita, joihin voi lähettää kysymyksiä esimerkiksi kunnan päätöksenteosta. Kysymys ja siihen annettava vastaus julkaistaan verkkosivuille kaikkien tiedoksi. Yksi tällainen palvelu on Tampereen kaupungin kansalaiskioski. [Seppälä, 2003] Usein kuntalaiset haluavat kuitenkin ottaa yhteyttä juuri tiettyyn päättäjään, jolloin edellä esitellyn kaltaisista ratkaisuista ei ole apua [Rosén, 2001].

4.2. Reaaliaikainen keskustelu

Verkkojuttelun mahdollistava reaaliaikainen keskustelu voidaan toteuttaa verkkosivulle sijoitetulla sovelluksella, esimerkiksi sopivalla Java-sovelluksella. Sovelluksia on saatavilla internetistä täysin ilmaiseksi useita erilaisia vaihtoehtoja erilaisilla ohjelmointikielillä toteutettuina. Lisäksi verkkojutteluun voidaan käyttää muitakin kanavia kuin verkkosivuille sijoitettua sovellusta. *IRC* (Internet Relay Chat), *pikaviestimet* (instant messenger) ja *videoneuvotteluohjelmat* (video conference program) mahdollistavat kahdenkeskeisen tai korkeintaan muutaman kymmenen henkilön välisen yhteydenpidon internetiä hyödyntäen.

Reaaliaikaista keskustelua pidetään usein vain nuorten käyttämänä ajan tappamisena, mutta joistakin pikaviestimistä on kehittymässä hiljalleen vakavasti otettavia ja monipuolisia kommunikointivälineitä, jotka voidaan myös integroida osaksi palvelevia verkkosivuja *XML*-kieltä (eXtensible Markup Language) rajapintana käyttäen [Coleman and Gøtze, 2001].

4.3. Keskustelufoorumit

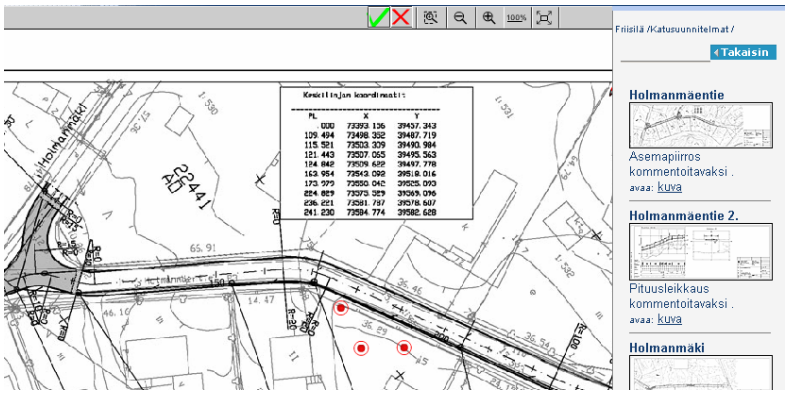
Keskustelufoorumit ovat välineitä monen ihmisen väliseen viestintään. Nykyiset, erittäin paljon käytetyt WWW-pohjaiset keskustelufoorumit ovat saaneet alkunsa *uutisryhmäkeskusteluista* (newsgroup discussion), joiden käyttö on koko ajan vähentynyt. WWW-pohjaiset keskustelufoorumit ovat verkko-osallistumisen kivijalka, joiden avulla voidaan toteuttaa monenlaisia osallistumis- ja vaikuttamismahdollisuuksia. Tavallisesti keskustelufoorumi integroidaan kiinteäksi osaksi muuta verkkosivustoa.

Keskustelufoorumien tekninen toteutus perustuu yleensä jonkin tietokannan ja dynaamisia *HTML-sivuja* (HTML-page) tuottavan ohjelmointikielen yhteistyöhän. Yleisesti käytettyjä ohjelmointikieliä ovat mm. Java, Php, Asp ja Perl. Monet keskustelufoorumeista ovat ainutkertaisia, räätälöityjä erikoissovelluksia, mutta on olemassa myös paljon yleissovelluksia, jotka voidaan sovittaa melkein mihin tahansa käyttötarkoitukseen. Monet kehittyneimmistä yleissovelluksista on kaupallisia, mutta on olemassa myös erittäin monipuolisia ilmaisia sovelluksia. Laajasti käytettyjä keskustelufoorumeita ovat mm. phpBB (http://www.phpbb.com/), yaBB (http://www.yabbforum.com/) ja UBB (http://www.ubbcentral.com/).

4.4. Kuvapohjainen kommentointi

Kuvapohjainen kommentointi on toistaiseksi vähän käytetty, mutta erittäin havainnollinen osallistumisväline. Kuvakommentoinnissa on tarkoituksen tehdä merkintöjä suoraan kommentoivaan kuvaan. Kuvaan tehtävien kommenttien tukena voidaan käyttää tekstikommentointia, jonka avulla voidaan paremmin selittää kuvaan tehtyjä merkintöjä. Kuvakommentointia on kokeiltu jonkin ver-

ran mm. Espoossa, jossa on ollut mahdollisuus "piirtää" omia merkintöjä mm. joihinkin asemapiirroksiin. Espoon käyttämän Feasy-sovelluksen ongelmana on kuitenkin sen vaatima Macromedia Flash -selainlaajennus, jonka vuoksi kaikki käyttäjät eivät pysty vaivattomasti hyödyntämään sovellusta [Feasy, 2004].



Kuva 5. Espoon kaupungin käyttämä kuva- ja tekstidokumenttien arviointijärjestelmä

4.5. Pelit

Vuorovaikutteiset pelit tarjoavat valtavan potentiaalin monipuolisille verkko-osallistumiselle, joiden avulla käyttäjät voivat osallistua asioiden päätöksentekoon lähes huomaamattaan. Viihteelliseen ulkoasusuun peitetty osallistuminen on osoittautunut helposti lähestyttäväksi keinoksi kannustaa päätöksenteosta välinpitämättömiä kuntalaisia tutustumaan päätöksentekoon. Pelien on myös todettu olevan tehokkaita oppimisen kannalta erityisesti nuorten keskuudessa. [Seppälä, 2000]

Osallisuuteen kannustavien verkkopelien tekniikka vaihtelee yksinkertaisista, dynaamisista HTML-sivuista erittäin monimutkaisiin sovelluksiin. Tampereella Viinikka-Nekalan -alueen suunnittelun yhteydessä annettiin kuntalaisille mahdollisuus osallistua kaavoitukseen vuorovaikutteisen kaupunkisuunnittelupelin (http://www.tampere.fi/ tiedotus_v/viinikka/frames.htm) avulla [Seppälä, 2000]. Pelin toteutus on yksinkertainen Javascriptin ja staattisten HTML-sivujen yhdistelmä, mutta karusta ulkoasustaan huolimatta lopputulos on innovatiivinen ja palvelee hyvin tarkoitustaan.

Kuva 6. Viinikka-Nekala -alueen kaupunkisuunnittelupeli

4.6. Kyselyt ja äänestykset

Kyselyillä ja äänestyksillä pyritään saamaan tietoa kuntalaisilta päätöksenteon tueksi. Usein verkkokyselyitä ja -äänestyksiä kritisoidaan niiden vinoutuneella vastaajakunnalla ja rajoitetuilla vastausvaihtoehdoilla, mutta silloin unohdetaan, että näillä sovelluksilla on myös huomattava tiedottava rooli. Kyselyihin ja äänestyksiin osallistuvat kuntalaiset ikään kuin houkutellaan kyselyn tai äänestyksen avulla lukemaan ohessa tarjottavaa lisäinformaatiota päätöksenteon kohteena olevasta asiasta.

Teknisesti kyselyt ja äänestykset toteutetaan tavallisesti HTML-lomakkeiden avulla. Joissain erikoistapauksissa Java- tai Flash-sovellukset tulevat kyseeseen. Kyselyiden tekemiseen on myös olemassa sovelluksia, joiden avulla verkkokyselyn tekeminen on mahdollista ilman teknistä osaamista. Yksi tällainen kaupallinen sovellus on E-lomake (http://www.e-lomake.net/), jonka avulla luotujen kyselyiden vastaukset tallennetaan suoraan tietokantaan, josta vastauksia on helppo käsitellä [E-lomake, 2004].

Erityishuomiota kyselyissä ja äänestyksissä on kiinnitettävä vastaajien identifioimiseen varsinkin, jos halutaan saada luotettavuudeltaan kelvollisia tuloksia. Kevyeen vastaajien tunnistamiseen on mahdollista käyttää esimerkiksi selaimen *evästeitä* (cookie). Jos vastaajat halutaan identifioida luotettavammin on apuna käytettävä esimerkiksi *sähköistä henkilökorttia*, *kaupunkikorttia* tai jonkin luotettavan kolmannen osapuolen tarjoamaa tunnistusta, kuten postin Netposti -tunnuksia [Seppälä, 2003]. Kunnissa yksi varteenotettava vaihtoehto voisi olla laajalle levinneiden kirjastokorttien ja niihin liittyvien tunnusten ja salasanojen hyödyntäminen.

4.7. Tiedottavat verkkosovellukset

Monet Suomen kunnista ovat siirtyneet käsin päivitettävistä verkkosivuista *sisällönhallintajärjestelmän* (content management system, CMS) käyttöön. Sisällönhallintajärjestelmät helpottavat oleellisesti internetsivujen ylläpitoa ja tehostavat siten erityisesti päätöksenteon julkisuuden kannalta oleellista tiedottamista. Kattavat sisällönhallintajärjestelmät ovat laajoja systeemejä, joihin voi olla sisällytettyinä verkko-osallistumisvälineitä, kuten keskustelufoorumi ja kysely- tai äänestystoimintoja.

Webocrat-järjestelmä (http://www.webocrat.org/) on yksi sähköiseen hallintoon ja verkkodemokratiaan erikoistunut tutkimushanke, jonka tarkoituksena on rakentaa kaikille ilmainen sisällönhallintajärjestelmä erityisesti verkko-osallistumista silmälläpitäen. Ohjelmassa on otettu huomioon aivan erityisesti verkko-osallistumisen tarpeet. [Webocrat, 2004]

Sähköpostilistat ja sähköpostikirjeet tarjoavat tietoa ajankohtaisista asioista tavallisen sähköpostin muodossa. Kunnat eivät juurikaan ole innostuneet tiedottamaan sähköpostin välityksellä, vaan katseet on jo suunnattu *RSS*-tekniikkaa (Really Simple Syndication, Rich Site Summary) kohden. RSS on XML-kielen laji, ja RSS-lukuohjelmalla voidaan lukea RSS:ää tukevan palvelun tarjoamia uutisia ja tiedotteita. Esimerkiksi Jyväskylä ja Tampere tarjoavat jo tiedotteitaan RSS-palveluna. [Kuusisto, 2004]

Tiedottaviksi sovelluksiksi voidaan lukea myös *suoratoistoa* (streaming) käyttävät audiovisuaaliset tiedotustekniikat, kuten Realaudio-lähetykset. Suoratoistolähetykset voivat olla joko aidosti suoria lähetyksiä tai nauhoituksia esimerkiksi kunnanvaltuuston kokouksista. Lähetykset voivat sisältää pelkästään ääntä tai kuvaa tai molempia yhtä aikaa.

5. Yhteenveto

Sähköinen hallinto on tullut kiinteäksi osaksi kuntien toimintaa. Internet on kasvava palvelukanava, johon kannattaa tulevaisuudessa panostaa entistä enemmän. Verkkodemokratia on yksi sähköisen hallinnon osa-alue, jota ei ole toistaiseksi kehitetty kovin paljoa, mutta joka tarjoaa paljon mahdollisuuksia suoran päätöksenteon edistämiseksi ja kansalaisten aktivoimiseksi mukaan päätöksentekoon.

Verkossa tapahtuvan osallistuvan kuntapäätöksenteon toteuttamiseksi on olemassa useita erilaisia menetelmiä, joita on yritetty tässä tutkielmassa jakaa isompiin ryhmiin paremman kokonaiskuvan aikaansaamiseksi. Verkkomenetelmät pääasiassa jäljittelevät perinteisen osallistumisen ja vaikuttamisen menetelmiä, mutta perinteisistä, ei-sähköisistä menetelmistä poiketen ne mahdollistavat huomattavasti helpompia ja tehokkaampia tapoja osallistua päätöksente-

koon. Verkko-osallistumisen avulla päätöksentekoon voi osallistua aikaisempaa paljon suurempi määrä kuntalaisia ilman valtavia kustannuksia.

Lisäksi tutkielmassa on pyritty tarkastelemaan menetelmien toteuttamiseen olennaisesti vaikuttavia tietoteknisiä välineitä ja niiden erityispiirteitä. Löydetyt välineet ovat enimmäkseen entuudestaan tuttuja kommunikointivälineitä, joita ei kuitenkaan ole totuttu ymmärtämään virallisen kuntapäätöksenteon osallistumisvälineiksi. Toisaalta kuntapäätöksenteko vaatii myös aivan uudenlaisia välineitä, joiden toimivuutta ei ole vielä testattu laajan yleisön keskuudessa. Uusien välineiden kehittäminen vaatii vielä paljon työtä ennen kuin välineiden laajamittainen hyödyntäminen on riittävän sujuvaa.

On huomattava, että tietotekniikka ja tietoverkot ovat kuitenkin vain välineitä, jotka tekevät mahdolliseksi kunnan päätöksenteon uudistamisen ja vallan uusjaon. Muutoksen laajuus ja onnistuminen riippuu lopulta näiden välineiden käyttäjistä.

Jatkoselvitystä vaille jäi, miten verkko-osallistumisen menetelmiä hyödynnetään päätöksenteon eri vaiheessa, ja mitä tietoteknisiä välineitä käytetään eri menetelmissä. Lisäksi olisi aiheellista pohtia kuinka merkityksellisiä eri menetelmät ja välineet todellisuudessa ovat osallistumisen ja päätöksenteon kannalta. Tähän vaikuttaa oleellisesti menetelmien toteutuksen helppous ja toteutuksesta aiheutuvat kustannukset. Myös välineiden toteutustekniikat kaipaisivat tarkempaa selvitystä.

Viiteluettelo

[Anttiroiko, 2002] Ari-Veikko Anttiroiko, eGovernment: eGovernment-alan tutkimuksen ja opetuksen kehittäminen Tampereen yliopistossa. Tampereen yliopisto, Tietoyhteiskuntainstituutti, Raportti **1/2002**, maaliskuu 2002. Tampereen yliopisto, Tampere, 2002. Saatavilla *http://www.uta.fi/laitokset/ISI/julkaisut/eGovernment-raportti_1-2002.html* (30.10.2004).

[Beynon-Davies, 2004] Paul Beynon-Davies, *e-Business*. Palgrave Macmillan, Basingstoke, 2004.

[Borg, 1995] Olavi Borg, Demokratia ja tietotekniikka. Teoksessa: Auli Keskinen (toim.), *Telademokratia: tietoverkot ja yhteiskunta*. Painatuskeskus, Helsinki, 1995, 36-41.

[Bouras et al., 2003] C. Bouras, N. Katris and V. Triantafillou, An electronic voting service to support decision-making in local government. *Telematics and Informatics* **20**, 3 (August 2003), 255-274.

[Coleman and Gøtze, 2001] Stephen Coleman and John Gøtze, Bowling together: Online public engagement in policy deliberation. Hansard society, 2001. Available as *http://bowlingtogether.net/bowlingtogether.pdf* (11.11.2004).

[E-lomake, 2004] E-lomake. *http://www.e-lomake.net/* (15.11.2004).

[Kuusisto, 2004] Päivi Kuusisto, Tampere tarjoaa tietoa monessa muodossa. *http://www.etampere.fi/infocity/uutiset/nayta.tmpl?id=640;etusivu=1* (15.11.2004).

[Feasy, 2004] Feasy, Kuva- ja tekstidokumenttien arviointia internetissä. *http://www.feasy.net/* (14.11.2004).

[Grönlund, 2002] Åke Grönlund, Introduction. In: Åke Grönlund (ed.), *Electronic Government: Design, Applications and Management*. Idea Group, Hershey, 2002, 1-22.

[Keskinen, 1995] Auli Keskinen, Johdatusta teledemokratiaan ja tietoverkkoihin. Teoksessa: Auli Keskinen (toim.), *Teledemokratia: tietoverkot ja yhteiskunta*. Painatuskeskus, Helsinki, 1995, 19-27.

[Kivekäs et al., 2003] Pekka Kivekäs, Reijo Linnamaa, Virpi Leino, Jari Seppälä, Leena Viitasaari, Veikko Vänskä ja Outi Teittinen, Tampere toimeenpanee strategiaa: Kuntalaisten osallistumismahdollisuuksien parantaminen -työryhmä, loppuraportti. Saatavilla *http://www.tampere.fi/tiedostot/4VO5E8EVP/loppuraportti.pdf* (29.10.2004).

[OECD, 2001] Citizens as partners. Information, consultation and public participation in policy-making. OECD publication, 2001. Available as *www1.oecd.org/publications/e-book/4201131e.pdf* (10.11.2004).

[Otakantaa.fi, 2004] Otakantaa.fi, Valtionhallinnon kansalaisfoorumi. *http://www.otakantaa.fi/online.arkisto.cfm* (11.11.2004).

[Rosén, 2001] Tommy Rosén, e-Democracy in practice - Swedish experiences of a new political tool. Svenska kommunförbundet, 2001. Available as *http://uno.svekom.se/brsbibl/kata_documents/doc28331_1.pdf* (11.11.2004).

[Seppälä, 2000] Jari Seppälä, Verkkomedia osallistumisen välineenä - case Tampereen kaupunkisuunnittelupeli. Kaupunkisuunnittelun ja Internetin interaktiivisuuden yhdistävä evaluointitutkimus. Tampereen yliopisto, tiedotusopin laitos, pro gradu -tutkielma, 2000.

[Seppälä, 2003] Jari Seppälä, Sähköinen viestintä osallistumisen välineenä. Liite Tampere toimeenpanee strategiaa: Kuntalaisten osallistumismahdollisuuksien parantaminen -työryhmä loppuraporttiin. Saatavilla *http://www.tampere.fi/osallistu/tyoryhma/pdf/lp_liite2.pdf* (29.10.2004).

[UK cabinet office, 2002a] In the service of democracy. A consultation paper on a policy for electronic democracy. UK Cabinet office, Office of the e-Envoy, Consultation paper, 2002. Available as *http://www.edemocracy.gov.uk/downloads/e-Democracy.pdf* (30.10.2004).

[UK cabinet office, 2002b] Background on implementation issues for e-participation in government. Background note for: In the service of de-

mocracy. A consultation paper on a policy for electronic democracy. UK cabinet office, Office of the e-envoy, 2002. Available as *http://www.edemocracy.gov.uk/downloads/Background_on_Imlementation.doc* (30.10.2004).

[Valtioneuvosto, 2002] Valtioneuvoston selonteko eduskunnalle kansalaisten suoran osallistumisen kehittymisestä 4.4.2002. Saatavilla *http://www.intermin.fi/intermin/images.nsf/files/C93962CC359D6F51C2256B91003EDEE1/$file/osallisuusselonteko.pdf* (29.10.2004).

[Vuores, 2004] Vuores, Tampereen ja Lempäälän yhteinen asuinalue. *http://www.vuores.fi/* (14.11.2004).

[Webocrat, 2004] Webocrat, Web technologies supporting direct participation in democratic processes. *http://www.webocrat.org/docs/R15_2_2.pdf* (14.11.2004).

User-Interfaces for Illiterate People

Deepa Mathew

Abstract

The modern world is facing a unique problem with illiteracy. It is estimated that half the world's population is illiterate [7] and many more use languages that are not supported in commonly used user-interfaces. Even young children who have not yet mastered the skills of reading/writing, the elderly people who find it difficult to read, understand and the visually impaired people can find the interface for illiterate people useful.

In this essay I will discuss the need of user-interfaces for illiterate people and the various design approaches that can be used and that are being used for developing this user interface and the limitations faced.

Key words and terms

User-Interface for illiterate people, Speech input and output, Cultural barriers, Graphical reading, Animations, Touch screen, Design approaches.

CR-classification: H.5

1. Introduction

Human Computer Interaction (HCI) is making revolutionary changes around the world. New ideas pop up everyday around the globe for a better, attractive, friendly, easy-to-use user interfaces. But sadly there are people who are completely or partially shut out of this modern world and the information society, like people who are physically or mentally challenged and the functional illiterate or illiterate people. An estimated 875 million adults are illiterate worldwide. [12]

Around the world, developed counties like USA, developing countries like India and the underdeveloped countries in Africa, are in an effort to break open these barriers. Nearly 98% of the world's illiterate population lives in developing countries. As of the year 2000, every second illiterate person in the world lives in India. [12]

Creating user-interfaces for illiterate people will make a great difference not only for the development of the country but also change the life of these people for the best. Information is knowledge and knowledge is life and for these people using computers and mobile phones to seek information or to communicate is a challenge.

User interface for illiterate people may seem strange to many people since we are talking about modern technologies when 40% of the world's people are abysmally

poor. These people do not even have proper food to eat and we are talking about new interfaces for computers and mobile phones?"

But the fact is that illiteracy is part of our economic problems. Not just an economic problem but also face social, health and community problems too. It limits their employment opportunities, their lives and their ability to earn a living or even teach their children. Helping them to help themselves to overcome this problem is a noble thing to do. I call this as '*Ray of hope*' for these people.

We should also keep in mind that illiteracy is not a disability but a lack of literal skill which is of course vital in this world today but should not be a limitation to access information. And as the technology develops we should try removing all kinds of barriers, let it be language, culture, distance or even illiteracy.

Listed below are few of the benefits. These interfaces

- Help people to access information which can be vital for employment.
- Bring awareness among their users by helping in reducing risk factors of various diseases, like AIDS, bring safety awareness and so on.
- Help improving the living standards for these people.
- Help the users in learning and knowing the world around them better.

Very few researches have been done on this topic and a workshop by Edwards and Kimble [2004] is still pending. There are many ideas that are jostling in our heads about how to design a user interface for illiterate/functional illiterate. But the key points to remember is that, whatever we design or create, it should be as simple as possible and also keep in mind the human psychology and cultural barriers, variations or diversity. Understanding cultures, human psychology and respecting it is also an important factor while developing user interfaces.

2. Basic User Requirements

Basic user requirements are some of the initial user requirements suggested by Deo et al. [2]

- Target users have a high need for *ease of learning*, and *ease of remembrance*, due to their low levels of formal education, little or no computer experience, low frequency of use, and potential uncertainty or anxiety towards technology.
- Minimal or no typing/reading skills suggest an interface with *no textual requirements.*
- No reading skill suggests that *icons and visual displays*, possibly with the use of speech synthesis will be necessary.
- The potentially broad cultural and ethnic backgrounds of users suggested an interface that supports *internationalisation*.
- Potential difference in language, ethnic, and cultural backgrounds of users also suggested an interface that accommodates *localisation.*

- The design should be *simple, easy to navigate, easy to use*, and *tolerant of errors* that will inevitable in the process of exploration and learning that the system promotes. The design must be as self-explanatory as possible, and thus easy to learn and remember, by leading users through steps and by providing adequate on-line support.
- To promote learning and exploration, the design content should be *useful*, and the design should be *robust.*

3. Design approaches and Technologies

There are many designs that can be used in developing this interface. As I mentioned earlier that simplicity is a key factor because the people going to use this interface are people with little or no computer skills. While designing keep in mind the needs of the target users since we are designing for them. It is always a good idle to have proper knowledge regarding the target users before designing the interface. Below are the golden rules of design [4]:

- Understand computers.
  - There limitations, capacities, tools platforms etc.
- Understand people.
  - The psychological, social aspects, human errors.

3.1. Touch screen [1]

Touch screen is a monitor screen that can detect and respond to something, such as a finger or stylus, pressing on it [3]. Touch screen (See Figures 1 and 2) is a very fast, simple and easy to use technology and is used worldwide. The screen acts as an input device as well as an output device. A person with or with out previous knowledge of using keyboard, mouse, keypad or any other input devise can easily access with just a touch on the screen with the finger or stylus.

[1] A monitor screen that can detect and respond to something, such as a finger or stylus, pressing on it.[3]

Figure1. Touch screen monitor.

Figure2. Touch screen in mobile phones.

Limitations:

- Bare hand pointing on touch screens both benefits and suffers from the nature of direct input.[6]
- Using the finger to point is not always suitable, as it can leave greasy marks on the screen.[4]
- A finger being a fairly blunt instrument can be quite inaccurate. That is, selecting small regions is very difficult with a finger touch.[4]

3.2. Speech Technologies

Speech is no doubt one of the best ways of communication. And it is not just limited between human-human but human-computer and computer-human also. But we also need to keep in mind that speech-based interaction should not mimic human-human interaction but has to support multitasking with speech appropriately. There are already a lot of speech technology software's available in the market worldwide.

Speech technology consists of mainly two parts: (See Figure 3) Speech synthesis (Speech Output) and Speech recognition (Speech Input).

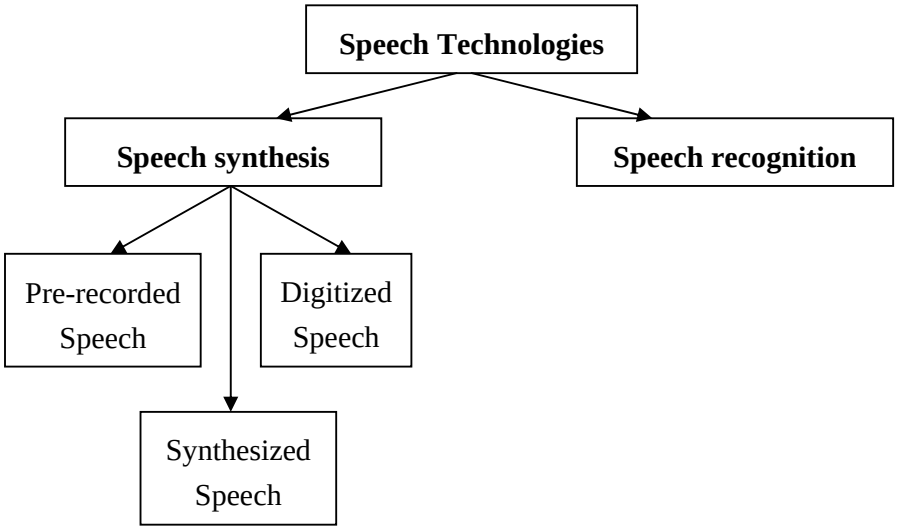


Figure 3. Mainparts of the speech technology

3.2.1. Speech synthesis [2]

Speech synthesis is the process of converting text to speech. Since our target users can not read we can depend on speech synthesis to read the text to them. There are mainly three types of speech synthesis. They are

- **Pre-recorded Speech:** Major limitation: not for spontaneous or dynamic dialogue.
- **Digitized Speech:** Input: entire vocabulary is recorded from a person, which is more natural sounding. To change it, you need a speaker.
  Example: Interactive encyclopaedias.
- **Synthesized Speech:** It is synthesized from text files. Technically challenging to produce naturally sounding speech. The text is easily editable and can be produced dynamically.
  Example: Computer: "What is your name?"
  User: "Mary".
  Computer: "Hello Mary, Welcome!" [4]

3.2.2. Speech recognition [3]

Speech recognition is the process of converting speech to text. Since our target users lack literature, speech recognition is very useful in communicating or sharing information.

Limitations:

---

[2] The generation of an sound waveform of human speech from a textual or phonetic description. [3]

[3] The identification of spoken words by a machine. The spoken words are digitised (turned into sequence of numbers) and matched against coded dictionaries in order to identify the words. [3]

- Speech as input is highly prone to errors.
  For example [13]: Rejection, Substitution, and Insertion errors.
  1. **Rejection error**: It occurs when the recognizer has no hypothesis about what the user said.
  2. **Substitution error**: This involves the recognizer mistaking the user's utterance for a different legal utterance.
  3. **Insertion error**: The recognizer interprets noise as a legal utterance.
- Indicating the expected input is challenging.

3.3. Graphical Designing

Designing icons and visual displays are very important and play a vital role in the success of the whole interface. Since the target users are mostly depending on visual reading, graphical designing becomes one of our main priorities. Unlike audio or speech, bitmaps needs less storage space.

Limitations:

- **Cultural variations**: When designing icons and visual displays keep in mind the cultural barriers. Different images or colours have different significances and meanings in different cultures.
  For example:
  1. The colour ‘Red’ signifies strength, love etc. in some cultures and signifies danger in some other cultures.
  2. A ‘Cat’ in Ancient Egyptian culture is considered as a God and signifies royalty, protection etc. and in some Asian cultures the ‘Cat’ is considered as evil or bad luck.
- **Colour blindness**: People with colour blindness cannot differentiate certain colours like red, green and so on, so while designing, these factors should be taken into consideration.
  1. Do not use the combination of red and green. Use magenta (purple) and green shades instead.
  2. Do not convey information in colour only; try showing difference both in colour and shape (solid and dotted lines, different symbols, various hatching, and so on.)
  3. Avoid using red characters on dark background. Etc.
- **Human psychology**: Some images or colours may seem funny or pleasing to some users but seem offending or irritating to some other users.

3.4. Animations [4]

Animating the images can give an illusion of motion. It is not only interesting to watch the animated images but it also helps in demonstrating the information. It can be simple and very affective in bringing out the information. For example, an animated image demonstrating to the user on how to use and where to click and so on.

Limitations:

- There are many different internet browsers available. For example: Internet Explorer, Netscape, Godzilla and many more and it is not necessary that all the browsers will support all kinds of animations.
- Cultural variations and Human psychology also is a limitation in designing animating images too.
- When there is a java script running in the webpage the animation does not work properly.

4. Existing Technologies

There are already some technologies that people who cannot read or write can use with ease. Listed below are few of the devices that can be referred to, when developing the user interface for illiterate or functional illiterates. These devices have been designed so as, even the people who are illiterate or functional illiterates can learn to use it.

4.1. Smart device

The device called "*Simputer*" (See Figure 4) which is used in India is a low cost portable alternative to PCs, by which the benefits of information technology can reach the common man. It plays a special role in the third world because it ensures that illiteracy is no longer a barrier to handling a computer. The key to bridging the digital divide is to have shared devices that permit truly simple and natural user interfaces based on sight, touch and audio. The Simputer meets these demands through a browser for the Information Markup Language (IML). IML has been created to provide a uniform experience to users and to allow rapid development of solutions on any platform. [10]

[4] The creation of artificial moving images. [3]

Figure 4. Simputer.

The Simputer Text-to-Speech Software aims to ensure that literacy and knowledge of English are not essential for using the Simputer. Using images in conjunction with voice output in local languages makes the Simputer accessible to a larger fraction of the Indian population. [10]

4.2. Wireless phone

There are already many mobile phones which support graphical reading and have voice recognition software for voice dialling, voice recording and so on. And many of these mobile phones are localised too. So the command on English is not a necessity.
For example: Series 40, 60, 80 and 90 platforms in Nokia phones support graphical reading, voice dialling etc. and is localised too. Series 90 platform (See Figure 5) has an inbuilt touch screen.

Figure 5.

But still it can be very challenging to the target people to use these devices because it is not designed for illiterates but for people with literal/computer skills. Thought it may seem simple and is easy to use for computer-literates, it can be very challenging to the target user because we are talking about people with or with out any previous experience using mobile phones or computers. *It is like asking a person who had never even seen an aeroplane before to fly an aeroplane.*

5. Conclusion

Technology today is aiming at new heights. Things that seemed impossible a decade ago are a reality now. New researches are being done on how to improve technology and we are on a verge to know how far our intelligence can stretch and how far technology will lead us. Creating a user interface for illiterates should not be a problem when compared to the accomplishments made by the technology.

User interface for illiterate people will help in wiping away the distance created by literacy/society and help in building a new world, a world of information and knowledge. A new way of accessing information, with a vast source of information that is vital for employment, health, education, environment, life and so on.

There are many technologies and designs that can be used in creating this user interface. For example, the most obvious technologies/designs that I mentioned in this essay, like: Touch screen, speech technologies, graphics and animations, can be used. But there are of course some limitations faced by these technologies, and in this essay I managed to bring out few of them. Knowing the technology and its limitations, help us in taking proper decisions and steps to overcome these limitations. Also we should keep in mind the end users for whom we are designing the interface. Understanding them and their needs are also vital for the development of this user interface.

References

1. [Cambridge Dictionary, 2004] *Cambridge Advanced Learner's Dictionary*, Cambridge University Press.
2. [Deo et al., 2002] Shaleen Deo, Sally Jo Cunningham, David M. Nichols, and Ian H. Witten, Lecture Notes in Computer Science , *Digital Library Access for Illiterate Users*, Department of Computer Science, University of Waikato, Hamilton, New Zealand, ISRN UIUCLIS--2002/4+ISRL, Available: http://www.cs.waikato.ac.nz/~daven/docs/UIUCLIS-2002-4.pdf, Checked October 12, 2004.
3. [Dictionary, 2003] *Dictionary.com* Homepage. http://dictionary.reference.com
4. [Dix et al., 2004] Alan Dix, Janet Finlay, Gregory D. Abowd and Russell Beale. *Human-Computer Interaction*, Third edition, 2004.
5. [Edwards and Kimble, 2004] Alistair D N Edwards and Chris Kimble, Department of Computer Science, University of York, UK. *Workshop on Human-Machine Interfaces for Illiterate Users.*
6. [FOI, 2003] Swedish defence research agency, *High Precision Touch Screen Interaction.* Available: http://www.mind.foi.se/touch/

7. [Goetze and Strothotte, 2001] Marcel Goetze and Thomas Strothotte, An approach to help functionally illiterate people with graphical reading aids. Presented in the First Interational Symposium on Smart Graphics, Hawthorne, New York, 2001. Available:http://www.dfki.de/~krueger/sg2001/schedule/goetze.pdf

8. [Huenerfauth, 2002] Matthew Paul Huenerfauth, *Design Approaches for Developing User-Interfaces Accessible to Illiterate Users*.

9. [Media Lab Europe, 2001-2004] European Research Partner of MIT Media Lab, *Adaptive Speech Interfaces, Designing Interfaces for Illiterate Users.* Available: http://www.medialabeurope.org/asi/index.html Checked October 11, 2004.

10. [Rheingold,1987] Howard Rheingold**,** *Visual thinking - using drawings to communicate ideas*. Available: www.findarticles.com/p/articles/mi_m1510/is_n57/ ai_6203521 Checked October 10, 2004

11. [Simputer Trust, 2001] *Simputer* Homepage. http://www.simputer.org/, Checked October 10, 2004.

12. [WLC, 2004] World Literacy of Canada, *Facts & Figures.* Available: http://www.worldlit.ca/facts.html

13. [Yankelovich and Levow, 2003] Nicole Yankelovich and Gina-Anne Levow. Computing out loud. *The challenge of recognition errors.* Available: http://www.out-loud.com/speechacts.html

Tietämyksen hankinta ja esittäminen tietämyskannoissa

Katja Moilanen

Tiivistelmä

Tässä kirjallisuuskatsauksessa käsitellään tietämyskantoja tietämyksen hankinnan ja esittämisen näkökulmasta. Tietämyksen hankinnassa esitellään tietämyksen hankintaa helpottavia käyttöliittymiä, ihmismassoja tietämyksen tuottajina sekä käyttäjän toimintaa seuraamalla tietämystä hankkivia käyttöliittymiä. Tietämyksen esittämistavoista esitellään säännöt, kehykset, semanttiset verkot ja neuroverkot.

Avainsanat ja -sanonnat: tietämyskannat, tietämyksen hankinta, tietämyksen esittäminen.
CR-luokat: I.2 , I.2.4 , I.2.6

1. Johdanto

Tietämyskantoihin on tallennettu tietämystä siten, että tietämyksestä voidaan tehdä päätelmiä. Ne ovat hyvin moninainen joukko niin sisältönsä kuin sen hankinnan, esittämistavan ja käytön suhteen. Tässä kirjallisuuskatsauksessa tarkastellaan tietämyskantoja erityisesti tietämyksen hankinnan ja tietämyksen esittämisen näkökulmasta.

Tietämyksen hankintaan ja esittämiseen on olemassa lukuisia menetelmiä. Tässä kirjallisuuskatsauksessa esitellään tietämyksen hankintatapoja, joissa käyttäjät tai käyttöliittymä toimivat tietämyksen tallentajina. Tietämyksen esittämistavoista käsitellään sääntöjä, kehyksiä, semanttisia verkkoja sekä neuroverkkoja. Tämä esitys ei pyri olemaan kaiken kattava kuvaus tietämyksen hankinta- ja esitystavoista. Pyrkimyksenä on ollut käsitellä hankinta- ja esitystavat mahdollisimman selkeästi.

Luvussa 2 määritellään tietämyskannat. Luvussa 3 tutustutaan erilaisiin tietämyksen hankintatapoihin. Lopuksi luvussa 4 keskitytään tietämyksen esittämistapoihin.

2. Mikä on tietämyskanta?

Tietämyskannoille on vaikeaa antaa kattavaa määritelmää. Kaikille tietämyskannoille on kuitenkin yhteistä, että niihin on tallennettu tietämystä sellaiseen muotoon, että tallennetun tietämyksen avulla voidaan tehdä päätelmiä. Tietämys, joka tallennetaan tietämyskantaan, voi olla hyvin monentyyppistä laki-

tekstitietämyksestä hahmontunnistustietämykseen. Tietämyskantoja voidaan siis käyttää hyvin moneen tarkoitukseen.

Tietämys, jota kantaan halutaan tallentaa, ohjaa sitä, kuinka tietämystä kannattaa hankkia. Tietämyksenhankintatapoja on monenlaisia asiantuntijoiden haastatteluista koneoppimiseen. Erilaisia tietämyksenhankintatapoja voidaan luonnollisesti käyttää myös yhdessä. Tietämyksen luonne vaikuttaa myös siihen millaisessa muodossa tietämystä tallennettaan kantaan. Tietämyksen esittämis- eli tallennustapoja on lukuisia erilaisia. Esimerkiksi Hayes-Roth mainitsee 12 erilaista tietämyksen esitystapaa jo vuonna 1994: *säännöt* (rules), *kehykset* (frames), *luokat* (classes), *hierarkiat* (hierarchies), *attribuutit* (attributes), *väitteet* (propositions), *rajoitteet* (constraints), *demonit* (demons), *proseduurit* (procedures), *varmuuskertoimet* (certainty factors), *todennäköisyysintervallit* (possibility intervals) ja *sumeat muuttujat* (fuzzy variables). Yhdessä tietämyskannassa voidaan käyttää useita tietämyksen esittämistapoja, jolloin kysymyksessä on hybriditietämyskanta.

Tietämyksen esittämistapa vaikuttaa osaltaan siihen, kuinka tietämyskannasta voidaan tehdä päätelmiä. Esimerkiksi sääntöpohjainen tietämyskanta vaatii erillisen *päättelykoneeksi* (inference engine) kutsutun ohjelman, jonka avulla tietämyskannan tietämyksestä voidaan tehdä päätelmiä esimerkiksi *eteenpäinketjuttamiseksi* (forward chaining) kutsutulla menetelmällä. Päättelemisen toteutustapojakin on hyvin erilaisia, ja niiden valintaan vaikuttaa millaista tietämystä kantaan on tallennettu, kuinka tietämystä on tallennettu ja millaisia päätelmiä kannasta halutaan tuottaa.

Tietämyskanta mahdollisine päättelykoneineen voidaan ohjelmoida erityyppisillä ohjelmointikielillä. Toteutuskielenä voi yhtä hyvin olla oliopohjainen Java [esim. Parpola, 2004], logiikkapohjainen Prolog [esim. Sergot *et al.*, 1986] tai proseduraalinen Basic [esim. Fisher, 1997]. Tämä lisää vielä tietämyskantojen kokonaiskuvan hajanaisuutta.

Tietämyskannat ovat siis hyvin hajanainen joukko hyvin monenlaista eri tavoin esitettyä tietämystä sisältäviä kantoja, joissa tietämyksestä päättely tapahtuu hyvin erilaisin menetelmin. Kaikille tietämyskannoilla on kuitenkin yhteistä se, että niihin on tallennettu tietämystä siten, että tietämyksestä voidaan tehdä päätelmiä.

3. Tietämyksen hankinta

Tietämyksen hankintaa pidetään yleisesti tietämyskantojen yleistymisen esteenä [mm. Richardson and Domingos, 2003; Shipman and McCall, 1994; Clark *et al.*, 2001] Tunnettu, mutta hyvin kallis ja raskas, tietämyksen hankintamenetelmä on se, että *tietämysinsinööri* (knowledge engineer) kyselee tarvittua tie-

tämystä asiantuntijalta ja sitten formalisoi sen valittuun tietämyksen esittämismuotoon. Tietämyksen hankinnan raskautta on yritetty keventää siten, että asiantuntijoille on annettu mahdollisuus toimia tietämysinsinöörinä. Tällöin tietämyksen saannin vaikeutena voi myös olla se, että tietämyskantaa käyttävien asiantuntijoiden on ollut liian vaikeaa syöttää tietämystään kantaan ilman erityisen tietämysinsinöörin apua [Shipman and McCall, 1994; Clark *et al.*, 2001].

Tietämyksen hankinnan ja formalisoinnin ongelmiin on yritetty löytää monenlaisia ratkaisuja. Ratkaisuyrityksiä on sekä ihmislähtöisiä että konelähtöisiä. Ihmislähtöisissä ratkaisuissa ihmiset toimivat tietämyksen tuottajina mahdollisesti tietokoneohjelman avustuksella. Konelähtöisissä ratkaisuissa ohjelma hankkii tietämystä ihmisen määrittelemän mallin mukaan. Usein tietämyksenhankintamenetelmät ovat kuitenkin sekoitus niin ihmisten aktiivista toimintaa kuin koneoppimista. Seuraavaksi tutustutaan tietämyksen syöttämisessä avustaviin käyttöliittymiin, ihmismassojen ja koneoppimisen käytön yhdistelmään sekä tietämystä käyttäjiä tarkkailemalla hankkiviin käyttöliittymiin.

3.1. Asiantuntijoita avustavat käyttöliittymät

Tietämyksen hankinnan pullonkaulana on pidetty sitä, että tietämyksen hankinnassa on tarvittu sekä tietämysinsinöörejä että asiantuntijoita. Tietämysinsinöörien tarvetta on pyritty vähentämään sillä, että asiantuntijoille on annettu mahdollisuus formalisoida itse tietämyksensä. Shipmanin ja McCallin [1994] mukaan tietämyksen formalisointi koneen ymmärtämään muotoon voi olla asiantuntijalle vaikeaa, koska asiantuntijan tietämys ei välttämättä ole riittävän jäsentynyttä ja koska formalisointi vaati tietämystä tietämyksen esittämisen tavasta. Formalisointi vaati myös asiantuntijalta perusteiden ymmärrystä ohjelmointikielestä, jolla formalisointi toteutetaan. Formalisointiin liittyvien ongelmien vuoksi asiantuntijat usein kieltäytyvät käyttämästä järjestelmiä, jotka vaativat formalisointia [Shipman and McCall, 1994].

Ratkaisuna formalisoinnin vaikeuksiin Shipman ja McCall [1994] ovat kehittäneet formalisoinnissa avustavan järjestelmän. Avustavassa järjestelmässä käyttäjä syöttää formalisoimatonta tietämystä järjestelmälle, ja järjestelmä auttaa käyttäjiä formalisoimaan tietämyksen. Tietämyksen formalisointi tapahtuu asteittain, joten käyttäjän ei tarvitse pystyä yhdellä kertaa formalisoimaan tietämystään. Myös formalisoidun tietämyksen muuttaminen jälkikäteen on mahdollista. Kun formalisointi on valmis, tietämyskanta säilyttää myös formalisoimattoman tietämyksen, sillä formalisointi usein hävittää osan tietämystä. Käyttäjä saa tukea formalisointiinsa tietämyskantaan aiemmin formalisoidun tietämyksen kautta, sillä järjestelmä osaa ehdottaa formalisointia aiemmin formalisoidun tietämyksen perusteella. Järjestelmä löytää ehdotuksensa teksteissä ole-

vien kaavojen perustella. Käyttäjä voi hyväksyä ehdotuksen, muokata sitä tai kieltää sen toteuttamisen. Käyttäjällä on myös mahdollisuus kysyä selitystä, miksi järjestelmä antoi ehdotuksen. [Shipman and McCall, 1994] Asteittaisen formalisoinnin järjestelmässä käyttäjä siis suorittaa formalisoinnin, mutta järjestelmä auttaa siinä ehdotusten kautta. Shipmanin ja McCallin [1994] kokemuksen mukaan asteittainen formalisointi voi auttaa käyttäjää oppimaan lisää sekä huomaamaan formalisoinnissa tapahtuneita virheitä.

Clarkin ja muiden [2001] ratkaisuyritys asiantuntijoiden suorittaman formalisoinnin ongelmiin on erilainen kuin Shipmanin ja McCallin. Clarkin ja muiden [2001] esittämä tietämyksen hankintatapa perustuu siihen, että käyttäjällä on käytössään valmiiksi laadittu käsitekirjasto. Käsitekirjaston käsitteet on valittu niin, että ne ovat yleisiä ja kuvattavissa tavanomaisin englanninkielen sanoin. Käsitekirjaston käsitteet on myös määritelty mahdollisimman yleisesti, jotta käsitekirjasto olisi mahdollisimman uudelleenkäytettävä eikä se olisi alariippuvainen. [Barker *et al.*, 2001] Clarkin ja muiden [2001] järjestelmä toimii siten, että käyttäjä hakee käsitteen käsitekirjastosta. Käsite tulee käyttäjän muokattavaksi graafin muodossa ja graafisen käyttöliittymän kautta. Käyttäjä voi täysin vapaasti muokata käsitegraafeja lisäten ja poistaen siitä solmuja. Käyttäjä voi myös rakentaa uusia käsitteitä käsitekirjaston käsitteiden avulla. [Clark *et al.*, 2001] Tietämyksen lisääminen ja muokkaaminen tapahtuu siis graafien muokkaamisen kautta. Graafien muokkaaminen ei vaadi käyttäjältä tietoa siitä, kuinka tietämys konkreettisesti tallennetaan tietämyskantaan. Se vaatii kuitenkin kykyä hahmottaa käsitteitä ja käsitteiden suhteita verkostomaisena rakenteena, jollainen graafi on.

Clarkin ja muiden [1994] kokemuksen mukaan tietämyskantaa käyttävät asiantuntijat pystyivät lisäämään käsitteitä odotettua nopeammin tämän käyttöliittymän kautta. Käyttäjät olivat luoneet jopa 100 solmua sisältäviä graafeja. Vaikka käyttäjät olivat havainneet puutteita systeemissä, he olivat mielestään saaneet kuitenkin syötettyä tämän järjestelmän avulla alansa ydintiedot tietämyskantaa. Käyttäjät olivat myös testanneet tietämyskantaa ja tuloksena oli, että kyselyiden tulokset olivat pääosin oikein. [Clark *et al.*, 1994]

Tietämyksen formalisoinnissa avustavat käyttöliittymät ovat osoittautuneet hyväksi ratkaisuksi tietämyksen hankinnan ongelmiin. Kun käyttöliittymä avustaa formalisoinnissa, ei käyttäjän tarvitse tietää kuinka tietämys konkreettisesti tallennetaan. Käyttäjän tarvitsee vain osata käyttää käyttöliittymää.

3.2. Ihmismassat asiantuntijoina ja tietämyksen formalisoijina

Yhtenä tietämyksen hankinnan mahdollisuutena on nähty suurten ihmismassojen osallistuminen tietämyksen tuottamiseen. Tällöin niin sanotusti tavalliset ihmiset voivat toimia asiantuntijoina ja tietämyksen tuottajina.

Richardson ja Domingos [2003] esittävät uudenlaisen tavan käyttää suurten ihmismassojen ja koneoppimisen yhdistelmää tietämyskannan kokoamiseen. Heidän mallissaan tietämyskanta toimii Internetissä, joka mahdollistaa sen, että tietämyksen lisääminen tietämyskantaan on avointa kaikille, jotka sitä haluavat tehdä. Tietämys syötetään tietämyskantaan Hornin lauseiden muodossa. [Richardson ja Domingos, 2003] Tietämyksen syöttäminen Hornin lauseiden muodossa kuitenkin rajoittaa osallistumista tietämyksen syöttämiseen. Hornin lauseet eivät kuulu yleissivistykseen, jolloin henkilöt, jotka voisivat hyvinkin toimia tietämyksen syöttäjinä, voivat rajautua ulos tietämyksen syöttämismuodon vuoksi. Richardson ja Domingos [2003] ovat tietoisia Hornin lauseiden ongelmasta, sillä heidän mukaansa tietämyksen syöttäminen voisi yhtä hyvin tapahtua graafisen käyttöliittymän kautta.

Richardsonin ja Domingoksen [2003] mallissa myös käyttäjillä on aktiivinen rooli tietämyskannan muodostamisessa. Käyttäjät syöttävät tietämyskannalle sekä kyselyitä että tietämystä, joiden perusteella kyselyyn voidaan saada vastauksia. Tietämyskanta antaa käyttäjälle vastauksia sekä tietämyskannan tietämyksen, kyselyn että käyttäjän syöttämän tietämyksen perusteella. Tämän jälkeen käyttäjä antaa palautetta tietämyskannasta saamistaan vastauksista. Käyttäjän antama palaute välitetään sitten henkilölle, joka oli syöttänyt käyttäjän saaman tietämyksen tietämyskantaan. Myös tietämyskanta oppii koneoppimisen kautta käyttäjän palautteesta. Jos käyttäjät antavat saamastaan vastauksesta huonoa palautetta, ei tätä vastausta enää annetta käyttäjille. Richardson ja Domingos ovat vasta hieman testanneet esittämäänsä ihmisjoukkojen ja koneoppisen yhdistelmää tietämyskannan rakentamisessa. Heidän mielestään tulokset ovat olleet rohkaisevia. [Richardson and Domingos, 2003]

Ihmismassojen käyttäminen tietämyksen tuottajana on varmasti kannattava lähtökohta tietämyksen hankintaan. Erityisesti nyt kun Internetin käyttö alkaa olla maailmanlaajuisesti levinnyt ilmiö, voisi kaikille avoin tietämyskanta tulla toimivaksi. Jokaisella ihmisellä on tietämystä jostakin asiasta, ja jos ihmismassat saataisiin innostumaan tietämyskannan käytöstä, voisi tietämystä kerääntyä aina kengännauhojen solmimisesta ydinfysiikkaan.

3.3. Käyttöliittymä tietämyksen hankkijana

Ohjelman käyttöliittymä voi toimia aktiivisena tietämyksen hankkijana parantaen tallentamansa tietämyksen avulla ohjelman toimintaa. Kyseessä on siis käyttöliittymäagentin kautta tapahtuvasta koneoppimisesta. Pattie Maes [1998] esittää neljä erilaista tapaa, kuinka agentti voi hankkia tietämystä: 1) agentti voi pitää kirjaa käyttäjän kaikista toiminnoista ja siten löytää toimintakaavoja 2) agentti voi saada käyttäjältä suoraa (negatiivinen ohje) tai epäsuoraa (käytöksen muutos) palautetta 3) agentti voi saada tietämystä käyttäjältä suorien esi-

merkkien avulla 4) agentti voi pyytää toisten agenttien tietämystä asiaan josta se ei itse tiedä.

Kozierokin ja Maesin [1993] kokousten ajoittamisohjelmassa käyttöliittymäagentti toimii aktiivisena oppijana keräten tietämystä. Käyttöliittymäagenttiin on ennen käyttöönottoa tallennettu perustietämystä kokouksien ajoittamisesta. Agentti tallentaa kaiken, mitä käyttäjä tekee tilanne-toiminta –pareina ja vertaa käyttäjän toimintaa jo muistissa oleviin tilanne-toiminta –pareihin. Agentti tallentaa myös tärkeyspainoja niin kokouksen teeman avainsanoille kuin kokouksen muille osallistujille. Agentti käyttää oppimiaan asioita siihen, että se pyrkii ehdottamaan käyttäjälle, kuinka agentti toimisi missäkin tilanteessa. Käyttäjä voi hyväksyä tai olla hyväksymättä agentin ehdotuksia, ja agentti oppii tästä käyttäjän antamasta palautteesta. Agentti oppii pikkuhiljaa käyttäjän toimintamallit, ja oppimisen edetessä sen toimintaehdotukset käyvät virheettömimmiksi. Jos ohjelman käyttäjä alkaa luottamaan agenttiin, hän voi antaa agentille esimerkiksi oikeuden ehdottaa kokousaikaa. [Kozierokin and Maes, 1993] Kozierokin ja Maesin käyttöliittymäagentti käytti siis kahta erityyppistä tietämyksen hankintatapaa: käyttäjän tarkkailua ja käyttäjältä saatua palautetta.

Käyttöliittymä voi olla myös useiden käyttäjien yhteiskäytössä. Tällöin käyttöliittymä tallentaa kaikkien käyttäjien käyttäytymistä ja toimii sitten tallentamansa tietämyksen pohjalta. Warnerin ja muiden [2001] tekemä RightNowWeb (RNW) WWW-sovellus käyttää tietämyksen keräämisessä sekä käyttäjien tarkkailua että käyttäjiltä saatua palautetta. RNW sisältää useita Usein Kysyttyjä Kysymyksiä (FAQ) eri aloilta. Verkkosivu kerää kaikista kävijöistä tietoa siitä missä FAQ:issa he vierailivat. Sovelluksessa on oletuksena, että ihminen toimii rationaalisesti ja etsii FAQ:sta tietoa tavoitteellisesti. [Warner *et al.*, 2001] RNW käyttää siis ihmismassojen tietämystä hyväkseen tietämyskannan muodostamisessa. RNW-sivustolla käyttäjien ei kuitenkaan tarvitse itse syöttää tietämystään vaan tietämystä tallennetaan sivulla kävijöiden toiminnasta.

Kaikki Usein Kysytyt Kysymykset, joissa kävijä vierailee saavat lisäpisteitä ja kävijän viimeiseksi vierailema FAQ saa korotetut pisteet, sillä käyttäjän etsimän informaation oletetaan löytyneen sieltä. Käyttäjillä on myös mahdollista antaa eksplisiittisesti palautetta käyttämänsä FAQ:n hyödyllisyydestä. Sekä implisiittisesti että eksplisiittisesti kerätyistä tiedoista lasketaan sitten kunkin FAQ:n hyödyllisyys, jonka mukaan FAQ laitetaan näkyville WWW-sivuille. Käyttäjän vierailemista linkeistä muodostetaan myös käyntikartta, johon kirjataan linkistä toiseen siirtymiset. Käyntikartoista saatua tietämystä käytetään siihen, että niistä luodaan linkkilista samankaltaisiin vastauksiin. Käyttäjäsessioiden tarkkailemisen kautta kerätyn tietämyksen avulla järjestetyt FAQ ovat

vähentäneet eri firmoille lähetettyjä henkilökohtaisia kysymyksiä 10-99,5%. [Warner *et al.*, 2001] Tutkimusryhmän oletukset ihmisten tavoitteellisesta toiminnasta ja siitä, että käyttäjän viimeiseksi lukema sivu on ollut hyödyllisin, vaikuttavat siis osuneen oikeaan.

Käyttöliittymät voivat siis kerätä tietämystä käyttäjiensä toiminnasta ja muokata siten toimintaansa. Yksittäisistä toimintakaavoista muodostuu vähitellen tietämystä, kun samantyyppiset toimintakaavat toistuvat. Myös käyttäjien antamalla suoralla palautteella on merkittävä rooli, sillä sen kautta saadaan tietoa siitä, kuinka oikeaa käyttöliittymän muodostama tietämys on.

4. Tietämyksen esittäminen

Tietämyskantoihin tallennettava tietämys voidaan formalisoida useaan eri muotoon eli *tietämyksen esitystapaan* (knowledge representation). Koska tietämyksen esitystavat ovat malleja, on niistä olemassa erilaisia määritelmiä ja tulkintoja. Seuraavaksi esitellään tulkintoja ja määritelmiä säännöistä, kehyksistä, semanttisista verkoista ja neuroverkoista.

4.1. Säännöt

Säännöt (rules, production rules) ovat tietämyksen esittämistapa, joissa tietämys on kuvattu **Jos → niin (If → then**) rakenteen avulla. Säännöt ovat helposti ymmärrettävä tapa esittää tietämystä, sillä ihmisen arkinen päättely on helppo mieltää päättelyiksi sääntöjen avulla. Esimerkiksi sadepäivänä tapahtuvan päättelyn voisi hyvin ajatella olevan sääntöjen mukaista: **Jos** ulkona sataa, **niin** otan sateenvarjon ulos mukaan.

Youngin [1987] mukaan sääntöjä on kahdenlaista muotoa 1) ehto/ehdot → johtopäätös 2) ehto/ehdot → toiminta/toiminnat. Lehmann ja Magidor [1992, p. 5] määrittelevät muodon ehto/ehdot → johtopäätös sääntötietämyskannan (jota he kutsuvat ehdolliseksi tietämyskannaksi) seuraavasti: on kaksi *kaavaa* (formulas) a ja b. Kaavat muodostavat parin, jossa kaavasta a voidaan tehdä johtopäätös b. Tällaista kaavaparia he kutsuvat *ehdolliseksi väitteeksi* (conditional assertion). Sääntötietämyskanta muodostuu joukosta edellä luonnehdittuja ehdollisia väitteitä. [Lehmann and Magidor, 1992, p. 5]

Ehdollinen tietämyskanta voisi esimerkiksi muodostua seuraavista säännöistä:

Jos X on nisäkäs, **niin** X on lämminverinen.

Jos X on kissa, **niin** X on nisäkäs.

Jos X on Karvinen, **niin** X on kissa.

Säännöt voivat olla myös muotoa ehto/ehdot → toiminta/toiminnat. Tämä esittämistapa tarkoittaa sitä, että säännön **Jos**-osassa ovat ehdot ja **niin**-osassa

ovat toiminnat. **Jos**-osan ehtojen toteutuessa suoritetaan **niin**-osan toiminnot, jotka tavalliset muuttavat vallitsevaa tilannetta. **Jos**-osan toteutuminen siis *laukaisee* (fires) säännön. Tietämyskantaa, joka muodostuu tätä muotoa olevista säännöistä, kutsutaan produktiotietämyskannaksi. [Young, 1987]

Syksyistä toimintaa kuvaava sääntökanta voisi sisältää esimerkiksi tällaisia sääntöjä:

Jos on syksy ja perunat pellossa, **niin** nosta perunat ja laita säkkeihin

Jos perunat ovat säkeissä ja säkit ulkona, **niin** vie perunasäkit kellariin

Jos perunasäkkejä on kellarissa ja rahat vähissä, **niin** vie perunasäkki myyntiin jne.

Sääntötietämyskanta muodostuu siis joukosta sääntöjä, jossa **Jos**-osan ehtojen toteutuessa tehdään **niin**-osan johtopäätös tai suoritetaan **niin**-osan toiminnot.

Fikesin ja Kehlerin [1985] mukaan säännöt sopivat hyvin käyttäytymistä esittävän tietämyksen esittämiseen, mutta ovat sopimattomia staattisten suhteiden ja objektien esittämiseen. Ratkaisuna sääntöjen sopimattomuuteen suhteiden ja objektien esittämisessä he pitävät sääntöjen ja seuraavaksi esiteltävien kehyksien yhdistämistä hybriditietämyskannaksi.

4.2. Kehykset

Marvin Minsky esitteli kehykset tietorakenteena jo vuonna 1974. Hänen teoriansa oli, että ihmisen ajatusrakenteet ovat kehysten kaltaisia. Minskyn [1974] mukaan kehys kuvastaa hyvin yleistettyä tilannetta, oletustilannetta. Kehys on siis ikään kuin stereotypia asiasta, jota se esittää.

Kehys-järjestelmä muodostu solmujen ja suhteiden verkosta, jossa ylemmät kerrokset sisältävät ominaisuuksien oletusmäärittelyjä ja alemmissa kerroksissa on ominaisuuksissa *aukkoja* (slots), jotka täytetään joko ylemmän kerroksen oletusmäärittelyillä tai alempaan kerrokseen liittyvällä spesifimmällä tietämyksellä. Jokaisella ominaisuudella voi olla ehtoja, jotka määritellään joko arvoilla tai alakehyksillä. [Minsky, 1974] Ylemmässä kerroksessa voisi olla esimerkiksi kehys kukka, jossa olisi oletustietämyksenä, että kukalla on juuret, varsi, lehdet, terälehdet, heteet ja emi. Alemmassa kerroksessa voisi olla esimerkiksi ruusu, joka saisi kukka-kehykseltä sen oletustiedot ja sillä olisi omana ominaisuutena, että ruusulla on piikit.

Minskyn [1974] idean mukaan oletustiedot pitää voida helposti korjata uudella paremmin tilanteeseen sopivalla tiedolla. Edelliseen kukka-ruusu -esimerkkiin lisäten: jos olisi olemassa piikitön ruusulajike, niin ruusun oletusfakta ”on piikit” olisi pystyttävä korvaamaan piikittömien ruusujen kohdalla määrittelyllä ”ei ole piikkejä”. Kehykset tietämyksen esittämistapana mahdollistavat

siis periyttämisen oletustietojen avulla ja erikoistamisen oletustietojen korvaamisen avulla.

Fikesin ja Kehlerin [1985] mukaan kehys-esitystavassa jokainen kuvattava subjekti esitetään kehyksen avulla. Jokaisella kehyksellä on attribuutteja eli ominaisuuksia, jotka kuvataan aukkojen avulla. Kehykset voidaan järjestää taksonomioiksi käyttämällä jäsenyys- ja alaluokkalinkkejä. Linkkien kautta kehys saa itselleen ominaisuudet niiltä kehyksiltä, joiden jäsen tai alaluokka kehys on. Kehysjärjestelmään voidaan myös liittää toiminnallisia ominaisuuksia (mm. olio-ohjelmointia tai arvojen dynaamista laskentaa), vaikka ne eivät varsinaisesti kehysesitystapaan kuulukaan [Fikes and Kehler, 1985].

Reimer ja Schek [1989] pyrkivät määrittelemään kehykset mahdollisimman yleispätevästi, kaikkia kehysesitystapoja kuvaten. Heidän mukaansa erilaisia kehyksiä on kolmenlaisia: 1) prototyypit, 2) instanssit ja 3) erikoistuneet prototyypit. Prototyyppi on käsitteen kuvaus eli se kertoo, minkä tyyppisiä ominaisuuksia eli attribuuttiaukkoja käsitteellä on. Prototyypin ominaisuuksilla ei kuitenkaan ole arvoja. Instanssi taas on prototyypin ilmentymä, jossa ominaisuudet saavat arvot. Prototyypin ja instanssin välinen suhde on is-instance-of –suhde. [Reymer and Schek, 1989] Jos prototyyppinä on esimerkiksi koira, jolla on ominaisuudet säkäkorkeus, karvanpituus ja koiraluokka, voisi instanssina olla koira nimeltä Timi, jonka säkäkorkeus on 56 cm, karvanpituus pitkä ja koiraluokka palvelukoira.

Reimerin ja Schekin [1989] mukaan käsitteiden erikoistaminen tapahtuu prototyyppien kautta. Prototyypistä tehdään toinen, erikoistunut prototyyppi lisäämällä siihen ominaisuus ja/tai lisäämällä rajoituksia, jonkin ominaisuuden luvallisiin arvoihin. Prototyypin ja erikoistuneen prototyypin välillä on is-a -suhde. [Reimer and Schek, 1989] Koiraesimerkissä voitaisiin erikoistaminen rajoituksia lisäämällä tehdä siten, että luotaisiin erikoistunut prototyyppi palvelukoira, jonka koiraluokan ainoa sallittu arvo olisi palvelukoira. Attribuuttilisäyksen avulla voitaisiin luoda myös prototyyppi metsästyskoirasta lisäämällä attribuutti ”ajetut eläimet”.

Kaikilla kehyksillä on ominaisuuksia. Ominaisuuksille määritellään sallitut arvot joko eksplisiittisesti, jolloin ominaisuus ei ole rakenteinen tai implisiittisesti jolloin ominaisuus on rakenteinen. Rakenteinen ominaisuus määritellään prototyyppikehyksellä, jolla on instansseja. [Reimer and Schek, 1989]

Minskyn alkuperäinen idea kehyksistä vaikuttaa siis säilyneen kehysmallin pohjana. Sekä Fikesin ja Kehlerin että Reimerin ja Schekin kuvaukset kehyksistä vaikuttavat noudattavan pitkälti Minskyn vuonna 1974 esittämää linjaa.

4.3. Semanttinen verkko

Semanttinen verkko on tietämyksen esittämistapa, jossa tietämys kuvataan suunnatun graafin avulla.

Shastrin [1989] mukaan semanttinen verkko koostuu 1) käsitteistä, 2) käsitteiden ominaisuuksista, jotka voivat olla rakenteisia, ja 3) käsitteiden tai käsitteiden ja ominaisuuksien välisistä hierarkkisista suhteista. Semanttinen verkko on siis graafi, jonka jokainen solmu kuvaa käsitettä tai käsitteen ominaisuutta ja solmujen väliset linkit kuvaavat suhteita. Shastrin [1989] yksinkertaistetussa semanttisen verkon mallissa ainoa suhdetyyppi on olla-suhde (is-a), joka mahdollistaa hierarkiat ja ominaisuuksien perimisen ylä- ja alaluokkien kautta. Klassinen lintuesimerkki ylä- ja alaluokkasuhteesta voidaan esittää vaikka näin: 1) varis on lintu, 2) variksella on ominaisuus musta-harmaa väritys ja 3) linnulla on ominaisuus siivet. Näistä seuraa, että variksella on ominaisuudet 1) musta-harmaa väritys suoraan ja 2) siivet perinnän kautta

Griffith [1982] esittää semanttisista verkoista hyvin kompleksisen ja formaalin määrittelyn, joka ei kuitenkaan ole ristiriidassa edellä esitetyn Shastrin määrittelyn kanssa. Griffith määritelmä esitetään seuraavaksi yksinkertaistettuna siten, että virtuaalisia subjekteja ei esitellä ja suhdetyypeistä esitellään vain binääriset suhteet. Grittihin [1982] semanttinen verkko muodostuu subjekteista, kokoelmista, kokoajista, kategorioista, suhteista, nimistä sekä kaikista ”ylimpänä” olevasta kategoria-kokoajasta, jota kutsun tässä selvyyden vuoksi nimellä *pääkategoria*.

Jokainen subjekti on jakamaton ja atominen, ja se on suhteessa johonkin toiseen subjektiin. Subjektiryhmät muodostavat *kokoelmia* (collections). Jokaisella kokoelmalla on *kokoaja* (collector), joka edustaa kokoelmaa. Kokoelman jäsenten ja kokoajan välillä on *jäsenyys-suhde* (is_member_of). Subjekti voi olla jäsenenä useammassa kokoelmassa. Jokainen kokoaja taas kuuluu *kokoelmat-kategoriaan* (collections). Kategoriat ovat kokoelmien erityissovellutuksia. Kategorioiden kokoajat voivat olla jäseniä toisissa kategorioissa. Kategoriat myös sisältävät jäsentensä ominaisuuksia. [Griffith, 1982]

Esimerkiksi jos ajattelemme, että meillä on subjekteina Nissan Almera Tino, Toyota Yaris Verso ja Opel Zafira, niin tämä kokoelma subjekteja kuuluu kategoriaan tila-autot. Tila-autot kuuluvat puolestaan kategoriaan henkilöautot ja henkilöautot kategoriaan autot jne. Kategoria voi siis kuulua toiseen kategoriaan ja kaikki kategoriat kuuluvat pääkategoriaan. Kategorioiden avulla tapahtuu ominaisuuksien perintää. Esimerkin autolla Toyota Yaris Versolla on kaikki tila-autojen, henkilöautojen ja autojen ominaisuudet.

Griffithin [1982] mukaan myös *suhde* (relationship) on kokoaja nimeltään *relaattori* (relator). Binaarisissa suhteissa relaattori kuvaa kokoelmaa järjestettyjä

pareja, joissa järjestys kuvaa suhteen suuntaa [Griffith, 1982]. Subjekteja voisi esimerkiksi olla tyttö ja poika ja subjektien välinen suhde tykätä. Suhde voidaan kirjoittaa (noudattaen Griffithin esitystapaa) muotoon tyttö <tykkää> poika. Näin ollen järjestetty pari (tyttö, poika) on osa tykkää-relaattorin kokoelmaa. Griffithin [1982] mukaan jokainen relaattori-kategoria kuuluu *relaatiokategoriaan* (relation), joka on kokoja relaattoreille, joilla on sama semanttinen merkitys. Jokainen relaatio-kategoria taas kuuluu *relaatiot-yläkategoriaan* (relations), joka puolestaan kuuluu pääkategoriaan. [Griffith, 1982]

Griffithin [1982] mukaan tulee semanttisessa verkossa olla myös nimet-kategoria (names). Nominaalit (nominal) ovat kokoajina subjekteille, joilla on samat nimet. Nominaalit taas puolestaan kuuluvat nimet-kategoriaan. Nominaaleja ovat kaikki nimisanat. [Griffith, 1982] Ajatellaan vaikka nominaalia Sokrates. Sokrates nominaali voi olla kokoajana hyvinkin erilaisille subjekteille tietämysverkossa. Sokrates voi olla esimerkiksi filosofi, EU:n koulutusohjelma tai golfklubi. Sokrates-nominaali kuuluisi myös nimet-kategoriaan kuten muutkin nominaalit.

Semanttinen verkko -esittämistavassa on sekä etuja että haittoja. Se ei ainoastaan esitä tietämystä vaan tarjoaa myös tietämyksenhaulle relevantteja faktoja. Jokainen subjekti esitetään vain kerran verkossa, jolloin siihen liittyvä tietämys on aina yksiselitteisesti saavutettavissa verkon kautta. Subjektiin liittyvästä tietämyksestä iso osa voi kuitenkin olla löydettävissä vain perinnän kautta. Tällöin tietämyksen löytäminen voi vaatia pitkiä etsintöjä verkossa. Lisäksi perinnän kautta saavutettava laaja tietämys subjektista ei välttämättä ole ainoastaan etu, sillä usein subjektista haluttu tietämys on vain osa siihen liittyvästä tietämyksestä. [Allen and Frisch, 1982, p. 25]

Shastri [1989] mainitsee semanttisten verkkojen ongelmaksi poikkeukset ja moniperinnän aiheuttamat ristiriidat. Poikkeus syntyy silloin kun perinnän kautta saatu attribuutin arvo ei vastaa alakäsitteen arvoa. [Shastri, 1989] Esimerkkinä poikkeuksesta voisi toimia nisäkkäisiin kuuluvat valaat. Jos nisäkkäät olisi tietämyskannassa määritelty siten, että nisäkkäät elävät maalla, niin meressä elävät valaat olisivat poikkeus. Shastrin [1989] mukaan tämä ongelma on kuitenkin ratkaistavissa sillä, että hierarkiassa alempana olevan käsitteen attribuutit korvaavat yläkäsitteiden attribuutit näiden ollessa samat. Moniperintäongelmilla Shastri [1989] tarkoittaa sitä, että perinnän kautta käsitteen attribuutit voivat saada ristiriitaisia arvoja ilman että ristiriita voidaan ratkaista hierarkian avulla kuten poikkeuksissa. Ratkaisuksi moniperintäongelmiin hän esittää, että käsitteen ominaisuudet saisivat todennäköisyysarvoja niiden esiintymisestä alakäsitteissä.

4.4. Neuroverkot

Rumelhartin ja muiden [1994] ja Gallantin [1988, pp. 152-153] mukaan neuroverkot koostuvat itsenäisistä prosessointiyksiköistä, *soluista* (neuron, cell). Solut ovat yhteydessä toisiin soluihin suunnattujen kaarien kautta. Suunnattuja kaaria on jokaisella solulla kahdenlaisia: on *sisääntulokaaria* (inputs) ja *ulosmenokaaria* (outputs). [Rumelhart *et al.*, 1994; Gallant, 1988, pp. 152-153] Rumelhartin ja muiden [1994] mukaan sisääntulokaaret vastaavat dendriitteja ja synapseja ja ulosmenokaaret aksoneja. Neuroverkot ovat siis sääntöihin, kehyksiin ja semanttisiin verkkoihin nähden hyvin erilainen tapa esittää tietämystä. Säännöt ym. pyrkivät jäljittelemään ihmisten symbolisella tasolla tapahtuvaa ajattelua, kun taas neuroverkko pyrkii jäljittelemään ihmisaivoja.

Rumelhartin ja muiden [1994] ja Gallantin [1988, pp. 152-153] mukaan jokaisella solun sisääntulokaarella on paino. Paino voi olla positiivinen tai negatiivinen. Positiivinen paino toimii kiihdyttävänä ja negatiivinen paino estävänä. Jokaisella solulla on *syötearvo* (output value) ja *aktivaatioarvo* (activation value), jotka lasketaan dynaamisesti. Gallantin [1988] mukaan aktivaatioarvo lasketaan kaikkien soluun yhteydessä olevien sisääntulokaarien painolla painotettujen syötearvojen summana. Rumelhartin ja muiden [1994] mukaan tähän summaan lisätään vielä bias-arvo, joka kuvaa solun sisäistä *lepotasoa* (resting level). Syötearvo eli arvo, jonka solu syöttää eteenpäin ulosmenokaariensa kautta, voidaan joko laskea funktiolla, jonka osana on solun aktivaatioarvo [Rumelhart *et al.*, 1994] tai se voi olla sama kuin solun aktivaatioarvo [Gallant, 1988, pp. 152-153]. Neuroverkoissa kaikki tietämys on numeerista; ne ovat siis hyvin matemaattinen tietämyksen esittämistapa.

Sekä Rumelhart ja muut [1994] että Gallant [1988, p. 152-153] kuvaavat tyypilliseksi neuroverkon arkkitehtuuriksi neuroverkkoa, jossa on kolmenlaisia soluja: 1) *syötesoluja* (input cells/units) 2) *välisoluja* (intermediate cells/hidden units) ja 3) *vastesoluja* (output cells/units). Syötesolut eivät laske itse syötearvoaan, vaan niiden arvot annetaan ulkoa [Gallant 1988, pp. 152-153]. Syötesolut ovat yhteydessä välisoluihin, joita voi olla useissa eri kerroksissa. Lopulta osa välisoluista on yhteydessä vastesoluihin, jotka antavat neuroverkon laskeman tuloksen ulos. [Rumelhart *et al.*, 1994] Neuroverkot voivat olla syklittömiä (feedforward networks) tai syklillisiä (feedback networks) [Gallant 1988, p. 152-153].

Neuroverkkojen ongelmana pidetään sitä, ettei sen sisältämä tietämys ole sellaisenaan ihmisen ymmärrettävissä. Neuroverkot eivät myöskään tarjoa selitystä sille, kuinka tiettyyn tulokseen päädyttiin. [mm. Setiono *et al.*, 2000; Boz, 2002] Näiden ongelmien vuoksi neuroverkoista on pyritty saamaan tietämystä ulos muotoon, joka on ihmisten paremmin ymmärrettävissä, ja josta voidaan

tuottaa selitys. Esimerkiksi Setiono ja muut [2000] ovat kehittäneet algoritmin, jonka avulla neuroverkkojen tietämyksestä saadaan tuotettua sääntöjä. Boz [2002] puolestaan on tuottanut menetelmän, jossa neuroverkkojen tietämys muunnetaan päätöspuuksi, joka on helposti käännettävissä säännöiksi.

5. Yhteenveto

Tässä kirjallisuuskatsauksessa käsiteltiin tietämyskantoja ja erityisesti tietämyksen hankintaa ja esittämistä. Katsauksen tarkoituksena ei ollut antaa kaiken kattavaa näkemystä tietämyskannoista vaan kertoa muutamista tietämyksen hankinta- ja esittämistavoista.

Tietämyskantoja on monenlaisia. Ne eroavat toisistaan niin tietämyksen hankinnan, esittämisen kuin tietämyksestä päättelemisen suhteen. Kaikille tietämyskannoille on kuitenkin yhteistä se, että niihin on tallennettu tietämystä muodossa, josta voidaan tehdä päätelmiä.

Avustavat käyttöliittymät esiteltiin yhtenä tietämyksen hankinta muotoina. Ne helpottavat sellaisten henkilöiden tietämyksen syöttämistä tietämyskantaan, jotka eivät ole perehtyneitä tietämyksen esittämisen rakenteisiin. Myös uusi idea suurten ihmisjoukkojen ja koneoppimisen yhdistelmän käyttämistä tietämyksen hankinnassa esiteltiin. Lisäksi katsauksessa käsiteltiin käyttäjän toiminnan tarkkailusta ja käyttäjän palautteesta tietämystä muodostavia käyttöliittymiä.

Tietämyksen esittämistavoista käsiteltiin sääntöjä, kehyksiä, semanttisia verkkoja ja neuroverkkoja. Vaikka kyseiset tietämyksen esittämisen tavat ovat malleja, joista olisi mahdollista esiintyä hyvin erilaisia tulkintoja, sisälsivät tässä kirjallisuuskatsauksessa käytetyt lähteet hyvin samanlaisia tulkintoja kustakin esittämistavasta. Säännöt, kehykset ja semanttiset verkot esittävät tietämystä symbolisesti kun taas neuroverkkojen tietämys on numeerista.

Tässä kirjallisuuskatsauksessa ei kuitenkaan perehdytty esiteltyihin tietämyksen hankinta- tai esittämistapoihin syvällisesti. Tietämyksen hankinta- ja esittämistapojen syvällisempi pohdinta olisi tuonut lukijalle paremmat mahdollisuudet sisäistää esitetyt asiat. Sekä tietämyksen hankinta- että esittämistapoja olisi voinut vertailla keskenään esimerkiksi pohtimalla kirjallisuuden avustuksella millaisiin tilanteisiin kukin tapa sopii parhaiten. Kirjallisuuskatsauksessa olisi todennäköisesti kannattanut keskittyä vain joko tietämyksen hankintaan tai esittämiseen, jotta edes toinen niistä olisi tullut kattavammin käsiteltyä.

Tietämyskannat ovat monipuolisuutensa vuoksi erittäin haastava ja mielenkiintoinen teema. Jatkossa olisi mielenkiintoista syventyä tarkemmin johonkin

tietämyksen esittämistapaan ja tehdä tällä esittämistavalla jonkin sovellusalan toteutus.

Viiteluettelo

[Allen and Frisch, 1982] James F. Allen and Alan M. Frisch, What's in a semantic network? In: *Proceedings of the 20th Conference on Association for Computational Linguistics* (1982), 19-27.

[Barker *et al.*, 2001] Ken Barker, Bruce Porter and Peter Clark, A library of generic concepts for composing knowledge bases. In: *Proceedings of the International Conference on Knowledge Capture* (2001), 14-21.

[Boz, 2002] Olcay Boz, Extracting decision trees from trained neural networks. In: *Proceedings of the Eight ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2002), 456-461.

[Clark *et al.*, 2001] Peter Clark, Pat Hayes, Thomas Reichherzer, John Thompson, Ken Barker, Bruce Porter, Vinay Chaudhri, Anders Rodriquez, Jerome Thomere, Sunil Mishra and Yolanda Gil, Knowledge entry as the graphical assembly of components. In: *Proceedings of the International Conference on Knowledge Capture* (2001), 22-29.

[Fikes and Kehler, 1985] Richard Fikes and Tom Kehler, The role of frame-based representation in reasoning. *Communications of the ACM* **28**, 9 (September 1985), 904-920.

[Fisher, 1997] Rodney Fisher, Determination of residence status for taxation law: development of a rule-based expert system. In: *Proceedings of the Sixth International Conference on Artificial Intelligence and Law* (1997), 161-169.

[Gallant, 1988] Stephen I. Gallant, Connectionist expert systems. *Communications of the ACM* **31**, 2 (1988), 152-169.

[Griffith, 1982] Robert L. Griffith, Three principles of representation for semantic networks. *ACM Transactions on Database Systems* **7**, 2 (September 1982), 417-442.

[Maes, 1998] Pattie Maes, Agents that reduce work and information overload. In: Mark T. Maybury and Wolfgang Wahlster, *Readings in Intelligent User Interfaces*. Morgan Kauffman Publishers, 1998, 525-535.

[Minsky, 1974] Marvin Minsky, A framework for representing knowledge, MIT-AI Laboratory Memo 306, June, 1974. http://web.media.mit.edu/~minsky/papers/Frames/frames.html (27.10.2004)

[Lehmann and Magidor, 1992] Daniel Lehmann and Menachem Magidor, What does a conditional knowledge base entail? *Artificial Intelligence* **55**, 1 (May 1992), 1-60.

[Parpola, 2004], Päivikki Parpola, Inference in the SOOKAT object-oriented knowledge acquisition tool. To appear in *Knowledge and Information Systems* (2004).

[Reimer and Schek, 1989] U. Reimer and H.–J. Schek, A frame-based knowledge representation model and its mapping to nested relations. *Data & Knowledge Engineering* **4**, 4 (1989), 321-352.

[Richardson and Domingos, 2003] Matthew Richardson and Pedro Domingos, Building large knowledge bases by mass collaboration. In: *Proceedings of the International Conference on Knowledge Capture (2003)*, 129 – 137.

[Rumelhart *et al.*, 1994] David E. Rumelhart, Bernard Widrow, Michael A. Lehr, The basic ideas in neural networks. *Communications of the ACM* **37**, 3 (1994), 87-92.

[Setiono *et al.*, 2000] Rudy Setiono, Wee Kheng Leow and James Y. L. Thong, Opening the neural network black box: an algorithm for extracting rules from function approximating artificial neural networks. In: *Proceedings of the Twenty First International Conference on Information Systems* (2000), 176-186.

[Sergot *et al.,* 1986] M. J. Sergot, F. Sadri, R. A. Kowalski, F. Kriwaczek, P. Hammond and H. T. Cory, The British nationality act as a logic program. *Communications of the ACM* **29**, 5 (1986), 370-386.

[Shastri, 1989] Lockendra Shastri, Default reasoning in semantic networks : a formalization of recognition and inheretance. *Artificial Intelligence* **39**, 3 (1989), 283-355.

[Shipman and McCall, 1994] Frank M. Shipman, III and Raymond McCall, Supporting knowledge-base evolution with incremental formalization. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (1994), 285-291.

[Warner *et al.,* 2001] Doug Warner, J. Neal Richter, Stephen D. Durbin and Bikramjit Banerjee, Mining user session data to facilitate user interaction with a customer service knowledge base in RightNowWeb. In: *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2001), 467 – 472.

[Young, 1987] Richard M. Young, An introduction to production systems. In: Tim O'Shea, John Self and Glan Thomas (eds.), *Intelligent Knowledge-Based Systems : An Introduction*, Harper & Row, 1987, 68-82.

Lähestymistapoja OLAP-toteutuksiin

Turkka Näppilä

Tiivistelmä

Moniulotteinen analyysi tai OLAP (On-line Analytical Processing) on yleinen päätöksenteon tukijärjestelmien toteutuksissa käytetty tekniikka. OLAP-analyysissä pyritään havaitsemaan toimintayksikön kannalta olennaiset muutokset, löytämään toimintoihin liittyvät trendit tai niiden muutokset sekä vertailemaan eri yksilöitä tai yksikköjä keskenään. OLAP-toiminnallisuuksissa hyödynnetään yhteenvetotietoja, jotka on tavallisesti koottu keskitettyyn, yhteen aihepiiriin liittyvään tietovarastoon. Tietovaraston tiedot on yhdistelty operationaalisista tietokannoista ja muista käytettävissä olevista sähköisistä tiedonlähteistä. OLAP-ratkaisujen tärkein eroavuus perinteisiin operationaalisiin OLTP-tietokantaratkaisuihin nähden on se, että OLAP-toteutukset käsittelevät aiemmista päivitystapahtumista laskettua yhteenvetotietoa, jonka pitää olla tuoretta mutta ei välttämättä reaaliaikaista. Vaikka OLAP-ratkaisuja ja -käsitteistöä on nyt kehitelty varsin pitkään, ei kehitys- ja tutkimusyhteisöissä vallitse OLAP-käsitteistöstä vieläkään täyttä yksimielisyyttä. Tietokuutio on yleisesti käytetty moniulotteinen tietomalli. Tietokuutiossa data on järjestetty erilaisten tekijöiden eli dimensioiden mukaan. OLAP-operaatioiden (roll-up, drill-down, slice, dice, pivot) suorittamisen kautta tietokuutiosta saadaan tuotettua uudenlaisia kokonaisnäkemyksiä, jolloin tietokuutioon talletettua dataa analysoidaan moniulotteisesti. Tietokuutiota ei voida suoraan implementoida tietokannan kaavioksi, vaan tietokantatoteutuksissa tietokuutio esitetään joko moniulotteisten taulukkojen tai relaatiotaulujen avulla.

Avainsanat ja -sanonnat: OLAP, moniulotteinen analysointi, tietokuutio OLAP-operaatiot, ROLAP, MOLAP.
CR-luokat: H.2.1,H.2.4,H.2.7.

1. Johdanto

Moniulotteinen analyysi tai OLAP (On-line Analytical Processing) on käsite, joka on kehitetty kuvaamaan mielenkiinnon kohteena olevien kokonaistietojen analysointia. Codd ja muut [1993], jotka ensimmäisinä esittelivät käsitteen OLAP, painottavat, että perinteiset relationaaliset tietokantajärjestelmät on suunniteltu varmistamaan tietokantatapahtumien (lisäykset, päivitykset ja poistot) mahdollisimman tehokas prosessointi, mutta niiden kyselyominaisuudet ovat varsin rajoitettuja. Esimerkiksi päätöksenteon tukijärjestelmissä annetusta datasta pyritään löytämään suuntauksia tai tunnuslukuja, joiden avulla pystyt-

täisiin arvioimaan nykyisiä asiantiloja tai ennustamaan tulevaa kehitystä. Tällaisessa tilanteessa tapahtumakeskeiset OLTP (On-line Transaction Processing) -tietokantajärjestelmät osoittavat riittämättömyytensä ja tarvitaan OLAP-järjestelmiä, joissa kyselyjen käsittely on tehokkaampaa ja joustavampaa ja jotka tukevat *moniulotteista tiedon analysointia* (multidimensional data analysis). OLAP onkin yleinen päätöksenteon tukijärjestelmien toteutuksessa käytetty tekniikka.

Moniulotteisessa tiedon analysoinnissa hyödynnetään etupäässä yhteenveto- ja koostedataa (summary/aggregated data). Relaatiomalliin perustuvissa tietokannoissa data on tallennettu tauluihin, joissa data on riveillä yksittäistietoina. Yhteenvetojen muodostaminen relaatiotietokantojen datasta edellyttää yksittäisten tietokannan taulujen ja niissä olevien tietojen yhdistämistä näkymiksi. Näkymien muodostaminen ja kyselyjen tekeminen muodostettujen näkymien perusteella on kuitenkin käyttäjän kannalta monimutkaista ja johtaa perinteisillä tietokantajärjestelmillä suoritettuna huonoon vasteaikaan.

Moniulotteisessa analysoinnissa tarvittavat tiedot onkin tyypillisesti tallennettu *tietovarastoon* (data warehouse). Sen tiedot on saatu yhdistelemällä operationaalisissa tietokannoissa ja muissa sähköisissä tietolähteissä (esim. ASCII-tiedostot, XML-dokumentit) olevia tietoja. Tietovarastoon tuotavat tiedot voivat perustua erilaisiin tietomalleihin, jolloin ne tietovarastossa pitää saattaa keskenään yhdenmukaisiksi. Moniulotteisen analyysin toiminnalliset ja suorituskykyyn liittyvät vaatimukset eroavat perinteisille OLTP-tietokantatoteutuksille asetetuista vaatimuksista. Siksi tietovarasto on tyypillisesti muuten erillään operationaalisista tietojärjestelmistä paitsi, että sitä päivitetään jatkuvasti operationaalisista järjestelmistä koottavalla tiedolla.

Koska tietovarastoihin on tallennettu yleensä ajan suhteen jaksotettuja yhteenvetotietoja, jotka on johdettu mahdollisesti useiden operationaalisten tietokantojen tiedoista, on tietovaraston tilantarve tavallisesti huomattavasti suurempi kuin operatiivisen tietokannan: Chaudhurin ja Dayalin [1997] mukaan tyypillisen operationaalisen tietokannan koko vaihtelee sadoista megabiteistä gigabitteihin, kun tietovarastojen kokoa mitataan sadoista gigabiteistä terabitteihin. Toki voi käydä myös niin, että tietojen koostamisen seurauksena tietovaraston tilantarve on vastaavien operationaalisten tietokantojen tilantarvetta pienempi. Tietovarasto voi olla keskitetty tai hajautettu pienemmiksi *paikallisvarastoiksi* (data mart).

Tietovarastoihin tallennettu data on myös luonteeltaan erilaista kuin operatiivisissa tietojärjestelmissä käytettävä data. Chaudhuri ja Dayal [1997] korostavat, että moniulotteisessa tiedon analyysissa monesti tarvitaan sellaista dataa, joka saattaa puuttua operationaalisista tietojärjestelmistä: nykyisten tai men-

neiden asiantilojen arvioiminen ja tulevan kehityksen ennustaminen perustuu nimittäin historiallisen datan analysointiin. Operationaalisissa tietojärjestelmissä sitä vastoin käytetään useimmiten ainoastaan ajantasaista dataa eikä niissä ole tarvetta säilyttää aktiivikäytöstä poistettua dataa. Lisäksi operationaalisissa tietojärjestelmissä oleva data on usein reaaliaikaista, kun taas tietovarastoissa olevaa dataa päivitetään yleensä vain silloin, kun aikadimension alin taso muuttuu.

Tietovaraston kuormitus on kyselypainotteista. Moniulotteisessa analyysissä ovat tyypillisiä ennakoimattomat, kompleksiset kyselyt, jotka perinteisissä tietokantajärjestelmissä saattaisivat vaatia miljoonien tietueiden läpikäyntiä, tietueiden liitoksia ja tietueiden arvojen yhdistelyjä [Chaudhuri and Dayal, 1997]. OLTP-tietokantatoteutuksissa on ollut tärkeää varmistaa tavanomaisten tietokantatapahtumien mahdollisimman sujuva suorittaminen ja siksi moniulotteisessa analyysissä tarvittavien, ja perinteisillä kyselykielillä toteutettujen, kyselyiden ajaminen niissä johtaa helposti heikkoon suorituskykyyn.

Vaikka OLAP ja tietovarasto ovat läheisesti toisiinsa liittyviä käsitteitä, niillä on kuitenkin painotuksellinen ero. Tietovarastossa huolehditaan ensisijaisesti sinne tulevan datan integroinnista ja varastoinnista sekä varmistetaan, että se on eheää ja oikeellista. Cabibbo ja Torlone [1998] painottavat, ettei moniulotteista tiedon analysointia suoriteta suoraan tietovaraston raakadatalle vaan siitä johdetuissa erityisrakenteissa, tietokuutioissa. Tietovarasto tuottaa tietokuution ja ylläpitää sitä, kun taas OLAP-sovellus käsittelee tietovaraston tuottamaa kuutiota luottaen siihen, että se on oikeellinen ja ajantasainen. Moniulotteinen analysointi tapahtuu usein tietovarastosta erotetuilla OLAP-palvelimilla.

Kirjallisuudessa termeillä moniulotteinen (tiedon) analysointi ja OLAP tarkoitetaan suunnilleen samaa. Tässä tutkielmassa kuitenkin termillä OLAP viitataan ensisijaisesti (tekniseen tai formaaliin) ratkaisuun, joka on suunniteltu mahdollistamaan moniulotteinen analysointi. OLAP-toiminnallisuudet voidaan toteuttaa monilla eri tavoilla, mutta – ainakin korkealla tasolla – erilaisilla OLAP-ratkaisulla yhteisiä piirteitä. Colliat [1996] esittää, että OLAP-ratkaisuilta vaaditaan seuraavien perusominaisuuksien täyttymistä:

(1) Analysoitava perusdata on koostettua.

(2) Analysoitava data voi olla käytöstä poistettua, käytössä olevaa tai ennustettua.

(3) Yhteenvetodataa on voitava edelleen koostaa, ja koostetasojen välillä on voitava liikkua interaktiivisesti kaikkiin suuntiin.

(4) Koostetuista tiedoista on voitava johtaa uutta koostettua tietoa.

(5) Analysoitaviin koostetietoihin on voitava muodostaa moniulotteisia näkymiä.

(6) Tiedon ennakoimattoman ja interaktiivisen analysoinnin on oltava riittävän nopeaa.

(7) Analysoitavan koostedatan muodostamiseen on käytetty suuria tietojoukkoja.

(8) Tietomallilla esitettävän todellisuuden sisältö vaihtuu nopeasti.

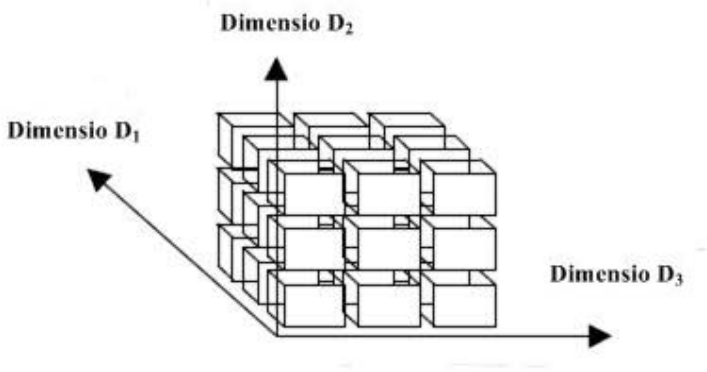
Kohdista 1, 2 ja 7 on keskusteltu edellä moniulotteisen analysoinnin vaatimusten yhteydessä. Kohdat 5 ja 8 liittyvät moniulotteisen tiedon esittämiseen ja niitä käsitellään tarkemmin luvussa 2. Kohtia 3 ja 4 käsitellään luvussa 3, jossa puhutaan moniulotteisen tiedon käsittelystä ja analysointitavoista. Luvussa 4 esitetään vaihtoehtoja moniulotteisen tietomallin toteuttamiseksi tietokantatasolla ja tässä yhteydessä sivutaan myös kohtaa 6. Lopuksi luku 5 sisältää yhteenvedon käsitellyistä asioista.

2. Moniulotteinen tietomalli

Moniulotteinen analysointi edellyttää, että on olemassa looginen tietomalli, jonka avulla moniulotteista tietoa voidaan esittää. Moniulotteinen tiedon analysointi perustuu siihen, että analysoitavaa dataa voidaan tarkastella ja käsitellä erilaisten näkökulmien mukaan jäsennettynä. Datan analysoinnissa käytettävät näkökulmat voivat olla toisistaan riippuvia tai riippumattomia. Näkökulmia kutsutaan moniulotteisessa analysoinnissa ulottuvuuksiksi eli *dimensioiksi*. Perusdata, jota dimensioiden avulla halutaan analysoida, koostuu erilaisista *mitta-arvoista* (measure). Mitta-arvot ovat yleensä numeerista dataa, jota tulkitaan dimensioiden avulla. Dimensiot esitetään dimensioattribuutteina ja mitta-arvot mitta-attribuutteina. Dimensioattribuuttien arvot ovat tyypillisesti merkkijonoja. Ei ole olemassa formaalia tapaa ratkaista, mistä attribuuteista tehdään dimensioattribuutteja ja mistä mitta-attribuutteja, vaan attribuuttien tyyppien valinta on aina riippuvainen käyttökontekstista [Agrawal *et al.*, 1997].

Tietokuutio (data cube) on yleinen tapa visualisoida moniulotteinen avaruus. Tietokuutio on myös käytetyin looginen malli moniulotteisen tiedon esittämiseksi. Kuvassa 1 on esimerkki tietokuutiosta. Tietokuutio on dimensioiden rajoittama avaruus. Esimerkiksi kuvan 1 kuutio on dimensioiden D_1–D_3 rajoittama kolmiulotteinen avaruus. Yksittäiset dimensioattribuutit tai dimensioattribuuttien joukot rajaavat tietokuutiosta pienempiä osa-avaruuksia: tietokuutio toisin sanoen rakentuu pienemmistä tietokuutioista. Toisiinsa liittyvien tietokuutioiden kokoelma muodostaa *moniulotteisen tietokannan* (multidimensional database, MDD) [Pedersen and Jensen, 2001].

Dimensioattribuuttien rajoittamissa tietokuution soluissa on alkioina mitta-arvoja. Tietokuution solu voi olla yksi- tai monipaikkainen, so. yhteen tietokuution soluun voidaan tallettaa yksi tai useampi mitta-arvo; myös tyhjä arvo on mahdollinen. Eri dimensioattribuuttien kombinaatiot määrittävät aina yksikäsitteisesti yhden mitta-attribuutin arvon tietokuutiossa. Toisin sanoen mitta-attribuuttien arvot ovat riippuvaisia dimensioattribuuttien arvoista. Oletetaan, että kuvan 1 dimensiot muodostavat joukot $D_1 = \{v_{1,1}, v_{2,1}, \ldots, v_{n,1}\}$, $D_2 = \{v_{1,2}, v_{2,2}, \ldots, v_{n,2}\}$ja $D_3 = \{v_{1,3}, v_{2,3}, \ldots, v_{n,3}\}$ ja mitta-arvot ovat joukossa $M_1 = \{v_{1,4}, v_{2,4}, \ldots, v_{n,4}\}$. Silloin kolmikko $\langle v_{i,1}, v_{i,2}, v_{i,3} \rangle$ määrittää yksikäsitteisesti jonkin mitta-attribuutin $v_{i,4}$ $(1 \geq i \geq n)$ arvon.



Kuva 1. Tietokuutio.

Dimensioattribuutit voivat muodostaa keskenään hierarkian. Dimensioattribuutin hierarkiataso ilmaisee sen yksityiskohtaisuuden asteen, jolla analysoitavaa dataa voidaan tarkastella. Tyypillinen esimerkki hierarkkisessa suhteessa olevista dimensioattribuuteista on kauppaketjun myynnin tarkastelu yksittäisen liikkeen, paikkakunnan, maakunnan ja koko valtakunnan näkökulmasta. Dimensioattribuutteihin voi liittyä myös sellaisia tietoja, jotka pitää ilmaista eksplisiittisesti mutta joita ei voida esittää suoraan tietokuution attribuutteina [Hirvonen, 2001]. Tällöin kutakin dimensioita varten pitää luoda oma, erillinen tietorakenteensa.

On syytä huomata, että tietokuutio on ainoastaan metafora moniulotteiselle avaruudelle. Moniulotteisessa analyysissa tarvitaan tavallisesti eri määrä dimensioita, kuin mitä tietokuution graafiseen esitykseen sisältyy [Pedersen and Jensen, 2001]. Moniulotteisessa avaruudessa voi olla kaksi tai useampi ulottuvuus. Kaksiulotteisen avaruuden luonnollinen visualisointi on taulu(kko), joka jakautuu riveihin ja sarakkeisiin. Kun ulottuvuuksia on enemmän kuin kolme, puhutaan usein hyperkuutiosta (hypercube) [Marcel, 1999]. Myös useampiulotteinen avaruus visualisoidaan tavallisesti kaksiulotteisten taulukkojen avulla.

Tietokuutiota ei voida myöskään implementoida sellaisenaan tietokannan kaavioksi [Shoshani, 1997], vaan tietokantaratkaisuissa moniulotteinen tietorakenne pitää toteuttaa muilla keinoin. Yleisin tapa esittää moniulotteinen tietorakenne on ilmaista se taulukkorakenteiden avulla.

Moniulotteisen analyysin avulla tuotetaan datasta näkemys, joka heijastaa analysoijan toimintatapoja, sillä analyysissä käytettävät dimensiot edustavat käyttäjän tietotarpeita. Kun käyttäjän tietotarpeet muuttuvat, on myös moniulotteisen tietomallin avulla tuotetun näkemyksen oltava helposti muutettavissa. Moniulotteinen analyysi tarkoittaa käytännössä sitä, että tietokuutiota manipuloimalla siitä saadaan muodostettua uudenlaisia näkymiä. Koska annettu data on yleensä koosteista ja johdettua, aiheutuu siitä moniulotteiseen analyysiin erityispiirteitä, jotka erottavat OLAP-analyysin perinteisten tietokantasovellusten tietojenkäsittelystä.

3. Moniulotteisen tiedon käsittely

Relaatioalgebra määrittää operaatiot, joilla relaatiotietokantoihin tallennettuja tietoja kyetään käsittelemään. Relaatioalgebran operaatioita on kahdenlaisia: klassiset joukko-opilliset operaatiot (unioni, leikkaus, erotus ja tulo) ja erityisesti relaatiotietokantojen käyttötarpeisiin kehitetyt operaatiot [Elmasri and Navathe, 2000] (jotka itse asiassa ovat myös joukko-opillisia). Relaatiotietokanta-operaatioista yleisimmät ovat valinta, projektio ja liitos. Valinta-operaatiossa annetusta relaatiotaulusta poimitaan ne rivit, jotka täyttävät valintaoperaatiossa ilmaistun valintaehdon. Projektiossa puolestaan relaatiotaulusta erotetaan ne sarakkeet, jotka ovat tarkasteltavan asian kannalta olennaisia. Liitoksessa yhdistetään yhdeksi tauluksi relaatiotaulut, jotka toteuttavat annetun liitosehdon.

Koska OLAP on kehittyvä ala, ei OLAP-ratkaisujen tarpeita varten ole vielä olemassa yleisesti hyväksyttyä algebraa. Esimerkiksi Agrawal ja muut [1997], Gyssens ja Lakshmanan [1997], Vassialidis [1998] ja Pedersen ja Jensen [1999] ovat antaneet omat ehdotuksensa OLAP-algebraksi. Codd ja muut hahmottelivat uraauurtavassa työssään [1993] joitakin OLAP-operaatioille luonteenomaisia piirteitä, joiden vaikutus näkyy myös edellä mainituissa algebraehdotuksissa. Ilmeistä on, että OLAP-toiminnallisuuden mahdollistavan algebran tulee olla ilmaisuvoimaltaan relaatioalgebran veroinen ja sisältää lisäksi operaatioita, jotka mahdollistavat joustavan navigoinnin moniulotteisessa avaruudessa.

Osa OLAP-toiminnallisuuden perusoperaatioista on relaatiotietokantaoperaatioiden variantteja, mutta moniulotteiseen analysointiin sisältyy myös sellaisia operaatiota, joilla ei ole relaatioalgebrassa vastineita. Koska OLAP-terminologia ei ole vakiintunutta [Vassiliadis and Sellis, 1997], käytetään kirjal-

lisuudessa OLAP-operaatioista vaihtelevia nimityksiä. OLAP-perusoperaatioiden nimet ovat lähinnä kuvailevia, sillä samalle termille on monesti useita tulkintatapoja. OLAP-operaation suorittaminen merkitsee tietokuutioon tehtävää kyselyä ja usein OLAP-operaatio tuottaa uuden kuution, joka eroaa rakenteellisesti lähtökuutiosta. Tavallisimpia OLAP-operaatioita käytetään tietokuutiossa navigointiin (roll-up ja drill-down), tulosnäkymän muokkaamiseen (slice, dice ja pivot) ja attribuuttien merkityksen vaihtamiseen (push ja pull). Seuraavassa esitellään tyypillisimmät OLAP-operaatiot.

3.1. Tietokuutiossa navigointi

Tietokuutiossa navigointi tarkoittaa liikkumista ylös- tai alaspäin dimensioattribuutien hierarkiassa. Dimensiohierarkiassa ylöspäin siirtymisen mahdollistaa OLAP-operaatio *roll-up*; hierarkiassa alaspäin siirtyminen on puolestaan *drill-down*-operaation soveltamista. Vaihtoehtoisia nimityksiä roll-up-operaatiolle ovat *aggregation* ja *consolidation* ja drill-down-operaatiolle *roll-down* ja *drill-through* [Vassiliadis, 1998]. Käytettävästä loogisen tietomallin toteutuksesta riippuu, voidaanko roll-up- ja drill-down-operaatioita soveltaa ainoastaan dimensioattribuuteille, joille on ennalta määritelty yksi ainoa karkeistushierarkia, vai voidaanko niitä soveltaa myös dimensioattribuuteille, joille ei ole määritelty karkeistushierarkiaa tai joilla on vaihtoehtoisia karkeistushierarkioita. Tietokuutiossa navigoitaessa siitä luodaan näkemys, joka eroaa lähtökuutiosta jonkin dimension merkityksen suhteen. Tietokuutiosta luotava näkymä on itsessään (uusi) tietokuutio.

Roll-up-operaation suorittaminen vähentää tietojen yksityiskohtaisuuden tasoa. Dimensiohierarkiassa ylemmille tasoille siirtyminen on suhteellisen suoraviivainen prosessi: jos esimerkiksi halutaan siirtyä tarkastelemaan kauppakohtaisten myyntilukujen sijasta paikkakuntakohtaisia myyntilukuja, riittää, kun tietyn paikkakunnan kauppojen myyntiluvut kootaan yhteen jatkotoimenpiteitä varten. Intuitiivisesti katsottuna roll-up-operaation suorittaminen merkitsee sitä, että kaikki alemman hierarkiatason dimensioattribuuttien arvot korvataan ylemmän hierarkiatason arvoilla [Vassiliadis, 1998].

Kuten on jo aiemmin todettu, tietokuution soluissa olevat mitta-attribuuttien arvot ovat yleensä koostettua tietoa eli ne on saatu laskemalla yhteen yksittäisten tietoalkioiden arvoja. Moniulotteisessa analyysissä mitta-attribuuttien arvoja tyypillisesti yhdistellään dimensioattribuuttien hierarkiatasojen perusteella edelleen, jolloin saadaan koostettuja eli *aggregoituja arvoja*. Mitta-attribuuttien arvojen yhdistely tarkoittaa käytännössä sitä, että niihin sovelletaan jotakin aggregaatiofunktiota. Tavallisimpia aggregaatiofunktioita käytetään tietokuution alkioiden yhteissumman, keskiarvon ja lukumäärän laskemiseen sekä suurimman ja pienimmän arvon määrittämisen. Mitta-

attribuuttien arvoista voidaan myös tuottaa *johdettuja arvoja*. Tyypillinen johtamisoperaatio on mitta-arvojen keskinäisen suhteen laskeminen. Roll-up- ja drilldown-operaatioiden suorittamiseen sisältyy implisiittisesti aina jonkin aggregaatiofunktion soveltaminen [Pedersen and Jensen, 1999].

Roll-up-operaatioilla tuotettua tulosnäkymää ei voida käyttää hyväksi sitä seuraavassa kyselyssä, jos uudessa kyselyssä nykyisen kuution dimensioita muutetaan (ks. luku 3.2). Tällöin aggregoidut mitta-attribuuttien arvot eivät enää pidä paikkaansa ja roll-up-operaatioilla lähtökuutiosta tuotetut yhteenvetotiedot joudutaan laskemaan uudelleen uuden kyselyn prosessoinnin yhteydessä. Erityisen raskaita ovat sellaiset roll-up-operaatiota käyttävät kyselyt, joissa mitta-arvojen koostaminen tapahtuu dimensiohierarkian alimmalta hierarkian ylimmille tasoille. Tähän vaikuttaa luonnollisesti alimman tason ja kohdetason välinen etäisyys ja kyselyssä käytettävien dimensioiden määrä. Edeltävän kyselyn tulosnäkymää voidaan käyttää sitä seuraavan roll-up-operaatiota käyttävän kyselyn syötteenä ainoastaan silloin, kun jälkimmäisessä kyselyssä tarvittavan tietokuution dimensioattribuuttien arvot ovat samalla hierarkiatasolla kuin aiemman kyselyn tuottaman kuution dimensioattribuuttien arvot [Kanerva, 2003].

Ongelmallisempaa sen sijaan on, kun näkymän yksityiskohtaisuuden tasoa kasvatetaan eli siirrytään nykyistä hierarkiatasoa alemmille tasoille. Ylemmän hierarkiatason mitta-attribuuttien arvot saatiin soveltamalla jotakin aggregaatiofunktiota alemman hierarkiatason mitta-attribuutien arvoille, mutta ylemmällä hierarkiatasolla olevia aggregoituja arvoja ei enää voida purkaa osatekijöikseen ilman, että tiedetään entuudestaan, minkälaisista arvoista alempien hierarkiatasojen attribuutit koostuvat. Esimerkiksi paikkakuntakohtaisista myyntiluvuista ei pystytä triviaalisti päättelemään kauppakohtaisia myyntilukuja.

Drill-down-operaatio voidaan toteuttaa soveltamalla tietokuutioon liitosoperaatiota sellaisen tietokuution kanssa, jonka dimensioattribuutit ovat halutulla hierarkiatasolla. Tietokuutioiden liitos on kuitenkin yleensä raskas operaatio. Vaihtoehtoinen tapa toteuttaa drill-down-operaatio on laskea aggregoidut arvot uudelleen dimensiohierarkian alimmalta tasolta halutulle hierarkiatasolle; tällöin tarkennusoperaation suoritusaikaan vaikuttaa dimensiohierarkian alimman tason ja tavoitellun tarkkuustason välinen etäisyys. Drill-down-operaation suoritusta voidaan nopeuttaa hyödyntämällä materialisoituja näkemyksiä (materialized view), joihin mahdollisia aggregointeja on laskettu ja talletettu etukäteen, ja indeksointirakenteita (ks. esim. [Chaudhuri and Dayal, 1997; Chaudhuri *et al.*, 2001]). Vassiliadis [1998] tehostaa drill-down-operaatiota sisällyttämällä tietokuution formaaliin määritelmään toisen tieto-

kuution, peruskuution (basic cube), jossa on tarvittavalla tarkkuustasolla olevaa tietoa aina nopeasti saatavilla. Peruskuution avulla vältytään suorilta tietokuutioiden liitoksilta.

3.2. Näkymän muuttaminen

Slice- ja *dice*-operaatioita käytetään tietokuution ulottuvuuksien ja sisällön rajaamiseen. Slice- ja dice-operaatioiden suorittaminen muuttaa tietokuutiota rakenteellisesti. Slice-operaatiosta käytetään myös nimitystä *destroy dimension* ja dice-operaatiosta *restriction* [Agrawal *et al.*, 1997], *selection*, *screening* ja *filtering* [Vassiliadis, 1998]. Slice-operaatio on analoginen relaatioalgebran projektio-operaation kanssa. Slice-operaatiossa rajataan tietokuutiosta osakuutio, joka täyttää operaation yhteydessä annetun loogisen tai matemaattisen ehdon. Käytännössä tämä tarkoittaa, että tietokuutiosta poistetaan annetun valintaehdon täyttävä dimensio (kuution ”siivu”), jolloin tietokuution ulottuvuuksia siis vähennetään ja tietokuutiosta saadaan muodostettua uudenlainen näkemys. Jos tietokuutioon sisältyy aggregoituja mitta-attribuuttien arvoja, ne pitää slice-operaation suorittamisen jälkeen laskea uudelleen.

Dice-operaatiolla tietokuutiosta poistetaan kokonaisen dimension sijasta ainoastaan sellaiset dimensioattribuuttien arvot, joiden katsotaan olevan kyselyn kannalta epäolennaisia. Toisin sanoen dice-operaatiolla tuotetaan tietokuutiosta näkemys, jossa on vain käyttäjää kiinnostavaa tietoa. Dice-operaatiota vastaa relaatioalgebrassa valinta-operaatio. Dice-operaatiota voidaan soveltaa sekä dimensio- että mitta-attribuutteihin, jos dimensio- ja mitta-attribuutteja käsitellään symmetrisesti (so. dimensioattribuutista voidaan tehdä mitta-attribuutti ja vastaavasti mitta-attribuutista dimensioattribuutti).

Pivot-operaatiota käytetään organisoimaan tietokuutiossa esitetyt tiedot uudelleen. Toinen nimitys pivot-operaatiolle on *rotate* [Marcel, 1999]. Käytännössä pivot-operaation suorittaminen merkitsee sitä, että joistakin tietokuution dimensioattribuuttien arvoista tehdään kuution dimensioita. Tietokuutiosta tuotettavassa uudessa näkemyksessä korostetaan joitakin dimensiota ja samalla peitetään toisia. Pivot-operaatio voi joissakin tilanteissa lisätä saatavan esityksen selkeyttä mutta ei muuten vaikuta tietokuution attribuuttien arvoihin tai niiden rakenteeseen [Vassiliadis, 1999].

3.3. Attribuuttien merkityksen vaihtaminen

Push- ja *pull*-operaatioiden suorittaminen edellyttää, että käytettävässä loogisessa tietomallissa dimensio- ja mitta-attribuutteja voidaan käsitellä symmetrisesti. Push-operaatiolla dimensioattribuutteja liitetään mitta-arvoihin, jolloin dimensioattribuuteista tulee tietokuution solujen sisältöä. Pull-operaatio on

push-operaatiolle vastakkainen operaatio: sen soveltamisella tuotetaan tietokuutio, jossa on uutena dimensioattribuuttina jokin lähtökuution mitta-arvo.

Useissa OLAP-ratkaisuissa on lisäksi mahdollista suorittaa relaatiomallin perusoperaatiot (ks. esim. [Agrawal *et al.*, 1997; Gyssens and Lakshmanan, 1997; Vassiliadis, 1998; Pedersen and Jensen, 1999]). Moniulotteisen kyselyn tekeminen voidaan nähdä loppukäyttäjän tekemänä, peruskuutiosta etenevänä vaiheistettuna sarjana peräkkäisiä OLAP-operaatioiden suorituksia. Tällainen näkemys OLAP-analyysistä on proseduraalinen. Koska kuitenkin suurin osa OLAP-sovellusten loppukäyttäjistä ei ole ohjelmointitaitoisia [Niemi *et al.*, 2003b], proseduuraalinen tapa suorittaa moniulotteista analysointia sisältäviä kyselyitä ei aina ole loppukäyttäjän kannalta luonnollista ja intuitiivista. Siksi on kehitetty myös deklaratiivisia tapoja suorittaa OLAP-kyselyjä. Gray ja muut [1997] esittelivät *cube*-operaation, jonka avulla on suoraviivaista käsitellä moniulotteisesti esitettyä tietoa. Cube-operaatiossa on taustalla oletus, että yksi ilmaisuvoimainen, relationaalinen operaatio riittää moniulotteisen analyysin suorittamiseen. Cube-operaatio on SQL (Structured Query Language) -kyselykielen group by -operaation yleistys. Niemi ja muut [2003b] ovat puolestaan kehittäneet näkemysorientoituneen lähestymistavan moniulotteisen analysoinnin suorittamiseksi. Kehitetyssä lähestymistavassa loppukäyttäjä ainoastaan määrittää attribuutit, jotka hän haluaa mukaan kyselyllä tuotettavaan näkymään, ja taustalla vaikuttava, logiikkaohjelmointiin perustuva, mekanismi huolehtii OLAP-operaatioiden suorittamisesta.

Edellisissä luvuissa on käsitelty moniulotteisen tiedon esittämistä ja moniulotteisesti esitetyn tiedon käsittelymahdollisuuksia. Seuraavaksi tarkastellaan, miten moniulotteinen tietomalli, tietokuutio, voidaan toteuttaa tietokantatasolla.

4. Moniulotteisen tietomallin toteuttaminen

Moniulotteisen tietomallin toteuttamiseen tietokantatasolla on kolme perusvaihtoehtoa: tietokuution fyysisessä toteutuksessa voidaan hyödyntää relaatiomallin ominaisuuksia (relationaalinen OLAP eli ROLAP) tai tietokuutio voidaan esittää suoraan moniulotteisena taulukkorakenteena (moniulotteinen OLAP eli MOLAP). Risteytysratkaisuissa (hybridi OLAP eli HOLAP) pyritään minimoimaan kummankin edellä mainitun toteutustavan ei-toivotut piirteet. Lisäksi on kehitetty olio-orientoituneita tapoja (ks. esim. [Nguyen *et al.*, 2000]) toteuttaa moniulotteinen tietomalli (olio-orientoitunut OLAP eli OOLAP), mutta ne jäävät tässä tutkielmassa tarkastelun ulkopuolelle.

4.1. MOLAP-toteutukset

MOLAP-järjestelmissä moniulotteinen avaruus linearisoidaan (moniulotteiseksi) taulukoksi [Shoshani, 1997], jonka dimensiot vastaavat tietokuution dimensioita. Tietokuution mitta-attribuutin sijainti moniulotteisessa taulukossa määritellään indeksein. Indeksoinnin avulla dimensioiden erilliset arvot talletetaan taulukkoon vain kertaalleen. Esimerkiksi vastaavissa ROLAP-toteutuksissa samassa relaation visualisoinnin sarakkeessa voivat samat arvot esiintyä useamman kerran. Taulukkolinearisointi toimii hyvin, kun esitettävä moniulotteinen avaruus on tiivis, so. taulukon jokaista solua vastaa jokin mitta-attribuutin arvo eikä ole tyhjiä arvoja. Harvan tietokuution sisällön tiivistämiseksi on kehitetty erilaisia menetelmiä (ks. esim. [Niemi *et al.*, 2003a]).

4.2. ROLAP-toteutukset

ROLAP-järjestelmissä hyödynnetään suoraan relaatiotietokantojen piirteitä ja tallennusrakenteina käytetään relationaalisia tauluja. ROLAP-järjestelmässä on relationaalisen taustapalvelimen lisäksi välittäjäkerros, jossa käyttäjän antaman kyselyn perusteella identifioidaan materialisoidut näkemykset, joita pystytään kyselyssä käyttämään hyväksi, muotoillaan käyttäjän tekemä kysely materialisoitujen näkemysten mukaiseksi ja lopuksi lähetetään kysely palvelimelle [Chauhuri *et al.*, 2001]. ROLAP-järjestelmissä tietokantakyselyt tehdään SQL-tyyppisillä kyselykielillä. Yleisiä ROLAP-järjestelmien toteutustapoja ovat *tähtimalli* (star schema) ja sen muunnelmat *lumihiutale-* (snowflake schema) ja *konstellaatiomalli* (constellation schema), jotka esitellään lähemmin seuraavissa alaluvuissa.

4.2.1. Tähti- ja lumihiutalemalli

Tähtimalli muodostuu yhdestä keskitetystä faktataulusta ja joukosta sen ympärille ryhmitettyjä dimensiotauluja. Faktataulu ja dimensiotaulut yhdistetään toisiinsa relaatiomallin tapaan pää- ja vierasavainten kautta. Faktataulun rivit vastaavat alkuperäisen tietokuution mitta-attribuutteja. Faktataulun sarakkeisiin sijoitetaan mitta-arvot ja ne yksilöivien dimensioattribuuttien arvot, jotka toimivat samalla relaation vierasavaimena. Alkuperäisen tietokuution jokaista dimensiota varten on oma dimensiotaulunsa. Dimensiotaulun sarakkeet koostuvat ominaisuuksista, jotka liittyvät kyseiseen dimensioattribuuttiin.

Dimensiotaulut ovat tähtimallissa normalisoimattomia tai ensimmäisessä normaalimuodossa, mikä tarkoittaa, että niissä esiintyy redundanssia. Redundanssi aiheutuu siitä, että dimensiohierarkioita ei tähtimallissa ilmaista eksplisiittisesti, vaan dimensiohierarkian eri tasojen attribuutit ovat samassa dimensiotaulussa. Dimensiot vievät kuitenkin yleensä vain hyvin pienen osan tietokuution vaatimasta levytilasta, joten redundanssista ei aiheudu tilankäytöllises-

ti suurta haittaa [Pedersen and Jensen, 2001]. Koska dimensiohierarkian eri tasojen tiedot on talletettu samaan tauluun, eivät dimensioiden päivitykset vaikuta datan yhdenmukaisuuteen. Tähtimallin etu on yksinkertainen ja helposti käsiteltävä rakenne, joka antaa mahdollisuuden tehokkaasti prosessoitavien kyselyjen yksinkertaiseen muotoilemiseen. Tähtimallissa haittapuolena on, että dimensiohierarkia ei ole selvästi nähtävissä.

Lumihiutalemalli on normalisoitu versio tähtimallista. Lumihiutalemallissa on jokaista dimensiohierarkian tasoa kohti oma dimensiotaulunsa, joka on rakenteeltaan tähtimallin dimensiotaulun kaltainen paitsi, että alemman hierarkiatason dimensiotaulussa on vierasavaimena sitä välittömästi ylemmän hierarkiatason dimensiotaulun pääavain. Vaihtoehtoiset hierarkiatasot on lumihiutalemallissa helppo ilmaista. Dimensiohierarkioiden määrästä ja syvyydestä riippuen lumihiutalemallin mukaisessa tietokannassa on tähtimalliin verrattuna suurempi määrä tauluja, mikä voi joissakin tapauksissa olla haittaava tekijä. Lumihiutalemallissa dimensiohierarkiat ovat kuitenkin selvästi nähtävissä.

4.2.2. Konstellaatiomalli

Niemi ja muut [2003b] näkevät, että kompleksiset moniulotteiset sovellusalueet sisältävät usein dataa, joka liittyy eri käyttökonteksteihin. Siksi toisinaan on selkeämpää tehdä jokaista kontekstia kohden oma faktataulu kuin sisällyttää kaikki dimensioattribuutit samaan tauluun. Tällöin faktataulussa on ainoastaan ne dimensioattribuutit, jotka ovat kaikille taulun mitta-arvoille yhteisiä. Yleisesti mallia, jossa on dimensiotaulujen lisäksi useita faktatauluja, kutsutaan *konstellaatiomalliksi* [Chaudhuri and Dayal, 1997]. Faktataulut voivat jakaa keskenään samoja dimensiotauluja. Konstellaatiomallissa dimensiohierarkiat voidaan esittää implisiittisesti (tähtimallin tapaan) tai eksplisiittisesti (kuten lumihiutalemallissa). Tähti- ja lumihiutalemalliin verrattuna konstellaatiomallissa on uutta se, että myös faktataulut voivat olla keskenään hierarkkisessa suhteessa. Konstellaatiomallin avulla pystytään tuottamaan tähti- ja lumihiutalemallia ilmaisuvoimaisempi kuvaus moniulotteisesta avaruudesta.

Niemi ja muut [2003b] esittelevät konstellaatiomallin variantin, joka poikkeaa standardimallista siinä, että faktataulussa voi olla dimensioita, joihin ei liity dimensiotaulua ja dimensiohierarkian tasot ilmaistaan erillisessä hierarkiataulussa. Faktatauluihin, joita mallissa kutsutaan tietokuutioiksi, ei liity hierarkiatauluja. Dimensiotaulut linkittyvät keskenään niissä esiintyvien yhteisten arvojen kautta. Faktatauluissa oleva data on aina dimensiohierarkian alimman tason mukaista. Esitetty konstellaatiomallin muunnelma on antanut Niemelle ja muille mahdollisuuden kehittää ilmaisuvoimaisen ja käyttäjäystävällisen OLAP-kyselykielen.

On syytä huomata, etteivät edellä esitellyt ROLAP-tietokannan toteutustavat ole pelkästään fyysisiä tietomalleja. Kyseisissä malleissa ei nimittäin sinällään oteta kantaa siihen, miten ne fyysisesti tietokantajärjestelmässä implementoidaan. Tähti-, lumihiutale- ja konstellaatiomalleja voidaan siten pitää myös moniulotteisina loogisina tietomalleina.

4.3. MOLAP- ja ROLAP-toteutusten vertailu

MOLAP-järjestelmissä datasta tuotetaan suoraan moniulotteinen näkemys, kun taas ROLAP-järjestelmät tarjoavat lähinnä moniulotteisen käyttöliittymän relaatiomallilla esitettyyn dataan [Vassiliadis and Sellis, 1997]. ROLAP-järjestelmä on kuitenkin MOLAP-ratkaisuihin verrattuna helpompi integroida useimpiin olemassa oleviin tietojärjestelmiin. Datan taltioinnissa ROLAP-järjestelmät pystyvät hyödyntämään relaatiotietokantojen erinomaisia tallennusrakenteita. Lisäksi ROLAP-järjestelmät skaalautuvat hyvin, jos tallennettavan datan määrä kasvaa. ROLAP-järjestelmissä dataa on lisäksi helpompi päivittää kuin MOLAP-järjestelmissä [Pedersen and Jensen, 2001]. ROLAP-järjestelmien negatiivisena puolena on välittömien moniulotteisten tallennusrakenteiden puute ja kyselykielenä käytettävä SQL. OLAP-kyselytarpeiden ja SQL-kyselykielen perimmäiset eroavuudet voivat aiheuttaa yhteensopimattomuustilanteita tai kyselyiden pitkiä vasteaikoja, mutta toisaalta useat nykyiset relaatiotietokantapalvelimet tukevat SQL-kyselykielen laajennoksia, joissa on otettu huomioon OLAP-toiminnallisuuden vaatimuksia.

MOLAP on puhdas moniulotteinen palvelinarkkitehtuuri, joka ei hyödynnä relaatiotietokantojen piirteitä vaan käyttää suoraan moniulotteista tallennusmekanismia. MOLAP-järjestelmissä moniulotteiset kyselyt on mahdollista ulottaa suoraan tallennusrakenteeseen ilman välittävää kerrosta. MOLAP-järjestelmien etuna ovat niiden erinomaiset indeksointimekanismit, jolloin moniulotteinen tieto on mahdollista tallettaa siten, että se on tarvittaessa nopeasti haettavissa. Haittapuolena MOLAP-järjestelmissä on niiden tehoton tilankäyttö – etenkin jos kuutio on harva. Vaikka MOLAP-järjestelmät takaavat moniulotteisen analysoinnin kannalta hyvän suorituskyvyn ja toiminnallisuuden, ne eivät pysty skaalautumaan riittävästi, jos dataa on hyvin suuri määrä. MOLAP-järjestelmät toimivat parhaiten, kun data on kohtuullisen tiivistä, kun taas ROLAP-järjestelmät yltävät parempaan suorituskykyyn, kun data on erittäin harvaa [Pedersen and Jensen, 2001].

5. Yhteenveto

Moniulotteinen analysointi eroaa perinteisissä relaatiotietokannoissa tapahtuvasta tietojenkäsittelystä: OLAP-analysointi on kyselypainotteista ja ei-

rutiininomaista. Analysoinnissa käytettävä data on tyypillisesti ajan suhteen jaksotettua yhteenvetodataa. Moniulotteisessa analysoinnissa data on järjestetty tietokuutioksi, josta muodostetaan näkemyksiä. Tietokuutiosta tuotettuja näkemyksiä manipuloidaan analysoinnin edetessä suorittamalla OLAP-operaatioita. OLAP-operaatioilla vaikutetaan saatavan näkemyksen yksityiskohtaisuuteen tai tietokuution rakenteeseen. OLAP-kyselykielet poikkeavat toisistaan sen suhteen, tarvitseeko käyttäjän spesifioida kyselyssä käytettävät operaatiot vai ei.

Tietokantatasolla moniulotteinen tietomalli yleensä toteutetaan joko ROLAP- tai MOLAP-arkkitehtuurissa. ROLAP-arkkitehtuuri hyödyntää relaatiomallin ominaisuuksia ja tietokuutio esitetään siinä relaatiotaulujen muodossa. ROLAP-toteutuksessa relaatiot voivat olla normalisoimattomia (tähtimalli) tai normalisoituja (lumihiutalemalli). MOLAP-arkkitehtuuri on puhdas moniulotteinen tietokantatoteutus. MOLAP-arkkitehtuurissa data sijoitetaan moniulotteiseen taulukkoon, jonne se tallennetaan indeksien avulla. ROLAP-toteutuksissa data pystytään tallentamaan tehokkaasti; MOLAP-toteutuksien analysointikyky on puolestaan ROLAP-toteutuksiin verrattuna suurempi.

Viiteluettelo

[Agrawal *et al.*, 1997] Rakesh Agrawal, Ashish Gupta, and Sunita Sarawagi. Modeling multidimensional databases. In A. Gray, P.-Å. Larson (Eds.): *Proceedings of the 13th International Conference on Data Engineering, Birmingham, U.K., April 7–11, 1997*. IEEE Computer Society, 1997, 232–243.

[Cabibbo and Torlone, 1998] Luca Cabibbo and Riccardo Torlone. A logical approach to multidimensional databases. In H.-J. Schek, F. Saltor, I. Ramos, G. Alonso (Eds.): *Advances in Database Technology – Proceedings of the 6th International Conference on Extending Database Technology, Valencia, Spain, March 23–28, 1998. Lecture Notes in Computer Science* **1377** (1998), Springer, 183–197.

[Chaudhuri and Dayal, 1997] Surajit Chaudhuri and Umeshwar Dayal. An overview of data warehousing and OLAP technology. *ACM SIGMOD Record* **26**, 1 (March 1997), 65–74.

[Chaudhuri *et al.*, 2001] Surajit Chaudhuri, Umeshwar Dayal, and Venkatesh Ganti. Database technology for decision support systems. *Computer* **34**, 12 (December 2001), 48–55.

[Codd *et al.*, 1993] E. F. Codd, Sharon B. Codd, and Clynch T. Salley. *Providing OLAP (on-line analytical processing) to user-analysts: An IT mandate.* Technical report. E. F. Codd Associates, 1993.

[Colliat, 1996] George Colliat. OLAP, relational, and multidimensional database systems. *ACM SIGMOD Record* **25**, 3 (September 1996), 64–69.

[Elmasri and Navathe, 2000] Ramez Elmasri and Shamkant B. Navathe. *Fundamentals of Database Systems (Third Edition)*. Addison-Wesley, 2000.

[Gray *et al.*, 1997] Jim Gray, Surijit Chaudhuri, Adam Bosworth, Andrew Layman, D. Reichart, M. Venkatrao, F. Pellow, and Hamid Pirahesh. Data cube: A relational aggregation operator generalizing group-by, cross-tab, and sub-totals. *Data Mining and Knowledge Discovery* **1**, 1 (March 1997), 29–54.

[Gyssens and Lakshmanan, 1997] Marc Gyssens and Laks V. S. Lakshmanan. A foundation for multi-dimensional databases. In M. Jarke, M. J. Carey, K. R. Dittrich, F. H. Lochovsky, P. Loucopoulos, M. A. Jeusfeld (Eds.): *Proceedings of the 23rd International Conference on Very Large Data Bases, Athens, Greece, August 25–29, 1997*. Morgan Kaufmann, 1997, 106–115.

[Hirvonen, 2001] Lasse Hirvonen. *Helppokäyttöisen OLAP -kyselykielen suunnittelu ja toteutus*. Pro gradu -tutkielma. Tampereen yliopisto, tietojenkäsittelytieteiden laitos, maaliskuu 2001. 62 sivua. Saatavana: *http://www.cs.uta.fi/opiskelu/tutkielmat/pro_gradut.html*.

[Kanerva, 2003] Kaarlo Kanerva. *Lähestymistapoja OLAP -kieliin*. Pro gradu -tutkielma. Tampereen yliopisto, tietojenkäsittelytieteiden laitos, lokakuu 2003. 81 sivua. Saatavana: *http://www.cs.uta.fiopiskelu/tutkielmat/pro_gradut.html*.

[Marcel, 1999] Patrick Marcel. Modeling and querying multidimensional databases: An overview. *Networking and Information Systems Journal* **2**, 5–6 (December 1999), 515–548.

[Nguyen *et al.*, 2000] Thanh Binh Nguyen, A Min Tjoa, and Roland Wagner. An Object Oriented Multidimensional Data Model for OLAP. In H. Lu, A. Zhou (Eds.): *Proceedings of the 1st International Conference on Web-Age Information Management, Shanghai, China, June 21–23, 2000. Lecture Notes in Computer Science* **1846** (2000), Springer, 83–94.

[Niemi *et al.*, 2003a] Tapio Niemi, Jyrki Nummenmaa, and Peter Thanisch. Normalising OLAP cubes for controlling sparsity. *Data & Knowledge Engineering* **46**, 3 (September 2003), 317–343.

[Niemi *et al.*, 2003b] Timo Niemi, Lasse Hirvonen, and Kalervo Järvelin. Multidimensional Data Model and Query Language for Informetrics. *Journal of the American Society for Information Science and Technology* **54**, 10 (August 2003), 939–951.

[Pedersen and Jensen, 1999] Torben Bach Pedersen and Cristian S. Jensen. Multidimensional data modeling for complex data. In: *Proceedings of the 15th International Conference on Data Engineering, Sydney, Australia, March 23–26, 1999*, 336–345.

[Pedersen and Jensen, 2001] Torben Bach Pedersen and Cristian S. Jensen. Multidimensional database technology. *Computer* **34**, 12 (December 2001), 40–46.

[Shoshani, 1997] Arie Shoshani. OLAP and statistical databases: Similarities and differences. In: *Proceedings of the 16th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, Tucson, Arizona, May 12–14, 1997*, 185–196.

[Vassiliadis, 1998] Panos Vassiliadis. Modeling multidimensional databases, cubes and cube operations. In: *Proceedings of the 10th International Conference on Scientific and Statistical Data Management, Capri, Italy, April 1–3, 1998*, 53–62.

[Vassiliadis and Sellis, 1999] Panos Vassiliadis and Timos Sellis. A survey of logical models for OLAP databases. *ACM SIGMOD Record* **28**, 4 (December 1999), 64–69.

XML-skeema tiedonsiirtovälineenä

Riina Pakarinen

Tiivistelmä.
Tutkielmassa käsitellään informaation lisäämistä siirrettävään dataan xml-teknologian avulla. Lisäksi esitellään xml-skeeman syntaksi ja verrataan sitä dtd-syntaksiin. Tutkielmassa esitellään myös kritiikkiä ja arviointiperusteita xml-teknologian käyttöön. Lopuksi käsitellään skeeman käyttöä sähköisissä palveluissa ja sitä hyödyntäviä xml-viestien hallintaohjelmia.

Avainsanat ja -sanonnat: xml, xml-skeema
CR-luokat: D.3.2

1. Johdanto

Automaattinen tiedon välitys on olennainen osa yritysten sähköistä liiketoimintaa. Tässä tutkielmassa tutkitaan, voisiko tiedon välityksessä siirtyvää dataa parantaa siten, että sen käsittely olisi tehokkaampaa. Ratkaisuna esitellään datan jalostus informaatiota sisältäväksi tiedoksi. Lisäksi käsitellään jatkumoa yksinkertaisesta datasta monimutkaisempiin rakenteisiin.

Tutkimuksen toisessa osassa esitellään xml-skeema ratkaisuna yksinkertaisen datan kehittämiseen. Xml-skeeman ominaisuuksia ja syntaksia tarkastellaan tarkemmin kolmannessa luvussa. Tämän jälkeen esitellään xml-skeemaan pohjautuvan ratkaisun kritiikkiä ja arviointiperusteita. Lopuksi käsitellään esimerkkinä xml-skeeman hyödyntämistä sähköisissä palveluissa.

2. Rakenteinen data viestin välittäjänä

Jotta liiketoiminta olisi mahdollista eri osapuolten välillä, niiden täytyy myytävän tuotteen tai palvelun lisäksi välittää toisilleen myös oheistietoa, esimerkiksi tuotehinnasto tai lasku. Sitä voidaan siirtää erimerkiksi puhelimitse ja sähköpostitse. Jos siirrettävä tieto on toistuvaa, se kannattaa automatisoida siirtymään sähköisesti. Tällöin tietoa voidaan siirtää kaupan osapuolten välillä ohjelmalta toiselle.

Niiniluoto [1996] jakaa tiedon käsitteen kolmeen osaan: dataan, informaatioon ja tietoon. Datalla tarkoitetaan merkkijonoja ilman merkitystä. Informaatio muodostuu kun dataan liitetään tulkinta. Tieto ja tietämys syntyvät puolestaan silloin, kun informaatio muuttaa ihmisen kognitiivisia rakenteita. Yritysten välillä sähköisesti siirtyvä tieto on dataa. Tietoa voidaan siirtää esimerkiksi *csv-muodossa* (comma separated values, pilkulla erotetut arvot). Jos

tiedoston lukijalla ei ole tiedossa kenttien merkityksiä, tiedoston sisältö on vain merkkijonoja vailla tulkintaa. Sekä siirrettävän tiedoston kirjoittavalla että sitä lukevalla ohjelmalla täytyy olla tiedossa malli, jonka mukaan tiedosto muodostuu. Kun siirrettävää tiedostoa tulkitaan mallin avulla, siitä tulee informaatiota.

Viestin lähettävä osapuoli muodostaa siirrettävän viestin keräämällä omasta tietokannastaan tarvittavat tiedot siirrettävään viestiin. Viestin vastaanottaja puolestaan purkaa ensin viestin ja käyttää saatua tietoa edelleen. Viestin muodostaminen ja purkaminen tapahtuvat jokaisella kerralla kun viestiä välitetään. Jos esimerkiksi yritys saa kaikki tilauksensa tällä tavalla, viestin välitys muodostaa merkittävän osan tiedonsiirrosta. Olisi siis hyödyllistä löytää keino, jolla viestin välitystä voitaisiin tehostaa. Viestin muodostamisen ja purkamisen tehokkuus riippuu paljolti itse viestin mallista.

Daconta ja muut [2003] tarjoavat ratkaisuna viestinvälityksen tehostamiseen informaation lisäämistä siirrettävässä datassa. Tällöin viestin purkaminen sujuisi nopeammin, koska osa datan tulkinnasta olisi jo mukana itse viestissä. He käyttävät tästä käsitettä *fiksu data* (smart data). Tavoitteena on muuttaa viestin sisältämä data viestin sisältämäksi informaatioksi. Tällöin viestissä on siirrettävän datan lisäksi myös tulkintaa helpottavia osia, jotka tekevät viestistä ikään kuin fiksumman. Tulkintaa auttavia osia lisätä esimerkiksi muuttamalla siirrettävän datan rakennetta. [Daconta *et al.,* 2003]

Daconta ja muut esittelevät siirrettävän tiedon jatkumon. Matalimmalla tasolla ovat teksti ja tietokannat. Molemmat ovat sellaisenaan dataa. Vasta silloin kun esimerkiksi tietokantaa käytetään ohjelmassa, sen sisältämälle tiedolle löytyy merkitys. Jatkumon seuraavilla tasolla on mukana xml-kieli (Extensible Markup Language). Nimeämällä siirrettävän datan osat kuvaavilla nimillä tiedostoon sisältyy enemmän informaatiota kuin pelkkää dataa sisältävään tiedostoon. Ensimmäinen xml-kieltä hyödyntävä taso kuvaa yhden toimialueen sisällä käytössä olevaa mallia. Tällöin käytössä on rajattu sanasto. Seuraavalla tasolla sanasto laajentuu hierarkkiseen taksonomiaan. Tällöin eri kategorioissa olevien käsitteiden välille voidaan luoda viittaussuhteita. Korkeimmalla xml-kieltä hyödyntävällä tasolla mukaan tulee ontologia ja säännöt. Tässä tasossa siirrettävä tieto niin rakentunutta, että siitä voidaan loogisten sääntöjen avulla muodostaa uutta tietoa.

Sitä mukaa kun data muuttuu rakenteiseksi eli Dacontan ja muiden [2003] mukaan fiksummaksi, sen luettavuus ihmiselle heikkenee. Näin toteaa Lawrence [2004], joka esittää, että cvs-tiedostosta on helppoa etsiä tarvittava tieto, kun tietää mitä saraketta tulee katsoa Sen sijaan xml-dokumentista, jossa ei ole erottimia tagien välillä, ei ole yhtä helppo löytää tarvitsemaansa tietoa. Joten kun siirrettävästä datasta tehdään siirrettävää informaatiota, se ei välttämättä

tarkoita, että siirrettävä tiedosto sellaisenaan olisi ihmiselle helpompi käsitellä. Voi käydä niin, että sitä mukaan kun tiedosto muuttuu ohjelmalle helpommin työstettäväksi, sen lukeminen ihmiselle hankaloituu.

Daconta ja muut [2003] määrittelevät semanttisen verkon fiksun datan verkoksi, jota voidaan koneellisesti prosessoida. Prosessoitavan datan täytyy olla sovellusriippumatonta ja luokiteltavaa. Jos ohjelmat käyttävät vain tietokantaa, niiden käyttöön tarvittava tieto tai paremminkin säännöt sisältyvät niitä käyttävään ohjelmaan, eivät itse tietokantaan. Ohjelmien 'viisaus', toisin sanoen niiden sisältämät säännöt ja käsitteiden suhteet olisi hyödyllistä pitää mukana, kun muodostetaan siirrettäviä viestejä yritykseltä tai ohjelmalta toiselle. Käsitteiden muodostamaa verkkoa on myös helpompi uudelleen käyttää kuin ohjelmakomponentteja. [Daconta *et al.,* 2003] Kun yritys alkaa esimerkiksi vastaanottaa sähköisiä laskuja uudelta osapuolelta, on helpompaa tehdä muutoksia käsitemalleihin, jotka sisältävät säännöt niiden jatkokäsittelystä, kuin itse ohjelmiin. Tällöin ohjelmaan ei välttämättä tarvita monia muutoksia, kun ohjelman sisäänlukema tiedosto sisältää jo tarvittavan informaation.

3. Xml-skeema

Xml-kieltä on esitetty ratkaisuksi siirrettävän datan rakenteisuuden ja sitä kautta sen sisältämän informaation lisäämiseen. Xml-kieli on World Wide Web -konsortio (W3C) kehittämä kieli datan rakenteen kuvaukseen. Se koostuu sekä itse datasta että sen tulkintamallista. Malliksi voidaan antaa ainoastaan dokumentti kenttien merkityksestä, mutta tällöin siitä ei saada kaikkea hyötyä irti. Mallista on eniten hyötyä, jos se on toteutettu sitä varten muodostetuilla kuvailukielellä. Näitä ovat esimerkiksi dtd (Document Type Definition) ja xdr (External Data Representation). Näiden mallien avulla xml-dokumentti voidaan validoida automaattisesti. Validoimalla varmistetaan, että dokumentti sisältää oikeat tiedot oikeassa järjestyksessä ja muodossa [Campbell *et al.*, 2003]. Tällöin siirrettyä dataa ei tarvitse erikseen validoida ennen sen jatkokäsittelyä. Jos esimerkiksi mallin mukaan arvo on validoitu kaksidesimaaliseksi numeroksi, se voidaan tallentaa ilman tarkistusta vastaavaan tietokantataulun kenttään.

3.1. Xml-skeeman erityispiirteitä

Yksi kuvailukielistä on xml-skeema, joka on W3C:n 1998 aloittaman kehitystyön tulos [Campbell *et al.*, 2003]. Sitä alettiin työstämään, kun todettiin, että sen hetkinen suosittu kuvailukieli dtd kaipasi laajennusta. Dtd on ollut mukana jo xml-spesifikaation ensimmäisessä versiossa [Campbell *et al.*, 2003]. Tarkoituksena oli sisällyttää xml-skeemaan dtd:tä vastaavat ominaisuudet. Lisäksi xml-skeemaan on lisätty uusia hyödyllisiä ominaisuuksia.

Xml-skeema on kirjoitettu xml-kielellä toisin kuten edeltäjänsä dtd, jossa on oma erillinen syntaksinsa. Samanlaisesta syntaksista on apua esimerkiksi silloin, kun xml- ja skeema-tiedostot tallennetaan samaan paikkaan [Lee and Chu, 2000]. Näin voi olla esimerkiksi ohjelmilla, jotka lähettävät ja vastaanottavat xml-tiedostoja. Myös xml-skeemojen muokkaaminen on helpompaa, koska editointiohjelman tai käyttäjän ei tarvitse opetella uutta syntaksia. Editointiohjelma voi myös validoida sekä xml-tiedoston että xml-skeeman samoilla säännöillä. [Campbell *et al.*, 2003].

Xml-skeemassa on uutena ominaisuutena nimiavaruudet [Namespaces2004]. Ne ratkaisevat ongelmatilanteet, joissa sama käsite tarkoittaa kahta asiaa. Nimiavaruuksien avulla ne voidaan liittää omaan toimialaansa ja niille ei tarvitse väkisin keksiä synonyymejä. Esimerkiksi rautatieaseman tietojärjestelmässä voisi esiintyä asema-käsite sekä junan päätepisteenä että tietokoneen jaettuna asemana. Nimiavaruudet lisäävät tiedon rakenteisuutta ja mahdollistavat eri nimiavaruuksista olevien käsitteiden yhdistämisen. Dacontan ja muiden [2003] siirrettävän tiedon jatkumossa eri toimialojen käsitteiden yhdistäminen merkitsee sitä, että tiedosto sisältää enemmän informaatiota. Eri nimiavaruuksien avulla voidaan yhdistää eri toimialojen skeemoja ja vähentää nimikonflikteja.

Xml-skeeman spesifikaatio sisältää monipuolisia tietotyyppejä, jotka helpottavat datan määrittelyä ja sitä kautta sen validointia [Campbell *et al.*, 2003]. Xml-skeemassa on enemmän tietotyyppejä (37 kpl) kuin edeltäjässään dtd:ssä (10) [Lee and Chu, 2000]. Kun data voidaan validoida automaattisesti, sen jatkokäsittely on helpompaa.

Champbellin mukaan xml-skeeman tärkein tehtävä on tuottaa metadataa xml-dokumentista [Campbell *et al.*, 2003]. Sen tarkoitus on kuvata mahdollisimman tarkasti, minkälainen xml-dokumentti voi olla. Suurikaan joukko xml-dokumentteja ei pysty sitä tekemään tyhjentävästi, koska ne ovat vain xml-skeemaa noudattavia erilaisia ilmentymiä.

Erilaisia kuvailukieliä on monia, niitä on sekä yritysten että yksityisten ja julkisten yhtymien kehittämiä. Xml-skeema ja dtd ovat molemmat W3C:n kehittämiä. Aiemmin mainittu Xdr on puolestaan koalition, johon kuuluu mm. Microsoft, kehittämä kuvailukieli. Se on käytössä Microsoftin Biztalk-ohjelmassa, jota esitellään tarkemmin myöhemmin. Xdr on hieman rajoittuneempi kuin xml-skeema. Se ei esimerkiksi tue uusien tietotyyppien luontia. Toisaalta se rajoittaa vähemmän xml-dokumenttia. Se esimerkiksi sallii xml-dokumentissa tageja, joita ei ole mallissa määritelty. [Lee and Chu, 2000]

3.2. Xml-skeeman syntaksi

Xml-skeeman syntaksi on samanlainen kuin xml-kielen, sekin siis koostuu itse datan lisäksi sitä kuvaavista tageista. Xml-skeemassa on lisäksi varattuja sanoja. Liitteessä 1 on esimerkki xml-skeemasta ja liitteessä 2 on skeemaa noudattava xml-tiedosto. Jokainen xml-tiedostossa esiintyvä tagi määritellään xml-skeemassa erikseen. Lisäksi xml-skeemassa määritellään tagien järjestys.

Xml-skeema koostuu *yksinkertaisista* (simpletype) ja *monimutkaisista* (complextype) tageista. Yksinkertainen tagi sisältää vain yhden datan osan ja monimutkainen voi sisältää useita tageja. Jokaiselle yksinkertaiselle tagille määritellään sen tietotyyppi. [XMLSchema] Lisäksi tagille voidaan antaa sen esiintymiseen ja sisältöön liittyviä rajoituksia. Jos yksinkertaiselle tagille ei anneta muita määrityksiä kuin sen tyyppi, tagi voidaan esittää yhdellä rivillä. Seuraavassa xml-skeeman määrittely Tuote-tagille, jonka tietotyyppi on string:

```
21        <xsd:element name="Tuote" type="xsd:string"/>.
```

Edellä mainittu rivi vastaa xml-dokumentin riviä:

```
8       <Tuote>---</Tuote>.
```

Jos tagille määritellään muita ominaisuuksia kuin sen tietotyyppi, ne esitellään ennen lopetustagia. Esimerkiksi numeroille voidaan määritellä mm. arvoalue, maksimiarvo ja oletusarvo [Lee and Chu, 2000]. Seuraavassa esimerkissä on määritelty Kplmaara-tagi, jonka tietotyyppi on positiivinen kokonaisluku ja maksimiarvo 100. Esimerkistä on jätetty kolmen ensimmäisen tagin lopetustagi pois:

```
22        <xsd:element name="Kplmaara">
23          <xsd:simpleType>
24            <xsd:restriction base="xsd:positiveInteger">
25              <xsd:maxExclusive value="100"/>.
```

Edellä oleva kplmaara-tagi on tyypiltään yksinkertainen, koska se sisältää vain yhden datan, kappalemäärän. Monimutkaisemmat tagit voivat sisältää sekä yksinkertaisia että monimutkaisia tageja. Lisäksi ne sisältävät tiedon elementtien järjestyksestä, pakollisuudesta ja lukumääristä. Yksinkertaisen tai monimutkaisen tagin yhteydessä kerrotaan, kuinka monesti se voi tai sen

täytyy esiintyä. Jos lukumääriä ei anneta, oletetaan että tagin täytyy esiintyä vain kerran. Seuraavassa esitellään Rivi-tagi, jonka ei tarvitse esiintyä xml-dokumentissa kertaakaan tai se voi esiintyä enintään 1000 kertaa:

```
18    <xsd:element name="Rivi" minOccurs="0" maxOccurs="1000">.
```

Tagien järjestyksen voi määritellä sequence-tagilla. Seuraavassa OstolaskuTyyppi-tagissa täytyy olla ensin Laskuttaja-tagi ja sen jälkeen Laskurivit-tagi. Molempien tagien tietotyyppinä on itse määritellyt tyypit (Osapuoli ja Rivit), jotka on nimetty käyttökohteen mukaan uudelleen (Laskuttaja ja Laskurivit):

```
4   <xsd:complexType name="OstolaskuTyyppi">
5    <xsd:sequence>
6     <xsd:element name="Laskuttaja" type="Osapuoli"/>
7     <xsd:element name="Laskurivit"  type="Rivit"/>
8   </xsd:sequence>.
```

On riittävää, että tagi määritellään rajoituksineen vain kerran ja siihen viitataan myöhemmin kaikista niistä paikoista, missä tagia mahdollisesti käytetään. Seuraavassa on xml-skeeman määrittely lisatieto-tagille:

```
3   <xsd:element name="Lisatiedot" type="xsd:string"/>.
```

Esittelyn jälkeen samaan tagiin voi viitata useamman monimutkaisen tagin sisältä. Tällöin sen ilmentymillä voi olla erilaisia lukumäärään liittyviä ehtoja. Seuraavassa esimerkki edellä määritellyn Lisatiedot-tagin käytöstä kahden eri monimutkaisen tagin osana xml-skeemassa:

```
13     <xsd:element ref="Lisatiedot"   minOccurs="1"/>
30          <xsd:element ref="Lisatiedot"   minOccurs="0"/>.
```

Xml-skeeman tärkein tehtävä on kuvailla rakenteista dataa mahdollisimman tarkasti siten, että dataa voi käyttää uudelleen ilman xml-skeeman ulkopuolista validointia [Campbell *et al.*, 2003]. Kaikista tarkistuksista ei tosin päästä eroon. Jos xml-tiedosto sisältää esimerkiksi yrityksen sisäisen

avainnumeron tai tuotekoodin, tämä täytyy erikseen tarkistaa tietokantahaulla tai muulla tavalla.

Validoinnin helpottamisen lisäksi xml-skeemassa on piirteitä, jotka mahdollistavat sen uudelleen käyttämisen. Esimerkiksi xml-skeemaan voidaan muodostaa yleisiä perustyyppejä, kuten osoite, ja käyttää niitä uudelleen eri yhteyksissä, kuten myyntilaskuosoitteena tai ostolaskuosoitteena. On myös mahdollista sisällyttää xml-skeemaan tageja muista xml-skeemoista ja nimiavaruuksista. [XMLSchema]

4. Ratkaisun arviointia

4.1. Arviointikriteereistä

Xml-skeema ja yleisemmin informaation lisääminen siirrettäviin viesteihin on saanut osakseen niin tukea kuin myös kritiikkiä. Yen ja muut [2002] arvioivat xml-teknologian vaikutusta sähköiseen kaupankäyntiin sekä mikro- että makrotasolla. Mikrotasolla Yen toteaa teknologian hyödyllisyyden mittareiksi mm. sen, että ohjelmointikielet, editorit ja portaalit tukevat sitä. Xml-kielellä on ohjelmointikielien tuki takanaan, koska suurimmat kielet käyttävät jo W3C:n julkaisemaa dom-mallia [Lawrence, 2004; Dom]. Editorien tuki puolestaan tarvitaan, jotta voidaan muodostaa ja muokata xml- ja skeema-tiedostoja [Yen *et al.*, 2002]. Nämä pitävät huolta tiedostojen validoinnista. Niissä on useita ominaisuuksia, kuten sanastoja, jotka nopeuttavat xml-skeemojen kirjoittamista. Portaalien tukea tarvitaan, jotta xml-tiedostojen lähetys ja vastaanotto sujuvat ongelmitta. Ne voivat myös huolehtia vastaanotettujen tiedostojen validoinnista, arkistoinnista ja muuttamisesta mallista toiseen.

Makrotasolla Yen ja muut [2002] esittävät kysymyksen, pitäisikö xml-pohjaisten ratkaisujen kehitys olla nopeaa vai hidasta. Lisäksi he myös pohtivat sitä, pitäisikö xml-teknologian kehityksen olla avointa. [Yen *et al.*, 2002] Jos kehitys on nopeaa, uudet tulokset saadaan nopeammin käyttöön. Toisaalta tällöin käyttöön saattaa päätyä useampia versioita kuin hitaan kehityksen avulla. Sama ongelma saattaa tulla vastaan myös avoimen kehityksen kohdalla. Molemmissa tapauksissa uudet ominaisuudet saadaan nopeasti käyttöön, mutta erilaisten versioiden tukeminen saattaa muodostua ongelmaksi. Hitaan kehityksen hyvä puoli on tarkan standardin kehittäminen, mutta toisaalta osa ominaisuuksista saattaa olla vanhentuneita siinä vaiheessa, kun uusi versio viimein julkaistaan [Yen *et al.*, 2002]. Jos xml-teknologian kehitys on vapaata, voi tulla vastaan tilanne, jossa eri ohjelmistoyrityksillä on omanlaisensa standardit. Kuten esimerkiksi Microsoftin Biztalk-ohjelmassa, jossa on käytössä

xdr-syntaksi. Tällöin xml-skeemat eivät enää ole niin riippumattomia käyttöympäristöstään kuin jos käytössä olisi vain yksi yleinen standardi.

4.2. Kritiikkiä

Daconta ja muut [2003] esittelevät kriitikkojen useimmiten käyttämiä argumentteja ja esittää näille vasta-argumentteja. Xml-teknologian on pelätty tulevan lopulta kalliimmaksi kuin sen tuoma hyöty. Taloudellinen näkökulma on tärkeä, silloin kun teknologia otetaan kaupalliseen käyttöön. Daconta kuitenkin toteaa, että lisääntynyt avoin lähdekoodi ja runsas tutkimus ovat tuoneet sovellusten hintoja alas. Joten taloudellinen riski xml-teknologian käytössä on pienentynyt.

Daconta ja muut [2003] toteavat kriitikoiden myös ennustavan, että xml-teknologia on vain harha-askel etenevässä teknologian kehityksessä. Tätä argumenttia he eivät toki pysty kumoamaan, koska tuloksen näkee yleisemmin vasta vuosien kuluttua. Mutta he korostavat, että samantyyppisiä kommentteja annettiin myös kun www-teknologia otettiin käyttöön. Siirrettävän datan jatkumoa on puolestaan kritisoitu siitä, että sen tavoitteena on tehdä tietokoneesta älykkäämpi. Osa kriitikoista on jopa luonut uhkakuvia, joissa tietokone pystyy oppimaan kuten ihminen ja ei enää ole itsestään selvää, että ihminen hallitsisi koneita. Daconta ja muut [2003] lohduttavat, että informaation lisäämisellä dataan ei ole tarkoitus tehdä koneista älykkäämpiä vaan sen sijaan tehokkaampia. Tietokoneiden ei ole tarkoitus oppia älykkääksi kuten ihminen. Paremminkin ihmisen on tarkoitus oppia, miten käsitemalleja voi esittää ohjelmille siten, että ne pystyvät sääntöjen avulla käsittelemään tehokkaasti myös monimutkaisia käsitemalleja.

5. Xml-teknologian hyödyntäminen sähköisissä palveluissa

Xml-kieltä voidaan käyttää viestien välitykseen liiketoimien osapuolien välillä. Viestit voidaan validoida mallikielien, esimerkiksi xml-skeeman avulla. Viestien lähettäminen tarvitsee oman hallintatyökalunsa, koska viestejä voi joutua lähettämään uudelleen, jos viestin vastaanotto ei ole esimerkiksi verkkokatkoksen takia onnistunut. Myös osa lähetettävistä dokumenteista, kuten esimerkiksi sähköiset laskut, voivat tarvita arkistointia [Kim *et al.*, 2003]. Hallintaohjelma voi myös kirjata lokia, josta näkee lähetetyt ja saapuneet viestit sekä mahdolliset virheilmoitukset.

Kim ja muut [2003] mainitsevat esimerkkinä sähköiset palvelut, missä viestien välitys osapuolten välillä on olennainen osa liiketoimintaa [Kim *et al.*, 2003]. Viestien lähetys voi olla päivittäin tai useamminkin toistuvaa, esimerkkinä laskujen lähetys tai tuotteen hinnan kysyminen. Viestin lähettäjä tai

vastaanottaja voi olla esimerkiksi verkkokauppaa käyttävä yksityinen asiakas tai toinen yritys. Molemmissa tapauksissa voidaan käyttää samanlaista mallia viestissä, joka sisältää esimerkiksi tuotteen hinnan kysyminen –transaktion. Useimpien hallintaohjelmien periaatteina on liiketoiminnan periaatteisiin pohjautuvat transaktiot ja niiden perusteella luodut viestimallit [Kim *et al.*, 2003]. Esimerkkejä hallintaohjelmista ovat mm. Microsoftin BizTalk ja RosattaNet, jota kehittämässä olevaan koalitioon kuuluvat mm. IBM ja Compaq. Ohjelmat esitellään tarkemmin myöhemmin.

Yenin ja muiden [2002] mukaan hallintaohjelmien tärkein tarkoitus on mahdollistaa sujuva tiedon jakaminen liikekumppaneiden kanssa. Myös Kimin ja muiden [2002] mukaan xml-teknologian tarkoitus viestinvälityksessä on yksinkertaistaa tiedonvälitystä yritysten välillä. Se onnistuu, koska käyttäjät voivat määritellä viesteille omanlaisensa rakenteisen syntaksin. Tästä tosin seuraa se, että samalla käsitteellisellä asialla, kuten esimerkiksi ostolaskulla, on erilaisia malleja. Voi käydä jopa niin, että yhdessä yrityksessä samaa asiaa kuvataan yhdellä mallilla ja toisessa yrityksessä kokonaisuus on kuvattu kahdella erillisellä mallilla. Esimerkiksi lasku ja sen tuotteiden toimitustiedot voisivat olla joko yhdessä yhteisessä tai kahdessa erillisessä mallissa. Yksi yritys todennäköisesti siis vastaanottaa useita eri skeeman mukaisia xml-tiedostoja samasta asiasta. Joten hallintaohjelmien yksi tärkeä tehtävä on muuntaa nämä erimalliset tiedostot yhteen yrityksen oman mallin mukaiseen muotoon. Tällöin tietoja on helpompi käsitellä edelleen. Koska hallintaohjelmalle jätetään tehtäväksi erilaisten xml-tiedostojen muunto yhteisen mallin mukaiseksi, yrityksen omien ohjelmien ei tarvitse käsitellä muita kuin yrityksen omaa mallia. Tällöin kun yritys alkaa vastaanottaa uudenlaista viestiä uudelta asiakkaalta, muutos tarvitaan vain hallintaohjelmaan, ei varsinaista tietoa käsittelevään ohjelmaan.

Säännöt erilaisten xml-skeemojen muuttamisesta muodosta toiseen täytyy toistaiseksi syöttää hallintaohjelmiin käsin [Rahm and Bernstein, 2001]. Graafisen käyttöliittymän avulla tämä käy useimmissa ohjelmissa helposti. Tosin jos mallien muoto muuttuu uuden version tai asiakkaan vuoksi, täytyy muutossäännöt päivittää käsin muutosten yhteydessä. Tällöin täytyy ottaa huomioon se, että samaa asiaa kuvaavissa erilaisissa skeemoissa voi olla erilaisia kenttiä. Esimerkiksi kenttien tyyppi, pakollisuus tai pituus voi olla erilainen. Myös kenttien rakenne voi olla erilainen. [Rahm and Bernstein, 2001] Esimerkiksi laskussa toimitusosoite-kenttä voi esiintyä joko vain kerran tai toistua jokaisen laskurivin kohdalla uudelleen Tällaiset ominaisuudet hankaloittavat tietojen siirtämistä skeemasta toiseen. Daconta ja muut [2003] tavoittelevat siirrettävän datan jatkumossa tietoa, joka on rakentunut siten, että

loogisten sääntöjen avulla voidaan jalostaa sitä edelleen. Tämän periaatteen avulla voidaan opettaa ohjelma yhdistämään erilaisia skeemoja erilaisten käsitemallien avulla ja niiden välisten suhteiden avulla.

RosettaNetin on kehittänyt koalitio, johon kuuluvat mm. IBM, Microsoft, SAP ja Compaq. RosettaNetin tarkoituksena on helpottaa kahden osapuolen välistä kommunikointia. Tämä tapahtuu luomalla standardeja erilaisiin transaktioihin osapuolten välillä. Ohjelmassa määritellään erilaisia myyntiketjun osapuolia, kuten esimerkiksi valmistaja ja loppukäyttäjä. Lisäksi määritellään osapuolien välille erilaisia transaktioita. Osapuolten liiketoimintaprosessit jaetaan osiin, kuten esimerkiksi tilausten hallinta ja tukipalvelut. Nämä puolestaan jaetaan pienempiin segmentteihin, kuten tuotteen personointi tai takuuvaihdot. Jokainen näistä koostuu dialogista, jossa osapuolet vaihtavat keskenään viestejä [RosettaNet].

Biztalk on Microsoftin kehittämä tuote, jonka tarkoituksena on hallita webpalveluja ja vuoropuhelua osapuolten ohjelmien välillä. Se määrittelee mm. tiedon kulkua webbipalvelun ja sen taustalla olevien ohjelmien välillä. [Daconta *et al.*, 2003] Biztalk mahdollistaa xml-tiedostojen muuttamisen mallista toiseen. Ohjelmassa yhdistetään erilaiset liiketoimintaprosessit niitä vastaaviin pyyntö- ja vastausskeemoihin. Skeemat kertovat minkälaisia vastaanotettavien ja lähetettävien tiedostojen tulee olla ja liiketoimintaprosessit puolestaan kertovat miten saatu tieto prosessoidaan eteenpäin. [Kim *et al.*, 2003] Biztalkissa käytetään tiedostojen mallina Microsoftin kehittämää xdr-skeemaa, joka eroaa hieman W3C:n kehittämästä xml-skeemasta.

Sekä Biztalkin, RosettaNetin että muiden samoja palveluja tarjoavien ohjelmien tavoitteena on helpottaa tiedon välitystä liiketoimien osapuolten välillä. Molemmat ohjelmat huolehtivat datan validoinnista skeeman avulla ja tiedostojen siirron hallinnoinnista. Ohjelmissa on valmiina erilaisia transaktiomalleja eri myyntiketjun osapuolten välillä. [Kim *et al.*, 2003] Ohjelmien palveluja kaipaavan yrityksen tulee valita sellainen, jossa on valmiina yrityksen tarvitsemat prosessit tai sellainen ohjelma, jossa voi vähällä vaivalla personoida prosessit yritykselle sopiviksi.

6. Yhteenveto

Informaation lisääminen yritysten välillä vaihdettavaan dataan on yksi tiedonsiirron haasteista. Dacontan ja muiden [2003] mukaan tiedonsiirtoa voidaan tehostaa muokkaamalla tiedonsiirrossa käytettäviä rakenteita. He näkevät xml-kielen ja sen mallikielet ratkaisuna ongelmaan. W3C:n kehittämä xml-skeema on yksi xml-teknologian mallikielistä. Skeeman syntaksi on samankaltainen kuin xml-kielen syntaksi, lisänä siinä on vain lista varattuja sanoja.

Skeeman tarkoitus on validoida xml-tiedosto ennen tiedon jatkokäsittelyä. Xml-teknologiasta on hyötyä esimerkiksi sähköisissä palveluissa. Niiden taustalla voidaan käyttää välitettävien viestien hallintaohjelmia. Nämä muuttavat liiketoimintaperiaatteet prosesseiksi ja vuoropuheluiksi osapuolten välillä. Dialogia voidaan käydä xml-muotoisten viestien avulla. Xml-tiedoston muuttaminen mallista toiseen automaattisesti on yksi seuraavista tiedonsiirron kehittämisen haasteista. Informaation lisäämistä siirrettävän dataan on kritisoitu siitä, että se vie teknologian kehityksen sivuraiteille. Daconta ja muut [2003] puolestaan ennustavat sen kehittyvän kohti käsitemalleja, joiden perusteella voidaan muodostaa uusia rakenteita loogisten sääntöjen avulla.

Viiteluettelo

[Campbell *et al.*, 2003] Charles E. Campbell, Andrew Eisenberg and Jim Melton, XML schema. *SIGMOD Record* **32**, 2 (June 2003), 96 – 101.

[Daconta *et al.*, 2003] Michael C. Daconta, Leo J. Obrst and Kevin T. Smith, *The Semantic Web : a Guide to the Future of XML, Web Services, and Knowledge Management*. Wiley, 2003.

[Dom] World Wide Web Consortium, Document Object Model (DOM) http://www.w3.org/DOM/ (24.11.2004).

[Han *et al.*, 2003] Wook-Shin Han, Ki-Hoon Lee and Byung Suk Lee, An XML storage system for object-oriented/object-relational DBMSs. *Journal of Object Technology* **2**, 3 (May-June 2003) 113-126.

[Kim *et al.*, 2003] Dan Jong Kim, Manish Agrawal, Bharat Jayaraman, and H. Raghav Rao, A comparison of B2B e-service solutions. *Communications of the ACM* **46**, 12 (Dec 2003), 317-324.

[Lawrence, 2004] Ramon Lawrence, The space efficiency of XML. *Information and Software Technology* **46**, 11 (Sep. 2004), 753-759.

[Lee and Chu, 2000] Dongwon Lee and Wesley W. Chu, Comparative analysis of six XML schema languages. *SIGMOD Record* **29**, 3 (Sep. 2000), 76 – 87.

[Namespaces] World Wide Web Consortium, Namespaces in XML http://www.w3.org/TR/REC-xml-names/ viitattu 24.11.2004.

[Niiniluoto, 1996] Ilkka Niiniluoto, *Informaatio, tieto ja yhteiskunta: Filosofinn käsiteanalyysi.* Edita, 1996.

[Rahm and Bernstein, 2001] Erhard Rahm and Philip A. Bernstein, A survey of approaches to automatic schema matching. *The VLDB Journal* **10**, 4 (Dec 2001), 334 – 350.

[RosettaNet] RosettaNet Homepage http://www.rosettanet.org/rosettanet viitattu 24.11.2004.

[Yen *et al.*, 2002] David C. Yen, Shi-Ming Huang and Cheng-Yuan Ku, The impact and implementation of XML on business-to-business commerce. *Computer Standards & Interfaces* **24**, 4 (Sep 2002), 347-362.

[XMLSchema] World Wide Web Consortium, XML schema Specification http://www.w3.org/TR/2004/REC-xmlschema-0-20041028/viitattu 24.11.2004.

Liite 1

```
1  <xsd:schema xmlns:xsd="--">
2  <xsd:element name="Ostolasku" type="OstolaskuTyyppi"/>
3   <xsd:element name="Lisatiedot" type="xsd:string"/>
4   <xsd:complexType name="OstolaskuTyyppi">
5    <xsd:sequence>
6     <xsd:element name="Laskuttaja" type="Osapuoli"/>
7     <xsd:element name="Laskurivit"  type="Rivit"/>
8    </xsd:sequence>
9  </xsd:complexType>
10 <xsd:complexType name="Osapuoli">
11  <xsd:sequence>
12   <xsd:element name="Yritys"   type="xsd:string"/>
13    <xsd:element ref="Lisatiedot"   minOccurs="1"/>
14  </xsd:sequence>
15 </xsd:complexType>
16 <xsd:complexType name="Rivit">
17  <xsd:sequence>
18   <xsd:element name="Rivi" minOccurs="0" maxOccurs="1000">
19    <xsd:complexType>
20     <xsd:sequence>
21      <xsd:element name="Tuote" type="xsd:string"/>
22      <xsd:element name="Kplmaara">
23       <xsd:simpleType>
```

```
24              <xsd:restriction base="xsd:positiveInteger">
25                <xsd:maxExclusive value="100"/>
26              </xsd:restriction>
27            </xsd:simpleType>
28          </xsd:element>
29          <xsd:element name="Hinta"  type="xsd:decimal"/>
30          <xsd:element ref="Lisatiedot"   minOccurs="0"/>
31        </xsd:sequence>
32      </xsd:complexType>
33    </xsd:element>
34  </xsd:sequence>
35 </xsd:complexType>
36 </xsd:schema>
```

Liite 2

```
1  <?xml version="1.0"?>
2  <Ostolasku Laskupaiva="2004-01-01">
3    <Laskuttaja>
4      <Yritys>Oy Firma AB</Yritys>
5      <Lisatiedot>=)</Lisatiedot>
5    </Laskuttaja>
6    <Laskurivit>
7      <Rivi>
8        <Tuote>---</Tuote>
9        <Kplmaara>1</Kplmaara>
10       <Kplhinta>1.00</Kplhinta>
11     </Rivi>
12   </Laskurivit>
13 </ Ostotilaus>
```

SuperOffice NetServer

Janne Penttilä

Tiivistelmä.
Tämä tutkielma käsittelee SuperOffice CRM5 -asiakkuudenhallintajärjestelmän NetServer –rajapintaa, joka on kyseisen järjestelmän sovelluskehitysympäristö. Aluksi luon lyhyen katsauksen asiakkuudenhallinnan perusteisiin ja kerron hieman SuperOffice CRM5 -järjestelmän toiminnasta. Sen jälkeen tutkin tarkemmin NetServer-rajapinnan ominaisuuksia ja käyttömahdollisuuksia tilaajayrityksen tarpeisiin.

Avainsanat ja -sanonnat: Asiakkuudenhallinta, Asiakkuudenhallintajärjestelmä
CR-luokat: H.2.4, J.1

1. Johdanto

Tutkielman tilaajaana on toiminut DB-Manager Oy, joka on vuonna 1984 perustettu tamperelainen ohjelmistoalan yritys. Oman sovelluskehityksen lisäksi yritys markkinoi ja jälleenmyy norjalaisen SuperOffice ASA:n kehittämää SuperOffice CRM5 -asiakkuudenhallintajärjestelmää.

Tutkielman lähtökohtana oli DB-Manager Oy:n toimitusjohtaja Simo Saarimaan pyyntö tutkia SuperOffice CRM5 -järjestelmän NetServer-rajapintaa ja arvioida sen käyttömahdollisuuksia yrityksen ohjelmistokehitystyössä. Samalla tavoitteena oli perehdyttää joku yrityksen työntekijöistä NetServer-ympäristön osaajaksi. Koska olen itse kyseisen yrityksen työntekijä, tutkielman tekeminen tämän aiheen tiimoilta vaikutti hyvältä ajatukselta sekä omasta että työnantajani näkökulmasta: yritys saisi samalla sekä selvityksen SuperOffice NetServer –rajapinnasta että siihen perehtyneen työntekijän. Itselleni karttuisi sekä kokemusta tutkimustyön tekemisestä että ammattitaitoa, jota voin suoraan hyödyntää työssäni.

Käyn aluksi lyhyesti läpi asiakkuudenhallintaa käsitteenä sekä asiakkuudenhallinan ja asiakkuudenhallintajärjestelmien piirteitä ja kehitystä viime vuosien aikana. Sen jälkeen esittelen SuperOffice CRM5 -järjestelmää, sen käyttötarkoitusta, rakennetta ja perustoimintoimintoja. Lopuksi paneudun syvemmin tutkielman varsinaiseen aiheeseen eli SuperOffice NetServer-rajapintaan, jota tarkastelen melko yksityiskohtaisesti ohjelmistokehittäjän näkökulmasta. Tämän tarkastelun avulla pyrin selvittämään, olisiko NetServer hyödyllinen työkalu DB-Manager Oy:n tarpeisiin.

2. Asiakkuudenhallinta eli CRM

Asiakkuudenhallinta (Customer Relationship Management, CRM) on yleiskäsite, jolla tarkoitetaan kaikkea sitä, mikä liittyy yrityksen tapaan hallita ja kehittää asiakassuhteitaan. Siihen sisältyy sekä teknologiset ratkaisut että yrityksen asiakkaitaan koskevat strategiset linjaukset ja tavoitteet. Asiakkuudenhallintajärjestelmä puolestaan viittaa tekniseen ratkaisuun, tapaan hoitaa asiakkuudenhallintaa. Asiakkuudenhallintajärjestelmät voidaan jakaa myynnin, markkinoinnin sekä asiakaspalvelun ratkaisuihin [Jantunen *et al.*, 2001].

2.1. Määritelmiä

Yleisen suomalaisen asiasanaston [YSA, 2004] mukaan käsite asiakkuudenhallinta tarkoittaa liiketoimintastrategiaa, jonka tavoitteena on organisoida toiminta asiakassegmenteittäin, edistää asiakaslähtöistä ajattelua ja integroida toisiinsa liiketoimintaprosessit asiakkaista toimittajiin. Martin [2004] puolestaan määrittelee käsitteen tarkoittamaan tiedon keräämistä, hallinnointia ja analysointia asiakkuudenhallintajärjestelmän avulla sekä syntyneen tietämyksen käyttämistä yrityksen toiminnan kehittämiseen sellaiseksi, että yritys yhä tehokkaammin ja paremmin tyydyttää asiakkaan tarpeet.

2.2. Kehitys

Ensimmäiset CRM-järjestelmät tuotiin markkinoille 1990-luvun vaihteessa ja niiden tarkoitus oli automatisoida asiakkaiden hankkimiseen, palvelemiseen ja säilyttämiseen liittyviä prosesseja. Internetin yleistymisen jälkeen asiakkuudenhallinnan kenttä on muuttunut: asikkaat ovat nykyään interaktiivisempia ja omatoimisempia, ja tämä on aiheuttanut muutoksia myös asiakkuudenhallintajärjestelmissä. [Martin, 2004]

Nykyaikaiset asiakkuudenhallintajärjestelmät auttavat yrityksiä kannattavimpien asiakkaiden löytämisessä sekä asiakkaiden yksilöllisessä palvelemisessa. CRM-järjestelmä yhdistää asiakkaat eri kanavien kautta yrityksen myynnin, markkinoinnin ja asiakaspalvelun järjestelmiin. Asiakkuudenhallintajärjestelmien on myös integroiduttava yrityksen muihin järjestelmiin, jotta asiakasohjautuvuus lävistäisi koko organisaation. [Martin, 2004]

3. SuperOffice CRM5

SuperOffice CRM5 on norjalaisen SuperOffice ASA:n kehittämä tuote, joka on yksi Euroopan eniten myydyistä asiakkuudenhallintajärjestelmistä. Asiakkaita on maailmanlaajuisesti noin 11.500 ja ylläpitopalvelussa olevia toimitetettuja lisenssejä noin 150.000 [DB-Manager, 2004].

3.1. Ominaisuudet ja toiminnot

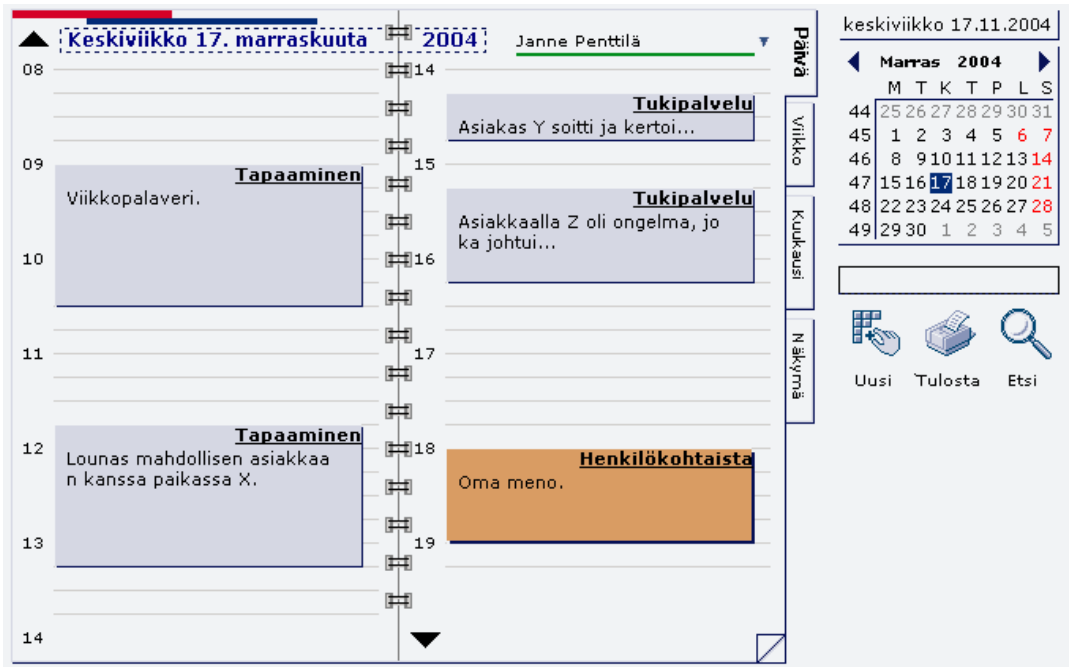
SuperOffice CRM5 on relaatiotietokannan päällä toimiva Windows-sovellus, joka tukee yleisimpiä markkinoilla olevia tietokantoja (mm. DB2, Microsoft SQL Server, Oracle, Sybase Anyware). Sovellusta voidaan ajaa joko keskustietokantaan kytkettynä monen käyttäjän versiona tai erillisenä yhden käyttäjän *Travel*-versiona. Tätä tutkielmaa tehdessäni olen käyttänyt ainoastaan Microsft SQL Server 2000 –tietokannan päällä ajettavaa monen käyttäjän ohjelmaversiota, joten käyttökokemukseni perustuvat ainoastaan siihen. Tietääkseni ohjelma kuitenkin toimii samoin kaikissa tuetuissa ympäristöissä.

SuperOffice CRM5 kattaa kaikki kolme asiakkuudenhallinnan osa-aluetta. Myyntitoimenpiteitä on pyritty tehostamaan siten, että potentiaalisia asiakkaita voidaan jakaa kategorioihin ja antaa toteutumattomille kaupoille totetumistodennäköisyyksiä. Erityisesti markkinointia tukee segmentointiajattelu, jonka avulla asiakkaita voidaan ryhmitellä eri perustein, joka puolestaan edesauttaa kohdennettua markkinointia. Asiakaspalvelutehtävissä käyttäjä voi kohdistaa asiakkaalle erilaisia aktiviteetteja (esim. puhelu), jotka tallentuvat tietokantaan asiakkaan tietoihin. Näin asiakkaisiin kohdistuneita toimenpiteitä voidaan selata tarvittaessa jälkikäteen.

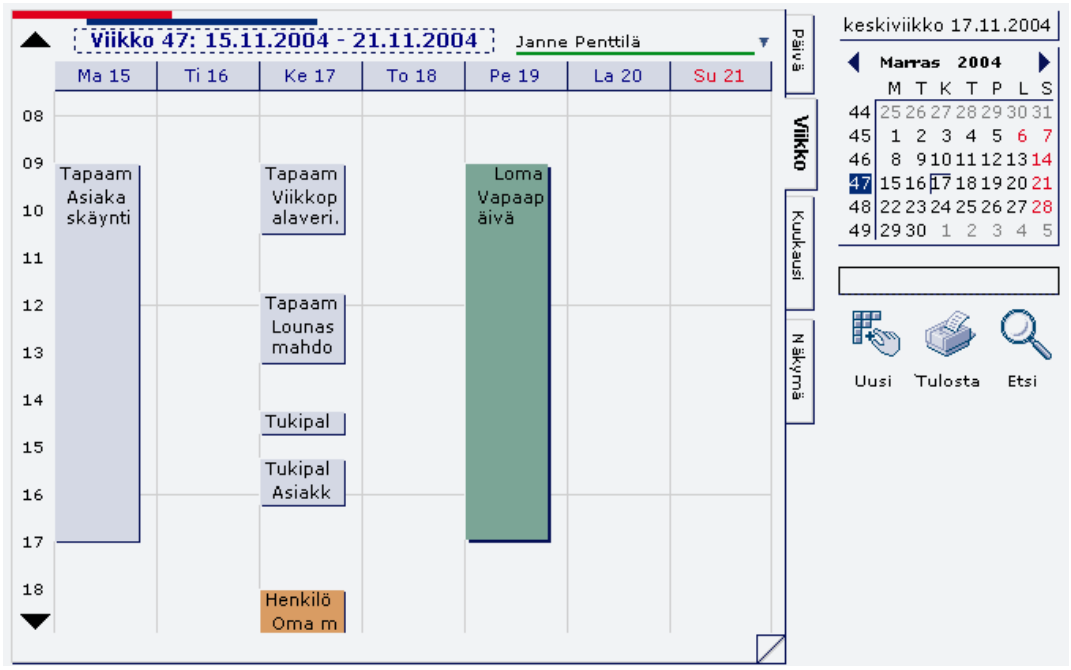
Ohjelmassa on myös kalenteri, johon käyttäjät voivat tehdä merkintöjä (mm. tukipalvelut, poissaolot, palaverit yms.), joita voidaan tarkastella päivä-, viikko- tai kuukausitasolla. Kalenterin avulla voidaan myös sopia ryhmäpalavereita ja muita tapahtumia, joihin osallistuu useampi henkilö. Useimmat ohjelman keskeisistä toiminnoista perustuvat kalenteriin.

3.2. Käyttöliittymä

Ohjelman käyttöliittymä on selkeä ja visuaaliset elementit ovat keskeisessä osassa. Käyttöliittymässä on pyritty tuomaan kaikki ohjelman perustoiminnot käyttäjän ulottuville. Kuva 1 esittää ohjelman päiväkalenteria, joka muistuttaa ulkoisesti tavallista pöytäkalenteria ja on siksi melko helppo omaksua ja käyttää. Kuvassa 2 on ohjelman viikkokalenterinäkymä, jonka avulla käyttäjä voi tarkastella karkealla tasolla joko omaa tai toisen käyttäjän viiko-ohjelmaa.



Kuva 1. Super Office CRM5 päiväkalenterinäkymä



Kuva 2. Super Office CRM5 viikkokalenterinäkymä

Käyttöliittymän toteutus pyrkii selvästi erottumaan ns. standarin mukaisista Windows-sovelluksista. Tasapainoisessa esityksessä [Sinkkonen *et al.*, 2002] olennaiset asiat on tuotu esille selkeästi, mutta liioittelematta, käyttäen värejä ja kontrastia sopivasti. Mielestäni Super Officen käyttöliittymä on onnistunut esimerkki tasapainoisesta esityksestä.

3.3. Käyttäjien kommentteja

Asiakkuudenhallinta on tutkimusalueena vielä niin nuori, että aihetta käsittelevää tutkimustietoa on vaikea löytää, eikä Super Office -järjestelmää koskevia tutkimuksia ei ole yleisesti saatavilla vielä ollenkaan. Seuraavassa on kuitenkin joitakin käyttäjien kommentteja, jotka perustuvat Idearitsa Oy:n teettämään haastattelututkimukseen [Pinomäki, 2004]. Käyttäjille esiteltiin neljää eri CRM-järjestelmää, jotka olivat Microsoft CRM, Prospekti, Super Office CRM5 ja Vineyard Vintage 4.0.

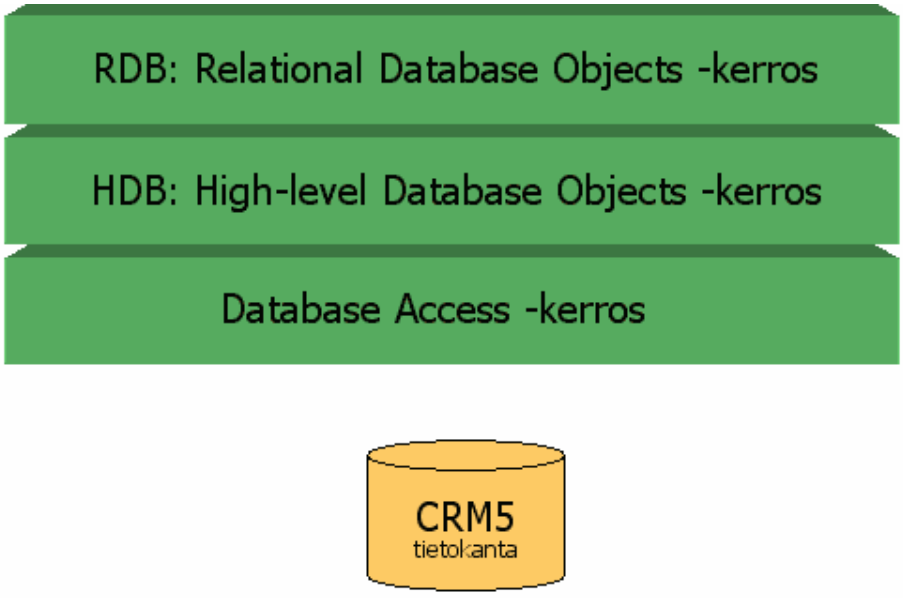
Käyttäjien mielestä Super Officen käyttöliittymä on selkeä ja helppokäyttöinen. Myös ohjelman matkakäyttöominaisuudet sekä sähköpostiohjelman (MS Outlook) kytkeminen Super Office -järjestelmään saivat kehuja käyttäjiltä. Puhelinjärjestelmän integrointi (puuttuu Super Office CRM -järjestelmästä) ja myyjien ryhmätyömahdollisuudet olivat joidenkin käyttäjien mielestä ominaisuuksia, jotka on huomioitu paremmin muissa CRM-järjestelmissä.

4. NetServer

SuperOffice NetServer on Visual Studio .NET -kehitysympäristöön toteutettu laajennos, jonka avulla SuperOffice CRM5 -järjestelmään voidaan kehittää ulkoisia sovelluksia, esimerkiksi internet-sovelluksia. Käytännössä NetServer on joukko DLL-kirjastoja, jotka tarjoavat rajapinnan SuperOffice-tietokannan käsittelyyn.

4.1. Tietokanta-arkkitehtuuri

NetServer-rajapinnan ydin koostuu kolmesta kerroksesta, jotka muodostavat kokonaisuuden, jonka kautta SuperOffice-tietokannan käsittely tapahtuu. Kuvassa 3 on esitetty NetServer-rajapinnan kerrokset, joita käsitellään tarkemmin seuraavissa kappaleissa.



Kuva 3. NetServer-rajapinnan kerrokset [SuperOffice, 2003]

Database Access Layer on NetServer-rajapinnan alin kerros, joka pitää sisällään istuntojen (session) käsittelyn ja tietokannan autentikointimekanismit. Ja koska NetServer (kuten myös koko SuperOffice CRM5 –järjestelmä) tukee useita eri relaatiotietokantoja, täytyy järjestelmässä olla myös tuki eri SQL-versioiden murre-eroille, ja *Database Access Layer* huolehtii myös tästä. Kun käyttäjä tekee kyselyn, *Database Access Layer* konvertoi sen käytössä olevan tietokannan ymmärtämään SQL-kieliseen muotoon ja ajaa tietokantaan.

HDB eli *High-level Database Objects* –kerros pitää sisällään tietokannan taulujen rakennekuvauksen, huolehtii tietokannan eheydestä ja tarjoaa optimoituja hakurutiineja, jotka parantavat suorituskykyä. Jokaista taulua varten on oma olionsa, ja taulujen sarakkeet näkyvät olioiden atribuutteina. Myös tietokannasta haettavaa dataa käsitellään olioiden kautta: jokaista riviä vastaa olio, ja useita rivejä käsitellään oliokokoelmien avulla.

RDB eli *Relational Database Objects* -kerros kytkeytyy suoraan HDB:n päälle ja huolehtii osittain samoista rutiineista. Tämä NetServer-rajapinnan ylin kerros ei ole välttämätön sovelluskehityksen kannalta, koska monet rutiinit voidaan hoitaa myös HDB-tasolla. RDB-kerroksen lisäarvo on siinä, ettei kehittäjän tarvitse tuntea tietokannan relaatioiden välisiä suhteita (ainakaan kovin tarkasti), vaan RDB tarjoaa valmiita näkymiä tietokantaan, joissa data saattaa olla peräisin useasta eri taulusta.

4.2. Tietokantakyselyt

SuperOffice-tietokannan rakenne on melko monimutkainen ja sen käsittely suoraan SQL-kielen avulla olisi vaarallista tietokannan eheyden kannalta. Tästä syystä NetServer-rajapintaan on luotu oma kyselykieli, jonka avulla kannasta voidaan sekä hakea dataa että lisätä ja poistaa sitä turvallisesti. Tässä on kuitenkin rajapinnan suurin ongelma: oma kyselykieli tekee järjestelmästä huomattavasti vaikeamman oppia – ainakin ulkopuolisen kehittäjän näkökulmasta. Kehittäjältä vaaditaan sekä hyvää SQL-kielen osaamista että tämän erityispiirteisen oliokyselykielen hallitsemista.

Ohjelmakatkelmassa 1 on kuvattu kuvitteellinen kysely SuperOffice-tietokantaan sekä SQL-kielellä että NetServer-oliokyselykielellä. Vaikka kysely on yksinkertainen, näyttää se NetServer-kyselykielellä toteutettuna melko monimutkaiselta. Kun kyselyyn lisätään useita tauluja sekä lisää ehtoja ja vielä mahdollisia matemaattisia funktioita, NetServer-kyselystä muodostuu todella monimutkainen kokonaisuus.

```
-------------------------------------------------------
-- Luodaan perinteinen SQL-kysely
-------------------------------------------------------
SELECT T1.Field1, T1.Field2
FROM Table1 T1
WHERE T1.Field1 = 1
OR T1.Field2 LIKE 'A%'

' -----------------------------------------------------
' Luodaan SuperOffice NetServer -kysely (Visual Basic)
' -----------------------------------------------------
Dim T1 As Table1TableInfo
Dim sqlSelect As [Select]

T1 = TablesInfo.GetTable1TableInfo

sqlSelect = S.NewSelect
sqlSelect.ReturnFields.Add(T1.Field1, T1.Field2)
sqlSelect.Restriction = T1.Field1.Equal(S.Parameter(1)). _
Or(T1.Field2.Like(S.Parameter("A%")))
```

Ohjelmakatkelma 1. Tietokantakysely kahdella eri tavalla

4.3. Käyttöoikeudet ja tietoturva

SuperOffice-tietokanta ja NetServer-rajapinta pitävät sisällään tarkkaan määritellyt säännöt siitä, kuka saa käsitellä tietokannan sisältämää dataa ja kuinka datan käsittely tapahtuu. Sen lisäksi, että tietty käyttäjä saa lukea vain tiettyjen taulujen sisältämää dataa, voidaan datan luku rajoittaa koskemaan vain tiettyjä rivejä tietokannan tauluissa. Jos esimerkiksi käyttäjä tekee kyselyn asiakastauluun, NetServer jättää automaattisesti pois ne tulosjoukon rivit, joihin käyttäjällä ei ole oikeuksia. Nämä ominaisuudet ovat mielestäni erittäin käyttökelpoisia ja niiden toteutus hyvin onnistunut.

Käyttäjien kirjautuminen SuperOffice-tietokantaan NetServer-rajapinnan kautta sen sijaan vaikuttaa vähintäänkin kyseenalaiselta. Yhteyden muodostaminen ja käyttäjän autentikointi tapahtuu ilman minkäänlaista salausta, joten käyttäjätunnus ja salasana välitetään selväkielisenä. Saattaa kuitenkin olla, etten joko ymmärtänyt kirjautumismenettelyä täysin tai käyttämäni NetServer-kehitysversio on vanhentunut, ja uudessa versiossa myös suojaus on hoidettu paremmin. Tämä ei ole omaa asiantuntemusaluettani, eikä NetServer-dokumentaatio ollut riittävän kattava tietoturva-asioiden kannalta, joten tämän asian tarkempi selvitys täytyy tehdä myöhemmin, jos NetServer-sovelluksia aletaan oikeasti toteuttamaan.

4.4. Collaboration

NetServer Collaboration on joukko valmiiksi toteutettuja komponentteja, jotka on tarkoitettu helpottamaan ja nopeuttamaan NetServer-sovelluskehitystä. Nämä komponentit eivät pidä sisällään käyttöliittymää, mutta tarjoavat rajapinnan, jonka avulla Microsoft Visual Studio .NET:n kontrolleja voidaan sitoa Super Office –tietokannan dataan. Collaboration-komponenttien avulla Super Office -järjestelmän osia voidaan melko yksinkertaisesti sulauttaa muihin järjestelmiin: esimerkiksi jonkun toiminnanohjausjärjestelmän käyttöliittymässä voi olla komponentteja, joihin haetaan Super Office –tietokannasta dataa.

5. Yhteenveto

NetServer-rajapinta tarjoaa ulkopuolisille kehittäjille mahdollisuuden tehdä sovelluslaajennoksia SuperOffice-tietokantaan niin, että tietokannan eheys säilyy. Toteutus on mielestäni onnistunut, joskin monimutkainen ja vaatii runsaasti perehtymistä. Suurin yksittäinen ongelma on mielestäni se, ettei tietokantakyselyjä muodosteta suoraan SQL-kielen avulla, vaan rajapinnan omien kyselyolioiden kautta. Tämä nostaa kynnystä alkaa kehittämään kaupallisesti hyödynnettäviä sovelluksia NetServerin päälle.

Rajapinnan tietoturva-ominaisuudet jäivät hieman epäselväksi: vaikutti siltä, ettei käyttäjätunnuksia ja salasanoja salata mitenkään, vaan ne kulkevat järjestelmässä selväkielisinä. Saattaa tosin olla, että tähän on joko jo olemassa tai ainakin on tulossa korjaus – järjestelmän dokumentaatio oli tältä osin puutteellinen eikä esimerkkisovelluksistakaan löytynyt tähän asiaan ratkaisua.

Akateemisessa mielessä tämän tutkielman asiasisältö ei välttämättä ole kovin hedelmällinen, koska aihe käsitteli pientä ja rajattua erityisaluetta. Myöskään varsinaista aiempaa tutkimustietoa aiheesta ei ollut, ja tärkeimpänä lähdemateriaalina minulla oli sarja Power Point –esityksiä, jotka ovat peräisin SuperOffice ASA:n aiheesta pitämältä kurssilta. Kuitenkin sekä tekijälleen että tutkielman tilaajalle tästä selvityksestä on varmasti paljon hyötyä, joten tarpeellisuutensa ansiosta tämä tutkielma ja sen tuloksena syntynyt kartoitus NetServer-rajapinnasta on mielestäni onnistunut ja hyödyllinen.

Viiteluettelo

[DB-Manager, 2004] SuperOffice – DB-Manager Oy. http://www.dbmanager.fi (22.11.2004).

[Jantunen *et al.*, 2001] Mika Jantunen, Elina Mäkelä, Sirje Nikulainen, Mika Ollikainen ja Lauri Nousiainen, CRM-ratkaisujen valinta ja hyödyntäminen – tutkimusraportti. Market-Visio Oy, 2001.

[Martin, 2003] Hanna Martin, Liiketoiminnan sähköistäminen pk-yrityksessä : tapaustutkimus asiakkuudenhallinnan kehittämishankkeesta. Tampereen yliopisto, Kauppatieteiden laitos. Pro gradu –tutkielma, 2004.

[Pinomäki, 2004] Teemu Pinomäki (Idearitsa Oy), Asiakkuudenhallintajärjestelmien vertailu – haastattelututkimus 2004.

[Sinkkonen *et al.*, 2002] Irmeli Sinkkonen, Hannu Kuoppala, Jarmo Parkkinen ja Raino Vastamäki, *Käytettävyyden psykologia*. Edita Publishing Oy, Helsinki, 2002.

[SuperOffice, 2003] SuperOffice ASA, SuperOffice NetServer SDK Training seminar course material, February 2003.

[YSA, 2004] Yleinen suomalainen asiasanasto, VESA - verkkosanasto. http://vesa.lib.helsinki.fi/ysa (22.11.2004).

Älykodit tänään

Sami Summanen

Tiivistelmä.
Tieteiskirjallisuuden ja elokuvien ennustamat älykodit kehittyvät piilossa suurelta yleisöltä ja tekevät hitaasti mutta varmasti tuloaan yleisille markkinoille. Tässä tutkielmassa tutkitaan älykotien nykytilannetta erityisesti Suomessa ja älykotien yleistymisen jarruna olevia ongelmia. Dokumentissa käydään läpi älykodin määritelmä, keskeiset laitteet ja verkkoratkaisut sekä esitetään lyhyesti joitakin saatavilla olevia älykotiratkaisuja.

Avainsanat ja -sanonnat: Älykoti, digikoti, älykäs asumisympäristö, ubiquitous computing, älytalo.

CR-luokat: K.4.m

1. Johdanto

Tietokonekomponenttien pieneneminen, prosessointitehon ja muistikapasiteetin kasvaminen yhdessä langattomien verkkojen yleistymisen kanssa mahdollistavat ubiquitous computing- tai ns. läsnä-älysovellusten (jatkossa ubi-sovellusten) kehittämisen. Yhtenä merkittävänä läsnä-älyn sovelluskenttänä voidaan pitää tulevaisuuden älykkäitä työskentely- ja asumisympäristöjä.

Tieteiskirjallisuus ja tieteiselokuvat esittelevät usein näitä tulevaisuuden asumisympäristöjä. Tämä osaltaan totuttaa meitä muutokseen, joka muuttaa pysyvästi käsitystämme kodista. Muutos ei tule olemaan nopea vaan älykkäisiin asumisympäristöihin tai älykoteihin (joista joskus käytetään myös nimitystä digikoti tai älykäs koti) siirrytään vaiheittain, alkaen todennäköisesti erityisryhmien, kuten vammaisten ja vanhuksien, tarpeista sekä turvallisuuteen liittyvistä sovelluksista. Myöhemmin älykodin eri osat, esimerkiksi älykäs jääkaappi ja mediaterminaali, integroidaan käyttäen saatavilla olevia verkkoratkaisuja. Myöhemmin älykoti sisältää runsaasti itsenäisesti toimivia ja toisaalta verkottuneita laitteita, jotka kaikki osallaan pyrkivät helpottamaan asukkaiden jokapäiväistä elämää.

Tutkimusta älykodeista tehdään runsaasti niin Suomessa kuin maailmalla. Erittäin merkittävä maa älykotien kehityksessä on ollut Etelä-Korea, jossa paikallinen viestintäministeriö ja mm. Samsung ovat jo käynnistäneet älykotihankkeen, jonka tarkoituksena on saada kymmenen miljoonaa verkotettua älykotia rakennettua vuoteen 2007 mennessä. Samsung on muutenkin tunnetuista yrityksistä ehkä merkittävimpiä älykotihankkeiden uranuurtajia. Toinen mer-

kittävä yritys tällä alueella on Philips, jonka markkinat ovatkin enemmän painottuneet Eurooppaan. [Hintikka, 2004]

Tässä dokumentissa esitetään ensin älykodin määritelmä ja olennaiset elementit, joista tyypillisen älykodin voidaan olettaa rakentuvan. Tämän jälkeen pohditaan älykodin merkitystä – miksi älykäs asumisympäristö on ylipäänsä tarpeen. Neljännessä luvussa käsitellään joitakin kaupallisia ja vielä tutkimusvaiheessa olevia älykotitekniikoita ja projekteja. Viidennessä ja viimeisessä luvussa käsitellään älykkäässä asumisympäristössä kohdattavia ja sen suunnitteluun ja toteuttamiseen liittyviä ongelmia.

Älykodeissa myös olennaisena osana olevat ns. älymateriaalit on rajattu tämän tutkimuksen ulkopuolelle. Muiltakin osin tutkimuksen ulkopuolelle on jouduttu rajaamaan useita älykotihankkeita, tekniikoita ja tutkimustuloksia, johtuen niiden erittäin runsaasta määrästä.

2. Yleistä älykodista

2.1. Älykodin määritelmä

Älykoti on asumisympäristö, jonka erottaa normaalista kodista lukuisat tiettyjä tehtäviä hoitavat ”älykkäät” ja verkottuneet laitteet. Tässä älykkyydellä ei välttämättä tarkoiteta älykkyyttä tai tekoälykkyyttä sen varsinaisessa merkityksessä. Pelkät yksittäiset älykkäät laitteet eivät välttämättä tee vielä kodista älykästä, vaan älykkään ympäristön luomisessa tarvitaan laitteiden verkottamista toisiinsa mielekkäällä tavalla. Toiseksi kodin ns. älykkyys on vain suunnittelijoiden kehittelemää älykkyyttä – sääntöjä joiden mukaan laitteiden on tarkoitus toimia tietyissä tilanteissa. [Hyvönen et al, 2002]

Leppäsen [2001, s. 119] mukaan Mervi Lehto on määritellyt älykodin seuraavasti:

> Älykäs rakennus tarkoittaa kehittyneillä tietoteknisillä automaatio- ja tietoliikennejärjestelmillä varustettua rakennusta, jonka tarkoituksena on palvella mahdollisimman hyvin rakennuksessa tapahtuvia toimintoja.

Termi on käännös englanninkielen smart home -termistä, jota American Association of House Builders käytti ensimmäistä kertaa vuonna 1984. Termi viittaa enemmän ”fiksuuteen” kuin älykkyyteen, mutta suomessa ei ole eroteltu älyä ja ”fiksuutta”. Näin älykoti on terminä jäänyt käyttöön. [Jokinen, 2004]

2.2. Älykodin laitteista

Älykkäässä asumisympäristössä älykkäitä laitteita voivat olla periaatteessa mitkä tahansa nykyiset kodinkoneet ja muut sähköä hyödyntävät laitteet.

Älykkäät valaistusjärjestelmät ovat tieteiselokuvista erittäin tuttuja ilmestyksiä. Elokuvissa ja kirjallisuudessa käyttäjät tyypillisesti ohjaavat valaistusta yksinkertaisilla puhekomennoilla, vaikka muitakin valojen ja verhojen kontrollointimenetelmiä on kehitelty. Valaistuksen puheohjausta on tutkinut mm. Microsoft, joka ei ole yleisesti ottaen tunnettu tällaisista sovelluskohteista, mutta jota kuitenkin kiinnostaa varmasti tulevaisuudessa kasvavat älykoti ja erityisesti sen viihdepuolen markkinat. [Brumitt and Cadiz, 2001]

Suomessakin on jo saatavissa jääkaappeja, joissa on älyominaisuuksia. Kuluttajaversioissa ne tosin rajoittuvat kosketusnäytöllä varustettuihin www-selaimiin tai elektronisiin reseptikirjoihin.

Toinen tyypillinen keittiön kodinkone, jota voidaan vaivatta parannella älyominaisuuksilla, on hella. Hellat ovat tyypillisiä huolenaiheita jokapäiväisessä elämässä, koska useita ihmisiä vaivaa usein kysymys siitä, että tuliko hella sammutettua vai ei. Pienellä automaatiolla hella voidaan kuitenkin asettaa sammuttamaan itsensä automaattisesti tietyn ajan kuluttua ja näin voidaan rajoittaa virrankulutusta ja ehkäistä tulipaloriskejä. Samalla asujien huoli päälle jääneestä hellasta poistuu tai ainakin vähenee. [Leppänen, 2001]

Kehittyneempiäkin keittiölaitteita on tulossa. Whirlpoolin Polara-sarjan hellaan voi huoletta pistää aamulla helposti pilaantuvatkin ainekset ja ohjelmoida sen lämmittämään ruoka kotiintuloaikaan. Hella pitää ruoan sopivassa lämpötilassa, vaikka siinä olisi esimerkiksi maitoa tai muuta nopeasti pilaantuvaa ainesta. [Schaub, 2003]

Turvalaitteita, kuten varashälyttimiä, on jo ollut pitkään saatavilla Suomessakin, mutta niiden älykkyydestä voidaan olla montaa mieltä.

Pelkkä liikkeentunnistinjärjestelmä ei vielä tee varashälyttimestä älykästä. Mutta toisaalta tällainen yksinkertainenkin hälytinjärjestelmä helpottaa ja turvaa kodin asukkaiden elämää.

Turvallisuuden lisäksi kokemukset, elämykset ja viihde ovat myös keskeisessä roolissa kodin toimintasisältöjä ajatellen. Niin sanotut mediakeskukset ovat jo joissakin talouksissa arkipäivää. Nämä mediakeskukset integroivat kodin viihdeobjektit kuten videot, musiikin ja valokuvat yhteen järjestelmään sekä mahdollistavat näiden mediaelementtien esittämisen esimerkiksi eri huoneissa tai eri televisioissa tai muissa audiovisuaalisia toimintoja tukevissa järjestelmissä. [Falck, 2004]

Erilaiset kommunikaatiomenetelmät ovat myös erittäin keskeisiä tulevaisuuden älykodeissa. Perheen sisäisen viestinnän ja myös ulkomaailman välinen viestintä voi helpottua älykoteihin erityisesti kehitetyillä ratkaisuilla. Samaan ryhmään kuuluvat myös älykkäät muistutussovellukset, jotka voivat muistuttaa yksittäisiä käyttäjiä kännykän kalenterin kaltaisesti, mutta poistuttaessa

ulko-ovesta. Erityisiä tällaisista muistutussovelluksista, kuten tutkimustuotteesta Gate Reminder, tekee niiden suomat mahdollisuudet perheiden sisäisten muistutusten luomisessa, joita henkilökohtaiset kännykät eivät mahdollista. [Kim et al, 2004]

Suomalaisille rakas saunakaan ei välttämättä säästy älyominaisuuksien integroimiselta. Jo nyt Suomessa on markkinoilla kiukaita, jotka käyttäjä voi käynnistää puhelinsoitolla tai tekstiviestillä, vaikka olisi itse vasta matkalla kotiin. Mm. Saunamare[1] valmistaa kiukaita, joiden ohjausjärjestelmät tukevat liittymiä älykkääseen talotekniikkaan.

Käyttäjien tunnistusta varten kodeissa voi olla useita erilaisia ja osin päällekkäin toimivia tekniikoita. Käyttäjä voidaan tunnistaa kameran tai videokameran avulla kasvoista. Lisäksi käyttäjä voidaan tunnistaa puheesta, sormenjäljestä tai RFID-tunnisteita mukana kuljetettavista esineistä tai vaatteista. Nämä ovat perinteisiä tunnistusmenetelmiä, mutta älykoteihin on myös kehitetty esimerkiksi askellukseen pohjautuva autentikointimenetelmä, Smart Floor, joka tutkimusten valossa toimii todella erittäin luotettavasti huolimatta siitä, käyttävätkö asukkaat kenkiä vai eivät. [Orr and Abowd, 2000]

Näiden lisäksi älykoti saattaa parhaimmillaan sisältää satoja erilaisia sensoreita, jotka palvelevat useita käyttötarkoituksia aina turvallisuudesta ja kulunvalvonnasta valaistuksen hallintaan ja lämpötilan, ilmastoinnin sekä sähkönkäytön kontrollointiin. Periaatteessa ainoat rajat erilaisten laitteiden määrälle asettaa suunnittelijan näkemykset ja kuluttajan valinnat sekä todelliset tarpeet.

2.3. Verkoista

Saadakseen aikaan aidosti kommunikoivan ympäristön, joka voisi aistia esim. asukkaidensa liikkeet ja reagoida asiaankuuluvalla tavalla, pitää komponenttien välillä liikkua tietoa.

Perinteiset langalliset ratkaisut toimivat myös älykodeissa moitteetta. Erityisesti uusia taloja rakennettaessa kaapeloinnit usein rakennetaan jo varhaisessa vaiheessa riittävän kattaviksi, jos tiedetään, että kodista pitäisi tulla älykkääseen talotekniikkaan perustuva asuinympäristö. Langallisten verkkojen hyötyjä ovat edullisuus pienissä, yksinkertaisissa ympäristöissä ja turvallisuus.

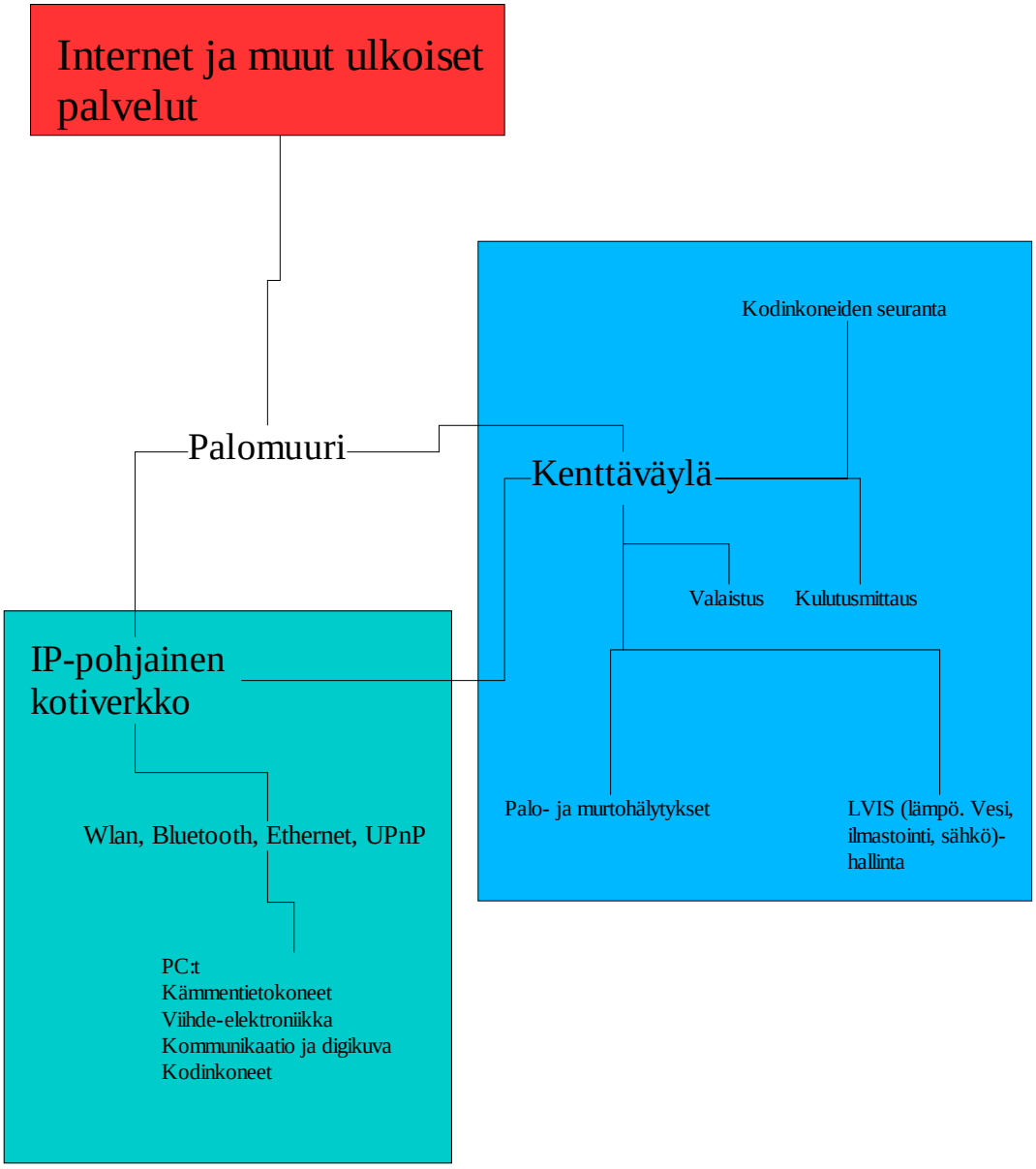
Langattomien tekniikoiden (WLAN, Bluetooth) yleistyessä kuitenkin on erittäin todennäköistä, että langattomat verkot tulevat huolehtimaan suurilta osin älykotien verkottuneisuudesta. Vanhoihin asuntoihin langallisten verkkojen rakentaminen on kallista ja joskus jopa mahdotonta. Langattomia verkkoja sen sijaan voidaan asentaa myös vanhoihin rakennuksiin. Lisäksi kiinteiden langallisten verkkojen joustamattomuutta ei esiinny esim. WLAN-verkoissa,

[1] http://www.saunapolar.fi/ (7.12.2004)

vaan näitä verkkoja tukevat laitteet voidaan asentaa käytännössä melko vapaasti huoneistoon. Langattomat verkot tuovat toisaalta mukanaan lisääntyneet turvallisuusriskit, joita käsitellään tarkemmin luvussa ”Ongelmia”.

Käytetään älykodissa sitten langallisia tai langattomia laitteiden yhdistämistekniikoita, on erittäin todennäköistä että ratkaisut tulevat olemaan vielä pitkälle tulevaisuuteen IP-pohjaisia johtuen tämän standardin laajasta tuesta ja valmistajien kokemuksesta ko. protokollasta. IP-protokollan käyttäminen verkoissa mahdollistaa hyvin langattomien ja langallisten verkkojen yhdistämisen, koska liikkuvat datavirrat eivät välitä siirtovälineestä, kunhan datavirta on IP-protokollan mukaista. Tarkemmin ilmaistuna tulevaisuudessa tullaan varmasti hyödyntämään UPnP (Universal Plug and Play) ja IPv6-standardia, jotka mahdollistaa käytännössä kaikille laitteille oman IP-osoitteen ja laitteiden konfiguroimisen automaattisesti verkkoon ilman käyttäjälle koituvaa ylimääräistä vaivaa asennustöistä. [Hintikka, 2004b]

Eriäviä mielipiteitäkin tulevaisuuden älykotien verkkoratkaisuista on. Esimerkiksi Risto Linturi, Suomessa tunnettu älykotien pioneeri, toteaa Hintikan [2004b] artikkelissa: ”Vallitsevana visiona on analogia-antureiden ja toimilaitteiden kytkeminen digitaalisten säätimien kautta kenttäväylään, joka puolestaan kytketään IP-verkkoon.” Kenttäväylällä Linturi tarkoittaa verkkoa, jonka tarkoituksena on pääasiassa siirtää tietoa rakennuksen tyypillisten kiinteiden osien välillä (valaistus, lämmitys, ilmastointi, sähkö) ja välittää tätä tietoa myös ohjaus- ja mittausjärjestelmille. [Hintikka, 2004b] Älykodin verkkoratkaisuja on hahmoteltu kuvassa 1.



Älykodin verkkoratkaisut. Kenttäväylä (LON) ja normaali IP-pohjainen kotiverkko voivat yhdistää kaikenlaista elektroniikkaa yhteiseen hallittavaan verkkoon. Kuva mukailee Tietokone-lehden [Hintikka, 2004b] kuvitusta.

Tiedonsiirtoverkkojen lisäksi tulee ottaa huomioon laitteiden erityiset energiansaantitarpeet. Vaikka langattomia verkkoja hyödyntävät laitteet olisivatkin vapaasti asennettavissa huoneistoissa, aiheuttaisi virranjakelu tiettyjä rajoituksia. Osa laitteista voi toimia paristoilla tai akuilla, mutta joidenkin laitteiden virrankulutus voi olla sitä luokkaa, että verkkovirtaa tarvitaan. Tällöin langaton tiedonsiirtoverkko menettää merkityksensä, koska sitä hyödyntävä laite olisi kuitenkin riippuvainen verkkovirrasta. Tämä voi olla erityisesti WLAN-laitteiden ongelmana, koska ko. laitteet tunnetusti kuluttavat paljon virtaa.

3. Miksi älykäs ympäristö?

Edellä esitellyt älykodin määritelmä ja esimerkit älykodin laitteista vastaavat jo jossain määräin kysymykseen, miksi älykäs ympäristö on tarpeen. Tässä luvussa kuitenkin pohditaan asiaa hieman enemmän todellisten käyttäjätarpeiden näkökulmasta. Älykodin määritelmä ottaa yleensä kantaa vain älykodin tekniseen puoleen ja tarpeet jäävät toiselle sijalle.

3.1. Vanhuksien ja vammaisten tarpeet

Teollisuusmaissa tapahtuva ikärakenteen nopea muutos aiheuttaa sen, että yhä useammat vanhukset joutuisivat nykyisellä kehitysvauhdilla siirtymään vanhainkoteihin. Tämä on sinänsä jo merkittävä kustannuskysymys yhteiskunnalle. Toiseksi vanhukset usein ovat mieltyneet asumisympäristöönsä eivätkä sieltä näin ollen haluaisi muuttaa pois. Tämä tilanne on innoittanut useita tutkijoita suunnittelemaan ja kehittelemään vanhuksien tarpeisiin sopivia älykoteja. Älykkäässä asumisympäristössä voisi vanhus ideaalitapauksessa viettää elämää joitakin vuosia ennen mahdollista laitoshoitoa. Älykodin sovellukset voivat ensinnäkin tarkkailla asukkaan terveydentilaa ja toimintaa niin lyhyellä kuin pitkällä aika välillä ja pitää samalla kirjaa esimerkiksi vireystasosta ja ruokailun säännöllisyydestä sekä muista terveyden ja sairauksien hoidon kannalta olennaisista asioista. Nämä tiedot voidaan edelleen lähettää tietoverkon kautta asukkaan omalle lääkärille tai perheenjäsenille, jotka voivat reagoida tilanteisiin sopivalla tavalla. Tällainen järjestelmä, joka on suunnattu erityisesti kriisitilanteita varten, on esim. LifeLine, joka tarkkailee asukkaan tilaa ja huomatessaan ongelmia, ilmoittaa käyttäjälle niistä. Jos käyttäjä ei reagoi järjestelmän kutsuihin, järjestelmä ottaa yhteyttä esimerkiksi hälytyspalveluun tai omaisiin. [Mynatt et al, 2000]

LifeLinen kaltaisia, mutta yksinkertaisempia kriisiapujärjestelmiä on myös Suomen markkinoilla, mutta näissä tyypillisenä rajoituksena on se, että käyttäjän on kyettävä itse hälyttämään apua paikalle esimerkiksi rannekkeessa olevaa hälytysnappia painamalla. Sairaskohtauksen sattuessa tämä voi osoittautua mahdottomuudeksi. (Ks. esim. [Korkiakoski, 2004])

Äärimmilleen vietynä terveydentarkkailu voi saada jo arveluttavia piirteitä. Twyford Bathrooms[2] -yrityksen konseptina olevassa WC-pöntössä on sensoreita, jotka analysoivat käyttäjänsä ulosteita ja voivat näin ollen välittää tietoja eteenpäin esimerkiksi omalääkärille, joka voi edelleen tarkkailla käyttäjänsä terveydentilaa pitkällä aikavälillä. Myös lyhyen aikavälin muutokset olisivat havaittavissa nopeasti. Vaikka tällainen terveydentilan analysointi voisi eh-

[2] http://www.twyfordbathrooms.com/ (7.12.2004)

käistä ongelmia etukäteen, voi kuitenkin kohderyhmän vastustus estää tällaisien tuotteiden yleistymisen. Älyä ei välttämättä haluta tuoda WC:hen, joka on tyypillisesti kodissa se viimeinen paikka, jossa voi viettää aikaa yksityisesti. [Forse, 2002]

Vanhuksien ja vammautuneiden tarpeet voivat erota joiltakin osin toisistaan, mutta osaltaan tarpeet voivat olla hyvin samankaltaisia ja sen vuoksi vanhuksille tai vammautuneille suunnattujen asumisympäristöjen tulisi olla pohjimmiltaan samankaltaisia. Kuitenkin erityisiä tarpeita varten tulisi olla myös mahdollisuuksia sovellusten pienelle räätälöinnille. (ks. esim. [Taylor et al, 2000])

Vanhentuvan sukupolven ja vammautuneiden ihmisten tarpeet innoittavat useita yrityksiä mukaan älykotien kehittämiseen. Tälle ryhmälle luodut tekniikat ja niiden mukanaan tuomat sovellukset voivat siirtyä myöhemmin myös ns. tavallisiin älykoteihin. Esimerkkinä tästä voi mainita erilaiset terveydentilaa tarkkailevat palvelut, jotka alun perin ovat nimenomaan kehitetty vanhempia tai sairauksista kärsiviä ihmisiä varten.

3.2. Muu turvallisuus

Kotia pidetään syystäkin tietyssä mielessä pyhänä paikkana. Se on tyypillisesti ollut se paikka, jossa voi olla täysin oma itsensä ja rentoutua. Olennaista rentoutumiselle on se, että asukas tuntee olonsa turvatuksi. Edellä mainittujen terveydentilaa edistävien ja sairauskohtauksien varalta toimivien järjestelmien lisäksi älykoti voi tarjota myös muunlaista turvallisuutta.

Varashälyttimet ovat tyypillisesti esimerkiksi liikkeentunnistimiin pohjautuvia, joten niissä ei periaatteessa ole kovinkaan paljoa ns. älykkyyttä. Sen sijaan varashälyttimet reagoivat muutoksiin ollessaan toimintakunnossa ja toimivat ennaltamääriteltyjen ohjeidensa mukaisesti soittaen esimerkiksi vartiointiliikkeeseen. Vaikka nämä hälyttimet eivät välttämättä olekaan kovin älykkäitä, voidaan ne kuitenkin lukea myös älykodin yhdeksi perusmekanismiksi, mitä tulee turvallisuuden takaamiseen.

Paloturvallisuus on Suomessa tiettyyn pisteeseen asti säädeltyä, mutta älykoti voi tuoda paloturvallisuuteenkin oman lisänsä. Normaalien palohälyttimien sijaan, palohälyttimet voisivat huolehtia hälytyksestä pelastusasemalle omatoimisesti tietyissä tilanteissa. Lisäksi ennaltaehkäisevänä toimena esimerkiksi älykkään ja itsensä, tilanteen niin vaatiessa, sulkevan hellan asentaminen talouteen voi omalta osaltaan laskea paloriskiä merkittävästi [Leppänen, 2001].

Turvallisuutta voidaan kohentaa myös energian, ilmastoinnin ja lämmityksen älykkäillä kontrollointimekanismeilla, jotka informoivat tarvittaessa käyttäjää ongelmista ja säätelevät talon elinoloja muokkaavia järjestelmiä tilanteen vaatimalla tavalla säilyttäen asuinympäristössä asukkaille edulliset elinolot op-

timaalisella energiankulutuksella. Tämä, yhdessä muiden laitteiden älykkäiden virransäästöominaisuuksien kanssa, voi osaltaan myös näkyä alentuneina sähkölaskuina älykotitalouksissa. [Hintikka, 2004]

3.3. Koti rentoutumisympäristönä

Nykyaikaisessa yhä kiihtyvässä työtahdissa ja mobiililaitteiden arkipäiväistyessä työt tulevat yhä useammin kotiin ja näin kodin ja työpaikan välinen ero hämärtyy. Tätä informaatioähkyä torjumaan on ehdotettu älykotiympäristöön tiedonsuodatusratkaisuja. Älykäs asuinympäristö voitaisiin asettaa hiljentämään puhelimet tai rajoittamaan ylipäänsä informaatiotulvaa illan pimetessä. Näin kodin asukkaat saisivat nauttia kotielämästä ilman jatkuvia yhteydenottoja esimerkiksi työpaikalta. Vastaavasti älykoti voisi myös ilmoittaa käyttäjälle ajankulusta, jotta rentoutuminen ei venyisi liian pitkäksi aiheuttaen omalla tavallaan stressiä. [Leppänen, 2001]

Olennaista rentoutumiselle on myös vapaa-ajan ja perheen yhdessä vietetyn ajan tarve. Tätä ongelmaa varten älykotiympäristöt pyrkivät automatisoimaan rutiininomaisia toimia, jotka kumuloituessaan voivat säästää merkittävästikin aikaa. Esimerkkinä tällaisista monien turhaksi ajankuluiksi mieltämistä toimista voi mainita imuroinnin ja ruohonleikkuun. joista huolehtimaan on jo kehitetty erilaisia robotteja.

3.4. Muita syitä

Älykotien kehitys on ainakin Suomessa vielä alussa. Kaikkia todellisia tarpeita on hankalaa havaita etukäteen, koska tekniset ratkaisutkaan eivät ole vielä päässeet yleistymään ja suuri osa ns. älykkäistä laitteista on usein vielä konseptivaiheessa.

Tekniikan kehittyessä on varmaa, että useat laitevalmistajat pyrkivät luomaan ihmisille uusia tarpeita tai ainakin vastaamaan jo ennalta tunnettuihin tarpeisiin yhä paremmilla sovelluksilla.

4. Esimerkkejä älykotien kokonaisratkaisuista ja standardeista

Tässä luvussa esitellään joitakin älykotien kokonaisratkaisuja. Aiheet ovat kuitenkin niin laajoja, että niitä ei tässä yhteydessä käsitellä kovin laajasti. Lisätietoja on saatavissa mainituilta verkkosivuilta.

4.1. X10[3] ja Jini[4]

X10 (ks. esim. [Hintikka, 2004b]) on jo melkein 30 vuotta vanha järjestelmä. X10 perustuu sähköverkon hyödyntämiseen laitteiden hallinnassa ja näin ollen sen asentaminen vaatii hieman uskallusta ja osaamista sähköverkkojen käsittelystä.

X10 järjestelmää hallitaan kaukosäätimellä ja sillä pystytään tekemään melko rajattuja toimenpiteitä kuten käynnistämään ja sammuttamaan laitteita. Järjestelmä soveltuu parhaiten juuri valaistuksen ja hyvin yksinkertaisten ympäristöjen luomiseen ja kontrolloimiseen.

Koska järjestelmällä on jo ikää ja muut tekniikat älykotien hallintaan alkavat yleistymään, ei X10:lle voida olettaa kovin pitkää elinikää. Pienien hallintaa vaativien ympäristöjen luomiseen se on kuitenkin yhä paikallaan edullisen hintansa vuoksi.

X-10:n lisäksi ohjelmointitaitoisille on olemassa Sun Microsystemsin luoma Jini-kotiverkko järjestelmä. Sitä tukee muutamat keskeiset laitevalmistajat, mutta mitään merkittäviä valmiita sovelluksia järjestelmään ei ole, joten ainakin hieman monimutkaisemmat järjestelmät käyttäjän on ohjelmoitava itse.

4.2. COBA-älykoti[5]

COBA (connected open building automation) (ks. esim. [Salminen, 2004] ja [Hintikka, 2004]) edustaa suomalaista osaamista älykoti-tekniikassa. COBA on avoin rajapinta, jonka pohjalta Lämpötalo[6] ja HomeSoft Oy[7] ovat tuotteistaneet kokonaisratkaisun älykotia haluavien tarpeeseen.

COBA yhdistää kaikki kiinteistön keskeiset hallintajärjestelmät yhteiseen käyttöliittymään. Näihin hallintajärjestelmiin kuuluu mm. lämmitys, jäähdytys, ilmanvaihto, valaistus, kulunvalvonta, kulutusmittaus, videovalvonta sekä murto-, palo- ja kosteushälytykset. Yhdistämisen seurauksena järjestelmät toimivat yhteen ongelmitta ja ohjautuvat tilannetta ja tarpeita vastaavasti. Tarveohjattu suunnittelu- ja toteutus perustuu avoimien rajapintojen tuelle ja yksinkertaiseen järjestelmämallinnukseen. [Salminen, 2004]

COBA-standardin mukaisesti, HomeSoft on jo kehittänyt viisi erilaista älykoti-profiilia, joista voidaan valita ostajan talouteen sopivin malli. Järjestelmällä päästään myös merkittäviin energiansäästöihin älykotiympäristön muiden tarpeiden tyydyttämisen ohessa. Laskelmien mukaan parhaimmillaan

---

[3] http://www.x10.com/

[4] http://www.jini.org/

[5] http://www.coba-group.com/

[6] http://www.lampotalo.com/

[7] http://www.homesoft.fi/

päästään jopa 30 % säästöihin virrankulutuksessa. Toistaiseksi järjestelmää kaupataan ainoastaan uusiin, rakennettaviin asuntoihin. [Hintikka, 2004, s. 21]

Lisätietoa lupaavasta COBA-standardista[8] saa mm. osoitteesta COBA-järjestelmän kotisivuilta.

Lukuisia muita älykotiratkaisujakin on olemassa (ks. esim. [Hintikka, 2004] ja [eKoti, 2004]).

5. Älykodin hallinnasta

Älykodin määritelmien mukaisesti älykkään asumisympäristön tarkoituksena on helpottaa asukkaan elämää mm. huolehtimalla tietyistä arkirutiineista ja muista aikaa kuluttavista toimista. Koska tarkoituksena on helpottaa elämää, on erittäin oleellista, että älykodin järjestelmien hallinta on yksinkertaista, vaivatonta ja toimintavarmaa kaikissa tilanteissa. Energia, ilmanvaihto, kodinkoneet, valaistus, verhojen hallinta, kulunvalvonta ja viihdesovellukset kuitenkin asettavat runsaasti haasteita intuitiivisien hallintajärjestelmien luomiselle. Tutkimusta älykotien hallintajärjestelmistä tehdään runsaasti, mutta ala on silti vielä uutta ja tutkiminen on hankalaa vähäisten kokemusten vuoksi.

Älykodin laitteiden hallinnan tulisi olla tavalliselle käyttäjälle mahdollisimman vaivatonta. Kolmiulotteisuutta hyödyntävät käyttöliittymät (ks. esim. [Shirenjini, 2004]) ovat tältä osin melko hyviä idealtaan, koska ne antavat helposti tulkittavan mallin ympäristössä, jossa laitteiden ID-koodit ja muut keskivertokäyttäjälle liian tekniset asiat voidaan unohtaa ja keskittyä olennaiseen. Shirenjinin mukaan järjestelmän etuna on uusien talojen CAD-mallien (3d) oleminen osa nykyaikaista rakennusten suunnittelustandardia. Käytännössä tällaisen 3d-mallin tuominen uuteen älykotiympäristöön pitäisi sujua vaivatta ja ainoastaan järjestelmään tuotavat uudet laitteet pitäisi rekisteröidä ympäristöön kuuluviksi. Tämän jälkeen hallinta toimisi valitsemalla reaalimaailmaa vastaava kohde 3d-mallista ja aukeavista valikoista käyttäjä voisi valita kohdekohtaisia toimintoja.

Graafisia käyttöliittymiä voi olla myös muunlaisia. Esimerkiksi Tampereen teknillisellä yliopistolla on tutkittu kolmen erilaisen käyttöliittymän toimivuutta älykotiympäristössä. PC, mediaterminaali ja kännykkä mahdollistavat erilaisia, sekä tietyssä mielessä rajaavat, käyttömahdollisuuksia. Tutkimuksessa tietokoneella oli merkittävin rooli, koska sillä pystyttiin kontrolloimaan käytännössä kaikkia älykodin toimintoja, jonka lisäksi PC:llä oli mahdollista kehittää eräänlaisia pienien tehtävien sarjoja eli automatisoida joitakin toimenpiteitä. Mediaterminaalin avulla käyttäjät pystyivät vastaavasti kontrolloimaan ainoas-

[8] http://www.coba-group.com/ (7.12.2004)

taan valoja ja verhoja. Kännykällä käyttäjät pystyivät mobiilisti, jopa kodin ulkopuolelta, kontrolloimaan valaistusta, verhoja ja tarkastelemaan esimerkiksi hellan tilannetta. Tämä tutkimus antaa hieman kuvastaa hyvin sitä, että tulevaisuuden älykodissa, interaktiokeinoja on useita ja ne täydentävät toisiaan. [Koskela and Väänänen-Vainio-Mattila, 2004]

Puhekäyttöliittymät (ks. esim. [Lines and Hone, 2002] ja [Brumitt and Cadiz, 2001]) ovat myös yksi merkittävä älykodin hallintamenetelmä. Puheella tapahtuvan ohjauksen selvä etu on sen vähäinen kognitiivinen kuorma käyttäjälle. Käyttäjä voi keskittyä esimerkiksi ostoskassien purkamiseen ja samalla antaa yksinkertaisia käskyjä älykodille. Puhekäyttöliittymää tulee myös ajatella toiseen suuntaan eli osana käyttäjän saamaa palautetta älykodin järjestelmiltä. Puheen tai äänen käyttäminen palautekanavana on myös tietyissä tilanteissa kohdallaan, koska se ei ole paikasta riippuvainen palautteenanto keino. Siinä missä esimerkiksi mediaterminaalissa näkyvä hälytys kiukaan ylikuumenemisesta ei välttämättä näy käyttäjälle, puhe voi kiinnittää käyttäjän huomion asiaan nopeasti ja paikkariippumattomasti. Lisäksi puhekäyttöliittymien yksi etu on luonnollisesti liikkumaan kykenemättömien ihmisten palvelu.

Todellisuudessa älykodin hallintajärjestelmät tuskin tulevat olemaan yksittäiseen hallintakanavaan, kuten puheeseen, rajoittuneita. Multimodaaliset, useita rinnakkaisia interaktiotapoja tukevat ratkaisut lienevät todennäköisempiä vaihtoehtoja ja käyttäjän kannalta ennen kaikkea joustavampia ja erilaisiin tilanteisiin paremmin sopivia.

Tästä multimodaalisuudesta esimerkkinä voi mainita Microsoftin rahoittaman tutkimuksen valaistuksen hallinnasta älykotiympäristössä. Tässä järjestelmässä yhdistettiin puheohjausta, käyttäjän sijaintitietoa ja osoittamisesta saatavaa kuvitteellista sensoridataa kontrolloimaan useista valoista koostuvan huoneen valaistusta. Kuvitteellista tästä teki se, että käyttäjät eivät tienneet, että todellisuudessa valaistusta käyttivät tutkijat, eikä älykodin järjestelmät, käyttäjien toimien perusteella. Vaikka tällaisen toimivan järjestelmän luominen vaatisi vielä nykyisellään melkoisesti resursseja, saatiin tästä tutkimuksesta joitakin viitteitä siitä, miten käyttäjälle voisi olla luonnollista kontrolloida ympäristön valaistusta. [Brumitt and Cadiz, 2001]

Perinteisten graafisten ja puhe- tai jopa elekäyttöliittymien lisäksi asunnon hallintaa voidaan helpottaa myös muilla innovatiivisilla ratkaisuilla. Hyvänä esimerkkinä tästä on syytä mainita jo edellä mainittu Smart Floor, joka tunnistaa käyttäjänsä askelluksen perusteella ja vielä melko luotettavasti. Tällainen lattialaatta, joka huolehtii käyttäjän tunnistamisesta pelkän askelluksen perusteella, ei vaadi käyttäjältä minkäänlaista ylimääräistä suoriutumista, kuten esi-

merkiksi sormenjälkitunnistimen tai kasvotunnistimen käyttö edellyttäisi. [Orr and Abowd, 2000]

Toisena merkittävänä käyttäjää rasittamattomana ympäristön hallintamenetelmänä voi mainita ubi-sovelluksille tyypillisen ympäristön havaitsemisesta (ks. esim. [Meyer and Rakotonirainy, 2003]) saatavan datan käsittelyn siten, että siitä saadaan seulottua mielekkäitä tapahtumaketjuja, joiden avulla mm. automaatioon voidaan vaikuttaa. Järjestelmä voisi esimerkiksi havaita käyttäjälle arkipäivisin toistuvat aamurutiinit ja käynnistää esimerkiksi asiaankuuluvalla hetkellä oikeissa paikoissa, käynnistää kenties television käyttäjän lempikanavalta, soittaa musiikkia ja asettaa vedenkeittimen päälle. Tällaisien ketjujen automatisointi tosin vaatisi joko tekoälyllisien sovelluksien kehittämistä tai todellisia ponnisteluita ympäristön suunnittelijoilta.

6. Ongelmia

Älykotien yleistymisen myötä saadaan havaita sellaisia ongelmia, joita tässä vaiheessa kehitystä ei edes osata arvioida. Tähän on kuitenkin koottu joitakin yleisesti tiedossa olevia ongelmia ja arviointeja tulevaisuuden ongelmista.

6.1. Turvallisuus

Vaikka turvallisuusnäkökohdat ovat juuri älykotien markkinoinnissa ja suunnittelussa keskeisellä sijalla, on ironista, että turvallisuus on yksi merkittävimmistä ongelmista (ks. mm. [Hintikka, 2004]) älykotien yleistymisen ja suunnittelun esteenä.

Yhtenä turvallisuusriskinä voidaan pitää laitteiden vikaantumisista aiheutuvia ongelmia. Tämän vuoksi laitteiden toimintavarmuus on keskeisellä sijalla niiden suunnittelemisessa ja toteuttamisessa.

Toinen ongelmia todennäköisesti aiheuttava asia on nykyäänkin PC:n käyttäjiä riivaavat turvariskit: virukset ja hakkerit. Älykotien yleistymisen myötä voidaan olettaa hakkerien siirtyvän hyödyntämään osaamistaan älykotiympäristöihin tunkeutumisina ja jopa sabotointeina. Erilaiset älykotien suositut pakettiratkaisut voivat yleistymisen johdosta kohdata ongelmia myös tietoturva-aukkoja hyödyntävien virusten, matojen tai troijalaisten muodossa. Kauhuskenaarioita voi kehitellä miltei loputtomiin, sillä langattomat verkot ja yleisenä standardina oleva IP-protokolla ovat omiaan lisäämään älykoteihin kohdistuvia riskejä. Pahimmillaan älykoti voisi muuttua jopa käyttäjälleen vaaralliseksi tai muuten vain käyttökelvottomaksi.

Nämä turvallisuusriskit on otettava huomioon älykotien verkkoratkaisuja ja laitteita suunnitellessa sekä toteuttamisessa.

6.2. Yksityisyys

Ubi-ympäristöissä, joissa lukuisat sensorit tarkkailevat ympäristöä ja käyttäjiä sekä tallentavat tietoja näistä, herää myös kysymys yksityisyyden turvasta. Tietokantoihin voi kertyä pahimmillaan huomattava määrä tietoa, josta voi olla ulkopuolisille tahoille joutuessaan huomattavaakin vahinkoa järjestelmän käyttäjälle. Meyer ja Rakotonirainy [2003, s. 5] mainitsevat, että olennaista on informaatiomäärän minimointi jo sensoritasolla ja sitä pitäisi säilyttää vain niin kauan kuin sille on mielekästä käyttöä. Tällä menetelmällä ylimääräistä tietoa ei kertyisi tietokantoihin, jotka voivat olla muuten tietovuodoille alttiita. Lisäksi tiedon vuotaminen ympäristöön pitäisi poistaa tai ainakin minimoida, mutta nykyisissä langattomissa verkoissa tämä voi olla melkein mahdotonta.

Ongelmaa ovat lähestyneet myös Hong ja muut [2004] kehittämällä ns. yksityisyyden riskimalleja. Näiden mallien perusteella suunnittelijat voivat jo älykkään ympäristön suunnitteluvaiheessa pohtia olennaisia kysymyksiä, kuten ketkä ovat järjestelmän käyttäjiä ja miten informaatiota kerätään. Näitä kysymyslistoja hyödyntämällä, ja niitä täydentämällä, ongelmia voidaan ratkaista jo ennakolta. Nämä kysymyslistat, yhdessä Meyerin ja Rakotonirainyn [2003, s. 5] mainitseman sensoridatan minimoinnin kanssa, voivat ainakin vähentää älykotien yksityisyydelle aiheuttavia riskejä.

Tässä tulee kuitenkin ottaa huomioon myös se, että osa käyttäjistä on valmiita luovuttamaan osia yksityisyydestään saadakseen mahdollisesti parempia palveluita tai mukavamman elämisympäristön.

6.3. Standardit

Älykotien yleistyminen vaatii standardeja ja tällä hetkellä on saatavilla useita osin kilpailevia standardeja. Tilanne on hieman sama kuin tallentavien DVD-levyjen markkinoilla, mutta huomattavasti suuremmassa mittakaavassa. Sijoitusriskit tietyn standardin mukaisiin ratkaisuihin ovat merkittäviä kuluttajalle, ja koska älykodit eivät ole vielä yleistyneet merkittävästi, ei kuluttajille ole saatavilla selkeää informaatiota eri standardeista etuineen ja haittoineen. Kuluttajan pitää näissä asioissa ottaa asioista itse selvää, jos haluaa älykodin omistajaksi.

6.4. Virrankulutus

Langattomat ratkaisut mahdollistavat joustavia ratkaisuja älykotien rakentamiseen myös vanhoihin asuntoihin, joissa kaapelointien tekeminen voisi olla jopa kiellettyä. Langattomuuden kääntöpuolena seuraa virrankulutus ja ennen kaikkea kysymys siitä, miten virransaanti mahdollistetaan langattomille komponenteille.

Yhtenä vaihtoehtona on verkkovirta, mutta tällöin langattomuuden suomat edut on unohdettava. Toisena vaihtoehtona ovat erilaiset paristot ja akut. Näiden ongelmana on taas niiden vaatima huolenpito. Jos asunnossa on esimerkiksi 20 langatonta sensoria tai joitakin muita paristoilla toimivia laitteita, voi käyttäjälle aiheutuva ylläpitokuorma olla jopa näistä laitteista saatavia etuja suurempi.

Yhä paranevat akkutekniikat auttavat asiaa, mutta eivät täysin poista virrankulutuksen tuomia ongelmia. Laitteista pitää vain yksinkertaisesti kehittää mahdollisimman vähän virtaa käyttäviä ja samalla akkujen tai muiden virranlähteiden pitää parantua merkittävästi nykyisestä.

6.5. Hinta

Nykyiset älykotijärjestelmät alkavat olla jo hinnoiltaan kohtuullisen hintaisia. Silti investointi kattaviin älykotiratkaisuihin vaatii rahaa merkittävästi. Esimerkiksi Zürichin lähelle rakennetussa älykodissa perustekniikan osalta kustannukset olivat normaaliin taloon verrattuna 2-4 % suurempia [Hintikka 2004, s. 20]. Tämä ei vielä merkittävästi älykodin hintaa nosta, mutta yleisesti ottaen esimerkiksi älykkäät jääkaapit ja muut kodinkoneet ovat merkittävästi normaaliversioita kalliimpia. Ennen kaikkea kysymys ei ole välttämättä pelkästä hinnasta, vaan siitä kokevatko käyttäjät saavansa vastinetta investoinneilleen.

6.6. Käyttäjien tarpeet ja ennakkoluulot

Käyttäjien tarpeet vaihtelevat runsaasti. Vanhuksilla ja nuorilla voi olla täysin erilaisia tarpeita, mutta omista tottumuksista johtuen, tarpeet voivat esimerkiksi erota merkittävästi oman ikäluokan keskiarvosta. Täysin räätälöityä talopakettia voi olla hankala saada, mutta älykotien valmistajat alkavat reagoimaan yksilöllisiin tarpeisiin, kun älykodit alkavat lisääntymään. (Lisätietoja älykotien räätälöinnistä esim. [Taylor et al, 2000])

Ihmisillä on tyypillisesti terveitä ennakkoluuloja uusia asioita kohtaan. Älykodit eivät ole tästä poikkeuksia. Niiden tarpeellisuutta voidaan epäillä ja tekniikalle ei haluta luovuttaa kontrollia liiaksi, koska sen toimintavarmuudesta ei voida olla varmoja. Tämä voi olla hidasteena, muttei esteenä, älykotien yleistymiselle. Suomen tapauksessa täytyy kuitenkin todeta, että Suomi on ollut mobiilitekniikan kärkimaita jo pitkään ja täällä ei tyypillistä uuden tekniikan vieroksumista esiinny siinä määrin kuin esimerkiksi Englannissa. Näin ollen Suomi voi hyvin olla merkittävien älykoti-projektien testikenttänä.

6.7. Suunnittelun hankaluudet

Älykotitekniikat ovat verrattain uusi markkina-alue ja näin ollen tutkimus on vielä useisiin muihin aloihin nähden vähäistä. Tämä, yhdessä älykotien moni-

tahoisten käyttötapojen kanssa, aiheuttaa suunnittelijoille päänvaivaa. Myöskään ulkomailta kerätyt positiiviset kokemukset tietyistä tekniikoista eivät välttämättä toimi toisella puolella maapalloa johtuen kulttuurienerojen mukanaan tuomista käytettävyysongelmista. Näin ollen esimerkiksi Samsungin laajat älykotihankkeet Etelä-Koreassa eivät ole sellaisenaan käyttökelpoisia Suomessa. Myös älykodin tekniikat pitää siis lokalisoida. Kulttuurieroista havainnollisen esimerkin antoi mm. Gate Reminder -tutkimus [Kim et al, 2004b]. Tutkimuksessa huomattiin, että kenkiin sisälletyt RFID-tunnisteet ja niiden hyödyntäminen Gate Reminder -sovelluksessa, ja erityisesti sen käyttäjien tunnistamisessa, voi olla eri kulttuureissa ongelmallista. Näin ollen käyttäjien tunnistaminen pitäisi mahdollistaa muulla tavalla kuin kengissä olevilla tunnisteilla.

Suunnittelussa tulee myös ottaa huomioon eettiset ongelmat. Vaikka älykotitutkimusta vauhdittaneet vanhuksille ja vammautuneille suunnatut älykotihankkeet ovatkin varmasti lähtökohdiltaan hyvää tarkoittavia, voi asiaa pohtia myös siltä kannalta, että onko tarkoituksena sulkea vanhukset ja vammautuneet neljän seinän sisälle, kotiinsa, ja säästää omaiset huolenpidolta ja vaivalta.

Tekniikalla on taipumuksena hankaloittaa asioita ja vaikka ratkaisut olisivat kuinka hyvin suunniteltuja, niin vikoja kuitenkin ilmenee ja asuinympäristössä ne voivat olla pahimmillaan kohtalokkaita tai ainakin äärimmäisen ärsyttäviä keskivertokäyttäjälle. Normaalin tietokoneen parissa useat käyttäjät jo saavat työympäristössä ratkoa tarpeeksi erilaisia tietoteknisiä ongelmia ja jos niitä ongelmia ilmenee jossain täysin itsestään selvässä laitteessa, kuten jääkaapissa, voi älykodin tarkoitus elämää helpottavana järjestelmänä helposti romuttua, ainakin käyttäjien mielessä. Blackwell [2004] otti erinomaisesti kantaa osittain juuri tätä sivuavaan aiheeseen: kehittyykö älykotien haltijoista älykodin ohjelmistoasiantuntijoita vai muodostuuko täysin uusia ammattiryhmiä huolehtimaan älykotien perustoiminnallisuudesta.

Hintikka [2004] herättää kysymyksen myös älykodin muuttamisesta uuteen sijaintiin. Kunnollista kysymykseen ei vielä ole, mutta se on olennainen arvioidessa muuttojen kustannuskysymyksiä ja älykotien todellista arvoa. Kalliit investoinnit ja mahdollisesti omia tarpeita varten räätälöidyt ratkaisut tuntuisivat itsestään selviltä muuttaa uuteen sijaintiin tarvittaessa, mutta onko se teknisesti mahdollista, riippuu täysin älykodin käyttämistä tekniikoista.

7. Yhteenveto

Toistaiseksi älykotien rakentaminen ei ole helppoa, ja jos sellaista mielii Suomessa saada, ovat mahdollisuudet melko rajattuja. Ongelmana ei enää nykyisin ole niinkään saatavuus vaan se, että älykoteja ei vielä markkinoida suurelle yleisölle. Ostajan pitää siis tietää kohtuullisen hyvin mitä haluaa ja mistä tuot-

teita saa. COBA-rajapintaan tai muihin vastaaviin tekniikoihin perustuvien älykotipakettien tulevaisuus voi riippua paljolti myös markkinoinnista ja mediassa saamasta ilmaisesta mainonnasta. Älykotien kokonaisasiantuntemusta on vähäisesti, vaikka eri yritykset kyllä tuntevat omat osa-alueensa varsin hyvin. Integroituja älykotiratkaisuja on hankala tilata, koska mitään keskeistä pakettiratkaisujen valmistajaa ei ole ainakaan julkisuuteen asti tuotu.

Tällä hetkellä älykodin hankkiminen vaatii käyttäjältä huomattavaa panostusta. Käyttäjältä edellytetään uudehkoa asuntoa, internet-yhteyksiä, paljon rahaa ja ennen kaikkea ennakkoluulotonta asennetta uuden tekniikan mukanaan tuoviin haasteisiin.

Viiteluettelo

[Blackwell, 2004] Alan F. Blackwell, End-user developers at home, *Communications of the ACM* **47**, 9 (September 2004), 65-66.

[Brumitt and Cadiz, 2001] Barry Brumitt and JJ Cadiz, "Let There Be Light" examining interfaces for homes of the future, Microsoft Research, 2001, http://www.research.microsoft.com/research/coet/Homes/INTERACT2001/paper.pdf (6.10.2004)

[COBA, 2004] COBA: Connected Open Building Automation – kiinteistön käyttöjärjestelmän standardointiprojekti, http://www.coba-group.com/index_fi.shtml (6.10.2004).

[eKoti, 2004] eKoti-hankkeen kotisivu, Tampereen teknillinen yliopisto, Elektroniikan laitos, http://www.ele.tut.fi/research/personalelectronics/projects/ekoti_03/ (6.10.2004)

[Falck, 2004] Kennet Falck, Viihteen ytimenä mediakeskus, *Tietokone* 12 (lokakuu 2004), 22-24.

[Forse, 2002] Heather Forse, Sanitary vision of the future, 2002, http://www.chennaionline.com/health/homearticles/2002/sanitary.asp (7.12.2004)

[Future Home-hanke] VTT:n Future Home-hanke, http://www.vtt.fi/rte/projects/yki4/futurehome.htm

[Hintikka, 2004] Kari A. Hintikka, Elämää digikodissa, *Tietokone* **12** (lokakuu 2004), 18-21.

[Hintikka, 2004b] Kari A. Hintikka, Digikodin verkkoratkaisut, *Tietokone* **12** (lokakuu 2004), 25-27.

[Hong et al., 2004] Jason I. Hong, Jennifer D. Ng, Scott Lederer and James A. Landay, Privacy risk models for designing privacy-sensitive ubiquitous computing systems, In: *Proc. of the 2004 Conference on Designing Interactive Systems: Processes, Practices, Methods, and Techniques*, ACM Press, New York, 2004, 91-100.

[Hyvönen et al., 2002] Pirkko Hyvönen, Katja Järvinen ja Asko Soukka, Älykoti mahdollisuutena – Näkökulmia älykodin käsitteeseen, harjoitustyöraportti, http://www.cs.joensuu.fi/pages/marjomaa/oppiva/parhaat/alykoti/alykoti.html (25.10.2004).

[Jokinen, 2004] Marika Jokinen, *Onko älykoti totta vai utopiaa?*, 2004, http://www.tut.fi/dmi/projects/tatu/MarikaJ.pdf (7.12.2004).

[Kim et al., 2004] Sung Woo Kim, Min Chui Kim, Sang Hyun Park, Young Kyu Jin and Woo Sik Choi, Gate reminder: a design case of a smart reminder, In: *Proc. of the 2004 Conference on Designing Interactive Systems: Processes, Practices, Methods, and Techniques*, ACM Press, New York, 2004, 81-90.

[Korkiakoski, 2004] Tiina Korkiakoski, Geronteknologia luo välineitä ikävää vanhuutta vastaan, http://www.elsi.net/articles/gerontek.htm (7.12.2004).

[Koskela and Väänänen-Vainio-Mattila, 2004] Tiiu Koskela and Kaisa Väänänen-Vainio-Mattila, Evolution towards smart home environments: empirical evaluation of three user interfaces, *Personal and Ubiquitous Computing*, 8, 3-4 (July 2004), 234-240.

[Leppänen, 2001] Sanna Leppänen, Älykäs koti, *Digitalisoituvan viestinnän monet kasvot*, Teknologiakatsaus 118/2001, Tekes, Helsinki, 2001, 113-131, http://www.tekes.fi/julkaisut/Digitalisoituvan_viestinnan_monet_kasvot.pdf (25.10.2004).

[Lines and Hone, 2002] Lorna Lines and Kate S. Hone, Older adults' evaluations of speech output, In: *Proc. of the 5th International ACM Conference on Assistive Technologies*, ACM Press, New York, 2002, 170-177.

[Meyer and Rakotonirainy, 2003] Sven Meyer and Andry Rakotonirainy, A survey of research on context-aware homes, In: *Proc. of the Australasian Information Security Workshop Conference on ACSW Frontiers 2003* 21, Australian Computer Society, Inc., 2003, 159-168.

[Mynatt, Essa and Rogers, 2000] Elizabeth D. Mynatt, Irfan Essa and Wendy Rogers, Increasing the opportunities for aging in place, In: *Proc. on the 2000 Conference on Universal Usability*, ACM Press, New York, 2004, 65-71.

[Orr and Abowd, 2000] Robert J. Orr and Gregory D. Abowd, The smart floor: a mechanism for natural user identification and tracking, In: *CHI '00 Extended Abstracts on Human Factors in Computing Systems*, ACM Press, New York, 2000, 275-276.

[Salminen, 2004] Oili Salminen, Massaräätälöity, hartiapankkirakentajallekin soveltuva Älykäs talo- ja turvatekniikka sarjatuotantoon, 2004, http://www.tieke.fi/online/uutiset.nsf/SFrameset/aloitus?OPENDOCUMENT&D=DUID/3BBACFBC89E29146C2256F15004C63CE (7.12.2004).

[Schaub, 2003] Charlyne Varkonyi Schaub, *Whirlpool's new Polara range allows cooks to change their plans, 2003,* http://www.internethomealliance.com/press_room/in_the_news/docs/South_Florida_Sun_Sentinal.May_23.pdf (7.12.2004)

[Shirenjini, 2004] Ali. A. Nazari Shirehjini, A novel interaction metaphor for personal environment control: direct manipulation of physical environment based on 3D visualization, *Computer & Graphics* **28** (2004), 667–675.

[Taylor at al, 2000] Bruce Taylor, Guy Dewsbury and Edge, Designing safe smart home systems for vulnerable people, Technical Report, 2001, http://www.smartthinking.ukideas.com/_DIRC.pdf (6.10.2004).

Exploitation of non-speech audio in user interfaces

Leena Vesterinen

Abstract

This paper provides an insight to the non-speech audio and explains its use in every day life situations. The use of non-speech audio interfaces is in increasing demand globally. The demand is being fuelled by visually disabled users, those whose working conditions or protective clothing mean that a keyboard or screen cannot be used, hands and eyes busy situations, navigation systems, telephone based services, computer games, process control and flight management systems, to mention some examples.

Keywords: Auditory Interfaces, non-speech audio, earcons, auditory icons, music.
CR- Categories: H.5.2

1. Introduction

This paper introduces the basic concepts of exploitation of non-speech audio in interfaces. Just as we receive information from the computer in a text format, or as images and speech, we are able to receive information in a non-speech audio format. For example, information in an application could be presented in a form of musical sounds or earcons, synthetic sounds.

The research of non-speech audio was started many decades ago, but it is since 1970's that we have seen some remarkable development in this field. These simple applications provided services, by transmitting information to the user by simple computer processed sounds.

Also the development in computer hardware technology has increased the possibility of applying non-speech audio technology to everyday applications.

Non-speech audio is still a new field of study in human computer interaction technology but it is gaining in interest in research, technology and business markets. This phenomenon is due to the increasing amount of visually impaired computer users, navigation systems, telephone based services, computer games, process control, flight management systems and hands and eyes busy situations.

One of the most interesting properties in the non-speech audio interfaces is the ability of most users to monitor simultaneously a number of non-speech audio signals while performing a motor or visual task. [Buxton et al., 1994]

2. Auralization – Representation of Data as Sound

Auralization is to represent complex data in sounds, by mapping the parameters of the data to the parameters of sound.

Non-speech audio presents data in a form of synthetic sounds – earcons, natural sounds – auditory icons or music. These sounds are used to aid in motor or visualization tasks in the user interfaces.

One of the most important factors in auralization, are the methods and parameters used in the design and development of such applications. These parameters are, for example, pitch, timbre, loudness and rhythm of the sound.

Information auralization can be used to provide overview of large data sets, which are difficult to express using speech.

2.1. History of auralization

In 1970, the American composer Charles Dodge wrote a piece of computer music called "The Earth's Magnetic Field". In his composition the sounds correspond to the magnetic activity for the earth. Though, the scientific version of representing data in sounds was represented by S. D. Speeth in 1961. [Buxton et al., 1994]

Later, in 1970's Max Mathews and colleagues used graphics and sound to present up to five dimensions of data. Mathews proposed a number of aural dimensions: loudness, pitch, vibrato, rate of modulation, aspects of timbre, and tempo. He was also interested in using chords, stereo and localization for discrete events. [Buxton et al., 1994]

Mathews was interested in presenting data so that as many people as possible could benefit from it. He wanted to follow the idea of 'what one hears is what the other one can see'. [Buxton et al., 1994]

These aural dimensions were important parts of the development of a non-speech audio in interfaces. In all cases, the early work of scientists in representing data as sounds were concerned with the human ability to find patterns in a complex set of data. They were focused upon making use of the fact that we easily make a distinction between a variety of sounds and this gestalt listening could be used in data exploration. [Buxton et al., 1994]

This started the active research for the perceptual abilities of human ear; which is the 'base element' for the non-speech audio interfaces. The design in the non-speech interfaces has to consider the users' capabilities of hearing, memory and recognition for the different sounds.

2.2. Hearing and Psychoacoustics

Why do we consider the psychoacoustics, perception of sounds when exploiting the non-speech audio in interfaces?

Human can hear sounds in 20 Hz to 20 kHz frequency range and differences of sounds around 1.5 Hz. It is also important to consider the fact that human hearing loss occurs with age. In age of 50, the maximum hearing frequency rate is 14 kHz. In age of 70, the maximum hearing frequency rate is 10 kHz. [Jauhiainen, 1995]

We live in a multiple auditory scene, where we can recognise various sounds simultaneously. We listen and identify sounds linking them to some particular sounds. Identification of sounds is due to how similar and correlative their unique sound objects are by their pitch, volume and other characteristics. [Jauhiainen, 1995]

In the 'audio space' we can recognise sounds, for example, by the difference of their pitch or loudness. The recognition of sounds demonstrates the fine sensitivity of hearing, primarily when comparing the sounds and for recognition of loudness and pitch. Categorical recognition of sounds describes a selective talent to recognise sounds as concepts and apply them in communication. It is not known if the recognition is due to matured hearing and learning, or possibly natural talent. [Jauhiainen, 1995]

3. Non-Speech Audio in Interfaces

The usage of non-speech audio could be considered when user's hands are involved in a physical task, to support the visual interface, to aid visually impaired users, to aid when the user is occupied by activities which that demand high level of attention or if the users are mobile.

For example, one of the main deprivations caused by blindness is the problem of accessing the information. Almost always, visualization is a fundamental method for understanding information in an application but at the present time these applications has not been implemented in a way that they would be accessible by blind people. [Holland et al., 2002]

Current techniques for displaying information non-visually rely on synthetic speech, natural sounds or music. For example, a matrix read out for a blind person would be just hearing rows of numbers spoken, one after another. This is extremely slow. Properties of human short-term memory also mean that listeners are unable to hold in mind enough information to make any non-trivial observations. The short term memory becomes overloaded. [Brewster, 2002]

Sounds exploited in the applications, should be presented as a collection of sounds. The sounds used, should be easy to recognise and differentiated from each other. This can be done by altering the pitch, rhythm, timbre and loudness of the sound.

3.1. Auditory Icons – Natural Sounds

Auditory icons in the interfaces are audible events and their dimensions are encoded as sounds. These sounds are based on real world events, which are familiar to us from our environment. Auditory icons convey a meaning which is derived from their origin rather than acoustic properties.

For instance, if the material of a sound-producing event is used to stand for the type of object, all auditory icons regarding that type of object would use sounds made by that kind of material. Text files might always sound wooden, whether they are selected, copied or deleted. Because auditory icons rely on everyday sounds, it is relatively easy to make them match with existing graphic interfaces. [Buxton et al., 1994]

The use of auditory icons is related to cultural conventions. If they are misused, they can easily convey misleading information. Auditory icons are typically recorded sound samples, which limits parameterization. [Turunen and Salonen, 2004]

3.2. Earcons – Synthetic Sounds

Earcons are known as structured audio messages and are abstract, musical tones, which can be used in combination to create sound messages to represent parts of an interface. As already stated, the different levels of sounds are created by manipulating the parameters of sound.

Earcon design is based on a motive. The motive must be meaningful for some particular action in the user interface. These are short rhythmic sequences that can be combined in different ways. Combinations of earcons allow producing more complex messages. This can be done by using more complex

manipulations of the sound parameters. Earcons can be produced for a set of operations, such as 'open', 'close', 'file' and 'program'. These earcons can be also combined to create, for example, earcons for 'close file' or 'open program'. [Brewster, 1993]

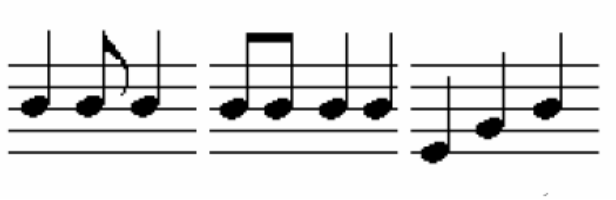


Figure 1: Rhythm and pitch structures for Folder, File and Open used in one of the Brewster et al. [1993] experiments in 1993.

Blattner et al. [1989] suggest the use of simple timbres such as sine or square waves but psychoacoustics (study of perception of sounds) suggests that complex musical instrument timbres may be more effective. Deutsch [1980] claimed that the rhythm would be one of the most effective methods for differentiating sound sources.

3.3. Music

Music can be used in non-speech interface design to convey information in the same way as earcon icons. Several structural parameters should be considered when designing music used in the applications.

Music can provide a non-obtrusive, coherent aural environment which is also aesthetically pleasant. Typical way of using music in the non-speech application is by playing recorded songs when the system is busy, which does not convey much information. More elaborate use of music can represent graphical entities, such as diagrams by using music. [Turunen and Salonen, 2004]

4. Interface Design with Non-Speech Audio

Using sounds in widgets is very popular in today's software applications. The most common widgets known to us are such as frames, buttons, menus, scrollbars, alert boxes, windows and 'drag and drop' operations.

Frames are the simplest widgets and are just coloured region in the user interface. They are used as containers of other widgets. They do not usually react to mouse or keyboard events. [Brewster, 1998b]

Buttons are very common widgets in graphical user interfaces. These may be highlighted when they are pressed. In addition to this type of button there are also checkboxes and radio buttons. [Brewster, 1998b]

Dialog and alert boxes are information windows that are used by the system to bring the important information to user's attention. Scrollbars are connected with another widget and control the view in that particular widget. [Brewster, 1998b]

Menus can be of several different types, but normally all of them allow a user to choose from a set of items. The menu is normally invisible but can be made seen by clicking the menu bar, normally at the top of the window. [Brewster, 1998b]

The widgets in an application are structured as a hierarchy. This hierarchy is the base for the sounds used in an application. In the design of non-speech applications, the use of the sound parameters must be considered carefully. As the Brewster's et al. study states, the most effective parameter to distinguish sounds is timbre. Timbre is known as a tone colour. It's the quality of sound that makes the sound of one instrument or voice different from another. For example, a flute has a different timbre than a clarinet.[Brewster et al.,1996]

Homogeneity of sound is particularly important and should be as unique as possible. This is to make the sound recognition easier. When performing many horizontal and vertical movements it is difficult to keep track of the moves. Each level of the hierarchy can be presented in different changed sound. Time is a critical factor in the interaction; the interaction of sounds in a menu should not slow users down. Therefore the sounds should be brief. Or, more precisely, the meaningful part of the sound should be short, but the sound itself could possibly last longer. If the level of a hierarchy is deep, the sounds become more complex. These deep hierarchies are seen in menus more often than in other widgets of the user interface. [Leplâtre and Brewster, 2000]

Non-speech sounds, auditory icons and mainly earcons have been used in telephone based applications, process control and flight management systems for longer periods of time. These simple synthetic sounds, earcons create messages by beep or melodic tones as an indicator of events and actions starting, ending or processing. Sounds used in such systems should be heard as collection of sounds, differing from each other by their timbre, pitch or rhythm. The rhythm of the

sounds should not be too slow or too fast. Sound should not be annoying the user by its tempo neither let the user to forget what process is currently going on.

5. Strengths and Weaknesses of Non-Speech Audio

Non-speech audio interfaces can extend the usage of many current applications. These applications can be excellent support for visual interfaces as well as they can become a secondary way of presenting information for visually impaired people and people working in restricted conditions.

Sounds are good for communicating information quickly. Unlike speech, non-speech sound is not language dependent; user is not tied to any language, which is important for the increased international and universal use of computer systems. There is also great potential for the results of this work in other non-graphical interfaces. [Brewster, 1998a]

Non-speech audio, unlike speech, is not serial, and therefore is a faster way of communication or completing operations. Non-speech audio requires a lower quality of sound.

Some drawbacks of the non-speech sounds may be the time taken for learning to recognise the sounds used in an application. This is not an issue with a speech interfaces as everyone is able to understand the language spoken to them.

Also the non-speech applications tend to increase the time spent on completing the tasks.

6. Discussion

Non-speech audio in user interfaces is a solution for visually impaired computer users, navigation systems, telephone based services, computer games, process control, flight management systems, hands and eyes busy situations and as another way of handling information.

Sounds are seen to be a good way to communicate information quickly. Unlike speech, non-speech sound is not language dependent and user is not tied to any language, which is important for the increased universal use of computer systems.

On the other hand, non-speech interface usage demands more time for learning the application as well as completing the tasks might become lengthy.

Non-speech user interface design uses auditory icons, earcons and music to produce the interactive sounds between the system and the user. Auditory sounds are natural sounds taken from our everyday environment. Earcons are synthetically produced sounds. Music used, are played recorded songs. The base for the earcons and music design are the sound parameters such as timbre, register, intensity, pitch and rhythm.

Industrial organizations and businesses are forced to co-operate with the non-speech audio industry, due to laws established for the rights and equal opportunities of the disabled people. For example, a visually impaired user is to be provided with suitable applications in their working environments. Also the non-speech audio is to support visual interfaces and help in navigation. Overall, non-speech interfaces are to bring universality for the user interfaces.

References

[Deutsch, 1980] D. Deutsch, The processing of structures and unstructured tonal sequences. *Perception and Psychophysics* **28,** 5, (1980), 381 - 389.

[Blattner et al., 1989] M. Blattner, D. Sumikawa and R. Greenberg, Earcons and icons: Their Structure and common design principles. *Human Computer Interaction* **4**, 1, (1989), 11-44.

[Brewster et al., 1993] Stephen A. Brewster, Peter c. Wright and Alistair D. N. Edwards, An evaluation of earcons for use in auditory human-computer interfaces. In: *Proceedings of InterCHI93*, ACM Press, (1993), 222-227.

[Brewster et al., 1996] Stephen A. Brewster, V.-P. Raty, and A. Kortekangas. Earcons as a method of providing cues in menu hierarchy. In: eds. A. Sasse, R. Cunningham, and R. Winder, *Proceedings of BCS HCI'96,* UK, (1996), 169 – 183.

[Brewster, 1998a] Stephen A. Brewster, Using non-speech sounds to provide navigation cues. *Journal ACM Transactions on Computer Human Interaction,* **5,** 3, (Sept. 1998), 224 - 259.

[Brewster, 1998b] Stephen A. Brewster, The design of sonically-enhanced widgets. *Interacting with Computers,* **11,** 2, (Dec. 1998), 211-235.

[Brewster, 2002] Stephen A. Brewster, Visualization tools for blind people using multiple modalities. *Disability and Rehabilitation,* **24,** 11 - 12, (2002), 613-621.

[Buxton et al., 1994] William Buxton, William Gaver and Sarah Bly, Auditory Interfaces: The use of non-speech audio at the interfaces,

http://www.billbuxton.com/Audio.Toc.html, 1994.

[Jauhiainen, 1995] Tapani Jauhiainen, *Kuulo ja viestintä.* Yliopistopaino, Helsinki, 1995.

[Leplâtre and Brewster, 2000] Grégory Leplâtre and Stephen Brewter. Designing Non-Speech Sounds to Support Navigation in Mobile Phone Menus. In: *Proceedings of ICAD2000* (2000), 190 -199.

[Mäkelä et al., 2003] Kaj Mäkelä, Jaakko Hakulinen and Markku Turunen, The use of walking sounds in supporting awareness. Proceedings of the 2003 International Conference on auditory Display, Boston, USA (July 2003).

[Turunen and Salonen, 2004] Markku Turunen and Esa-Pekka Salonen, Speech Interface Design Lectures, University of Tampere, 2004).
http://www.cs.uta.fi/SID/materials/SID2004-lecture2.pdf.

EXPLORATORY TESTING: A DIFFERENT APPROACH TO TESTING

Martin Zechner

Abstract

In today's hectic and fast culture with short product cycles, less and less emphasis on explicit requirements and design, it becomes generally more and more important to focus testing activities and to employ an efficient testing mindset. Badly tested software can very easily break a company's reputation. Traditionally, software is tested in a scripted fashion, by carefully designing test cases based on given requirements and specifications, before any actual testing commences. This paper presents a different approach to software testing, called exploratory testing, sometimes also referred to as ad-hoc testing and best described as "learning, test design, and test execution at the same time". This testing approach will be contrasted with the more traditional way of scripted testing.

Key words and terms: ad-hoc testing, exploratory testing, scripted testing, GUI application testing.
CR classification: D.2.5

1. Introduction

In today's hectic and fast culture with short product cycles, less and less emphasis on explicit requirements and design, it becomes generally more and more important to focus testing activities and to employ an efficient testing mindset. Whatever model of software development is being adhered to, testing remains a crucial activity within that model. Badly tested software can very easily break a company's reputation.

Graham [2002] argues that a better link between requirements and testing would eventually result in better tests and better software, but what if the software developed is of such a nature that requirements simply cannot be very explicit, or static and not change?

Traditionally, graphical user interface (GUI) application design is quite difficult and the only reliable way to achieve good interfaces is through iteration [Myers, 1995], taking feedback from testers and end-users into

consideration. It is exactly this iteration and change to the interface which makes GUI application testing demanding. Requirements may be vague and become clearer once the system evolves, or they may change altogether. Scripted testing, a testing method whereby test cases are generated based on requirements and other relevant specifications and then executed, may prove to be an inefficient way on its own, because changes in requirements together with iterations bring forth a need for constant documentation updates, including test case updates and those updates may temporally lag behind the actual implementation. Also, at the time of initial test case creation, the exact final system may not have been conceivable, adding to the challenge.

A different approach to software testing, exploratory testing, will be presented in this paper, together with the traditional scripted approach and both methods will be contrasted. Exploratory testing, sometimes referred to as ad-hoc testing can best be described as ”learning, test design, and test execution at the same time” [Bach, 2003a]. This is in stark contrast to the more “traditional” scripted testing, where tests are designed in advance.

Several concepts of software testing will be introduced in Chapter 2, followed by Chapter 3, which describes the concept of scripted testing, gives an example and discusses the advantages and disadvantages. Chapter 4 introduces exploratory testing and its principles, together with an example and discusses advantages and disadvantages of the approach. Chapter 5 concludes the paper.

2. Testing

Testing is an extremely creative and intellectually challenging task and may even be destructive, but at the same time adding value to the application under test. (Less errors and stability are likely to have a positive effect on end-users.) In this chapter, several basic concepts related to testing will be defined. Firstly terms commonly used to denote application defects will be clarified in the following section. Thererafter testing as an activity will be defined, followed by a testing model. The final section of this chapter focuses on test cases.

2.1. Errors, faults, failures, defects and mistakes

It is very common to hear some people talk about software having defects or errors, others may be talking about faults that have been found, yet others about failures. Terms appear to be used interchangeably. As this work deals with software testing, it is important to clearly define each of these.

Human action during software development or operation leads to mistakes, which in turn manifest themselves as software faults. As a result of such faults,

failures occur. A failure is the inability of a system or component to perform its required functions within specified performance requirements, resulting in an incorrect result. (According to Beizer [1995, p. 9] in good software developed under a good process, failures are rather caused by complexity and to a lesser extent by mistakes of individual programmers.)

The amount by which the results are incorrect are the errors. [IEEE, 1990]. The definition of failure includes a reference to "specified performance requirements", which in turn implies that such must have been specified before any actual testing commences and must be available during testing. This does not necessarily need to be so in exploratory testing, where testers may test software not only against specified requirements, but also against certain guidelines (heuristics) and against the expected outcome. Given that, the definition of failure could be shortened to "the inability of a system or component to perform its required function resulting in an incorrect result", without losing the essence.

The term "error" is commonly used to denote mistakes, faults, failures and errors as defined above [IEEE, 1990]. An "error" is also known as a "bug". The term "defect" is used to denote both faults and failures and will be used also in this work in that way.

2.2. Testing definitions

Testing has been defined in various ways. Myers [1979, p. 5] defines testing as "the process of executing a program with the intent of finding errors", which clearly points towards a destructive mindset [Gelperin and Hetzel, 1988].

Kaner *et al.* [1993, pp. 124-125], like Myers, see the purpose of testing in "finding errors" (here meaning faults, failures and mistakes) and define one of several characteristics a good test should have as a "reasonable probability of catching an error". Also this is a destructive point of view about the primary goal of testing.

The IEEE standard glossary of software engineering defines testing as "the process of analyzing a software item to detect the differences between existing and required conditions (that is, bugs) and to evaluate the features of the software items" [IEEE, 1990]. This definition focuses on both the bug-finding (destructive) aspect and an evaluative aspect. It is unclear whether the evaluative aspect is meant to demonstrate in some way or other that the software under test is working satisfactorily. This definition does not include where the information about the "required" condition may be found and hence one may argue that this does not necessarily have to mean specifications, but could include for example guidelines (heuristics) which a tester may be using.

Beizer [1995, p. 3] views testing as the act of designing, debugging, and executing tests. Test in this case refers to a sequence of one or more subtests executed as a sequence because the outcome and/or final state of one subtest is the input and/or initial state of the next. Beizer's definition lacks the intent, which is so elegantly expressed in Myers', the IEEE´s and Kaner *et al.*'s definitions, but includes test design and debugging. Even though not stated in his definition, software, according to Beizer, is tested for several reasons. Amongst those are breaking the software (taken from Myers' definition), demonstrating that it works, and providing information that can be used for prevention of mistakes in the future. The latter is seen as highest goal of software testing [Beizer, 1995, p. 7].

Myers [1979] states that testing cannot demonstrate that errors (here meaning faults and failures) are not present, which is starkly contrasted by Beizer [1995, p. 7] who makes a distinction between dirty tests, i.e. tests designed to break the software and clean tests, i.e. tests designed to demonstrate the software's correct working. This reflects a difference in mindset, i.e. non-demonstrative vs. demonstrative. According to Myers, a possible definition of testing as being the process of demonstrating that a program does what it is supposed to do, i.e., satisfies its specification, is no good, because even if it does what it is supposed to do, there may still be errors present, causing the program to do things it is not supposed to do. [Myers, 1979, p. 7]. What is elegant about Beizer´s definition is that it allows and incorporates three goals of testing, namely the "breaking of the software" , the "verification", and the "provision of information". There is no reason to believe that these activities need to be mutually exclusive and therefore testing can be seen as both a destructive and a demonstrative activity.

Psychologically, a definition of testing as a process of finding errors is superior to any definition claiming the contrary as this could become a self-fulfilling prophecy, i.e. if we set out to show that there are no errors we will use a different mindset as when the goal is to show that there are errors. [Myers, 1979, pp. 4-7]

It is generally recognised that the earlier that defects are found, the lower are the costs of correcting them [Beizer, 1995]. Hence, code inspections (a set of procedures and error detection techniques for group code-reading) and walkthroughs (similar to code inspection, but with different procedure e.g. playing computer, and different error detection techniques) are seen as a good start of any "testing" [Myers, 1979]. Such testing activities, including requirements reviews, can be both evaluative and preventative in nature [Gelperin and Hetzel, 1988], even though their goal is still to find faults.

Depending on the kind of software development process models followed (waterfall [Royce, 1970], waterfall-based V-model [Rook, 1986], incremental or concurrent) the timing of testing takes on different roles. In the nowadays considered antiquated [Kaner, 2003] waterfall model, testing linearly follows coding and typically comes at the end of the software development process when coding is complete. Myers [1979] tends to adhere to this model as also he views testing as following coding and does not leave room for intertwined evolution of coding and testing, something which is considered essential [Hetzel, 1988].

2.3. Testing models

Even though the process of testing may and should be viewed as being an inherent part of software development, and consequently part of the software development models, this does not imply that specific testing models are not required. Several testing models have been described by Gelperin and Hetzel [1988] and they differ basically along the dimensions of activity scope and primary goals.

The scope of activity has very much to do with the software development process model that is being adhered to. In the waterfall model of software development [Royce 1970], the scope of testing is typically at the end, whereas in the V-model for example [Rook, 1986], testing is performed on each stage of integration and the scope is, as a result, much wider. Typically, as depicted in Figure 1, a test model would include test planning, test design, test implementation, test execution, test result gathering and test maintenance [Gelperin and Hetzel, 1988; Hetzel, 1988].

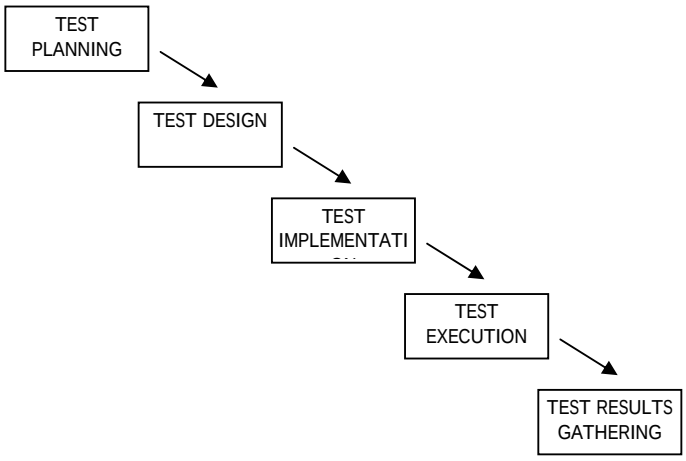


Figure 1. A typical model of testing [Hetzel, 1988].

2.4. Test cases

Test cases are the instructions for testers about how they should test and what they should test. It is impractical or often impossible to find all errors in a program, because exhaustive input testing is impossible or just plainly not feasible. Neither is it possible to create error-free software, even though such was still believed by some in the early 1970s [Royce, 1970].

If the testing mindset is a destructive one, then a successful test case is one that finds an error and an unsuccessful test case is one that causes a program to produce the correct result. [Myers, 1979, p. 16]. A good test case is therefore one that has a high probability of detecting an as-yet undiscovered error. This means conversely that once the error has been discovered, the test case's usefulness decreases since the possibility of finding the same error again most likely is smaller [Bach, 2003a]. If, however, the testing mindset is a demonstrative one, then a successful test case is one that does not find an error and would thus show that the item tested does indeed function as expected.

According to Myers [1979, pp. 37-75] there are two basic testing methodologies, namely black-box and white-box testing that can be employed when creating test cases. Black-box testing (data-driven or input/output driven testing) is a testing strategy whereby the tester views the program as a black box and is not concerned with the internal behaviour or structure thereof. Test data is derived solely from the specifications.

White-box testing (logic-driven testing) is a testing strategy whereby the tester is able to examine the internal structure of the program. Test data is derived from the program's logic and the tester would try and achieve coverage of all paths of control flow with the test cases. Again, such exhaustive testing is impractical and not feasible, maybe even impossible because the number of paths of control flow may be extremely large. Exhaustive path testing also in no way would guarantee correct functionality, because the path coverage has nothing to do with functionality. All paths may have been covered, but the program nevertheless is not doing what it should do. Also missing functionality cannot be detected in that way.

A necessary part of a test case is a definition of the expected output or result. Test cases must also be written for invalid and unexpected as well as valid and expected input conditions. [Myers, 1979, pp. 12-14]. The requirement about the need of an expected output or result is tricky when it comes to exploratory testing, because as shall be shown later, due to the nature of exploration, specific and detailed outputs or results cannot always be anticipated.

3. Script-based (scripted) testing

This chapter explains the fundamentals of scripted testing and will move on to a simple example of a hypothetical application. Thereafter, advantages and disadvantages associated with scripted testing are presented and discussed.

Script-based or scripted testing is testing performed by using a script or collection of step-by-step pre-defined test cases. These test cases are typically, but not necessarily, defined early during software development and are usually based on software requirements and internal logic, depending upon which procedure in their conception has been adhered to (black-box, white-box or a combination). The idea is that once the software can be tested it will be tested against the scripted test cases. Black-box and white-box methods are commonly used to arrive at the actual test cases

Consider a hypothetical GUI application called "Opener" that consists of only one button which, when pressed, opens another application "A". One scripted test case for "Opener" could look like this:

Precondition:	*Application "Opener" is open.*
Test Steps:	*1. Move the mouse pointer over the button.*
	2. Press the button.
Expected result:	*Application "A" is opened.*
Exceptions:	*Application "A" is not opened.*

An advantage of scripted testing is that it lends itself very well to measurement. A total number of test cases can be calculated, test metrics can be created, measuring e.g. how many test cases have been run compared to how many have been planned. The measurement is something, which fits very well with document-driven development models. It is all about measurement and testing progress can be measured easily. It is also easy to assign testing resources, once the overall testing effort is known through the test specifications, greatly helping project management.

Another advantage is that from a testing skills perspective, nearly anyone can execute basic scripted tests, because that what is required is broken down into clear steps, which are easily followable. Executing scripted tests does not necessarily require very experienced and highly skilled testers and even testers just starting their testing career can cope.

One drawback of scripted testing is that if one test case finds a defect in the software, then after the defect has been fixed, the probability of finding another defect with the same test case is much lower than the first time. This is turn

implies that test cases lose their effect to some extent and new test cases may be needed, i.e. a change in the script may be required [Bach, 2003a].

Another disadvantage of scripted testing is that in practise the software being developed usually changes from the time of the first requirements being written down to the time when it will be released to the market.

In a highly competitive environment adjustments are required in order to respond to challenges unknown at the time of writing requirements to make the software as "good" as possible from and end-user perspective. (GUIs are usually developed through many iteration cycles [Myers, 1995].)

Such changes in requirements over the development cycle of the software are very typical for interactive applications [Myers, 1995].

Scripted test cases do not only lose their power the longer they are used [Bach, 2003a], but they also may get outdated. This in turn requires regular updates to the scripted test cases in order to keep up with the ongoing software development.

Scripted test cases may also limit the testers' creativity and exploration, because they are bound by the already laid out test cases, unless of course deviations from the laid out test cases are specifically allowed.

4. Exploratory testing

This chapter explains the fundamentals of exploratory testing and how they can be applied in practice by means of an example. Exploratory testing is also contrasted against scripted testing and advantages and disadvantages of the approach are discussed.

Exploratory testing, also sometimes called ad-hoc testing, can be best described as "learning, test design, and test execution at the same time" [Bach, 2003a]. Learning here refers to what is observed and retained in memory, i.e. learned during the actual testing and serving as input for new tests to be planned and executed.

It is important to know that exploratory testing is not a testing technique as such, but rather a different approach to testing – a different way of thinking about testing [Kaner and Bach, 2004]. However this does not appear to be viewed as such by all, since e.g. a chapter on exploratory testing by Bach in "The Testing Practitioner" [Van Veenendaal, 2002] is located in the section of test techniques. Even though such apparent contradictions exist, these do not matter within the scope of this work.

Exploratory testing as such is not a radically new idea and most likely testers do perform some exploratory testing occasionally, because they are after all not mere robots but human beings [Bach, 2002] – what is new, however, is

that exploratory testing is emerging as a subject of its own alongside the more "traditional" views on testing in handbooks on software testing (e.g. [Van Veenendaal, 2002]).

The earliest mention of a way of thinking one could relate to some extent to exploratory testing is by Myers [1979, p.73-75], called "error guessing" with the basic idea being the enumeration of a list of possible errors or error-prone situations and then writing test cases based on such a list.

This would technically make it scripted testing, but one may argue that it is also at least hypothetically possible, to design a test through error guessing, writing it down and executing it and that would make it exploratory in nature. The main difference in what Myers says and true exploratory testing however is that Myers requires the creation of more than one test case prior to any testing and in that way the interactivenss and the feedback, i.e. the "information gained while testing" as Bach [2003a] calls it, will not lead to any new test cases generated on the fly and exactly that is the essence of exploratory testing. Furthermore, even though Myers mentions "error guessing" (a technique which can be very valuable also for exploratory testing), he is absolutely not in favour of ad-hoc testing and is rather of the opinion that "throw-away test cases should be avoided unless the program under test is a throw away program" [Myers, 1979]. One reason for that is that after testing, test cases will be 'lost'. Nonetheless, this does not have to be so, because documenting ad-hoc test cases can overcome the problem of 'loss', even though within exploratory testing very detailed test cases are usually not written down, but rather notes of what has been done, so that in case of error a test may be repeated [Bach, 2003a], Agruss and Johnson, 2000].

Microsoft´s exploratory test procedure (based on Bach) for testing third-party applications for Microsoft Windows compliance explicitly states that very detailed test cases should not be written down, but rather an outline of what was done suffices, because such activities take too much time and interrupt the flow of testing [Microsoft, 2004]. Kaner *et al*. [1993] have rather been vaguer and merely state that what has been done and what has happened should always be written down when testing in an exploratory way. If how it has been done is not preserved, then re-use is indeed impossible. Re-running exploratory (ad-hoc) test cases no longer make the process exploratory but merely scripted, but for the sake of accountability and error reproduction, testing notes may be required. The point is that in any case, an exploratory testing approach to testing brings forth much lighter documentation than a purely scripted testing approach, which may, for example, follow the 52-page IEEE standard for software test documentation 829-1998 [IEEE, 1998]. This IEEE standard describes

the basic test documents that are associated with software testing and is waterfall-oriented.

Exploratory testing can be thought of as a "journey" with a desired outcome or a purpose (charter), but with no clear path defined and with many possible ways to reach the destination. Incidences during the "journey" will be used as guidance (direct feedback) in deciding how to continue.

Consider again the hypothetical GUI application called "Opener" that consists of only one button which, when pressed, opens another application "A". An exploratory testing session could look like this:

> *Purpose: Test all functions of the application "Opener".*
>
> *The tester would open application "Opener" take the mouse, move over the button, press it and notice that application "A" opens. The tester however also notices that the opening takes some time and that it feels slow. This observation makes the tester come up with a new test, one where the button would be pressed several times in succession, because that may or may not cause some problems, given the observed delay. After the tester has executed this new case, she is amazed to find that application "Opener" has crashed.*

This is what exploratory testing is all about; starting testing with a certain goal, learning about the product under test while testing and using gained information and feedback in the further testing design on-the-fly.

Exploratory testing is not against the idea of scripting [Bach, 2003a], it may complement it and viewing script-based testing and exploratory testing on a continuum would be most appropriate.

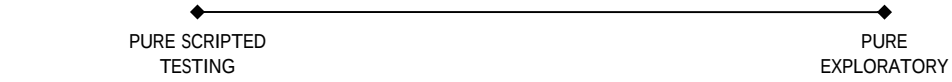


Figure 2. The testing continuum.

The continuum, as depicted in Figure 2, ranges from purely script-based testing (prescribed in advance in every detail) and pure exploratory testing (every test emerges at the moment of execution) [Bach, 2003a]. A purely "scripted" tester would not be allowed to deviate from the planned testing, whereby one with some "exploratory" traits would be allowed to do so. Conversely, a purely "exploratory" tester would not be allowed to do any planning before embarking on the testing, whereby one with a slight "scripted" trait would be allowed to do so. This has also been referred to as a narrow and broad view of exploratory testing [Bach, 2002], whereby a narrow view would

mean no scripting and a broad view would allow scripting and offer a possibility for testers who use scripting to deviate and improvise on the scripted tests.

Exploratory testing is not bound by the kind of software development processes or models being followed and can hence be used independently of these. Neither is it bound by the testing methods used for scripted testing. Such testing methods can also be used in an exploratory way [Kaner and Bach, 2004].

Exploratory testing very much happens inside the minds of the testers and careful observation, critical thinking and good ideas or guidelines are therefore essential. More specifically, Bach [2003b] has defined 8 key elements that distinguish an expert exploratory tester from an amateur. These are test design skills, careful observation skills, critical thinking skills, diverse idea generation, rich resources inventory, self-management skills, rapid learning skills and status reporting skills. All these are skills that are more likely to be found in experienced than in inexperienced testers and even then such testers may not be so readily available in any given software project, which is one of the criticisms exploratory testing has received [Van Veenendaal, 2004].

4.1. Testing oracles

An oracle can be said to be “any human or mechanical agent which decides whether or not a program behaved correctly in a given test, and accordingly produces a verdict of “pass” or “fail” [SWEBOK, 2004], or as Microsoft [2004] puts it “an oracle is a strategy for determining whether an observed behavior of the product is or is not correct. An oracle is some device that knows the ‘right answer.’ “

When the testing approach is a scripted one, then the determination of whether a program behaves correctly is already intrinsic to the test cases, which will include a mention of an expected result. This expected result serves as the oracle and will guide the tester. With an exploratory approach this is not so straightforward, because no expected results are defined. This in turn implies that nevertheless the tester needs to have some guidance, an oracle, to tell whether the observed behaviour is as intended or not. This is where the experience of the tester is crucial. Experienced testers are able to compare observed behaviour to that of other applications they may have seen or tested or they can employ guidelines, so-called heuristics, in their testing. A set of guidelines for exploratory testing, based on work by James Bach, has been published by Microsoft [2004].

Whittaker [2003] proposes the "fault model" to guide testing. He claims that "understanding what software does -- and how it may fail doing it -- is crucial to being an effective tester.” There are 4 fundamental capabilities of Software: 1.

Software accepts input, 2. Software produces output, 3. Software stores data internally, 4. Software performs computations using input and stored data. The fault model is therefore "if software does any of these 4 things wrong, it fails." This is one of many models that can guide testers and there is no restriction on which models and techniques may be used for exploratory testing and each new model or technique will just add to the richness of the available inventory of resources.

Less experienced testers may lack such a rich set of guidelines or experiences with similar products (e.g. Windows applications). Exploratory testing requires hence a greater level of experience than would be required for scripted testing, which has also been identified as a disadvantage of exploratory testing [Van Veenendaal 2004], because such experience level is usually not found abundantly in software projects.

It is important to understand that exploratory testing, due to its exploratory nature, can provide information about an application under test on a different level than scripted testing. It is not simply a matter of “pass” or “fail” in many cases, but there is also an element of feedback. Testers can provide feedback about their testing session about observations they have made, irrespective of whether they relate to actual failures or not. This, in turn, implies that the oracle does not necessarily have to be present at the time of testing but could be used after an exploratory testing session. There is then a slight danger that issues will be reported that are not really failures as such, or are already known problems, but they will be nevertheless very valuable product feedback. In order to reduce the number of reports of already known issues, such can be provided at the beginning of a test session.

Since the probability of the existence of more errors in a section of a program is proportional to the number of errors already found in that section [Myers, 1979] such information may also aid an exploratory tester and may add to the already existent repertoire of methods and techniques.

4.2. Six principles of testing

Hetzel [1988] presents six principles of testing, elements of which are based on earlier work by others. Following are the six principles and thereafter a short reflection on how these principles apply to a scripted testing approach and an exploratory testing approach.

1. Complete testing is not possible. This is due to practical limitations and also a theoretical impossibility. The total number of possible test cases may be infinite which makes it practically impossible to try them all. There will furthermore never be a way to be sure that we have a perfect understanding of what a program is supposed to do.

2. Testing work is creative and difficult. Creativity, business knowledge and testing experience and methodology are required.

3. An important reason for testing is to prevent deficiencies from occurring. Testing is not a phase, but it must be an intrinsic part of the software development model, whichever one is chosen.

4. Testing is risk-based. Risk is to serve as the basis for deciding what and how to test.

5. Testing must be planned. Ad hoc testing does not provide enough information to reasonably measure software quality and may even be harmful due to the possibility of it leading to a false sense of security.

6. Testing requires independence. The tester should be unbiased and independent in spirit.

Principle 1 does not directly relate to which view is taken regarding testing; this principle is relevant for both exploratory and scripted testing alike with a possibility of the exploratory testing being less "complete" by nature than the scripted testing.

Exploratory testing requires even more experience and creativity than may be required for scripted testing, which is also one of the drawbacks of exploratory testing [Van Veenendaal, 2004], because it is more likely to have a lack of very experienced testers than a sufficient number. Nevertheless principle 2 is relevant for both scripted testing and exploratory testing.

Principle 3 clearly fits with a preventative testing mindset. Due to the rather destructive nature of exploratory testing, principle 3 does not apply. Exploratory testing very often deals with complete systems and not with requirements or specifications, something which is also a downside [Van Veenendaal 2004a]. Problems that are spotted early (in requirements or system specifications) can also be fixed early at a much lower cost. Graham for example advocates a very tight link between testing and requirements [Graham, 2002]. This however presupposes that requirements and specifications are readily available and detailed enough. Especially within the field of GUI application development with its numerous iterations such may not be the case. Scripted testing does not necessarily have to abide by this principle either, because a destructive mindset may also be employed there. Principle 3 is therefore very much dependent on the overall organizational view of testing, whether it is seen as a destructive process whereby errors need to be found or as a preventative process whereby the goal is on prevention. Both are possible as well, of course. Also the possibility of employing an exploratory way of testing throughout the software life-cycle are theoretically

possible, but not within the scope of this work, where the application is limited to system testing.

Principle 4 is relevant for both scripted testing and exploratory testing.

Principle 5 is the one principle that exploratory testing completely does not abide by. Even though exploratory testing is test planning and execution at the same time, this is a different planning than is implied by Hetzel, who sees great relevance and benefits in thinking through and defining expected outcomes. Within exploratory testing expected outcomes are not defined per se, because by definition it is an exploratory process with an unknown result. Problems may or may not be encountered, formerly unknown issues may appear. Only the charter guides the exploratory tester.

Principle 6 is relevant for both scripted testing and exploratory testing alike.

5. Conclusion

In this paper I have presented the concept of exploratory testing and have contrasted it with the more traditional scripted testing approach. Strengths and weaknesses of both approaches were stated. Neither of the approaches has clear advantages over the other under all circumstances. An exploratory approach to testing may be more appropriate and beneficial when the application being tested is GUI-based and developed in a highly iterative fashion, with fully specified requirements not available or "outdated" or when it is important to get feedback about the application under test.

There exists a lot of anecdotal evidence, but currently no research comparing the productivity of a script-based testing approach to an exploratory testing approach [Bach, 2002; Bach, 2003a; Agruss and Johnson, 2000] and this could be the basis of future work.

References:

[Agruss and Johnson, 2000] Chris Agruss and Bob Johnson, Ad hoc software testing – a perspective on exploration and improvisation. Available at http://www.testingcraft.com (Checked September 2004)

[Bach, 2003a] James Bach, Exploratory testing explained v1.3.4/16/03. First published as a chapter in The Test Practitioner, 2002. Essay available from http://www.satisfice.com (Note: v1.3.4/16/03 is different from the version published in The Test Practitioner.)

[Bach, 2003b] James Bach, Inside the mind of an exploratory tester. *Software Testing & Quality Engineering (STQE)* **5**, 6 (November/December 2003), 16-23.

[Bach, 2002] James Bach, Exploratory testing. In: Erik van Veenendaal (ed.) *The Testing Practitioner*, UTN Publishers, 2002, 209-221.

[Beizer, 1995] Boris Beizer, *Black-Box Testing: Techniques for Functional Testing of Software and Systems.* John Wiley & Sons, 1995.

[Gelperin and Hetzel, 1988] David Gelperin and Bill Hetzel, The growth of software testing. *Communications of the ACM* **31**, 6 (June 1988).

[Graham, 2002] Dorothy Graham, Requirements and testing: seven missing-link myths. *IEEE Software* **19**, 5 (September/October 2002).

[Hetzel, 1988] Bill Hetzel, *The Complete Guide to Software Testing 2nd Edition.* John Wiley & Sons, 1988.

[IEEE 1990] *IEEE Standard Glossary of Software Engineering Terminology*. IEEE Std 610.12-1990.

[IEEE 1998] *IEEE Standard for Software Test Documentation.* IEEE Std 829-1998 (Revision of IEEE Std 829-1983).

[Kaner and Bach, 2004] Cem Kaner and James Bach, The nature of exploratory testing. *Slides presented in Tampere, Finland, May 7th 2004*. Available at http://www.kaner.com/articles.html (Checked September 2004)

[Kaner *et al.*, 1993] Cem Kaner, Jack Falk and Hung Q. Nguyen, *Testing Computer Software 2nd Edition*. International Thomson Publishing, 1993.

[Kaner, 2003] Cem Kaner, How many lightbulbs does it take to change a tester. Presented at: *Pacific Northwest Software Quality Conference 2003*. Available from http://www.kaner.com/articles.html (Checked September 2004)

[Microsoft, 2004] Windows Applications Exploratory Test Procedure v. 3.1 as part of Windows Application Compatibility Toolkit 3.0, 2004. Available from http://www.microsoft.com

[Myers, 1979] Glenford J. Myers, *The Art of Software Testing*. John Wiley & Sons, 1979.

[Myers, 1995] Brad A. Myers, User interface software tools. *ACM Transactions on Computer-Human Interaction* **2**, 1 (March 1995), 64-103.

[Royce, 1970] Winston W. Royce, Managing the development of large software systems. In: Proceedings *IEEE WESCON* (August 1970), 1-9.

[Rook, 1986] Paul E. Rook, Controlling software projects. *IEE Software Engineering Journal* **1**, 1 (January 1986), 7-16.

[SWEBOK, 2004] Alain Abran and James W. Moore, *Guide to the Software Engineering Body of Knowledge (SWEBOK)*. IEEE Computer Society, 2004.

[Van Veenendaal, 2002] Erik van Veenendaal, *The Testing Practitioner*. UTN Publishers, 2002.

[Van Veenendaal, 2004] Erik van Veenendaal, Exploratory testen – zinvol of onzin. *Automatisering Gids* 14 (2.4.2004).

[Whittaker, 2003] James A. Whittaker, *How to Break Software: A Practical Guide to Testing.* Addison Wesley, 2003.