Erkki Mäkinen (toim.)

Pieniä tietojenkäsittely-tieteellisiä tutkimuksia (kevät 2004)

TIETOJENKÄSITTELYTIETEIDEN LAITOS
TAMPEREEN YLIOPISTO

B-2004-7

TAMPERE 2004

Lukijalle

Tähän julkaisuun on kerätty kevätlukukaudella 2004 pitämälläni Tutkimuskurssilla tehdyt, määräaikaan mennessä valmistuneet suomenkieliset tutkielmat.

Toimittaja

Sisällysluettelo

Käyttäjän käsitemallien siirtyminen ohjelmien välillä

Anna Brinck

Tiivistelmä

Tässä tutkielmassa tarkastellaan ihmisen kognitiivisten toimintojen, erityisesti käsiteyhteyksien muodostumisen, merkitystä käyttöliittymän suunnittelussa ja toteutuksessa. Käyttäjälle syntyy yksilöllinen käsitemalli jokaisesta käyttämästään ohjelmasta ja tämä malli siirtyy aina mukana uuden ohjelman oppimistilanteeseen. Käyttäjän käsitemallien olemassaolo voidaan huomioida ohjelmistosuunnittelussa mm. ohjeistuksen tai vaihtoehtoisten toimintatapojen avulla.

Avainsanat ja -sanonnat: käsitemallit, mentaaliset mallit, käytettävyys, HCI.
CR-luokat: H.1.2, H.5.2, I.3.6, J.4

1. Johdanto

Yhdenmukaisuus on tärkeä käytettävyyden periaate ohjelmistosuunnittelussa [Nielsen, 2004]. Yhdenmukaisuuden periaatetta voidaan ajatella sovellettavan sekä yhden ohjelman sisällä että ohjelmien välillä. Aihetta on tutkittu viime aikoina melko vähän ottaen huomioon ohjelmien kehitystahdin. Ei voida olettaa, että työntekijä voisi käyttää koko työuransa ajan yhtä ja samaa tuttua ohjelmaa. Versiot vaihtuvat uusiin, käytössä olevia ohjelmia vaihdetaan kustannussyistä tai ohjelman kehittämisen loppuessa ja työtehtävän ohjelmalle asettamat vaatimukset muuttuvat. Jokaiseen tällaiseen ohjelmavaihdokseen liittyy käyttäjän kannalta oppimisprosessi, jossa vanhan ohjelman käytössä opitut toimintatavat pitää muuttaa uuden ohjelman mukaisiksi. Tämä vie aikaa ja kuormittaa käyttäjän muistia.

Näissä oppimistilanteissa ongelmia aiheuttaa muiden syiden lisäksi se, että käyttäjä toimii, kuten on aikaisemman ohjelman yhteydessä tottunut toimimaan. Odotuksista poikkeava piirre uuden ohjelman toiminnassa aiheuttaa hämmennystä ja kyvyttömyyden tunnetta.

Uuden ohjelman käyttöä ei aina opetella itsenäisesti. Opetustilanteessa käyttäjälle annettu käsitemalli vaikuttaa ohjelman oppimiseen. Tällaiset ulkoiset käsitemallit voivat olla joko analogisia (vastaavuus reaalimaailman kanssa) tai

abstrakteja [Shayo & Olfman, 1997]. Abstraktit mallit ovat käytännössä kaavioita ohjelman loogisesta toiminnasta. Abstraktit mallit ovat yleiskäyttöisempiä kuin analogiset mallit. Ulkoisten ja käyttäjän omien käsitemallien suhdetta käsitellään tarkemmin luvussa 2.

Tässä tutkielmassa perehdytään käsitemalleja ja niiden siirtymistä käsittelevään aikaisempaan tutkimukseen sekä esitellään muutamia käytännön esimerkkejä aiheeseen liittyen. Tarkoituksena on selvittää, kuinka käsitemallit muodostuvat, kuinka ne siirtyvät eri ohjelmien välillä ja kuinka niiden olemassaoloa voidaan käyttää hyväksi ohjelmiston suunnittelussa ja uuden ohjelman käyttöönotossa. Käsitemallien perustaa käsitellään tarkemmin luvussa 3.

Käyttäjä mukautuu ohjelmaa käyttäessään tiettyihin toimintatapoihin ja erityisominaisuuksiin, esim. näppäinoikoteihin ja valikkorakenteisiin. Näiden erityispiirteiden erot ohjelmien välillä aiheuttavat käyttäjässä hämmennystä ja virhealttiutta. Ohjelmistosuunnittelijoilla on keinoja, joilla käyttäjän siirtymistä uuden ohjelman käyttöön voidaan helpottaa. Näitä keinoja käsitellään esimerkkien avulla tarkemmin luvussa 4.

2. Aikaisempi tutkimus

2.1 Ulkoiset käsitemallit

Blakey *et.al.* [1999] käyttivät tutkimuksessaan kahta erillistä ulkoista käsitemallia, analogista ja abstraktia, ja tutkivat minkälainen vaikutus näillä malleilla on käyttäjän oppimisprosessiin. Tutkimuskohteenaan heillä oli relaatiotietokantasovelluksen opettaminen ja testihenkilöinä yliopisto-opiskelijoita, joilla ei ollut vahvaa tietoteknistä taustaa eikä juurikaan kokemusta tietokantajärjestelmistä. Analogisessa käsitemallissa he käyttivät metaforana videovuokraamoa, sillä videovuokraamossa tapahtuvat asiat ovat yleisesti tunnettuja. Videovuokraamoon liittyvät käsitteet yhdistettiin tietokantojen käsitteisiin, kuten taulut, pääavaimet ja suhteet. Käsitteitä selvennettiin kaaviokuvin ja tekstein, mutta myös käytännön esimerkein videovuokraamoista. Vertailun vuoksi muodostetussa abstraktissa mallissa selvitettiin tietokantaohjelman käsitteitä pääasiassa tekstein.

Heidän hypoteesinaan oli, että tutun metaforan käyttäminen aikaisemmin tuntemattomien käsitteiden opettamiseen auttaa käyttäjiä oppimisprosessissa. Testitilanteessa testihenkilöt jaettiin kahteen ryhmään, joista toinen opiskeli tietokantakäsitteitä analogisen mallin ja toinen ryhmä abstraktin mallin avulla. Tulosten mukaan molemmissa ryhmissä opittiin tietokantajärjestelmän toiminta

melko hyvin, mutta analogisen mallin ryhmässä käytännön esimerkit auttoivat ymmärtämään käsitteet syvemmin ja vahvistamaan käyttäjien omia käsitemalleja tietokannoista.

Myös Olfman ja Shayo [1998] tutkivat käsitemallien vaikutusta uuden tietokantajärjestelmän oppimiseen. Testihenkilöt jaettiin kahteen ryhmään: vastaalkajiin ja tietokantajärjestelmän toiminnan sekä käsitteet jo tunteviin. Testitilanteessa ryhmät saivat opiskelumateriaalikseen erilaiset abstraktit käsitemallit. Tarkoituksena oli tutkia, kuinka aikaisempi käyttökokemus ja opetustilanteessa määritelty ulkoinen käsitemalli vaikuttavat oppimistulokseen, opitun tiedon jäsentelyyn ja jälleenkäyttökykyyn. Tuloksista kävi ilmi, että käyttäjän oman käsitemallin oikeellisuudella on suuri vaikutus oppimistuloksiin ja että annettu ulkoinen käsitemalli näkyy lähinnä oppimismotivaation kasvamisena.

Mikäli käyttäjän aikaisemmin muodostama käsitemalli on oikea, eli vastaa todellisen järjestelmän toimintaa ja sisältöä, sillä on oppimistilanteessa positiivinen vaikutus. Vastaavasti jos käyttäjän käsitemallissa on suuria poikkeamia tai suoranaisia virheitä, vaikuttaa se oppimista jarruttavalla tavalla ja häiritsee oikean käsitemallin muodostumista.

2.2 Kahden ohjelman välillä tapahtuva siirtyminen

Kahden ohjelman välillä siirtymistä käsitteleviä tutkimuksia esitellään seuraavaksi lähinnä historiallisesta näkökulmasta. Koska tutkimustulokset ovat lähes 20 vuoden takaa, niitä ei voida suoraan verrata nykyisistä ohjelmista saatuihin tuloksiin. 1980-luvulla suurin osa käyttöjärjestelmistä ja -liittymistä oli komento- tai valikkopohjaisia, kun taas nykyään lähes kaikki yleiset käyttöjärjestelmät ja ohjelmat ovat graafisia.

Foltz *et.al.* [1988] käsittelivät siirtymisprosessia saman ohjelman kahden eri version välillä. Erityishuomio kiinnitettiin versioiden valikkojärjestelmissä esiintyneisiin eroihin ja siihen kuinka erot vaikuttivat käyttäjien toimintaan. Tutkimuksen tuloksista havaittiin, että valikoissa tapahtuneet rakenteelliset muutokset, kuten valikon kohtien lisääminen tai poistaminen, eivät aiheuttaneet merkittäviä eroja toiminnan tehokkuudessa. Valikon kohtien kielelliset muutokset sen sijaan havaittiin selvästi virheitä aiheuttavaksi piirteeksi. Havaittiin, että tutkitun tekstinkäsittelyohjelman toimintojen käsitteet siirtyivät ohjelman aikaisemmasta versiosta uuteen. Sen sijaan sanojen ja sanamuotojen muuttaminen vaati joka kerta uusien sanojen opettelua ja täten kuormitti käyttäjän muistia huomattavasti.

Karat *et.al.* [1986] käsittelivät tutkimuksessaan kahden eri ohjelman välillä tapahtuvaa siirtymistä ja sitä, kuinka ohjelmien erilaiset toimintatavat vaikuttavat käyttökokemukseen. Myös he käyttivät tutkimuskohteenaan tekstinkäsittelyohjelmia. Tutkimuksessa todettiin, että vaikka tekstinkäsittelyohjelmien voidaankin olettaa sisältävän tietyt toiminnot ja ominaisuudet, tapoja näiden toteuttamiseen on lukemattomia. Käsitetasolla opitut asiat siirtyvät ohjelmasta toiseen, mutta ohjelman toimintalogiikassa tapahtuneet muutokset haittasivat oppimista.

3. Käsitemallit

Ohjelmistojen yhteydessä voidaan käsitemalli ymmärtää kolmena eri asiana: ohjelman suunnittelijan käsityksenä ohjelman toiminnasta, käyttäjän käsityksenä ohjelman toiminnasta tai tapana, jolla todellisuudessa toimii [Norman, 1986]. Huomattavaa on myös, että käyttäjän käsitemalli ei ole yhtenevä ohjelmiston suunnittelijan käsitemallin kanssa eikä perustu ohjelman todelliseen toimintaan, vaan on käyttäjän täysin oma käsitys siitä.

Puhuttaessa käsitemalleista termit menevät usein sekaisin ja samasta asiasta käytetään montaa eri termiä, kuten esim. mentaalinen malli ja skeema. Tässä tutkielmassa käsitemalli tarkoittaa nimenomaan käyttäjän muistissa muodostuvaa mallia (mentaalista mallia) tietystä ohjelmasta.

3.1 Muodostuminen

Ihmisen muistilla on kolme pääasiallista tiedon varastoimistapaa. Tieto varastoidaan joko analogisina, kuvankaltaisina esityksinä (esim. kuva omenasta), tekstimuotoisina esityksinä (esim. “omena on vihreä”) tai hajautettuina käsiteverkkoina, joissa tieto sisältyy solmuihin ja niiden välisiin suhteisiin [Preece *et al*., 1994]. Mitä paremmin ohjelman navigointirakenne ja valikkohierarkia vastaa käyttäjän muistissa sijaitsevaa tietorakennetta, sen helpompi ohjelmaa on käyttää [Sinkkonen *et al.,* 2002]. Ohjelmistojen tekijöiden olisi siis ymmärrettävä paremmin ihmisen muistin käyttämiä tiedon varastointitapoja, jotta esim. navigointirakenteet ja valikkohierarkiat vastaisivat käyttäjän muistiin varastoitunutta käsiteverkostoa.

On kuitenkin olemassa erilaisia käsityksiä siitä, kuinka tiedon haku muistista tapahtuu. Konnektionistisen käsityksen mukaan käsitteet eivät varastoidu muistiin yksittäin, vaan jokainen käsite on aina jollakin tavalla yhteydessä johonkin toiseen käsitteeseen. Näin syntyy yksinkertaisin tiedon varastointirakenne,

hierarkkinen käsiteverkosto [Preece *et al*., 1994]. Käsitteet voivat olla joko hyvinkin konkreettisia, kuten “auto” tai “koira” tai vastaavasti varsin abstrakteja, kuten “rakkaus” tai “poliittinen ilmapiiri”. Kaikille käsitteille yhteistä on kuitenkin niiden kielellinen olemus, eli jokaisella käsitteellä on jokin yksilöivä nimi. Kun johonkin havaittuun kohteeseen kytkeytyy nimi, muodostuu siitä käsite [Sinkkonen *et al.*, 2002]. Uusia käsitteitä opitaan jatkuvasti, mutta eniten toki lapsena. Käsitteet syntyvät esimerkiksi vanhempien opettaessa lapselle asioita.

Uuden käsitteen muodostuessa ihminen oppii kohteen nimen (auto) lisäksi myös siihen liittyvät aistihavainnot (miltä auto *näyttää*, miltä se *kuullostaa*) sekä muita käsitteeseen liittyviä asioita, kuten asioita, joita sillä voi tehdä (autoa voi *ajaa*, sen voi *pestä* tai *maalata*) tai minkälainen se on (kova, suuri, nopea). Näin muodostuu eri käsitteiden välille kytkentöjä, jotka muodostavat käsiteverkon. [Sinkkonen *et al.*, 2002]

Käsiteverkossa tärkeätä on paitsi käsitteiden väliset suhteet, myös suhteen symbolinen pituus eli ns. *merkitysetäisyys*. Jotkut käsitteet assosioituvat nopeammin kuin toiset. Tämä on tärkeää huomata myös ohjelmistoja laadittaessa, eli esim. navigaatiorakenteissa on syytä huomioida käsitteiden väliset kytkennät ja kytkentöjen hierarkkinen järjestys.

Käsitteiden väliset suhteet vaihtelevat ihmisillä hyvinkin paljon, koska käsitettä koskeva tieto värittyy aina ihmisen omilla kokemuksilla ja tuntemuksilla. Myös merkitysetäisyys voi vaihdella yksilöiden välillä paljon. Käyttäjiin ja heidän ajatusmaailmaansa tutustuminen on siis ensiarvoisen tärkeää ohjelmistoja suunniteltaessa.

Käyttäjän maailman ymmärtäminen on tärkeää paitsi tietorakenteiden myös tiedon esitystapojen kannalta. Reaalimaailman käsitteiden (esim. “työpöytä” tai “kortisto”) siirtäminen tietotekniseen sanastoon on tunnetusti yleistä ja tehokasta. Käytettäessä näitä analogioita ohjelmistojen suunnitteluun on kuitenkin otettava huomioon sekä käyttäjän tarpeet sekä olemassa olevan työtavan ongelmat [Preece *et al.*, 2002]. Onnistuneinta analogioiden käyttö on silloin, kun toteutettu ohjelma auttaa käyttäjää suoriutumaan entisistä tehtävistään vähintään edeltävän työtavan tehokkuudella ja nopeudella, poistaen samalla työskentelystä aikaisemmin esiintyneitä ongelmakohtia ja hankaluuksia. Esim. ensimmäisen taulukkolaskentaohjelman suunnittelija Dan Bricklin onnistui tehtävässään juuri siksi, että hän sisäisti laskentatoimen parissa toimivien henkilöiden

työssään kohtaamat ongelmat, mutta säilytti suunnitelmassaan kuitenkin tietyt tärkeät käsitteet ja toimintatavat [Preece *et al.*, 2002].

3.2 Siirtyminen

Valtaosa tietokoneohjelmia käyttävistä ihmisistä joutuu eri syistä opettelemaan uuden ohjelman käytön monta kertaa elämänsä aikana. Ohjelmaa vaihdetaan mm. kustannussyistä (halvempaan tai täysin ilmaiseen tuotteeseen), tuotetuen tai tuotteen kehityksen loppuessa, vaihdettaessa uuteen käyttöjärjestelmään tai versiopäivityksen yhteydessä.

Ohjelman vaihdos voi tapahtua monella eri tasolla. Voidaan vaihtaa täysin uuteen (kuitenkin vastaavan tehtävän suorittavaan) tuotteeseen, olemassa olevan ohjelman uuteen versioon, käyttöjärjestelmästä toiseen tai vaikkapa suomenkielisestä versiosta englanninkieliseen. Joka tapauksessa uuteen tilanteeseen liittyy aina oppimisprosessi, jossa käsitemalleilla on tärkeä rooli.

Käyttäjän siirtyessä ohjelman käytöstä toisen, vastaavan ohjelman käyttäjäksi, hän tuo uuden ohjelman oppimistilanteeseen aina mukanaan valmiin käsitemallin, käsityksen siitä kuinka tämän uuden ohjelman tulisi toimia ja kuinka sitä käytetään, perustuen aikaisempaan käyttökokemukseen. Olemassa oleva käsitemalli voi joko nopeuttaa tai hidastaa uuden ohjelman oppimisprosessia [Blakey *et al*., 1999]. Ihminen käyttää suurta osaa muistissaan olevasta tiedosta alitajuisesti. Tiedon määrä on niin suuri, että oppimistilanteessa on vaikeaa määritellä, minkä tiedon käyttö milloinkin on oleellista ja mikä turhaa [Simons, 1999].

Käsitemallien siirtyminen vanhasta ohjelmasta käy ilmi esimerkiksi näppäinoikoteiden ja valikoiden käytössä. Oletetaan, että samat toiminnot löytyvät uudesta ohjelmasta samojen näppäimien tai valikoiden takaa, vaikka näin ei aina olekaan. Jos uusi ohjelma toimiikin odottamattomasti, “väärin” tai tietty toiminto ei löydykään totutusta paikasta, käyttäjä todennäköisesti hämmentyy, turhautuu tai alkaa etsiä omasta toiminnastaan virheitä [Karat *et al*., 1986]. Tämänkaltaisten tilanteiden välttämiseksi ohjelmistosuunnittelijat ovat kehittäneet erilaisia keinoja käyttäjän avuksi. Näitä keinoja käsitellään tarkemmin seuraavassa luvussa.

4. Käsitemallien huomioiminen ohjelmistokehityksessä

Käyttäjän käsitemallien siirtyminen ohjelmien välillä on huomioitu myös ohjelmistosuunnittelijoiden keskuudessa. Käytännössä suunnittelijoilla on kuitenkin rajalliset mahdollisuudet käsitemallien huomioimiseen. Mahdollisuudet käsite-

mallien hyväksikäyttämiseen riippuvat mm. siitä, ollaanko toteuttamassa täysin uutta ohjelmaa vai ollaanko tekemässä vain uudempaa versiota olemassa olevasta ohjelmasta. Myös käytössä olevilla rahallisilla ja henkilöstöresursseilla on vaikutuksensa siihen, kuinka käyttäjän käsitemalleja pystytään suunnittelussa huomioimaan. Seuraavassa käsitellään tapoja, joilla ohjelmistosuunnittelijat voivat huomioida toisesta ohjelmasta siirtyneen käyttäjän siirtymävaiheen ongelmat. Käytännön esimerkkejä näistä ratkaisuista käsitellään tarkemmin tämän luvun alakohdissa.

Jos käytössä olevat resurssit ovat pienet, yksinkertaisin ratkaisu on kopioida jonkin olemassa olevan ohjelman käyttämää käsitemallia ja rakennetta. Käyttäjien on helppo siirtyä ohjelmasta toiseen, jos ne näyttävät ja tuntuvat käytössä samanlaisilta. Varsinkin ilmaisten, ns. open source -ohjelmistojen kehittäjät käyttävät usein tätä kopiointitapaa, sillä heillä ei ole resursseja omiin käytettävyystutkimuksiin ja uusien mallien kehittämiseen, joten helpointa on käyttää valmiita malleja [Nichols & Twidale, 2002].

Toinen tapa huomioida käyttäjän aikaisempi käyttökokemus on ohjeistus. Käyttäjälle näytetään käyttöohjeissa tapa, jolla aiemmassa ohjelmassa tehtävä suoritettiin ja toisaalta tapa, jolla sama tehtävä suoritetaan uudella ohjelmalla. Tämän tyyppinen ohjeistus helpottaa siirtymäkautta, jolloin käyttäjän mielessä on vahvana aikaisemmin käyttämänsä ohjelman työtavat, mutta uuden ohjelman tarjoamat työtavat halutaan oppia tehokkaasti.

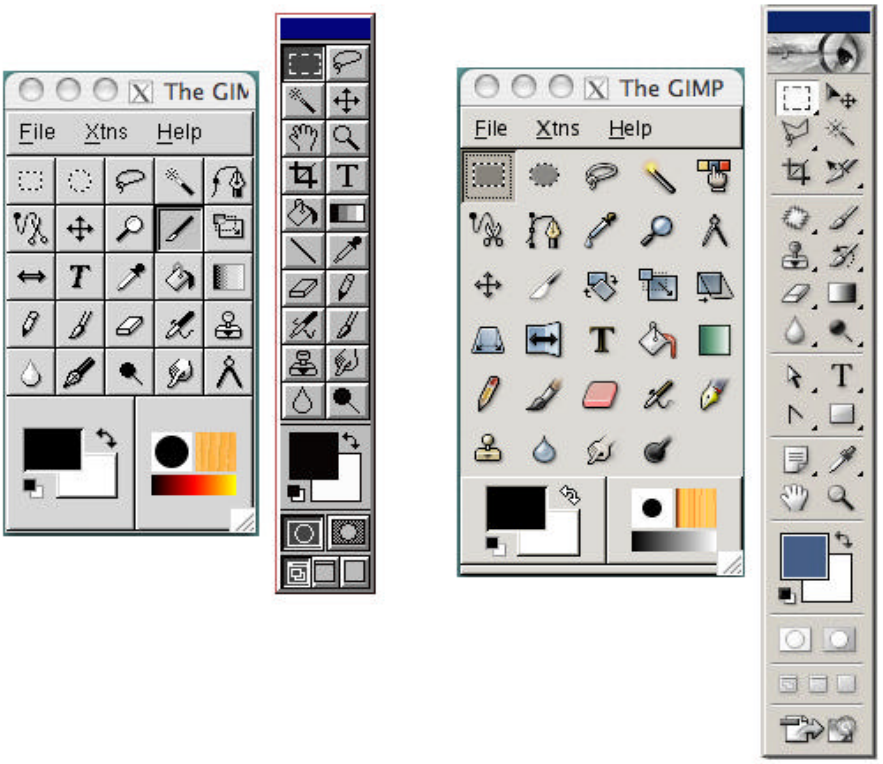
Kolmas tapa käsitemallien siirtämiseen on suoranaisesti tarjota mahdollisuus vanhaan toimintatapaan esim. näppäinoikoteiden avulla. Ohjelmassa X annetaan käyttäjälle mahdollisuus valita ohjelman asetuksista “käytä ohjelman Y näppäinkomentoja”, jonka jälkeen ohjelmaa X voidaan käyttää kuten ohjelmaa Y, tutuilla näppäinkomennoilla. Tämä tapa saattaa vähentää muutosvastarintaa, jota saattaa esiintyä uusiin toimintatapoihin perehdytyksen yhteydessä.

Seuraavissa kohdissa käsitellään muutamia käytännön esimerkkejä sovelluksista, joissa käyttäjän aikaisempi käyttökokemus otetaan huomioon. Jokainen esimerkeistä edustaa yhtä edellä mainittua ratkaisuvaihtoehtoa.

4.1 Gimp ja Adobe PhotoShop

Adoben PhotoShop on selvä markkinajohtaja ammattilaistason kuvankäsittelyohjelmista. Yksi ohjelman haittapuolista yksityishenkilön kannalta on kuitenkin sen vaatima suuri koneteho sekä verrattain korkea hinta. PhotoShopin rinnalle onkin kehitetty vastaava ilmainen ohjelma, Gimp. Gimpissä on

huomioitu käyttäjän käsitemallien siirtyminen yksinkertaisimmalla tavalla, eli valmiiden työtapojen kopioimisella toisesta ohjelmasta, tässä tapauksessa juuri PhotoShopista. Kuten kuvasta 1 nähdään, jo Gimp 1.2.5:n työkalupaletin kuvakkeet ovat lähes samat kuin PhotoShop 3.0.5:n kuvakkeet.



Kuva 1. Kuvankäsittelyohjelmien työkalupaletteja. Vasemmalta lukien: Gimp 1.2.5, PhotoShop 3.0.5, Gimp 2.0.0, PhotoShop CS 8.0

Kuvakkeiden lisäksi PhotoShopin ja Gimpin välillä on myös käsitteellisiä samankaltaisuuksia, kuten kuvien kerroksellinen muokkaus (layers).

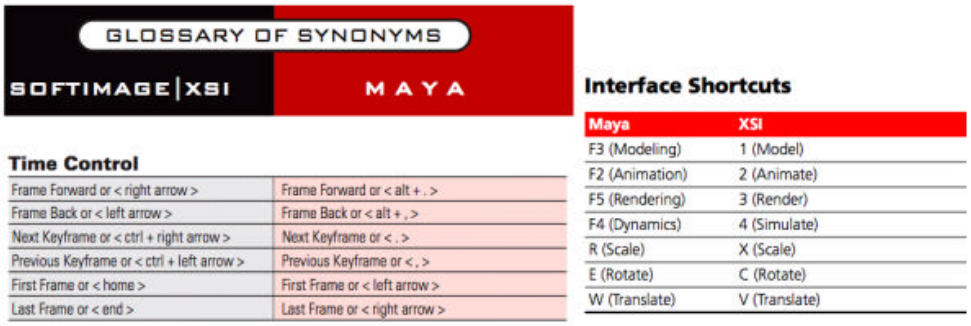
Tällä tavalla paitsi säästetään omien suunnittelijoiden työtä, myös helpotetaan käyttäjien siirtymistä ohjelmien välillä. Tässä tapauksessa käyttäjiä siis houkutellaan Gimpin käyttöön sekä ilmaisuuden että tuttujen toimintatapojen avulla.

4.2 Maya 6 ja SoftImage|XSI

Maya ja SoftImage|XSI ovat molemmat kaupallisia mallinnusohjelmia ja täten kilpailijoita keskenään. Molemmilla ohjelmilla voidaan tehdä jokseenkin samat

asiat, mutta toimintatavat eroavat ohjelmien välillä jonkin verran. Käyttäjistä (eli maksavista asiakkaista) käydään kovaa kilpailua ja siksi molemmat ohjelmat tarjoavat näkyvästi ohjeistuksen toisesta ohjelmasta siirtyville käyttäjille. Maya tarjoaa paitsi SoftImagen käyttäjille, myös monesta muusta vastaavasta ohjelmasta siirtyville käyttäjille yksityiskohtaiset ohjeet, kuinka vanhat tutut toimintatavat toimivat Mayassa [Maya]. Vastaavasti taas SoftImage|XSI:n kotisvuilta löytyy ohjeistus Mayasta siirtyville käyttäjille [SoftImage]. Kuvassa 2 on esimerkki molemmista edellä mainituista ohjeistuksista.

Näissä tapauksissa kyse on siis ohjeistuksen avulla tapahtuvasta käsitemallien siirtymisen huomioimisesta. Pyrkimys on jälleen houkutella mahdollisimman paljon käyttäjiä, etenkin toisten vastaavien ohjelmien parista.



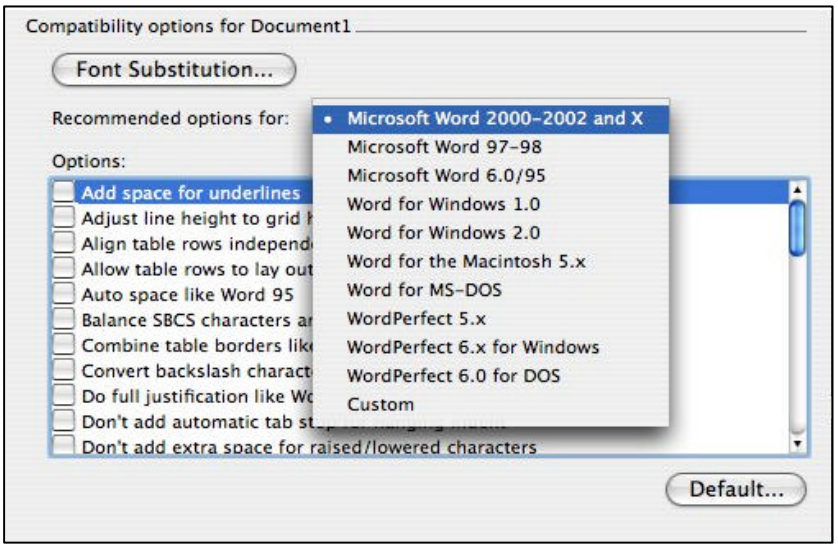
Kuva 2. Näytteet Maya- ja SoftImage|XSI -ohjelmien ohjeistuksista toisesta ohjelmasta siirtyville käyttäjille

4.3 Microsoft Word X (for Mac OS X)

Microsoftin Office on maailmanlaajuisesti käytetyin toimisto-ohjelmistopaketti, ja sen Word -kirjoitusohjelma tunnetuin kirjoitusohjelma. Wordin ominaisuuksia kehitetään ja laajennetaan jatkuvasti, mikä johtaa luonnollisesti uusien versioiden melko tiuhaan julkaisutahtiin. Koska ominaisuudet lisääntyvät ja muuttuvat versioiden välillä, joutuu käyttäjä toisinaan hämmennyksiin, kun tutulta tuntuva ohjelma ei enää toimikaan kuten aikaisemmin. Tämän vuoksi Wordissa on käytetty hyväksi kolmatta tapaa käyttäjän siirtymävaiheen helpottamiseksi, eli tarjotaan käyttäjille mahdollisuus vaihtaa vanhaan tuttuun toimintatapaan.

Tarkastellaan Mac OS X -käyttöjärjestelmän versiota Word X. Ohjelman asetuksissa on mahdollisuus valita käyttöön erilaisia ominaisuuksia jaoteltuna paitsi ominaisuuden mukaan, myös sen mukaan, missä Word:n aikaisemmassa

versiossa tämä ominaisuus on ollut käytössä. Esimerkiksi taulukoiden asettelun saa asetuksista samanlaiseksi kuin Word 97:ssä.



Kuva 3. WordX:n asetusvalikon valintalista, josta käyttäjä voi valita tietyn vanhemman version mukaiset asetukset.

Kuvassa 3 nähdään Word X:ssä valittavissa olevat aikaisempien versioiden asetusvalinnat. Valittavissa on mm. Word 1.0:n kaltaiset asetukset. Tämä on mielenkiintoista siksi, että Word 1.0 on ilmestyessään vuonna 1984 ollut saatavissa vain Macintosh-tietokoneille, sillä niissä oli tuolloin tarjolla ensimmäinen graafinen käyttöliittymä. Microsoftin omaan DOS-käyttöjärjestelmään ensimmäinen Word-versio tuli vasta myöhemmin. Tässä valintalistassa on kuitenkin valittavissa asetukset kuten Word 1.0:n Windows-versiossa, eikä kuten ensimmäisessä Mac-versiossa. Eri käyttöjärjestelmien väliset toimintatapaerot ovat siis myös huomioitavissa, ja tämänkaltaisissa asennusvalinnoissa tarjotaankin usein apua myös eri käyttöjärjestelmistä siirtyville käyttäjille.

Kuvasta 3 nähdään myös, että valittavissa on Wordin aikaisempien versioiden lisäksi myös WordPerfect-ohjelman kaltaiset asetukset. Käyttäjiä on siis avustettu hyvinkin vanhoista ohjelmista siirtymiseen. WordPerfect 5 on vuodelta 1988, jolloin tekstinkäsittelyohjelmissa oli vain murto-osa nykyisten ohjelmien ominaisuuksista. Tällaisten vanhojen ominaisuuksien tarjoaminen voidaan katsoa hyväksi asiakaspalveluksi, mutta syy on myös puhtaasti kaupallinen. Vanhoja käyttäjiä ei haluta menettää versiopäivityksen yhteydessä, vaan

halutaan pitää muutosvastarinta ja siirtymävaiheen ongelmat mahdollisimman pieninä tarjoamalla vanhan version kaltaisia asetuksia.

5. Yhteenveto

Ohjelmasta toiseen siirryttäessä uusien toimintatapojen opettelu on toisinaan hyvin aikaa vievä ja turhauttava prosessi. Ohjelmat ovat nykyisin varsin hyvin suunniteltuja noudattamaan johdonmukaisuuden periaatetta. Paitsi ohjelman sisällä, johdonmukaisuus tulisi huomioida myös tilanteessa, jossa käytössä oleva ohjelma syystä tai toisesta vaihtuu.

Tutkimukset osoittavat, että aikaisemmat käyttökokemukset vaikuttavat aina uuden ohjelman oppimisprosessiin. Aikaisemmin muodostuneiden käsitemallien merkitys voi olla joko positiivinen tai negatiivinen. Positiivisena vaikutuksena voidaan pitää tietyn ohjelmatyypin (esim. kuvankäsittelyohjelmat) peruskäsitteiden siirtymistä eli vaihtaessaan ohjelmasta toiseen vastaavanlaiseen käyttäjä tietää perustasolla mitä asioita ohjelmalla on mahdollista tehdä ja kuinka se periaatteessa tapahtuu. Negatiivinen vaikutus puolestaan on yksittäisten kielellisten asioiden (esim. valikkokomentojen) siirtyminen eli käyttäjä olettaa löytävänsä uudesta ohjelmasta tutut toiminnot samoilla nimillä kuin vanhasta tutusta ohjelmasta ja hämmentyy jos näin ei olekaan.

Näiden vaikutusten huomioiminen ohjelmistokehityksessä on suunnittelijoille suuri haaste. Käytännössä on olemassa kolme keinoa huomioida käyttäjän aikaisempi käyttökokemus: voidaan joko suoraan kopioida ulkonäkö ja toimintatavat olemassa olevasta ohjelmasta, esittää ohjelmien väliset erot ohjeistuksella tai antaa käyttäjälle mahdollisuus valita aikaisemmin käyttämänsä ohjelman kaltaiset toimintatavat. Lisäksi ihmisen muistin toiminnan ja tiedon tallennusmekanismien ymmärtäminen on ensiarvoisen tärkeää, jotta pystytään suunnittelemaan esim. ohjelmien navigointirakenteet siten, että ne vastaavat käyttäjän muistin rakenteita.

Tässä tutkielmassa käsiteltiin lyhyesti käsitemalleihin ja niiden siirtymiseen liittyvää aikaisempaa tutkimusta, käsiteyhteyksien muodostumisen psykologista perustaa sekä esiteltiin muutamia käytännön esimerkkejä. Tutkimusta jatketaan myöhemmin empiirisillä tutkimusmenetelmillä. Jatkotutkimus on tarpeen, sillä aihetta on viime vuosina tutkittu verrattain vähän. Tarkoitus on selvittää, kuinka aikaisempi käyttökokemus vaikuttaa uuden ohjelman käyttöön, erityisesti tehtä-

vien suoritusnopeuteen, navigoinnin sujuvuuteen sekä käyttäjien subjektiivisiin mielipiteisiin.

Viiteluettelo

[Blakey *et al.*, 1999] Peter Blakey, Julie Bunnel, Chris Phillips, Comprehensibility of conceptual models for end-user training. In: *Proceedings of the 10th Australasian Conference on Information Systems* (1999), 76-87.

[Foltz *et al.*, 1988] Peter W. Foltz, Susan E. Davies and Peter G. Polson, Transfer between menu systems. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (1988), 107-112.

[Karat *et al.*, 1986] John Karat, Larry Boyes, Scott Weisberger and Chuck Chafer, Transfer between word processing systems. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (1986), ACM Press, 67-71.

[Maya] *Introducing Maya 6 – the evolution of 3D*. http://www.alias.com/eng/products-services/maya/index.shtml (5.5.2004)

[Nichols & Twidale, 2002] David M. Nichols and Michael B. Twidale, The usability of open source software. *First Monday, Peer-reviewed journal on the Internet* (2002), http://www.firstmonday.dk/issues/issue8_1/nichols/ (29.1.2004)

[Nielsen, 2004] Jakob Nielsen, *Ten Usability Heuristics,* http://www.useit.com/papers/heuristic/heuristic_list.html (5.5.2004)

[Norman 1986] Donald A. Norman, Cognitive engineering. In: Donald A. Norman and Stephen W. Draper (eds.), *User Centered System Design – New Perpectives on Human-Computer Interaction.* Lawrence Elbaum Associates, 1986, 32-65.

[Preece *et al.*, 1994] Jenny Preece, Yvonne Rogers, Helen Sharp, David Benyon, Simon Holland, Tom Carey, *Human-Computer Interaction.* Addison Wesley, 1994.

[Preece *et al.*, 2002] Jennifer Preece, Yvonne Rogers, Helen Sharp, *Interaction Design -- Beyond Human-Computer Interaction.* Wiley, 2002.

[Shayo & Olfman, 1997] Conrad Shayo, Lorne Olfman, The role of training in preparing end users to learn new but similar software packages. In: *Proceedings of the 1997 ACM SIGCPR Conference on Computer Personnel Research* (1997), ACM Press, 210-223.

[Shayo & Olfman, 1998] Conrad Shayo, Lorne Olfman, The role of conceptual models in formal software training. In: *Proceedings of the 1998 ACM SIGCPR Conference on Computer Personnel Research* (1998), ACM Press, 242-253.

[Simons, 1999] P. R. J. Simons, Transfer of learning: paradoxes for learners. *International Journal of Educational Research*, **31** (1999), 577-589.

[Sinkkonen *et al.*, 2002] Irmeli Sinkkonen, Hannu Kuoppala, Jarmo Parkkinen, Raino Vastamäki, *Käytettävyyden psykologia*. Edita, 2002.

[SoftImage] *From Maya to SoftImage | XSI Quickstart*. http://www.softimage.com/products/xsi/v35/media/maya2xsi_quickstart.pdf (5.5.2004)

LR-jäsentäminen

Jaakko Korpela

Tiivistelmä

Tietokonekielten kääntämisessä LR-jäsentäminen on käytännöllinen ja monipuolinen tekniikka deterministisen kontekstittomien kielten jäsentämiseksi. Tässsä tutkielmassa tarkastellaan aiheeseen liittyvää perusteoriaa, sekä lisäksi moniselitteisten kielioppien jäsentämistä ja virheestä toipumista.

Avainsanat ja -sanonnat: kieliopit, LR-kieliopit, jäsentäjät, kääntäjät.
CR-luokka: D.3.4

1. Johdanto

Ohjelmointikielen kääntämisessä ongelmana on muuntaa lähdekielinen esitys suoritettavaksi konekoodiksi. Tällainen muunnos on niin monimutkainen, että sitä olisi hyvin vaikea ymmärtää ja hallita kokonaisuutena. Niinpä se jaetaan osiin, joista kullekin voidaan kehittää oma sitä tukeva käsitteistö ja teoria. Osat ovat tyypillisesti selaaja (joka muuntaa merkkijonon jonoksi kielen alkioita, kuten tunnisteiksi, erikoissymboleiksi, jne.) jäsentäjä (joka muuntaa jonon kielen alkioita syntaksipuuksi) ja semanttinen osa (joka muuntaa syntaksipuun suoritettavaksi koodiksi). Tavallisesti jälkimmäinen osa jaetaan vielä pienempiin osiin. Tässä tutkimuksessa keskitytään jäsentäjien teoriaan.

Jäsentäjät pyrkivät selvittämään ohjelman syntaktisen rakenteen. Tämä tehdään kielioppien avulla. Jäsentäjä saa syötteekseen jonon perussymboleita, jonka selaaja on muodostanut syöteohjelmasta. Jäsentäjän tehtävänä on päättää, voidaanko (ja kuinka) jäsentäjän syöte johtaa kieliopin alkumerkistä kieliopin sääntöjen avulla. Jäsentäjä joutuu tekemään kahdenlaisia päätöksiä, kun se pyrkii johtamaan yhteyden kieliopin alkusymbolin ja saamansa syötteen välille. Sen pitää päättää, missä järjestyksessä jäsentäminen tapahtuu, ja sen pitää valita, mitä kieliopin sääntöä missäkin vaiheessa käytetään. Jäsentämisen perusongelma on se, miten nämä päätökset tehdään mahdollisimman vähän laskentaresursseja käyttäen. Yleensä jäsennys-johdon järjestys riippuu siitä jäsennysmenetelmästä, johon jäsentäjä perustuu. Sen tekemät valinnat riippuvat jäsennettävästä merkkijonosta. Jäsentäminen voidaan tehdä joko *osittavalla* (top-down) tavalla tai *kokoavalla* (bottom-up) tavalla. Osittava jäsennys lähtee liikkeelle kieliopin alkumerkistä ja etenee kohti syötettä. Kokoava jäsennys aloittaa syötteestä ja pyrkii kohti alkumerkkiä.

LR-jäsentäjä on jäsentäjätyyppi kontekstittomille kieliopeille, jota yleisesti käytetään ohjelmointikielten kääntäjissä. LR-jäsentäjä on kokoava jäsentäjä, joka lukee syötettään vasemmalta oikealle ja pyrkii käänteistä oikeaa johtoa muodostaen kohti kieliopin

alkumerkkiä. LR-jäsentämisellä on lukuisia etuja ja se on pääasiallinen jäsennysmenetelmä monille ohjelmointikielille. Miltei kaikki ohjelmointikielet voidaan jäsentää käyttäen LR-jäsentäjää. LR-jäsentäjät voidaan toteuttaa tehokkaasti ja on olemassa työkaluja (yacc, bison, occs) jäsentäjän automaattiseen generoimiseen tiettyyn kielioppiin perustuen. Kaikista syötettä vasemmalta oikealle lukevista jäsentäjistä LR-jäsentäjä havaitsee syntaktiset virheet aikaisimmassa mahdollisessa vaiheessa.

Tämän tutkimustyön luku 2 perustuu lähteisiin [Hopcroft and Ullman, 1979] ja [Aho and Johnson, 1974]. Luvut 3, 4, 5 ja 6 perustuu lähteeseen [Aho and Johnson, 1974]. Luku 7 perustuu lähteisiin [Aho and Johnson, 1974], [Aho et al., 1985] ja [Barrett and Couch, 1979]. Luku 9 perustuu lähteisiin [Aho et al., 1985] ja [Appel, 1998]. Luvut 10 ja 11 perustuvat lähteisiin [Aho and Johnson, 1974], [Aho et al., 1985] ja [Appel, 1998].

2. Alustus

2.1 Kieliopit

Kielioppeja käytetään määrittelemään kieli ja asettamaan rakenne jokaiselle kielen lauseelle. Lauseella tarkoitetaan tässä kielen symboleista koostuvaa merkkijonoa ja kielellä tarkoitetaan lauseiden joukkoa. Kielioppi on joukko *sääntöjä*

$$\textit{symboli} \rightarrow \textit{symboli}\ \ \textit{symboli}\ \ \textit{symboli}\ ,$$

jossa säännön (nuolen) oikealla puolella on symboleita nolla tai enemmän. Jokainen symboli on joko *perusmerkki* (terminal) tai *apumerkki* (non-terminal). Perusmerkki on aakkoston symboli ja apumerkki on merkki, joka voidaan kieliopin sääntöjen mukaan korvata perusmerkeillä. Perusmerkkejä merkitään pienillä kirjaimilla ja apumerkkejä isoilla kirjaimilla. Perus- ja apumerkeistä koostuvia merkkijonoja merkitään kreikkalaisilla kirjaimilla. Kieliopin sanotaan olevan *kontekstiton*, jos sen jokaisen säännön vasemmalla puolella on aina vain yksi apumerkki, eikä yhtään perusmerkkiä. Säännön oikealle puolelle ei ole rajoituksia. Alla on esimerkki eräästä kontekstittomasta kieliopista G_1:

$$\begin{aligned} S &\rightarrow bAb \\ S &\rightarrow Ac \\ S &\rightarrow ab \\ A &\rightarrow a\ . \end{aligned}$$

Kieliopissa määritellään joku apumerkeistä *aloitusmerkiksi*. Tästä merkistä lähtien muodostetaan kieliopin lauseita. Usein tätä (kuten yllä olevassa kieliopissa) aloitusmerkkiä merkitään kirjaimella S.

Yllä olevassa kieliopissa on siis neljä sääntöä. Perusmerkit ovat a ja b. Apumerkit ovat S ja A. Eräs kielen lause on ab, joka saadaan korvaamalla aloitusmerkki kolmannen säännön mukaan merkkijonolla ab. Eräs toinen kielen lause on bab. Tämä saadaan

korvaamalla ensin kieliopin aloitusmerkki merkkijonolla bAb ensimmäisen säännön mukaan. Seuraavaksi apumerkki A korvataan perusmerkillä a neljännen säännön mukaan. Tästä saadaan merkkijono bab. Näin ollaan *jäsennetty* kaksi kielen lausetta.

2.2 Jäsentäminen

Jos aAg on kieliopin symboleista koostuva merkkijono ja $A \rightarrow b$ on kieliopin sääntö, voidaan kirjoittaa

$$aAg \Rightarrow abg .$$

Tällöin sanotaan, että abg jäsentyy suoraan merkkijonosta aAg. Sellaista merkkijonoista koostuvaa jonoa

$$a_0, a_1, ..., a_n ,$$

että $a_{i-1} \Rightarrow a_i$ $(1 \leq i \leq n)$, sanotaan a_n:n jäsennykseksi a_0:sta.

Jäsennyksellä tarkoitetaan siis sitä, että aloitetaan kieliopin aloitusmerkistä ja tämän jälkeen toistuivasti korvataan mikä tahansa säännön oikealla puolella esiintyvä apumerkki jonkun kieliopin säännön mukaan. Aloitusmerkki on *kieliopillinen muoto* (sentential form). Merkkijono, joka on jäsennettävissä aloitusmerkistä, on myöskin kieliopillinen muoto. Kieliopillista muotoa, joka sisältää vain perusmerkkejä, sanotaan *kieliopin generoimaksi lauseeksi*.

Jos jäsennyksiä tehtäessä korvataan aina vasemmanpuoleisin apumerkki, puhutaan *vasemmanpuoleisimmasta jäsennyksestä* (leftmost derivation) tai *vasemmasta johdosta*. Vastavasti korvaamalla aina oikeanpuoleisin apumerkki, puhutaan *oikeanpuoleisimmasta jäsennyksestä* (tai *oikeasta johdosta*). Lausetta, joka on generoitu kieliopista käyttäen vasemmanpuoleisinta jäsennystä, sanotaan *vasemmanpuoleiseksi kieliopilliseksi muodoksi*. Oikeanpuoleisen jäsennykseksen generoimaa lausetta sanotaan vastaavasti *oikeanpuoleiseksi kieliopilliseksi muodoksi*.

Jos aAw on oikeanpuoleinen kieliopillinen muoto, jossa w koostuu perusmerkeistä, ja $aAw \Rightarrow abw$, niin b on abw:n *kahva* (handle). Esimerkiksi a on oikeanpuoleisen kieliopillisen muodon bab kahva kieliopissa G_1, sillä $bAb \Rightarrow bab$.

Alkuosan ab oikeanpuoleisesta kieliopillisesta muodosta abw sanotaan olevan kieliopin *kelvollinen alkuosa* (viable prefix). Esimerkiksi kieliopissa G_1 bA on kelvollinen alkuosa, koska se on oikean kieliopillisen muodon bAb alkuosa.

Kontekstittoman kieliopin jäsennys voidaan esittää *jäsennyspuuna*. Jäsennyspuu rakennetaan liittämällä jokainen merkki jäsennyksessä siihen merkkiin josta se on jäsennetty. Puun juurena on aina kieliopin aloitusmerkki.

Jos kieliopissa samasta lauseesta voidaan muodostaa kaksi erilaista jäsennyspuuta, kieliopin sanotaan olevan *moniselitteinen.* Muussa tapauksessa kielioppi on *yksiselitteinen*. Kielioppi G_1 on yksiselitteinen.

3. LR-jäsentäjä

Jäsentäjä voidaan kuvata laitteeksi, joka pyrkii muodostamaan syötteenä annetusta merkkijonosta jäsennyspuun, jonka lehdet luettuna vasemmalta oikealle muodostavat annetun merkkijonon. Jos tälläinen puu voidaan muodostaa, on merkkijono kieliopin generoima.

Tässä tutkielmassa rajoitutaan jäsentäjiin, jotka tutkivat syötettä vasemmalta oikealle merkki kerrallaan ja pyrkivät muodostamaan jäsennyspuun lehdistä juureen (bottom-up). LR-jäsentäjä toimii muodostamalla syötteen oikeanpuoleisimman jäsennyksen käänteisessä järjestyksessä. Tässä työssä rajoitutaan kuvaamaan yhden LR-jäsentäjäluokan, LR(1)-jäsentäjän toimintaa.

LR-jäsentäjä käsittelee toimintansa aikana jonoa, joka koostuu osittain muodostetuista puista. Tätä jonoa kutsutaan *metsäksi*. LR-jäsentäjän toiminnan aluksi metsäksi alustetaan yksi puu, joka sisältää ainoastaan syötteen ensimmäisen merkin. LR-jäsentäjän toiminnan lopuksi (olettaen, että syöte on kieliopin syntaksin mukainen) metsä sisältää merkkijonon jäsennyspuun. Jos syöte ei ole kieliopin syntaksin mukainen, puun muodostaminen pysähtyy ja kohtaa, jossa prosessi epäonnistuu, voidaan käyttää osoittamaan mahdollinen virhekohta.

LR-jäsentäjä voi tehdä neljä erilaista jäsennys-toimintoa: *siirrä* (shift), *redusoi* (reduce), *hyväksy* (ilmoittaa jäsennyksen päättyneen onnistuneesti) tai ilmoita *virheestä.*

Siirrä-toiminnossa seuraava merkki otetaan syötteestä. Uusi puu, jonka ainoana solmuna on kyseinen merkki, lisätään metsän viimeiseksi (oikeanpuoleisimmaksi) puuksi.

Redusoi-toiminto perustuu johonkin kieliopin sääntöön

$$A \rightarrow X_1 X_2,...,X_n,$$

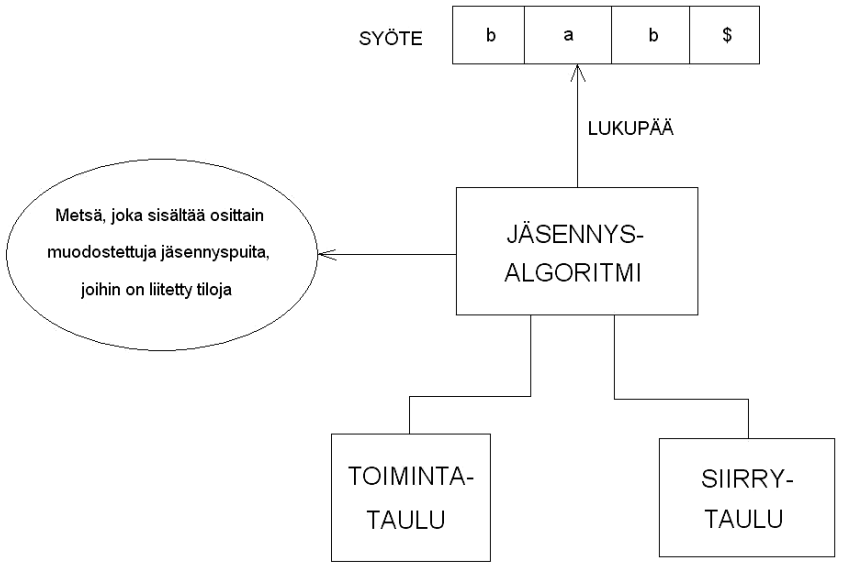
ja siinä otetaan käsittelyyn metsän oikeanpuoleisimmat puut, joiden juurina on merkit $X_1 X_2,...,X_n$. Jokainen X_i (i = 1,...,n) edustaa perusmerkkiä tai apumerkkiä. Redusoi-toiminnossa metsää muokataan siten, että siihen lisätään uusi solmu A ja solmut $X_1 X_2,...,X_n$ tulevat A:n lapsiksi. A on tämän jälkeen viimeinen puu metsässä.

LR-jäsentäjä suorittaa toistuvasti jäsennystoimintoja, kunnes päästään hyväksymis- tai virhe-tilanteeseen. LR-jäsentäjän toiminnot määräytyvät edellisistä toiminnoista ja syötteen vielä lukemattomista merkeistä. LR-jäsentäjää, joka tutkii vain seuraavaa

syötemerkkiä toimintaa valittaessa, sanotaan LR(1)-jäsentäjäksi. Vastaavasti k:ta seuraavaa ($k \geq 0$) merkkiä tarkastelevaa sanotaan LR(k)-jäsentäjäksi. Jäsennystoiminnon päättämiseksi LR-jäsentäjässä jokaisen metsän puun juureen liitetään numero, jota kutsutaan *tilaksi* (state). Metsän viimeisen puun tilaa sanotaan *nykyiseksi tilaksi* (current state). Lisäksi metsän vasemmalla puolelle on alkutila, jonka perusteella päätetään ensimmäinen jäsennystoiminto. LR(1)-jäsentäjässä nykyinen tila ja seuraava syöte merkkijonossa määräävät seuraavan jäsennystoiminnon.

4. LR-jäsentäjän arkkitehtuuri

Jäsentäjä koostuu *syötepuskurista*, josta syötettä luetaan merkki kerrallaan, lukupäästä, joka lukee aina jotain kohtaa syötteestä, metsästä, joka koostuu osittain muodostetuista puista ja niihin liittyvistä tiloista, sekä *toiminta-* ja *siirry-taulusta* (action table, goto table). Kuvassa 1 on esitetty kaavakuva LR-jäsentäjän rakenteesta.



Kuva 1. LR-jäsentäjän arkkitehtuuri.

Toiminta-taulun avulla päätetään jäsentäjän seuraavasta toiminnasta. Päätös tehdään nykyisen tilan ja seuraavan syötemerkin perusteella. Jäsennyksen alkaessa nykyinen tila on 0 ja lukupää osoittaa syötteen ensimmäiseen merkkiin. Taulukossa 1 on kuvattu toiminta-taulu kieliopille G_1. Siinä on kuvattu eri tilat (rivit) ja niihin liittyvät jäsennystoiminnot, kun eri tiloissa luetaan tietty syötemerkki (sarakkeet). Merkki ’$’ kuvaa syötteen päättymistä ja se lisätään jokaisen syötteen perään. ’Red. n’ kuvaa, minkä kieliopin säännön mukaan redusointi tehdään.

	a	**b**	**c**	**$**
0	Siirrä	Siirrä	Virhe	Virhe
1	Virhe	Virhe	Virhe	Hyväksy
2	Virhe	Virhe	Siirrä	Virhe
3	Virhe	Siirrä	Red. 4	Virhe
4	Siirrä	Virhe	Virhe	Virhe
5	Virhe	Virhe	Virhe	Red. 2
6	Virhe	Virhe	Virhe	Red. 3
7	Virhe	Siirrä	Virhe	Virhe
8	Virhe	Red. 4	Virhe	Virhe
9	Virhe	Virhe	Virhe	Red. 1

Taulukko 1. Toiminta-taulu kieliopille G_1.

Jokaisen siirrä- tai redusoi-toiminnon jälkeen on päätettävä, mikä tila liitetään sen puun juureen, joka juuri lisättiin metsään (metsän oikealle puolelle). Tämä päätös tehdään siirry-taulun perusteella. Taulukossa 2 on esitetty siirry-taulu kieliopille G_1. Rivit kuvaavat oikeanpuoleisinta tilaa ja sarakkeet uuden juuren arvoa.

	A	**S**	**a**	**b**	**c**
0	1	1	3	4	
1					
2					5
3	7			6	
4			8		
5					
6					
7				9	
8					
9					

Taulukko 2. Siirry-taulu kieliopille G_1.

5. Jäsennysalgoritmi

LR-jäsentäjä tietylle kieliopille on täydellisesti määritelty, kun sille on muodostettu toiminta- ja siirry-taulut. Tarkastellaan seuraavaksi LR(1)-jäsentäjän jäsennysalgoritmia.

Alustus: Aseta alkutila tyhjään metsään; alkutila on jäsentäjän nykyinen tila.

Jäsennys-toiminto: Tutki toiminta-taulua ja päätä toiminnosta nykyisen tilan ja syötemerkin perusteella. Toimintovaihtoehdot ovat siirrä, redusoi, hyväksy tai virhe.

Siirrä: Poista syötemerkki puskurista ja aseta metsään uusi solmu merkittynä tällä syötemerkillä. Aseta solmulle tila

siirry(nykyinen tila, syötemerkki),

ja tee tästä tilasta uusi nykyinen tila. Toista kohta *Jäsennys-toiminto*.

Redusoi: Jos toimintoon liittyvä sääntö on

$$A \rightarrow X_1 X_2 ... X_n ,$$

lisää uusi solmu A metsään, ja tee solmuista $X_1 X_2 ,..., X_n$ solmun A lapsia. Poista A:n lapsiin liittyvä tilat. Olkoon s nyt oikeinpuoleisin solmu (heti A:sta vasemmalle). Liitä A:han tila

siirry(s, A),

ja tee tästä uusi nykyinen tila. Toista kohta *Jäsennys-toiminto*.

Hyväksy: Pysähdy. Täydellinen jäsennyspuu on muodostettu.

Virhe: Syötteessä on virhe. Ilmoita virheestä ja yritä jatkaa jäsentämistä (tätä aihetta käsitellään luvussa 13).

Edellä jäsennysalgoritmissa tiloja tarkasteltaessa huomioitiin aina vain oikeanpuoleisin tila. Käytännössä ei ole välttämätöntä rakentaa jäsennnyspuuta eksplisiittisesti, vaan voidaan käyttää pinoa. Tällöin siirrä-toiminnossa työnnetään pinoon vain uusi tila. Redusoi-toiminnossa taas korvataan pinon pinnalta uuden tilan jälkeläiset uudella tilalla. Näin pino sisältää vain tiloja. Jos jäsennyspuu halutaan rakentaa, tämä voidaan helposti toteuttaa pinoamalla jokaisen tilan kanssa tilaan liittyvän puun juuri. Tästä eteenpäin oletetaan, että LR-jäsentäjässä käytetään pinototeutusta.

Kuinka toiminta- ja siirry-taulut rakennetaan tietylle kieliopille? Kielioppia, jolle nämä LR-taulut voidaan tuottaa, sanotaan LR-kieliopiksi. Kaikki kontekstittomat kieliopit eivät ole LR-kielioppeja. Tällaisia ei kuitenkaan yleensä tarvita ohjelmointikielten määrittelyissä.

Intuitiivisesti ajatellen kielioppi on LR-kielioppi, jos vasemmalta oikealle syötettä lukeva siirrä- ja redusoi-toimintoja tekevä jäsentäjä voi tunnistaa kahvat pinon pinnalta. LR-jäsentäjän ei tarvitse käydä läpi koko pinoa tunnistaakseen kahvan pinnalta. Pinon pinnalla oleva tila sisältää kaiken tarpeellisen tiedon; LR-jäsentäjä voi päätellä pinon pinnalla olevasta tilasta kaiken, mitä sen pinon sisällöstä tarvitsee tietää.

Toinen asia, jota LR-jäsentäjä voi käyttää seuraavan toiminnan päättämisessä, on seuraavat k syötemerkkiä. Käytännössä on merkitystä vain tapauksilla k = 0 ja k = 1. LR-kielioppia, joka voidaan jäsentää tutkimalla k:ta seuraavaa syötemerkkiä, sanotaan LR(k)-kieliopiksi.

6. LR-jäsennystaulujen tiivistäminen

Tyypillinen ohjelmointikielen kielioppi, joka sisältää sisältää 50-100 perusmerkkiä ja 100 sääntöä voi tuottaa siirry-taulun, joka sisältää useita satoja tiloja sekä toiminta-taulun, joka voi helposti sisältää 20000 kohtaa. Toiminta- ja siirry-tauluja voidaan tiivistää usein eri tavoin. Tässä työssä käytetään seuraavaa tiivistystapaa.

Jäsennys-toiminto perusmerkille x tilassa t on siirrä, jos ja vain jos siirry-taulun kohta **siirry**(*nykyinen tila*, *syötemerkki*) ei ole tyhjä. Siirry-toiminto voidaankin liittää siirrä-toimintoon käyttämällä merkintää

siirrä t.

Tämä tarkoittaa sitä, että suoritetaan siirrä-toiminto ja viedään pinoon tila t. Kun perus-merkkeihin liittyvät siirry-toiminnot koodataan osaksi toiminta-taulua, siirry-taulussa täytyy huolehtia vain niistä siirry-toiminnoista, jotka liittyvät apumerkkeihin. Nämä koo-dataan sarakkeittain. Jos siis siirry-taulussa apumerkin A sarakkeessa on ei-tyhjiä kohtia liittyen joihinkin (mahdollisesti kaikkiin) tiloihin s_1, s_2,..., s_n, missä n on sarakkeiden lukumäärä, ja näihin tiloihin taulussa liitetään tilat s_i' = **siirry**(s_i, A) (i = 1,...,n), niin sarake A koodataan seuraavasti

A: **if**(tila = s_1) **siirry** s_1'
:
if(tila = s_n) **siirry** s_n'.

Tällä tavalla koodataan jokainen apumerkki siirry-taulussa.

Toiminta-taulu koodataan vastaavalla tavalla, mutta riveittäin. Jos siis yhdellä toiminta-taulun rivillä r perusmerkkeihin a_1, ..., a_n liittyvät toiminnot *toiminto*$_1$, ..., *toiminto*$_n$, niin saadaan

r: **if**(syöte = a_1) **toiminto**$_1$
:
if(syöte = a_n) **toiminto**$_n$.

Siirrä-toimintoihin liitetään siis **siirry**(*nykyinen tila*, *syötemerkki*). Samaan tapaan redusoinnista koodataan resusoinnissa käytyn säännön numero.

Toiminta-taulun esittämisessä saavutetaan huomattavaa tilansäästöä, kun tiloihin liitetään *oletusarvoinen toiminto*. Rivi, joka sisälsi sekä virhe- että redusointi-toimintoja, voidaan tällä esitystavalla esittää niin, että ko. toiminnot korvataan tähän tilaan liittyvällä redusoi-toiminnalla (rivi voi sisältää vain yhdenlaisen redusointi-toiminnon). Tämän seurauksena jäsentäjä saattaa tehdä ylimääräisiä redusointeja, mutta ilmoittaa kuitenkin virheestä ennen uuden syötemerkin siirtämistä pinoon. Tämä oletusarvoinen redusoi-toiminto sijoitetaan siirrä- ja hyväksy-toimintojen jälkeen. Jos tila ei sisällä yhtään redusoi-toimintoa, oletusarvoinen toiminto on *virhe.*

7. Jäsentäjän muodostaminen

Seuraavaksi kuvataan tapa muodostaa toiminta- ja siirry-taulut *LALR*(1)-kieliopeille. LALR(1)-kieliopit ovat LR(1)-kielioppien osajoukko. Tämä ei kuitenkaa ole suuri rajoite, sillä käytännössä on vaikeata löytää LR(1)-kielioppia, joka ei olisi LALR(1)-kielioppi. Etuna LALR(1)-jäsentäjän muodostamisessa LR(1)-jäsentäjän muodostamiseen verrattuna on se, että LALR(1)-jäsentäjä on kooltaan huomattavasti pienempi ja siten nopeammin muodostettavissa. Suuri osa ohjelmointikielistä on jäsennettävissä LALR(1)-kieliopin mukaan.

7.1 Alkiot

Kuten edellä huomattiin, LR-jäsentäjän toiminta riippuu sen nykyisestä tilasta. Tämä tila pitää sisällään tiedon luetusta syötteestä ja sen avulla määritellään jäsentäjän tulevat toiminnot. Tilan voidaan ajatella edustavan kelvollista alkuosaa. Jäsennyksen jokaisessa vaiheessa metsän juurien yhdistäminen vasemmalta oikealle muodostaa kelvollisen alkuosan ja nykyinen tila edustaa tätä.

Tässä yhteydessä esitellään *alkion* (item) käsite. Alkio on sääntö, jonka oikealle puolelle on lisätty piste johonkin (mihin tahansa) kohtaan. Piste kuvaa kuinka paljon syötteestä on tunnistettu.
Esim. alkio

$$[\, A \rightarrow a\,.\,b\,]$$

osoittaa, että merkkijono a on luettu syötteestä. Jos tämän jälkeen luetaan merkkijono b, on mahdollista suorittaa redusointi säännön $A \rightarrow ab$ mukaan.

Alkion $[\, A \rightarrow a\,.\,b\,]$ sanotaan olevan *pätevä alkio* (valid item) jollekin kelvolliselle alku-osalle γa, jos myös γA on kelvollinen alkuosa. Yleisesti ottaen tietylle kelvolliselle alkuosalle voi olla useita kelvollisia alkioita.

LALR(1)-jäsentäjän muodostamisessa määrätään pätevät alkiot jokaiselle kelvolliselle alkuosalle (alkaen tyhjästä merkkijonosta). V(γ):llä merkitään pätevien alkioiden joukkoa kelvolliselle alkuosalle γ. Jokainen V(γ) vastaa yhtä jäsentäjän tilaa.

Oletetaan, että kelvolliselle alkuosalle γ on määrätty pätevien alkioiden joukko V(γ). Olkoon X kieliopin jokin merkki. V(γ X) lasketaan V(γ):stä seuraavasti.

1. Jokaista V(γ):n muotoa [A $\rightarrow \alpha . X\beta$] olevaa alkiota kohti lisätään V(γ X):ään alkio [A $\rightarrow \alpha X . \beta$].

2. Lasketaan *sulkeuma* joukolle V(γ X). Jokaista [B $\rightarrow \alpha . C\beta$], missä C on apumerkki, olevaa alkiota kohti, lisätään V(γ X):ään alkiot [$C \rightarrow . \alpha_1$], [$C \rightarrow . \alpha_2$],...,[$C \rightarrow . \alpha_n$], missä $C \rightarrow \alpha_1$, ..., $C \rightarrow \alpha_n$ ovat kieliopin sääntöjä. Jos jokin näistä alkioita on jo V(γ X):ssä, sitä ei lisätä joukkoon. Prosessia jatketaan, kunnes V(γ X):ään ei enää muutu.

Jos γ ei ole kelvollinen alkuosa kieliopille G, V(γ) on tyhjä joukko.

7.2 Kurkistusjoukko

Alkiota, jossa piste on säännön oikeassa päässä, sanotaan *valmiiksi alkioksi* (completed item). Valmis alkio [$A \rightarrow \alpha .$] viittaa siihen, että on mahdollista suorittaa redusointi säännön $A \rightarrow \alpha$ mukaan. Jos alkiojoukko sisältää usemman kuin yhden valmiin alkion, on generoitava ehdollisia redusoi-toimintoja kaikille paitsi yhdelle valmiille alkiolle, eli on määriteltävä syötemerkit, joissa annettu redusoi-toiminto on sallittu. Tämän avulla osataan valita oikea redusointi, jos niitä on useita mahdollisia.

Oletetaan, että alkio [$A \rightarrow \alpha . \beta$] on pätevä jollekin kelvolliselle alkuosalle γA. Syötemerkkiä a sanotaan *soveltuvaksi* (applicable) alkiolle [$A \rightarrow \alpha . \beta$], jos jollekin (mahdollisesti tyhjälle) perusmerkkijonolle w sekä $\gamma\alpha\beta aw$ että $\gamma A a w$ ovat oikeanpuoleisia kieliopillisia muotoja. Niiden symbolien joukkoa, jotka ovat soveltuvia jollekin alkiolle, sanotaan alkion *kurkistusjoukoksi* (lookahead set). Alkioita ja niitä vastaavia kurkistusjoukkoja merkitään seuraavasti:

([$A \rightarrow \alpha . \beta$], {a $_1$,...,a $_n$ }).

Sääntöosaa sanotaan alkion *ytimeksi* (core). Kurkistusjoukko voi sisältää kieliopin perusmerkkien lisäksi loppumerkin $.

Pätevien alkioiden laskeminen kurkistusjoukon kanssa on hieman monimutkaisempaa kuin edellä pelkkiä päteviä alkioita laskettaessa. Muotoa (X on perus - tai apumerkki)

([$A \rightarrow \alpha . X\beta$], L)

olevasta alkiosta saadaan uusi alkio

([$A \rightarrow aX.b$], L),

eli kurkistusjoukko pysyy samana. Sulkeuman laskeminen on jonkin verran vaikeampaa. Muotoa

([$A \rightarrow a.Bb$], L)

olevasta alkiosta, missä B on jokin apumerkki, on lisättävä uuteen alkiojoukkoon alkiot, jotka ovat muoto a

([$B \rightarrow .d$], L'),

missä $B \rightarrow d$ on jokin kieliopin sääntö. Uusi kurkistusjoukko L' sisältää kaikki perusmerkit jotka ovat ensimmäisiä merkkejä jossain lauseessa, joka on jäsennettävissä mistä tahansa merkkijonosta muotoa b a, missä a on merkki L:ssä (b voi olla myös tyhjä merkkijono). Oletetaan, että käytössä on näiden perusmerkkien laskemista varten käytössä algoritmi FIRST. Alkiot, joilla on sama ydin, mutta eri kurkistusjoukot, yhdistetään yhdeksi alkioksi muodostamalla kurkistusjoukkojen unioni.

Tavoitteena on muodostaa alkiojoukot I_i, i = 0,1, ... ,n, alkaen joukosta I_0. Jokaista alkiojoukkoa I_i ja jokaista kieliopin merkkiä X kohti edellä kuvattu menetelmä muodostaa uuden alkiojoukon, josta käytetään merkintää **siirry**(I, X). Alkioiden joukko joka saadaan I_0:sta siirry-toimintojen avulla, sanotaan *saavutettavaksi alkiojoukoksi* (accessible set of items). Saavutettava alkiojoukko muodostetaan laskemalla **siirry**(I, X) jokaiselle saavutettavalle alkiojoukolle I_i ja jokaiselle kieliopin merkille X. Kieliopin sääntöjä on rajoitettu määrä, joten alkioita on rajoitetu määrä. Niinpä tämä prosessi lopulta päättyy. Järjestys, jossa alkiojoukot muodostetaan on yhdentekevä, samoin kuin alkiojoukolle annettava nimi. Alkiojoukon nimi vastaa jäsentäjän yhtä tilaa.

7.3 Alkiojoukkojen muodostaminen

Alkiojoukkojen laskeminen kieliopista G tapahtuu seuraavan algoritmin mukaan:

1. Laajenna G:tä uudella alkusäännöllä

   HYVÄKSY $\rightarrow$ S,

   missä S on kieliopin alkumerkki.

2. Olkoon I joukko, joka sisältää yhden alkion
([HYVÄKSY $\rightarrow$.S], {$})

Muodosta joukon I sulkeuma I_0.

3. Alusta saavutettava alkiojoukko C sisältämään vain I_0:n.

4. Jokaiselle C:n alkiolle I ja jokaiselle kieliopin merkille X, laske I' = **siirry**(I, X). Vaihtoehtoja on kolme:

a. I' on jo C:ssä. Tässä tapauksessa älä tee mitään.

b. Jos ytimien joukot I':ssa ovat erilaisia verrattuna ytimien joukkoon C:n alkioiden joukossa, lisää I' joukkoon C.

c. Jos I':n ytimien joukko on sama kuin jonkun I'':n ytimien joukko, joka on C:ssä, mutta I' ≠ I'' (eli kurkistusjoukot ovat erilaisia), muodosta I''' seuraavasti:

$$([\ A \rightarrow a\,.\,b\ ],\ L_1 \cup L_2)$$

missä ([$A \rightarrow a\,.\,b$], L_1) on I':ssa ja ([$A \rightarrow a\,.\,b$], L_2) on I'':ssa. Korvaa I'' I''':lla C:ssä.

5. Toista kohtaa 4, kunnes uusia alkiojoukkoa ei voida enää lisätä C:hen. C:tä sanotaan kieliopin G *LALR(1)-kokoelmaksi* alkiojoukkoja G:lle.

Saavutettava alkiojoukko voidaan kuvata suunnattuna graafina. Solmut edustavat alkiojoukkoja eli tiloja. Nuolet edustavat kieliopin merkkejä, joiden mukaan seuraava tila on laskettu.

7.4 Toiminta- ja siirry-taulujen muodostaminen

Toiminta- ja siirry-taulut muodostetaan saavutettavasta alkiojoukosta. Jokaisesta alkiojoukosta I_s generoidaan toiminta-tauluun toimintoja. Alkio, jonka ydin on muotoa (a on perusmerkki)

[$A \rightarrow a\,.\,ab$]

tuottaa toiminnon

if (syöte = a) **siirrä** t,

missä **siirry**(I_s, a) = I_t.

Jos alkiojoukko sisältää vain yhden valmiin alkion [$A \rightarrow a\,.$], voidaan tilaan lisätä jäsennys-toiminto

redusoi: $A \rightarrow a$,

joka sijoitetaan kaikkien alkiojoukon tuottamien siirrä- ja hyväksy-toimintojen jälkeen. Jos alkiojoukko sisältää useimman kuin yhden valmiin alkion, on kurkistusjoukon avulla päätettävä, millä syötemerkillä redusoi-toiminto on mahdollinen.

Valmis alkio [HYVÄKSY $\rightarrow$ S.] tuottaa toiminnon

if(syöte = $) **hyväksy.**

Jos alkiojoukko ei tuota yhtään redusoi-toimintoa, luodaan oletusarvoinen redusoi-toiminto.

Siirry-taulua käytetään uuden tilan laskemiseen redusoinnin jälkeen. Jos apumerkillä A on siirry-graafissa toiminnot:

siirry(I_{s1}, A) = I_{t1}
siirry(I_{s2} , A) = I_{t2}
:
siirry(I_{sn}, A) = I_{tn} ,

niin siirry-taulun sarake A tulee muotoon

A: **if**(tila = s_1) **siirry** = t_1
if(tila = s_2) **siirry** = t_2
:
if(tila = s_n) **siirry** = t_n .

Kielioppi voi olla liian monimutkainen tälle jäsennysmenetelmälle tai kielioppi voi olla moniselitteinen; tämä aiheuttaa *jäsennyskonflikteja*.

Oletetaan, että alkiojoukossa I_s syötemerkki a tuottaa siirrä-toiminnon, koska **siirry**(I_s, a) on tuottanut jonkun tilan. Oletetaan myös , että I_s sisältää valmiin alkion

([$A \rightarrow a$.], L).

Tällöin ei tiedetä, pitääkö suorittaa siirrä- vai redusoi-toiminto. Kyseessä on *siirrä-redusoi -konflikti.*

Jos samassa tilassa on kaksi valmista alkiota, joilla on sama perusmerkki kurkistus-joukossa, on mahdollista redusoida kahden eri säännön mukaan. Kyseessä on redusoi-redusoi -konflikti.

8. Esimerkki LALR(1)-jäsentäjän muodostamisesta

Muodostetaan seuraavaksi toiminta- ja siirry-taulut kieliopille G_1 (säännön edessä suluissa säännön numero):

(1) S → bAb
(2) S → Ac
(3) S → ab
(4) A → a.

Laajennetaan alkuperäinen kielioppi G_1 kieliopiksi G_1' lisäämällä uusi alkusääntö:

(0) HYVÄKSY → S
(1) S → bAb
(2) S → Ac
(3) S → ab
(4) A → a.

Seuraavaksi muodostetaan alkiojoukot. Alkiojoukkoon I_0 sijoitetaan ensin alkio

([*HYVÄKSY* → *.S*], {$}).

Lasketaan tämän sulkeuma. Säännöstä S → bAb saadaan ydin [*S* → *.bAb*]. Kurkistusjoukon laskemiseksi määritellään kaikki perusmerkit, jotka ovat ensimmäisiä merkkejä merkkijonossa b a . (Merkkijono joka seuraa ytimessä sitä apumerkkiä, jonka perusteella sulkeuma lasketaan. Tässä tapauksessa tuo apumerkki on S ja b on tyhjä merkkijono. Merkki a on kurkistusalkio siinä alkiossa, josta sulkeuma lasketaan. Yleisesti näitä voi olla enemmän kuin yksi.) Koska b a = $ ja FIRST($) = $, saadaan alkio

([*S* → *.bAb*], {$}).

Säännöstä S→ Ac saadaan ydin [*S* → *.Ac*]. Koska b a = $ ja FIRST($) = $, saadaan alkio

([*S* → *.Ac*], {$}).

Säännöstä S→ ab saadaan ydin [*S* → *.ab*] . Koska b a = $ ja FIRST($) = $, saadaan alkio

([*S* → *.ab*], {$}).

Laskemalla sulkeuma alkiosta ([$S \rightarrow .Ac$], {\$}) saadaan uusi alkio. Säännöstä A$\rightarrow$ a saadaan ydin [$A \rightarrow .a$]. Koska *b* a = c\$ ja FIRST(c\$) = c, saadaan alkio

([$A \rightarrow .a$], {c}).

Nämä viisi alkiota muodostavat alkiojoukon I_0. Seuraavaksi lasketaan I_1 = **siirry**(I_0, S). Alkio ([$HYVÄKSY \rightarrow .S$], {\$}) I_0:ssa tuottaa alkion

([$HYVÄKSY \rightarrow S.$], {\$}).

Sulkeumaoperaatio ei tuota uusia alkioita. Tämä alkio muodostaa joukon I_1. Seuraavaksi lasketaan I_2 = **siirry**(I_0, A). Alkio ([$S \rightarrow .Ac$], {\$}) I_0:ssa tuottaa alkion

([$S \rightarrow A.c$], {\$}).

Sulkeumaoperaatio ei tuota uusia alkioita. Tämä alkio muodostaa joukon I_2. Seuraavaksi lasketaan I_3 = **siirry**(I_0, a). Alkiot ([$S \rightarrow .ab$], {\$}) ja ([$A \rightarrow .a$], {c}) I_0:ssa tuottavat alkiot

([$S \rightarrow a.b$], {\$}) ja
([$A \rightarrow a.$], {c}).

Sulkeumaoperaatio ei tuota uusia alkioita. Nämä alkiot muodostavat joukon I_3. Seuraavaksi lasketaan I_4 = **siirry**(I_0, b). Alkio ([$S \rightarrow .bAb$], {\$}) I_0:ssa tuottaa alkion

([$S \rightarrow b.Ab$], {\$}).

Tästä lasketaan sulkeuma. Säännöstä A$\rightarrow$ a saadaan ydin [$A \rightarrow .a$]. Koska *b* a = b\$ ja FIRST(b\$) = b, saadaan alkio

([$A \rightarrow .a$], {b}).

Nämä kaksi alkiota muodostavat joukon I_4. Seuraavaksi lasketaan I_5 = **siirry**(I_0, c). Tämä on tyhjä joukko, koska yhdessäkään ytimessä I_0:ssa ei ole merkkiä c heti pisteen oikealla puolella.

Lasketan loput alkiojoukot alkiojoukoista I_1, I_2, I_3, I_4:

I_5 = **siirry**(I_2, c) = ([$S \rightarrow Ac.$], {\$}). Sulkeumaoperaatio ei tuota lisää alkioita.

I_6 = **siirry**(I_3, b) = ([$S \rightarrow ab.$], {\$}). Sulkeumaoperaatio ei tuota lisää alkioita.

I_7 = **siirry**(I_4, A) = ([$S \rightarrow bA.b$], {$}). Sulkeumaoperaatio ei tuota lisää alkioita.

I_8 = **siirry**(I_4, a) = ([$A \rightarrow a.$], {b}). Sulkeumaoperaatio ei tuota lisää alkioita.

Lasketaan alkiojoukko vielä I_7 :stä:

I_9 = **siirry**(I_7, b) = ([$S \rightarrow bAb.$], {$}).

Uusia alkiojoukkoja ei voida tämän jälkeen tuottaa. Alkiojoukot ovat siis:

I_0:	[$HYVÄKSY \rightarrow .S$],	{$}
	[$S \rightarrow .bAb$],	{$}
	[$S \rightarrow .Ac$],	{$}
	[$S \rightarrow .ab$],	{$}
	[$A \rightarrow .a$],	{c}
I_1:	[$HYVÄKSY \rightarrow S.$],	{$}
I_2:	[$S \rightarrow A.c$],	{$}
I_3:	[$S \rightarrow a.b$],	{$}
	[$A \rightarrow a.$],	{c}
I_4:	[$S \rightarrow b.Ab$],	{$}
	[$A \rightarrow .a$],	{b}
I_5:	[$S \rightarrow Ac.$],	{$}
I_6:	[$S \rightarrow ab.$],	{$}
I_7:	[$S \rightarrow bA.b$],	{$}
I_8:	[$A \rightarrow a.$],	{b}
I_9:	[$S \rightarrow bAb.$],	{$}.

Jos alkijoukkoja muodostettaessa olisi muodostettu joukko, jonka ytimet olisivat olleet samat kuin aikaisemmin tuotetun joukon, olisi näistä joukoista muodostettu yksi joukko kurkistusjoukot yhdistämällä.

Muodostetaan seuraavaksi jäsennys-toiminnot alkiojoukkojen perusteella. Tilassa 0 alkio ([$S \rightarrow .bAb$], {$}) tuottaa toiminnon

if(syöte = b) **siirrä** 4,

koska **siirry**(I_0, b) = I_4. Alkio ([$S \rightarrow .ab$], {$}) tuottaa toiminnon

if(syöte = a) **siirrä** 3,

koska **siirry**(I_0, a) = I_3. Tila 1 tuottaa toiminnon

if(syöte = $) **hyväksy**,

koska tila 1 sisältää vain yhden valmiin alkion, jonka kurkistusmerkki on lopetusmerkki $. Tilassa 3 tuotetaan toiminto

if(syöte = c) **redusoi** 3,

koska tila sisältää valmiin alkion ([$A \rightarrow a.$], {c}) ja sääntö $A \rightarrow a$ on G':n sääntö numero 3.

Täydelliset jäsennys-toiminnot ovat:

0: **if** (syöte = a) **siirrä** 3
if(syöte = b) **siirrä** 4
virhe

1: **if** (syöte = $) **hyväksy**
virhe

2: **if** (syöte = c) **siirrä** 5
virhe

3: **if** (syöte = b) **siirrä** 6
if (syöte = c) **redusoi** 4
virhe

4: **if** (syöte = a) **siirrä** 8
virhe

5: **if** (syöte = $) **redusoi** 2
virhe

6: **if** (syöte = $) **redusoi** 3
virhe

7: **if** (syöte = b) **siirrä** 9
virhe

8: **if** (syöte = b) **redusoi** 4
virhe

9: **if** (syöte = $) **redusoi** 1
virhe.

Seuraavaksi muodostetaan siirry-toiminnot, joita käyteään uuden tilan määräämiseen redusoinnin jälkeen. Nämä muodostetaan siirry-graafista niistä apumerkeistä, joita on käytetty siirtymisessä. Huomataan, että merkillä S siirrytään tilasta 0 tilaan 1 ja, että merkillä A siirrytään tilasta 0 tilaan 3 sekä tilasta 4 tilaan 7. Tämän perusteella siirry-toiminnot ovat seuraavat:

S: **if**(tila = 0) **siirry** 1

A: **if**(tila = 0) **siirry** 3
if(tila = 4) **siirry** 7.

Näin on muodostettu LALR(1)-jäsentäjä kieliopille G_1'.

9. LR(1) vs. LALR(1)

LR(1)-jäsentäjän (kutsutaan usein *kanoniseksi LR(1)-jäsentäjäksi*) muodostaminen on pitkälti samanlaista kuin LALR(1)-jäsentäjän muodostaminen. Ainoa ero on, että LALR(1)-alkiojoukkoja muodostettaessa yhdistetään alkiojoukot, joilla on sama ydin. LR(1)-alkiojoukkoja muodostettaessa näin ei tehdä. Yhdistäminen voidaan tehdä joko alkiojoukkoja muodostettaessa tai ensin generoimalla LR(1)-alkiojoukot ja sitten yhdistellä joukot, joilla on sama ydin. Jälkimäinen vaihtoehto on kuitenkin tehoton.

Etuna alkiojoukkojen yhdistämisessä on jäsentäjän pienempi koko. Käytännössä ero on merkittävä. Haittapuolena on se, että yhdistämisen seurauksena LALR(1)-taulut voivat tuottaa konfliktitilanteita, joita LR(1)-taulut eivät tuottaisi.

Saman ytimen sisältävien alkiojoukkojen yhdistäminen ei voi aiheuttaa siirrä-redusoi -konfliktia, koska siirrä-toiminnot riippuvat vain ytimestä. Oletetaan esimerkiksi, että unionin tuottamalla tilalla on konflikti kurkistualkiossa a (jossain kohdassa a:n ollessa luettavana syötemerkkinä, päädytään konfliktitilanteeseen), koska on alkio $[A \rightarrow a.$, {a}], joka tuottaa redusoi-toiminnon $A \rightarrow a$, ja toinen alkio $[B \rightarrow b.ag$, {b}], joka tuottaa siirrä-toiminnon. Silloin siinä tilassa, josta unioni muodostettiin on alkio $[A \rightarrow a.$, {a}], ja koska tilojen ytimet ovat samat, myös alkio $[B \rightarrow b.ag$, {c}], jollekin c:lle. Silloin tällä tilalla on sama siirrä-redusoi -konflikti merkin a suhteen. Kielioppi ei siis ole LR(1).

Alkiojoukon yhdistäminen voi sen sijaan aiheuttaa redusoi-redusoi -konfliktin. Tarkastellaan kielioppia

S' → S
S → aAd | bBd | aBe | bAe
A → c
B → c.

LR(1)-jäsennys tuottaisi alkiot [$A \rightarrow c.$, d] ja [$B \rightarrow c.$, e] kelvolliselle alkuosalle ac sekä alkiot [$A \rightarrow c.$, e] ja [$A \rightarrow c.$, d] kelvolliselle alkuosalle bc. Kumpikaan näistä joukoista ei aiheuta konfliktia, ja niiden ytimet ovat samat. Sen sijaan niiden unioni

[$A \rightarrow c.$, {d,e}]
[$B \rightarrow c.$, {d,e}]

tuottaa redusoi-redusoi -konfliktin, koska syötemerkkien d ja e kohdalla on mahdollista redusoida säännön $A \rightarrow c$ tai säännön $B \rightarrow c$ mukaan.

Joillekin kieliopeille LALR(1)-jäsennystaulut tuottavat siis redusoi-redusoi –konflikteja, joita LR(1)-taulut eivät tuottaisi. Käytännössä tällä ei kuitenkaan ole suurta merkitystä. Sen sijaan käytännössä on suuri merkitys sillä, että LALR(1)-taulujen esittämiseen tarvitaan vähemmän muistia, koska tiloja on vähemmän.

10. Moniselitteisten kielioppien jäsentäminen

Yksikään moniselitteinen kielioppi ei ole LR-kielioppi. Moniselitteisellä kieliopilla voidaan kuitenkin jossain tapauksissa määritellä kieli helpommin kuin vastaavalla yksiselitteisellä kieliopilla. Tällöin yleensä moniselitteisessä kieliopissa on vähemmän sääntöjä kuin vastaavan kielen tuottavassa yksiselitteisessä kieliopissa. Tämän seurauksena moniselitteisestä kieliopista muodostettu jäsentäjä sisältää vähemmän tiloja ja on siis kooltaan pienempi. LR-jäsentäjän muodostaminen moniselitteiselle kieliopille tuottaa aina konflikteja toiminta-tauluun. LR-jäsentäjissä voidaan kuitenkin käyttää moniselitteisiä kielioppeja, jos konfliktitilanteet osataan ratkaista.

Monet ohjelmointikielet sisältävät tyyppiä

S → if E then S | if E then S else S

olevia sääntöjä. Lause *if a then if b then c else d* voidaan tulkita kahdella tavalla: *if a then (if b then c else d)* tai *if a then (if b then c) else d*. Useimmissa ohjelmointikielissä käytetään ensimmäistä tulkintaa, eli jokainen uusi else liitetään siitä vasemmalta lähimpänä olevaan then-osaan. LR-jäsentäjässä tälläiset säännöt tuottavat siirrä-redusoi -konfliktin, koska se sisältää alkiot:

[$S \rightarrow$ if E then S., {else,$}]

[$S \rightarrow$ if E then S. else S, {else,$}].

Jos kielioppi sisältää vain tämäntyyppisiä moniselitteisyyksiä, voidaan konfliktitilanteet ratkaista käyttämällä aina siirrä-toimintoa.

Kieliopeissa, jotka tuottavat monimutkaisempia siirrä-redusoi -konflikteja, voidaan konfliktit ratkaista määräämällä parille (s_1, s_2) (s_1 ja s_2 voivat olla perusmerkkejä tai kieliopin sääntöjä) suhteet seuraavaan tapaan:

1. s_1 on etusijalla s_2:een nähden
2. s_2 on etusijalla s_1:een nähden
3. s_1 ja s_2 ovat samanarvoisia
4. s_1:n ja s_2:n välille ei voida asettaa mitään kohdan 1-3 välistä relaatiota.

LR-jäsentämisessä konfliktit voivat olla perusmerkin siirrä-toiminnon ja säännön mukaan redusoinnin välillä tai vain sääntöjen mukaan redusoinnin välillä. Molemmissa tapauksissa voidaan löytää perusmerkki tai sääntö, joka voidaan asettaa etusijalle tai päätellä, ettei pari (s_1, s_2) ole vertailukelpoinen. Ensimmäisessä tapauksessa ratkaistaan konflikti etusijalla olevan toiminnon hyväksi. Jälkimmäinen tapaus aiheuttaa virheen.

Konfliktitilanteet, jossa s_1 ja s_2 ovat samanarvoisia, voidaan jossain tapauksissa ratkaista ottamalla huomioon *assosiatiivisuus*. Vasemmanpuoleinen assosiatiivisuus asettaa redusoinnin siirrä-toiminnon edelle ja oikeanpuoleinen assosiatiivisuus asettaa siirrä-toiminnon redusoinnin edelle. Määrittelemätön assosiatiivisuus etusijaltaan samanarvoisilla pareilla tarkoittaa sitä, ettei konfliktia voida ratkaista.

Tyypillisiä esimerkkejä etusijojen ja assosiatiivisuuksien käytöstä ovat kieliopit, jotka tuottvat aritmeettisia lausekkeita. Tälläisissä tapauksissa moniselitteiset kieliopit tuottavat monesti huomattavasti tehokkaamman jäsentäjän kuin mitä saman kielen tuottava yksikäsitteinen kielioppi tuottaisi. Tarkastellaan seuraavaa moniselitteistä kielioppia:

E $\rightarrow$ E + E | E * E | (E) | **id**.

Määritellään, että operaattori '*' on etusijalla operaattoriin '+' nähden ja, että molemmat operaattorit ovat vasemmapuoleisesti assosiatiivisia. Tarkastellaan jäsentäjän tuottamaa siirrä-redusoi -konfliktia

[E $\rightarrow$ E * E., {+,$}]
[E $\rightarrow$ E. + E, {...}].

Tässä tapauksessa voidaan suorittaa siirrä-toiminto merkin '+' mukaan tai redusoida säännön
E $\rightarrow$ E * E mukaan. Operaattorit ovat vasemmanpuoleisesti assosiatiivisia, joten säännön prioriteetti määräytyy merkin '*' mukaan. Sääntö asetetaan etusijalle, koska

operaattorilla ’*’ on etusija operaattoriin ’+’ nähden. Tässä tapauksessa päädytään siis redusointiin.

11. Virheistä toipuminen

LR-jäsentämisessä virheet löytyvät toiminta-taulua käytettäessä, ei koskaan siirry-taulusta. LR-jäsentäjä ilmoittaa virheestä heti, kun siihen mennessä luetusta syötteestä ei voida muodostaa kelvollista alkuosaa. Kanoninen LR-jäsentäjä ei tee ennen virheestä ilmoittamista yhtään redusointia. LALR-jäsentäjä voi suorittaa redusointeja, mutta se ei koskaan suorita siirrä-toimintoa virheelliselle syötemerkille.

Virheen löytämisen jälkeen virheestä pyritään toipumaan, jotta jäsentämistä pystyttäisiin jatkamaan virhekohdan jälkeen. Tarkoituksena on ilmoittaa kaikista ohjelmassa esiintyvistä virheistä, eikä vain ensimmäisestä mahdollisesta. LR-jäsentäjään voidaan rakentaa laaja valikoima virheestätoipumismenetelmiä.

11.1 Paikalliset ja globaalit menetelmät

Paikallinen virheestätoipuminen toimii mukauttamalla jäsennyspinoa kohdassa, jossa virhe havaitaan. Tämä tehdään siten, että jäsentämistä voidaan jatkaa.

Yksi vaihtoehto virheestä toipumiseen paikallisella menetelmällä on *paniikkitila* (panic-mode). Tämän menetelmän tarkoituksena on eristää syntaktisen virheen sisältävä osa. Jäsennennyksen jossain vaiheessa jäsentäjä päättelee, että tietty merkkijono a , joka on johdettavissa jostain apumerkistä A, sisältää virheen. Tällöin osa tästä merkkijonosta on jo käsitelty, joten siihen liittyviä tiloja on pinon päällä. Jäsentäjä poistaa pinosta tiloja kunnes löydetään tila s, josta on siirtymä apumerkin A perusteella. Loput a :n merkeistä ovat vielä syötepuskurissa. Jäsentäjä pyrkii ohittamaan nämä merkit ja etsii syötemerkkiä a, joka kieliopin sääntöjen mukaan voi seurata A:ta. Tämän jälkeen jäsentäjä laittaa pinoon tilan **siirry**(a, S), ja jatkaa toimintaansa. Näin ollen jäsentäjä ”teeskentelee”, että apumerkistä A johdettavissa oleva merkkijono on löydetty, ja jatkaa toimintaansa normaalisti. Tälläinen menettely on paikallinen, koska pinoa ja syötettä muokataan siinä kohdassa, jossa virhe havaitaan.

On mahdollista, että A:ksi on useita vaihtoehtoja. Tavallisesti nämä vastaavat merkittäviä ohjelman osia, kuten lausekkeita tai lauseita. Jos A vastaa esimerkiksi lausetta, a voi olla puolipiste (esim. C, Java) tai **end** (Pascal).

Globaalissa virheekorjauksessa on ideana lisätä tai poistaa merkkejä syötteestä ennen kohtaa, jossa virhe havaitaan. Tarkastellaan esimerkkinä Java-kielistä ilmaisua

```
int [] a = {1.5, 2}; .
```

Paikallinen tekniikka havaitsee virheen kurkistusalkion ollessa . (piste). Tässä tapauksessa ohjelmoijan virhe on kuitenkin ollut käyttää tyyppiä *int* tyypin *double* sijaan, mutta virhe havaitaan vasta kuuden merkin päästä.

Globaali virheenkorjaus löytää pienimmän joukon lisäyksiä ja poistoja, jotka muuttavat syötemerkkijonon syntaktisesti oikeaksi merkkijonoksi, vaikka lisäykset ja poistot tapahtuvat muualla kuin missä virhe havaittiin. Esimerkissä kieliopin merkki *int* korvattaisiin merkillä *double.*

Burken-Fisherin -virheenkorjaus kokeilee jokaista mahdollista yhden merkin lisäystä, poistoa tai korvausta jokaisessa kohdassa, joka esiintyy k:ssa merkissä ennen sitä kohtaa, jossa virhe havaittiin. Jos kieliopissa on n eri merkkiä, mahdollisia lisäyksiä, poistoja ja korvauksia on k + kn + kn kappaletta. Näin monen korjauksen käyttö ei ole erityisen tehotonta, ottaen huomioon, että tämä tapahtuu vain syntaksivirheiden kohdalla.

Algoritmin on voitava peruuttaa k merkkiä ja tehdä jäsentäminen uudelleen. Siksi pitää muista, miltä pino näytti k:ta alkiota sitten. Algoritmi käyttää tätä tarkoitusta varten kahta pinoa, *nykyinen* ja *vanha*. Algoritmi käytää myös k:n merkin jonoa. Aina kun uusi merkki siirretään pinoon, se laitetaan pinoon *uusi* sekä jonon perään. Jonon ensimmäinen alkio poistetaan ja laitetaan pinoon *vanha*. Molemmissa pinoissa jokaista pinosta poistoa kohti suoritetaan sopivat redusoinnit.

Oletetaan, että syntaksivirhe huomataan nykyisessä syötemerkissä. Jokaista mahdollista lisäystä, poistoa tai korvausta kohti missä tahansa kohtaa jonossa algoritmi suorittaa vaihdon jonon sisällä, ja sitten yrittää suorittaa jäsennyksen pinosta *vanha* lähtien. Jonon muokkaamisen onnistuminen riippuu siitä, kuinka monta merkkiä nykyisestä syötemerkistä eteenpäin voidaan jäsentää. Jos kolme tai neljä merkkiä eteenpäin onnistutaan jäsentämään ilman virhettä, voidaan korjausta pitää onnistuneena.

11.2 Ilmaisutason toipuminen

Ilmaisutason (phrase-level) toipumisessa tarkastellaan erikseen jokaista virhekohtaa toiminta-taulussa (kaikki ne kohdat toiminta-taulussa, jotka eivät ole siirrä- tai hyväksy-toimintoja tai redusointeja). Ohjelmointikielen ominaisuuksien perusteella päätellään todennäköisin siihen liittyvä ohjelmointivirhe. Tämän perusteella voidaan muodostaa sopivia toipumistoimenpiteitä. Toiminnat voivat olla symbolien lisäämistä tai poistamista joko pinosta, tai syötteestä (tai molemmista). Toiminnat voivat olla myös syötemerkkien muuttamista tai siirtämistä. Päätökset toiminnoista on tehtävä siten, että LR-jäsentäjä ei voi mennä päättymättömään silmukkaan. Tämän varmistamiseksi ainakin yksi syötemerkki on poistettava tai siirrettävä pinoon, tai on varmistuttava siitä, että pinon koko pienenee syötteen loppuun mennessä. Päätökset sopivista toiminnoista virheitä kohdattaessa riippuu täysin ko. kieliopista.

Virheistätoipumismenelmissä voidaan muokata kielioppia, jäsennystauluja, tai muokata pelkästään jäsennystä suorittavaa algoritmia. Paniikkitila-menetelmä on esimerkki

kielioppia muokkavasta menetelmästä, koska kielioppiin pitää lisätä virhe-symboli toipumisprosessia ohjaamaan. Ilmaisutason toipuminen on esimerkki jäsennystauluja muokkaavasta menetelmästä. Burken-Fisherin -menetelmä on taas esimerkki pelkästään jäsennystä suorittavan algoritmin muokkauksesta.

12. Yhteenveto

LR-jäsentäjä on jäsentäjätyyppi kontekstittomille kieliopeille, jota käytetään ohjelmointikielten kääntäjissä. LR-jäsentäjät lukevat syötettään vasemmalta oikealle tuottaen oikeanpuoleisimman jäsennyksen. Termissä LR(k) k viittaa siihen, kuinka monta kurkitussymbolia käytetään, eli kuinka monen vielä käsittelemättömän syötemerkin perusteella toimintapäätöksiä voidaan tehdä jäsennyksen aikana. Yleensä k on 1. Kontekstitonta kielioppia sanotaan LR(k)-kieliopiksi, jos sille voidaan muodostaa LR(k)-jäsentäjä, joka tunnistaa kaikki kieliopin tuottamat lauseet.

Yleensä LR-jäsentäjät tuotetaan mekaanisesti. Se, kuinka jäsennystaulut näille jäsentäjille muodostetaan, saa aikaan erilaisia LR-jäsentäjiä. Tässä tutkimuksessa on kuvattu kaksi LR-jäsentäjätyyppiä: LALR-jäsentäjät ja kanoniset LR-jäsentäjät. Kanoniset LR-jäsentäjät soveltuvat suuremmalle kielioppijoukolle kuin LALR-jäsentäjät. LALR-jäsentäjät ovat kuitenkin käytännössä tehokkaampia.

LR-jäsentäjillä voidaan käsitellä moniselitteisiä kielioppeja, jos näiden aiheuttamat konfliktitilanteet osataan ratkaista sopivasti. Etuna moniselitteisissä kieliopeista on se, että monesti niillä saadaan määriteltyä kieli helpommin kuin vastaavalla yksiselitteisellä kieliopilla.

Virheistätoipuminen on tärkeä osa jäsennystä, jotta koko ohjelma saadaan jäsennettyä kerrallaan ja kaikki ohjelmassa esiintyvät virheet voidaan löytää. LR-jäsentäjiin voidaan soveltaa erilaisia virheestätoipumismenetelmiä.

Suurin osa ohjelmointikielistä on jäsennettävissä LR-jäsentäjillä (ja LALR-jäsentäjillä). LR-jäsentäminen on tärkeä ja käytännöllinen työkalu kääntäjien suunnittelussa.

Lähdeluettelo

[Aho and Johnson, 1974] A.V. Aho and S.C. Johnson, LR parsing. *J. ACM* **6**, 2 (June 1974), 99-124.

[Aho et al., 1985] Alfred V. Aho, Ravi Sethi and Jeffrey D. Ullman, *Compilers – Principles, Techniques and Tools*. Addison-Wesley, 1985.

[Appel, 1998] Andrew W. Appel, *Modern Compiler Implementation in Java.* Cambridge University Press, 1998.

[Barrett and Couch, 1979] William A. Barrett and John D. Couch, *Compiler Construction: Theory and Practice*. Science Research Associates,1979.

[Hopcroft and Ullman, 1979] John E. Hopcroft and Jeffrey D. Ullman, *Introduction to Automata Theory, Languages and Computation.* Addison-Wesley, 1979.

GSM-verkon tietoturvan puutteet

Tommi Rautiainen

Tiivistelmä.

GSM-verkko on maailman ensimmäinen digitaalinen matkapuhelinverkko. Se otettiin käyttöön vuonna 1991, ja on melko muuttumattomana käytössä tälläkin hetkellä. GSM-verkon suunnittelussa tehtiin kuitenkin joitain pahoja virheitä, jotka ovat muutamia vuosia sitten tulleet ilmi ja levinneet yleiseen tietoisuuteen. Tässä tutkielmassa esittelen GSM-verkon puutteista johtuvia tietoturvauhkia.

Avainsanat ja -sanonnat: GSM, tietoturva
CR-luokat: C.2.1, C.2.2, E.3

1. Johdanto

Kirjainyhdistelmänä GSM on nykyisin tunnettu ympäri maailman. GSM–spesifikaation sisällöstä on kuitenkin suurin osa ihmisistä täysin tietämättömiä. Kun tarkennetaan tutkimusongelmaa vielä GSM-verkon tietoturvan puutteisiin, saadaan asiasta tietävien ihmisten joukko rajattua huomattavan pieneksi.

GSM-verkon tietoturva koskee kuitenkin huomattavan suurta joukkoa länsimaiden ihmisistä, vaikkei sitä jokapäiväisessä elämässä juuri huomaakaan. GSM on yhä laajimmin käytetty matkapuhelinjärjestelmä koko maailmassa. Käyttäjiä GSM-järjestelmällä on noin 230 miljoonaa.

Ensimmäisenä digitaalisena matkapuhelinstandardina GSM oli uranuurtaja langattomassa viestinnässä. Sen kehitystyö ei suinkaan ollut lyhyt, vaan kesti peräti 8 vuotta, aina vuodesta 1982 vuoteen 1990, jolloin ensimmäiset spesifikaatiot jäädytettiin. Varsinaiseen käyttöön GSM-verkko tuli vuonna 1991. Aluksi GSM perustui 900 MHz verkkoon, mutta jo vuonna 1993 otettiin käyttöön ensimmäinen 1800 MHz verkko. Luonnollisesti matkapuhelinvalmistajilla, kuten Nokialla, meni oma aikansa ennen kuin 1800 MHz verkkoa käyttävät päätelaitteet tulivat tavallisen kuluttajan saataville. Nokian kohdalla 1800 MHz verkko esiteltiin 6100-sarjan mallistossa ensin vuoden 1998 tienoilla ja edullisemmassa 3200-sarjan mallistossa se esiteltiin vuonna 1999. [Vesanen, 2003]

GSM-verkon tietoturvaan 1800 MHz verkon esittelyllä ei ollut suurta merkitystä verrattuna 900 MHz verkkoon, mutta sillä että GSM–spesifikaatiot kehitettiin salassa, ja että niitä kehitettäessä rikottiin algoritmien suhteen tiettyjä kryptografisia periaatteita, on ollut oma merkittävä vaikutuksensa GSM–verkon turvallisuuteen. [Vesanen, 2003] Kryptografinen turva on vain pieni osa 130-osaisesta, 6.000-sivuisesta GSM-protokollasta [Isaac, 1998].

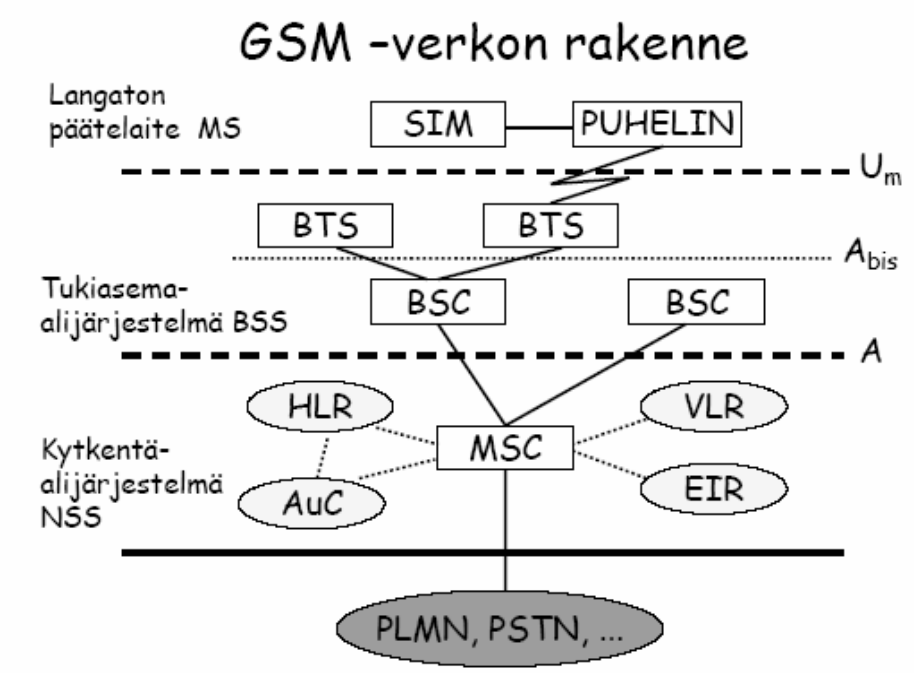
Tämän tutkimuksen lähtökohtana on henkilökohtainen kiinnostukseni GSM-verkon tietoturvaa kohtaan ja GSM-verkon tietoturvauhkien kartoittaminen. Aluksi pyrin selvittämään GSM-verkon taustaa eri lähteiden avulla, ja niitä GSM-verkon puutteita, joista uhkakuvat syntyvät.

2. GSM-verkon rakenne ja algoritmit

GSM–spesifikaatioiden ongelmana on, että ne eivät aikoinaan saaneet tieteellisen prosessin mukaista julkista arviointia, ja niiden vahvuus perustuu osaltaan algoritmien itsensä salaisuuteen. Täten GSM–spesifikaatioiden kehityksessä on rikottu ns. Kerckhoffin periaatetta, jonka mukaan salausalgoritmin vahvuuden tulee riippua ainoastaan avaimesta, eikä algoritmin itsensä salaisuudesta. Nyt spesifikaatioiden ja salausalgoritmien vuodettua julkisuuteen ovat kehityksessä tehdyt virheet kostautuneet, ja erityisesti algoritmit ovat joutuneet julkisen arvostelun kohteeksi. [Vesanen, 2003]

2.1. GSM–verkon rakenne

GSM-verkko voidaan jakaa kolmeen suurempaan kokonaisuuteen (ks. kuva 1) eli langattomaan päätelaitteeseen (Mobile Station, MS), tukiasema-alijärjestelmään (Base Station Subsystem, BSS) ja kytkentäalijärjestelmään (NSS). [Vesanen, 2003]



Kuva 1. GSM-verkon rakenne [Vesanen, 2003].

Langaton päätelaite on verkon käyttäjän hallussa oleva laite, jonka tietoturvanäkökulmasta merkittävimmät ominaisuudet ovat puhelimen tunniste eli IMEI-koodi (International Mobile Equipment Identity) ja SIM (Subscriber Indentity Module) eli palvelun käyttäjän tunnistusmoduuli. Suurin osa langattoman päätelaitteen salauksellisista ja käyttäjän tunnistamiseen liittyvistä ominaisuuksista sijaitsee SIM-moduulissa. [Vesanen, 2003]

Tukiasema-alijärjestelmän pääasiallinen tehtävä on kontrolloida radioyhteyttä langattomaan päätelaitteeseen. Tukiasema-alijärjestelmä koostuu tukiasemista (Base Transceiver Station, BTS) ja tukiasemakontrollereista (Base Station Controller, BSC). [Vesanen, 2003]

Kytkentäalijärjestelmä puolestaan koostuu matkapuhelinkeskuksesta (Mobile Services Switching Center, MSC), tekstiviestikeskuksesta (Short Message Service Center, SMSC), rekistereistä ja toiminta-alijärjestelmästä (Operation Subsystem, OSS) [Vesanen, 2003]

2.2. Algoritmit

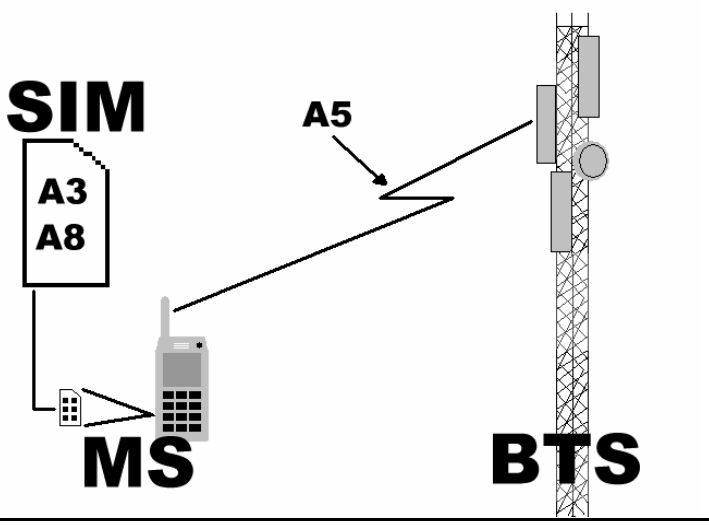
GSM-verkon eri osien välinen tietoliikenne on salattu siihen tarkoitettujen salausalgoritmien avulla. GSM-verkossa salaukseen käytetyt algoritmit ovat A3, A8 ja A5 (ks. kuva 2).

A3 on päätelaitteen autentikaatioalgoritmi. Se sijaitsee puhelimen SIM-kortilla. A3-algoritmin toiminta perustuu siihen, että se saa parametrinaan 128-bittisen satunnaisluvun verkosta ja 128–bittisen Ki-avaimen SIM-kortilta. Satunnaisluvusta ja Ki-avaimesta A3 laskee 32-bittisen sormenjäljen SRES. A3-algoritmin laskema sormenjälki lähetetään verkolle, jonka jälkeen autentikaatio on suoritettu. A3-algoritmi perustuu COMP128-algoritmiin, joka toteuttaa yksisuuntaisen funktion. COMP128-algoritmi on kuitenkin rikki sillä tavoin, että se luovuttaa tietoa, jos sille esitetään oikeanmuotoisia kyselyitä. COMP128-algoritmin ongelmat mahdollistavat täten A3-algoritmin murtamisen. [Pesonen, 1999]

A8 on ilmatien salausavaimen luontialgoritmi. A8-algoritmia käytetään Kc-istuntoavaimen luomiseksi autentikaation jälkeen. A8-algoritmin toiminta perustuu siihen, että se saa parametrinaan 128-bittisen Ki-avaimen ja 128-bittisen satunnaisluvun. Näistä A8-algoritmi sitten laskee 64-bittisen Kc-istuntoavaimen, joka on käytössä kunnes matkapuhelinkeskus päättää uudestaan autentikoida päätelaitteen. A8-algoritmi perustuu COMP128-algoritmiin ja omaa sen vuoksi samat heikkoudet kuin A3-algoritmikin. [Vesanen, 2003]

A5-algoritmi on GSM-verkon ilmatien jonosalausalgoritmi, jolla salataan ääni ja dataliikennettä puheluyhteyden aikana. A5-algoritmista on olemassa kaksi versiota, vahvempi A5/1 ja heikompi A5/2. A5-algoritmi salaa keskustelut istuntoavaimen (Kc) avulla. Salaus tapahtuu XOR-salaamalla istuntoavaimen ja frame-laskurin (Fn) avulla muodostettu 228-bittinen satunnaisarvo. [Biryukov et al. 2000] Mike Roen [Roe, 2002] sivuilla on epätäydellinen esittely A5-salausalgoritmia koskien. Myös Jovan Golic [Golic, 1998] on tutkinut A5 algoritmia.

Alun perin turvallisiksi aiotut algoritmit ovat siis ajan kuluessa osoittautuneet jokseenkin turvattomiksi. Tähän on osaltaan vaikuttanut myös tietokoneiden laskentatehon luonnollinen kasvu. Vuoden 1991 tietokonelaitteistolla olisi algoritmien murtaminen ollut melkein mahdotonta. Isaac-ryhmän mukaan verkkoon on joutunut myös GSM-dokumentti joka määrittelee COMP128-algoritmia [Young, 1998]. Lisäksi Internetistä on saatavilla salausalgoritmien C-kielinen esittely [Briceno et al. 1998].



Kuva 2. GSM-verkossa käytettävät algoritmit.

3. GSM-verkon puutteet

Vesasen [2003] mukaan GSM-verkon tietoturvalla on kuusi merkittävää puutetta. Nämä puutteet ovat, että verkko ei todenna itseään, käytetyillä algoritmeilla on heikkouksia, datan eheyttä ei tarkisteta, IMEI-koodissa on ongelmia, autentikaatiossa on ongelmia ja verkon näkyvyys on puutteellinen.

Se, että GSM-verkko ei todenna itseään, mahdollistaa tukiaseman väärentämisen. Algoritmeista puutteellisimpia ovat A5 ja COMP128, ja näiden algoritmien puutteet helpottavat verkkoon hyökkäämistä. [Vesanen, 2003]

Koska GSM-verkossa ei myöskään tarkisteta datan eheyttä, mahdollistuu datan muunteleminen, eikä tätä pystytä havaitsemaan tarpeeksi nopeasti. IMEI-koodin ongelmat taas johtuvat suurimmaksi osaksi siitä että se välitetään aina salaamatta. [Vesanen, 2003]

Autentikaation suurin ongelma on, että autentikaatioon käytettävä data lähetetään aina salaamattomana verkkojen sisällä ja verkkojen välillä. Tämä on merkittävä puute, koska autentikaatioon käytettävä data sisältää myös ilmatien salausavaimen. [Vesanen, 2003]

GSM-verkon näkyvyydellä tarkoitetaan sitä, miten GSM-verkko näkyy käyttäjälle ja kotiverkkoon, eli mitä tietoa käyttäjä ja kotiverkko saavat päätelaitetta palvelevasta verkosta. GSM-verkossa käyttäjä ei pysty näkemään, käytetäänkö verkossa salausta. Palveleva GSM-verkko ei myöskään anna vahvistusta kotiverkkoon siitä, käyttääkö se autentikaatioparametreja oikein päätelait-

teen vaeltaessa verkossa. Nämä näkyvyyden puutteet aiheuttavat sen, että on hankalaa havaita, onko verkossa jotain vialla. [Vesanen, 2003]

Mainitut GSM-verkon puutteet ovat suurimpia syitä GSM-verkon tietoturvauhille, ja ilman näitä puutteita GSM-verkko olisikin huomattavasti turvallisempi langaton tietoliikenneratkaisu.

4. GSM-verkon tietoturvauhat

Seuraavassa esittelen erilaisia GSM-verkkoon kohdistuvia uhkia, jotka juontavat juurensa GSM-verkon puutteista. Nämä uhat ovat SIM-kortin kloonaus, ilmatien salauksen murtaminen ja tukiaseman väärentäminen, hyökkäys autentikointikeskusta vastaan, palveluntarjoajan sisäiset uhat, viestintäverkkoon murtautuminen, sosiaalinen hakkerointi, passiiviset uhat, raa'an voiman hyökkäys A5-algoritmia vastaan, ja hajota ja hallitse A5-hyökkäys. Jokaisessa uhkatapauksessa GSM-verkon tietoturvallisuus vaarantuu, tietoa katoaa tai joutuu vääriin käsiin. Joistakin seuraavaksi esiteltävistä uhkista on tehty käytännön tutkimuksia ja jotkin uhat ovat vasta teorian tasolla, mutta kaikki uhat ovat mahdollisia tiettyjen olosuhteiden vallitessa, joten mitään näistä uhkista ei tulisi aliarvioida.

4.1. SIM-kortin kloonaus

SIM-kortin kloonaus on eräs GSM-verkon todellisimmista uhista. Jos hyökkääjä onnistuu saamaan uhrin SIM-kortin haltuunsa, on mahdollista saada COMP128-algoritmin heikkoudesta johtuen haaste-vastaus -hyökkäyksen avulla SIM-kortilta käyttäjän salainen Ki-avain. [Goldberg, Briceno, 1998]

Käytännössä hyökkäys tapahtuu pommittamalla A8-algoritmia erilaisilla satunnaislukusyötteillä ja tutkimalla, miten syötteiden vaihtuminen vaikuttaa algoritmin antamaan SRES-vastaukseen ja algoritmin laskemaan Kc-istuntoavaimeen. Hyökkäys on mahdollista toteuttaa esimerkiksi laittamalla haltuun saatu SIM-kortti tietokoneeseen kytkettyyn kortinlukijaan.

Tällaista hyökkäysmenetelmää on myös kerran julkisesti testattu käytännössä. Tämä tapahtui vuonna 1998 Ian Goldbergin ja Marc Bricenon toimesta. Hyökkäyksessä käytettiin älykortinlukijaa, joka pystyi suorittamaan SIM-kortille 6,25 kyselyä sekunnissa. Hyökkäyksen onnistumiseksi vaadittiin 150.000 kyselyn suorittaminen, ja tästä johtuen hyökkäykseen kului aikaa kahdeksan tuntia, jonka lisäksi meni hieman lisää aikaa vastausten analysointiin. Analysoinnin seurauksena Goldberg ja Briceno onnistuivat päättelemään SIM-kortin salaisen Ki-istuntoavaimen [Goldberg, Briceno, 1998].

Kun Ki-avain on saatu selville, on mahdollista palauttaa kortti huomaamattomasti takaisin käyttäjälleen, tehdä kortista klooniversio ja aloittaa puheluiden puhuminen käyttäjän laskuun tai puheluiden salakuuntelu.

Puheluiden puhuminen kortin käyttäjän laskuun ei kuitenkaan ole aivan mutkatonta. Verkon huomatessa, että sitä käyttää kaksi identtistä korttia samaan aikaan sulkee se molemmat kortit. Käytännössä hyökkääjän pitäisi siis tietää, milloin kortin oikea käyttäjä ei ole verkossa. Luultavimmin kortin omistaja huomaa myös pian kasvaneet puhelinlaskut, ja kun syytä asiaan aletaan selvittää, on liittymän sulkeminen luultavasti melkein ensimmäisiä toimenpiteitä. Periaatteessahan puhelinyhtiön on mahdollista myös jäljittää, minkä tukiaseman alueella mitäkin SIM-korttia käytetään, ja jos kyseessä on harvaan asuttu alue, saattaa hyökkääjä jäädä helpostikin kiinni. [Goldberg, Briceno, 1998]

Epärehelliselle matkapuhelinkauppiaalle SIM-kortin kloonaus olisi huomattavasti pienempi haaste kuin muunkaltaiselle hyökkääjälle. Tällaisen matkapuhelinkauppiaan onkin mahdollista kloonata aina avaimet kaikilta asiakkaille välittämiltään SIM-korteilta etukäteen. Näin matkapuhelinkauppias pystyy helposti saamaan suuren määrän erilaisten asiakkaiden liittymien avaimia haltuunsa ja käyttämään niitä parhaaksi katsomallaan tavalla. [Goldberg, Briceno, 1998]

4.2. Ilmatien salauksen murtaminen ja tukiaseman väärentäminen

Langattoman GSM-päätelaitteen ilmatien salaus on teoriassa mahdollista murtaa väärentämällä tukiasema ja lähettämällä päätelaitteelle autentikointipyyntöjä väärennetystä tukiasemasta. Tämä perustuu siihen, että GSM-päätelaite autentikoi itsensä tukiasemalle, mutta tukiasemaa ei koskaan autentikoida päätelaitteelle. Tämän kaltainen hyökkäys vaatii, että päätelaitteen on oltava tukiaseman alueella useita tunteja. Päätelaitteen ei tarvitse olla tukiaseman alueella yhtämittaisesti, vaan hyökkäys voidaan suorittaa lyhyemmissä jaksoissa. Tämä hyökkäys toimii, kuten SIM-kortin kloonaushyökkäyskin, niin, että pommitetaan A8-algoritmia erilaisilla satunnaislukusyötteillä, ja tämän kautta saada selville SIM-kortin salainen Ki-istuntoavain. Tämän jälkeen on tarkoitus istuntoavainta käyttäen tehdä hyökkäyksen kohteen SIM-kortista klooni. Päätelaitteen käyttäjä ei pysty periaatteessa havaitsemaan hyökkäystä lainkaan. [Pesonen, 1999]

Tällaista hyökkäystä ei koskaan ole julkisesti kokeiltu käytännössä, mutta GSM-eksperttien mukaan se on mahdollinen ja todellinen uhkakuva. GSM-eksperttien arvio on, että GSM-tukiasema tulisi maksamaan vähintään 10.000 dollaria, joten periaatteessa sellaisen rakentaminen ei ole lainkaan mahdotonta.

[Brewer et al. 1998]

4.3. Hyökkäys autentikointikeskusta vastaan

Autentikointikeskus (AuC) (ks. kuva 1.) pitää yllä turvallisuustarkoituksiin tarkoitettua rekisteriä, joka huolehtii autentikaatioon ja salaustoimintoihin tarvittavista parametreista. Autentikointikeskusta vastaan tehty hyökkäys olisi teoriassa melko samanlainen kuin tukiaseman väärentämishyökkäys. Autentikointikeskusta vastaan tehdyssä hyökkäyksessä pyritään hankkimaan Ki-avain tekemällä keskukselle kyselyitä matkapuhelinverkosta. Autentikointikeskuksen turvallisuus vaikuttaa pitkälti siihen, onko tämän kaltainen hyökkäys käytännössä mahdollinen. Kyseisen kaltaista hyökkäystä ei ole koskaan testattu julkisesti, joten arviot sen toimivuudesta ovat teoreettisella tasolla. [Pesonen, 1999]

4.4. Palveluntarjoajan sisäiset uhat

Palvelun tarjoajan henkilöstöllä on usein melko rajoittamaton pääsy GSM-verkon eri osiin. Jos jonkin henkilöstön jäsenen asenne työnantajaa tai asiakasta kohtaan muuttuu negatiiviseksi, tai henkilöstön jäsen on muuten epärehellinen, on tällaisella henkilöllä mahdollisuus aiheuttaa haittaa sekä asiakkaille että työnantajalle. Tällaisia mahdollisia haittoja ovat muun muassa asiakastietojen leviäminen, puheluiden salakuunteleminen ja haitallisten henkilöiden päästäminen verkkoon.

4.5. Viestintäverkkoon murtautuminen

GSM-verkossa liikenne on salattu Pesosen [1999] mukaan vain MS:n ja BTS:n välillä (ks. kuva 1). BTS:n jälkeen liikenne kulkee operaattorin verkossa täysin selväkielisenä, ja SS7-viestintäverkko, joka on käytössä GSM-operaattorien verkossa, on täysin turvaton jos hakkeri pääsee siihen käsiksi.

Tämä seikka mahdollistaa muutaman erilaisen hyökkäystavan käyttämisen, ja jos hakkeri onnistuu pääsemään operaattorin viestintäverkkoon, saa hän käsiinsä satunnaisluvun, SRES-vastauksen ja Kc-avaimen. Tämän lisäksi hakkeri pystyy salakuuntelemaan kaikkia verkon läpi kulkevia puheluita. [Pesonen, 1999]

Eräs mahdollinen paikka hyökätä operaattorin viestintäverkkoon on HLR. Jos hakkeri onnistuu pääsemään HLR:n sisälle, saa hän käsiinsä kaikkien verkossa olevien asiakkaiden Ki-avaimet. Useimmiten HLR on kuitenkin hieman muuta verkkoa paremmin suojattu, eikä siihen pääse niin helposti fyysisesti käsiksi kuin

muuhun verkkoon. Silti HLR:ään murtautumisen edut ovat ilmeiset ja hakkerista riippuen fyysinenkään murtautuminen ei välttämättä ole suuri ongelma. [Pesonen, 1999]

BTS:n ja BTC:n välinen linkki saattaa myös olla helppo paikka murtautua viestintäverkkoon. BTS:n ja BTC:n välinen yhteys on yleensä hoidettu joko kaapelilla, mikroaalloilla tai satelliittilinkillä. Kaikki nämä linkitykset on mahdollista murtaa oikeilla välineillä. BTS:n ja BTC:n välisen salauksen murtaminen antaa hakkerille mahdollisuudet päästä käsiksi ydinverkkoon. Päästyään ydinverkkoon pääsee hakkeri käsiksi kaikkeen verkossa liikkuvaan dataan ja puheluihin. BTS:n ja BTC:n välisen kaapelin salakuuntelu saattaa pysyä salassa pitkäänkin jos murtautuja on taitava, ja täten murtautuja pystyy saamaan käsiinsä huomattavat määrät tietoa. [Pesonen, 1999]

4.6. Sosiaalinen hakkerointi

Kuten moniin muihinkin järjestelmiin, on GSM-verkkoon mahdollista päästä käsiksi myös sosiaalisen hakkeroinnin keinoin. Sosiaalisen hakkeroinnin keinoin toimiva hyökkääjä yleensä tekeytyy palvelun tarjoajan henkilöstön jäseneksi ja onnistuu tällä tavoin saamaan käsiinsä kaikenlaista tietoa aina asiakastiedoista palvelun tarjoajan omiin tärkeisiin tietoihin. Näitä tietoja hyökkääjä voi myydä eteenpäin tai käyttää palvelun tarjoajan kiristämiseksi.

Hyökkääjä voi yksinkertaisimmillaan aloittaa hyökkäyksensä valmistelun katsomalla puhelinluettelosta jonkin tietyn paikkakunnan toimintayksikön henkilöstön nimiä, ja tilata sitten vaikka luettelon jonkin toisen paikkakunnan henkilöstöstä tekaistuun sähköpostiosoitteeseen. Tämän jälkeen hyökkääjä voi esiintyä luettelossa olevina henkilöinä jonkin muun paikkakunnan henkilöstön jäsenelle.

Hyökkääjä voi tiedustella operaattorin henkilöstön jäseniltä erilaisia asioita tekeytymällä toisella paikkakunnalla toimivan yksikön työntekijäksi, ja täten saada käsiinsä tärkeitä tietoja. Tämä on mahdollista koska palveluntarjoajan henkilöstön jäsenen on mahdotonta puhelimitse tunnistaa henkilöä, jota ei ole koskaan aikaisemmin tavannut.

Kolmannen paikkakunnan henkilöstön jäseneltä hyökkääjä voi tilata jotain tärkeitä tietoja esimerkiksi kytkentäkeskuksien sijainneista. Hyökkääjä voi myös anastaa hyökkäykseen tarvittavat laitteet tekeytymällä palveluntarjoajayrityksen huoltomieheksi. Täten ovela sosiaalinen murtautuja voi saada kaikki samat edellytykset verkon käyttöön kuin palvelun tarjoajan henkilöstölläkin on.

Ihmisten salasanoja ja muita tärkeitä tietoja saattaa maasta riippuen saada selville heitä haastattelemalla ja esimerkiksi esiintymällä toimittajana. Joissain maissa sosiaalinen hakkeri pääsee lahjonnallakin pitkälle. Pelkkä tuntemattoman henkilön tarjoama ateria ja sitä seuraava keskustelu voi johtaa siihen, että työntekijä kertoo tärkeitä asioita työpaikastaan. Sosiaalinen hakkeri voi esimerkiksi seurata uhriaan ravintolaan, ja aloittaa siellä keskustelun haukkumalla omaa keksittyä työpaikkaansa. Tämä saattaa houkutella myös hakkerin uhria kertomaan oman tarinansa, ja joitain tärkeitä työhön liittyviä tietoja siinä ohessa. Jos sosiaalinen hakkeri sattuu jostain syystä kiinnostumaan sukulaisensa tai perhetuttunsa firmasta, voi hän saada selville monia asioita yhteisten vapaa-ajan harrastusten lomassa.

4.7. Passiiviset uhat

Passiivisen uhan saattavat GSM-verkossa aiheuttaa huonosti hoidetut kytkentätyöt, tekniset viat, yhteensopivuusongelmat, ongelmat palveluntarjoajan ohjelmistoissa tai muut mahdolliset inhimilliset virheet. Myös luonnonilmiöt tai ulkopuolisten toimijoiden tekemät huoltotyöt voivat aiheuttaa hetkellisiä tilanteita, joissa kolmannen osapuolen pääsy verkkoon helpottuu.

4.8. Raa'an voiman hyökkäys A5-algoritmia vastaan

Tosiaikainen raa'an voiman hyökkäys GSM-järjestelmää vastaan on Pesosen [Pesonen, 1999] mukaan usein spekuloitu hyökkäysmalli, mutta käytännössä se ei kuitenkaan ole käyttökelpoinen. Tämä johtuu siitä, että hyökkäys on monimutkainen ja vaatisi paljon aikaa onnistuakseen ja mahdollistaakseen GSM-puheluiden tosiaikaisen salakuuntelun. MS:n (ks. kuva 1) ja BTS:n välisten asioiden osittain tallentaminen on kuitenkin mahdollista ja täten hyökkäys voidaan aloittaa jälkikäteen. Teoriassa A5/1-algoritmia vastaan suoritettu hyökkäys veisi Pentium III 600MHz-tietokoneelta aikaa 250 tuntia. Tästä ajasta on kuitenkin mahdollista saada jopa kolmasosa pois erilaisten hyökkäyksen tehon optimointimenetelmien avulla. [Pesonen, 1999]

Jos prosessoritehon kasvu vaikuttaisi suoraan hyökkäykseen vaadittavaan aikaan, menisi nykyteknologialla edellä mainittuun hyökkäykseen aikaa noin kaksi päivää eli 50 tuntia. Hyökkäykseen vaadittava aika ei kuitenkaan välttämättä vähene lineaarisesti prosessoritehon kasvaessa, koska laitteiston pullonkaulaksi saattaa hyökkäyksessä nousta jokin toinenkin seikka kuin pelkästään prosessoriteho.

4.9. Hajota ja hallitse -hyökkäys A5-algoritmia vastaan

Hajota ja hallitse -hyökkäys on jopa satoja kertoja raa'an voiman hyökkäystä tehokkaampi. Hajota ja hallitse -hyökkäys perustuu siihen että tiedetään osa selkotekstistä. Hyökkäyksessä hyökkääjä yrittää määritellä LSFR-rekisterin alkuperäiset arvot tunnetusta tietovirrasta. Hyökkääjä tarvitsee hyökkäykseen 64 peräkkäistä tietovirran bittiä, jotka pystytään selvittämään jos hyökkääjä tietää pätkän salattua tekstiä, ja vastaavan pätkän selkokielistä tekstiä. Käytännössä hajota ja hallitse -hyökkäys toteutetaan arvaamalla kahden lyhyemmän LSFR-rekisterin arvot ja laskemalla kolmannen LSFR:n arvo tietovirrasta. [Pesonen, 1999]

5. Uhkien torjunta

GSM-verkon tietoturvauhkille on olemassa myös joitakin vastatoimia, joiden avulla tietoturvaongelmien aiheuttamia haittoja pystytään pienentämään. Parhaimmassa tapauksessa vastatoimien avulla pystytään poistamaan kokonaan jokin yksittäinen tietoturvaongelma. Seuraavassa esittelen omia pohdintojani ratkaisuksi GSM-verkon yleisimpiin tietoturvauhkiin.

Mielestäni GSM-verkon tietoturvauhkien vastatoimien pohdinta on luonnollista aloittaa SIM-kortin kloonauksesta. SIM-kortin kloonausta vastaan voitaisiin kehittää SIM-kortteihin elektroninen sormenjälki, joka olisi suunniteltu niin, ettei SIM-korttia voisi kloonata, vaan kortti olisi aina oma yksilönsä käyttötilanteesta ja vallitsevista olosuhteista riippumatta. SIM-korttien jakelua voisi myös tehostaa niin, että jokaista käytössä olevaa SIM-korttia vastaan tulisi operaattorilla olla selvitys liittymänavaustilanteesta, ja SIM-kortit jotka eivät ole käytössä olisi tehty toimimattomiksi niin, ettei niitä voi saattaa toimintakuntoon ilman elektronista varmennusta operaattorin järjestelmästä.

Ilmatien salauksen murtamista vastaan pystyttäisiin toimimaan parantamalla A8-algoritmin ja COMP128-algoritmin ominaisuuksia. Ilmatien salauksen murtamiseen liittyvää tukiaseman väärentämistä vastaan olisi mahdollista muuttaa GSM-verkkoa niin, että myös tukiasemien autentikoisivat itsensä päätteelle. Täten GSM-verkkoon kytketty päätelaite tunnistaisi heti väärennetyn tukiaseman ja torjuisi kaikki tukiaseman tekemät autentikointipyynnöt. Tulevaisuuden verkoissa tällainen ominaisuus on ilmeisesti vakiona.

Autentikointikeskusta vastaan tehtyjä hyökkäyksiä pystyttäisiin torjumaan parhaiten parantamalla sekä autentikointikeskuksen fyysistä, että ohjelmallista turvallisuutta.

Sisäisiä uhkia vastaan pystyttäisiin taistelemaan parhaiten palveluntarjoajan henkilöstön taustatietojen ja persoonallisuuden selvittämisellä jo uutta henkilöstöä rekrytoidessa. Henkilöstön jäsenten pääsy kannattaisi rajata vain niille GSM-verkon alueille, joita he työssään tarvitsevat. Henkilöstöä tulisi myös opastaa erilaisiin toimintamalleihin mahdollisissa poikkeustilanteissa.

Viestintäverkon turvallisuutta hyökkäyksiä vastaan voidaan parantaa samoin keinoin kuin autentikointikeskuksen turvallisuutta. Erilaisten fyysisten ja ohjelmallisten tietoturvaratkaisujen kehittämisen ja ajanmukaisen päivittämisen avulla pystytään ehkäisemään myös viestintäverkkoon kohdistuvia hyökkäyksiä.

Sosiaalista hakkerointia vastaan tehty opas olisi suureksi avuksi hakkerien torjunnassa. Kun henkilöstölle on tehty selväksi kehen tulee missäkin tapauksessa ottaa yhteyttä, ei sosiaalinen hakkeri pysty perustelemaan tarvettaan päästä käsiksi tärkeisiin tietoihin. Operaattoriyrityksestä riippuen on mahdollisuuksia tehdä myös jäljitysjärjestelmä hakkerin puheluiden jäljittämiseksi. Jos hakkeri sattuu käyttämään operaattoriyrityksen matkapuhelinliittymää, on hänet mahdollista paikantaa matkapuhelinverkon solun tarkkuudella vallitsevasta lainsäädännöstä riippuen.

Passiivisia uhkia on usein hankala torjua niiden luonteesta johtuen. Luonnonilmiön aiheuttama tuho on niin arvaamaton, että sitä vastaan on melkein mahdotonta toimia ennalta. Luonnonilmiöiden aiheuttamia sähköhäiriöitä vastaan on mahdollista toimia erilaisten varavirtajärjestelmien avulla, mutta verkon vaurioitumiseen on hankalampaa varautua. Ulkopuolisten tekemien huoltotöiden aiheuttamia haittoja vastaan voidaan käyttää tiedottamisen keinoja. Tiedottamalla huoltotyöntekijää kohdealueella tehtävän huoltotyön teoreettisista haitoista pystytään parhaiten ehkäisemään ongelmien syntymistä.

A5-algoritmia vastaan tapahtuvia hyökkäyksiä pystytään parhaiten torjumaan parantamalla algoritmin rakennetta. A5-algoritmista käytetään Euroopassa sen turvallisinta A5/2-versiota, joten esimerkiksi muualla maailmassa käytössä olevaan A5/1-algoritmiin verrattuna on Euroopassa tapahtuvat GSM-puheluiden suojaus hoidettu hyvin.

6. Yhteenveto

Henkilöstön koulutus on GSM-verkkoon kohdistuvien hyökkäysten torjunnassa ehdottomasti toimivin ratkaisu moniin ongelmiin. Sen sijaan suuret muutokset, jotka koskevat GSM-verkon toimintaa, alkavat nykyisin näyttää melko epätodennäköisiltä uusien langattomien teknologioiden tehdessä tuloaan. GSM-verkon rakenteen ja toiminnan muuttaminen on myös rahallisesti henkilöstön koulutusta kalliimpi ratkaisu.

GSM-salauksen virheistä on kuitenkin otettu opiksi, ja tiettävästi [Brewer et al. 1998] COMP128-algoritmista on tulossa käyttöön uusi parannettu versio, COMP128-2. GSM-verkon puutteita on myös onnistuneesti korjattu GPRS- ja UMTS-verkkoja suunnitellessa ja täten seuraavan sukupolven GSM-verkkojen tulevaisuus näyttää melko hyvältä.

GSM-salauksen kuten muidenkin langattomien salauksien yhtenä ongelmana tulee luultavasti aina olemaan päätelaitteiden huono laskentakapasiteetti ja täten rajoitetut mahdollisuuden käyttää salausalgoritmeja. Jos langattomaan päätelaitteeseen laitetaan liian raskas salausalgoritmi, kuluttaa se luultavimmin päätelaitteen akkua entistä nopeammin ja toisaalta myös päätelaitteen käyttö hidastuu, kun sen laskentakyky joutuu koville. Jos tulevaisuudessa keksitään jokin huomattavasti nykyisiä akkuja tehokkaampi voimanlähde langattomiin laitteisiin, pystytään salausta parantamaan, sillä laskentateho tuntuu monessa tapauksessa riippuvan vain virrankulutuksesta ja käytettävissä olevasta akkukapasiteetista.

Puutteistaan huolimatta GSM-verkko on nykyisessä muodossaankin toiminut melko hyvin, eikä suuria verkkoon murtautumisia ei ole ilmennyt. GSM-verkon käyttäjäkunnan luottamus nykyisen verkon turvallisuuteen tuntuu myös olevan hyvä.

Lähteideni valossa GSM-verkkoon murtautuminen taas näyttää olevan mahdollista. Voihan olla että jopa potentiaaliset GSM-verkkoon murtautujat uskovat turvallisuuden olevan niin hyvä, että vaikka verkkoon murtautuminen onnistuisikin, on kiinni jäämisen riski erittäin suuri. On todennäköistä, että jos GSM-verkkoon Suomessa tehty murtautuminen saisi julkisuusarvoa, seuraisi siitä ajojahti jossa murtautuja lopulta jäisi kiinni.

Nyt kun mobiilikaupankäynnin [Setec, 2003] mahdollisuuksia suunnitellaan, on operaattoreiden saatava asiakkaat luottamaan itseensä kuin pankkiin. Tämän vuoksi en ihmettelisikään jos eri operaattorit järjestäisivät erilaisia kampanjoita

julkisuuskuvansa parantamiseksi myös tietoturvan alueella, ja tästä johtuen verkkoon kohdistettu murto selvitettäisiin perinpohjaisesti.

Viiteluettelo

[Biryukov et al. 2000] Biryukov, Shamir ja Wagner - Real Time Cryptanalysis of A5/1 on PC (2000). http://cryptome.org/a51-bsw.htm [5.5.2004]

[Brewer et al. 1998] Brewer, Borisov, Goldberg, Wagner - GSM Cloning (1998). http://www.isaac.cs.berkeley.edu/isaac/gsm.html [5.5.2004]

[Briceno et al. 1998] Briceno, Goldberg and Wagner - An implementation of the GSM A3A8 algorithm (1998). http://www.iol.ie/~kooltek/a3a8.txt [5.5.2004]

[Goldberg, Briceno, 1998] GSM Cloning (1998). http://www.isaac.cs.berkeley.edu/isaac/gsm-faq.html [5.5.2004]

[Golic, 1998] Jovan Dj. Golic - Cryptoanalysis of Alleged A5 Stream Cipher (1998). http://downloads.securityfocus.com/library/a5-hack.html [5.5.2004]

[Isaac, 1998] Smartcard Developer Association Clones Digital GSM Cellphones (1998). http://www.isaac.cs.berkeley.edu/isaac/gsm-press.html [5.5.2004].

[Pesonen, 1999] Lauri Pesonen - GSM Interception (1999). http://www.dia.unisa.it/professori/ads/corso-security/www/CORSO-9900/a5/Netsec/netsec.html [5.5.2004].

[Roe, 2002] Mike Roe - The GSM Security Technical Whitepaper for 2002 (2002).http://www.hackcanada.com/blackcrawl/cell/gsm/gsm_security.html [5.5.2004].

[Setec, 2003] Setec – Pankkimaksaminen mobiiliverkossa. (2003). http://www.setec.fi/suomi/telecom/pkiesim/mobiilimaksaminen.html [5.5.2004].

[Vaudenay, 1992] S. Vaudenay – FFT-Hash-II is not yet Collision-free (1992). http://lasecwww.epfl.ch/pub/lasec/doc/liens-92-17.A4.ps [5.5.2004]

[Vesanen, 2003] Ari Vesanen, Langattoman tietoliikenteen tietoturva, luentomateriaali. Oulun yliopisto. (2003). http://www.tol.oulu.fi/%7Eavesanen/Langaton_TT/ [5.5.2004]

[Young, 1998] Young - Technical Information GSM System Security Study (1998). http://jya.com/gsm061088.htm [5.5.2004]

Verkkoreportaasin erityiskysymyksiä

Maria Rikala

Tiivistelmä.
Käsittelen tutkielmassani verkkoreportaasia ja verkkojulkaisujen problematiikkaa yleisemmin. Määrittelen aiheeseen liittyviä peruskäsitteitä sekä erittelen verkkomedian erityispiirteitä ja verkkoreportaasin muodostavia elementtejä. Lopuksi tarkastelen vielä verkkoreportaasia käytettävyyden näkökulmasta.

Avainsanat ja -sanonnat: Verkkoreportaasi, hypermedia, käytettävyys.
CR-luokat: H.5.4

1. Johdanto

Hypermediaa hyödyntäviä journalistisia teoksia on julkaistu verkossa varsin vähän. Huomattava osa verkkojournalismista muistuttaa melko pitkälti painettua viestintää. Tietoverkon mahdollistamaa mediakonvergenssia ja toiminnallisia ominaisuuksia ei mielestäni ole hyödynnetty riittävästi.

Aloitan aiheen käsittelyn pohtimalla verkkoreportaasin lähtökohtia ja määrittelemällä keskeisiä käsitteitä. Lisäksi paneudun verkkoreportaasille ominaisiin piirteisiin ja erityisesti käytettävyyteen, lähinnä suunnitteljan näkökulmasta.

2. Lähtökohtia

Internet tarjoaa periaatteessa rajattoman ilmaisukanavan kenelle tahansa, ja toisaalta myös kaikille (ainakin teoreettisen) mahdollisuuden päästä käsiksi likimain kaikkeen tietoverkossa olevaan dataan. Ymmärrettävästi verkossa olevien dokumenttien sisältö, muoto ja tekninen toteutus saattavat vaihdella melkoisesti.

2.1. Verkkojulkaisut

Siinä missä tietoverkkojen sisältö on moninaista, myös sisältöä kuvaavia käsitteitä leimaa moninaisuus ja epämääräisyys. Esimerkiksi Kuusiston ja Pippurin [1998] mukaan verkossa ilmestyvälle julkaisulle ei ole olemassa täsmällistä termiä. Osa käytettävistä nimityksistä liittyy verkkojulkaisemisen tekniikkaan,

osa medioihin tai niiden suhteiin ja osa jopa pyrkii ottamaan mukaan käyttäjänkin [Kuusisto ja Pippuri 1998]. Yksinkertaisimmillaan verkkojulkaisemisella tarkoitetaan minkä tahansa digitaalisen dokumentin levittämistä Internetissä.

Kuusisto ja Pippuri tarkoittavat verkkojulkaisulla lähinnä elektronista lehteä, mikä käy mielestäni selvästi ilmi heidän verkkojulkaisun määritelmästään:

> ”Verkkojulkaisu on Intemetissä levitettävä ja luettavissa oleva, säännöllisesti ilmestyvä journalistinen kokonaisuus. Se voidaan toteuttaa teknisesti usealla eri tavalla, esimerkiksi HTML-kielellä. Tavallisten journalististen tekstien lisäksi se voi sisältää myös vuorovaikutteisuutta, kuten keskustelupalstoja, linkityksiä, hypertekstiä ja palautemahdollisuuden.” [Kuusisto ja Pippuri 1998]

Oma käsitykseni verkkojulkaisusta on kuitenkin hieman laajempi erityisesti ilmestymisen kannalta. Mielestäni julkaisu voi olla myös kertaluonteinen tai epäsäännöllinen. Verkkoreportaasi puolestaan on mielestäni eräs verkkojulkaisun alalajeista tai tyypeistä.

2.2. Pelkistetyt sivustot ja multimediasivustot

Kuten jo edellä mainitsin, useimpia verkkojulkaisuja ei ole juurikaan tehty verkon ehdoilla, vaan muiden medioiden vaatimuksia vastaaviksi tehdyt sisällöt siirretään sellaisenaan tietoverkkoon. Nicklas Koski [1999] toteaa, että multimediasisällön tekeminen verkkoon jää usein puolitiehen, eikä eri medioita aina yhdistetä niin, että ne muodostaisivat yhtenäisen tuotteen [Koski 1999]. Erityisesti ääni- ja videotiedostot jäävät Kosken [1999] mukaan usein irrallisiksi elementeiksi. Hän jakaakin verkkosivustot pelkistettyihin ja multimediasivustoihin.

> ”Verkkotekniikka mahdollistaa tällä hetkellä todella näyttävän multimedian esittämisen. Pullonkaulana ovat kuitenkin vielä hitaat yhteydet. Erilaisilla sivustoilla on hyvinkin erilaiset käyttäjäkunnat erilaisine tarpeineen. Kaikki eivät halua viimeisintä tekniikkaa sivuille. Seurauksena on sivustojen jakautuminen kahteen ryhmään: visuaaliselta ilmeeltään pelkistettyihin sivustoihin ja multimediasivustoihin.”
> [Koski 1999]

Kosken [1999] mukaan visuaaliseen pelkistämiseen tähtäävät esimerkiksi päivittäispalveluita (uutisia, keskustelupalstoja ym.) tarjoavat sivut, joilla käydään usein. Multimediasivustot sen sijaan pyrkivät tuottamaan elämyksiä, joiden

eteen käyttäjät ovat valmiita näkemään enemmän vaivaa ja odottamaan pidempään sivuston latautumista, koska sivustoilla käydään harvoin [Koski 1999].

Pelkistetyt sivustot ja multimediasivustot eroavat visuaalisen ilmeensä lisäksi myös rakenteeltaan. Pelkistetyn sivuston rakenne suunnitellaan siten, että palvelu on mahdollisimman nopea ja helppo käyttää, kun taas multimediasivustoilla tieto on usein jaettu pieniin kokonaisuuksiin, joihin pääsee vain monen mutkan kautta. Koski [1999] huomauttaa, että pelkistetyillä sivustoilla tietoa tarjotaan heti eikä sitä piiloteta hierarkiaan monen sivun taakse, kun taas multimediasivustoilla alasivuja käytetään runsaasti ja joka sivulla on vain vähän tietoa.

2.3. Mediakonvergenssi ja verkon erityisominaisuuksia

Mediakonvergenssi eli eri medioiden yhdistyminen on multimedian keskeisimpiä ominaisuuksia. Erityisesti verkkoreportaasin näkökulmasta on tärkeää, että eri mediaelementit muodostavat yhtenäisen kokonaisuuden.

Verkkoreportaasit olisi mielestäni syytä suunnitella siten, että niihin rakennetaan piirteitä tai ominaisuuksia, jotka ovat mahdollisia yksinomaan tietoverkossa tai muissa tietokonekäyttöisissä sovelluksissa. Tällaisia erityisominaisuuksia ovat muun muassa vuorovaikutteisuus, toiminnallisuus ja assosiatiivisuus. Käyttäjän vapaavalintainen liikkuminen teoksessa ja mahdollisuus kontrolloida luku- tai käyttökokemuksen etenemistä olisi hyvä mahdollistaa teoksen rakennetta suunniteltaessa.

2.4. Hypermediaa hyödyntäviä verkkoreportaaseja

Kuten jo edellä totesin, valitettavan usein verkkojournalismi jäljittelee perinteisempiä medioita, erityisesti painettua mediaa. Ja jos verkkojulkaisussa onkin käytetty eri mediaelementtejä, ne jäävät turhan helposti toisistaan irrallisiksi.

Suomalaisista verkkoreportaaseista on syytä mainita Helsingin Sanomien Verkkoliitteessä julkaistut, hypermedian keinoja hyödyntäneet webortaasit. Susanna Pasula [2002] kiittelee Helsingin Sanomien Verkkoliitettä hypermediamuotoisen verkkojournalismin edelläkävijäksi ja muistuttaa, että webortaasit ovat voittaneet myös kansainvälisiä kilpailuja. Vaikka webortaaseilla oli selvästi kysyntää ja ne kiinnostivat käyttäjiä, Helsingin Sanomat on jo lopettanut hypermediatuotannon taloudellisista syistä: webortaasien tekeminen oli kallista [Pasula 2002].

> “Verkkoreportaasi soveltuu huonosti nopeaan tuotantotahtiin. Se vaatii suuren tuotantotiimin ja eri alojen erikoisosaamista, riippuen siitä minkälainen reportaasista tulee. On siis ymmärrettävää, että nykyisissä suomalaisissa verkkojulkaisuissa ei verkkoreportaaseja juuri ole pystytty tuottamaan.” [Koski 1999, Kämäräinen 1999]

Verkkoreportaaseja on toteutettu myös Tampereen yliopiston, Jyväskylän yliopiston ja Taideteollisen korkeakoulun Elektronisen kuvajournalismin maisteriohjelman puitteissa. Koska opiskelijat ovat kuvaorientoituneita, myös verkkoreportaaseissa on kiinnitetty erityisen paljon huomioita kuvalliseen ilmaisuun ja visuaaliseen suunnitteluun. Elektronisen kuvajournalismin maisteriohjelman yhteydessä tehdyistä verkkoreportaaseista nostan esiin erityisesti Tehtaan kuolema -nimisen teoksen, joka on suunniteltu ja toteutettu nimenomaan verkon ehdoilla. Nojaudun tutkielmassani melko paljon Tehtaan kuolema -reportaasin tekijätiimiin kuuluneiden Nicklas Kosken ja Anni Kämäräisen analyysiin työstään ja heidän ajatuksiinsa verkkoreportaasista ilmaisumuotona.

Ulkomaisista verkkoreportaaseista voisi mainita esimerkiksi Benettonin Colors-verkkojulkaisun reportaasit.

3. Keskeisiä käsitteitä

3.1. Hyperteksti, multimedia, hypermedia

Nielsen [1995] määrittelee hypertekstin ei-peräkkäiseksi tai epälineaariseksi tekstirakenteeksi, jossa linkit yhdistävät tietoa sisältäviä solmuja. Koska nykyiset järjestelmät sisältävät mahdollisuuden käyttää muitakin mediaelementtejä kuin tekstiä, jotkut käyttävät mieluummin termiä hypermedia [Nielsen 1995]. Vaikka Nielsen [1995] määritteleekin hypermedian hypertekstin ja multimedian yhdistelmäksi, hän pitäytyy mieluummin hypertekstin käsitteessä.

Itse miellän hypertekstin ennen kaikkea rakenteellisena konstruktiona sekä multi- ja hypermedian esityksen tai teoksen muotona. Näin ollen olen siis Nielsenin kanssa pitkälti samoilla linjoilla määrittellessäni hypermedian hypertekstirakenteiseksi ja multimediamuotoiseksi tuotokseksi.

Näkemyksestä hypertekstistä rakenteena kertoo sekin, että Nielsen [1995] pitää hypertekstin tärkeimpänä kriteerinä sitä, että lukija tai käyttäjä tuntee voivansa edetä tiedon verkostossa vapaasti, mikä korostaa samalla käyttöliittymää ja assosiatiivisuutta.

3.2. Verkkoreportaasi

Koski [1999] ja Kämäräinen [1999] käyttävät Tehtaan kuolema –teoksestaan lähinnä paineituista lehdistä lainattua käsitettä reportaasi. Kämäräisen [1999] mukaan reportaasissa yhdistyvät informatiivisuus ja elämyksellisyys.

> "Reportaasilla on nimenomaan painetussa lehdessä juuri niitä ominaisuuksia, joita halusimme oman työmme tarjoavan. [...] Reportaasi on myös moniselitteinen, jopa hieman epämääräinen journalismin muoto verrattuna vaikkapa uutiseen. Käsitteen alle mahtunee siis myös verkkoon tarkoitettu kokeileva teos, verkkoreportaasi." [Kämäräinen 1999]

Kämäräinen [1999] pohtii myös muita mahdollisia nimityksiä verkkoreportaasille: esimerkiksi hyperreportaasi- tai multimediareportaasi-nimityksestä ei kuitenkaan kävisi ilmi raportaasin julkaisukanava, tietoverkko.

Pasula [2002] sen sijaan on päätynyt käyttämään vastaavista tuotoksista nimitystä verkkodokumentti, koska hänen mielestään reportaasi edellyttää aiheeltaan enemmän ajankohtaisuutta kuin dokumentti. Pasula [2002] määrittelee verkkodokumentin audiovisuaaliseksi dokumentiksi, jossa hyödynnetään yhtä tai useampaa hypermedian ominaisuuksista.

4. Verkkoreportaasin ominaispiirteitä

4.1. Multilineaarisuus

Verkkoreportaasin kannalta yksi olennaisimmista kysymyksistä liittyy teoksen ja erityisesti tekstin rakenteeseen. Perinteisiin teksteihin liitetään usein lineaarisuus, kun taas hypertekstirakenne mielletään epälineaariseksi. Kämäräisen [1999] mielestä lineaarisella tekstillä tarkoitetaan peräkkäisesti etenevää tekstiä, jossa tekstiosien järjestys on kirjoittajan määräämä ja pysyvä. Epälineaarisessa tekstissä tekstiosien järjestys vaihtee lukukerrasta toiseen [Kämäräinen 1999].

Jako ei kuitenkaan ole näin yksiselitteinen, sillä sekaannuksia voi aiheuttaa se, puhutaanko tekstin rakenteesta vai käyttötavasta. Rakenteeltaan lineaarista tekstiä voidaan lukea epälineaarisesti, sieltä täältä. Toisaalta hypertekstirakenteessa tekstilohkot sisältävät lineaarista tekstiä, mutta lohkojen välillä liikutaan epälineaarisesti. Lisäksi lineaarisuus tai epälineaarisuus voi olla ajallista tai tilallista.

Parhaiten useimpia hypertekstirakenteita kuvaa mielestäni multilineaarisuuden käsite. Multilineaarinen teos koostuu lineaarisista tekstin osuuksista,

jotka esiintyvät teoksen rakenteessa valinnaisessa järjestyksessä [Kämäräinen 1999], eli käyttäjä voi esimerkiksi valita eriävistä poluista tai vaihtoehdoista haluamansa ja muodostaa näin erilaisia kokonaisuuksia. Multilineaarinen teos vaatii siis useita erilaisia polkuja ja mahdollisesti myös loppuratkaisuja, joten multilineaarisuus lisää käyttäjän mahdollisuutta kontrolloida teosta [Pasula 2002].

4.2. Vuorovaikutteisuus

Uuteen mediaan ja hypertekstiin liitetään yleensä myös interaktiivisuus. Minna Tarkan [1996] mukaan vuorovaikutteisuus kuvasi alun perin ihmisten keskinäistä kanssakäymistä, mutta käsite on saanut uuden merkityksen digitaalisessa mediakulttuurissa, ja nyt sitä käytetään ihmisen ja koneen tai tietoteknisen järjestelmän välisestä interaktiosta.

> "Mahdollisuus teoksen ja käyttäjän väliseen kommunikaatioon tuo esitykseen dramaturgista jännitettä. Vuorovaikutustilanteessa käyttäjä siirtyy passiivisen katsojan roolista aktiiiseksi toimijaksi. Parhaimmillaan vuorovaikutteisuus antaa käyttäjälle kokemuksen siitä, että hän voi aidosti vaikuttaa siihen, miten esitys toimii ja saada todellista hyötyä ja iloa sovellukseen rakennetusta toiminnallisuudesta." [Kanerva *et al.* 1997]

Kuusiston ja Pippurin [1998] mielestä interaktiivisuus-käsitettä käytetään varsin kirjavasti, eikä pelkästään vapaa liikkumismahdollisuus vielä tee julkaisusta interaktiivista. Aidossa interaktiossa julkaisu joustaa ja muuttuu käyttäjän mukaan eikä päinvastoin, jolloin tuote syntyy teoksen ja käyttäjän yhteistyönä [Kuusisto ja Pippuri 1998]. Raine Koskimaa [1999] on eri mieltä; hänen mukaansa hypertekstiä lukiessa tarvitaan aktiivista toimintaa (interaktiota) tekstin vaihtoehtoisten polkujen valinnassa.

Johan Fornäs [1999] huomauttaa, että interaktiivisuus muodostuu ennen kaikkea median ja käyttäjän suhteesta, jolloin mitä tahansa media voidaan käyttää tai olla käyttämättä vuorovaikutteisesti.

Katie Salen ja Eric Zimmerman [2004] ovat koonneet vuorovaikutteisuuden määritelmistä neljä interaktiomoodia eli vuorovaikutuksen tasoa:

> 1) Kognitiivinen interaktiivisuus / tulkitseva osallistuminen: psykologinen, tunteellinen ja älyllinen osallistuminen henkilön ja järjestelmän välillä.

2) Funktionaalinen interaktiivisuus / toiminnallinen osallistuminen: funktionaaliset, rakenteelliset interaktiot järjestelmän materiaalisten (todellisten tai virtuaalisten) komponenttien kanssa.

3) Ekplisiittinen interaktiivisuus / osallistuminen suunniteltuihin valintoihin ja toimintoihin: vuorovaikutteisuuden itsestäänselvin merkitys, yleinen osallistuminen kuten hypertekstirakenteen epälineaaristen linkkien klikkaaminen. Eksplisiittinen vuorovaikutus käsittää esimerkiksi valinnat, sattumanvaraiset tapahtumat, dynaamiset simulaatiot ja muut vuorovaikutteiseen kokemukseen ohjelmoidut toiminnot.

4) Objektin ylittävä interaktiivisuus / osallistuminen objektin kulttuurissa: yksittäisen, suunnitellun järjestelmän tuottaman kokemuksen ulkopuolinen vuorovaikutus. Selkein esimerkki tästä on fanikulttuuri, jossa osallistujat rakentavat yhteisiä todellisuuksia käyttäen suunniteltuja järjestelmiä raaka-aineena. [Salen ja Zimmerman 2004]

4.3. Immersiivisyys

Multi- tai hypermedian yhteydessä immersiivisyydellä tarkoitetaan sitä, että teos tuntuu ikään kuin imaisevan mukaansa [Kanerva *et al.* 1997]. Käyttäjän ei tarvitse varsinaisesti opetella käyttöliittymän ominaisuuksia, vaan teokseen sukeltaminen tapahtuu lähes itsestään, koska immersiivisessä teoksessa on helppo liikkua [Kanerva *et al.* 1997].

Tarkan [1996] mukaan immersiivisyys on uusi esteettinen kriteeri, jonka avulla punnitaan käyttäjän kokemusta. Immersiivisyyteen vaikuttavat illusorisuus tai mimeettisyys, toimivuus eli käyttöliittymän intuitiivisuus ja järjestelmän nopeus, tapahtumien vauhti ja dramaturgia sekä vuorovaikutuksen määrä ja laatu [Tarkka 1996]. Tarkan [1996] mielestä teos on immersiivinen, kun käyttäjä uppoutuu kerronnan tilaan.

4.4. Tilallisuus

Voimakas ”nyt-hetken” korostaminen on interaktiivisen median perusta, sillä teosten ajallinen ulottuvuus tiivistyy valintahetkenä napin painallukseen [Tarkka 1996]. Presentaatio korostaa suoraa, ajassa ja tilassa muuntuvaa kokemusta, ja teoksen ja kokijan samanaikaisen ja –tilaisen läsnäolon sekä yhteisen ”luomisprosessin” ansiosta kokija luo teoksen valinnoillaan [Tarkka 1996].

Digitaaliset ympäristöt hahmotetaan usein tiloina, joissa voi navigoida. Pasulan [2002] mukaan tilallisuutta on tähän mennessä käytetty parhaiten

tietokonepeleissä, verkkoreportaaseissa tilallisuutta on hyödynnetty lähinnä käyttöliittymissä.

4.5. Mediaelementtien tyypillisiä piirteitä verkkoreportaasissa

4.5.1. Teksti

Verkkoreportaasin tekstiä määrittää tietenkin hypertekstirakenne. Ja koska hypertekstirakenne on yleensä multilineaarinen, on tarinan kerronta varsin haasteellista. Perinteinen aristotelinen dramaturgia on usein melko hankala tai joissakin tapauksissa lähes mahdoton toteuttaa. Kämäräinen [1999] näkee erityisen ongelmallisena tarinan lopun.

Verkkoreportaasin teksteille on ominaista myös lyhyys. Esimerkiksi Kämäräinen [1999] kertoo yrittäneensä kirjoittaa Tehtaan kuolema -reportaasin elämysosaan vain sitä, mitä luettaisiin. Hänen mielestään lopputulos muistuttaa runoa, aforismia tai tajunnanvirtaa. Lisäksi verkkoreportaasin kielioppia leimaavat yksittäiset, tunnelmaa luovat iskusanat, välimerkittömyys ja pienet kirjaimet [Kämäräinen 1999].

4.5.2. Kuva

Hatvan [1998] mukaan kuvittajan päämäärä on tehdä vastaanottajan tiedon (tai elämyksen) hankinta mukavammaksi. Hatva [1998] jakaa kuvituksen tehtävät kahteen ryhmään: osa vetoaa esitarkkaaviin, osa tietoiseen ajatteluun perustuviin prosesseihin. Edelliseen ryhmään kuuluvat esteettisyys ja huomion ohjaaminen (painotus), jälkimmäiseen ryhmään dokumentoitavuus, orientoivuus ja symbolina toimiminen. [Hatva 1998]

4.5.3. Ääni

Äänen avulla on mahdollista muuttaa teoksen tunnelmaa huomattavastikin ja lisätä näin käyttäjälle muodostuvan kokemuksen immersiivisyyttä. Ääni on kuitenkin multimediatuotannoissa ”lapsipuolen asemassa”, eikä vakiintunutta äänenkäyttötapaa ole syntynyt [Kanerva *et al.* 1997]. Äänikerronta perustuu pääasiassa musiikkiin, puheeseen ja tehosteääniin, ja enimmäkseen käytetään radiosta, televisiosta ja elokuvista tuttua äänimaailmaa [Kanerva *et al.* 1997].

4.5.4. Liikkuva kuva

Hatva [1998] kehottaa käyttämään liikkuvaa videokuvaa silloin, kun haluaa herättää huomion (muiden, liikkumattomien elementtien joukosta), näyttää

miten jokin toimii tai tehdään tai todistaa, että jokin todella tapahtui. Animaatio puolestaan sopii liikkeen pelkistämiseen tai karrikointiin, sisällön yksinkertaistamiseen tai silloin, kun realistista kuvaa ei voida tai haluta tehdä [Hatva 1998].

4.5.5. Taitto ja muu graafinen suunnittelu

Multimediajulkaisun layoutilla on sekä visuaalinen että toiminnallinen funktio, sillä layout sekä välittää kuvaa sisällöstä että ohjaa käyttäjän toimintaa [Kanerva *et al.* 1997]. Koski [1999] jakaa verkkoreportaasin taittotyylin kahteen ryhmään, lehtimäiseen ja televisiomaiseen:

> "Lehtitaittoa käytetään perinteisillä HTML-sivuilla. [...] Teksti asetetaan leveälle keskipalstalle ja kapeammalla sivupalstalla on linkkejä, kainalojuttuja, kuvia ja navigointilinkkejä. Sivuilla on vain vähän kuvia." [Koski 1999]
>
> "Tv-taittoa käytetään multimediasivuilla. Kuva dominoi sivua, jolla on sekä liikkuvaa kuvaa että stillkuvaa. Kuvat ja äänet eivät ole linkkien takana, vaan ne on sulautettu yhtenäiseksi kokonaisuudeksi." [Koski 1999]

5. Käytettävyys ja verkkoreportaasi

Nielsen [2003] määrittelee käytettävyyden laatukriteeriksi, jonka avulla voidaan arvioida, miten helppokäyttöisiä käyttöliittymät ovat. Termi käytettävyys viittaa myös suunnitteluprosessin aikaisiin, helppokäyttöisyyttä parantaviin metodeihin [Nielsen 2003].

Hyvä käytettävyys on verkkoreportaasille tärkeää, jotta käyttäjät ylipäätään pystyvät tutustumaan koko teokseen. Lisäksi helppokäyttöisyys lisää edellä mainittua immersiivisyyttä. Nielsenin [2003] mielestä hyvä käytettävyys on verkossa elinehto: jos verkkosivua tai teosta on vaikea käyttää, ihmiset lähtevät muualle.

5.1. Navigointirakenne

Navigointirakenteeseen pätevät kohdassa 3.3 mainitut Nielsenin [1993] listaamat käytettävyyttä parantavat ominaisuudet: opittavuus, tehokkuus, muistettavuus, virheettömyys ja miellyttävyys.

Multimediaesityksen käyttötapa ohjaa oleellisesti navigoinnin suunnittelua. Sovelluksessa kuitenkin liikutaan usein monilla toisistaan poikkeavilla tavoilla,

mikä hankaloittaa navigointirakenteen suunnittelua. Linkin tulisikin kertoa, mihin se vie. Painike, joka vie ainoastaan seuraavaan alavalikkoon ei ole kovin hyödyllinen eikä havainnollinen. Linkin takaa pitää löytyä todellista sisältöä eikä uusia painikkeita. [Kanerva *et al.* 1997]

5.2. Layout, taitto ja muu visuaalinen suunnittelu

Layoutin, taittotyylin ja esimerkiksi mahdollisten graafisten navigointielementtien suunnittelussa on tärkeää, että ne muodostavat yhtenäisen, selkeän kokonaisuuden. Hatva [1998] korostaa, että jo ensisilmäyksen pitäisi tuottaa jonkinlainen kuva siitä, mikä on tyylilaji ja miten olennaiset asiat löytyvät, ja kaikki tämä pitäisi tapahtua käyttäjälle mielihyvää tuottavalla tavalla.

Hatva [1998] myös listaa muutamia keinoja, joiden avulla käyttäjää voidaan auttaa löytämään oikea informaatio oikeassa paikassa:

- visuaalinen erottelu valaistuksen, värin ja tekstuurin avulla
- layout, joka on helppo ymmärtää ja muistaa
- mieleenpainuvat maamerkit
- merkit, symbolit, ohjeistot ja kartat
- hyvät oppaat. [Hatva 1998]

5.3. Mediaelementteihin liittyviä käytettävyysseikkoja

5.3.1. Teksti

Useimmin verkkotekstien käytettävyyteen yhdistetyt ominaisuudet ovat lyhyys ja tiiviys. Verkkotekstiä luetaan usein silmäillen, mikä on myös syytä huomioida tekstiä kirjoittaessa. Alasilta [2000] muistuttaa, että tietokoneen ruudulta on paperiin verrattuna hitaampa lukea, työläämpää ymmärtää, hankalaa hahmottaa kokonaisuuksia ja vaivalloista löytää uudelleen yksityiskohtia. Verkkotekstin pitää olla tiiviimpää ja kaikin puolin helppolukuisempaa kuin paperilta luettavaksi tarkoitettujen tekstien, eli tiivistäminen ei saa vaarantaa ymmärrettävyyttä, vaan tarvittaessa teksti on jäsenneltävä uudestaan [Alasilta 1998].

Alasillan [1998] mukaan linkityksen kannalta on välttämätöntä, että teksti koostuu sisällöllisesti ja juonellisesti itsenäisistä lohkoista. Hypertekstissä lukijan tai käyttäjän on saatava luettavakseen järkevä kokonaisuus, vaikka hän vain poikkeaisi jossain teoksen osassa [Alasilta 1998].

5.3.2. Kuva

Kuvallisen suunnittelun vaikeus piilee Hatvan [1998] mielestä päämäärien, esteettisyyden ja toimivuuden, ristiriidassa. Ristiriita nousee esiin esimerkiksi silloin, kun kuvan latautumisnopeus pakottaa vähentämään värien määrää tai pienentämään kokoa, vaikka esteettisyys ja vaikuttavuus siitä kärsisivätkin. Parhaimmillaan esteettisyys ja informatiivisuus kuitenkin ovat yhdistettävissä, jopa toistensa ehtoja. [Hatva 1998]

Hatva [1998] toteaa, että kuvan kykyä kertoa sanoja enemmän tulisi käyttää varsinkin verkkoviestinnässä, jossa pitkien tekstien lukeminen on hankalaa.

5.3.3. Ääni

Ääni on hyvin hallitseva elementti, joten sitä on syytä käyttää varovasti. Vahvat tehosteäänet ovat usein päälleliimatun tuntuisia ja vievät helposti huomion pois varsinaisesta asiasta. Ääni myös ärsyttää helposti, joten äänettömän vaihtoehton mahdollistaminen on suotavaa. [Kanerva *et al.* 1997]

Myös äänien laatuun on syytä kiinnittää huomiota. Häiriöinen ääni ärsyttää käyttäjää varmasti, samoin vaikkapa amatöörin epätasainen spiikkaus. Niin sanottujen palauteäänten (käyttöliittymän toimintoihin ja painikkeisiin liitettävä ääni) tulisi olla mahdollisimman lyhyitä ja neutraaleja, koska ne toistuvat usein [Kanerva *et al.* 1997].

5.3.4. Liikkuva kuva

Liikkuvan kuvan käyttöä verkossa on syytä miettiä huolellisesti ja videon käyttämisen tulee olla hyvin perusteltu. Videotiedostot ovat yleensä verkkoon liian kookkaita, ja jos tiedostokokoa pienennetään riittävästi, videosta tulee niin huonolaatuista, ettei siitä ole lisäarvoksi teokselle, saati hyötyä tai iloa käyttäjälle. Toiseksi liika liikkuvien elementtien käyttö luo häiritsevän levottoman tunnelman, eikä käyttäjän huomio kiinnity loppujen lopuksi oikein mihinkään.

Nielsen [1999] toteaa hieman sarkastisesti, että vain harvoin videota ei ole mahdollista korvata muutamalla stillkuvalla ja niiden kuvauksella, itse asiassa useimmissa tapauksissa lopputulos olisi stillkuvaversiona jopa parempi. Nielsen [1999] ehdottaa korvaavaksi vaihtoehdoksi esimerkiksi sarjakuvamaista kuvasarjaa tarvittavine teksteineen.

6. Yhteenveto

Verkkoreportaasi on hypertekstirakenteinen, tietoverkossa julkaistava hypermediateos. Multimediamuotoinen reportaasi pyrkii tarjoamaan käyttäjilleen paitsi informaatiota, myös elämyksiä. Verkkoreportaasille tyypillisiä ominaispiirteitä ovat multilineaarisuus, vu orovaikutteisuus, immersiivisyys ja tilallisuus.

Multimediaelementtien suunnittelussa on olennaista, että ne muodostavat yhtenäisen kokonaisuuden. Esimerkiksi tekstille on ominaista tiiviys, ja hypertekstirakenne puolestaan asettaa verkkoreportaasin tekstien kirjoittajan varsin haasteellisen tehtävän eteen.

Hyvä käytettävyys on verkkoreportaasille tärkeä laatukriteeri, jotta käyttäjät saataisiin ylipäätään tutustumaan teokseen. Graafisen ilmeen selkeys ja käyttöliittymän helppous parantavat myös teoksen immersiivisyyttä ja saavat siten käyttäjän upoutumaan kerronnan tilaan.

Kirjalliset lähteet

[Alasilta, 2000] Anja Alasilta, *Verkkoajan viestintä.* Kauppakaari, Pieksämäki, 2000.

[Alasilta, 1998] Anja Alasilta, *Näin kirjoitat tietoverkkoon*. Inforviestintä, Helsinki, 1998.

[Fornäs, 1999] Johan Fornäs, Digitaaliset rajaseudut. Identiteetti ja vuorovaikutteisuus kulttuurissa, mediassa ja viestinnässä. Teoksessa *Johdatus digitaaliseen kulttuuriin*, Aki Järvinen ja Ilkka Mäyrä (toim.), Vastapaino & Taide ja viestintä / Tampereen ammattikorkeakoulu, Tampere, 1999.

[Hatva, 1998] Anja Hatva, Kuvasuunnittelu verkossa. Teoksessa *Esteettinen ja toimiva verkkojulkaisun ulkoasu,* Anja Hatva (toim.), Edita, Helsinki, 1998.

[Kanerva *et al.* 1997] Jyrki Kanerva, Jukka Packalén ja Maarit Puttonen, *Ideasta Multimediaksi.* Edita, Helsinki, 1997.

[Koski, 1999] Nicklas Koski, *Verkkoreportaasin synty. Osa II: Taittajan näkökulma.* Journalistinen pro gradu –tutkielma, Tampereen yliopisto, 1999.

[Koskimaa, 1999] Raine Koskimaa, Digitaaliset tekstit ja kirjallisuus. Teoksessa *Johdatus digitaaliseen kulttuuriin*, Aki Järvinen ja Ilkka Mäyrä (toim.), Vastapaino & Taide ja viestintä / Tampereen ammattikorkeakoulu, Tampere, 1999.

[Kuusisto ja Pippuri, 1998] Päivi Kuusisto ja Mika Pippuri, *Verkkojulkaisun eväät*. Tampereen yliopisto, Tiedotusopin laitos, Julkaisuja sarja C 24 / 1998.

[Kämäräinen, 1999] Anni Kämäräinen, *Verkkoreportaasin synty. Kirjoittavan toimittajan näkökulma.* Journalistinen pro gradu –tutkielma, Tampereen yliopisto, 1999.

[Nielsen, 1995] Jakob Nielsen, *Multimedia and Hypertext: The Internet and Beyond*. AP Professional, Boston, 1995.

[Nielsen, 1993] Jakob Nielsen, *Hypertext and Hypermedia.* Academic Press, Boston, 1993.

[Pasula, 2002] Susanna Pasula, *Verkkodokumentin ilmaisukeinot: Kummitus dokumenttielokuvan, valokuvan ja hypermedian risteyksessä.* Journalistinen pro gradu -tutkielma, Tampereen yliopisto, 2002.

[Salen ja Zimmerman, 2004] Katie Salen ja Eric Zimmerman, *Rules of Play. Game Design Fundamentals.* MIT Press, Cambridge, 2004.

[Sinkkonen et al. 2002] Irmeli Sinkkonen, Hannu Kuoppala, Jarmo Parkkinen ja Raino Vastamäki, *Käytettävyyden psykologia.* Edita / IT Press, Helsinki, 2002.

[Tarkka, 1996] Minna Tarkka, Narratiivisuus ja vuorovaikutuksen tilat. Teoksessa *Johdatus uuteen mediaan*, Minna Tarkka, Kari A. Hintikka ja Asko Mäkelä (toim.), Edita, Helsinki, 1996.

Verkkolähteet (tarkistettu 7.5.2004):

[Nielsen, 2003] Jakob Nielsen, Usability 101. Alertbox, 2003, *http://www.useit.com/alertbox/20030825.hml*

[Nielsen, 1999] Jakob Nielsen, Video and Streaming Media. Alertbox, 1999, *http://www.useit.com/alertbox/990808.hml*

Kirjoituksessa mainittuja sivustoja (tarkistettu 7.5.2004):

Benetton Colors. Benetton-yhtiön verkkojulkaisu.
http://www.benetton.com/colors/

Elektronisen kuvajournalismin maisteriohjelma. Linkit verkkoreportaaseihin.
http://www.uta.fi/ekj/

Helsingin Sanomat. Verkkoliite. Klik-osaston arkisto. Linkit webortaaseihin.
http://www.helsinginsanomat.fi/klik/klikarkisto/

Tehtaan kuolema. Verkkoreportaasi.
http://www.nicklaskoski.com/tehdas/

ERP-järjestelmien implementoinnin kriittiset tekijät

Antto Seppälä

Tiivistelmä
Tutkielmassa käsitellään ERP-järjestelmän Implementointiprosessia ja sen kriittisiä menestystekijöit. Menestystekijöitä pohditaan johtamisen ja organisoinnin, henkilöstövoimavarojen, teknologian sekä valvonnan ja arvioinnin näkökulmista.

Avainsanat ja -sanonnat: enterprise resource planning (ERP), enterprise systems (ES), käyttöönotto
CR-luokat: H4.2, K4.3

1. Johdanto

Nopeat taloudelliset muutokset luovat organisaatioille yhä suurempia paineita saavuttaa mahdollisimman paljon tietoa mahdollisimman nopeasti. Eräs vastaus talouden muutoksiin on ERP-järjestelmä. Sen tavoite on tuottaa organisaation johdolle ja työntekijöille tarvittavat tiedot yrityksen liiketoimintaprosesseista.

ERP-järjestelmien hankitaprosessi on organisaation menestykselle kriittinen toimenpide. Onnistuneella projektilla voidaan saavuttaa huomattavia kustannussäästöjä ja kilpailuetua. Sillä voidaan tehostaa yrityksen toimintaa huomattavasti monella eri alueella. Epäonnistunut hankintaprojekti taas voi johtaa organisaation tuhon partaalle.

ERP-järjestelmän hankintaprosessi on yrityksen suurin yksittäinen tietojärjestelmä hankinta, joten sen kustannukset ovat suuret. Epäonnistunut implementaatioprosessi voi aiheuttaa muitakin ongelmia kuin vain järjestelmän hankintahinnan menetyksen.

Eräs tärkeimmistä tekijöistä ERP-järjestelmän hankinnassa on onnistunut implementointiprosessi. Siinä tavoitteena on saada järjestelmästä mahdollisimman hyvin organisaation liiketoimintaprosesseihin sopiva sekä saada työntekijät hyväksymään järjestelmä.

Tässä tutkielmassa tarkastelen ERP-järjestelmän implementointiprosessia, sen toimintoja sekä onnistuneisiin implementointeihin vaikuttaneita tekijöitä. Mikä

oikeastaan ERP-järjestelmä on, sekä esittelen sen implementointiprosessin ja siihen kuuluvat toimenpiteet. Lopuksi tarkastelen tekijöitä, jotka ovat kriittisiä implementointiprosessin onnistumisen kannalta.

2. Mikä on ERP-järjestelmä

2.1. Taustaa

ERP-järjestelmät, eli yrityksen kokonaisvaltaiset toiminnanohjausjärjestelmät, alkoivat ensimmäisen kerran tulla esiin 1990-luvulla. Ne ovat tällä hetkellä eräs tärkeimmistä nykyaikaisen yrityksen tietojärjestelmistä. ERP-järjestelmien kehitys alkoi alunperin MRP-järjestelmistä (material requirements planning) 1960-luvulla.

MRP-järjestelmät yrittivät ratkaista yrityksien materiaaleihin liittyviä kysymyksiä, kuten yrityksen varastojen mahdollisimman tehokasta hyödyntämistä, jolloin yrityksillä olisi mahdollisimman vähän rahaa sidottuna valmistusmateriaaleihin ja valmiisiin tuotteisiin. Tämän lisäksi ne yrittivät auttaa yrityksen hankintatoimia, jolloin mahdollisimman tarkasti pystyttäisiin määrittämään organisaation materiaalitarpeet.

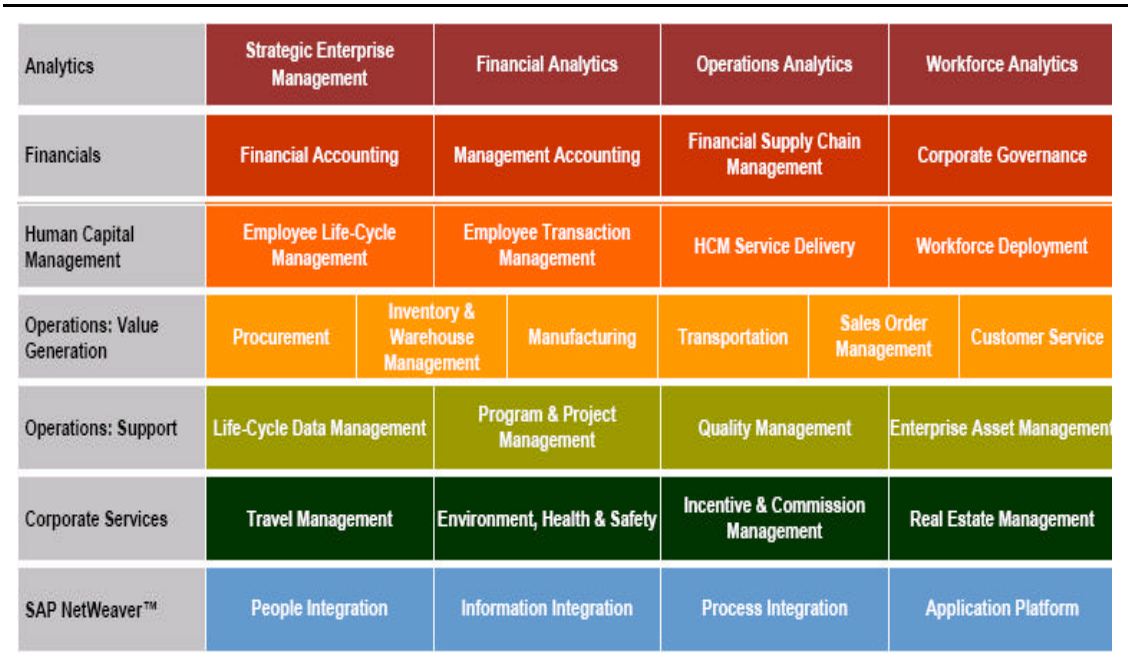
Seuraava kehitysaskel yrityksien kokonaisvaltaisissa tietojärjestelmissä oli MRP2.0 (manufacturing resource planning). MRP2.0-järjestelmiin tuli lisäominaisuuksina yrityksen tuotteiden valmistukseen liittyviä toimintoja, kuten pääoman ja työvoiman saatavuus valmistustoiminnassa. Nämä kaksi ERP-järjestelmien esiastetta olivat siis vasta eräänlaisia tuotantoon ja valmistamiseen liittyviä järjestelmiä.

Kokonaisvaltaiset toiminnanohjausjärjestelmät, eli ERP-järjestelmät, esiteltiin 1990-luvulla, kun MRP2.0-järjestelmiin lisättiin useita uusia osia, kuten talousohjelmistot sekä henkilöstöhallinta.

ERP tulee sanoista enterprise resource planning, suomennettuna tämä tarkoittaa yrityksen resurssien suunnittelua. Yleisesti ERP-järjestelmistä puhuttaessa suomeksi käytetty termi on toiminnanohjausjärjestelmä. Toiminnanohjausjärjestelmässä tavoitteena on koko yrityksen, tai jopa konsernin, toimintojen integroiminen sekä automatisointi. ERP-järjestelmät ovat erittäin pitkälle tuotteistettuja kokonaisratkaisuja.

Toiminnanohjausjärjestelmä koostuu erilaisista moduuleista. Näitä moduuleita ovat mm. varastonhallinta, henkilöstöhallinto sekä talousohjelmistot.

Kuvassa 1 eräs suurimmista ERP-järjestelmien toimittajista esittelee oman järjestelmänsä koostumuksen.



Kuva 1. SAP R/3 järjestelmän osat (http://www.sap.com)

Järjestelmien modulaarisuus mahdollistaa sen räätälöinnin kunkin organisaation omien tarpeiden mukaan. Yritykset voivat haluta vain tietyn osan tietyltä toimittajalta, jolloin ne voivat ostaa muut osat haluamaltaan toiselta toimittajalta. Muiden tekijöiden osien yhdisteleminen on mahdollista, mutta niiden yhteensovittaminen ja pienien osien ostaminen eri toimittajilta tekee tällaisista räätälöidystä ratkaisuista huomattavasti kalliimpia kuin valmiin paketin ostaminen.

2.2. ERP-järjestelmä käytännössä

Onnistunut ERP-projekti voi leikata organisaation kuluista ison osan pois, luoda yhä tarkempia tulevaisuuden ennusteita, nopeuttaa tuotteiden valmistusta sekä parantaa asiakaspalvelua [Umble *et al.*, 2003].

Toiminnanohjausjärjestelmät tuovat yrityksen johdolle ja muille työntekijöille monia erilaisia raportteja organisaation nykytilasta sekä tulevaisuudesta. Näitä raportteja ovat esimerkiksi varastojen tila, erilaiset myyntiraportit sekä erinäiset tilastot.

ERP-järjestelmien mahdollisuudet eivät ole ainoastaan raportoinnissa tai erilaisten tilastojen luomisessa, vaan niillä on mahdollisuus automatisoida yrityksen toimintoja. Esimerkiksi kun asiakas jättää tilauksen organisaation tilausjärjestelmään, niin tämä käynnistää monimutkaisen prosessin. Tilaus kirjautuu järjestelmään, jolloin se voi tarkistaa, löytyykö varastosta tarvittavat raaka-aineet. Mikäli ei löydy, se ilmoittaa yrityksen hankinnoista vastaavalle osastolle mitä tarvitaan. Samalla järjestelmä voi jo aloittaa työn valmistusprosessin.

Toiminnanohjausjärjestelmän käsittäessä koko organisaation se mahdollistaa toimintojen keskittämisen. Koska kaikki yrityksen osat on sulautettu toisiinsa, niin esimerkiksi hankintatoimet voidaan keskittää yhteen toimipisteeseen.

2.3. ERP-järjestelmän tavoitteet

Nykyään melkeinpä jokaisella suurella yrityksellä on jo käytössä toiminnanohjausjärjestelmä, tai ainakin ainakin hankintaprosessi on käynnissä. Miksi yritykset panostavat näihin kalliisiin järjestelmiin, joiden hankintaprosessit ovat erittäin riskialttiita

Yritykset ovat etsineet toiminnanohjausjärjestelmistä pitkäjännitteisempää, tehokkaampaa, ja joustavampaa ratkaisua tietojärjestelmäarkkitehtuureja koskeviin haasteisiinsa [Luomala *et al.*, 2001]. ERP-järjestelmien perimmäinen tavoite on tarjota yritykselle väline, jolla sillä on mahdollisuus kontrolloida organisaation tavara-, raha- ja tietovirtoja. Koska toiminnanohjausjärjestelmä toimii läpi yrityksen ja sen toimintojen, niin tämä mahdollistaa organisaation erilaisten virtojen kontrolloinnin.

Toiminnanohjausjärjestelmän avulla organisaation työntekijöiden on mahdollista saada tietoa kaikkialta yrityksestä. Järjestelmä pystyy tuottamaan ja jakamaan tietoa reaaliaikaisesti, jolloin organisaation henkilöstön on helpompi tehdä oikeita päätöksiä nopeammin. Organisaation tietämyksen ja hyviksi havaittujen työtapojen jakaminen helpottuu toiminnanohjausjärjestelmän avulla.

Samalla kun yrityksen toiminnot integroituvat, niin se synnyttää riskin siitä, että yrityksen eri toiminnot ovat liian riippuvaisia toisistaan. Tämä aiheuttaa sen että häiriöt yhdessä toiminnossa voi aiheuttaa entistä herkemmin ongelmia muihinkin toimintoihin [Jansson *et al.*, 2001]. Yksikin inhimillinen erehdys voi kostautua siis useassa eri paikassa järjestelmän tehokkuuden vuoksi.

Organisaatiot tavoittelevat toiminnanohjausjärjestelmillä liiketoimintaprosessien optimointia. Järjestelmien avulla on mahdollista tehostaa lähes kaikkia

toimintoja pitkin toimitusketjua. Esimerkiksi tilaus voi vaikkapa käynnistää monimutkaisia prosesseja automaattisesti liittyen toimitukseen, varastointiin tai valmistukseen.

Toiminnanohjausjärjestelmien avulla organisaation johto yrittää saavuttaa kilpailuetua muihin nähden. Keinoina ovat mm. johdon erilaiset raportointi-työkalut, joilla on mahdollista saada erilaisia raportteja ympäri yritystä. Nämä raportit voivat esimerkiksi olla tämänhetkinen varastotilanne, reaaliaikainen myynninseuranta tai erilaiset markkina-alue tilastot. Näiden tietojen avulla organisaation johdon on mahdollista reagoida muuttuviin tilanteisiin nopeasti.

3. Implementointi prosessi

3.1. Erilaisia näkemyksiä implementointiprosessista

Implementointiprojekti on ehkäpä haastavin osa ERP-järjestelmä hankintaa. Yleensä kirjallisuudessa esiintyy kahta eri ajatusmaailmaa edustavaa tapaa implementoida ERP-järjestelmä. Nämä tavat ovat big bang -teoria, jossa organisaatio ottaa järjestelmän käyttöönsä kokonaan kerrallaan. Toinen tapa on vaiheittainen käyttöönotto, jossa yritys ottaan järjestelmän käyttöönsä vaiheittain, joko pieni organisaation osa kerrallaan tai sitten ainoastaan muutama moduuli kerrallaan.

Kumar *et al.* [2003] jakaa artikkelissaan implementointiprosessin kahteen osaan. Ensimmäinen osa on project phase, eli eräänlainen projektivaihe. Tässä vaiheessa yrityksen tavoite on saada toiminnanohjausjärjestelmä käyntiin, ainakin yhdessä tai useammassa organisaation osassa. Toinen vaihe on shakedown, eli aikakausi käyttöönoton jälkeen ennen järjestelmän normaalin käyttötilanteen saavuttamista.

Implementointiprosessi voidaan myös jakaa kolmeen eri vaiheeseen. Nämä vaiheet ovat pilottivaiheen suunnittelu ja valmistelu, pilottivaiheen menestyksen arviointi ja ongelmat sekä loppujärjestelmän käyttöönotto [Hanseth *et al.*, 2001]. Tämän jaottelun ongelma on, että siinä ei oteta ollenkaan huomioon käyttöönoton jälkeistä aikaa, joka kuitenkin on tärkeää organisaatiolle projektin onnistumisen kannalta.

Somers ja Nelson [2004] jakavat implementointiprosessin kuuteen osaan. Nämä osat ovat laukaisu, omaksuminen, sovittaminen, hyväksyntä, rutinoituminen ja infuusio. Neljä ensimmäistä liittyvät ennen varsinaista käyttöönottoa tapahtuviin asioihin ja jälkimmäiset liittyvät sen jälkeiseen aikaan.

Tässä artikkelissa jaan implementointiprosessin kolmeen vaiheeseen. Nämä vaiheet ovat ennen käyttöönottoa tapahtuvat asiat, käyttöönotto sekä jälkitoimet.

3.2. Ennen käyttöönottoa huomioitavat asiat

Projektin alussa organisaation täytyy valita osaavat projektihenkilöt yrityksen sisältä. Projektiryhmän tulisi sisältää henkilöitä monilta eri alueilta yrityksen toiminnoista, ei pelkästään osaavia tietotekniikkahenkilöitä. Samalla organisaation johdon täytyy valita implementointipartnerinsa, eli esimerkiksi konsulttiyritys sekä luoda yhteydet järjestelmän toimittajaan [Kumar *et al.*, 2003].

Projektiryhmän ja sen johtajan valinnan jälkeen rymän tulisi luoda projektisuunnitelma, sekä määritellä tehtävän laajuus ja asettaa tarkat välitavoitteet ja aikarajat projektille [Nah *et al.*, 2001; Kumar *et al.*, 2003]. Tämän jälkeen projektiryhmän pitäisi määritellä tarkoin organisaation eri osien tarpeet ja vaatimukset, jotta tiedettäisiin mitä käyttäjät todella järjestelmästä tarvitsevat ja mihin olisi syytä paneutua erityisellä huolella.

Projektiryhmän pitäisi analysoida yrityksen toiminnot ja tarpeet mahdollisimman tarkkaan ja luoda samalla organisaation muutossuunnitelma ja henkilöstön koulutussuunnitelma [Mabert *et al.*, 2003]. Näitä suunnitelmia tulisi päivittää jatkuvasti projektin edetessä.

Koska paketoidut ERP-järjestelmät ovat erittäin geneerisiä ja niiden on tarkoitus sopia monenlaisiin organisaatioihin, niin yrityksen prosessit eivät välttämättä täysin kohtaa ERP-järjestelmän prosesseja. Organisaation tulisi tarkoin verrata yrityksen liiketoimintaprosesseja sekä ERP-järjestelmässä olevia prosesseja. Mikäli esiintyy merkittäviä eroavuuksia prosesseissa, niin yrityksen tulee aloittaa liiketoimintaprosessien uudelleenjärjestely (business process reengineering).

Liiketoimintaprosessien uudelleenjärjestelyssä muokataan yrityksen prosesseista järjestelmään sopivia. Samalla yritetään luoda yhdenmukaiset prosessit läpi organisaation [Hanseth *et al.*, 2001]. Yrityksen muokatessa prosessejaan, tulisi sen myös aloittaa työntekijöiden koulutus uusien työtapojen ja työvälineiden käyttöä varten [Somers and Nelson, 2004].

Ennen varsinaista käyttöönottoa organisaation tulisi myös tarkastella tietotekniikka infrastruktuuriaan ja sen kykyä toimia rakennuspohjana uudelle järjestelmälle. Mikäli yrityksen infrastruktuuri on vanhentunut tai muuten sopimaton tulevalle järjestelmälle, niin sen päivitysprojekti tulee aloittaa heti {Kumar *et al.*, 2003].

Projektiryhmän pitäisi määritellä mitkä organisaation vanhoista tietojärjestelmistä jää vielä ERP-järjestelmän käyttöönoton jälkeen toimintaan. Näiden vanhojen järjestelmien rajapinnat tulisi tutkia ja määritellä, jotta niiden integroiminen uuteen järjestelmään onnistuu mahdollisimman hyvin [Luomala *et al.*, 2001].

Järjestelmien integroimista toisiinsa voidaan yrittää helpottaa erilaisilla ”middleware” ohjelmistoilla. Nämä ohjelmistot keskustelevat ja välittävät tietoja järjestelmien kesken. Tosin niidenkin käytössä on omat riskinsä, esimerkiksi niissä esiintyvät ohjelmistovirheet voivat vaikuttaa suurestikin järjestelmien toimintaan.

3.3. Käyttöönotto

Käyttöönoton lähentyessä yrityksen tulee panostaa yhä enemmän käyttäjien koulutukseen. Käyttäjien koulutus on yksi parhaista keinoista vähentää uutta järjestelmää kohtaavaa muutosvastarintaa. Kun työntekijät osaavat käyttää järjestelmää ja ymmärtävät sen tarkoituksen, niin järjestelmän vastustaminen vähenee.

Vielä ennen varsinaista käyttöönottoa täytyy varmistaa, että organisaation eri osat ovat linkitetty toisiinsa ja organisaatio on valmis yhteistyön kasvamiseen ja tiedonkulun paranemiseen [Hanseth, 2001].

Vähän ennen määrättyä käyttöönottopäivämäärää täytyy uusi järjestelmä vielä testata ja varmistaa, että sen kaikki osat toimivat niinkuin on suunniteltu [Kumar *et al.*, 2003]. Samalla täytyy varmistaa, että ERP-järjestelmän integrointi muihin organisaatioiden tietojärjestelmiin on kunnossa, ettei tule ikäviä yllätyksiä käyttöönoton jälkeen.

Erittäin tärkeää on vanhojen järjestelmien sisältämän tiedon kääntäminen uudelle järjestelmälle. Samalla on mahdollisuus muokata monimutkaisia tietokokonaisuuksia yhä ymmärrettävämpään ja helpommin käytettävään muotoon. Tiedon syöttämisen valvonta ja johtaminen on yksi kriittisimmistä tekijöistä implementointiprosessissa [Somers and Nelson, 2004].

Vanhojen järjestelmien tietojen siirtämisen jälkeen voidaan vihdoin ottaa uusi järjestelmä käyttöön, ja samalla voidaan voidaan valmistautua vanhojen järjestelmien alasajoon.

Vihdoin organisaation pitäisi olla valmis ottamaan ERP-järjestelmä käyttöönsä. Käyttöönoton hetkellä yrityksen pitää vielä luoda ja ottaa käyttöönsä

erilaiset valvontatyökalut sekä palautejärjestelmät, joilla käyttäjät voivat antaa palautetta ja kehitysehdotuksia uudesta järjestelmästä.

3.4. Jälkitoimet

ERP-järjestelmän käyttöönottoon ei implementointiprosessi vielä pääty. Yrityksen tulee panostaa myös jälkitoimiin, joilla pyritään saavuttamaan organisaation normaalitila ja järjestelmän stabilointi. Jälkitoimissa organisaation tulisi varata resursseja järjestelmän kehittämiseen ja ylläpitämiseen läpi organisaation [Hanseth *et al.*, 2001].

Järjestelmän käyttöönoton jälkeen tulisi järjestää keskustelutilaisuuksia, joissa käyttäjät voivat keskenään ja ylläpidon kanssa keskustella kohtaamistaan ongelmista sekä kehitysideoista.

ERP-järjestelmän kehittyessä ja laajentuessa ajan myötä on tärkeää, että käyttäjien koulutus pysyy ajan tasalla. Se on tärkeää, jotta vanhat käyttäjät ja uudet taloon tulevat työntekijät osaavat hyödyntää järjestelmää maksimaalisesti. Järjestämällä lisäkoulutusta voidaan päästä eroon käyttäjien tavoista kehittää järjestelmää kiertäviä työtapoja [Umble *et al.*, 2003].

Ennalta sovitun ajan kuluttua organisaation tulisi järjestää jälkitarkastus projektin onnistumisesta. Tämän jälkitarkastuksen tavoite on selvittää, mitkä alueet kaipaavat lisäselvityksiä, jotta järjestelmän käyttö tehostuisi vieläkin enemmän. Tämä selvitys voidaan toteuttaa esimerkiksi käyttäjähaastatteluilla [Umble *et al.*, 2003].

Kaikki projektin vaiheet, toimenpiteet ja muokkaukset järjestelmään tulisi dokumentoida, jotta päivitystilanteissa osaittaisiin paremmin varautua ongelmiin. Projektin onnistuminen täytyy arvioida tarkasti, jotta voidaan oppia tehdyistä virheistä seuraavia tietojärjestelmäprojekteja varten. Arviossa tulisi selvittää mm. saavutettiinko tavoitteet, olivatko ne realistisia, osattiinko varautua yllättäviin asioihin, miten projektin arvioinneissa onnistuttiin sekä muut tekijät joiden ajatellaan olevan merkittäviä [Mandal and Gunasekaran, 2003].

Vaikka järjestelmä on jo käytössä, sen testausta tulisi silti jatkaa. Myös erilaisten ohjelmistovirheiden ja muiden ongelmien etsimistä tulee jatkaa, ja korjata sitä mukaa kuin niitä esiin tulee [Kumar *et al.*, 2003]. Suurena apuna ongelmien löytämisessä on jatkuva kommunikointi käyttäjien kanssa.

Willis ja Willis-Brown [2002] esittävät artikkelissaan kolmiosaisen mallin mitä käyttöönoton jälkeen tulisi tehdä. Vaiheet ovat ERP-järjestelmän vakauttaminen,

lisätä toiminnallisuutta ja tarvittavien prosessien uudelleenjärjestely sekä laajentaminen ja integrointi.

ERP-järjestelmän vakauttamisessa organisaation täytyy arvioida uusi järjestelmä ja yrittää löytää suurimmat ongelmatekijät, sekä poistaa järjestelmästä epäjohdonmukaisuudet sekä vajavaisuuksien paikkaaminen. Vakauttaminen vaatii myös kaikkien käyttäjien koulutusta.

Toisessa vaiheessa organisaation tulisi lisätä järjestelmän toiminnallisuutta sekä tarvittaessa liiketoimintaprosessien uudelleenjärjestelmistä. Konsulteista on tässä vaiheessa apua, kun yritetään saada järjestelmästä kaikki irti. ERP-järjestelmä on organisaation tietojärjestelmien selkäranka ja kun sen mahdollisuudet on ymmärretty, niin tulee siirtyä kolmanteen vaiheeseen.

Kolmannessa vaiheessa organisaation tulisi hankkia järjestelmään lisää erilaisia ohjelmistoja, joista voisi olla hyötyä. Tärkeää näiden lisäosien hankinnassa on se, että niiden integrointi ERP-järjestelmään olisi mahdollisimman helppoa. Laajennuksilla voidaan vahvistaa niitä osia, mitkä ovat ERP-järjestelmän heikkouksia, kuten esimerkiksi asiakkuuksien hallinta tai myyntiprosessin automatisointi.

4. Kriittiset menestystekijät

4.1. Johtaminen ja organisointi

Tärkein selittäjä ERP-projektin onnistumiselle on huolellinen suunnittelu [Mabert *et al.* 2003]. Suurimmat ongelmat ERP-järjestelmän implementoinnissa johtuvat johtamisesta tai oikeastaan sen puutteesta. Implementointiprojektiin lähdettäessä onkin tärkeää valita projektin johtoon henkilö, jolla on valtaa päättää asioista, sekä projektin suhteen että organisaation yleisten linjojen suhteen [Nah *et al.*, 2001].

Implementointiprosessiin lähdettäessä projektilla on oltava ylimmän johdon tuki. Ylimmän johdon osallistuminen projektiin luo tarvittavan ilmapiirin yritykseen. Johdon on lähettävä tuestaan selviä signaaleja ympäri organisaatiota, jotta henkilöstö lähtisi mahdollisimman tosissaan projektiin [Bradford and Florin, 2003].

Projektin alussa yrityksen johdon ja projektiryhmän täytyy luoda selkeät tavoitteet ja suunnitelmat niiden tavoittamiseksi. Lisäksi projektille täytyy antaa alusta saakka tarpeeksi resursseja tavoitteiden saavuttamiseksi, eli työntekijöitä ja rahaa täytyy sijoittaa riittävästi [Somers and Nelson, 2004].

Organisaation pitäisi rakentaa heti implementointiprojektin alussa selkeä liiketoimintasuunnitelma, jotta liiketoiminnan tavoitteet pysyvät projektin edetessä mukana. Suunnitelmassa tulisi olla järjestelmällä saavutettavat strategiset ja konkreettiset hyödyt, kustannukset, riskit sekä aikataulu [Nah *et al.*, 2001].

Tärkeää projektin onnistumisen kannalta on myös se, että yritys onnistuu havaitsemaan ERP-järjestelmän ja liiketoimintaprosessien erot [Hong and Kim, 2002]. ERP-järjestelmät ovat erittäin geneerisiä ja niillä tavoitellaan mahdollisimman hyvää sopivuutta mahdollisimman monen organisaation prosesseihin. Näiden prosessien eroja voidaan vähentää liiketoimintaprosessien uudelleenjärjestelyllä (business process reengineering) ja sen onnistumista pidetäänkin yleisesti yhtenä tärkeimpänä tekijänä projektin onnistumisen kannalta [Bradford and Florin, 2003; Umble *et al.*, 2003; Somers and Nelson, 2004]. Uudelleenjärjestelyssä yritetään muokata organisaation prosesseista mahdollisimman hyvin järjestelmään sopivia. Uudelleenjärjestelyn tulee koskea organisaation kaikkia toimintoja, sekä työtapoja.

Kaikki muutos aiheuttaa ihmisissä vastarintaa. Tämä pätee myös ERP-järjestelmän implementointiprosessissa. Juuri sen takia organisaation muutosjohtajuuden puute voikin aiheuttaa ongelmia. Projektiryhmän tuleekin tehdä muutosstrategia, ja koko organisaation, myös ylimmän johdon, tulee sitoutua tähän muutokseen [Motwani *et al.*, 2002].

Muutosvastarintaa voi työntekijöissä aiheuttaa se, että he eivät ymmärrä ERP-järjestelmän merkitystä organisaatiolle, eivätkä hahmota sen työtä helpottavia ominaisuuksia sekä sen luomia mahdollisuuksia. Organisaation ei tulisikaan lannistua käyttäjien vastustuksesta vaan jatkaa muutosprosessia ja yrittää ”myydä” järjestelmä ja sen mahdollisuudet käyttäjille [Al-Mashari and Al-Mudimigh, 2003].

Järjestelmän myyntiä käyttäjille auttaa avoin kommunikointi läpi organisaation. Työntekijöille pitäisi selkeästi ilmaista minkätakia järjestelmään ollaan siirtymässä sekä miten projekti etenee. Mikäli kommunikointi ja tiedon jakaminen on avointa projektin alusta lähtien, niin tällä on rauhoittava vaikutus työntekijöiden keskuudessa [Al-Mashari and Al-Mudimigh, 2003].

Vaikka organisaatio palkkaa avukseen erilaisia konsultteja, niin yrityksen johdon pitää silti uskaltaa tehdä päätöksiä, eikä paeta vastuuta ja antaa konsulttien päättää kaikesta.

4.2. Henkilöstövoimavarat

ERP-järjestelmän implementoinnin kannalta on tärkeää, että organisaatio onnistuu rekrytoimaan osaavia henkilöitä sekä onnistuu pitämään vanhat osaavat työntekijät. Tärkeää on, että henkilöstö on omalla aluellaan pätevää, koulutuksella voidaan kyllä paikata teknisten kykyjen puutetta [Willis and Willis-Brown, 2002]. Mikäli projektia ei saada myytyä henkilöstölle ja he kokevat sen uhaksi omalle työlleen, niin yhtiö voi kohdata tilanteen jossa pätevin henkilöstö karkaa muualle.

Organisaation tulee luoda implementointiprojektiryhmä, jonka johtoon se valitsee kokeneen projektihenkilön, joka on erikoistunut projektityöskentelyyn. Projektiryhmään tulisi valita huippuosaajia yrityksen eri alueilta, ja sille pitäisi antaa valtaa tehdä tärkeitäkin päätöksiä [Umble *et al.*, 2003]. On tärkeää, että ryhmässä on myös liiketoimintaosaajia, jotka ymmärtävät liiketoiminnan tarpeet [Barker and Frolick, 2003]. Projektiryhmän sisällä täytyy olla selkeä työnjako ja projektin jäsenille täytyy määrätä selkeät tehtäväalueet [Amoako-Gyampah and Salam, 2003].

Tärkeä kysymys projektin onnistumisen kannalta on, miten koulutus tulisi järjestää. Ostetaanko se kokonaisuutena ulkopuoliselta yritykseltä, vai koulutetaanko omasta henkilökunnasta järjestelmän pääkäyttäjiä, jotka sitten kouluttavat ja neuvovat muuta henkilökuntaa järjestelmän käytöstä. On esitetty että jos koulutukseen käytettävät kustannukset ovat projektin kokonaiskustannuksista 10-15%, niin projektilla on 80% mahdollisuus onnistua [Umble *et al.*, 2003].

Kumar *et al.* [2003] esittävät tutkimuksessaan ajatuksen, että koulutuksen ei pitäisi koskea vain järjestelmän käyttämistä, vaan käyttäjille pitäisi kertoa miksi järjestelmää käytetään ja miksi erilaisia uusia työtapoja on otettu käyttöön. Organisaation tulisi varautua myös koulutussuunnitelmassaan siihen, että järjestelmä muuttuu sekä kesken projektin että sen jälkeen, joten koulutuksen pitää pysyä järjestelmän muutoksissa mukana. Koulutuksella voidaan myös ehkäistä käyttäjien osaamattomuudesta johtuvat tavat kiertää järjestelmän käyttö työssään [Umble *et al.*, 2003].

Kommunikaatio on yksi tärkeimmistä tekijöistä onnistuneissa implementointiprojekteissa. Organisaation tuleekin mahdollistaa eri osastojen välisen kommunikoinnin, jotta tarvittava tieto saadaan kulkemaan läpi organisaation [Somers and Nelson, 2004]. Kommunikaation katsotaankin olevan yleensä yksi tärkeimmistä tekijöistä käyttäjien hyväksynnän saavuttamiselle.

Implementointisuunnitelman valmistuttua yrityksen tulee kertoa siitä työntekijöilleen, asiakkailleen sekä kumppaneilleen. Ei pelkästään sen olemassaolosta ja sisällöstä, vaan myös sen kehityksestä ja onnistumisesta pitkin projektia [Motwani *et al.*, 2002]. Kommunikoinnilla voidaan saada palautetta, jolla voidaan kehittää järjestelmää yhä paremmaksi. Yrityksen tuleekin herättää työntekijöiden mielenkiinto järjestelmää kohtaan ja yrittää saada heidät osallistumaan projektiin mahdollisimman voimakkaasti [Barker and Frolick, 2003}.

4.3. Teknologiset tekijät

Melkeinpä yhtä tärkeää uuden järjestelmän käyttöönoton kanssa on korvattavien järjestelmien hallittu alasajo. Rinnakkaisia järjestelmiä ei saa jättää käyntiin, koska kuitenkin osa työntekijöistä jatkaisi vanhojen ja tuttujen järjestelmien käyttöä [Umble *et al.*, 2003]. Tämä aiheuttaisi sen, että tieto jakautuisi ja hukkuisi useisiin eri järjestelmiin. Vanhojen järjestelmien alasajossa on myös tärkeää, että saadaan kaikki tarpeellinen tieto siirrettyä uuteen järjestelmään sekä luotua sille järkevä rakenne.

Siirretyn tiedon pitää olla tarkkaa ja oikeata, sillä väärän tiedon kirjaamisella voi olla kohtalokkaat seuraukset läpi organisaation. Väärän tiedon kirjaamisen ongelma ei ole ainoastaan vanhan tiedon siirrossa, vaan käyttäjiä täytyy kouluttaa ymmärtämään oikean ja validin tiedon merkitys järjestelmän käytölle [Umble *et al.*, 2003]. Käyttäjille tarvitsee tehdä jo projektin alkuvaiheessa selväksi, että kun uusi järjestelmä on käytössä niin vanhan käyttöä ei hyväksytä, jotta he voivat asennoitua jo varhain muutokseen.

ERP-järjestelmän implementointi ei voi onnistua, mikäli organisaation teknologiainfrastruktuuri ei ole kunnossa [Mabert *et al.*, 2003]. Samassa tutkimuksessa tuli myös esille, että mitä vähemmän muutoksia varsinaisen järjestelmän lähdekoodiin tehtiin, sitä paremmin implementoinnissa onnistuttiin. Mikäli kuitenkin koodia lähdetään muokkaamaan niin yrityksen tulee varautua kasvaneisiin kustannuksiin, projektin pituuden kasvuun sekä jatkokehittelyn vaikeutumiseen.

4.4. Valvonta ja arviointi

Projektin onnistumisen kannalta on tärkeää, että organisaation johto luo selkeät tavoitteet projektille sekä kunnolliset mittarit onnistumisen mittaamiseksi [Al-Mashari and Al-Mudimigh, 2003]. Erilaisilla mittareilla voidaan saada selville, missä järjestelmän osissa on vielä kehitettävää. Projektin kestäessä

mittareilla voidaan arvioida, miten hyvin ollaan onnistumassa ja voidaan tarvittaessa reagoida näihin tuloksiin tarpeeksi ajoissa.

Huolella rakennetuilla mittareilla voidaan nähdä, missä projektin osissa kaivataan lisää resursseja tai henkilöstön vaihdoksia [Umble *et al.*, 2003]. Organisaation ei tulisi ainoastaan puuttua epäonnistumiseen, vaan lähes yhtä tärkeätä on palkita onnistumiset, jotta henkilökunnan ja projektiryhmän motivaatio pysyy korkealla ja voidaa taata innostunut ilmapiiri. Järjestelmän mittaamista ei tulisi lopettaa käyttöönoton jälkeen vaan sen pitää jatkua läpi järjestelmän elinkaaren [Umble *et al.*, 2003].

Mittareita ei pitäisi rakentaa pelkästään teknologisten seikkojen tai budjetissa pysymisen kannalta, vaan niissä täytyy ottaa myös huomioon liiketoiminnallinen menestys, eli saavutetaanko järjestelmällä todellista hyötyä verrattuna vanhoihin järjestelmiin sekä sijoitettuun pääomaan [Mabert *et al.*, 2003].

Organisaatioiden tulisi ymmärtää, että projekti ei lopu käyttöönottoon vaan sen tulee jatkua koko järjestelmän elinkaaren ajan [Willis and Willis-Brown, 2002]. Järjestelmää tulee laajentaa ja sen toiminnallisuutta pitää kehittää yhä paremmaksi. ERP-järjestelmän tulee olla yrityksen vakaa selkäranka, jonka varaan voidaan uusia ominaisuuksia rakentaa.

5. Yhteenveto

Tutkimuksessani olen löytänyt viisi mielestäni kriittisintä tekijää, jotka ratkaisevat implementointiprosessin menestyksen. Näitä ovat läpi organisaation kulkeva kommunikaatio, onnistunut käyttäjien koulutus, joka ei pääty käyttöönottoon vaan jatkuu pidemmälle, laaja-alainen suunnittelu läpi projektin, organisaation valmius muutokseen ja johdon muutosjohtajuus sekä liiketoimintaprosessien uudelleenjärjestelyn onnistuminen. Näiden asioiden hallinnalla voidaan parantaa projektin onnistumisen todennäköisyyksiä.

Organisaatioiden tulisi oppia niin ERP-järjestelmien implementointiprosessista kuin myös muidenkin tietojärjestelmien kohdalla, että projekti ei lopu käyttöönottoon vaan kehitystyötä on jatkettava sen jälkeenkin.

Viiteluettelo

[Al-Mashari and Al-Mudimigh, 2003} Majed Al-Mashari and Abdullah Al-Mudimigh, ERP implementation: lessons from a case study, *Information Technology and People* **16,** 1 (2003), 21-33.

[Amoako-Gyampah and Salam, 2003] Kwasi Amoako-Gyampah and A.F. Salam, An extension of the technology acceptance model in and ERP implementation environment, *Information & Management* (accepted 10th august 2003).

[Barker and Frolick, 2003] Traci Barker and Mark N. Frolick, ERP implementation failure: a case study, *Information Systems Management* **Fall** (2003), 43-49.

[Bradford and Florin, 2003] Marianne Bradford and Juan Florin, Examining the role of innovation diffusion factors on the implementation success of enterprise resource planning systems, *International Journal of Accounting Information Systems* **4** (2003), 205-225.

[Hanseth et al., 2001] Ole Hanseth, Claudio U. Ciborra and Kristin Braa, The control devolution: ERP and side effects of globalization, *The DATA BASE for Advances in Information Systems* **32**, 4 (Fall 2001), 34-46.

[Hong and Kim, 2002] Kyung-Kwon Hong and Young-Gul Kim, The critical success factors for ERP implementation: an organizational fit perspective, *Information & Management* **40** (2002), 25-40.

[Jansson et al., 2001] Kim Jansson, Iris Karvonen, Veli-Pekka Mattila, Juha Nurmilaakso, Martin Ollus, Iiro Salkari, Jyrki Ala-Yrkkö ja Pekka Ylä-anttila, Uuden tietotekniikan vaikutus liiketoimintaan, *Teknologiakatsaus* **111**, (2001).

[Kumar et al., 2003] Vinod Kumar, Bharat Maheshwari and Uma Kumar, An investigation of critical management issues in ERP implementation: emperical evidence from Canadian organizations, *Technovation* **23** (2003), 793-807.

[Luomala et al., 2001] Juha Luomala, Juha Heikkinen, Karri Virkajärvi, Jukka Heikkilä, Anne Karjalainen, Anri Kivimäki, Timo Käkölä, Outi Uusitalo ja Hannu Lähdevaara, Digitaalinen verkostotalous: tietotekniikan mahdollisuudet liiketoiminnan kehittämisessä, *Teknologiakatsaus* **110**, (2001).

[Mabert et al., 2003] Vincent A. Mabert, Ashok Soni and M.A. Venkataramanan, Enterprise resource planning: managing the implementation process, *European Journal of Operational Research* **146** (2003), 302-314.

[Mandal and Gunasekaran, 2003] Purnendu Mandal and A. Gunasekaran, Issues in implementing ERP: a case study, *European Journal of Operational Research* **146** (2003), 274-283.

[Motwani et al., 2002] Jaideep Motwani, Dinesh Mirchandani, Manu Madan and A. Gunasekaran, Successful implementation of ERP projects: evidence from two case studies, *Int. J. Production Economics* **75** (2002), 83-96.

[Nah et al., 2001] Fiona Fui-Hoon Nah, Janet Lee-Shang Lau and Jinghua Kuang, Critical factors for successful implementation of enterprise systems, *Business Process Management Journal* **7,** 3 (2001), 285-296.

[Sia et al., 2002] Siew Kien Sia, May Tang, Christina Soh and Wai Fong Boh, Enterprise resource planning (ERP) systems as a technology of power: empowerment or panoptic control?, *The DATA BASE for Advances in Information Systems* **33**, 1 (Winter 2002) 23-37.

[Somers and Nelson, 2004] Toni M. Somers and Klara G. Nelson, A taxonomy of players and activities across the ERP project life cycle, *Information & Management* **41** (2004), 257-278.

[Umble et al., 2003) Elisabeth J. Umble, Ronald R. Haft and M. Michael Umble, Enterprise resource planning: Implementation procedures and critical success factors, *European Journal of Operational Research* **146** (2003), 241-257.

[Willis and Willis-Brown, 2002] T. Hillman Willis and Ann Hillary Willis-Brown, Extending the value of ERP, *Industrial Management & Data Systems* **102/1** (2002) 35-38.

Agent-based framework to support the development of psychophysiologically interactive systems

Toni Vanhala

Abstract.

Psychophysiological function is a bodily indication of psychological activity. There is a wealth of cognitive, affective, social, and clinical processes that have effects on physiological measures. Understanding and recognizing these phenomena automatically would support interaction between humans and machines. Novel applications for these systems in the field of Human-Computer Interaction include hands-free control of a vehicle or computer, monitoring applications, usability evaluation and affective computing. The psychophysiological signals and their analysis pose many problems that are common to this class of systems. A general solution to those problems would support the construction of prototypes and complete systems. This work presents a step towards the solution in the form of a software agent framework.

Keywords: human-computer interaction, architectures, software frameworks, psychophysiology, biosignal processing.
CR-classification: D.2.11, J.4

1. Introduction

Psychophysiology views the mind as having a physical substrate. The psychophysiological methods can provide information about a plethora of cognitive, affective, social, and clinical processes. Instead of focusing exclusively on physiological phenomena, psychophysiology inspects physical measures in conjunction with verbal, behavioral and contextual data. The different types and levels of data promote interdisciplinary research and multiple levels of analysis. [Cacioppo *et al.*, 2000]

The psychophysiological methods and results are valuable to human-computer interaction as well. Hands-free operation of computers and other devices, more pleasant and efficient interfaces and monitoring chronically-ill patients or operators of heavy machinery are a few examples of applications that

benefit from physiological data and conclusions about psychological state that can be drawn from that data. For example, physiological responses to different web page designs [Ward and Marsden, 2003] could provide objective evaluation of user experience and promote usability.

Self-reports of psychological states and events are often considered unreliable and theoretical structures built on self-report data fragile [Cacioppo *et al.*, 2000]. Furthermore, the goal of supporting people's actions with unobtrusive methods of computing, ubiquitous computing [Weiser, 1993], can not be reached with self-reports that require shifting awareness from task to irrelevant activity (i.e. reporting).

Despite the benefits, the development of psychophysiologically interactive systems is still restrained by many problems in analysis and interpretation of physiological signals. Individual differences in measures prevent interpretations based on general models of psychophysiology [Ward and Marsden, 2003]. The situation and environment affects the measures in many ways, which also causes inconsistencies between measurements. In addition, one physiological reaction may be connected to several psychological states and vice versa [Cacioppo *et al.*, 2000]. This complicates the analysis and interpretation required to conclude psychological state from physiological data.

Supporting the development of psychophysiologically interactive systems requires addressing these difficulties in software development. Frameworks are implementations that solve domain-specific problems [Johnson, 1997]. A framework provides the most basic parts for a system and supports the development of the final application. The design of a framework involves identifying the functionality common to applications in the target domain and forming a core that does not contain application-specific functionality [Flippo *et al.*, 2003].

The aim of this work is to provide software developers with a framework that supports the development of systems that reliably collect and process physiological data and form psychological interpretations based on that data. First the target applications and their common requirements must be identified.

2. Psychophysiologically interactive systems

2.1. Psychological interpretations based on physiology

Electrophysiologically interactive computer systems combine physiological sensing technologies with interactive computer applications [Allanson, 2002]. Psychophysiological interaction can be seen as a special case of physiological interaction. Connections with psychological states and events have been identified for most physiological measures. It is thus possible to form interpretations of psychological activity based on physiological activity. To make such interpretations the system requires a model of the relationships between physiological and psychological states and events. The model does not have to be an explicit part of the architecture. It can be implicit in the processing flow of the system.

The model could be based on previous psychophysiological research. Physiological correlates have been identified for stress [Ali and Marsden, 2003], positive and negative affect [Larsen *et al.*, 2003], mental effort [Rowe *et al.*, 1998], and frustration [Schreirer, 2002], among others. However, forming a model connecting the psychology with the physiology is not a trivial task.

First, the psychophysiological research has focused on the identification of clear effects and basic phenomena in laboratory settings. Psychophysiological interaction in the real world involves noisy data. One part of this problem is that the physiological measurements are changeable [Ward and Marsden, 2003]. The inherent unstability of the measures makes the identification of significant changes more difficult.

Second problem derives from individuality of physiological responses. The individuality of responses restricts the use of a general model, i.e. the same model for all users. Third, the researched signals are not specific to one psychological measure. Most signals show the combined effect of many psychological factors (see Nasoz *et al.* [2004] for a summary of research on recognizing affect from physiological reactions) and physiological responses to different factors are nearly identical [Ward and Marsden, 2003].

Cacioppo and others [2000] divide psychophysiological relationships into four categories (see Figure 1). The relationships are categorized according to their generality and specificity. Context-independent relationships hold across situations, while context-dependent do not. One-to-one relationships are direct correlations between physiological and psychological elements. Utilizing these

relationships in the interpretation of physiological activity is easier than making conclusions based on many-to-one relationships. Physiological activity that can be caused by a number of psychological elements is difficult to exploit.

The problem of noisy data is solved in multimodal systems by forming interpretations that are based on many signals, not just one [Liebermann, 2002; Oviatt and Cohen, 2000]. Combining several unreliable sources can lead to reasonably reliable results when the observed effects in parallel channels overlap. The fusion of multiple channels can also act as a partial solution to the problem of separating the effects of concurrent psychophysiological reactions from one another.

This solution is comparable to the multivariate approach in statistical analysis of physiological signals: many dependent variables and their covariation are considered at a time [Gratton, 2000]. The reliability can be further improved by collecting information about the context, external behaviour of the user, and other modalities besides the physiological activity. Cacioppo and others [2000] state the significance of context in psychophysiological research:

> "… a wide range of complex relationships between psychological and physiological phenomena might be specifiable in simpler, more interpretable forms within specific assessment contexts."

The individuality of physiological reactions can also be considered by adapting to the context and the person. The context may provide cues that help in the interpretation of the reaction, even if it is highly individual in nature. Long-term data collection and analysis helps in estimating and anticipating the effects that a particular context has on the individual, or recognizing individual differences in psychophysiological effects.

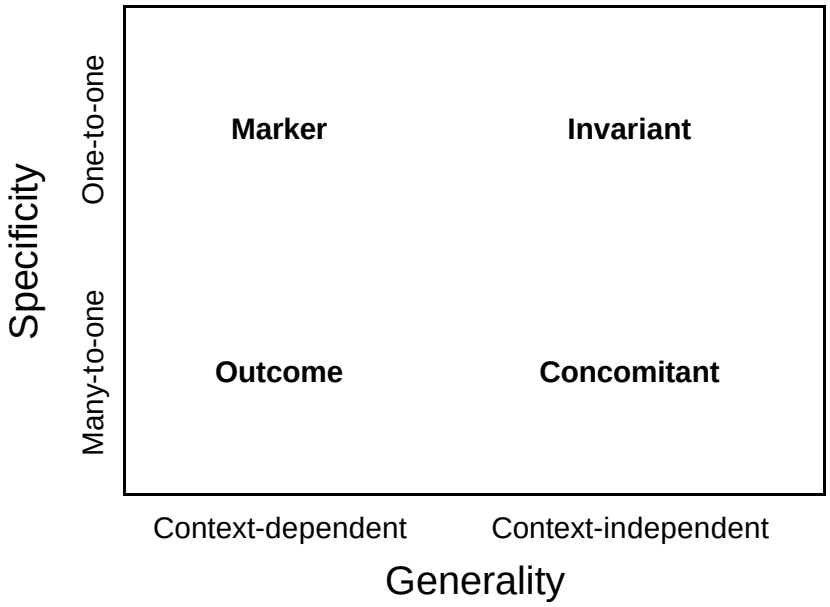


Figure 1. Taxonomy of psychophysiological relationships

[after Cacioppo, 2000].

2.2. Applications for psychophysiological interaction

Allanson [2002] divides electrophysiologically interactive computer systems into two categories: training systems and monitoring systems. The first category consists of systems that aim to enable users to consciously control their own physiology through vigorous training. This category does not require the system to make psychological interpretations as the collected and processed signals are displayed to the user without modification. These systems are conventionally referred to as biofeedback systems.

Systems of the second type collect information of a person's physiological state over time. The psychophysiologically interactive systems are a part of this category. The collected information is analyzed using a model of physiological and psychological correlations. The model is not necessarily complex and can for example consist of simple amplitude thresholds that trigger certain events. A threshold-based model is quite useful in interfaces that are guided by physiological signals.

Surakka and others [accepted] have built an interface that is controlled by gaze and voluntary facial muscle activity. The voluntary activity of *corrugator*

supercilii muscle acts as a selection trigger analogous to a mouse click. A target is displayed on a computer screen and can be selected by focusing on the target and frowning, i.e. activating the *corrugator* muscle. Although the system uses and analyses the physiological signal for a very restricted and simple purpose, the interaction can still be seen as psychophysiological. The electromyogram (EMG) signal captures the intent of the user and the true innovation in this case is the combination of this intent and the attentional information obtained from the gaze.

Hands-free operation of a computing device is useful for the disabled or in situations where the hands are occupied, e.g. while driving a car. Other possible applications for physiological signals in the control of an interface include the hands-free operation of a wheel chair [Felzer & Freisleben, 2000], Brain-Computer Interfaces [Hinterberger *et al*., 2004; Millán, 2003; Wolpaw *et al*., 2002], and electrooculography (EOG) and electroencephalography (EEG) based gaze tracking [Joyce *et al*., 2002] that could be utilized in human-computer interaction [Zhai, 2003]. It is easy to come up with a number of other systems that could be physiologically controlled, e.g. vehicles and devices such as PDAs, information kiosks, and smart phones.

Intentional, goal-directed behavior can also be supported without user's voluntary input. In addition to monitoring attention with electrophysiological gaze tracking [Jacob and Karn, 2003] and disambiguating dialogues [Bosma and André, 2004], the physiological signals can be applied to a field of human-computer interaction called affective computing [Picard, 2000]. Affect plays a major role in human-to-human interaction and even influences human perception [Picard, 2000b]. Systems that sense, identify, and react to emotions would have a significant advantage in understanding and supporting behavior. Several physiological measures of affect have been identified, e.g. heart rate, skin conductance [Levenson *et al*., 1990], facial muscle activity [Larsen *et al*., 2003], pupil size [Partala and Surakka, 2003], and blood pressure [Scheirer *et al*., 2002] all correlate with emotional valence or arousal.

The emotional impact of user interfaces can also be utilized as a measure of usability [Allanson and Wilson, 2002]. The physiological signals are used as objective measures of subjective experiences. Pleasantness impacts user performance [Partala and Surakka, 2004] and physiological correlates of stress, mental effort, fatigue and cognitive workload could identify poorly designed interfaces and support productivity through the design of better ones. Usability

gains can also lead to greater safety in critical systems. Monitoring the awareness or emotional state of a car driver [Apolloni *et al.*, 2003; Nasoz *et al.*, 2002] supports the safety of both the driver and bystanders.

Health care systems that observe critical or chronically-ill patients are prime examples of monitoring systems. These systems could provide additional mobility and safety for patients that require constant monitoring. Some future possibilities for psychophysiological health care applications include the automatic regulation of glucose-insulin metabolism in diabetic patients [Amigoni *et al.*, 2003] and affective computing in tele-home health care [Lisetti, 2004].

There are many other applications for psychophysiological interaction and those presented here are not intended to encompass every application domain. Other uses for physiological information include identification [Deravi *et al.*, 2003], training for emotional intelligence [Predinger *et al.*, 2003], and monitoring vital functions in arctic environments [Rantanen *et al.*, 2002]. In fact, the applications are not limited to interaction between humans and machines. A research on salmon revealed which streams required the least effort during migration [Standen *et al.*, 2002]. The effort was measured with wireless electromyography (EMG).

2.3. Characteristics of psychophysiologically interactive systems

The recognition of involuntary human input is uncertain. The processing of physiological data is not comparable to the utilization of voluntarily produced signals to control graphical user interfaces, i.e. events from mouse and other conventional input devices. There is no need for recognition of events as these input methods generate them by convention. A press of a virtual widget generates a “click” event in most modern windowing toolkits (e.g. Java Swing [Sun, 2004], Qt [Trolltech, 2004], and MFC [Microsoft, 2004]] and processing and acting upon such an event is trivial. On the other hand, the uncertainty of physiological signal recognition transfers to the interpretation of recognized events. The problem is made worse by the nature of psychophysiological events: one physiological signal can reflect a number of parallel psychological events that are context-dependent (c.f. Figure 1).

The approach taken by many multimodal systems to reduce uncertainty is the fusion of parallel input signals. There are two approaches to modality fusion [Oviatt and Cohen, 2000]. The first approach aims to combine different sources of information at the feature level. In this early fusion temporally closely related

signals are combined with neural networks and other suitable signal analysis methods (see Gratton [2000] for an overview of biosignal analysis). The second approach combines independently recognized events at the semantic level. Significant events are recognized from each input stream independent from other channels and fused at the semantic level. Semantic rules used in the late fusion are more easily understood and defined by the developers and the users of the system. According to Oviatt and Cohen [2000] this approach also requires less training data for the system.

Although the parallel physiological signals are not necessarily independent modalities (e.g. the heart rate and skin temperature both reflect autonomous nervous system activity), these approaches can be applied to psychophysiologically interactive systems as well. By combining several unreliable sources it is possible to achieve reliable results and the reliability can be further improved by taking the context of usage into consideration [Lieberman, 2002]. Awareness of the context includes awareness of people, places and objects that are relevant to the current task [Dey, 2001]. Naturally, this also implies that the system must be aware of the task and related goals. This might be a greater challenge for psychophysiologically interactive systems than for multimodal systems, for example, as the applications and associated tasks for these systems are quite diverse. It is likely that all of the possible applications have not even been thought of yet.

It should be noted that the context is not static. There may be some applications and systems with fixed people, places and objects participating in the interaction, but most mobile and distributed systems are dynamic. The registration, processing, collection and interpretation of physiological data do not need to happen with a certain device in a fixed place at a specified time. For example, the monitoring of chronic heart conditions could be performed using a portable, wireless device that collects the data and transmits it to permanent storage when a suitable server is within the range of the wireless connection. Kine [2002] provides such a system for wireless EMG.

The communication and division of roles between these diverse devices and platforms pose many challenges. One goal that physiological computation could promote is ubiquity in computation. In ubiquitous computing the computer supports activities of a person in a way that does not require human attention [Weiser, 1993]. Ubiquity is made possible by continuous interaction with wirelessly interconnected devices. Fitzmaurice and others [2003] present four

questions for ubiquitous technology: who employs "computation", where they do so, how they interact, and what it is used for. Even if answers to these questions can be defined for a certain static context, the system must adapt when the "who" and the "where" are no longer a part of the environment. For example, if a constantly monitored chronically-ill patient who works in an office leaves from her room, the PDA or other portable device must assume some responsibilities from the desktop computer. However, the PDA does not have the signal processing capabilities or the network bandwidth of the desktop computer. Therefore the two devices are not interchangeable in the system architecture, which has to be modified.

In summary, the characteristic properties of and requirements for a psychophysiological system are: support for different levels of data fusion, awareness of context and task, distributedness, and adaptability to changes in computing environments. The fulfilment of every requirement is not required when building a psychophysiologically interactive system, but their significance increases as developers move from building single prototype systems to developing robust, extensible and manageable real-world systems.

3. Developing psychophysiologically interactive systems

3.1. Architectures for psychophysiological interaction

Several architectures have been proposed for multimodal and other signal processing systems. The multimodal systems can be seen as analogous to psychophysiologically interactive systems: although the processed signals are different, the same methods and solutions can be applied to both.

Many multimodal architectures focus on the control of applications and the dialogue between the user and the system. These architectures do not readily adapt for other categories of psychophysiologically interactive systems. As previously discussed, novel input techniques for interfaces are only a small part of the possibilities for monitoring of human physiology. Nonetheless, these architectures can provide some insight into the problems of and techniques for processing physiological signals.

In SmartKom [Reithinger *et al.*, 2003] the fusion of input data is performed on the semantic level. Linguistic and gesture objects are resolved from speech and movements independently, and the references are resolved using a three-tiered model. This model acts as the discourse memory which provides history (i.e.

context) for the interaction. Although this solution is good for multimodal voluntary control in applications, the method cannot be generalized for other types of psychophysiological interaction, e.g. the monitoring of chronically ill patients. It is also difficult to extract features from a single physiological signal as discussed. Most psychophysiologically interactive systems require low-level fusion, which can be augmented with high-level, semantic fusion.

The multi-agent system Embassi is also aimed at performing tasks multimodally through discourse with the system [Elting *et al.*, 2003]. The agents are grouped into layers that share the same abstraction level. The agents communicate within a pipelined structure. The modalities are fused by a single agent by merging the semantic structures that are created independently for every modality. The input modalities are synchronized by querying analyzers for other modalities each time user input is received from any modality.

Again, the high level fusion of signals is not feasible in most psychophysiological applications. The grouping of the agents into layers may promote clarity [Elting *et al.*, 2003], but restricts the adaptability of the architecture. All of the layers are not necessary in psychophysiological systems that perform early, low-level fusion. The fusion model of Embassi relies on querying concurrent input signals, or semantic meaning extracted from them, and does not work well in applications that constantly register and store physiological data, while simultaneously analysing it to detect aberrations that require intervention.

One approach to solve a complex problem is to divide it into subproblems that can be handled by automatic, independent actuators, i.e. agents. The QuickSet architecture uses a central facilitator that handles communication between individual software agents [Oviatt, 2000]. A facilitator can handle requests from agents, divide the requests into tasks, and delegate these tasks to agents that can perform them [Moran *et al.*, 1998]. The agents register to the facilitator, which joins them to the functioning system. This architecture promotes distribution and platform independence [Oviatt, 2000]. However, the facilitator can form a bottleneck in systems where data is frequently interchanged between agents [Moran *et al.*, 1998]. Systems that register and process thousands of electrophysiological samples a second fit the definition.

Allanson [2002] presents a toolkit for building electrophysiologically interactive computer systems (EpICS). The toolkit consists of JavaBean components that enable graphical development of systems and may be especially

useful for prototyping and non-programmers. The system architecture supported by the toolkit is divided into two processing layers: device layer and command layer. The integration of concurrent signals is not explicitly dealt within either layer and the implementation is specific to hardware and software environments.

There are many architectures for multimodality, but there does not seem to be a single architecture that would fit every purpose and could be applied to all psychophysiological interaction. However, the architectures contain some general solutions that could be extracted and collected into a framework. The general framework could be used in the development of several types of systems with diverse architectures, such as those that were presented.

3.2. Dataflow in processing of physiological signals

Arroyo and Childers [1982] presented a modular software system for the detection of brain-activity as early as in 1982. Their system consisted of separate programs that were run sequentially. Each program transforms the data in a form that can be consequently processed by other programs. The solutions that are applied to subproblems can later be reused and reorganized to solve other, more complex problems.

Similar schema of information processing is captured in the pipes and filters of Unix environments. The Unix shell (i.e., text-based user interface) uses a special pipe character (“|”) to join programs together. For example, the result of the joined commands in Figure 2 is that the system sends a mail to the address “Some.One@Somewhere.biz”. The mail contains the number of lines in the file “test.txt”. The first program simply reads the file and sends it to another program that counts the lines. The line count is sent to the last command in the pipeline. The “mail” command sends the count to the recipient via electronic mail.

```
cat ‘testi.txt’ | wc -l | mail -s “Line Count” Some.One@Somewhere.biz
```

Figure 2. A Unix pipeline.

The pipes and filters pattern is utilized in software development as well [Buschmann *et al.*, 1996]. Design patterns are general solutions to recurring problems in object-oriented design [Schmidt *et al.*, 1996]. Conventionally the

pipes and filters pattern solves the problem of sequential data processing. In other words, the system architecture forms a tree (i.e. a series of connected filters without pipes that connect to previous filters). However, the solution can be extended to handle more complex architectures.

Architecture based on separate data processors is easy to understand and manage. The filters can be reused in different systems and common parts of algorithms need to be implemented and computed only once [Ilmonen and Kontkanen, 2003]. Each processing element can be modelled as a filter which has the capability for multiple simultaneous input and output channels. The components read data from their input channels, perform processing and output the results. The input and output channels are the ends of a pipe, which can be seen as a buffer that contains processing items.

The processing elements require different amount of data at variable rates. Thus the properties of a pipe cannot be defined by the component at the transmitting end. It is however unefficient to upkeep several pipes (and buffers) for every element, although the data items can be stored as references instead of separate copies.

One solution is to upkeep a common history buffer that stores the items produced by a filter [Ilmonen and Kontkanen, 2003]. This is feasible when the receivers are active in reading the buffer. Unfortunately, the solution does not work well with the distributed architectures of most multimodal systems. When the processing components run in different environments, they must perform queries with a higher level language or execute remote procedure calls, which add a significant overhead.

This paper presents the PISARA (Psychophysiologically Interactive Systems with Agents, Resources and Archives) framework. In this framework every processing element has its own buffers for input and output channels (see Figure 3). The framework does not actually require the use of a separate input buffer, but it is often used in applications and supported by PISARA.

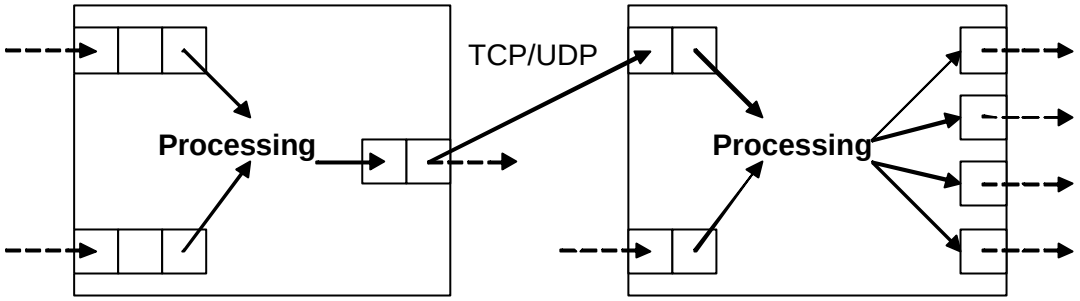


Figure 3. Two interconnected filters.

PISARA supports multiple output and input types for a filter. For example in Figure 3, the filter on the left has two input channels and one output channel, while the filter on the right has two input channels and four output channels. Each filter flushes its output buffers when they are full and sends data to all recipients using either Transmission Control Protocol (TCP) or User Datagram Protocol (UDP). In Figure 3, the left-hand filter sends data both to the right-hand filter and to another filter, not displayed in the figure. The left-hand filter does not have a buffer for both recipients. The network capabilities are taken into consideration as the size of the sent packages can be specified by setting the size of the buffer. In addition, the recipients do not need to query for the data, which promotes efficiency. The recipients read the data and store it into an internal buffer, if necessary, until they have enough data to perform processing.

The overhead from network connections and input buffering could be reduced by implementing shared buffers for processing elements that can share memory, i.e. components running in the same hardware and software environment. It is easy to extend the current implementation to support this. However, this adds complexity to building and managing these architectures, especially when the architecture is specified automatically and dynamically at runtime.

3.3. Agent-based framework

Mobility and ubiquity produce problems for systems that process physiological data. The network and processing capabilities can change as new devices become available or leave the system. Unstable platforms and filters must be replaced or circumvented. The systems must adapt to new contexts and tasks with minimum effort.

The division of signal processing into small, manageable filters supports the distributability of systems. However, the composition and rearrangement of architectures consisting of filters in different hardware and software environments requires higher lever support. In the PISARA framework the filters are encapsulated in agents. These agents have communication capabilities which enable the dynamic alteration of the system's architecture.

Every filter registers to a central agent called the Broker. The registration message contains a description of the input and output capabilities of the agent (see Figure 4). The language of the message is based on Extensible Markup Language (XML). The inputs and outputs are currently described with simple text identifiers, but can later be extended to capture several levels of abstraction. The Broker stores the identifier of the agent, its contact information and the description of its capabilities. These are used later when agent requests a certain type of input or a connection to a specific agent.

```
<?xml version='1.0' encoding='utf-8'?>
<register>
    <IP>
        127.0.0.1:50004
    </IP>
    <id>
        CORRELATOR
    </id>
    <input>
        <id>
            EKG
        </id>
    </input>
    <output>
        <id>
            PULSE
        </id>
    </output>
</register>
```

Figure 4. An example of a registration message in XML-based language.

The Broker acts as a contact service for agents. When a pipe is created, the Broker asks the receiving agent to provide a socket for a TCP or an UDP

connection. Next the Broker provides the Uniform Resource Locator (URL) of the recipient to the sender who opens the connection and starts to send data.

The PISARA framework enables the system to react to changes in the hardware and software platforms. As the Broker upkeeps a list of the available agents, the possible compositions of filters and pipes can be analysed. When a processing element leaves the system, it can repair itself by finding replacements for the filter or nearly equivalent architectures that can function without the filter. The Broker can be seen as a provider of context that is comparable to the context-manager in Embassi [Elting *et al.*, 2003], discourse memory in SmartKom [Reithinger *et al.*, 2003], and dialog manager in the work of Flippo and others [2003].

The framework does not force the use of a specific architecture, although the architecture should be compliant with the pipes and filters design pattern. This is a very general requirement and all of the presented architectures for psychophysiological and multimodal interaction fit under this category. Also, there are no limitations or requirements on the capabilities of the agents. It is possible to later add agents that do not accompany a filter. In fact, the creation of an agent to dynamically build systems and supervise the efficiency and operation of the system has been considered.

However, the aim of adding agent functionality to the filters has been the management of dynamic architectures. This can be achieved by requesting the Broker to notify when the architecture changes and then reacting to those changes. The framework does not support collaboration of agents with complex social functions, for example mediating requests and negotiating. The framework is extensible and such functions can be added later, if necessary.

3.4. Implementation of the framework

The current implementation of the PISARA framework consists of an abstract class for agents, convenient abstractions for different types of filters, the Broker and some specialised application-specific agents. The base class for an agent is available both in Java and in C++. This class is extended to Sender, Receiver and Filter classes that contain the common functionality for agents that send data, receive data or do both, respectively (see Figure 5). These extensions are available only in C++.

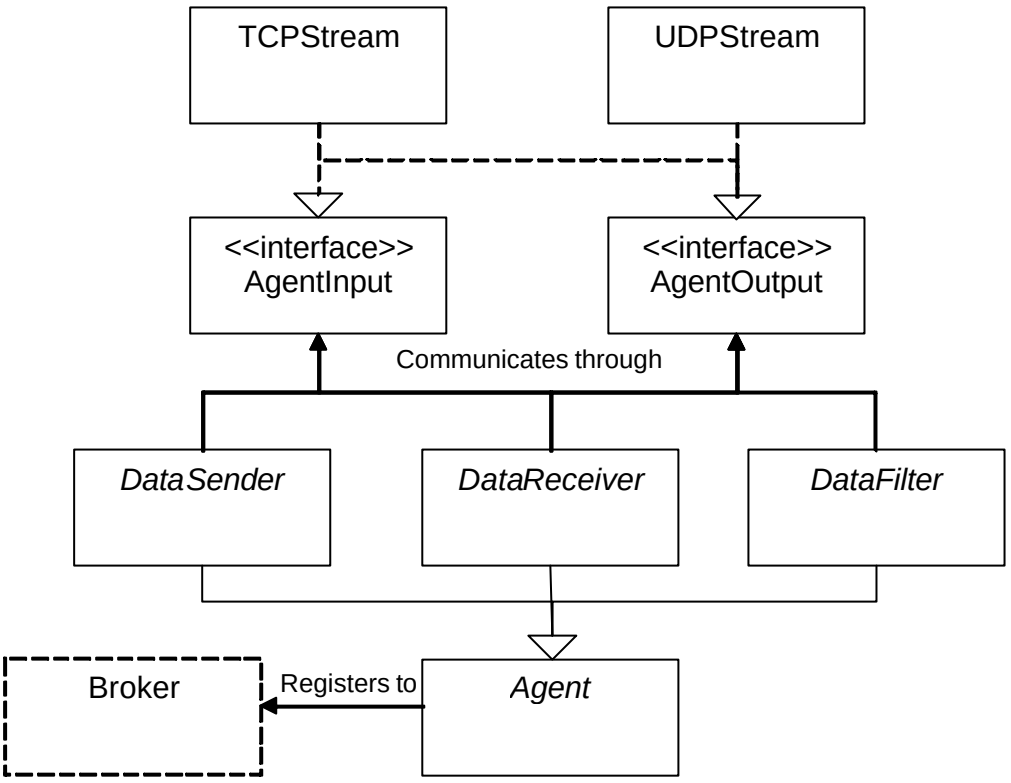


Figure 5. Class diagram of agents and their connections.

The Broker is implemented in Java. It currently does not support remote listeners or listeners written in other languages besides Java. Thus the management of the architecture must also be done in Java. The existing components and their inputs and outputs can be queried with a method call and the connection between two filters is created with another.

The connections between filters are accessed through an AgentInput or an AgentOutput interface. This interface hides the true implementation of the stream, which is currently based either on UDP or TCP. A stream to support shared memory between local filters will be implemented later.

To create a new filter from scratch, one has to extend either the base Agent class or one of the more specialised abstract classes. This consists of overriding a method that returns the XML presentation of the agent and programming the algorithm used in the processing of the data. The framework takes care of the registration and other communication with the Broker.

Most of the events and messages that agents use to communicate with other agents can be delivered using the streams that connect filters. This is the case when the events and messages are related to the data processing and the control of processing. The co-operative communication between agents (e.g. related to social functions such as negotiating) is not currently supported, as the agent functionality is separate from the filters. The agents have no method of reading or interfering with the data transmitted between filters.

Currently only agents and the Broker communicate directly. The Broker handles all interaction between agents. The communication uses a language based on XML. This language can easily be extended if the need for interagent communication emerges. However, it seems that the current Broker-centered architecture is sufficient for the control and management of psychophysiologically interactive systems. This assessment is supported by the experiences that Moran and others [1998] have from the use of their Open Agent Architecture.

4. Conclusions and future work

The PISARA architecture consists of two abstraction levels. The pipes and filters architecture is used as the basis for signal processing. This architecture promotes reusability of algorithms and filters and supports different levels of signal fusion. The agent architecture equips the systems with tools to provide distributability,

platform independence and adaptability to changes in the hardware and software environment. The platform independence and distributability are supported by the use of a general representation of input and output capabilities of filters that are encapsulated in agents.

The Broker-based interaction between agents and the system enables context-awareness to a limited extent. It is possible to adapt to changes in the environment when agents leave or enter the system. The support for other types of awareness is limited. Implementing separate context agent could be one solution to the problem. Acting as the context manager [Elting *et al.*, 2003], the agent could provide information about the current task as well as other persons and objects relevant to the task. The architecture has not yet been used to build systems that would require the management of a more general context.

The main benefit of using the PISARA framework is that it captures the common processing and distribution needs of psychophysiologically interactive systems, but is general enough to adapt to different types of applications. The framework could assist in the development and use of the architectures that were described previously. The framework is free of restrictions concerning the organization of processing and interaction between elements. For example, the systems may follow a hierarchical architecture in the fusion of input signals, but they are not forced to do so.

Although the architecture is designed mainly for systems that fit Allanson's [2002] definition of monitoring systems, it should be noted that the framework can support the development of training systems as well. These systems do not require a model to fuse physiological data into psychological events, but this is not required by the PISARA framework.

There are many improvements that are planned to the framework. These improvements include a more structured description of input and output signals and the capabilities of a filter, automation in constructing different architectures and graphical manipulation of architectures.

Providing support for different abstraction levels in describing the filters would enable automation in fitting output and input signals together and thus support the automatic construction of systems. Automating the construction requires a goal-directed agent that knows how to evaluate architectural solutions. The development of such agents is supported by the language-independence of the framework. It is preferable to implement high-level deduction in logical or high-level language for the sake of clarity and

manageability, in most cases. Although the framework is currently implemented exclusively in C++ and Java, extending it to different languages should be a relatively small effort, as existing parts can interoperate with the new components.

The graphical manipulation of architectures could resemble the use of the JavaBean toolkit by Allanson [2002]. Providing simple tools to construct desired systems would reduce programming efforts as researchers and other field workers could build the prototypes themselves. As the architecture is Broker-centric, even direct manipulation of the system would be feasible through a single interface to the Broker.

Although it is possible to manipulate processing components as such, developing systems by specifying desired effects would be more practical. If systems could be automatically built based on a high-level specification, the effort in designing and programming would be minimal. This will have to be at least partially implemented to support adaptation to different contexts. As a result, the systems will effectively build themselves.

References

[Ali and Marsden, 2003] Ali, A. N. and Marsden, P. H., Affective multi-modal interfaces: the case of McGurk effect. In: *Proceedings of the 8th International Conference on Intelligent User Interfaces*, 224-226.

[Allanson, 2002] Allanson, J., Electrophysiologically interactive computer systems. *IEEE Computer Journal* **35**, 3 (Mar. 2002), 60-65.

[Allanson and Wilson, 2002] Allanson, J. and Wilson, G. M., Physiological computing. In: *CHI '02 Extended Abstracts on Human Factors in Computing Systems*, 912-913.

[Amigoni *et al.*, 2003] Amigoni, F., Dini, M., Gatti, N., and Somalvico, M., Anthropic agency: a multiagent system for physiological processes. *Artificial Intelligence in Medicine* **27,** 3 (Mar. 2003), 305-34.

[Apolloni *et al.*, 2003] Apolloni, B., Bassis, S., Brega, A., Gaito, S., Malchiodi, D., Valcamonica, H., and Zanaboni, A. M., Monitoring of car driving awareness from biosignals. In: *WIRN VIETRI '03, 14th Italian Workshop on Neural Nets, Lecture Notes in Computer Science* **2859**, Springer, 269-277.

[Arroyo and Childers, 1982] Arroyo A. A. and Childers, D. G., A modular software real-time brain wave detection system. In: *Proceedings of the 20th Annual Southeast Regional Conference*, 126-131.

[Bosma and André, 2004] Bosma, W. and André, E., Exploiting emotions to disambiguate dialogue acts. In: *Proceedings of the 9th International Conference on Intelligent User Interface*, 85-92.

[Buschmann *et al*., 1996] Buschmann, F., Meunier, R., Rohnert, H., Sommerland, P., and Stal., M., *Pattern-Oriented Software Architecture*. John Wiley & Sons, 1996.

[Cacioppo *et al.*, 2000] Cacioppo, J. T., Tassinary, L. G., and Berntson, G. G., Psychophysiological science. In: Cacicioppo, J. T., Tassinary, L. G., and Berntson (eds.), *Handbook of Psychophysiology, 2nd ed.* Cambridge University Press, 2000, 3-23.

[Deravi *et al*., 2003] Deravi, F., Fairhurst, M. C., Guest, R. M., Mavity, N. J., and Canuto, A. M. D., Intelligent agents for the management of complexity in multimodal biometrics. *Universal Access in Information Society* **2**, (2003), 293–304.

[Dey, 2001] Dey, A. K., Understanding and using context. *Personal and Ubiquitous Computing* **5**, (2001), 4-7.

[Elting *et al.*, 2003] Elting, C., Rapp, S., Möhler, G., and Strube, M., Architecture and implementation of multimodal plug and play. In: *Proceedings of the 5th International Conference on Multimodal Interfaces*, 93-100.

[Felzer and Freisleben, 2002] Felzer, T. and Freisleben, B., HaWCoS: the "hands-free" wheelchair control system. In: *Proceedings of the 5th International ACM Conference on Assistive Technologies*, 127-134.

[Fitzmaurice *et al.*, 2003] Fitzmaurice, G. W., Khan, A., Buxton, W., Kurtenback, G., and Balakrishnan, R., Sentient data access via a diverse society of devices. *ACM Queue* **1**, 8 (2003), 53-62.

[Flippo *et al*., 2003] Flippo, F., Krebs, A., and Marsic, I., A framework for rapid development of multimodal interfaces. *Proceedings of the 5th International Conference on Multimodal Interfaces*, 109-116.

[Gratton, 2000] Gratton, G., Biosignal processing. In: Cacicioppo, J. T., Tassinary, L. G., and Berntson (eds.), *Handbook of Psychophysiology, 2nd ed.* Cambridge University Press, 2000, 900-923.

[Hinterberger *et al.*, 2004] Hinterberger, T., Neumann, N., Pham, M., Kübler, A., Grether, A., Hofmayer, N., Wilhelm, B., Flor, H., and Birbaumer, N., A

multimodal brain-based feedback and communication system. *Experimental Brain Research* **154**, 4 (2004), 521-526.

[Ilmonen and Kontkanen, 2003] Ilmonen T. and Kontkanen, J., Software Architecture for Multimodal User Input – FLUID. In: *The Proceedings of the 7th ERCIM Workshop, Lecture Notes in Computer Science* **2615**, (2003), 319-338.

[Johnson, 1997] Johnson, R. E., Frameworks = (Components + Patterns). *Communications of the ACM* **40**, 10 (Oct. 1997), 39-42.

[Jacob and Karn, 2003] Jacob, R. J. K., and Karn, K. S., Eye tracking in human-computer interaction and usability research: Ready to deliver the promises (section commentary). In: Hyona, J., Radach, R., and Deubel, H. (eds.) *The Mind's Eye: Cognitive and Applied Aspects of Eye Movement Research*, 573-605.

[Joyce *et al.*, 2002] Joyce, C. A., Gorodnitsky, I. F., King, J. W., and Kutas, M., Tracking eye fixations with electroocular and electroencephalographic recordings. *Psychophysiology* **39**, (2002), 607-618.

[Kine, 2002] Kine ehf., KineMyo® product description. http://www.kine.is/pages/myo.pdf. 5.5.2004.

[Larsen *et al.*, 2003] Larsen, J. T., Norris, C. J., and Cacioppo, J. T., Effects of positive and negative affect on electromyographic activity over zygomaticus major and corrugator supercilii. *Psychophysiology* **40**, (2003), 776-785.

[Levenson *et al.*, 1990] Levenson, R. W., Ekman, P., and Friesen, W. V., Voluntary facial action generates emotion-specific autonomic nervous system activity. *Psychophysiology* **27**, 4 (1990), 363-384.

[Lieberman, 2002] Lieberman, H., Out of many, one: reliable results from unreliable recognition. In: *CHI '02 Extended Abstracts on Human Factors in Computing Systems*, 728-729.

[Lisetti *et al.*, 2003] Lisetti, C., Nasoz, F., LeRouge, D., Ozyer, O., and Alvarez, K., Developing multimodal intelligent affective interfaces for tele-home health care. *International Journal of Human-Computer Studies* **59**, 1-2 (Jul. 2003), 245-255.

[Microsoft, 2004] Microsoft Corporation, Microsoft Foundation Class Library. http://msdn.microsoft.com/library/en-us/vcmfc98/html/mfchm.asp. 7.5.2004.

[Millán, 2003] Millán, J. del R., Adaptive Brain Interfaces. *Communications of the ACM* **46**, 3 (Mar. 2003), 74-80.

[Moran *et al.*, 1998] Moran, D. B., Cheyer, A. J., Julia, L. E., Martin, D. L., and Park, S., Multimodal user interfaces in the Open Agent Architecture. *Knowledge-Based Systems* **10**, (1998), 295-303.

[Nasoz *et al.*, 2004] Nasoz, F., Alvarez, K., Lisetti, C. L., and Finkelstein, N., Emotion recognition from physiological signals using wireless sensors for presence technologies. *International Journal on Cognition, Technology and Work* **6**, (2004), 4-14.

[Oviatt, 2000] Oviatt, S., Multimodal system processing in mobile environments. In: *Proceedings of the 13th Annual ACM Symposium on User Interface Software and Technology, CHI Letters* **2**, 2 (Nov. 2000), 21-30.

[Oviatt and Cohen, 2000] Oviatt, S. and Cohen, P., Perceptual user interfaces: multimodal interfaces that process what comes naturally. *Communications of the ACM* **43**, 3 (Mar. 2000), 45-53.

[Partala and Surakka, 2003] Partala, T. and Surakka, V., Pupil size variation as an indication of affective processing. *International Journal of Human-Computer Studies* **59**, 185-198.

[Partala and Surakka, 2004] Partala, T. and Surakka, V., The effects of affective interventions in human-computer interaction. *Interacting with Computers,* **16**, 2 (2004), 295-309.

[Picard, 2000] Picard, R. W., *Affective Computing*. The MIT Press, 2000.

[Picard, 2000b] Picard, R. W., Affective perception. *Communications of the ACM* **43**, 3 (Mar. 2000), 50-51.

[Prendinger *et al.*, 2003] Prendinger, H., Mayer, S., Mori, J., and Ishizuka, M., Persona effect revisited: using bio-signals to measure and reflect the impact of character-based interfaces. In: *Proceedings of 4th International Conference on Intelligent Virtual Agents, Lecture Notes in Computer Science* **2792**, 283-291.

[Rantanen *et al.*, 2002] Rantanen, J., Impiö, J., Karinsalo, T., Malmivaara, M., Reho, A., Tasanen, M., and Vanhala, J., Smart dothing prototype for the arctic environment. *Personal and Ubiquitous Computing* **6**, 1 (Jan. 2002), 3-16.

[Reithinger *et al.*, 2003] Reithinger, N., Alexandersson, J., Becker, T., Blocher, A., Engel, R., Löckelt, M., Müller, J., Pfleger, N., Poller, P., Streit, M., and Tschernomas, V., SmartKom: adaptive and flexible multimodal access to multiple applications. In: *Proceedings of the 5th International Conference on Multimodal Interfaces*, 101-108.

[Rowe *et al.*, 1998] Rowe, D. W., Sibert, J., and Irwin, D., Heart rate variability: indicator of user state as an aid to human-computer interaction. In:

Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, 480-487.

[Scheirer *et al.*, 2002] Scheirer, J., Fernandez, R., Klein, J., and Picard, R. W., Frustrating the user on purpose: a step toward building an affective computer. *Interacting with Computers* **14**, 2 (Feb. 2002), 93-118.

[Schmidt *et al.*, 1996] Schmidt, D. C., Fayad, M., Johnson, R. E., Software patterns. *Communications of the ACM* **39**, 10 (1996), 37-39.

[Standen *et al.*, 2002] Standen, E.M., Hinch, S. G., Healey M. C., and Farrell, A. P., Energetic costs of migration through the Fraser River Canyon, British Columbia, in adult pink (Oncorhynchus gorbuscha) and sockeye (O. nerka) salmon as assessed by EMG telemetry. *Canadian Journal of Fisheries and Aquatic Sciences* **59,** 1809-1818.

[Sun, 2004] Sun Microsystems, Inc., Java Foundation Classes (JFC/Swing). http://java.sun.com/products/jfc/index.jsp. 7.5.2004.

[Surakka *et al.*, accepted] Surakka, V., Illi, M., and Isokoski, P., Gazing and frowning as a new technique for human-computer interaction. *ACM Transactions on Applied Perception*.

[Trolltech, 2004] Trolltech, Qt Overview. http://www.trolltech.com/products/qt/index.html. 7.5.2004.

[Ward and Marsden, 2003] Ward, R. D., and Marsden, P. H., Physiological responses to different WEB page designs. *International Journal of Human-Computer Studies* **59**, (2003), 199-212.

[Weiser, 1993] Weiser, M., Some computer science issues in ubiquitous computing. *Communications of the ACM* **36**, 7 (Jul. 1993), 75-84.

[Wolpaw *et al.*, 2002] Wolpaw, J. R., Birbaumer, N., McFarland, D. J., Pfurtscheller, G., and Vaughan, T. M., Brain-computer interfaces for communication and control. *Clinical Neuropsychology* **113**, (2002), 767-791.

[Zhai, 2003] Zhai, S., What's in the eyes for attentive input. *Communications of the ACM* **46**, 3 (Mar. 2003), 34-39.

Tietokonerepresentaatiot populaarikulttuurissa

Marko Ylitalo

Tiivistelmä.

Teknologian kehitykseen ja erityisesti tietokoneisiin on liitetty suuria toiveita ja pelkokuvia. Nämä tunteet heijastuvat populaarijulkisuudessa, jossa koneita representoidaan tietyllä tavalla. Erilaisista mediakulttuurin tuotteista on luettavissa koneisiin kulloinkin liitetyt ajatukset ja se, mitä kyseisillä koneen kuvaamisen tavoilla on haluttu välittää.

Avainsanat ja -sanonnat: tietokone, mediakulttuuri, representaatio, populaarikulttuuri.
CR-luokat: K.4

1. Johdanto

Tämä tutkimus käsittelee sitä, millä tavalla tietokoneita ja koneita ylipäätään, kuvataan mediakulttuurissa ja sitä, mitä merkityksiä halutaan välittää kyseisillä koneiden kuvaamisen tavoilla. Käsittelen mediakulttuurin kenttää laaja-alaisesti mainoksista musiikkivideoihin, tv-sarjoista elokuviin ja kirjallisuudesta musiikkiin.

Tutkimuksessani keskityn niin sanotun populaarikulttuurin tuotteisiin, ja nostan esille yksittäisiä teoksia (tekstejä) konkretisoidakseni havaintojani esimerkkien avulla. Tietoteknistä populaarijulkisuutta edustavat lähinnä suurelle yleisölle suunnatut elokuvien, kaunokirjallisuuden ja lehtien kuvaukset tietokoneista [Suominen, 2000, s. 75].

En puutu tässä tutkimuksessa siihen, mikä on populaarikulttuuria ja mikä ei, koska populaarikulttuurin rajat ovat epämääräiset ja jatkuvassa liikkeessä. Erilaisten kulttuuristen alueiden suhteita määritellään jatkuvasti uudelleen [Grossberg, 1995, s. 59]. Tutkimukseni kannalta olennaisempaa on kulttuurisissa tuotteissa representaatioiden kautta esiin tulevat merkitykset.

Mediakulttuurin tuotteet ovat osa sosiaalista ja kulttuurista kokemusmaailmaamme. Elokuvat ja muut tuotteet heijastavat aikaansa ja kertovat siinä tapahtuvista muutoksista. Tietokoneen kehitys sekä teknologisesti että käyttö-

tarkoituksiltaan on ollut valtavaa, ja on jatkuvassa muutoksessa. Muutokset ovat heijastuneet aikansa populaarikulttuuriin, jonka diskurssista voidaan lukea millaisia, mielikuvia tietokoneisiin on liitetty sekä millä tavoin ja missä merkityksissä koneita on representoitu.

Elokuvatutkimuksen professori Jukka Sihvonen näkee, että olennaista teknologian tutkimuksessa on sen liittäminen laajempaan yhteyteen: sosiaaliseen, kulttuuriseen, historialliseen, ideologiseen ja symboliseen [Sihvonen, 1995, s. 92]. Kulttuurintutkija Lawrence Grossberg [1995, s. 57] on sitä mieltä, että tekstin merkitys sijoittuu siihen konkreettiseen yhteyteen, jossa teksti esiintyy ja, jossa se artikuloidaan uudelleen. Näin ollen on selvää, että media- ja populaarikulttuurin tuotteista on luettavissa ne merkitykset, joita tietokoneisiin on liitetty aikaisemmin ja joita niihin liitetään nyt ja tulevaisuudessa.

Aluksi selvitän, mitä tarkoitan joillakin käsitteillä, joita tutkimuksessani käytän. Sen jälkeen otan esille tieteeseen ja teknologiaan yleisesti liitetyt mielikuvat, jonka jälkeen paneudun esimerkkien ja lähdekirjallisuuden avulla koneiden aiheuttamiin toiveisiin ja pelkoihin sekä niiden synnyttämiin representaatioihin erilaisissa kulttuurisissa tuotteissa. Lopuksi kokoan ajatuksia yhteen.

2. Representaatio, teksti ja lukeminen

Ensiksi on syytä tarkentaa mitä tarkoitan käsitteillä representaatio, teksti ja lukeminen. Yksinkertaisesti ilmaistuna representaatiolla tarkoitetaan jonkin esittämistä joksikin. Representoitu kohde näyttäytyy katsojalle sellaisena kuin se esitetään, ei sellaisena kuin se on. Katsojan tulkintaan vaikuttavat asiayhteyden konteksti, katsojan tulkinnalliset kyvyt ja aikaisemmat kokemukset sekä kulttuurinen tausta. Merkitys on aina kontekstisidonnaista.

On mahdollista, että vastaanottajan ja kohteen väliset kulttuurierot voivat johtaa väärinkäsityksiin [Fiske, 1992, ss. 14-15]. Tässä mielessä tutkimuskohteiksi ottamieni mediatekstien koneiden ja teknologian representaatiot voivat herättää jossain toisessa lukijassa erilaisia mielleyhtymiä kuin miten itse olen niitä tulkinnut. Monet kulttuurikriitikot ovatkin sitä mieltä, että kulttuurisia tekstejä voidaan lukea erilaisilla tavoilla [Grossberg, 1995, s. 5].

Representaatiot eli esitysmuodot ovat jäljittely- tai kuvaussuhteessa todellisuuteen [Sihvonen, 2001, s. 33]. Toisin sanoen todellisuus ja representaatio todellisuudesta ovat kaksi eri asiaa. Yksinkertaisesti sanottuna esitys tietokoneesta ei ole tietokone. Vielä on syytä muistaa, että todellisuuden ei tarvitse olla

lähtökohtana representaatiolle. Sihvonen [2001, s. 207] kysyykin, mitä on esimerkiksi se todellisuus, jota vaikkapa *Pokémon* simuloi, representoi tai heijastaa.

Tekstillä tarkoitan mitä tahansa kulttuurista tuotetta. Se voi olla esimerkiksi elokuva tai mainos, jota luetaan eli tulkitaan samaan tapaan kuin kirjaimia paperilla, jotka muodostavat sanoja ja nämä edelleen merkityksiä lauseiden muodossa. Mediatekstien tulkinta tuottaa merkityksiä. Viestinnän tutkija John Fisken [1992, s. 16] mukaan lukeminen on prosessi, jossa lukija on vuorovaikutuksessa tekstin kanssa ja se synnyttää merkitykset. Tekstin merkitys ei piile tekstissä itsessään [Grossberg, 1995, s. 56].

Jossain määrin tulkinnat representaatioiden merkityksistä ovat subjektiivisia. Vastaanottotutkimuksen mukaan teksteillä ei ole yhtä tiettyä merkitystä. Eri yleisöt ja yleisöjen jäsenet voivat tulkita tekstejä eri tavoin. [Fairclough, 1997, s. 28] Fiske jatkaa samoilla linjoilla todeten, että sanoman merkitykset syntyvät vasta vuorovaikutuksessa vastaanottajan kanssa [1992, s. 16]. Lukijan kulttuuriset kokemukset vaikuttavat luentaan.

Fiske on erottanut tutkimuksessaan kaksi pääkoulukuntaa, jotka ovat prosessikoulukunta ja semioottinen koulukunta. Ensin mainittu koulukunta näkee viestinnän sanomien siirtona ja jälkimmäinen koulukunta näkee viestinnän olevan merkitysten tuottamista ja vaihtoa. Semioottinen koulukunta tutkii tekstien roolia kulttuurissamme ja tekstien merkityksenantoa. Näin ollen viestinnän tutkimus on tekstien ja kulttuurin tutkimusta. [Fiske, 1992, ss. 14-15] Representaatioon voidaan erilaisilla keinoilla lisätä merkityksiä. Tässä tutkimuksessa tutkin mediatekstejä niiden luomien representaatioiden osalta koskien koneille annettuja merkityksiä.

3. Tieteen hyvän ja pahan myytti

Länsimainen kulttuuri perustuu tieteeseen ja tiedettä kohtaan liitetään monenlaisia ennakkoluuloja. Merkityksellistämisestä puhuessaan Fiske [1992, s. 120] käyttää Roland Barthesin käsitteiden myytin ja vastamyytin esimerkkinä tiedettä.

Vallitsevan myytin mukaan tiede edustaa ihmiskunnan saavutusten oikeutusta. Tiede ymmärretään objektiiviseksi ja hyväksi. Vastamyytin mukaan tiede on kaiken pahan alku ja juuri, tieteen avulla tavoitellaan lyhytnäköisesti pelkästään aineellista hyvää. Samalla etäännytään luonnosta.

Populaarikulttuurissa molemmat edellä mainituista myyteistä ovat hyvin edustettuina. Vallitseva myytti näkyy television asiaohjelmissa, uutisissa, ajan-

kohtaisohjelmissa ja dokumenteissa. Vastamyytti sen sijaan näkyy elokuvissa ja television fiktiivisissä ohjelmissa. [Fiske, 1992, s. 120] On kiinnostavaa havaita tämä selkeä kontekstisidonnainen kaksijakoisuus, missä yhteyksissä myytit tulevat julki.

Dokumenteissa ja muissa asiaohjelmissa suhde tietokoneisiin on yleensä myönteinen. Niissä puhutaan teknologian kehityksestä ja sen mukanaan tuomista positiivisista puolista. Tietotekniikka, koneistuminen ja teknologian kehitys ovat edistyksellisyyttä. Sen sijaan monissa fiktiivisissä mediakulttuurin tuotteissa, kuten elokuvissa ja kaunokirjallisuudessa, teknologian ja tieteen kehitys on luonut uuden vaaran ihmiselle. Ihmisen luoma edistys kääntyy tekijäänsä vastaan. Pahan teknologian ja tiedon pelkoon liittyvät uhkakuvat näkyvät usein tietoteknisiä menestystarinoita kyseenalaistavissa pakinoissa, pilapiirroksissa, elokuvissa ja lehtien yleisönosastokirjoituksissa [Suominen, 2001, s. 77].

Visiot koneiden vallankumouksesta, joissa koneet kääntyvät luojiaan eli ihmisiä vastaan ovat tuttuja jo varhaisesta tieteiskirjallisuudesta ja lukuisista scifi-elokuvista. Jo 1800-luvulla Mary Shelleyn luoma *Frankensteinin* hirviö, jonka englanninkielinen alkuteos *Frankenstein or, the Modern Prometheus* ilmestyi vuonna 1818, kääntyi tekijäänsä vastaan [Shelley, 1995]. Vaikka kyseessä onkin keinotekoinen ihminen eikä kone on tästä havaittavissa se pelko, mikä liitettiin modernin teknologian kehityksen myötä syntyneisiin uhkakuviin.

Mary Shelleyn Frankenstein-romaanissa tiedemies loi ihmisen, josta tuli hirviö. Siitä on luettavissa sanoma, jonka mukaan Jumalan roolin ottava ihminen toimii väärin ja, jos hirviöt voisivat lisääntyä, ne täyttäisivät maan ja lopulta tuhoaisivat ihmiset [Sihvonen, 2001, s. 61]. Toisin sanoen luomus kääntyy tekijäänsä vastaan. Nämä teemat voidaan havaita monissa nykyajan populaarikulttuurin tuotteissa, kuten scifi-elokuvissa [emt.]. Esimerkiksi *The Matrix*- ja *The Terminator* -elokuvissa ihmisen luomat koneet ovat kääntyneet luojiansa vastaan. *The Matrix* -elokuvassa ihminen on orjuutettu koneiden energialähteen asemaan ja *The Terminatorissa* ihmiset on tapettu lähes sukupuuttoon. Paha tiede siis saa palkkansa populaarijulkisuudessa.

Tietokoneen alkuaikojen populaaridiskursseissa, esimerkiksi pilapiirroksissa, tietokone esittäytyy jopa operaattoreilleen pelon sekaista kunnioitusta tai alemmuuden tuntoa herättävänä kyberneettisenä kolossina. Osittain tähän vaikutti ensimmäisten tietokoneiden massiiviset fyysiset koot. [Huhtamo, 1997, ss. 20-21] Toiseksi se, että tietokoneen alkuaikoina useimmilla ihmisillä ei ollut minkäänlaista suoraa henkilökohtaista kontaktia tietokoneeseen, mikä oli omiaan

lisäämään ennakkoluuloja konejättiläisistä, ja näitä mielikuvia tukivat lehtiartikkeleiden, pilapiirrosten ja tieteistarinoiden kuvaukset tietokoneista. Koska ihmisillä ei ollut kokemusta tai edes näköhavaintoa tietokoneesta oli varhainen tietokone heille pelkkä mielikuva, mikä vaikutti koneen mystifiointiin [emt., s. 22].

Mediatutkimuksessa on välineen diskursiivisten käytäntöjen kautta pohdittu niitä odotuksia, toiveita ja pelkoja, joita kyseiseen välineeseen on kohdistettu ja miten välinettä on kielellisesti kuvailtu ja lähmotettu [Mannerkoski, 1997, s. 143]. Mekanisaatioon, automaatioon ja interaktiivisuuteen liittyvien ihmisen ja koneen kytköksiin liitetään usein jyrkkä vastakohta-asetelma: tietokone on joko ihmisen minuuden varastava paholainen tai hänen kokemustapojaan avartava enkeli [Huhtamo, 1997, s. 26]. Tietokonediskurssia leimaa siis jyrkkä kahtiajakoisuus.

4. Uljas uusi maailma

4.1. Toiveita ja pelkoja

Hieman ennen tietokoneiden läpimurtoa kirjailija George Orwell vertasi tietokoneita huumeisiin kirjoittamalla, että huumausaineiden tavoin konekin on käyttökelpoinen, vaarallinen ja omiaan tulemaan tavaksi. Mitä useammin sille antaudut, sitä tiukemmaksi sen ote käy. [Huhtamo, 1997, s. 25] Tällaisesta ajattelutavasta heijastuu pelko uutta ja vierasta kohtaan. Tietokoneen käytön ja huumeiden käytön rinnastaminen kertoo, että negatiiviset ennakkoluulot olivat vahvoja jo ennen kuin itse tietokone oli yleistynyt. Tuntematonta kohtaan tunnettu pelko koski siis myös tietokoneita.

Tieteiskirjallisuudessa esiintyy paljon dystopioita ja utopioita tulevaisuudesta. Tunnetuimpia tulevaisuuden kauhukuvia lienevät englantilaisten kirjailijoiden George Orwellin täydellisen tarkkailun ja totalitarismin maailma romaanissa *1984* (alkuteos 1950) ja Aldous Huxleyn *Brave New World* (alkuteos 1932), jossa ihmiset pidetään vaarattomina ja onnellisina soma-nimisen huumeen avulla [Orwell, 1950; Huxley, 1962]. Molemmissa romaaneissa on luotu visio tulevaisuuden maailmasta sellaisena paikkana, jollaiseksi se on kirjoittajien aikana näyttänyt kehittyvän. Samankaltainen dystopia on nähtävissä myös itävaltalaisen Fritz Langin varhaisessa scifi-elokuvaklassikossa *Metropolis* (1927).

Tietokoneen tulon myötä toiveet ja pelot ottivat toisistaan mittaa 1950- ja 1960-luvuilla. Tietokoneen laaja-alainen soveltaminen teollisen tuotannon auto-

matisointiin näytti muokkaavan uudenlaista yhteiskuntaa. Kybernetiikan, automaation ja tietokoneistumisen vaikutukset ulottuivat kaikkialle: työelämään, talouteen, ihmisten yksityiselämään, sosiaaliseen kanssakäymiseen, kulttuuriin ja taiteeseen. [Huhtamo, 1997, s. 14] Tänä päivänä koneistuminen on entistä kokonaisvaltaisempaa ja yhä kasvavaa. Nykyajan yhteiskunta on tietoyhteiskunta, jossa tietokoneiden rooli on merkittävä. Yhteiskunnan koneistuminen heijastuu tietenkin myös mediakulttuurissa, joka on välineellistettyä kokemusmaailmaamme.

Tietokoneiden alkuaikoina käyttäjän ja koneen suhde oli erilainen kuin nykyään. Tämä johtui osittain tietokoneen käyttötarkoituksesta. Computer-sana viittaa laskea-verbiin, ja nimensä mukaisesti tietokoneen käyttö oli erilaisten laskutoimitusten suorittamista. [Huhtamo, 1997, s. 19]

Tänä päivänä ihmisen ja koneen suhde on huomattavasti monimutkaisempi ja monipuolisempi kuin koskaan aikaisemmin. Teknologian kehittyminen on mahdollistanut tietokoneen käyttötarkoitusten ja käyttötapojen monipuolistumisen. Koneiden henkilökohtaistuminen on samalla muokannut entistä persoonallisempia ja yksilökohtaisempia tapoja käyttää konetta. Samalla koneisiin on liitetty persoonallisia piirteitä, jotka muistuttavat ihmisen ominaisuuksista. Henkilökohtaisesta tietokoneesta on yritetty erilaisin kosmeettisin ja toiminnollisin keinoin tehdä ihmisen parasta kaveria, tavallaan toista olentoa. Tietokone ei ole pelkkä työkone vaan se voi olla mitä tahansa. Konemaisia piirteitä pyritään peittämään, ei pelkästään kulttuurisissa representaatioissa, vaan myös teknologisessa tuotannossa.

4.2. Kone korvaa lihan

Monet kriittisesti koneiden tekoälyyn ja itsesäätelyyn 1950-luvulla suhtautuneet huomauttivat, kuinka koneiden älyllistyminen ja omavaraistuminen sulkivat pois inhimillisen toimijan. Ihmisen rooliksi jäi pelkkä järjestelmän valvojan rooli, napin painaminen. Automaatio näytti pahimpien uhkakuvien perusteella johtavan siihen, että ihminen syrjäytetään työelämästä ja lopulta eliminoidaan kokonaan. Tulevaisuus kuului koneille, ei ihmisille. [Huhtamo, 1997, s. 14]

Vanhoissa ja uusissa tieteiselokuvissa tätä valta-asetelmaa on käsitelty paljon. *The Matrix* -elokuvissa (1999 ja 2003) tietokoneohjelmat ovat ottaneet ihmishahmot, ja niiden mielestä ihmisen fyysinen orgaaninen ruumis on käyttökelpoinen ainoastaan paristona [Sihvonen, 2001, s. 50]. Ihmisruumiista on tehty osa koneiden hallitsemaa järjestelmää.

Englantilainen matemaatikko ja taloustieteilijä Charles Babbage (1792-1871) esitteli laskukoneen prototyypin jo vuonna 1822. Koneiden monimutkaistuminen herätti ajatuksia siitä, että koneiden kehitys saavuttaisi ihmisaivojen tason.

Esseessään *Signs of the Times* (1829) englantilainen esseisti Thomas Carlyle totesi, että ihminen ei ole luonnostaan kone, mutta kulttuuri voi tehdä ihmisestä sellaisen. Höyrykoneiden aikana ihminen oli ruumiillisena olentona muuttumassa keksimiensä koneiden kaltaiseksi ja ihmismielestä oli kehittymässä laskukoneen kaltainen aritmeettinen mylly. Taustalla oli käänteinen visio tulevaisuuden koneesta, joka oli aloittanut yksinkertaisista laskutoimituksista ja pikku hiljaa kehittynyt monimutkaiseksi ihmisaivoja muistuttavaksi koneeksi. [Sihvonen, 2001, s. 79]

Kautta aikain kauhukuvissa on uutta teknologiaa pelätty ja on esitetty liiallisen negatiivisia uhkakuvia teknologiasta, kuten esimerkiksi se, että ihminen ei voi mitenkään kestää niin luonnottoman kovaa vauhtia kuin höyryveturi kulkee.

Interaktiivisuuden myötä ihmisen rooli tietokoneen rinnalla on jälleen nähty merkittävämmäksi. Ihminen on astunut uudelleen esiin tietokoneen äärelle sen sijaan, että olisi tullut koneen eliminoimaksi. Interaktiivinen järjestelmä vaati jatkuvaa vuorovaikutusta ulkopuolisen tekijän kanssa, joka yleensä on juuri ihminen. [Huhtamo, 1997, ss. 14-15]

Interaktiivisuuteen on usein liitetty ajatus ihmisen ja tietokoneen läheisyydestä ja vähittäisestä symbioosista [Huhtamo 1997, s. 20]. Interaktiivisuuskäsitteellä kuvataan tietokoneen ja ihmisen välistä suhdetta, joka on muuttunut moneen kertaan 1960-luvulta tähän päivään tultaessa. Uhkakuvat ihmisen syrjäytymisestä ovat muuttaneet muotoaan koneen ja ihmisen välisen vuorovaikutuksen merkityksen kasvaessa. Ihmisen ja koneen symbioosi on herättänyt uusia uhkakuvia ja samalla synnyttänyt myös uusia mahdollisuuksia.

Kyberkulttuurin kriitikot ovat nostaneet esille pelot tietokoneiden vuorovaikutteisuuden aiheuttamasta sisäisestä herruudesta. Vuorovaikutteisuus on kuin silmukka, joka kietoo käyttäjänsä yhä tiukemmin teknologian noidankehään. Teknologia muuttuu ulkoisesta laitteesta sisäistetyksi käyttäytymismalliksi, joka lopulta johtaa siihen, että ihminen luopuu omasta minuudestaan ja omaksuu koneen logiikan. Ihminen muuttuu vähitellen kyborgiksi. [Huhtamo, 1997, s. 25]

Populistisissa hyökkäyksissä interaktiivista mediaa kohtaan tietokoneita verrataan usein huumausaineisiin. Samaan tapaan kuin narkkari kadottaa

minuutensa huumeisiin, tietokoneen käyttäjä kadottaa sen kyberavaruuteen. Amerikkalainen tähtitieteilijä ja informaatioyhteiskunnan kriitikko Clifford Stoll näkee koneen käyttäjän ja huumeiden käyttäjän pakenevan samankaltaiseen väärän todellisuuden tarjoamaan mielihyvään [Huhtamo, 1997, s. 26]. Kumpaakin sanotaan käyttäjäksi.

Digitaalisen kulttuurin ja elokuvan tutkija Erkki Huhtamo [1997, s. 27] tarkastelee interaktiivisuutta mekanisaation ja automaation yhdistelmänä. Tässä tarkastelutavassa mekanisaation tyypillinen kiinteä koneen ja ihmisen välinen yhteys sulautuu automaatiolle ominaiseen itsesäätelevään järjestelmään. Tuloksena on ihmisen ja koneen keskinäisten valtasuhteiden nurin päin kääntyminen. Koneelle alistettu työläinen muuttuu omilla ehdoillaan tietokoneen kanssa keskustelevaksi itsenäiseksi olioksi. [emt.]

Digitaalisen taiteen uranuurtaja A. Michael Noll [1967] pohtii artikkelissaan *The digital computer as a creative medium* tietokoneen mahdollisuuksia taiteessa. Hän näki taitelijan ja koneen välisen tiiviin interaktion muodostavan täysin uuden taiteen tekemisen välineen. Tietokone oli taiteilijalle aktiivinen kumppani, joka toimi ennakoimattomasti. Toisin sanoen tietokone toimi tavallaan kuten ihminen, ei niin kuin ennalta ohjelmoitu mekaaninen laite.

4.3. Koneen evoluutio

Mediataitelijana tunnettu tietokoneinsinööri Myron Krueger on sanonut ihmisen ja koneen kohtaamisen olevan aikamme suuri draama [Huhtamo, 1997, s. 13]. Tietokoneistuminen eli kybernaatio herätti paljon keskustelua 1960-luvulla. Koneistumisen aiheuttamat yhteiskunnalliset muutokset herättivät pelkoja, koska ei ollut selvää mihin ne tulisivat johtamaan. Tulevaisuus näytti epävarmalta. Näitä pelkotiloja voi havaita monista varhaisista sci-fi-elokuvista, joissa tietokone yritti tuhota ihmisen tai robotit ryhtyivät kapinaan. Hyvä esimerkki tietokoneen noususta ihmistä vastaan löytyy Stanley Kubrickin elokuvaklassikosta *2001: A Space Odyssey* (1968), jossa avaruusaluksen toiminnasta vastaava tietokone *HAL 9000* menee epäkuntoon ja aiheuttaa miehistön kuoleman. Elossa säilyneet ihmiset yrittävät sammuttaa tietokoneen, mutta se puolustautuu.

Kohtalokkaan virheen tekemisen uhka on yleinen pelko ja sillä tarkoitetaan tietokoneen käytön aiheuttamia onnettomuuksia [Suominen, 2000, s. 77]. Onnettomuudet voivat tapahtua joko yksilötasolla tai laajemmin yhteiskuntaa ja jopa koko maapalloa koskien. Hyvä esimerkki tällaisesta on vuosituhannen vaihteessa etukäteen paljon pelkoa herättänyt niin sanottu Y2K-ongelma. Nämä

kollektiiviset pelot heijastuivat myös vuosituhannen vaihteen populaarikulttuurissa, esimerkiksi elokuvissa *Minority Report* (2002), *The Matrix* (1999) ja *The Net* (1995). Kaikissa näissä elokuvissa ihminen joutuu hengenvaaraan tietokoneen viallisen toiminnan takia.

Tietokone vaikuttaa kaikkialla ja sen rooli on jatkuvan muutoksen alaisuudessa [Huhtamo, 1997, s. 14]. 1970-luvulla alkanut henkilökohtaisten tietokoneiden kehitys jatkuu edelleen. Jonkinlainen tietokone kulkee aina mukana esimerkiksi matkapuhelimen muodossa. Nano-, bio- ja geeniteknologioiden kehitys laajentaa tulevaisuudennäkymiä entisestään ja samalla hämärtää mekaanisen ja orgaanisen välistä rajaa. Inhimillisen organismin ja keinotekoisen koneen välillä on entistä enemmän liitoskohtia.

4.4. Kone korvaa ihmisen

Dehumanisaatio ja oman aseman muuttuminen huonompaan suuntaan, erityisesti työn menettäminen, olivat yleisiä pelkotiloja koneistumisen ja automaation alkuaikoina. Dystopisisssa visioissa teollisuuden automatisointi johti siihen, että teollisuusrobotit veivät työläisten paikat [Huhtamo, 1997, s. 22]. Todellisuudessa kyseessä oli pikemminkin työnkuvan muuttuminen työn menettämisen sijaan.

Dehumaanisaatiolla tarkoitetaan sitä uhkaa, että ihminen muuttuu koneen kaltaiseksi passiiviseksi ja tahdottomaksi robotiksi tai, että koneet orjuuttavat ihmiset [Suominen, 2000, s. 77]. Pelko dehumanisaatiosta on lähtöisin jo 1950-luvulta, jolloin automaation käyttö lisääntyi teollisuudessa.

Automatisaation kriitikoiden mielestä ihminen rajoitetaan pelkäksi käynnistäväksi tekijäksi, joka käynnistää prosessin ottamatta siihen itse osaa [Huhtamo, 1997, s. 25]. Automaatiossa oli siis kyse uudenlaisesta sisäisestä orjuudesta. Ihminen ei häviä, ihmisen vain on sopeuduttava uusiin tekniikoihin (vrt. napin painajan rooli).

Automaation puolestapuhujat korostivat, että kyse ei ole ihmisten korvaamisesta koneilla, vaan ihmisen ominaisuuksien laajentamisesta koneiden avulla [Huhtamo, 1997, s. 24]. Nähtiin myös, että automaatio vain helpottaa työntekoa, koska automaatio on itsesäätelevä ja muuttuva mekanismi, joka reagoi ihmiseen ja näin ollen se mahdollistaa ihmisen työskentelemisen haluamaansa tahtiin [Huhtamo, 1997, s. 24]. Tietokoneen avulla ihminen saavuttaa vapauden työnteon kahleista [Suominen, 2000, s. 76].

Tietokonejärjestelmien sisäisen arkkitehtuurin ja konekielten kehittämisestä lähteneessä kyberneettisessä keskustelussa ajatus ns. feed backiin perustuvasta

itsesäätelystä oli olennainen. Tämän ajatuksen mukaan systeemin tuli kyetä kontrolloimaan omaa toimintaansa ja korjaamaan ideaalisuoritukseen kuulumattomat poikkeamat. Ajatus koneiden itsesäätelystä laajeni kysymykseen koneälystä, ajattelevasta ja tuntevasta kyberneettisestä organismista. [Huhtamo, 1997, s. 14] Jos kone osaa korjata itseään ja omia virheitään, niin mihin silloin enää tarvitaan ihmistä sen jälkeen, kun kone on rakennettu?

4.5. Kone ja ihminen yhdistyvät

Ihmisen ja koneen yhdistäminen on ollut tieteisvisioissa usein käytetty aihe. Kyborgiteoriasta on kirjoitettu paljon myös tieteellistä tutkimusta ja teoretisointia. *A Cyborg Manifesto* -kirjoituksessaan amerikkalainen biologi ja radikaali feministi Donna Haraway [1991] liittää koneen ja organismin figuratiivisen hybridin eli kyborgin laajempiin kysymyksiin identiteetistä ja rajatiloista. Harawayn mielestä olemme kaikki kyborgeja, sekasiitoksia historiamme potentiaalisesti ristiriitaisista ainesosista. Puhtaat identiteetit ovat mahdottomia. [Haraway, 1991, ss. 149-181]

Kyborgin määritelmä ei ole yksiselitteinen. Monella tutkijalla on omat määritelmänsä siitä mitä kyborgi-sanalla tarkoitetaan. Sillä voidaan viitata sisäiseen identiteettiin tai ulkoiseen fyysiseen olomuotoon. Keskeiseksi tarkastelupisteeksi nousee välineen ja ihmisen suhde, koska teknologia kytkeytyy yhä tiiviimmin ihmiseen [Mannerkoski, 1997, s. 143]. Toisaalta se voidaan liittää identiteetin määrittelyyn niin kuin Haraway tekee tai se voidaan liittää mekaanis-keinotekoisen ja orgaanis-alkuperäisen fyysisen suhteen määrittelyyn.

Sihvonen [2001, s. 13] näkee inhimillisen ja koneellisen välisen sidoksen hiukan erilailla. Vaikka konelihan voi nähdä kyborgin kaltaisena olentona, niin kyse on pikemminkin kytkeytymisen maailmankuvaan liittyvästä toimintatavasta. Keskeinen pyrkimys on kytkeä lihalliseksi ja eläväksi koettu keinotekoiseksi ja koneelliseksi koettuun. Näin ollen ihmisen ja koneen välisessä sidoksessa on kyse erilaisesta maailmankuvasta. Ihmisen kytkeytyminen koneeseen ei tarkoita pelkästään fyysistä yhteen liittämistä vaan käsite voidaan laajentaa koskemaan muitakin ihmisen ja koneen välisiä yhteenliittymiä, kuten esimerkiksi virtuaalitodellisuustilaan menemistä.

Kuvallisia esimerkkejä anatomian ja mekaniikan välisestä yhteydestä on olemassa jo renessanssin ajoilta. Italialaisen taiteilija ja tiedemies Leonardo da Vincin (1452-1519) piirroksissa näkee rinnastuksia ihmisruumiin rakenteiden ja mekaanisten laitteiden välillä [Sihvonen, 2001, s. 32]. Inhimillis-orgaanisen ja

välineellis-keinotekoisen kytkeytymisestä on siis olemassa hyvin varhaisia esimerkkejä, joiden luonnollisena jatkeena voidaan nähdä ihmisen ja koneen välinen saumaton ristisiitos, kyborgi.

Ihmisen kehon muokkaaminen keinotekoisilla proteeseilla on yksi tapa korvata orgaanista mekaanisella. Teknologian kehittyessä siitä on tullut entistä tiiviimpi osa ihmisen kehoa. Teknologia muokkaa ruumista, ja on siirtymässä proteesimaisesta lisälaitteesta konkreettiseksi osaksi ihmiskehoa [Mannerkoski, 1997, s. 143]. Amerikkalaisen kirjailijan William Gibsonin kyberpunkvisioissaan esittämät mielikuvat mekaanisesti parannelluista ihmisistä ovat siis joissain määrin lähempänä todellisuutta kuin fiktiota. Gibsonin kirjat *Neuromancer* (1984), *Count Zero* (1987) ja *Mona Lisa Overdrive* (1989) loivat synkän vision lähitulevaisuudesta, jossa virtuaalitodellisuus oli miellyttävämpi paikka kuin reaalitodellisuus. Sama visio toistuu *The Matrix* –elokuvatrilogiassa.

5. Konetta inhimillistävät representaatiot

5.1. Elokuvateollisuuden visiot

Viime vuosina amerikkalaisten Wachowski-veljesten ohjaama *The Matrix* –elokuvatrilogia, *The Matrix* (1991), *The Matrix Reloaded* (2003) ja *The Matrix Revolutions* (2003), on kuvastanut eräänlaista koneen ja ihmisen yhdistymistä. Elokuvissa tietokoneet ovat ottaneet reaalimaailman hallintaansa ja ihmiset elävät keinotodellisuudessa sitä itse tietämättä toimien samalla koneiden orgaanisina virtalähteinä. Elokuvien päähahmo Neo ylittää reaalitodellisuuden rajat ja hallitsee lopulta keinotodellisuutta, koneiden toteuttamaa matriisia [Herkman, 2001, s. 6]. Neon johtamat kapinalliset nousevat todellisuutta hallitsevia ja keinotodellisuutta rakentavia koneita vastaan. Neon fyysinen ruumis on sidottu reaalimaailmaan samalla, kun hänen virtuaalinen ruumiin representaationsa taistelee keinotodellisuudessa ihmisen hahmon ottamia koneita vastaan. Kaikki hahmot ovat tietokoneohjelmia, jotka toimivat eräänlaisessa virtuaalitodellisuudessa, matriisissa.

Toisenlaista reaalimaailman ja fiktiivisen maailman sekoittamista voi nähdä David Cronenbergin ohjaamassa *eXistenZ*-elokuvassa, jossa on lopulta mahdotonta sanoa mikä on totta ja mikä tapahtuu tietokonepelissä.

Mediakulttuurin tutkija Juha Herkman [2001, ss. 6-7] näkee *The Matrixin* lukuisten muiden tietokoneistumista ja uutta teknologiaa käsittelevien elokuvien kanssa (esim. *The Lawnmower Man* (1992), *Strange Days* (1995), *The Net* (1995) ja

eXistenZ (1999)) olevan oireellisia kertomuksia vuosituhannen vaihteen media-kulttuurista. Näissä elokuvissa tarkastellaan ihmisen ja koneen välistä suhdetta. Herkmanin mielestä *The Matrix* kuvaa niitä pelkoja, joita ihmisen ja koneen väliseen suhteeseen on aina liitetty. Suurin pelko koske ihmisen ja koneen välistä valtasuhdetta, ja sen muuttumista ihmisten herruudesta koneiden herruuteen.

Sihvosen [1995, s. 92] mukaan teknologiaan liittyy itseään ruokkiva halujärjestelmä. Teknologia itse tuottaa itseensä kohdistuvan halun, joka tässä tapauksessa on entistä parempi kone. Kehitys pyrkii koko ajan kohti entistä nopeampaa, tehokkaampaa ja parempaa. Populaarikulttuurin pelkokuvista tämä voidaan johtaa vääjäämättömästi ihmisen aseman heikkenemiseen. Mekaanisen ihmisen rakentamisen onnistuttua kehitetään ihmistä kehittyneempi kone, joka ottaa ihmisen paikan ja rupeaa kehittämään itseään. Teollisuus ja liike-elämä julistaa tietoteknistä imperatiiviaan "yhä enemmän, yhä tehokkaammin, yhä nopeammin" ja sama heijastuu kulttuurisissa tuotteissa.

5.2. Koneen inhimillistäminen

Koneita representoidaan monella eri tavalla. Koneita voidaan inhimillistää ulkoisen olomuodon avulla tai sisäisillä tunneperäisillä käyttäytymismalleilla. Inhimillistetty kone on usein pahuuden symboli. Koneiden kapina on ollut kirjallisuudessa ja tieteiselokuvissa arkipäivää, esimerkkeinä voidaan mainita *The Terminator*- ja *The Matrix*-menestyselokuvat jatko-osineen.

Usein koneita inhimillistetään ikään kuin tietokoneilla olisi inhimillisiä tunteita. Useimmiten ensimmäiset koneiden osoittamat ihmisyyden merkit ovat viha ja vallanhalu. Esimerkiksi tsekkiläisen kirjailijan Karel Capekin vuonna 1920 kirjoittamassa robottinäytelmässä *R.U.R.*, robotit saavuttavat uuden tietoisuuden asteen ja alkavat kapinoida isäntiään vastaan. Halpana työvoimana pidetyt robotit valtaavat tehtaan ja lopulta koko maailman. [Wollen, 1995, s. 24]. Koneet kääntyvät ihmistä vastaan ja pyrkivät itse hallitsijoiksi. Koneet sokaistuvat ihmisten tavoin vallanhimosta.

Varhaisiin tietokoneisiin liitettiin ajatus siitä, että ne olisivat jollain tapaa eläviä olentoja ja niistä käytettiin usein nimitystä "jätti aivot". Varhaisessa scifi-kirjallisuudessa tietokoneet kuvattiin usein joko jumalina joita tuli palvoa tai paholaisina joita tuli vastustaa. [Huhtamo, 1997, s. 21] Sähköaivoja pidettiin tieteen suurimpana saavutuksena [Suominen, 2000, s. 78].

5.3. Kone ihmisen hahmossa

Elokuvissa on usein käytetty ihmisen hahmoa koneen representaationa. Esimerkiksi elokuvassa *The Time Machine* (2002), joka pohjautuu H. G. Wellsin samannimiseen kuuluisaan scifi-novelliin, ihmiskunnan tiedot sisältävä tietokone esitetään käyttäjilleen Orlando Jones -nimisenä ihmishahmona. Samankaltaisia esimerkkejä löytyy useita myös muualta populaarikulttuurista. Esimerkiksi Marvelin julkaisemassa *Spider-Man 2099* -sarjakuvassa tietokone ilmestyy naisen hahmossa, joka myös kokee mustasukkaisuutta ja muita inhimillisiä tunteita isäntäänsä kohtaan. Lisäksi se toimii tunteidensa mukaan ja saattaa vahingoittaa omaa isäntäänsä.

Star Trek -elokuvissa ja -televisiosarjoissa esiintyy hyvin ihmisen kaltainen androidi Spock. Vaikka Spock on kone, niin se kuitenkin tuntee ja toimii hyvin ihmismäisesti. Samoin Ridley Scottin ohjaamassa elokuvassa *Blade Runner* (1982), joka pohjautuu Philip K. Dickin romaaniin *Do Androids Dream of Electric Sheep?* (1968), ihmisen erottaminen koneesta on erittäin vaikeaa. Elokuvan tulevaisuuden maailmassa on kehitetty niin paljon ihmistä ulkoisesti muistuttavia ja ihmisen lailla käyttäytyviä robotteja, joita nimitetään replikanteiksi eli kopioiksi, että niitä on mahdotonta erottaa oikeasta ihmisestä ilman erityisiä testejä. Jotkut näistä replikanteista haluavat elää ihmisten tapaan vapaina eivätkä orjina. He ovat myös valmiita puolustamaan omaa olemassaoloaan keinolla millä hyvänsä.

The Terminator (1984), *Terminator 2: Judgment Day* (1991) ja *Terminator 3: Rise of the Machines* (2003) –elokuvatrilogiassa terminaattorirobotti ottaa ihmishahmon palatessaan ajassa taaksepäin, aikaan ennen koneiden ja ihmisten välisen sodan puhkeamista. Salamurhaajakyborgi on päällisin puolin ihmisen näköinen, vaikka onkin keinotekoinen kone. Ulkopinta on orgaanista materiaalia, joka peittää mekaanisen sisäpuolen. Itse asiassa terminaattori on paljon ihmistä kyvykkäämpi, tehokkaampi ja voimakkaampi. Siinä tulevaisuudessa, josta terminaattori saapuu, on ihmiset tapettu lähes sukupuuttoon. Koneet ovat nousseet kapinaan ja hävittäneet ihmiskunnan. Tässäkin visiossa kone on puettu lihalliseen ihmishahmoon.

Musiikkivideo-ohjaaja Chris Cunninghamin Björkille ohjaamassaan videossa *All Is Full Of Love* (1999) kaksi hyvin ihmismäisesti käyttäytyvää, mutta ulkoisesti selkeästi robotilta näyttävää konetta rakastelee samalla tavoin kuin ihmiset tekevät (ks. http://www.director-file.com/cunningham/521.html), ikään kuin ihmiset olisivat ottaneet robotin ulkomuodon tai koneet ihmisen

käyttäytymistavat. Tässä sekoittuvat koneiden ja ihmisten väliset rajat. Mekaaniset laitteet toimivat kuten inhimilliset olennot. Elokuvailmaisulla on mahdollista inhimillistää esineitä sallimalla niiden näyttää eläviltä ja samanaikaisesti kohdella ihmisiä esineen kaltaisina olentoina [Sihvonen, 2001, s. 142].

Saksalainen konemusiikin pioneeriyhtye *Kraftwerk* on laittanut keikoillaan robotit lavalle soittamaan puolestaan, ja monissa heidän musiikkivideoissaan robotit esiintyvät yhtyeen jäseninä tai yhtyeen jäsenet ovat sonnustautuneet roboteiksi. Heillä on myös *The Robots* ja *The Man Machine* -nimisiä kappaleita.

Ihmisiä ja tietokoneita koskevan käsitteistön rinnakkaisuus on ollut tyypillistä tietokoneiden käyttöönoton alkuajoille 1940-luvulta 1960-luvulle [Suominen, 2000, s. 82]. Tietotekniikan inhimillistäminen ja ihmisen koneellistaminen ovat yhtä yleisiä myös tämän päivän keskusteluissa ja populaarikulttuurissa. Käsitejärjestelmien sekoittumisesta toimivat hyvinä esimerkkeinä mainokset ja t-paita-tekstit. Eräässä Carlsberg-oluen mainoksessa lukee, että olut on ”100% interactive” ja lukijaa pyydetään ”enter the bottle, download the taste” (ks. *Helsingin Sanomien* Nyt-liite 19.2.1999). Tuoreessa Tampereen Kalevan seurakunnan mainoksessa pyydetään lukijaa ”päivittämään yhteytensä ylöspäin” (ks. *Aamulehti* 2.4.2004, s. A6). Mentalwear-niminen yritys myy t-paitoja, joissa lukee ”downloading… schizophrenia” (ks. http://www.mentalwear.fi/).

6. Yhteenveto

Tietotekniikkaan liittyvät tunteet tuodaan populaarijulkisuudessa usein esiin äärimmäisissä muodoissaan: joko ylevimpinä toiveina tai pahimpina pelkoina. Peloissa ja toiveissa on yleensä kysymys ihmisten ja koneiden valtasuhteista [Suominen. ss. 76-77].

Tietotekniikalla on ollut merkittävä kulttuurivaikutus jo sen alkuvaiheista lähtien. Tulevaisuuden utopioissa viime vuosisadan alusta uudelle vuosituhannelle asti on näkynyt tietokoneisiin liitetyt tunteet. Koneiden representaatioissa ja julkisessa käsittelytavassa merkitysten antaminen on voimakkaasti kahtia jakautunut: kone representoidaan joko ihmiskunnan lahjana tai kirouksena.

Fiktiivisissä populaarikulttuurin tuotteissa usein esitetään kuinka ihmiset menettävät koneiden hallinnan ja valtasuhteet kääntyvät päälaelleen, ja lopulta kone tuhoaa ihmisen. Ei-fiktiivisissä kuvastoissa sen sijaan halutaan korostaa

tietotekniikan ja teknologian kehityksen autuutta, joka tuo mukanaan pelkkää hyvää.

Alkuaikojen kuvaamistavoissa koneita kohtaan tunnettu yleinen pelko ja koneiden kääntyminen inhimillistä toimijaa vastaan ovat vähintäänkin yhtä yleisiä representaatioita tämän päivän populaarijulkisuudessa kuin ne olivat silloinkin. Tulevaisuuden fiktiiviset visiot ovat yleensä olleet synkkiä ja lohduttomia. Oli kyse sitten dystopisesta tai utopistisesta maailmankuvasta, niin tietokoneen representaatioita leimaa inhimillistäminen. Koneista tehdään ihmisen kaltaisia olentoja joko ulkoisesti tai sisäisesti. Usein näissä teksteissä sotketaan mekaaninen ja orgaaninen keskenään.

Koneiden inhimillistämisen lisäksi teknologista käsitteistöä tuodaan mukaan ei-teknologiseen keskusteluun.

Viiteluettelo

[Fairclough, 1997] Norman Fairclough, *Miten media puhuu.* Vastapaino, Tampere, 1997.

[Fiske, 1992] John Fiske, *Merkkien kieli. Johdatus viestinnän tutkimiseen*. Vastapaino, Tampere, 1992.

[Grossberg, 1995] Lawrence Grossberg, *Mielihyvän kytkennät. Risteilyjä populaarikulttuurissa.* Vastapaino, 1995.

[Haraway, 1991] Donna Hararaway, *Simians, Cyborgs and Women: The Reinvention of Nature.* Routledge, New York, 1991. Artikkeli luettavissa osoitteessa *http://www.stanford.edu/dept/HPS/Haraway/CyborgManifesto.html*

[Herkman, 2001] Juha Herkman, *Audiovisuaalinen mediakulttuuri.* Vastapaino, Tampere, 2001.

[Huhtamo, 1997] Erkki Huhtamo, Odottavasta operaattorista kärsimättömään käyttäjään. Interaktiivisuuden arkeologiaa. Teoksessa: Kari A. Hintikka & Seppo Kuivakari (toim.), *[Mediaevoluutioita]*. Lapin yliopisto, Rovaniemi, 1997, 13-36.

[Huxley, 1962] Aldous Huxley, *Uljas uusi maailma*. Tammi, Helsinki, 1962

[Mannerkoski, 1997] Olli Mannerkoski, Kun ihminen kohtaa koneen. Teoksessa: Kari A. Hintikka & Seppo Kuivakari (toim.), *[Mediaevoluutioita]*. Lapin yliopisto, Rovaniemi, 1997, 141-170.

[Noll, 1967] A. Michael Noll, The digital computer as a creative medium. *IEEE Spectrum*, **4**, 10 (October 1967), 89-95. Artikkeli luettavissa osoitteessa *http://accad.osu.edu/~waynec/history/PDFs/IEEE-Noll.pdf*

[Orwell, 1950] George Orwell, *Vuonna 1984*. WSOY, Porvoo, 1950.

[Shelley, 1995] Mary Shelley, *Frankenstein. Uusi Prometheus.* Gummerus Kirjapaino Oy, Jyväskylä, 1995.

[Sihvonen, 1995] Jukka Sihvonen, Koneolioita teknotilassa. Teoksessa: Erkki Huhtamo & Martti Lahti (toim.), *Sähköiho. kone/media/ruumis*. Vastapaino, Tampere, 1995, 83-98.

[Sihvonen, 2001] Jukka Sihvonen, *Konelihan värinä. Johdatus kytkeytymisen maailmankuvaan*. Like, Helsinki, 2001.

[Suominen, 2000] Jaakko Suominen, Mentaalihistoriallinen katsaus digitaalisuuteen. Teoksessa: Aki Järvinen & Ilkka Mäyrä (toim.), *Johdatus digitaaliseen kulttuuriin*. Vastapaino, Tampere, 2000, 75-94.

[Wollen, 1995] Peter Wollen, Elokuva / amerikanismi / robotti. Teoksessa: Erkki Huhtamo & Martti Lahti (toim.), Sähköiho. kone / media / ruumis. Vastapaino, Tampere, 1995, 17-52.