Tampereen yliopisto
Tietojenkäsittelyopin laitos

Käki, Mika: Epäsuora hallinta käyttöliittymässä

Pro gradu -tutkielma, 118 sivua
Elokuu, 1999

TIIVISTELMÄ

Eräs mahdollinen tapa parantaa tietokoneohjelmien käytettävyyttä on lisätä niihin epäsuorasti hallittua automaattista toiminnallisuutta. Skenaariossa ohjelma tekee työn, käyttäjälle esitetään valmiit tulokset.

Saaduista kokemuksista voidaan ymmärtää, etteivät käyttäjät pysty tai halua käyttää ohjelmia, jos heille tarjotaan yksinomaan niiden tuottamia tuloksia. Käyttöliittymän tulisi tarjota mahdollisuus ymmärtää ohjelman toimintaa laajemmin. Tavalliset käyttöliittymäsuunnittelumenetelmät eivät sovellu käytettäväksi sinällään eikä yhtenäistä näkemystä epäsuorasta hallinnasta tai siihen liittyvästä käyttöliittymästä ole olemassa.

Tässä tutkimuksessa muodostetaan teoriaa epäsuorasta hallinnasta sekä siihen liittyvästä käyttöliittymästä. Epäsuora hallinta on työn keskeisin käsite, jolle esitetään työn alussa määritelmä perustuen käsiteanalyysiin ja nykyiseen aihetta koskevaan tutkimukseen.

Käyttöliittymäkysymyksiin pureudutaan kartoittamalla epäsuoran hallinnan toteutustekniikoita ja niiden piirteitä. Näiden ja aiempien tutkimustulosten perusteella selvitetään millaisia vaatimuksia epäsuoran hallinnan käyttöliittymälle olisi asetettava. Konkreettisten vaatimusten avulla arvioidaan jo esitettyjä käyttöliittymäratkaisuja.

Lopuksi muodostettua teoriaa konkretisoidaan esittämällä se suunnitteluperiaatteiden muodossa. Periaatteita sovelletaan esimerkinomaisesti erään tiedonhakuagentin käyttöliittymän suunnittelussa, jonka tulos kuvataan. Suunnitteluperiaatteiden keskeinen idea on epäsuorasti hallittun toiminnallisuuden havainnollistus animaatiolla tavallisessa käyttöliittymässä, jota voidaan käyttää myös tavalliseen tapaan.

SISÄLLYS

1. JOHDANTO

Ihmisen ja tietokoneen vuorovaikutus (*HCI, Human-Computer Interaction*) on tutkimussuunta tietojenkäsittelyopin sisällä, joka lisää tietojenkäsittelyn tutkimukseen käyttäjän näkökulman. Tutkimussuuntauksen tavoitteena on ymmärtää ihmisen ja tietokoneen vuorovaikutuksen luonnetta niin, että järjestelmien käyttämisestä osattaisiin saadun tiedon perusteella tehdä käyttäjille luontevampaa. Luontevuus on HCI-alalla tavallisimmin ymmärretty ohjelman nopeana ja vaivattomana oppimisena ja toisaalta sen jatkuva käytön sujuvuutena.

Ihmisen ja tietokoneen vuorovaikutuksen tutkimuksessa ohjelman käyttöliittymä nousee tarkastelun keskiöön. Käyttöliittymä on se osa järjestelmää, jonka välityksellä ihminen ja tietokoneohjelma ovat tekemisissä keskenään. Sen välityksellä ihminen antaa ohjelmalle komentoja ja ohjelma esittää komentojen suorituksen tuottamat tulokset käyttäjälle. Tavat, joilla käyttäjä komentoja antaa tai ohjelma tuloksensa esittää, toimivat eräinä käyttöliittymien luokitteluperusteina.

Erääksi potentiaaliseksi tavaksi helpottaa ohjelmien käyttöä on esitetty käyttöliittymän toiminnallisuuden osittaista automatisointia. HCI-tutkimussuuntauksen sisälle onkin muodostunut uusia tutkijayhteisöjä, jotka ovat kiinnostuneita tällaisista ratkaisumalleista. Erityisesti agenttiohjelmia (S*oftware Agent, Agent Program*) ja älykkäitä käyttöliittymiä (*IUI, Intelligent User Inteface*) tutkivat ryhmät ovat valinneet strategiakseen toimintojen automatisoinin ja tutkivat siihen liittyviä tekniikoita ja toteutustapoja.

Toimintojen automatisoinnissa on yhtenä tutkimuskysymyksenä itse automatisoinnin toteutus eli se, miten ohjelmat voidaan rakentaa siten, että ne kykenevät itsenäisesti ongelmia ratkomaan. Toinen kysymys on se, miten automaattista prosessia hallitaan; miten sille annetaan ratkaistavat ongelmat ja miten toimintaa voidaan kontrolloida ja säädellä.

Ensimmäistä kysymystä automatisoinnin toteutuksesta on tutkittu hyvin runsaasti. Osaltaan se johtuu varmaankin siitä, että tutkimusala on tuonut uusia sovellusalueita aiemmin kehitetyille tekniikoille, kuten tekoälylle (*AI, Artificial Intelligence*), verkkoohjelmoinnille, hajautetuille järjestelmille, tiedon haulle (*IR, Information Retrieval*) ja suodattamiselle (*IF, Information Filtering*).

Automaattisen toiminnan hallinnan puolella tutkimus on keskittynyt antropomorfisten käyttöliittymien tarkasteluun ja kehittelyyn. Antropomorfisessa käyttöliittymässä osa toiminnallisuudesta kuvataan inhimillistetyn hahmon avulla, jonka käyttäytyminen pyrkii jäljittelemään ihmisen käyttäytymistä jollakin tasolla. Esimerkiksi

luonnollisen kielen käyttö tai ilmeisiin ja eleisiin perustuva kommunikointi ovat inhimillisiä piirteitä, joita tällaisissa käyttöliittymissä on tyypillisesti käytetty [esim. Koda 96, Towns *et al.* 98, André *et al.* 98, Johnson 98].

Antropomorfinen lähestymistapa on kuitenkin osoittautunut ongelmalliseksi. Ainakin toistaiseksi esitetyt toteutukset ovat tyypillisesti aluksi hauskoja, sitten yksitoikkoisia ja lopulta ne tuntuvat häiritseviltä [Shneiderman 95]. Toinen ongelma liittyy havaittuun älykkyyteen. Mitä realistisemman näköinen käyttöliittymässä toimiva hahmo on, sitä suurempia odotuksia ihmiset näyttävät asettavan sen älykkyydelle [Koda 96]. Odotuksissa ei sinänsä ole tietenkään mitään vikaa, mutta ne aiheuttavat pettymyksiä ja väärinkäsityksiä, koska ohjelmat eivät pysty vastaamaan korkeisiin odotuksiin [Maes and Shneiderman 97].

Koska antropomorfinen ratkaisumalli on edellä mainittujen kokemusten perusteella kyseenalaistettava, on automatisointiin perustuvien käyttöliittymien tutkimus itseasiassa heikoissa kantimissa. Nimenomaan automaattisen toiminnon *käyttöliittymään* keskittyvää tutkimusta on tehty käytännössä vain antropomorfisesta näkökulmasta, joten vaihtoehtoisia ratkaisuja ei ole systemaattisesti tutkittu. Toisaalta käyttöliittymän keskeinen merkitys osin automaattisesti toimivien ohjelmien (kuten agenttiohjelmien) hyödyllisyydelle on tunnustettu [Maes and Shneiderman 97], mikä entisestään korostaa tutkimuksen yksipuolisuudesta johtuvaa ongelmaa.

Tämä tutkimus ottaa ongelmaan kantaa keskittämällä tutkimuksen erityisesti automaattisen toiminnon käyttöliittymään. Tarkastelua jäsentäväksi käsitteeksi on valittu epäsuora hallinta, joka on tutkimusta ajatellen automaattista toimintoa tarkempi käsite. Tutkimuksen tavoitteena on:

1. Muodostaa selkeä käsitys siitä, mitä on epäsuora hallinta.
2. Analysoida käsitettä käyttöliittymän kannalta ja esittää sen käyttöliittymälle asettamia tavoitteita ja vaatimuksia.
3. Kartoittaa nykyisiä käyttöliittymäratkaisuja ja arvioida niitä.
4. Muotoilla edellä mainitut pohdinnat ymmärrettävään ja sovellettavaan muotoon.

Tavoitteiden viimeinen kohta toteutetaan muodostamalla edeltävän tietouden perusteella suunnitteluperiaatteita. Suunnitteluperiaatteiden soveltamisesta annetaan esimerkki, joka tuo periaatteet ja työn koko teoreettisen käsitteistön konkretian tasolle. Arvioitavana on toimiva käyttöliittymä esimerkkisovellukseen.

Tehty tutkimus voidaan jakaa kahteen osuuteen: tutkivaan ja soveltavaan. Työn kolme ensimmäistä lukua muodostavat yhdessä tutkivan osuuden. Tutkivan osuuden ensimmäisen luvun ‘Epäsuora hallinta’ alussa määritellään tutkimusta jäsentävä käsite, jonka avulla tarkastelu suoritetaan. Käsitteellä pyritään saamaan tarkasteluun yleisyyttä, mikä voisi olla ongelmallista, jos käytettäisiin ainoastaan tässä johdannossa viljeltyä automaattinen toiminto -käsitettä. Käsitteen esittelyn jälkeen seuraavassa luvussa keskitytään epäsuoran hallinnan toteuttavien tekniikoiden pohtimiseen, millä kerätään tämän vuorovaikutustavan erityspiirteitä.

Seuraavassa luvussa käsitellään epäsuoran hallinnan toteuttamista ja suurennuslasin alla ovat teknologiat, joita epäsuoran hallinnan toteuttamiseksi on käytetty. Tarkastelulla pyritään tekniikoiden

lisäksi kartoittamaan niiden piirteitä, jotka vaikuttavat epäsuoraa hallintaa soveltavaan käyttöliittymään.

Seuraavassa luvussa ‘Käyttöliittymävaatimukset’ on lähtökohtina edellisissä luvuissa hahmotellut vuorovaikutustavan ja sen toteuttavien tekniikoiden erityspiirteet. Luvussa niitä tarkastellaan käyttöliittymän kannalta kysyen: millaisia vaatimuksia epäsuora hallinta ja sen toteutustekniikat käyttöliittymälle asettavat? Samalla hahmottuu myös se, millaisiin eri toimintoihin käyttöliittymässä voidaan tai on syytä ottaa kantaa.

‘Käyttöliittymäratkaisut’ on tutkivan osuuden neljäs ja viimeinen luku. Se perustuu edellisessä luvussa nimettyihin vaatimuksiin ja vastaa niihin kartoittamalla nykyisiä käyttöliittymäratkaisuja. Ratkaisujen toteutusta ja niiden soveltuvuutta erilaisiin tilanteisiin arvioidaan, jotta saataisiin tietoa toisaalta toimivista ja toisaalta myös toimimattomista ratkaisuista. Luku vastaa kysymykseen: millaisin ratkaisuin epäsuoraan hallintaan perustuva käyttöliittymä voidaan rakentaa?

Seuraava luku ‘Esimerkkisovellus’ aloittaa tutkimuksen soveltavan osuuden. Siinä kuvataan lyhyesti eräs agenttiohjelma, jota käytetään esimerkkitapauksena tutkivan osuuden tulosten havainnollistamisessa.

Varsinainen soveltaminen tapahtuu viidennessä luvussa ‘Ehdotettu käyttöliittymä’. Siinä esitellään tapausperustaisesti muodostetut käyttöliittymän suunnitteluperiaatteet, jotka soveltuvat käytettäväksi epäsuorasti hallittujen toimintojen yhteydessä. Luvussa esitellään myös esimerkki periaatteiden soveltamisesta konkreettisten käyttöliittymäkuvien avulla. Luvun lopuksi pohditaan suunnittelukonseptin yleistettävyyttä.

2. EPÄSUORA HALLINTA

2.1 Vuorovaikutustapa

Kun tietokoneohjelman käyttöliittymän yhteydessä puhutaan *vuorovaikutustavasta*, tarkoitetaan sillä niitä käytäntöjä, joiden avulla käyttäjä ja ohjelma välittävät tietoja eli kommunikoivat toisilleen. Vuorovaikutustapa määrittelee siis sen, miten käyttäjä voi viestittää tavoitteitaan ohjelmalle. Toisaalta vuorovaikutustapa vaikuttaa myös siihen, millaisena ja miten ohjelma tuloksensa esittää.

Luokiteltaessa tietokoneohjelmien käyttöliittymiä vuorovaikutustavan mukaan, voidaan puhua esimerkiksi suoravaikutteisista, komentopohjaisista ja luonnollista kieltä käyttävistä käyttöliittymistä, valikoihin tai lomakkeiden täyttöön perustuvista [Shneiderman 98, s. 71]. Jaottelun perusteella voidaan ymmärtää vuorovaikutustavan tarkoittavan joukkoa käyttöliittymässä näkyviä objekteja ja niiden käyttämiseen liittyviä tekniikoita [Hix and Hartson 93, s. 57]. Vuorovaikutustapa määrittelee siis millaisia objekteja ohjelman käyttöliittymässä käsitellään ja miten niiden käsittely tapahtuu.

Seuraavassa tehdään käsiteanalyysiä, joka selventää käsitteiden ‘vuorovaikutustapa’ ja ‘esitystekniikka’ eroja. Tällä pyritään käsitteen vuorovaikutustapa selkeämpään ymmärtämiseen. Tämän jälkeen pohditaan vuorovaikutustavan suhdetta metaforaan. Tällä kaikella luodaan käsitteellinen pohja tälle työlle, jossa epäsuora hallinta käsitetään vuorovaikutustavaksi.

2.1.1 Vuorovaikutustapa ja esitystekniikka

On olemassa ilmeinen vaara sekoittaa vuorovaikutustapa ja esitystekniikka toisiinsa, sillä ne ovat käsitteinä läheisessä suhteessa keskenään. Siksi niiden erot on syytä selvittää heti aluksi.

Esitystekniikka tarkoittaa tietokoneohjelman käyttöliittymän toteuttamisessa käytettyä tekniikkaa, jolla käyttöliittymä tehdään käyttäjälle havaittavaksi. Esitystekniikkaan perustuen puhutaan graafista ja merkkipohjaisista käyttöliittymistä. Merkkiperustainen käyttöliittymä koostetaan joukosta merkkejä ja graafinen käyttöliittymä piirretään käyttäen graafisia primitiivejä.

Graafinen esitystapa on tekniikan luonteesta johtuen joustavampi ja siten useampiin tarkoituksiin soveltuva. Hieman yleistäen voisi sanoa, että se mikä pystytään tekemään merkkipohjaisesti, pystytään tekemään myös graafisella esitystekniikalla. Toiseen suuntaan suhde ei ole samanlainen.

Käytetty esitystekniikka on tyypillisesti riippuvainen käytetyn laitteen teknisistä ominaisuuksista. Tähän mennessä merkkipohjainen laite on ollut jossain määrin helpompi ja siten myös halvempi rakentaa, joten sen käyttö on usein perusteltu näistä ja historiallisista lähtökohdista.

Käsitteiden esitystekniikka ja vuorovaikutustapa sekaannuksen vaara on erityisen suuri tapauksissa, joissa vuorovaikutustapa voidaan toteuttaa vain tietyillä esitystekniikoilla. Esimerkiksi suoravaikutteiset käyttöliittymät on tyypillisesti toteutettu graafisella esitystavalla, mikä onkin ymmärrettävää, sillä graafinen esitystekniikka soveltuu tarkoitukseen erinomaisesti. Ainakin periaatteessa suoravaikutteinen käyttöliittymä voitaisiin kuitenkin toteuttaa myös merkkipohjaisesti.

Erottelemalla vuorovaikutustapa ja esitystekniikka on helpompi ymmärtää myös epätavallisia mahdollisuuksia tehdä käyttöliittymiä. Koska vuorovaikutustapa ja esitystekniikka ovat eräässä mielessä riippumattomia toisistaan, voidaan niitä periaatteessa yhdistellä mielivaltaisesti. Käytännössä niiden onnistuneet sovellukset riippuvat kuitenkin monimutkaisesti toisistaan.

2.1.2 Vuorovaikutustapa ja metafora

Käyttöliittymämetafora (myöhemmin metafora) on käyttöliittymän muodostamisessa käytetty idea tai ajatus, joka on muodoltaan toiseen ideaan viittaava [Webster 81]. Tyypillisesti käyttöliittymämetafora viittaa tosimaailman asioihin tai toimintoihin, joiden ajatellaan helpottavan tietokonejärjestelmän käyttöä [Nielsen 93, s. 127]. Esimerkiksi käyttöliittymissä usein käytetty painike on metaforisessa suhteessa tosimaailman esikuvaansa, fyysiseen nappiin. Se muistuttaa esikuvaansa visuaalisesti, mutta jo sen hiiriavusteinen käyttötapa poikkeaa esikuvasta melkoisesti.

Usein ajatellaan, että mitä lähempi suhde tietokoneohjelman ja todellisuuden välillä on, sitä helpommaksi ohjelman oppiminen muodostuu. Aina kovin läheinen suhde ei kuitenkaan ole mahdollinen, sillä ohjelman toiminnalle ei välttämättä ole suoraa vastinetta todellisuudessa. Metaforan valinta ja sen soveltaminen ovatkin hyvin keskeisessä asemassa onnistuneen vuorovaikutustavan luomisessa. Jos todellisuuden ilmiön ja ohjelman käyttöliittymän suhde on liian kaukainen tai läheinen, metafora ei toimi toivotulla tavalla, vaan johtaa ajattelua harhaan tai rajoittaa sitä liiaksi [Nielsen 93, s. 128]. Käyttäjän voi esimerkiksi olla vaikea ymmärtää ohjelman toimintoja, jotka eivät tunnu sopivan metaforaan.

Nykyisin käytössä olevien vuorovaikutustapojen metaforien tai mallien voidaan ajatella olevan seuraavat:

VUOROVAIKUTUSTAPA	METAFORA
Suoravaikutteinen	Fyysisten kappaleiden käsittely
Valikkovalinta	Vaihtoehtolistat
Lomakkeen täyttö	Fyysinen paperilomake
Komentoperustainen	Paperille kirjoitus / konekirjoitus
Luonnollinen kieli	Ihmisten välinen kommunikaatio

Metafora on käyttöliittymän keskeinen tekijä sekä suunnittelijalle että käyttäjälle. Metaforan avulla suunnittelijan on helpompi kehittää ohjelman käyttötapaa, kun hän voi ottaa työnsä lähtökoh-

daksi kokemuksesta tuttuja asioita. Käyttäjän on puolestaan helpompi ymmärtää näin suunnitellun ohjelman toimintaa, sillä metaforan avulla uusi ympäristö (tietokoneohjelma) asetetaan (esim. arkimaailmasta) tuttuun kontekstiin. Toiminta voi silloin perustua vanhaan tietouteen arkimaailmasta, jolloin uuden tiedon omaksumisen tarve vähenee.

Vuorovaikutustavan metaforan ymmärtämistä voidaan auttaa valitsemalla sopiva esitystekniikka. Esimerkiksi suoravaikutteisessa käyttöliittymässä on eduksi, jos käyttöliittymän havaittavat objektit saadaan muistuttamaan fyysisiä kappaleita, koska käytetty metafora perustuu niihin. Tämä pätee niin objektien ulkonäköön, niistä lähteviin ääniin kuin erityisesti niiden käsittelyyn ja objektien välisiin suhteisiin ja vuorovaikutukseen. Tätä taustaa vasten on ymmärrettävämpää myös se, miksi ja miten vuorovaikutustavat ja esitystekniikat ovat yhteydessä toisiinsa: joskus käytetty metafora asettaa esitystavalle erityisiä vaatimuksia, mikä rajoittaa mahdollisten esitystekniikoiden joukkoa.

Vuorovaikutustapaan ja tiettyyn yksittäiseen ohjelmaan liittyvien metaforien erottaminen ja niiden suhteen ymmärtäminen on tärkeää. Yksittäisen ohjelman käyttämä metafora on (tai sen tulisi olla) alisteinen vuorovaikutustapaan liittyvälle metaforalle, jotta ne tukisivat toisiaan. Käytetty vuorovaikutustapa (joka voi olla järjestelmä- tai sovelluskohtainen) antaa rajat, joiden puitteissa ohjelman metafora on valittava. Esimerkiksi suoravaikutteiseen vuorovaikutustapaan liittyvä metafora rohkaisee etsimään myös ohjelman metaforia fyysisestä maailmasta. Tyypillisesti sovellukset tarkentavat vuorovaikutustavan metaforaa: suoravaikutteinen vuorovaikutustapa puhuu vain fyysisistä objekteista, sovellus voi puhua dokumenteista ja kansioista.

2.2 Epäsuoran hallinnan määrittely

Epäsuora hallinta (*indirect management*) tarkoittaa tämän työn yhteydessä karkeasti ottaen tietokoneohjelmien käyttöliittymissä käytettävää erästä vuorovaikutustapaa. Epäsuoralla hallinnalla pyritään vähentämään käyttäjän työtaakkaa siirtämällä yhä suurempi osa tehtävistä tietokoneen automaattisesti suoritettavaksi. Nykyisissä kaupallisissa sovelluksissa epäsuora hallinta on vielä toistaiseksi melko tuntematon vuorovaikutustapa, sillä sen kehitys- ja tutkimustyö ovat vielä kesken.

Tässä alakohdassa kartoitetaan käsitteen historiaa, tarkastelemalla mistä se on peräisin ja miten se on ajan saatossa jalostunut. Käsitettä tarkastellaan myös toisista näkökulmista ajatellen sitä kielenä ja kulttuurina. Lisäksi pohditaan, millaisia ongelmia ja hyötyjä tällaisen vuorovaikutustavan käytöstä tietokoneohjelmien käyttöliittymissä olisi saatavissa.

2.2.1 Epäsuoran hallinnan määritelmä ja historia

Kunnian epäsuoran hallinnan idean keksimisestä Allan Kay [Kay 90] myöntää *John McCarthy*'lle ja *Oliver G. Selfridge*lle. He työskentelivät *MIT*:ssä (Massachussets Institute of Technology) ajatusta kehitellen jo 50-luvun puolivälin tienoissa. Tuolloin oli kyse *Advice*

*Taker*istä, tietokonejärjestelmästä, joka pystyisi itsenäisesti suorittamaan tehtäviä ja pyytämään tarvitessa apua käyttäjältä.

Ajatus epäsuorasta hallinnasta ei ole siis uusi, mutta nimi on tuoreempaa perua. Käsitteen epäsuora hallinta (*indirect management*) ensimmäinen maininta löytyy Alan Kayn 80- ja 90-lukujen vaihteessa julkaisemasta artikkelista 'User Interface: A Personal View' [Kay 90, s. 204]. Hänellä epäsuora hallinta tarkoittaa *älykkäiden taustaprosesseina toimivien agenttien ohjaamista*, jotka työskentelevät käyttäjän päämäärien tyydyttämiseksi.

Pattie Maes jatkaa epäsuoran hallinnan projektia luonnehtiessaan epäsuoraa hallitsemista *yhteistyöprosessiksi, jossa käyttäjä ja tietokoneohjelma ovat tasavertaisia toimijoita*. Tasavertaisina toimijana myös kone voi aloittaa kommunikaation, tarkkailla tapahtumia ja suorittaa tehtäviä [Maes 94, s. 31]. Lisäksi Maes mainitsee vuorovaikutustavan metaforaksi henkilökohtaisen apulaisen, joka on yhteistyössä käyttäjän kanssa samassa toimintaympäristössä.

Jos asetetaan epäsuora hallinta suhteeseen edellä vuorovaikutustavoista esitettyjen ajatusten kanssa, voidaan sen sanoa olevan *tietokoneohjelmien käyttöliittymien vuorovaikutustapa, joka perustuu metaforaan ihmisten välisestä vuorovaikutuksesta*. Tarkemmin sanottuna epäsuora hallinta perustuu ihmisavustajien tai -opastajien suhteeseen avustettavan tai opastettavan kanssa, Pattie Maesin ajatuksia mukaillen. Vuorovaikutustapana epäsuora hallinta ei ole sidottu mihinkään erityiseen esitystekniikkaan, vaan sitä voidaan käyttää yhtä hyvin niin merkkipohjaisessa kuin graafisessakin ohjelmassa.

Vuorovaikutustavan metaforaksi on valittu ihmisten välinen kommunikaatio sen tuttuuden vuoksi. Kommunikointitapa opitaan jo lapsena ja sitä käytetään läpi elämän. Jos samaa kommunikaatiotapaa voisi käyttää kommunikointiin myös tietokoneen kanssa, jäisi useampien kommunikointitapojen opettelutarve pois, joka puolestaan merkitsisi ohjelmistojen käytön opettelun nopeutumista. Vuorovaikutustavan muistaminen ei tuottaisi ongelmia edes satunnaiselle käyttäjälle.

Kun epäsuoraa hallintaa ajatellaan vuorovaikutustapana, se voidaan rinnastaa muihin, kuten suoravaikutteiseen vuorovaikutustapaan. Käsitteet pyrkivät siis ratkaisemaan ainakin osittain samoja ongelmia. Ainoastaan toteutustapa on erilainen: suoravaikutteisessa vuorovaikutustavassa käyttäjä manipuloi ruudulla näkyviä fyysisen kaltaisia objekteja saadakseen halutun efektin E, epäsuorassa hallinnassa käyttäjä tekee tässä yhteydessä määrittelemättömiä toimintoja (ehkä manipuloi objekteja kuten edellä), mikä tulkitaan pyynnöksi saada aikaan efekti E, jonka ohjelma sitten suorittaa. Valtavasta erosta huolimatta kysymys on kummassakin tapauksessa saman efektin E saavuttamisesta. Vuorovaikutustavat ovat siis periaatteessa jopa vaihdettavia.

Ajatus epäsuorasta hallinnasta on ajan saatossa hieman jalostunut ja uudelleen muotoutunut, mutta idean tavoite ei ole muuttunut miksikään. Se on yksinkertaisuudessaan sama kuin kaikella muullakin tietokoneohjelmien käyttöliittymätutkimuksella ja -kehitystyöllä: mahdollistaa helpompikäyttöisten tietokoneohjelmien valmistaminen.

2.2.2 Epäsuora hallinta kielenä

Epäsuoraa hallintaa vuorovaikutustapana voidaan luonnehtia myös uudenlaisena kielenä, jota käytetään ihmisen ja tietokoneen välisessä kommunikaatiossa. Tässä mielessä se määrittelee kommunikaation muodon kiinnittäen kommunikaatiossa käytetyt primitiivit sekä niiden yhdistelemisessä käytettävän kieliopin (syntaksin) ja siihen liittyvän merkityksen (semantiikan). Kielen primitiivit kertovat millaisista osasista kieli koostuu, kun syntaksi puolestaan määrittelee millaisia lauseita kielellä voidaan muodostaa. Semantiikka kertoo, mitä syntaksin mukaan muodostetut kielen lauseet tarkoittavat.

Esimerkiksi suoravaikutteisissa käyttöliittymissä käytettyjä kielen primitiivejä ovat hiiren osoitus, hiiren liike, näppäimen painallus jne. Primitiivit ovat vakiintuneet varsin pysyväksi joukoksi, jonka kaikki sovellukset jakavat. Epäsuorassa hallinnassa käytetyt primitiivit eivät ole vielä vakiintuneet näin selväksi joukoksi. Primitiiveinä on käytetty ainakin havaintoja käytetyistä komennoista ja niiden parametreistä, työskentelyaikaa ja käyttäjän avaamien dokumenttien sisältöä.

Primitiivien ohella myös kielioppi on sekin vakiintumaton. Toisissa sovelluksissa yksinomaan dokumentin sisältö voi kelvata syötteeksi sinällään, toisen määritelmän kieliopin mukaan myös dokumentin tarkasteluun käytetty aika vaikuttaa syötteen merkitykseen. Epäsuoran hallinnan voisi siis ajatella olevan vielä varsin vakiintumaton kieli.

Jos epäsuoraa hallintaa verrataan kielenä muihin ihmisen ja tietokoneen vuorovaikutuksessa käytettäviin kieliin, se poikkeaa niistä muistuttaen enemmän ihmisten arjessa käyttämää kieltä. Tällaiselle kielelle on ominaista, että osa kommunikaatiosta perustuu yhteiseen tietämykseen. Esimerkiksi samassa maassa asuvat ihmiset voivat puhua ongelmitta pääministeristä identifioimatta puheen kohteena olevaa ihmistä sen tarkemmin, jolloin kommunikaation toimivuus perustuu jaettuun tietämykseen. Ihmisten välisessä kommunikaatiossa viestinnän keinoihin kuuluu kielen ohella myös sanaton viestintä, jossa osa tiedosta välitetään ilmein, elein ja äänenpainoin [Wiio 94, s.104–109].

Epäsuora hallinta tietokoneohjelman käyttöliittymässä perustuu samoihin ajatuksiin. Käyttöliittymän toteutukseen on tällöin lisätty yleistä tietämystä esimerkiksi ohjelman sovellusalasta, jolloin kyseisestä aiheesta “puhuttaessa” ohjelma voi ymmärtää sille muuten käsittämättömät viittaukset. Ohjelma voi tuntea myös käyttötilannetta, jolloin keskustelussa juuri parhaillaan käsiteltäviin asioihin viittaaminen helpottuu. *Epäsuora hallinta on siis tietokoneohjelmien hallintaan käytetty kieli, joka kurottaa kohti inhimillistä kieltä yrittäen mahdollistaa sanattoman viestinnän ja epämääräiset viittaukset asioihin*.

2.2.3 Epäsuora hallinta kulttuurina

Uuden vuorovaikutustavan kehittämisessä on kyse myös uuden kulttuurin luomisesta, kuten Kristina Höök on todennut [Höök 99]. Ihmiset nojaavat kaikessa toiminnassa kulttuurisiin käytäntöihin ja pystyvät toimimaan tehokkaasti, kun kulttuuri on tuttu. Asialla on myös kääntöpuoli: jos toimintaympäristön kulttuuri on vieras, siinä toimiminen voi olla hankalaa. Tästä voidaan johtaa ajatus, jonka mukaan uusien järjestelmien tulisi noudattaa tuttuja kulttuureja.

Tähän viittaavat myös muutamat yleisesti hyväksytyt hyvän käytettävyyden tekijät, kuten yhdenmukaisuus ja standardeihin tukeutuminen.

Nykyisiin kulttuureihin pidättäytyminen voi olla myös rajoitus. Esimerkiksi Pattie Maes on kirjannut suoravaikutteisen vuorovaikutustavan rajoitteiksi sen vaatiman käyttäjän ainaisen huomion [Maes 94]. Joissain tapauksissa ongelma voidaan kiertää saman kulttuurin toisilla mekanismeilla, esimerkiksi kehittämällä suoravaikutteiseen käyttöliittymään kehittyneempiä visualisointitapoja, mutta joskus optimaalisempi vastaus voi edellyttää koko kulttuurin tai sen osan muutosta.

Tästä näkökulmasta *epäsuora hallinta on uusi kulttuuri* käyttöliittymäkulttuurien joukossa. Käyttäjälle se merkitsee siten uuden kulttuurin opettelua. Jos epäsuoran hallinnan tutkimista ajatellaan tästä näkökulmasta, on sen tavoitteena luoda uusia käytäntöjä käyttöliittymien rakentamiseksi. Luotujen käytäntöjen tulee olla riittävän monipuolisia ja joustavia, jotta niitä voidaan soveltaa tulevaisuudessa mahdollisimman laajasti. Tämä on edellytys sille, että käytännöt voivat muodostaa toimivan kulttuurin, joka on laajasti käytössä.

Ajatukseen uuden kulttuurin luomisesta ei saa tuudittautua liiaksi. Kaikkia käytettävyysongelmia ei pidä, eikä saa, selittää kulttuurin uutuudella. Muussa tapauksessa kaikki ongelmalliset ratkaisut voitaisiin selittää tällä perusteella. Suunnittelijan on tasapainoiltava hyvin kapealla alueella varoen liian vieraita ratkaisuja ja toisaalta tavoitellen uuden kulttuurin tuomia mahdollisuuksia.

2.3 Epäsuoran hallinnan käyttö

Pelkän määritelmän perusteella on vaikea saada kuvaa mistä epäsuorassa hallinnassa pohjimmiltaan on kysymys. Ongelmaa voi lievittä esittämällä sen käytöstä esimerkkejä ja pohtimalla sen soveltuvuutta. Myös epäsuoran hallinnan käytön etujen ja haittojen kartoittaminen palvelee samaa päämäärää.

Seuraavassa tarkastellaan epäsuoraa hallintaa näistä lähtökohdista. Tarkastelu kokoaa käsitteen määrittelyä yhteen ja antaa konkreettisia esimerkkejä siitä, mitä epäsuora hallinta käytännössä merkitsee.

2.3.1 Epäsuoran hallinnan edut ja haitat

Epäsuoran hallinnan eduista ja haitoista on keskusteltu viimeaikoina vilkkaasti tieteellisillä foorumeilla. Erityisesti Pattie Maes ja Ben Shneiderman ovat esittäneet mielipiteitään asiasta eri artikkeleissaan ja yhteisessä keskustelussa, joka kuultiin vuoden 1997 IUI (Intelligent User Interfaces) konferenssissa Orlandossa, Yhdysvalloissa [Maes and Shneiderman 97]. Keskustelussa tulee esiin yleisemminkin havaittava vastakkainasettelu: suoravaikutteinen käyttöliittymä, jossa kontrolli on käyttäjällä vastaan epäsuoraan hallintaan perustuva, ei suoraan kontrolloitu, käyttöliittymä. Pattie Maes on epäsuoran hallinnan puolesta Shneidemanin puolustaessa käyttäjän tiukasti kontrolloitavia käyttöliittymiä.

EPÄSUORAN HALLINNAN EDUT & HAITAT

- kognitiiviset tehtävät vähenevät
- työ tehostuu
- opettelu nopeaa
- opettaa ohjelman käytössä
- ei suoraa kontrollia
- ohjelman ymmärtäminen vaikeutuu
- toimintojen virhealttius

Tässä alakohdassa esitettävät edut ja haitat perustuvat osittain näihin keskusteluihin ja osittain epäsuoran hallinnan määritelmän pohdintaan. Oheinen lista kokoaa yhteen edut ja haitat, joista seuraavassa tarkemmin.

Käyttöliittymä suuren informaatiomäärän hallintaan on ollut tyypillinen esimerkkisovellusalue, josta keskusteluissa on esitetty mielipiteitä. Epäsuoraa hallintaa tällaisessa esimerkissä puoltaa käyttäjän *tehtävien vähentäminen ja työskentelyn tehostuminen*. Tämä *vähentää käyttäjän kognitiivista kuormaa* ohjelmaa käytettäessä. Käyttäjän ei tarvitse osata hakea häntä kiinnostavaa tietoa tai edes tietää sellaisen mahdollisesta olemassaolosta. Epäsuoraan hallintaan perustuvassa käyttöliittymässä käyttäjä tekee vain ensisijaisiaan tehtäviä ja ohjelma avustaa häntä etsimällä aiheeseen liittyvää lisätietoa. Näin käyttäjän on mahdollista keskittyä hänelle olennaiseen tehtävään ja jättää lisätiedonhaku ohjelman huoleksi. Työskentely tehostuu, kun tehtäviä voidaan suorittaa yhtäaikaa.

Tässä prosessissa *käyttäjä ei kontrolloi* ohjelman kaikkia toimintoja, eikä hän siten kykene määräämään millaista tietoa ohjelma hakee, milloin ja mistä. Tällaisiin ongelmiin suoravaikutteiset käyttöliittymät tarjoavat selviä ratkaisuja. Tiedonhakuohjelmassa käyttöliittymäideana voi olla tiedon mielekäs visualisointi vaikkapa tähtikarttana [Shneiderman 94] ja yksinkertaisten välineiden tarjoaminen tietoalkion löytämiseksi. Käyttäjä tietäisi tarkalleen milloin, mistä ja miten hän tietoa saa.

Esimerkissä mainituissa hakuprosessien kuvauksissa näkyy nähdäkseni oleellinen ero: jälkimmäisessä esimerkissä käyttäjä *tietää* mitä tietoa ja milloin hän tarvitsee. Tällaisessa tilanteessa tuntuu selvältä, että on paras vaihtoehto antaa käyttäjälle kontrolli ja pätevät työkalut tiedon hankintaan. Ensimmäisessä esimerkkitilanteessa käyttäjällä ei puolestaan ollut selkeää käsitystä tiedon tarpeesta, kun ehdotuksia asiaan liittyvästä tiedosta hänelle jo tarjottiin. Selvästikin käyttäjän työtaakka jää näin pienemmäksi ja ohjelman käyttö nopeutuu, kun toimintoja voidaan suorittaa rinnakkain. Käytön nopeutumisen ehtona on kuitenkin se, että käyttäjä todella saa haluamansa tiedon ja uskoo, että saatu tieto on kaikki, mitä asiasta tulee tietää.

Epäsuoran hallinnan yhteydessä käyttäjä *ei välttämättä ymmärrä ohjelman toimintaa*. Luottamus, joka seuraa toiminnan ymmärtämisestä, onkin erääksi ratkaiseva piirre epäsuoran hallinnan hyödyllisyyden kannalta. Epäsuoraan hallintaan perustuvasta ohjelmasta ei ole hyötyä, jos käyttäjä ei luota sen tuottamiin tuloksiin. Käyttäjän tarvitsemien tulosten aikaansaaminen on varmistettava.

Epäsuorasti hallitussa ohjelmassa käyttäjällä on mahdollisuus *saada tuloksia, joiden mahdollisuudesta käyttäjä ei tiennyt*. Esimerkiksi haku, jonka kriteerit ovat käyttäjälle vieraat, voi tuottaa tietoa, josta hän ei ollut tietoinen ja joka on hänelle kuitenkin arvokas. Tällaista tietoa käyttäjä ei itse tulisi etsineeksi, koska ei tiedä sen olemassaolosta, mutta käyttöympäristön ohjaamana tieto on voitu käyttäjälle mielekkäästi välittää.

Tähän läheisesti liittyen, epäsuoraa hallintaa käyttävä käyttöliittymä voi toimia käyttäjälle *oppimispolkuna ohjelman käyttöön*. Jos käyttäjä voi epäsuorasti hallita osaa ohjelman toiminnoista, hänen on mahdollista myös ymmärtää millaisia toimintoja järjestelmällä on mahdollista suorittaa ja miten niitä käytetään, kun ohjelma hä-

nelle uudenlaisia tuloksia tuottaa. Tämä edellyttää, että käyttäjä voi jotenkin seurata epäsuorasti hallitun toiminnon suorittamista.

Ei pidä unohtaa myöskään käytön *opetteluun tarvittavan ajan vähenemistä* tai peräti opiskelutarpeen poistumista, kun käytetään epäsuorasti hallittavaa ohjelmaa. Vaikka suoravaikutteinen käyttöliittymä perustuu kaikille tuttuun metaforaan fyysisten kappaleiden manipuloinnista, ei ole selvää, että sen soveltaminen onnistuisi kaikissa sovelluksissa. Käyttäjän on ymmärrettävä eri toimintojen väliset yhteydet, jotta hän kykenee käyttämään ohjelmaa. Toimintojen välisten suhteiden ymmärtämistä suoravaikutteisuus ei välttämättä edesauta.

Koska epäsuorasti hallittu järjestelmä ei voi aukottomasti tietää käyttäjän tarpeita ja haluja, näin hallitut *järjestelmät tekevät virheitä*. Tämä on ymmärrettävää, kun ajatellaan epäsuoran hallinnan metaforaa, inhimillistä kommunikaatiota. Myös siinä tapahtuu paljon virheitä, vaikka ihmisten välisessä kommunikaatiossa on tyypillisesti tarjolla runsaasti merkityksiä selvittävää redundanttia informaatiota. Virheistä aiheutuvia ongelmia voidaan pienentää ottamalla niiden mahdollisuus huomioon jo suunnittelussa. On muistettava, ettei epäsuoran hallinnan idean mukaan käyttäjältä vaadita erityisiä toimenpiteitä tulosten saamiseksi. Juuri tämä seikka voi vähentää virheiden merkitystä, kunhan virheellisten tulosten unohtaminen tai kumoaminen on riittävän helppoa.

2.3.2 Epäsuoran hallinnan soveltuvuus

Epäsuoran hallinnan soveltuvuuden pohdita on hyvä aloittaa palauttamalla mieleen ajatus, miksi sen ylipäätään ajellaan olevan käyttökelpoinen vuorovaikutustapa. Epäsuora hallinta perustuu metaforaan ihmisten välisestä kommunikaatiosta, joka on kaikille tuttua. Toisaalta suoravaikutteisuuteen perustuva vuorovaikutustapa perustuu metaforaan fyysisten kappaleiden manipuloinnista, joka on sekin kaikille ihmisille tuttua. Molempien vuorovaikutustapojen perusta on tutun ilmiön siirtäminen toiseen käyttöön, joten kysymys onkin sovellusmahdollisuuksista.

Soveltuva vuorovaikutustapa riippuu tehtävästä ja sovellusalueesta, eikä liene olemassa universaalisti parasta vuorovaikutustapaa. Erityisesti tämä on ymmärrettävää, jos ajatellaan epäsuoran vaikuttamisen perustuvan avustajan - avustettavan välisen suhteen metaforiseen käyttöön. Kaikki tehtävät eivät ole soveltuvia avustajalle välitettäviksi. On siis punnittava todellisia käyttö- tai sovellustilanteita, jotta saadaan näkemys, millaisiin käyttöliittymiin eri vuorovaikutustavat soveltuvat parhaiten.

Kirjallisuudessa käydyn keskustelun perusteella [Maes and Shneiderman 97] ei epäsuoran hallinnan soveltuvuutta nimenomaan avustaviin toimiin voi korostaa liiaksi. Epäsuoran hallinnan soveltumattomuutta yleiseksi käyttöliittymän vuorovaikutustavaksi perustellaan monesti pohtimalla mitä se merkitsisi ydinvoimalan hallinta- tai lennonvarmistusjärjestelmissä. Nämä esimerkit edustavat hyvin kriittisiä järjestelmiä, joiden yhteyteen on vaikea kuvitella avustavaa tehtävää, joka ei olisi virhekriittinen. Epäsuora hallinta soveltuu vain avustaviin tehtäviin, joiden yhteydessä tehdyt virheet ovat merkityksettömiä ja / tai helppoja oikaista.

- agenttiohjelmat
- esimerkkiperustainen ohjelmointi
- adaptiiviset käyttöliittymät
- mukautuvat opasteet

2.3.3 Epäsuoraa hallintaa käyttävät sovellukset

Epäsuoraa hallintaa käyttävät sovellukset voidaan jakaa käyttötarkoituksen perusteella neljään ryhmään, jotka oheinen lista tiivistää. Seuraavassa esitetään lyhyt luonnehdinta kaikista näistä sovellusalueista.

Agenttiohjelmiin epäsuora hallinta kuuluu lähtemättömästi. Niissä epäsuoralla hallinnalla mahdollistetaan toimintojen ympäristöstä riippuva automatisointi. Agenttiohjelmat mm. lajittelevat sähköpostia, indeksoivat websivuja ja suorittavat tiedonhakuja käyttäjän puolesta. Toimintoja ei tarvitse erikseen ohjata, sillä ne kykenevät ottamaan muuttuvan toimintoympäristön huomioon.

Esimerkein tapahtuvaa ohjelmointia (*PBD, programming by demonstration*) tukevien ohjelmien tavoitteena on mahdollistaa toistuvien rutiininomaisten tehtävien yksinkertainen automatisointi. Ohjelma tulkitsee käyttäjän toimet esimerkiksi halutusta toimintasarjasta ja pyrkii yleistämään toimet uudelleen käytettäviksi.

Kolmas tunnistettava ohjelmaryhmä, jossa epäsuoraa hallintaa on laajemmin käytetty on käyttöliittymien automaattista mukauttamista tukevat ohjelmat. Tällaisissa ohjelmissa ohjelman havaittavaa käyttöliittymää muokataan automaattisesti sopimaan käyttäjän mieltymyksiin, osaamistasoon tai tehtävätilanteisiin paremmin. Esimerkiksi ohjelman valikkorakennetta voidaan muuttaa kesken ohjelman suorituksen siten, että käyttäjän usein käyttämät komennot ovat helpommin saatavilla.

Neljänneksi epäsuoraa hallintaa käytetään tuottamaan käyttäjälle paremmin ymmärrettäviä ja käsillä olevaan tehtävään suoremmin liittyviä opasteita. Opasteita mukauttavat ohjelmat tarkkailevat käyttäjän toimia ymmärtääkseen, millaista tehtävää käyttäjä on suorittamassa, millaisessa ympäristössä käyttäjä toimii tai millaiset ovat käyttäjän edeltävät tiedot. Näiden tietojen perusteella käyttäjälle on mahdollista antaa tehtävään paremmin liittyvää ja samalla myös käyttäjän kannalta helpommin ymmärrettävää apua.

Apu voidaan tarjota joko käyttäjän pyynnöstä tai jopa automaattisesti, jos tehtävän seurannan perusteella voidaan tietää, milloin käyttäjällä on ongelma tai milloin hän toimii tehtävän suhteen virheellisesti. Opastus voi liittyä käytettävän tietokoneohjelman käyttöön tai tehtävään, jota käyttäjä ohjelman avustuksella suorittaa.

Näistä neljästä eri ohjelmaryhmästä on erotettavissa ainakin viisi erilaista tehtävää, joiden toteutuksessa epäsuoraa hallintaa on sovellettu:

1. Toimintojen suorituksen käynnistäminen
2. Toimintojen parametrien säätäminen
3. Suoritettavien toimintojen valinta
4. Tehtäväsarjojen ohjelmointi
5. Käyttöliittymien mukauttaminen

Paitsi mainittuihin neljään ryhmään epäsuorasti hallittavat ohjelmat voidaan jaotella myös toisin. Esimerkiksi voidaan ajatella ohjelmat jaettaviksi avustaviin ja opastaviin. Avustavat ohjelmat muuttavat järjestelmän tilaa, ne siis tekevät jotain käyttäjän puolesta. Opastavat eivät tee mitään käyttäjän puolesta, ainoastaan opettavat tai ohjaavat ohjelman käytössä.

Opastusjärjestelmissä epäsuoraa hallintaa käytetään sekä opastuksen sisällön valintaan ja muokkaamiseen että opastuksen aktiiviseen tarjoamiseen. Opastuksen sisältöä voidaan muuttaa riippuen käyttäjän suorittamista tehtävistä tai hänen tietämyksestään.

Avustavissa ohjelmissa epäsuoraa hallitsemista käytetään avustettavan toiminnon valitsemiseen ja/tai sen suoritusajankohdan määrittelemiseen. Esimerkiksi sähköpostin arkistoinnissa avustava ohjelma voisi olla tällainen. Avustavissa ohjelmissa epäsuoran hallinnan toteuttamista ja sovelluskohteiden valintaa on kuitenkin harkittava erityisen tarkasti, ettei epäsuorasti hallittavaksi määritellä epäsopivia toimintoja. Tällaisia voivat olla mm. toiminnot, joita on vaikea peruuttaa, tai joiden väärään aikaan tapahtuva suoritus voi olla vahingollista.

3. EPÄSUORAN HALLINNAN TOTEUTUS

3.1 Älykkäät käyttöliittymät

Älykkäät käyttöliittymät ovat eksplisiittisiin malleihin perustuvia käyttöliittymiä, jotka ideaalitapauksessa kykenevät muodostamaan annetulle sovellukselle käyttöliittymän automaattisesti. Älykkäät käyttöliittymät pyrkivät epäsuoran hallinnan ideoita mukaillen vähentämään käyttäjän tehtäviä automatisoimalla suuren osan nykyisin käyttäjälle osoitetuista tehtävistä.

Toisaalta älykkäiden käyttöliittymien voidaan ajatella hyödyntävän automatisoinnin toteutuksessa epäsuoraa hallintaa vuorovaikutustapana. Erityisesti tästä näkökulmasta niiden tarkastelu on hyödyllistä, kun pyritään ymmärtämään epäsuoran hallinnan toteuttamista.

Seuraavassa katsauksessa älykkäitä käyttöliittymiä tarkastellaan ensin kokonaisuutena, sitten sen osiin tarkemmin paneutuen. Lopussa pohditaan tarkemmin älykkäiden käyttöliittymien ja epäsuoran hallinnan suhdetta.

3.1.1 Älykkyys käyttöliittymän ominaisuutena

Käsitteen ‘älykäs’ liittäminen kuvaamaan uuden sukupolven käyttöliittymiä ei ole ongelmatonta. Älykkyys on inhimillisenkin elämän ominaisuutena varsin monimutkainen ja vaikeasti määriteltävissä. Mitä älykkyys sitten tarkoittaa tietokoneohjelmassa, on mahdollisesti vieläkin epäselvempää. Usein ohjelman älykkyys esiintyykin julkaisun otsikossa tai prototyypin nimessä, mutta varsinainen keskustelu ohittaa älykkyyden. Toisin sanoen, hyvin harvoin näkee nostettavan esille niitä yksittäisiä ominaisuuksia ohjelmasta, jotka tekevät siitä älykkään. Älykkyyden liittäminen ohjelman ominaisuudeksi tuntuukin monesti markkinointikeinolta.

Oli käsitteen määrittely hankalaa tai ei, älykkyys käyttöliittymässä pyrkii vähentämään paitsi käyttäjän myös sovelluskehittäjän työmäärää. Älykäs käyttöliittymä tuntee käyttäjän tavoitteet ja yleisiä periaatteita siitä, miten erilaisten tavoitteiden saavuttamista voidaan tukea. Sovelluskehittäjän ei tarvitse päättää miten eri asiat käyttöliittymässä esitetään vaan ratkaisu tehdään ohjelman suoritusaikana tunnettuja sääntöjä soveltaen annetussa käyttötilanteessa.

Mark Maybury on [Maybury 99] esittänyt hyvin strukturoidun käsityksen siitä, mistä osista älykkäät käyttöliittymät koostuvat (Kuva 1). Tällaisen kuvauksen perusteella pystytään muodostamaan huomattavasti tarkempi kuva siitä, mitä käsitteellä tarkoitetaan, kuin ajattelemalla ainoastaan käsitettä ‘älykäs’.

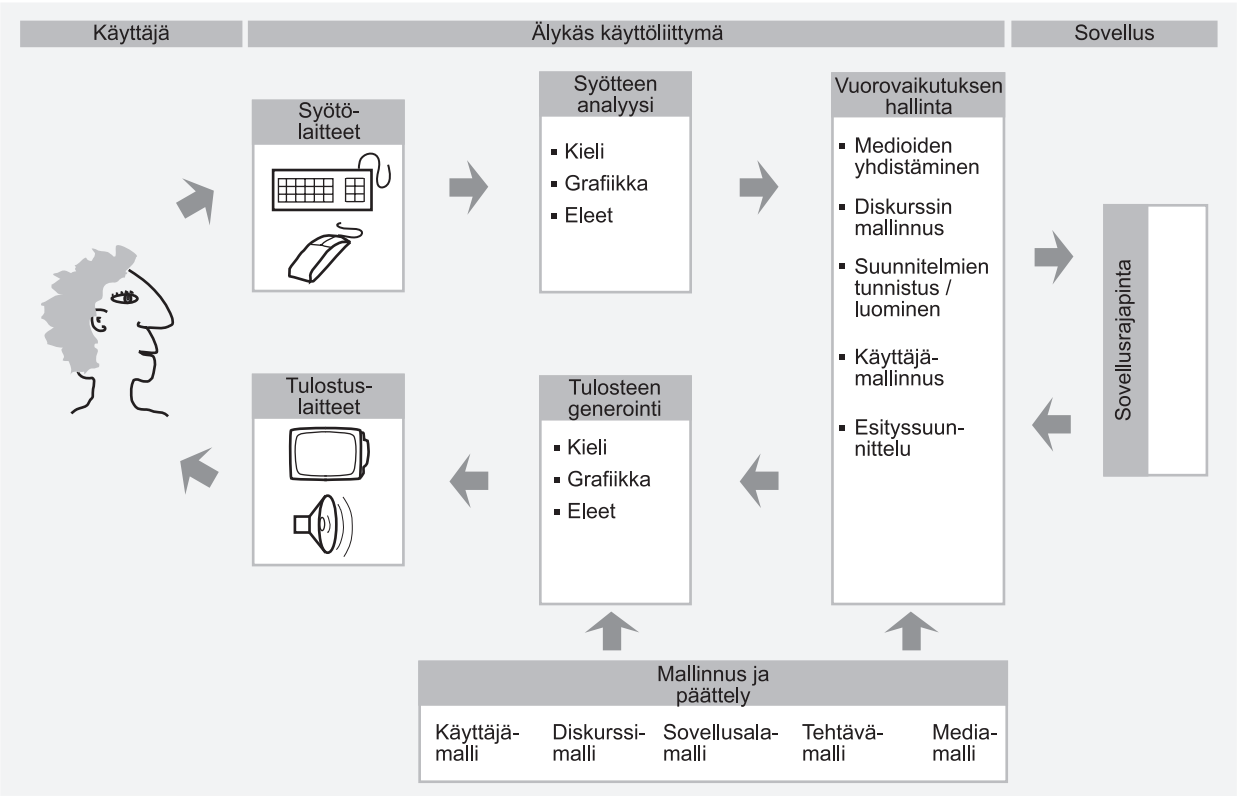
Maybury jakaa älykkään käyttöliittymän kolmeen osaan, jotka erottavat tällaisen käyttöliittymän traditionaalisista käyttöliittymistä:

1. Syötteen analysointi
2. Tulosteen generointi
3. Vuorovaikutuksen hallinta

3.1.2 Syötteen analysointi

Syötteen analysoinnin tavoitteena on nostaa syötteen abstraktiotasoa. Abstraktiotason mataluus syötteessä merkitsee semantiikan puuttumista syötteestä. Mitä korkeammaksi syötteen abstraktiotasoa voidaan nostaa eli mitä enemmän syötteeseen voidaan kiinnittää semantiikkaa, sitä yksinkertaisemmaksi ja tehokkaammaksi sen käyttö muodostuu. Syy on yksinkertainen: jos ohjelmalle tulevassa syötteessä ei ole semantiikka kiinnitetty, on ohjelman tehtävä se itse. Edelleen mitä suurempi osa syötteen semantiikassa on kiinnitetty sovellusohjelman ulkopuolella, sitä yhdenmukaisemmin eri ohjelmat käsittelevät samaa syötettä. Tämä edesauttaa ohjelmien yhdenmukaisuutta.

Nykyisissä käyttöliittymissä ohjelmat joutuvat tyypillisesti toimimaan hyvin matalatasoisten tapahtumien kanssa. Esimerkiksi nykyiset graafiset käyttöliittymät antavat ohjelmalle syötteeksi tietoa



Kuva 1 Älykkäiden käyttöliittymien rakenne

osoitinlaitteen liikkeistä, näppäinten painamisesta jne. Tällaiset tapahtumat ovat ohjelman logiikan kannalta hyvin matalatasoisia, niihin ei sinällään liity selvää semantiikkaa. Nykyisissä graafisissa käyttöliittymissä on mahdollista havaita myös joitain hieman korkeamman tason tapahtumia. Esimerkiksi käyttöliittymässä näkyvän painikkeen painaminen tai valikkovalinnan tekeminen ovat tällaisia.

Syötteen analysointi -osa Mayburyn hahmottelemassa älykkäässä käyttöliittymässä perustuu yhtäältä multimodaalisen syötteen käsittelyyn. Useita samanaikaisia syötteitä analysoimalla ja integroimalla on mahdollista lisätä syötteen ilmaisuvoimaa ja näin liittää syötteeseen mielekästä semantiikkaa. Syötteen analysoija voisi esimerkiksi yhdistää elesyötteen ja puheena annetun komennon siten, että puheessa mahdollisesti esiintyvät (tietokoneohjelman kannalta) epämääräiset viittaukset pystyttäisiin purkamaan. Tämän periaatteen ilmaisuvoimaa ja toimivuutta on demonstroitu klassisessa “Put-That-There” järjestelmässä [Bolt 80].

Paitsi samanaikaisten eri moodissa saatavien syötteiden integrointia, syötteen analysointi -osan tehtävänä on myös yksittäisten syötteiden saattaminen ohjelman vaatimaan muotoon. Erityisesti puheella annetun syötteen analysointi on tällainen monimutkainen tehtävä, jonka hyödyllisyys pitkälti riippuu miten hyvin syötteeseen pystytään liittämään semantiikkaa eli miten luotettavasti puhetta kyetään ymmärtämään. Lisäksi esimerkiksi katseenseuranta voi osoittautua tehokkaaksi kommunikaatiokanavaksi, kunhan senkin abstraktiotasoa saadaan nostettua.

Syötteen analysointi helpottaa käyttäjän työtaakkaa tarjoamalla rikkaan joukon mahdollisuuksia antaa tietokoneohjelmalle syötettä. Käyttäjä voi valita mielentilaansa, työtilanteeseensa, mieltymyksiinsä ja osaamistasoonsa nähden sopivan syötteenantotavan. Myös sovelluskehittäjän työmäärä vähenee, jos käyttöliittymät voidaan rakentaa tällaisen komponentin varaan. Syötteen analysointi on tehty jo valmiiksi, eikä samaa semantiikan kiinnittämistä syötteeseen jouduta tekemään jokaisessa ohjelmassa erikseen.

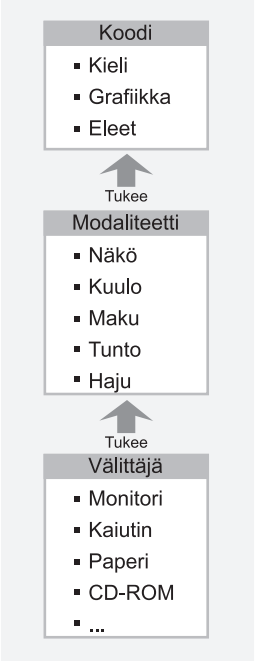
3.1.3 Tulosten generointi

Tulosten generoinnin tavoitteena on vapauttaa käyttäjä ja sovelluskehittäjä tekemästä päätöstä, millaisessa muodossa ohjelman tulos tulisi esittää. Tuloksen esitystavan laatua voidaan tällä menetelmällä potentiaalisesti nostaa, jos ohjelma pystyy ottamaan käyttötilanteen optimaalisesti huomioon. Käyttötilanteen huomioiminen on nykyisillä käyttöliittymien rakennusmenetelmillä mahdotonta. Tiedon omaksuminen on helpompaa, jos se esitetään muodossa, josta sen yhteydet muihin käsillä oleviin asioihin on helposti nähtävissä ja ymmärrettävissä.

Tiedon esitysmuodon valitsemiseksi Maybury erottelee käsitteet: modaliteetti (*mode*), välittäjä (*medium*) ja koodi (*code*) (Kuva 2). Modaliteetti ilmaisee sen, millä aistein ihminen viestin vastaanottaa. Modaliteettejä ovat esimerkiksi haju, näkö ja tunto. Välittäjä on puolestaan fyysinen laite tai välittäjäaine (esimerkiksi paperi, ilma tai CD-ROM), jonka välityksellä viesti kulkee lähettäjältä vastaanottajalle. Koodi on viestinnässä käytetty merkkijärjestelmä (esimerkiksi luonnollinen kieli, kuvallinen kieli, kehon kieli…).

Jo nykyisissä graafisissa käyttöliittymissä on olemassa monipuoliset mahdollisuudet käyttää eri koodeja ja valita niiden välittämiseen eri välittäjiä. Tyypillisesti on käytetty luonnollista kieltä tekstinä ja puheena sekä kuvakieltä vaikkapa kaavioiden ja videoiden muodossa. Automaattisessa tulosten generoinnissa eri koodien ja välittäjien ominaisuudet otetaan huomioon ja käytettäviksi valitaan tiedon esitystarpeen tyydyttämiseen parhaiten soveltuvat.

Esitysmuodon valintaan vaikuttavat käyttötilanteen lisäksi sekä esitettävä tieto että välittäjän ja koodin ominaisuudet. Esitettävä tieto on tällöin kuvattava metatiedolla niin tarkoin, että sille parhaiten sopiva esitystapa voidaan kuvauksen perusteella löytää. Samoin myös eri koodien ja välittäjien ominaisuudet on kuvattava vastaavalla tekniikalla. Etsimällä esitettävän tiedon kuvaukseen mahdollisimman hyvin soveltuva median kuvaus löydetään optimaalinen esitystapa tiedolle.

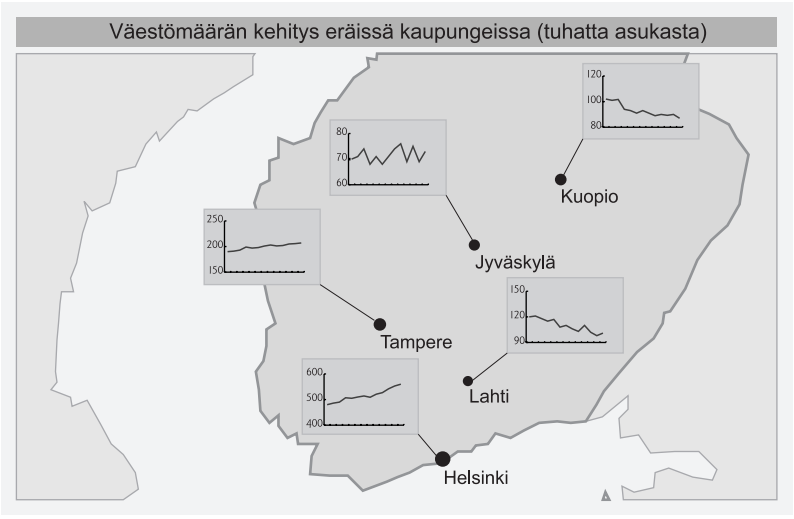


Kuva 2 Koodi, modaliteetti ja välittäjä, eräs näkemys käsitteiden suhteista

Paitsi sopivien esitystapojen valinta, tulosten generoimiseen liittyy myös niiden integrointi. Ohjelman tulos voi koostua suuresta määrästä tietoalkioita, joiden optimaalinen esittäminen vaatii useampien esitystapojen käyttöä. Esimerkiksi maantieteelliset paikat näytetään kartalla ja niihin liittyvä väestömäärän kehitys graafisella kuvaajalla. Samaan esitykseen liittyvät osaesitykset tulee integroida toisiinsa niin, että kokonaisuus muodostuu mielekkääksi ja helposti ymmärrettäväksi. Mainitussa esimerkissä tämä tarkoittaa sitä, että väestömäärän kehitystä kuvaajat sijoitetaan lähelle sitä pistettä kartalla, jonka väestömäärän kehitystä ne kuvaavat. Kuvaaja voidaan sitoa paikkaan tarkemmin esimerkiksi viivalla tai nuolella (Kuva 3).

Esitystapojen integrointi voi tapahtua myös ajallisesti. Ajallisesti integroitu esitys on tarpeellinen esimerkiksi silloin, kun yksi valituista esitystavoista on video. Tällöin esitettävään tietojoukkoon mahdollisesti kuuluva teksti voi loogisesti liittyä vain videon tiettyyn osaa, jolloin ajallinen integrointi on välttämätöntä.

Integrointia vaaditaan usealla eri tasolla. Kuten edellä kävi ilmi, on koordinoitava eri esitysmuotoja ja niiden ajallisia suhteita. Koordinointi voi tapahtua myös pienemmässä mittakaavassa. Jo yhden graafisen esityksen sisällä voidaan tarvita koordinointia valittaessa



Kuva 3 Esimerkki eri koodien yhdistämisestä (tiedot kuvitteellisia)

eri tietoalkioille parhaiten soveltuvia esitystapoja. Esimerkiksi SAGE-työkalu [Roth *et al.* 91] tukee tällaista tietojoukkojen esitystapojen koordinointia. Tämä mahdollistaa verraten sofistikoituneiden visualisointien muodostamisen automaattisesti.

Osa tulosten generoinnista on tulosten sommittelu. Tieto-objektien fyysinen sommittelu sinälläänkin välittää tietoa niiden rakenteesta, merkityksestä ja osien välisistä suhteista. Joissain tapauksissa, kuten edellä mainitussa karttaan ja väestömäärän kehitykseen viittaavassa esimerkissä, sommittelu ei ole suuri ongelma, jos käytetty välittäjä tai koodi sinällään jo määrittelee sommittelua. Aina näin ei kuitenkaan ole ja tiedot on sommiteltava muilla perusteilla. Tutkittuja menetelmiä ovat mm. ruudukkoon, taulukkoon, sääntöihin ja rajoitteisiin perustuvat sommittelutekniikat.

3.1.4 Vuorovaikutuksen hallinta

Älykkäistä käyttöliittymistä tunnistettu kolmas komponentti on vastuullinen ylläpitämään tietoa, joka mahdollistaa kahden edellä esitellyn osan toiminnan. Tämä osa ei suoranaisesti näy käyttäjälle, vaan sen vaikutus tuntuu ainoastaan välillisesti siinä, miten hyvin syötteen analysointi ja tulosteen generointi voidaan toteuttaa. Vuorovaikutuksen hallinta tai mallinnus, joksi sitä voisi myös kutsua, on täten hyvin keskeinen osa älykästä käyttöliittymää.

Ongelmallista on, että sen tutkimus ei ole kovin jäsentynyttä vaan koostuu yksittäisissä prototyypeissä kokeilluista osista ja niistä saaduista kokemuksista. Valitettavasti prototyyppeihin liittyvät vuorovaikutuksen hallintaan liittyvät osat ovat kovin sovellusriippuvaisia, eikä tuloksia ole näin ollen helppo yleistää.

Mayburyn mallissa vuorovaikutuksen hallinta perustuu eksplisiittisiin malleihin kaikista järjestelmän toimintaan liittyvistä objekteista. Näitä ovat: käyttäjämalli (*user model*), sovellusalueen ja tehtävien mallit (*domain/task model*), vuorovaikutusmalli (*interaction model*) sekä kieli- ja mediamallit (*language/media model*).

Käyttäjämalli sisältää tietoa käyttäjästä. Useimmiten käyttäjämalli on dynaaminen eli ajassa muuttuva. Käyttäjämallin kohdalla dynaamisuus on erityisen tärkeää, sillä useiden mallin sisältämien attribuuttien arvot muuttuvat usein. Esimerkiksi käyttäjän kiinnostuksen kohteet on tällainen ominaisuus. Toki myös muuttumattomia attribuutteja mallissa voi olla, kuten käyttäjän syntymäaika tai sukupuoli. Se, mitä käyttäjämalli todella sisältää, on nykyisissä prototyypeissä riippunut vahvasti sovelluksen tarpeista. Käyttäjämalli on tyypillisesti pidetty niin pienenä ja yksinkertaisena kuin sovelluksen kannalta on ollut mielekästä. Näin yleisiä ja uudelleen käytettäviä käyttäjämalleja ei ole muodostunut.

Sovellusalueen malli kuvaa sovellusalueen käsitteet ja niiden väliset suhteet. Sovellusalueen malliin liittyy läheisesti myös tehtävämalli, joka kuvaa sovellusalueen käsitteille tehtäviä operaatioita ja niiden välisiä suhteita. Tehtävämalleja on hyödynnetty prototyyppijärjestelmissä esimerkiksi suoritettavaan tehtävään liittyvän avusteen löytämisessä. Näin on menetelty mm. ACPA-järjestelmässä [Johnson *et al.* 99]. Tehtävämallin avulla järjestelmässä on kyetty antamaan käyttäjälle helpommin saavutettavaa ja tarkemmin tehtävään liittyvää apua, kuin mitä perinteisillä toteutustekniikoilla olisi mahdollista saavuttaa.

Vuorovaikutusmalli ylläpitää tietoa järjestelmän toimijoiden välisestä vuorovaikutuksesta pitäen kirjaa siitä, millaisia objekteja keskustelun polttopisteessä on milläkin hetkellä. Vuorovaikutusmallia tarvitaan, jotta luonnollisessa kielessä käytetyt epämääräiset viittaukset voidaan purkaa. Esimerkiksi pronominien viittaussuhteiden purkaminen on mallin avulla mahdollista. Tällaiseen tarkoitukseen vuorovaikutusmallia on käytetty CUBRICON-järjestelmässä [Neal and Shapiro 91, s. 19]. CUBRICON käyttää vuorovaikutusmallia myös tuloksen käsittelyyn, jolloin myös tietokonejärjestelmä voi käyttää enemmän luonnollisen kaltaista keskustelutapaa viittaamalla objekteihin pronominein. CUBRICON järjestelmä voi viitata objektiin sanalla ‘tämä’ ja korostamalla samalla kyseistä objektia näytöllä.

Käsitteellisesti vuorovaikutusmalliin kuuluu tietoa myös säännöistä, joiden alaisena vuorovaikutus toimii. Esimerkkinä säännöistä voitaisiin mainita vaatimukset jatkuvuudesta ja asiaankuuluvuudesta. Jos keskustelu ei noudata näitä sääntöjä, sen seuranta ja ymmärtäminen vaikeutuvat. Todellisuudessa tällaiset säännöt eivät ole eksplisiittisesti mukana kaikissa rakennetuissa prototyypeissä. Sen sijaan säännöt on implisiittisesti olemassa niissä ohjelmien osissa, jotka päättävät ohjelman toimista vuorovaikutusmallin säilyttämän tilan perusteella.

Kieli- ja mediamallit tuntevat käytettävissä olevien kielien ja medioiden ominaisuuksia. Kuten jo aiemmin tulosteen generoinnin yhteydessä mainittiin, tällaisia malleja käytetään valittaessa annetulle tiedolle parhaiten soveltuvaa esitystapaa.

3.1.5 Epäsuoran hallinnan suhde älykkäisiin käyttöliittymiin

Epäsuora hallinta on suorastaan sisään kirjoitettu älykkäiden käyttöliittymien ideaan: käyttöliittymien on mukauduttava “älykkäästi” käyttäjän tarpeisiin ja automaattisesti tarjottava tietoa käyttäjän tarvitsemassa muodossa. Käyttäjä siis ohjaa mm. käytettävää tiedon esitystapaa epäsuorasti suorittamillaan toiminnolla, ei suoraan valitsemalla mahdollisista vaihtoehdoista. Tätä epäsuora hallinta nimenomaan on.

Mayburyn esittämä käsitys älykkäistä käyttöliittymistä korostaa kuitenkin vain osaa toiminnallisuudesta, jota voidaan hallita epäsuorasti. Mayburyn malli nostaa tässä suhteessa erityisesti esiin vain tuloksen esittämisen ja jossain määrin syötteen antamisen. Hänen mallinsa pyrkii kuvaamaan ilmiötä laajasti pyrkien jäsentämään hyvin laajaa tutkimuskenttää asettamalla tehtyä tutkimusta älykkäiden käyttöliittymien kontekstiin. Tässä mielessä hänen mallinsa on myös onnistunut ja ansiokas.

Näkökulmassa epäsuoran hallinnan vivahteet kuitenkin hämärtyvät, koska sen mukaan epäsuorasti hallittaisiin vain ohjelman tuottaman tuloksen esitysmuotoa. Tämä ei tietenkään ole tahallista, vaan on ainoastaan mallin valitseman näkökulman eräs seuraus.

3.2 Agenttiohjelmat

Agenttiohjelmat ovat itsenäisiä tietokoneohjelmia, jotka kykenevät suorittamaan tehtäviä ottaen ympäristön tilan huomioon. Kuten älykkäissä käyttöliittymissäkin, on ajatuksena käyttäjältä vaadittu-

jen toimintojen vähentäminen, sovelluskohteet ovat vain hieman erilailla painottuneet.

Agenttiohjelmien idea on lähtöisin samasta MIT:ssä esitetystä ajatuksesta mistä epäsuora hallintakin sai alkunsa 50-luvun puolivälissä. Koska ajatukset perustuvat samaan kanta-ajatukseen on selvää, että niiden välillä on hyvin vahva yhteys: tavoitteet ovat samat. Agenttiohjelmien sovelluksia on kuitenkin löydettävissä hyvin erilaisista kohteista, joten niiden perusteella on mielekästä tarkastella epäsuoran hallinnan soveltuvuutta ja toteuttamista.

Agenttiohjelman käsitteellä on vahva yhteys myös älykkään käyttöliittymän ajatukseen. Esimerkiksi Mark Maybury [Maybury 99] puhuu myös agenttiohjelmista puhuessaan yleisemmin älykkäistä käyttöliittymistä. Hän näkee agenttiohjelmat erääksi tavaksi toteuttaa älykäs käyttöliittymä.

Agenttiohjelmien luonnetta selvitetään aluksi esittämällä agenttiohjelmien määritelmiä, joiden avulla saadaan kuvaa, mitä kaikkea agenttiohjelmilla käsitetään. Lopuksi tarkastellaan agenttiohjelmien ja epäsuoran hallinnan suhdetta.

3.2.1 Agenttiohjelman määrittelyjä

Kun ajatus tietokoneagenteista 50-luvulla syntyi, sen määritteleminen vaikutti vielä suhteellisen yksinkertaiselta. Käsite oli uusi, eivätkä eri tahot olleet vielä jättäneet siihen jälkeään. Toisin on nykyisin, kun useat tutkimusryhmät ja vielä monilukuisemmat tutkijat ja sovelluskehittäjät ympäri maailman toteuttavat ja tutkivat tietokoneagentteja. Vaikka idea agentista on jo vuosikymmenten ikäinen, idean toteutukset eivät vieläkään ole niin vakiintuneita, että yhtenäinen käsitys agenteista olisi muodostunut. Siksi selkeän määrittelyn antaminen on liki mahdoton tehtävä.

Agenttikäsitteen alla tehdään hyvin monimuotoisia tietokonesovelluksia yksinkertaisista tiedonhakukoneista käyttäjää tarkkaileviin ja hänen tapojaan oppiviin ihmishahmoisiin avustajiin. Juuri tapa kuvata ohjelmia agenttinimellä tuottaa kattavimman määrittelyn sille mitkä ohjelmat ovat agentteja: *agenttiohjelmia ovat ne, joita agenteiksi kutsutaan*. Vaikka määritelmä perustuu kehäpäätelmään, vaikuttaa se silti olevan käytössä. Joskus kahta ohjelmaa ei juuri muu yhdistä kuin se, että molempia kutsutaan agentiksi. Esimerkiksi Internetin hakupalvelu WebCrawler ja sähköpostia lajitteleva MAXIMS-agentti ovat kovin erilaisia, vaikka eräs lähde [Brenner *et al.* 98] molemmat agenteiksi mainitseekin. Toisaalta myös WebCrawlerin ja tavallisen kirjastotietokannan erokin on vähintäänkin yhtä epäselvä. Toista vain kutsutaan agentiksi, toista ei.

Jos nimittäminen agentiksi ei ole tyydyttävä määritelmä, vaatimus käytetystä teknologiasta voisi sellainen olla. Toisin sanoen *agenttiohjelma on tietokoneohjelma, joka on toteutettu agenttiteknologialla*. Agenttiteknologia ei kuitenkaan ole yksittäinen tai helposti määriteltävissä oleva teknologia, joten myös määrittelyongelmakin siirtyy koskemaan teknologiaa. Käsite agenttiteknologia on muodostunut tarkoittamaan kokoelmaa entuudestaan tunnettuja teknologioita, joita agenttiohjelmiksi kutsutut ohjelmat tyypillisesti käyttävät.

Yleisesti agenttiohjelmissa käytettyjä tekniikoita ovat mm. geneettiset algoritmit, neuroverkot, yhteisöllinen suodattaminen, ohjelman hajautettu suoritus ja tietoverkko-ohjelmointi. Agenttiteknologia

leikkaakin useita tietojenkäsittelytieteen osa-alueita, kuten (hajautettu) tekoäly (*artificial intelligence, AI, distributed artificial intelligence, DAI*), tiedon haku (*information retrieval, IR*), tiedon suodatus (*information filtering, IF*), koneoppiminen (*machine learning*), yhteisölliset järjestelmät (*collaborative systems*) ja loppukäyttäjäohjelmointi (*end user programming*). Tämän runsauden perusteella lienee selvää, ettei agentin käsitteestä saada kovin täsmällistä kuvaa ainoastaan tarkastelemalla käytettyjä tekniikoita. Toisaalta toteutustekniikkaan perustuva määrittely ei ole loppukäyttäjän kannalta erityisen relevantti. Käyttäjän huomioiminen käsitteen määrittelyssä olisi haluttavaa tutkittaessa tällaisten ohjelmien käyttöliittymiä.

Tällä hetkellä kirjallisuudessa ehkä suosituin tapa määritellä agenttiohjelma on luetella ominaisuuksia, joita jokaisella agentiksi laskettavalla ohjelmalla tulisi olla (katso Taulukko 1). Siis: *agenttiohjelma on sellainen ohjelma, jolla on ainakin osa agenteilta vaadituista ominaisuuksista.*

Lähestymistapa vaikuttaa lupaavalta, mutta pitää kuitenkin sisällään joukon ongelmia. Kirjallisuudessa on löydettävissä runsaasti eri tahojen esittämiä luetteloita, jotka määrittelevät agentilta vaadittavat ominaisuudet. Luettelot ovat osin yhteneväisiä, mutta myös eroja löytyy, mikä vaikeuttaa määritelmän käyttöä: mikä ominaisuusluetteloista on ilmiötä parhaiten kuvaava.

Esitetystä ominaisuuksien yhteenvedosta voidaan olettaa, että tärkein ominaisuus agenttiohjelmalle on sen itsenäisyys. Itsenäisyys tai autonomisuus merkitsee sitä, että ohjelman on pystyttävä suorittamaan pääasiallisesti tehtävänsä ilman jatkuvaa vuorovaikutusta käyttäjän kanssa. Perustavaksi ominaisuudeksi itsenäisyyden agenteille tekee se, että useat muut edellä esitetyistä ominaisuuksista pyrkivät mahdollistamaan agentin autonomisuuden ja sen mielekkään toiminnan. Esimerkiksi reagointi ympäristön tapahtumiin tarkoittaa olennaisesti sitä, että agentti havaitsee tapahtumia ja pystyy toimimaan havaintojensa perusteella autonomisesti. Samaan pyritään myös agentin oppimiskyvyllä, kommunikointikyvyllä, joustavuudella, tavoitesuuntautuneisuudella ja kyvyllä vai-

Taulukko 1 Agenttiohjelmilta vaadittavia ominaisuuksia eri lähteiden mukaan

[Schubert *et al.* 98]	[Brenner *et al*. 98]	[Franklin and Graesser 96]	[Caglayan and Harrison 97]
autonomy	autonomy	autonomous	autonomy
communication / cooperation	communication / cooperation	communicative	communication skills
learning ability / adaptability	reasoning / learning	learning	intelligence
reactivity	reactivity	reactive	monitoring
mobility	mobility	mobile	
goal-orientedness	proactivity / goal-oriented	goal-oriented	
	character	character	
		flexible	
		temporally continuous	
			delegation
			actuation

kuttaa ympäristöön.

On kuitenkin huomattava, että vaikka autonomisuus on agentiksi määrittelemiselle välttämätön ehto, se ei kuitenkaan ole yksistään riittävä. Agenttitutkijayhteisössä vaikuttaa vallitsevan lähes yksimielinen käsitys siitä, että esimerkiksi taustaprosesseina suoritettavat ohjelmat eivät ole agentteja. Eivät, vaikka ne voivatkin olla suorituksessaan täysin autonomisia jopa vuosikausia. Merkittävä ero agentti- ja autonomisen ohjelman välillä on tavassa, jolla käyttäjä kommunikoi ohjelman kanssa.

Usein ominaisuusluetteloon liitetään vaatimus ohjelman esittämiseksi inhimillistetyssä hahmossa. Jotkut asettavat persoonallisuuden tai luonteenpiirteen agenttiohjelman leimalliseksi tai määritteleväksi ominaisuudeksi [Laurel 90, s. 356]. Myöhemmin agenttiohjelmat ovat jakautuneet selvemmin tässä suhteessa kahteen ryhmään: antropomorfisiin ja perinteisiin käyttöliittymään luottaviin.

3.2.2 Epäsuora hallinta ja agenttiohjelmat

Koska agenttiohjelmien ydin näyttää olevan ohjelman käyttämä kommunikaatiotapa, ei harppaus vuorovaikutustapoihin ole pitkä. Epäsuora hallinta on juuri sellainen vuorovaikutustapa eli kommunikaatiomuoto, jota agentilta tunnutaan edellyttävän: se mahdollistaa ohjelman autonomisen toiminnan.

Agenttiohjelmat ja epäsuora hallinta vuorovaikutustapana ovat siten käsitteinä vahvasti kiinni toisissaan. Autonomisuus, jota agenttiohjelmilta vaaditaan edellyttää sitä, ettei käyttäjän tarvitse ohjata agentin jokaista toimintoa. Toisaalta agenttiohjelmille esitetty vaatimus kyetä reagoimaan ympäristönsä muutoksiin ohjaa ajatusta pois staattisesti etukäteen määritellyistä tehtävistä. Agentin toimintoja ei voida luetella tyhjentävästi kerralla, vaan ne muuttuvat ympäristön myötä. Käyttäjällä on siis tarve hallita agentin toimintoja epäsuorasti. Tyypillisesti hallinta tapahtuu muuttamalla sen toimintaympäristöä.

Näin ajatellen voidaan agenttiohjelmalle tarjota jopa uutta määritelmää nojautumalla epäsuoran hallinnan käsitteeseen: *agenttiohjelma on ohjelma, jonka vuorovaikutustapa on epäsuora hallinta*. Määritelmä voi olla puutteellinen käytettäväksi agenttiohjelmien tutkimuksessa, mutta kun tarkastellaan epäsuoraa hallintaa, se selkeyttää epäsuoran hallinnan ja agenttiohjelman suhdetta. Kun halutaan tutkia epäsuoraa hallintaa, voidaan tutkia agenttiohjelmien käyttöliittymiä.

Agenttiohjelmatutkimusta ajatellen määritelmässä on näkökulmaongelma. Esitetyssä muodossa siinä korostuu käyttäjän aktiivinen rooli ohjelman ohjauksessa. Sen mukaan käyttäjä ohjaa, joskin epäsuorasti, järjestelmän toimintaa ja järjestelmä reagoi käyttäjän toimiin. Asia voidaan ajatella myös toisin: agenttiohjelma tarkkailee käyttäjän toimia ja säätää toimintaansa havaintojensa perusteella. Molemmat näkemykset tarkoittavat samanlaista toimintoa käyttäjän kannalta, mutta sovelluskehittäjän näkökulmasta ero voi olla tuntuva.

Toinen potentiaalinen ongelma määritelmälle on sen kattavuus. Se sulkee agenttiohjelmien joukosta pois osan sellaisista ohjelmista, joita agenttitutkijayhteisössä on nimitetty agenttiohjelmiksi. Tämä voi olla joko tervetullutta rajanvetoa tai liiallista rajoittamista.

Esitetty määritelmä pitää ohjelman ja käyttäjän välistä vuorovaikutustapaa keskeisenä tekijänä. Siten se nostaa ohjelman käyttäjän ja käyttökokemuksen keskeisiin osiin agentin määrittelyssä. Tuodessaan käyttäjän reilusti agentin käsitteeseen mukaan, se tuntuu toimivalta määritelmältä tutkittaessa agenttiohjelmien tai laajemmin epäsuoraa hallintaa käyttävien ohjelmien käytettävyyttä ja käyttöliittymiä. Samalla se myös selventää tämän tutkimuksen suhdetta agenttitutkimukseen: jos määritelmä hyväksytään, mielenkiinto on keskittynyt enemmän ohjelman käyttöliittymään, kuten tässä työssä.

3.3 Toteutustekniikat

Epäsuoran hallinnan toteuttaminen ei ole ainakaan toistaiseksi yhtä suoraviivaista, kuin perinteisiin vuorovaikutustapoihin perustuvien käyttöliittymien toteutus. Tämä johtuu varmaankin perinteen puuttumisesta ja sitä myöden kokemuksen vähyydestä. Tämä tutkimus ottaa lähtökohdaksi edellä esitetyn agenttiohjelman määritelmän ja kartoittaa epäsuoran hallinnan toteuttamista agenttiohjelmien tai agenttiteknologioiden avulla.

Epäsuoraa hallintaa on sovellettu tiedon hakemiseen ja esittämiseen, tiedon lajitteluun ja havaittavan käyttöliittymän muuttamiseen – käsillä olevasta sovelluksesta riippuen. Se mihin käyttäjän toimiin epäsuora hallinta perustuu on sekin sovelluskohtaista. Osassa tapauksista hallinta perustuu käyttäjän viimeksi suorittamiin tehtäviin, osassa taas käyttäjän koko käyttöhistoria on ratkaisevassa asemassa.

Miten käyttäjä ohjelmaa epäsuorasti hallitsee riippuu loppujen lopuksi paljon käytetystä toteutustekniikasta. Vaikka toteutustekniikka ideaalisti valitaan sen tavoitteeseen soveltuvuuden mukaan, totuus on kuitenkin se, että toteutustekniikka säätelee voimakkaasti toteutettavia sovelluksia. Jo toteutettavaksi valitut sovellukset on tyypillisesti valittu sen mukaan, onko ohjelmaan liittyviä tavoitteita tukevia tekniikoita olemassa.

Seuraavassa luodaan katsaus erilaisiin toteutustekniikoihin, joita epäsuoran hallinnan toteuttamiseksi on käytetty. Katsauksen perusteella ymmärretään tekniikoiden luomia rajoitteita sekä periaatteita, joiden varassa epäsuoraa hallintaa voidaan käyttää. Näiden rajoitteiden, mahdollisuuksien ja periaatteiden tulee näkyä myös käyttöliittymässä, jotta myös sovelluksen käyttäjä voi ymmärtää niiden toimintaa ja tuntea hallitsevansa ohjelman käytön.

3.3.1 Implisiittisen syötteen keräys

Kaikki ohjelmat tarvitsevat syötettä, muuten ne olisivat ympäristöstään täysin eristettyjä. Epäsuorassa hallinnassa syötteen antaminen on kuitenkin erityislaatuista, sillä pyrkimyksenä on vapauttaa käyttäjä syötteen antamisesta aiheutuneesta vaivasta. Tätä epäsuorassa hallinnassa käytettyä syötteen muotoa kutsutaan tässä työssä implisiittiseksi syötteeksi.

Ohjelman sanotaan saavan *implisiittistä syötettä*, jos käyttäjä on tarkoittanut syötteen muuhun tarkoitukseen, kuin mihin ohjelma sitä käyttää. Ajatellaan esimerkkitilannetta, jossa käyttäjä vierittää dokumentin sivua liukusäätimellä. Tapahtumassa käyttäjä antaa eks-

plisiittistä syötettä sivun näkyvää osaa koskien. Samassa hän saattaa tulla antaneeksi myös implisiittistä syötettä, jos ohjelma tekee *saman* syötteen perusteella myös jotain muuta. Se saattaisi esimerkiksi tulkita annetun syötteen tarkoittavan sivun kiinnostavuutta ja tallentaa tämän tiedon.

Implisiittisen syötteen keräys perustuu yleensä erilaisten heuristiikkojen käyttöön. Syötteen tulkita perustuu tällöin sen yleisen merkityksen ymmärtämiseen ja tämän tietouden soveltamiseen annetussa tilanteessa. Usein heuristiset säännöt ovat sovellusalueriippuvia, joten niiden avulla on vaikea luoda yleisesti toimivaa järjestelmää. Jotkin heuristiset säännöt ovat kuitenkin luonteeltaan yleisempiä.

Heuristiset arviot kohdistuvat ohjelman käyttöliittymässä tapahtuviin vuorovaikutusrepliikkeihin (*dialog-act*). Repliikit voidaan jaotella repliikkityyppeihin [Pohl *et al.* 95], jolloin käyttäjän lausuma yksittäinen repliikki on instanssi repliikkityypistä. Tämän käsitteen avulla repliikkien käsittely on tehokkaampaa, koska tyypin avulla yksittäisen repliikin abstraktiotasoa pystytään nostamaan.

Esimerkkejä yleisesti sovellettavista vuorovaikutusrepliikkityypeistä ovat [Pohl *et al.* 95]:

1. Tiedon pyynnöt, jotka voidaan esittää suomen kielen kysymyssanoilla ‘mikä’, ‘miten’, ‘milloin’, …
2. Kyllä -vastaukset esitettyihin kysymyksiin
3. Myöntyminen esitettyihin ehdotuksiin

Tiedon pyyntö -tyyppiin kuuluvan repliikin suorittamisesta voidaan yleisen heuristisen säännön perusteella päätellä, että pyytäjä ei tunne aihetta, josta pyytää tietoa. Vastaavasti kyllä - ei kysymykseen myöntävästi vastaamisesta voidaan päätellä vastaajan haluavan kysymyksen osoittavan asiantilan tulevan voimaan. Myöntymisessä esitettyyn ehdotukseen voidaan tulkita pitkälti samoin.

Myös muita heuristiikkoja on implisiittisen syötteen keräämiseksi käytetty. Esimerkiksi tietoalkioon tutustumiseen käytettyä aikaa on pidetty sen kiinnostavuuden mittarina [Morita and Shinoda 94]. On mahdollista kerätä tietoa myös yleensä tietoalkioiden katselusta tai dokumentin sivun vierittämisestä [Sakagami and Kamba 97]. Tällaisia päättelyjä voidaan tukea käyttöliittymän suunnittelulla, jolla voidaan esimerkiksi vähentää tietoalkioiden määrää, jotka voivat olla yhtäaikaa huomion keskipisteessä. Tällöin päättelyn tarkkuus paranee.

Lisäksi on kokeiltu käyttäjän tuntemien käsitteiden ja komentojen mallintamista käytettyjen komentojen perusteella [Sukaviriya and Foley 93, s. 204]. Käytettyjen toimintojen ajatellaan heijastavan käyttäjän järjestelmän tuntemusta. Toiminnot voidaan luokitella, jolloin tiettyyn ryhmään kuuluvien toimintojen käyttämisen katsotaan merkitsevän laajempaa tietämystä kuin toisiin ryhmiin kuuluvien.

Vaikka heuristisia ratkaisuja on periaatteessa rajaton määrä, käytössä ei ole kovin kirjavaa joukkoa. Käyttökelpoisia menetelmiä rajoittaa käytettävissä olevat toteutustekniikat. Niinpä useat tekniikat perustuvat melko yksinkertaisten primitiivien havaitsemiseen. Kuten edellä esitetyistä esimerkeistä käy ilmi, primitiiveinä on tyypillisesti yksittäisiä toimintoja, kuten annettu komento, vastaus tai työhön käytetty aika.

3.3.2 Kiinnostuksen mallinnustekniikat

Kiinnostuksen mallinnuksella pyritään mahdollistamaan käyttäjäkohtainen tietosisällön, toimintojen ja toimintatapojen mukauttaminen. Toisin sanoen, sovellus pyrkii päättelemään, millaisista asioista käyttäjä on kiinnostunut tai miten hän haluaa työnsä tehdä. Käyttäjään liittyviä tietoja tallentaa ja ylläpitää työhön erikoistunut ohjelmakomponentti, jota kutsutaan käyttäjämalliksi *(user model)* tai käyttäjäprofiiliksi (*user profile*).

Käyttäjämallin rakenne ja sisältö vaihtelevat voimakkaasti riippuen sovelluksesta. Se voi sisältää yksinkertaisia lueteltuja ominaisuuksia ja niiden arvoja. Tällä tekniikalla käyttäjämalli voi esimerkiksi tallentaa käyttäjän osaamistason tiettyjen ennalta määrättyjen ominaisuuksien suhteen tai hänen mieltymyksiään esitystapojen suhteen [Strachan et al. 97, s.193]. Tällaiset ennalta määriteltyihin arvoihin perustuvat käyttäjämallit eivät kuitenkaan ole kovin joustavia, mikä rajoittaa niiden yleistä sovellettavuutta. Ne ovat kuitenkin nopeita ja yksinkertaisia toteuttaa.

Toinen käytetty tekniikka käyttäjän mieltymysten mallintamisessa on tietoalkioiden ja niihin liitettyjen kiinnostusta kuvaavien tunnuslukujen tallentaminen. Tietoalkiot voivat kuvata, sovelluksesta riippuen, vaikkapa tekstidokumentteja tai WWW-sivuja. Yleisenä ideana tällaisen käyttäjämallin hyödyntämisessä on uuden tietoalkion vertaaminen vanhaa vastaavaa tietotyyppiä olevaan tietoalkioon. Uuden tietoalkion kiinnostavuus päätellään vanhaan tietoalkioon liitetyn tunnusluvun ja tietoalkioiden samanlaisuuden perusteella. Jos tietoalkiot ovat samanlaisia, uuden kiinnostavuus on sama kuin vanhan, muuten oletettu kiinnostavuus vähenee tietoalkioiden erilaisuuden myötä.

Yleinen käyttäjämalleissa käytetty tietoalkioiden tallennustekniikka on *painotettu avainsanavektori*, kun tietoalkioon liittyy sanallinen esitysmuoto. Painotetut avainsanavektorit ovat suosittuja ehkä siksi, että ne soveltuvat suureen määrään erilaisia sovelluksia edellyttäessään vain tekstimuotoista tietoa. Tekniikka on myös melko yksinkertainen, hyvin ymmärretty ja se mahdollistaa kahden tietoalkion eron arvioinnin.

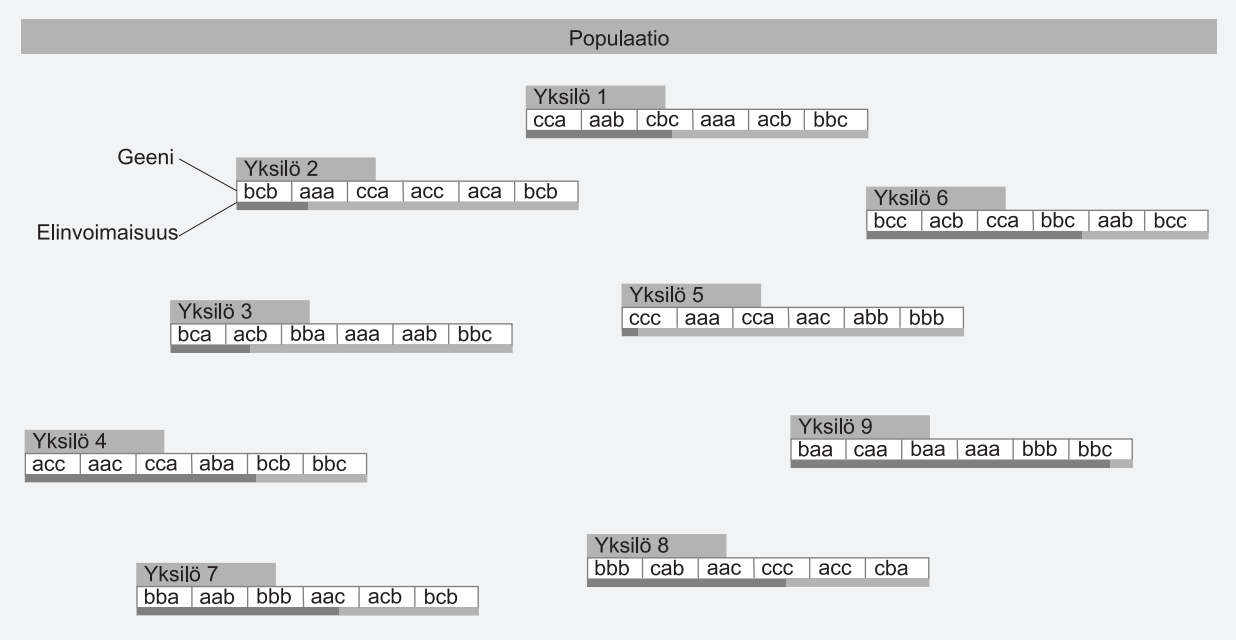
Painotettu avainsanavektori koostetaan sanallisesta dokumentista erottelemalla siitä sen sisältöä parhaiten kuvaavat sanat ja antamalla näille painoarvot. Painoarvot saadaan esimerkiksi *tfidf* menetelmällä [Salton 89], joka lasketaan kaavalla:

$$paino = termin\ frekvenssi\ dokumentissa * \frac{1}{termin\ frekvenssi\ kaikissa\ dokumenteissa}$$

Menetelmä painottaa sellaisten termien merkitystä, jotka esiintyvät runsaina mallinnettavassa dokumentissa, mutta harvoin muissa dokumenteissa. Paljon kaikissa dokumenteissa esiintyvien termien merkitys puolestaan pienenee, joten esimerkiksi yleinen ‘olla’ verbi ei kuvaa minkään dokumentin sisältöä hyvin.

3.3.3 Oppimistekniikat

Ohjelmat, jotka mallintavat käyttäjän kiinnostuksen kohteita, ovat tyypillisesti myös oppivia. Oppivat ohjelmat pyrkivät kehittämään tai muuttamaan toimintaansa ympäristönsä mukaan, jotta niiden toiminta olisi mielekästä myös ympäristön muutosten jälkeen. Oppiva ohjelma pystyy toimimaan hyvin myös sellaisissa tapauksissa,



Kuva 4 Geneettisen ohjelman rakenne

joita ohjelmoija ei ole ottanut huomioon. Tosin ohjelmat oppivat vain tiettyjä asioita, joita ne on ohjelmoitu oppimaan – ohjelmoijan työstä ne eivät siis suinkaan ole riippumattomia.

Tietokoneohjelmien oppiminen perustuu, inhimillisestä maailmasta otetun esikuvan tavoin, palautteeseen. Yksinkertaisuudessaan ohjelma suorittaa toiminnon ja saa siitä palautteen. Riippuen palautteesta se pyrkii joko vahvistamaan tai heikentämään toimintoa. Erilaisia algoritmeja oppimisen toteuttamiseksi on kirjallisuudessa olemassa melko runsaasti, mutta kaikki perustuvat tähän samaan perusideaan.

Geneettisiä algoritmeja on sovellettu esimerkiksi oppivissa agenteissa [Moukas 96]. Sovelluksessa on määriteltävä käytetty geeni, johon koodataan ohjelman käyttämä tieto ja johon ohjelman toiminta perustuu. Geenit ovat populaation jäsenillä, joiden lisääntymistä säädellään niiden elinvoimaisuuden perusteella (Kuva 4). Elinvoimaisuus voi perustua vaikkapa käyttäjän palautteeseen. Ne populaation jäsenet, joiden elinvoimaisuus on korkea, saavat eniten mahdollisuuksia geeninsä eli perimänsä välittämiseen jälkipolville. Tässä evoluutioksi kutsutussa prosessissa osaan geeneistä aiheutetaan satunnaisia muutoksia, mutaatioita.

Amalthaea-järjestelmä suodattaa järjestelmän hankkimasta tietomassasta käyttäjää kiinnostavan osan. Sen geenistö mallintaa käyttäjän kiinnostuksen kohteet painotettuina avainsanoina. Oppimistulosta sovelletaan hyvin suoraviivaisesti: hyväksytyksi tulevat sellaiset tietoalkiot, joista muodostettu avainsanavektori muistuttaa suodattajan geeniä, eli avainsanavektoria. Palautteen, joka määrittelee populaation jäsenten elinvoimaa, käyttäjä antaa tässä esimerkissä eksplisiittisesti.

Muistiin perustuva päättely (*memory based reasoning*) on toinen käytetty tekniikka. Siinä ohjelman oppiminen perustuu muistiin, johon on tallennettu konteksti - toiminto pareja. Muistin sisältöä käyte-

tään etsimällä nykyisen kaltaista tilannetta muistista ja soveltamalla siihen liittyvää toimintoa tähän uuteen tilanteeseen. Tyypillisesti toiminnot ovat käyttäjän suorittamia, jotka ohjelman oppimisesta vastaava osa on tallentanut yhdessä relevantin kontekstin kera.

Sähköpostin käsittelyssä avustava Mail Agent [Lashkari *et al.* 94] oppii tällaisella tekniikalla. Jos käyttäjä esimerkiksi siirtää saapuneen sähköpostin toiseen kansioon, järjestelmä muistaa tämän toiminnon ja siihen liittyvän tilanteen. Se, mitä tilanteesta tallennetaan, on muistissa olevan tiedon soveltamisen kannalta kriittistä ja Mail Agentin kohdalla muisti sisältää mm. tiedot sähköpostin lähettäjästä ja aiheesta. Näiden tallennettujen tapausten perusteella se pyrkii päättelemään oikean (käyttäjän mielestä) toimenpiteen uudessa tilanteissa.

Koneoppimistekniikoissa on usein ongelmana oppimisen hitaus tai kyvyttömyys toimia täysin uudessa tilanteessa. Esimerkiksi muistiin perustuvaa päättelyä käyttävä ohjelma voi helposti joutua tilanteeseen, jollaista sillä ei muistissaan vielä ole. Tässä tilanteessa ohjelma on toimintakyvytön. Ongelma voidaan osittain ratkaista käyttämällä *yhteisöllisiä tekniikoita.* Tällaisissa tekniikassa joukko ohjelmainstansseja toimii yhdessä ratkoen ongelmia yhteisten kokemusten perusteella.

Ongelma voidaan ratkaista yhteisöllisesti etsimällä ohjelmasta toinen instanssi, joka muistuttaa ensimmäistä. Asiallisesti tämä tarkoittaa samankaltaisen käyttäjän etsimistä ja tilanteeseen reagointi suoritetaan toisen käyttäjän tekemien toimien perusteella. Jos käyttäjät toimivat tilanteissa samoin, tekniikka on toimiva. Käytännön toimivuus riippuu voimakkaasti myös samankaltaisten käyttäjien etsinnän toteutuksesta.

Yhteisöllisiä tekniikoita voidaan soveltaa oppimisessa myös yksinkertaisemmin. Järjestelmä voi yksinkertaisesti tallentaa tietoalkioita ja niihin liittyviä käyttäjien antamia arviointeja. Näiden tietojen perusteella voidaan muodostaa käyttäjäryhmiä, joilla on samanlainen maku tietoalkioiden suhteen. Tällaisten käyttäjäryhmien sisällä voidaan tietoa levittää, jolloin käyttäjän näkökulmasta ohjelma on oppinut etsimään häntä kiinnostavaa tietoa. Tällaista tekniikkaa käytetään esimerkiksi Firefly-palvelussa [Firefly 97], joka suosittelee käyttäjille kiinnostavia elokuvia ja äänilevyjä.

3.3.4 Yleistys- ja päättelytekniikat

Esimerkkiperustainen ohjelmointi pyrkii helpottamaan loppukäyttäjäohjelmointia. Käyttäjän on mahdollista esittää esimerkki haluamansa ohjelman toiminnasta, jonka perusteella varsinainen ohjelman luodaan automaattisesti. Tässä prosessissa puhutaan oppimisen sijasta päättelystä tai yleistämisestä, sillä ohjelman generointi esimerkin perusteella edellyttää halutun toiminnon päättelyä ja havaittujen toimien yleistämistä.

Eräs käytetty tekniikka on *ehto - toiminto säännöt* (*condition-action rules*) [Myers 93]. Tekniikkaa käyttävä sovellus sisältää säännöstön, joka koostuu tilaa kuvaavista ehdoista ja niihin liittyvistä toiminnoista. Annetussa tilanteessa suoritettava toiminto päätetään etsimällä säännöstöstä nykyistä tilannetta kuvaava ehto ja valitsemalla siihen liitetty toiminto.

Tällaista tekniikkaa on käytetty esimerkiksi Peridot-järjestelmässä [Myers 93], jolla voidaan rakentaa graafisia käyttöliittymiä esimerkkiperustaisesti. Sen säännöstö sisältää tietoa sovellusalueesta: toisiinsa liittyvien objektien graafisista rajoitteista, kontrollirakenteiden käytöstä, muuttujien ja vakioiden erosta ja hiiren toiminnasta käyttöliittymässä. Näiden sääntöjen perusteella se kykenee yleistämään käyttäjän esimerkkejä.

Toinen, jossain määrin ehkä yksinkertaisempi tekniikka, on kaavaan sovittaminen (*pattern matching*). Tekniikassa on sääntöjen sijaan kuvattu yleisiä kaavoja, joiden katsotaan kuvaavan järjestelmän käyttöä tai toimintaa. Annetussa tilanteessa haluttu toiminto löydetään etsimällä tilannetta vastaava kaava ja valitsemalla toiminto löydetystä kaavasta. Ero ehtoihin on tilanteen kuvaamisessa käytetty kieli. Ehdot on tallennettu loogisina lauseina, jotka kuvaavat tilanteen, kaava koostuu konkreettisemmasta tilanteen kuvauksesta. Kaavan ilmaisuvoima on pienempi kuin ehdon.

Kaavaan sovittamisen heikkoutena onkin sen riippuvuus kaavojen kattavuudesta. Ei ole helppoa laatia sellaista toimintakaavojen kokoelmaa, joka kattaisi todellisissa käyttötilanteissa esiintyvän monimuotoisuuden.

Kaavoihin sovittamista voidaan soveltaa myös hieman abstraktimmalla tasolla kuin luetellen mahdollisia toimintosarjoja. Esimerkiksi Eager-järjestelmä [Cypher 93] etsii ohjelman suorituksesta samanlaisia toimintosarjoja, komentoja tai objekteja, joihin komennot kohdistuvat. Se tunnistaa joitain ennalta lueteltuja järjestettyjä joukkoja, kuten esimerkiksi kuukauden nimiä ja järjestysnumeroita. Näiden kaavojen lisäksi Eager etsii toistuvia toimintosarjoja, joita käytetään automatisoidun ohjelman runkona. Ohjelman suorittamat komennot perustuvat käyttäjän antamiin komentoihin, kun ne on yleistetty niihin sopivien kaavojen avulla.

Sääntöihin perustuvia tekniikoita on olemassa myös muita ja useita niistä on käytetty myös esimerkein ohjelmoitavissa ohjelmissa. Yhteistä sääntöperustaisille tekniikoille on annetun tilanteen vertaaminen järjestelmän tuntemiin sääntöihin ja toimiminen säännöstössä esitetyllä tavalla.

3.3.5 Tehtävien mallinnustekniikat

Erityisesti ohjelmissa, jotka pyrkivät tarjoamaan käyttötilanteeseen kiinteästi liittyvää apua, käytetään erilaisia tekniikoita käyttäjän suorittamien tehtävien seurantaan. Avun mukauttamisella toisaalta käyttäjän tietotasoon ja toisaalta vallitsevaan käyttötilanteeseen voidaan helpottaa tarjotun avun omaksumista. Nämä ovat huomioitavia asioita, sillä käyttäjät toimivat ympäristössä, jossa tehtävät ja tehtäväympäristö muuttuvat alati [Kühme and Schneider-Husfschmidt 93, s. 3]. Ollakseen käyttäjälleen mahdollisimman hyödyllisiä tarjottujen ohjeiden tulee ottaa nämä muuttuvat olosuhteet huomioon.

Toimintaympäristöön mukautuvien ohjejärjestelmien toteutus perustuu usein *tehtävämalleihin*. Tehtävämalli on sovelluksen osa, joka eksplisiittisesti kuvaa sovelluksella suoritettavia tehtäviä. Näihin malleihin perustuen on mahdollista seurata käyttäjän yksittäisiä toimia ja ymmärtää mitä suurempaa tehtävää hän on suorittamassa.

Tekniikassa yksinkertaisesti seurataan käyttäjän suorittamia ohjelmallisesti havaittavia matalan tason toimia ja etsitään tehtävämallista osatehtäviä, joihin nämä toimet kuuluvat. Yhden tai useamman tunnistetun osatehtävän jälkeen voidaan tehtävämallin perusteella päätellä mitä ylätason tehtävää käyttäjä on suorittamassa. Toisin sanoen tehtävämalli sisältää kuvauksen matalan tason tapahtumista ylätason tehtäviin [Hoppe 93, s. 167].

Tehtävämallien toteutukseen viitataan kirjallisuudessa usein käsitteellä *tietämyskanta*. Tietämyskanta on tietovarasto, jossa tieto on esitetty muodossa, joka mahdollistaa johdetun tiedon muodostamisen. Tieto voidaan tallentaa tällaiseen kantaan esimerkiksi sääntöinä, joista voidaan johtaa yksityiskohtaisia tietoja.

Tietämyskannan konkreettinen toteutustapa voi vaihdella voimakkaasti riippuen sovellusalueesta ja tarvitun tehtävämallin luonteesta. ACPA-järjestelmässä [Johnson *et al.* 99, s. 127] tehtävämallina toimii äärellinen tila-automaatti. Tehtävien osat kuvataan siinä tiloina, joiden välillä on tilasiirtymiä. Tunnistamalla tilat käyttäjän suorittamista matalan tason toimista päästään käsiksi tehtävään, johon tila kuuluu. Lähestymistavan etuihin kuuluu sen yksinkertaisuus sekä mahdollisuus helposti ja ymmärrettävästi visualisoida tehtävämalli.

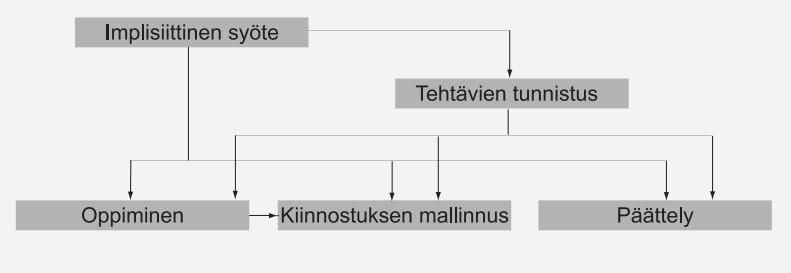
Tehtävien kuvaamiseksi on kehitetty myös tarkoitukseen soveltuvia kieliä. LEXITAS [Hoppe 93] on kieli, joka käyttää attribuuttikielioppia tehtävien kuvaamiseen. Siinä kielen terminaalisymbolit määrittelevät perustavat tai atomiset tehtävän osat, jotka voidaan kuvata suoraan käyttäjän ohjelmallisesti havaittaviin toimiin. Kielenä LEXITAS:in käyttö perustuu syötteen jäsentämiseen. Näin sen soveltamisessa voidaan käyttää hyödyksi jäsentämisestä nykyistä tietoutta. Kielen laatijat mainitsevat esitystavasta pystyttävän tuottamaan myös käyttäjälle mielekkäitä selityksiä tehtäväkuvauksista. Tämä on oleellinen piirre tehtävämallille, jos ja kun sen halutaan olevan ymmärrettävä myös käyttäjälle.

3.3.6 Tekniikoiden väliset suhteet

Epäsuoran hallinnan toteuttavilla tekniikoilla on monimutkaisia suhteita, jotka voivat vaikeuttaa järjestelmien toiminnan ymmärtämistä. Edellä esiteltyjä tekniikoita ei juurikaan sovelleta tai edes voida soveltaa yksinään. Tyypillisesti niitä käytetään yhdessä täydentämään toisiaan.

Implisiittisen syötteen keräystä voidaan pitää jossain mielessä perustavana tekniikkana, joka muodostaa perustan epäsuorasti hallittavan ohjelman toteutukselle. Se mahdollistaa käyttäjän ja sovelluksen epäsuoran kommunikaatiotavan. Implisiittisen syötteen keräämisestä sinällään ei välttämättä ole kuitenkaan suurtakaan hyötyä. Usein se toimiikin esimerkiksi yhteistyössä ohjelman oppivan osan kanssa tarjoten sille oppimisessa tarvittavaa palautetta.

Kiinnostusta mallinnettaessa voidaan myös käyttää implisiittistä syötettä mielenkiintoa kuvaavien tunnuslukujen muodostamisessa. Kiinnostusprofiilit eivät ole staattisena kovinkaan hyödyllisiä, joten niihin liittyy tyypillisesti oppimiskomponentti, joka pitää profiilin ajan tasalla. Kiinnostusprofiili onkin vastuussa lähinnä vain kiinnostustietojen tallentamisesta ja vertailujen tekemisestä.



Kuva 5 Epäsuoran hallinnan toteutustekniikoiden suhteet

Myös tehtävien tunnistamisessa käytetään tyypillisesti implisiittistä syötettä. Eksplisiittisen syötteen käyttäminen tarkoituksessa ei tunnu mielekkäältä, se kun tarkoittaisi tehtävän siirtämistä käyttäjälle. Myös päättelyiden ja yleistysten tekeminen on tyypillisesti riippuvaisia implisiittisestä syötteestä.

Kuva 5 esittää epäsuoran hallinnan toteuttamiseksi käytettyjen tekniikoiden väliset suhteet esittämällä tiedon kulun nuolin tekniikasta toiseen. Kuvio korostaa implisiittisen syötteen perustavaa merkitystä, sillä se vaikuttaa kaikkien muiden tekniikoiden toimintaan. Yllättäen myös tehtävien tunnistamisella on varsin monipuolisia sovellusmahdollisuuksia. Tämä johtuu tiedon nopeasta saatavuudesta. Tehtävä tunnistetaan varhaisessa vaiheessa, joten tietoa olisi mahdollisuus käyttää hyväksi monella eri tavalla.

3.3.7 Epäsuorasti hallitun ohjelman rakenne

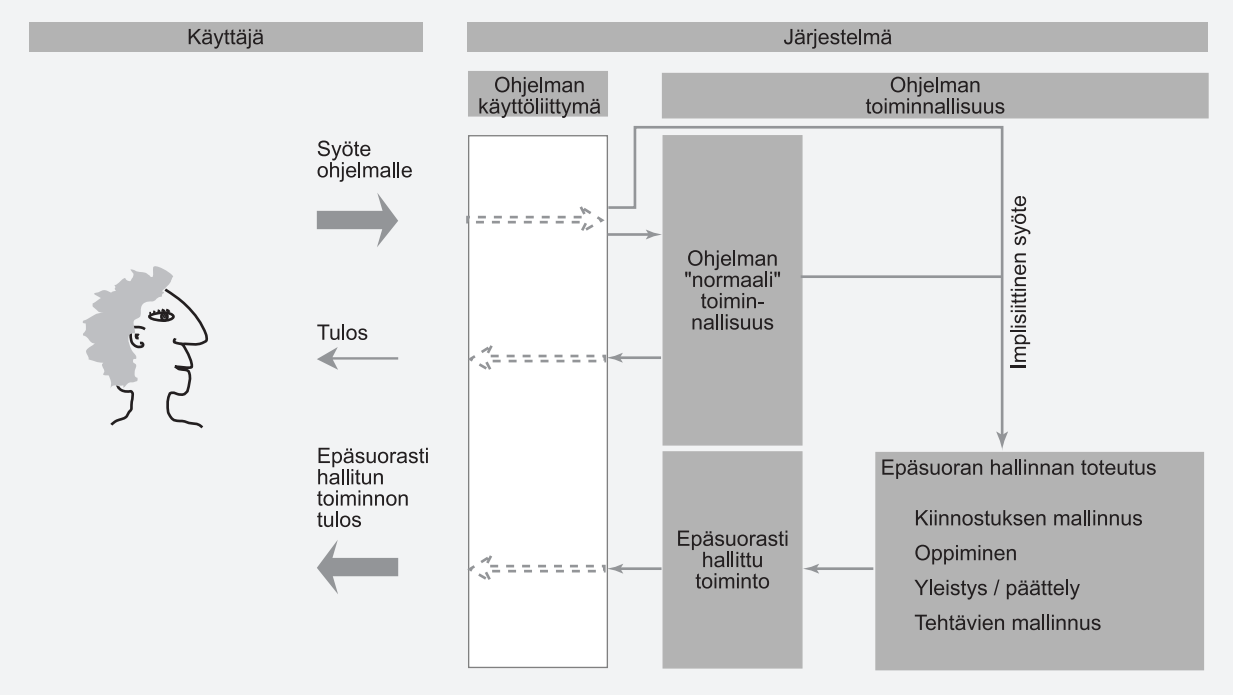
Jotta epäsuorasta hallinnasta saataisiin yhtenäinen ja ymmärrettävä käsitys on paikallaan tarkastella miten eri tekniikoita kootaan yhteen ja integroidaan ohjelman muun toiminnallisuuden kanssa. Kuva 6 esittää kaavion keinoin tekniikoiden paikat ja yhteydet ympäristöön.

Oleellisinta kuviossa on epäsuorasti hallitun toiminnon ja epäsuoran hallinnan toteutuksen erottaminen toisistaan sekä ympäröivästä ohjelmaympäristöstä. Kuvion ‘Epäsuorasti hallittu toiminto’ on jokin sellainen käyttäjälle näkyvä ja häntä hyödyttävä toiminto, jonka ohjaaminen tapahtuu epäsuorasti. Esimerkiksi sähköpostin lajitteluagentin yhteydessä tämä tarkoittaisi lajittelutoimintoa. Se käynnistetään ja lajitteluperusteet valitaan implisiittisen syötteen perusteella.

Osa ‘Epäsuoran hallinan toteutus’ viittaa niihin ohjelmistokomponentteihin, jotka toteuttavat epäsuorasti hallitun toiminnon ohjaamisen implisiittisen syötteen perusteella. Tämän komponentin toteutus turvautuu edellä kuvattuihin tekniikoihin luodessaan merkitystä implisiittiselle syötteelle ja päättäessään milloin ja miten epäsuorasti hallittu toiminnallisuus käynnistetään. Sähköpostin lajitteluesimerkissä tämä komponentti päättää milloin ja millä perusteella lajittelu tapahtuu.

Käsitteiden erottelulla saadaan selkeyttä ilmiön tarkasteluun ja muodostetaan yhtenäistä käsitteistöä siitä puhumiseen. Näiden käsitteiden osalta, tutkimuksen loppuosan käsitteistö viittaakin tähän kuvioon ja sen takana olevaan ajatukseen.

Kuviossa epäsuoran hallinnan toteutus saa syötettä kahta reittiä. Se voi havainnoida suoraan käyttöliittymän tapahtumia tai saada tie-



Kuva 6 Epäsuorasti hallitun ohjelman rakenne

toa ohjelman tilamuutoksista. Ensimmäinen tapa on sovellusriippumaton, mutta syötteen abstraktiotaso on matala. Toinen menetelmä sitoutuu tiettyyn sovellusalaan tai sovellukseen ja saa merkitykseltään selvempää syötettä. Kahta eri syötekanavaa voidaan käyttää myös rinnan.

Kuviossa on esitetty tilanne, jossa epäsuorasti hallittu toiminnallisuus on osa ohjelman koko toiminnallisuutta. Käyttäjän sähköpostia lajitteleva agentti voitaisiin toteuttaa tällaisella organisoinnilla. Epäsuorasti hallittu toiminto voisi olla myös itsenäinen ohjelma. Tällöin kuvion rakenneosa ‘Ohjelman normaali toiminnallisuus’ tarkoittaa jotain muuta itsenäistä ohjelmaa, jonka kautta epäsuoran hallinnan toteutus saa implisiittistä syötettä. Tällöin ohjelmilla on luonnollisesti myös eriävät käyttöliittymät.

3.4 Toteutustekniikoiden sivuvaikutuksia

Edellä esitellyt tekniikat mahdollistavat epäsuoran hallinnan toteuttamisen käyttöliittymän vuorovaikutustavaksi. Mahdollisia tapoja toteuttaa epäsuora hallinta on olemassa varmasti muitakin, mutta tässä työssä keskitytään edellä esiteltyihin, sillä niiden soveltamisesta on olemassa kokemusperäistä tietoa sekä agenttiohjelmista että älykkäiden käyttöliittymien tutkimuksesta.

Sen lisäksi, että tekniikat mahdollistavat epäsuoran hallinnan toteuttamisen, niillä on myös sivuvaikutuksia. Käytetyllä tekniikalla on tapana vaikuttaa toteutettavaan ilmiöön. Toteutettava asia ja sen toteuttamiseen käytetty tekniikka eivät ole irrallisia, vaikka ideaalinen toteutus tätä joskus vaatisikin. Esimerkiksi käyttöliittymien toteutukseen on pitkään vaikuttanut toteutustekniikan (ohjelmointikielien ja yleisemmin tietokoneiden) komentoperustai-

suus. Osaltaan tästä syystä useissa käyttöliittymissä toimiminen on organisoitu komentojen antamiseksi.

Se, että käytettävä tekniikka vaikuttaa toteutukseen, ei ole yksinomaan huono asia. Jos toteutustekniikka näkyy myös käyttäjälle, tekee se ohjelman toimintaperiaatteiden ymmärtämisen helpommaksi. Siksi on syytä tutkia toteutustekniikoiden ominaisuuksia huolellisesti, erityisesti käyttäjän kannalta. Tällaisen analyysin perusteella voidaan havaita, mitä ominaisuuksia tekniikasta on erityisesti syytä korostaa, kun mietitään millaisena ne käyttäjälle esitetään.

Edellisessä tekniikoiden kartoituksessa kävi ilmi, että epäsuoraan hallittu ohjelma tyypillisesti tarkkailee käyttäjän yksittäisiä toimenpiteitä. Tämä on toiminto, jota nykyisissä käyttöliittymissä ei useinkaan ole toteutettu ja siten toiminto on vieras käyttäjille. Toiminto on vieras nimen omaan osana tietokoneen ja ihmisen vuorovaikutusta, ihmisten välisessä kanssakäymisessä se ei liene outoa. Ihmisten välisessä kommunikaatiossa on osittain mahdollista havaita, millaisiin asioihin toinen osapuoli huomiotaan kiinnittää ja näin päätellä millaisiin asioihin hän saattaa keskustelussa reagoida.

Epäsuoraa hallintaa käyttävät ohjelmat saattavat myös tallentaa tietoalkioihin liittyviä kiinnostusta kuvaavia tunnuslukuja rakentaessaan käyttäjän kiinnostusprofiilia. Käyttäjään liittyvien tietojen tallentaminen on sekin vieras toiminto nykyisissä käyttöliittymissä, niinpä myös se on mahdollisesti käyttäjän kannalta outo toiminto.

Ihmisen ja koneen vuorovaikutuksessa ihminen on tyypillisesti muuttuva osapuoli ja kone mekaanisesti toimiva ja siten ennustettava. Epäsuora hallinta muuttaa tämän suhteen, sillä esitettyjä tekniikoita käyttävä ohjelma kykenee muuttamaan toimintaansa käyttäjän antaman palautteen perusteella. Käyttäjän näkökulmasta ohjelman toiminta muuttuu, mahdollisesti ennustamattomalla tavalla. Erityisen ongelmalliseksi tämän voi tehdä se, että ohjelman toiminta muuttuu epäsuoran hallinnan kautta, jolloin käyttäjä ei ehkä tiedä aiheuttaneensa muutosta.

Tietoverkot mahdollistavat eri ohjelmainstanssien toimimisen keskenään yhteistyössä. Kuten jo edellä nähtiin, tästä on saatavissa myös todellista hyötyä esimerkiksi oppivien ohjelmien kohdalla, joten on oletettavaa, että tekniikkaa myös käytetään. Käyttäjän näkökulmasta tämä saattaa aiheuttaa tietoturvaongelmia. Käyttäjä ei välttämättä tiedä mitä tietoja hänestä näin levitetään ja mihin tietoja käytetään. Käyttäjä saattaa olla myöskin tietämätön tietojen levittämisestä. Tekniikka mahdollistaa hyvin helposti tietojen levittämisen hiljaisesti, käyttäjän huomaamatta. Paljastuessaan tällainen toiminto voi aiheuttaa käyttäjissä turvattomuuden tunnetta.

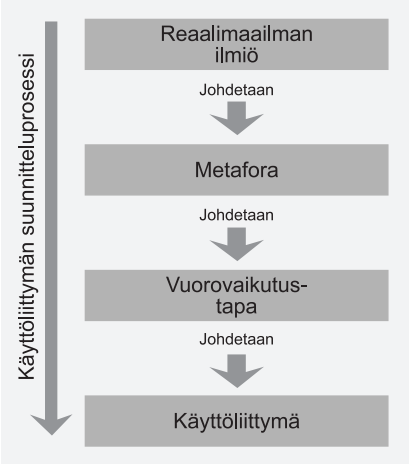
Ihmisavustajat päättelevät haluttuja toimintoja perustuen kokemukseen, sääntöihin tai kaavoihin. Kun samat periaatteet siirretään tietokoneympäristöön, voi käyttäjä olla ymmällään. Tämän voidaan ajatella johtuvan kahdesta syystä: 1) käyttäjät eivät odota, että tietokone päättelee mitään perustuen heidän toimiinsa ja 2) tietokoneen suorittamat päättelyt saattavat käyttäjän mielestä olla epäonnistuneita. Käyttäjän näkökulmasta tapahtuneet epäonnistumiset tarkoittavat yksinkertaisesti sitä, että ohjelma ei noudattanut sitä toimintamallia, jota käyttäjä oletti sen noudattavan. Tällöin käyttäjän ymmärrys ohjelman toiminnasta murenee, vaikka ohjelma toimisikin määrittelynsä suhteen oikein.

Epäsuoraa hallintaa vuorovaikutustapana käyttävät ohjelmat voivat tunnistaa käyttäjän suorittamia tehtäviä heidän suorittamiensa matalan tason tapahtumien perusteella. Myös tällainen toiminnallisuus on vierasta tyypillisissä tietokoneohjelmissa. Käyttäjän havainto toiminnosta riippuu paljolti siitä mihin sitä käytetään ja miten onnistuneesti se on toteutettu. Jos tehtävien tunnistusta käytetään esimerkiksi avusteiden muokkaamiseen, toiminto on melko harvoin käytetty ja käyttäjälle lähes näkymätön. Tällöin sen toimintaperiaatetta voi olla vaikea ymmärtää.

Useat mainitut sivuvaikutukset aiheuttavat ongelmia vain, kun ne on yhdistetty ihmisen ja tietokoneen väliseen vuorovaikutukseen. Tämä on odotettavaa, sillä epäsuoran hallinnan metafora on haettu ihmisten välisestä vuorovaikutuksesta, eikä metaforan soveltaminen ole suoraviivaista. Sivuvaikutukset ovat monilukuiset ja niillä on potentiaalisesti voimakkaitakin negatiivisia seurauksia ohjelman käytettävyydelle. Tällaisten käyttöliittymien rakentamisessa vielä melko harvinaisten tekniikoiden käytössä on huolellisesti suunniteltava niiden vaikutus ohjelman käyttöliittymälle. Näin negatiivisia vaikutuksia voidaan vähentää.

4. KÄYTTÖLIITTYMÄVAATIMUKSET

4.1 Epäsuoran hallinnan tavoitteet ja vaatimukset



Kuva 7 Käyttöliittymäsuunnittelussa vaikuttavien käsitteiden suhteet

Tietokoneohjelman käyttöliittymä on rajapinta ihmisen ja tietokoneohjelman välillä. Se pyrkii mahdollistamaan joustavan, tehokkaan, mielekkään ja miellyttävän tavan työskennellä järjestelmän avustuksella. Käyttöliittymä on osa koko järjestelmää, jolla on jokin tavoite: tyypillisesti mahdollistaa tai helpottaa käyttäjää suorittamaan jokin tehtävä. Käyttöliittymä on siis yksi komponentti, joka määrittelee koko järjestelmän toimivuuden sen tavoitteen suhteen [Mayhew 92, s. 6].

Seuraavassa tarkastellaan tarkemmin sitä, millaisia tavoitteita ja vaatimuksia epäsuoran hallinnan käyttö asettaa ohjelman käyttöliittymälle. Tarkastelu aloitetaan yleiseltä tasolta eli siitä, mitä tavoitteita valittu vuorovaikutustapa asettaa. Tämän jälkeen voidaan tarkastella vaatimuksia, jotka johtuvat epäsuoran hallinnan tähän mennessä tunnetuista ongelmista. Lopuksi voidaan vielä johtaa edellisistä ja nykyisen käyttöliittymätutkimuksen tuloksista yleisluontoisia vaatimuksia, jotka jäsentävät myöhemmin myös ratkaisujen tarkastelua.

4.1.1 Vuorovaikutustavan tavoitteet

EPÄSUORAN HALLINNAN TAVOITTEET

- ei tarvetta opetteluun
- käyttö huomaamatonta
- käyttö vaivatonta

Puhuttaessa vuorovaikutustavoista mainittiin, että käyttöliittymät voidaan luokitella sen mukaan, millaista vuorovaikutustapaa niissä käytetään. Käyttöliittymän voidaan ajatella perustuvan vuorovaikutustapaan ja sitä kautta metaforaan, jonka perusteella vuorovaikutustapa on kehitetty. Näin ollen käyttöliittymä toteuttaa tiettyä vuorovaikutustapaa ja on siten alisteinen sen valitsemalle metaforalle (katso Kuva 7). Metafora asettaa vuorovaikutustavalle tietyt tavoitteet, jotka on koottu oheiseen listaan.

Epäsuorassa hallinnassa vuorovaikutustavan metafora on ihmisten välinen vuorovaikutus. Metaforan valinta on perusteltu vuorovaikutustavan tuttuudella. Epäsuoran viestin lähettämistä *ei opetella*, vaan idean mukaan sen noudattelee ihmisten välisestä vuorovaikutuksesta jo tuttuja mekanismeja. Epäsuora hallinta pyrkii tavoittamaan sanattomien viestien ilmaisuvoimaa. Sanatonta ja tässä mielessä epäsuoraa kommunikaatiota esiintyy ihmisten välisessä kanssakäymisessä paljonkin.

Epäsuora viestintä on kognitiivisesti melko *huomaamatonta* ja lähettäjälleen *vaivatonta*. Viestin lähettäjä ei aina tiedä tai hän ei tunne lähettävänsä viestiä, vaikka tosiasiassa niin tekeekin. Vaikka

tällainen viestintä onkin varsin huomaamatonta, se on kuitenkin myös tiedollisesti havaittavissa, joten ihminen kykenee sitä halutessaan tarkkailemaan. Sen vastaanottaminenkaan ei edellytä viestinnän tietoista seuraamista, joten myös viestin vastaanottaminen on niin ikään vaivatonta.

Sen lisäksi, että epäsuoraa hallintaa käyttävän käyttöliittymän on tuettava metaforansa toteutumista, sen on myös vastattava muihin vaatimuksiin, joita hyville käyttöliittymille asetetaan. Loppujen lopuksi ne ovat tavoitteita, joita myös metaforan valinta pyrkii edistämään. Niinpä myös epäsuoraa hallintaa käyttävän käyttöliittymän tulee olla johdonmukainen, tarjota relevanttia palautetta, ehkäistä virheitä ja tarjota helppoja keinoja niiden korjaamiseksi, mahdollistaa toimintojen peruminen, mahdollistaa ohjelman kontrollointi ja ehkäistä käyttäjän muistin kuormitusta [Shneiderman 98, s. 74].

4.1.2 Vuorovaikutustavan ongelmista johdetut tavoitteet

ONGELMIEN ASETTAMAT TAVOITTEET

- kontrollointi
- ennustettavuus
- luottamus
- häiritsemättömyys

Tutkijoiden keskuudessa epäsuorasta hallinnasta aiheutuvia ongelmia tunnetaan melko hyvin ja ongelmia on käsitelty myös tässä työssä aiemmin (katso luku ‘Epäsuoran hallinnan edut ja haitat’, sivu 9). Kääntäen tunnistetut ongelmat ovat tavoitteita tai tavotteiden alkuja hyvin toteutetulle epäsuoralle hallinnalle. Hyvän järjestelmän rakentaminen ottamatta kantaa jo tiedettyihin ongelmiin ei ole uskottavaa. Ongelmista johdetut tavoitteet on koottu oheiseen listaan.

Kirjallisuudessa on havaittavissa suuntaus, jossa epäsuoraa hallintaa käyttävien ohjelmien käyttöliittymien tärkeimmäksi ongelmaksi nostetaan *kontrolli*. Kontrolli on merkittävä ominaisuus tavallisissa käyttöliittymissä ja se on sitä myös epäsuoraa hallintaa käyttävissä käyttöliittymissä, vaikka se onkin osittain ristiriitainen tavoite vuorovaikutustavan ideoiden kanssa.

Tärkeäksi käyttöliittymän tavoitteeksi käyttäjän kontrollia ovat olleet asettamassa ainakin Pattie Maes ja Ben Shneiderman [Maes and Shneiderman 97], jotka ovat keskustelleet asiasta useammassakin yhteydessä. Myös Donald Norman [Norman 94, s. 69] nostaa kontrollin tunteen tärkeimmäksi tekijäksi uuden teknologian hyväksymisessä. Erityisesti, jos se on aiempaa automaattisempaa, mikä on epäsuoran hallinnan kohdalla tilanne.

Brenda Laurel [Laurel 90, s. 363] muotoilee asian hieman toisin sanoessaan *ennustettavuudesta* seuraavan ohjelman käyttökelpoisuuden. Myös Kristina Höök [Höök 97] nostaa esiin ennustettavuuden, kontrollin ja läpinäkyvyyden ohella, epäsuorasti hallittavien käyttöliittymien kriittiseksi ominaisuudeksi ja sitä myöden käyttöliittymän tavoitteeksi.

Vaikka kontrolli onkin hyvin keskeinen vaatimus, on kirjallisuudessa esitetty myös toisenlaisia vaatimuksia epäsuorasti hallittaville käyttöliittymille. Esiin on nostettu esimerkiksi *luottamus* [Maes 94]. Vaatimus luottamuksesta on oleellinen etenkin sellaisille ohjelmille, jotka käyttävät toteutustekniikkanaan yhteisöllisiä menetelmiä, joista johtuu tietoturvaongelmia. Käyttäjien tulee voida luottaa siihen, ettei ohjelma paljasta luottamuksellista tietoa järjestelmän ulkopuolelle tai toisille sen käyttäjille ilman käyttäjän suostumusta.

Luottamuksesta voidaan puhua myös toimintojen kohdalla. Käyttäjän tulee voida luottaa ohjelman toimintaan siinä mielessä, ettei se aiheuta vahinkoa käyttäjän järjestelmälle tai mahdollisesti muille käyttäjille. Tässä kohdin vaatimus luottamuksesta tulee lähelle vaatimusta kontrollista tai ennustettavuudesta siinä mielessä, missä Höök asiasta puhuu. Käyttäjän on voitava ymmärtää mitä ohjelma tekee, millä perusteella ja mikä on sen tila [Höök 97].

Erityisen ongelmallista tämä on, jos ohjelmasta toteutus korostaa ohjelman älykkyyttä tai ihmishahmoisuutta. Tällaisessa lähtökohdassa voi jo suunnittelijalle olla epäselvää mitä ohjelma itseasiassa tekee ja millä perusteella [Shneiderman 93]. Tältä pohjalta on selvää, ettei käyttäjä voi ymmärtää tai ennustaa ohjelman toimintaa riittävästi saati luottaa siihen.

Kontrolli, luotettavuus ja ennustettavuus eivät ole vieraita vaatimuksia tavallisillekaan käyttöliittymille. Tässä mielessä ne eivät olisi erityisiä myöskään epäsuorasti hallittaville käyttöliittymille. On kuitenkin nähtävissä, että näiden ominaisuuksien merkitys kasvaa epäsuorassa vuorovaikutuksessa, koska erityisesti vaatimus kontrollista on eräässä mielessä ristiriidassa epäsuoran vuorovaikutustavan kanssa, kuten jo todettiin. Epäsuoruuteen ei käsitteellisesti kuulu voimakas kontrolloitavuus. Ihmisen ja tietokoneen (tai yleisemmin koneen) vuorovaikutuksessa kontrollin merkitys on kuitenkin kiistaton. Näiden kahden ristiriitaisen tavoitteen takia mainitut vaatimukset ovat erityisiä epäsuoraa hallintaa tukeville käyttöliittymille.

Kontrollin ja luotettavuuden ohella epäsuorasti hallitun ohjelman tulee olla käyttäjän ensisijaista toimintoa *häiritsemätön*. Koska epäsuorasti hallittaviksi sopivat vain vähän tarkkuutta vaativat eikriittiset toiminnot, on käyttäjän ensisijainen toiminto tyypillisesti jokin muu, kuin mitä epäsuorasti hallittu ohjelma tukee. Käyttäjän on kyettävä hyödyntämään epäsuorasti tuotettua tietoa työssään, mutta hänen on myös kyettävä tehokkaasti keskittymään ensisijaiseen toimintoonsa. Jatkuva käyttäjän keskittymisen katkaisu ei palvele ohjelman tavoitetta tukea käyttäjää tämän työssä.

4.1.3 Yleiset vaatimukset

Häiritsemättömyyden ja kontrollin tavoitteet ovat eräässä mielessä keskenään ristiriitaisia. Mitä enemmän käyttäjä on ohjelman toiminnasta tietoinen, sitä paremmin hän todennäköisesti kykenee sitä kontrollimaan. Samalla tietoisuus ohjelman toiminnoista voi olla häiritsevää. Tästä ristiriidasta monet kontrollointiongelmat ovat peräisin. Kun ohjelmasta on pyritty tekemään mahdollisimman häiritsemätön, jopa tieto sen olemassaolosta on käynyt epäselväksi.

Eri yhteyksissä kontrolli voidaan mahdollistaa eri tavoin. Esimerkiksi Ben Shneidemanin näkemyksen mukaan kontrolli mahdollistetaan parhaiten esittämällä ohjelman toiminta käyttäjälle visuaalisessa ja interaktiivisessa muodossa. Kun käyttäjä voi nähdä toimintojensa seuraukset samaan aikaan komentojen antamisen kanssa, hän kykenee ymmärtämään komennon ja sen tuloksen välisen suhteen. Tämä tapa ei kuitenkaan sovi kaikkiin tapauksiin, erityisesti epäsuorasti hallittavassa käyttöliittymässä tällainen lähestymistapa ei tue vuorovaikutustavan ideaa.

Häiritsemättömyyden ja kontrolloinnin pohdinnasta voidaan tehdä johdettuja vaatimuksia. Käyttöliittymän tulee tarjota *tietoa ohjelman toiminnasta ja sen olemassaolosta*. Suoravaikutteisessa käyttöliittymässä on näkyvissä mahdolliset kontrollointitavat, toimintojen vaikutus nähdään toimintojen suorituksen aikana ja samalla saadaan tietoa myös ohjelman toiminnasta syvemminkin. Käyttäjä tietää aina milloin toiminto suoritetaan, koska hän itse on vastuussa toiminnon käynnistämisestä. Eikä ohjelman olemassaolon havaitseminen ole mikään ongelma tavallisissa käyttöliittymissä.

Erilaisen tiedon tarjonta on siis tärkeässä asemassa, kun tehdään ymmärrettävää käyttöliittymää. Kysymys kuuluukin: millaista tietoa käyttäjälle on tarjottava epäsuoran hallinnan toiminnasta, jotta hän kykenee sitä ymmärtämään? Ratkaisujen toisessa äärilaidassa ovat ohjelmat, jotka eivät kerro käyttäjälle mitään toimintatavoistaan, vaan tarjoavat ainoastaan tulokset käyttäjälle. Tällaista tekniikka käyttää Letizia-agentti [Lieberman 97], joka etsii ja ehdottaa käyttäjää mahdollisesti kiinnostavia WWW-sivuja katsottavaksi.

Toisessa ääripäässä ohjelman tulisi kertoa pikkutarkasti kaikki suoritettavat toiminnot ja niihin vaikuttavat muuttujat. Tällainen skenaario ei ole realistinen tai edes toimiva, se hukuttaisi käyttäjän turhaan tietotulvaan. Liiallisen tietomäärän syytämisen välttäminen käyttäjälle onkin eräs vaatimus epäsuorasti hallitulle käyttöliittymälle.

Kaikkien ohjelmien kohdalla pitää paikkaansa, että niiden tuottamat tulokset on oltava helposti hyödynnettävissä, sillä muuten koko ohjelman hyödyllisyys on kyseenalainen. Epäsuorasti hallitun toiminnon kohdalla tämä pitää erityisesti paikkaansa, sillä usein käyttäjä on toiminnon kanssa tekemisissä ainoastaan sen tuottamien tulosten kautta. Siksi *tulosten esitysmuotoon ja hyödyntämiseen* on kiinnitettävä erityistä huomiota.

Koska epäsuorasti hallittujen ohjelmien kanssa käyttäjä ei ole jatkuvassa vuorovaikutuksessa, syntyy tarve muuttaa niiden toimintatapoja pidemmällä aikavälillä. Tavallisissa ohjelmissa tai toiminnoissa *konfigurointi* ei ole yhtä merkittävässä asemassa, koska käyttäjä suorittaa tyypillisesti itse kaikki toiminnot – haluamallaan tavalla. Myös *kontrollin mahdollistamiseen* on kiinnitettävä huomiota, kuten jo edellisessä alakohdassa mainittiin.

Oheinen lista esittää vielä yhteenvetona kaikki edellä tunnistetut epäsuoraan hallintaan perustuvalle käyttöliittymälle asetettavat vaatimukset.

YLEISET VAATIMUKSET

- tieto olemassaolosta
- riittävän tiedon tarjoaminen
- tulokset helposti hyödynnettävissä
- konfigurointi ja kontrollointi

4.2 Käyttöliittymän osat

Epäsuoran hallinnan asettamia käyttöliittymävaatimuksia voidaan analysoida tarkemmin jakamalla käyttöliittymä toiminnallisiin osiin. Jokaista osaa voidaan näin tarkastella epäsuoran hallinnan kannalta ja pohtia, millaisia erityisiä vaatimuksia se käyttöliittymälle asettaa.

Epäsuoraan hallintaan perustuvat käyttöliittymät tai niiden tehtävät on jaettu yleisiä vaatimuksia mukaillen neljään osaan: 1) palaute, 2) toimintojen konfigurointi, 3) toimintojen kontrollointi ja 4) tulosten esittäminen. Kolmella viimeksi mainitulla osalla on selvä

yhteys edellä kuvattuihin vaatimuksiin, mutta palaute näyttää vieraammalta käsitteeltä. Se viittaa kahteen vaatimukseen: tieto olemassaolosta ja riittävän tiedon tarjonta. Käytettäväksi on valittu käsite palaute, koska se kattaa molemmat vaatimukset ja yleisluontoisempana kykenee ottamaan kantaa myös esimerkiksi tulosten esittämiseen.

Osat käsitellään seuraavassa yksitellen pohtien millaisia vaatimuksia kullekin osalle on asetettava. Vaatimusten pohdinta perustuu yleisiin vaatimuksiin ja tarkentaa niissä mainittuja vaatimuksia.

4.2.1 Palaute

VAATIMUKSET PALAUTTEELLE

- toimintaperiaatteiden selventäminen
- häiritsemätön
- käyttäjän keskittymistä kunnioittava

Palautteen antaminen ohjelman toiminnasta on epäsuorasti hallittavissa ohjelmissa vähintään yhtä merkittävää kuin tavallisissa käyttöliittymissä. Palautteen perusteella käyttäjällä on mahdollisuus *ymmärtää ohjelman toimintaperiaatetta* muodostamalla ja korjaamalla mielikuvamallia ohjelman toiminnasta. Mallin perusteella ohjelman toiminnan ennustaminen tulee mahdolliseksi ja kontrollointi helpottuu [Mayhew 92, s. 93]. Yhteenveto palautteelle asetettavista vaatimuksista on nähtävissä oheisessa kuviossa.

Epäsuoran hallinnan toteuttavien ohjelman osien antaman palautteen muodolle on asetettava erityisiä ehtoja. Tämän perustelemiseksi on muistettava, että tällaiset ohjelman osat toimivat itsenäisesti, käyttäjän toimien rinnalla. Niinpä *palaute ei saa olla suurieleistä*, mikä voisi häiritä käyttäjän keskittymistä. Häiritsemättömyyshän kirjattiin edellä erääksi käyttöliittymän vaatimukseksi. Tämä on tilanne erityisesti, jos epäsuorasti hallittu toiminto tuottaa tuloksia yhtäaikaisesti käyttäjän muiden (ensisijaisten) tehtävien kanssa.

Ihmisen silmä havaitsee hyvin herkästi liikettä, mikä johtuu ihmisen visuaalisen havaintomekanismin kahdesta ominaisuudesta. Ensinnäkin havainto liikkeestä kyetään tekemään nopeammin kuin mitä kestää muun visuaalisen tiedon (esim. värin) havaitseminen samasta kohteesta [Card *et al.* 83]. Tämän lisäksi silmä kykenee havaitsemaan liikettä laajemmalla alueella näkökentässä kuin muuta tietoa, koska näkökentän laitojen havainnoinnista vastaavat sauvasolut havaitsevat liikettä (muutosta) herkästi [Gillan 98].

Tästä johtuen yleisesti käytetyt palautemekanismit voivat olla soveltumattomia käytettäviksi epäsuoraa hallintaa toteuttavissa ohjelmissa. Esimerkiksi visuaalisessa palautteessa liikkeen käyttö, jollaiseksi voidaan lukea uuden objektin tuominen näkyviin, objektin korostus värillä jne., voi olla liian häiritsevä.

Toinen mahdollisuus on käyttää – yleisesti vähemmän käytettyä – auditiivista palautetta. Kuuloon perustuvasta palautteesta voidaan jossain tapauksissa tehdä vähemmän häiritsevää kuin näköön perustuvasta. Tavallisessa kanssakäymisessä kuulo- ja näköaisteihin perustuvia viestejä vastaanotetaan lähes aina samanaikaisesti. Ihmisen havaitsemista tutkittaessa on muodostettu käsitys, jonka mukaan tällaiset usein päällekkäisinä esiintyvät ärsykkeet eivät häiritse toistensa prosessointia [Broadbend 87] eivätkä siten myöskään keskittymistä toiseen.

Näköaistin voidaan ajatella toimivan myös keskittymisen rakentajana, mutta kuulolla ei tunnu olevan yhtä voimakasta tällaista tehtävää. Kun ihminen keskittyy tehtävään, hänen katseensa tyypillisesti on mukana tehtävässä, mutta kuulon avulla voidaan olla yhteydessä muuhun ympäristöön. Tämä pitää erityisesti paikkansa

työskenneltäessä tietokoneohjelmien kanssa, koska ne tyypillisesti viestivät pääasiassa näköaistiin perustuen. Kuulon avulla tullutta tietoa osataan suodattaa niin, että ainoastaan hyvin voimakkaat ärsykkeet itse asiassa katkaisevat tehtävään keskittymisen. Silti kuulon perusteella ympäristöstä saadaan merkittävä määrä tietoa.

Kuuloaistin perusteella ei voida kuitenkaan antaa häiritsemättä suurta määrää yksityiskohtaista tietoa. Lähinnä tieto on luonteeltaan erilaisista tapahtumista kertomista. Näin ollen suuremman tietomäärän välittämiseen tulee joka tapauksessa käyttää visuaalista kanavaa.

4.2.2 Toimintojen kontrollointi

VAATIMUKSET KONTROLLILLE

- nopea ja helppo käyttö
- mahtuu pieneen tilaan

Kontrolloinnilla tarkoitetaan ohjelman suorituksen välitöntä muuttamista. Se voi tarkoittaa ohjelman kytkemistä toimintaan tai tietyn toiminnon suorittamista. Kontrolloinnin luonteeseen kuuluu, että sen vaikutukset ovat käyttäjän havaittavissa välittömästi. Kontrollille asetetut vaatimukset on koottu oheiseen kuvioon.

Vaikka epäsuoraa hallintaa toteuttavista toiminnoista voitaisiin johtaa suuri joukko käyttäjälle potentiaalisesti relevantteja kontrollointikohteita, ei kaikkia voida laittaa käyttöliittymään. Koska toiminnot ovat luonteeltaan toissijaisia, niiden hallintalaitteiden on *mahduttava pieneen tilaan*. Epäsuoran vuorovaikutustavan yleisten tavoitteiden mukaisesti on vältettävä myös käyttäjän kognitiivisen kuorman kasvattamista, joten hallintalaitteiden käytön tulee korostetusti olla *helppoa ja nopeaa*.

Erityisesti pinta-alan säästön kannalta kontrollin toteutuksessa joudutaan olemaan hyvin vähäeleisiä. On hyvin tarkkaan punnittava, mitkä toiminnot ovat kontrollin tunteen saavuttamiseksi tärkeimpiä. Esimerkiksi koko toiminnon keskeyttäminen ja uudelleen käynnistäminen ovat tällöin ehkä merkittävimpiä tapoja kontrolloida ohjelman toimintaa kuin kaikkien yksityiskohtien hallinnan mahdollistaminen.

Toisaalta on varottava myös liian vähäistä kontrolloinnin mahdollisuutta. Kuten aiemmin todettiin, epäsuoran hallinnan ja ihmisen ja tietokoneen vuorovaikutuksen ristiriitaisista tavoitteista johtuen on kontrollin merkitys suuri tällaisten käyttöliittymien hyväksynnässä. Millainen kontrollin taso sitten riittää tähän, jää nähtäväksi. Siitä viimeisen sanan sanovat käyttäjät oikeassa käyttötilanteessa.

4.2.3 Toimintojen konfigurointi

VAATIMUKSET KONFIGUROINNILLE

- nopeasti opittava
- helposti muistettava
- antaa palautetta
- mahdollistaa kontrollin

Konfigurointi on ohjelman suoritusta ohjaavien parametrien muokkaamista, jolla pyritään ohjelman mukauttamiseen haluttuun toimintaympäristöön. Konfiguroinnin vaikutukset ovat kontrollointiin verrattuna pitkäaikaisempia. Epäsuoran hallinnan toteuttavissa ohjelmissa on tyypillisesti runsaasti toimintaan vaikuttavia parametrejä, joiden arvojen määritteleminen etukäteen ei anna optimaalista tulosta. Siksi käyttäjälähtöinen konfigurointi on tarpeen. Asetettavat tavoitteet on esitetty kiteytetysti oheisessa kuviossa.

Tavallisen käyttöliittymän kohdalla toiminnan konfigurointi ei ole kovin keskeisessä asemassa ohjelman hallinnassa. Käyttöliittymät toimivat tyypillisesti aina samoin, joskin käyttäjällä on usein mahdollisuus muokata käyttöliittymän ulkoasua, komentojen nimiä ja paikkoja yms. haluamallaan tavalla.

Konfigurointi on epäsuorasti hallitussa ohjelmassa keskeisemmässä asemassa siksi, että se mahdollistaa itsenäisesti toimivan ohjelman toimintaperiaatteiden hallitsemisen. Käytännössä se tarkoittaa eri toteutustekniikoiden suoritusta ohjaavien parametrien muuttamista. Parametrien avulla käyttäjän on mahdollista säädellä esimerkiksi käytettyjä heuristisia sääntöjä omaan toimintatapaansa paremmin sopiviksi.

Ohjelman konfigurointi on ohjelman käytön suhteen toissijainen toiminto, joten sitä tehdään suhteellisen harvoin. Harvinaisena toimintona konfigurointikäyttöliittymä ei ole tyypillisessä tilanteessa näkyvissä. Tämä on konfiguroinnin toteuttavalle käyttöliittymälle sekä haaste että mahdollisuus.

Haasteelliseksi tilanteen tekee toimintojen vähäinen käyttö. Käyttöliittymän tulee olla siis *nopeasti opittava* ja *helposti muistettava*, sillä harvoin tarvittujen toimintojen opetteluun ei ole mukava käyttää paljoa aikaa. Käyttäjä haluaa tyypillisesti suoriutua konfiguroinnista nopeasti keskittyäkseen uudelleen ensisijaiseen tehtäväänsä.

Koska konfiguraation mahdollistava käyttöliittymä on käytössä vain hyvin rajallisen ajan, se voi käyttää kuvaruudusta suuren osan toimintojensa esittämiseen. Tämä voi helpottaa havainnollisten esitystapojen hyödyntämistä käyttöliittymän suunnittelussa, kun esimerkiksi näytettäviä toimintoja ei tarvitse karsia tilan puutteen vuoksi.

Eri toimintojen konfigurointimahdollisuus kertoo käyttäjälle niiden olemassaolosta ja niiden luonteesta. Tällä voi olla suuri merkitys ohjelman toiminnallisuuden ymmärtämisessä. Käytetyt tekniikat ovat kuitenkin käyttäjälle vieraita, mikä tekee niiden konfigurointikäyttöliittymien suunnittelusta hyvin haastavaa: vieraat käsitteet on onnistuttava esittämään käyttäjälle hänen kielellään ymmärrettävästi.

Runsaan näyttötilan antama mahdollisuus on syytä käyttää hyväksi myös *palautteen antamisessa* ja *konfiguroinnin mahdollistamisessa*. Esimerkiksi palautetta voidaan konfiguroinnin yhteydessä antaa monimuotoisesti myös visuaalisessa muodossa ilman, että on vaarana häiritä käyttäjän keskittymistä.

4.2.4 Tulokset

VAATIMUKSET TULOKSILLE

- häiritsemättömyys
- tulosten merkityksen arvioinnin helppous
- tulosten hyödyntämisen helppous

Epäsuorasti hallitun ohjelman varsinainen hyöty on tietenkin sen tuottamissa tuloksissa. Tulokset on aina jossain vaiheessa esitettävä käyttäjälle, muussa tapauksessa ohjelmasta ei liene mitään hyötyä. Minimaalista sovellusta, millaisia epäsuorasti hallitut toiminnot yleensä ovat, rakennettaessa tämä pakollinen interaktio käyttäjän kanssa on syytä käyttää tarkasti hyödyksi ja sen vaatimukset mietittävä tarkoin. Tärkeimmät vaatimukset tulosten esittämiselle on koottu oheiseen kuvaajaan.

Tulokset voidaan nähdä eräänä palautteen muotona, sillä ne antavat karkeaa tietoa sovelluksen tilasta ja sen toimintaperiaatteista. Molemmissa tapauksissa palaute on melko epämääräistä, sillä esimerkiksi toimintaperiaatteiden päättely vain tulosten perusteella on vaikeaa. Palautteena tuloksia ajatellen, niihin pätevät samat vaatimukset, joita edellä esitettiin palautteen antamisesta yleensä, erityisesti *häiritsemättömyys*.

Erityisen lähelle palautteen antamista tulosten esittäminen tulee silloin, jos epäsuorasti hallitun toiminnan tarkoituksena on suorittaa järjestelmän tilaan vaikuttavia toimia käyttäjän puolesta. Esimerkiksi sähköpostin automaattinen lajittelu voitaisiin ajatella tällaiseksi toiminnoksi. Ohjelman tulos ei ole mikään konkreettinen objekti (tieto- tai muu sellainen) vaan järjestelmän uusi tila. Tilan muutoksesta käyttäjä saa palautetta tuloksen esittämisen yhteydessä.

Toinen suuri tulosryhmä tilamuutosten rinnalla on uusien objektien tuottaminen. Tuotettu objekti on tyypillisesti joko tietovarastoista etsittyä tietoa tai epäsuoran syötteen perusteella muodostettua tietoa. Ensin mainitusta esimerkkinä ovat erilaisten hakuagenttien tulokset, kun taas esimerkkiperustainen ohjelmointi tuottaa tieto-objekteja eli tässä tapauksessa ohjelmia.

Tuloksen *hyödyllisyyden arviointi* on asia, joka on erityinen epäsuorasti hallittujen ohjelmien tuloksille. Kun käyttäjä itse luo tai etsii tieto-objektin, hän todennäköisesti tietää, mihin sitä voi käyttää. Hyödyllisyyden arviointia tarvitaan, kun käytetään esimerkiksi Internetin hakupalveluja. Käyttäjä joutuu suorittamaan karsintaa löytääkseen hakutuloksista oleellisen tiedon. Kun jo itse haku on suoritettu käyttäjän epäsuoran hallinnan alaisuudessa, tulosobjektien merkitys voi olla vielä vaikeammin ymmärrettävissä. Tästä syystä on tarpeen varustaa tulos tiedoilla, joiden perusteella käyttäjän on mahdollista *arvioida tuloksen käyttötarkoitusta ja hyödyllisyyttä*.

Sen lisäksi, että käyttäjä voi arvioida tulosten käyttökelpoisuutta, myös niiden *hyödyntäminen on oltava helppoa*. Konkreettisemmin tämä tarkoittaa sitä, että esimerkiksi generoidun ohjelman suorittaminen on yksinkertaista, eikä vaadi useita toimenpiteitä. Sama pätee vaikkapa löydettyjen tieto-objektien käsittelyssä. Niiden tulee olla nopeasti avattavissa ja katsottavissa. Tiedon muodon tulee olla yhteensopiva käyttäjän toimintaympäristön kanssa.

4.2.5 Automaattinen ja manuaalinen osa

PERUSTEET OSIEN EROTTELULLE

- rakenteen selkeyttäminen
- toimintojen vaikutusalan ymmärtäminen

Eräs mahdollisuus jaotella käyttöliittymän osia ohjelmissa, jotka käyttävät epäsuoraa hallintaa vuorovaikutustapana, on jakaa ohjelma manuaalisiin ja automaattisiin toimintoihin [Höök 97]. Jaossa automaattisiin toimintoihin kuuluvat kaikki epäsuorasti hallittavat toiminnot. Manuaalisia toimintoja ovat ohjelman muut toiminnot, joita hallitaan joillain perinteisillä vuorovaikutustavoilla. Näiden eri osien erotteleminen vastaa käyttöliittymän vaatimuksiin *selventää ohjelman rakennetta* käyttäjälle sekä *selvittää toimintojen vaikutusalaa* ohjelmassa. Perusteet on koottu myös oheiseen kuvioon.

Käyttöliittymän jaottelulla tällaisiin osiin voidaan ensinnäkin parantaa ohjelman kontrollin tunnetta, kuten Kristina Höök artikkelissaan [Höök 97] toteaa. Automaattinen toiminnallisuus voi tuntua käyttäjästä käsittämättömältä tai epäloogiselta varsinkin, jos ohjelma on käyttäjälle muutenkin vieras. Ohjelman keskeisten toimintojen opetteluun saattaa kulua käyttäjän kognitiivisia resursseja niin paljon, ettei automaattisen osan toimintaan ehdi kiinnittää juurikaan huomiota. Toisaalta ei ole tarkoituskaan kiinnittää huomiota liiaksi ohjelman automaattisiin osiin, niiden kun tulisi toimia ilman käyttäjän työpanosta. Ohjelman toimintojen jakaminen automaatti-

siin ja manuaalisiin helpottaa huomion suuntaamista tiettyyn asiaan kerrallaan.

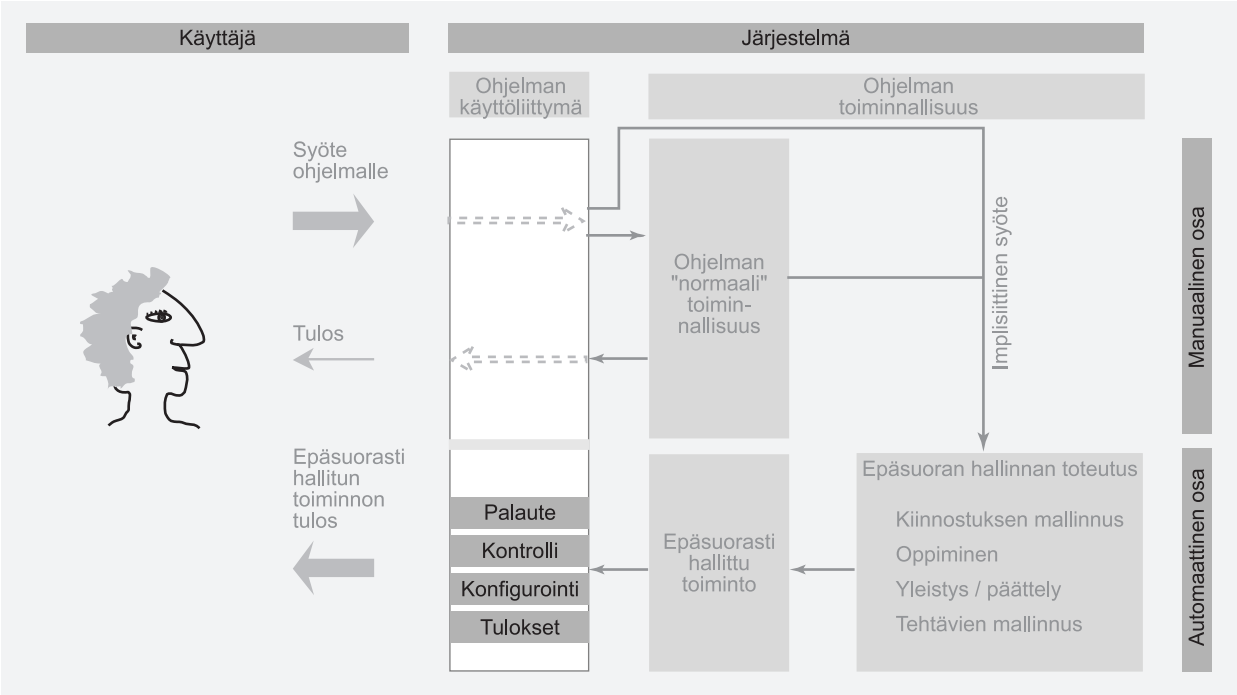
Jos käyttöliittymässä ohjelman toiminnot jaetaan automaattisiin ja manuaalisiin, se helpottaa konfiguraation aseman ja merkityksen ymmärtämistä. Konfiguroinnillahan tässä yhteydessä tarkoitetaan nimenomaan epäsuoran hallinnan toteuttavien tekniikoiden asetusten määrittelemistä. Konfigurointi liittyy siis yksinomaan automaattisiin toimintoihin, joten konfiguroinnissa määriteltyjen asetusten vaikutusalan ymmärtäminen voi olla helpompaa, jos toiminnot on jaoteltu myös käyttöliittymässä samalla perusteella.

Jotkin epäsuorasti hallitut toiminnot voivat olla luonteeltaan sellaisia, että ne soveltuvat yhtä hyvin myös manuaalisiksi toiminnoiksi. Ajatellaan vaikkapa ohjelmaa, joka etsii World Wide Webistä tietoa liittyen käyttäjän käsillä olevaan tehtävään, joka on tyypillinen sovellus epäsuoralle hallinnalle. Käyttäjän suorittaessa omaa tehtäväänsä automaattinen ohjelman osa etsii siihen liittyvää materiaalia. Sama hakutoiminto on luonteva myös manuaalisena. Käyttäjä voi halutessaan määritellä hakukriteerit ja suorittaa haun haluamallaan hetkellä. Tällaisissa tapauksissa käyttöliittymän jakaminen manuaaliseen ja automaattiseen osaan on ongelmallista.

4.2.6 Epäsuorasti hallitun toiminnon käyttöliittymän rakenne

Edellä esitetyn yhteenvetona voidaan muodostaa käsitys epäsuorasti hallitun ohjelman käyttöliittymän rakenteesta. Rakenteessa on erotettavissa kaikki toiminnalliset osat, joita edellä on tunnistettu. Kaaviokuva (Kuva 8) havainnollistaa näiden osien välisiä suhteita.

Kaavio perustuu jo aiemmin esitettyyn epäsuoraa hallintaa käyttävien ohjelmien rakennetta kuvaavaan kaavioon, josta tässä kohdassa kiinnostamaton osa on himmennetty. Lisättynä on käyttöliitty-



Kuva 8 Epäsuoraan hallintaan liittyvän käyttöliittymän osat

män osat.

Ylätasolla käyttöliittymä voidaan jakaa automaattiseen ja manuaaliseen osaan. Tällöin epäsuoraan hallintaan liittyvät käyttöliittymätoiminnot kuluvat sen automaattiseen osaan. Jaottelun voidaan ajatella jatkuvan myös käyttöliittymän taakse, itse ohjelman rakenteeseen, jolloin automaattisen käyttöliittymän kohde selventyy.

Toiseksi automaattisesta käyttöliittymäosasta voidaan erottaa edellä mainitut toiminnot: palaute, kontrolli, konfigurointi ja tulokset. Ne ovat automaattisen käyttöliittymäosan komponentteja.

4.3 Tekniikoiden havainnollistaminen

Erääksi käyttöliittymän tärkeäksi toiminnallisuudeksi oletetaan tässä tarkastelussa ohjelman toteuttavan tekniikan havainnollistus. Tällä olettamuksella etsitään edellistä tarkastelua konkreettisempia vaatimuksia ja käyttöliittymän toiminnallisuutta, joiden perusteella varsinainen käyttöliittymä kyetään tekemään.

Vaikka seuraavassa hahmotellaankin käyttöliittymävaatimuksia jokaista aiemmin tunnistettua epäsuoran hallinnan mahdollistavaa tekniikkaa kohden, ei johtopäätösten merkitys ole itsestään selvä. On erikseen pohdittava mitkä asiat ovat käyttäjän kannalta relevantteja missäkin tilanteessa. Suinkaan tilanne ei ole se, että mitä enemmän käyttäjälle annetaan tietoa, sitä paremmin hän voi ohjelman toimintaa ymmärtää. Päin vastoin, epäsuoran hallinnan ollessa kyseessä on aiheellista varoa liiallista toiminnallisuutta ja tietotulvaa. Kuten jo perustavissa vaatimuksissa todettiin: käytön tulee säilyä yksinkertaisena ja vähäeleisenä.

Tarkastelu etenee toteutustekniikoittain. Jokaisen tekniikan kohdalla pohditaan aluksi millaisia vaatimuksia tekniikan käyttö käyttöliittymälle asettaa. Kohdan lopussa on pieni pohdita siitä, millaisia kontrollointimahdollisuuksia tekniikan käyttämiseksi pitäisi tarjota. Tarkastelun tulokset on koottu oheen liitettyihin listoihin, jossa palautteelle asetettavaa vaatimusta merkitään pisteellä (•) ja kontrollivaatimusta viivalla (–).

4.3.1 Implisiittisen syötteen keräys

VAATIMUKSET IMPLISIITTISEN SYÖTTEEN KERÄYKSELLE

- toiminnon havaittavaksi tekeminen
- havaittujen toimien julkistus
- keräysajankohdan julkistus
- havaintojen käyttötarkoituksen julkistus

– lupa havainnointiin

Implisiittisen syötteen keräys eli käyttäjän toimien tarkkailu on eräässä mielessä perustava toiminto epäsuorasti hallituissa ohjelmissa ja luonteeltaan hiljaista toimintaa. Senhän paljastaa jo tarkkailu käsitteenäkin. Käyttöliittymän onkin annettava runsasta palautetta toiminnallisuudesta. Oheinen kuvio esittää käyttöliittymävaatimukset tiivistetysti.

Aivan ilmeinen vaatimus käyttöliittymää ajatellen on *tarkkailun havaittavaksi tekeminen*. Jos syötteen keräysmekanismi on käyttäjälle tuntematon, myös syötteeseen perustuvien tekniikoiden toiminnan ymmärtäminen on lähes mahdotonta.

Ensimmäinen käyttäjälle ilmaistava asia, on *mitkä toiminnot ohjelma tulkitsee implisiittiseksi syötteeksi*. Tämän tiedon perusteella käyttäjä voi saada aavistuksen siitä, milloin hänen toimintojaan tarkkaillaan ja mihin niitä käytetään. Tämä on toisaalta myös potentiaalinen ongelmakohta. Käyttäjät saattavat tehdä liian yksinkertaisia oletuksia ohjelman toimintalogiikasta, jos he tietävät ainoastaan sen,

millaisia asioita ohjelma voi havaita. Voihan olla, että ohjelman tarkkailuun liittyvä heuristiikka on huomattavan monimutkainen, eikä tämä tieto voi välittyä pelkästään tarkkailukohteen tunnistamisen perusteella.

Askel runsaampaan tiedon välittämiseen otetaan, jos käyttäjälle viestitään myös se, *milloin ohjelma kerää implisiittistä syötettä*. Tällöin käyttäjän on mahdollista huomata implisiittisen syötteen keräyksen tekemiä havaintosarjoja. Ohjelman heuristiikka voi esimerkiksi olla kiinnostunut tiettyjen kolmen toiminnon perättäisestä suorituksesta, jonka ymmärtämistä voi helpottaa ohjelman tekemien peräkkäisten havaintojen tietäminen. Aikaan sidottu palaute tukee myös ensimmäistä vaatimusta tehdä mahdolliset havaittavat asiat käyttäjälle ylipäätään näkyviksi.

Kolmanneksi käyttöliittymä voi tarjota tietoa käyttäjälle siitä, *mitä se havainnoilla tekee*. Tässä kohdassa on syytä erottaa implisiittisen syötteen keräys erilaisista muista toiminnoista, jotka tätä syötettä käyttävät. Implisiittisen syötteen keräysvaiheessa yksittäisiä havaintoja voidaan koostaa suuremmiksi kokonaisuuksiksi tai havaintoja voidaan tallentaa. Kummassakin tapauksessa käyttäjän toimia tarkkailevan komponentin tavoitteena on nostaa syötteen abstraktiotasoa. Näin kerättyä ja jalostettua syötettä voidaan sitten tarjota eri komponenteille.

Käyttäjän kannalta oleellista on pysyä ymmärtämään, millaisia heuristiikkoja ohjelma käyttää syötteen abstraktiotason nostamiseen. Tämän tiedon perusteella käyttäjän olisi mahdollista käsittää millaisia merkityksiä toimintoihin liitetään. Toisaalta tämän vaatimuksen mielekkyyttä voidaan myös ankarasti epäillä. Jos käyttäjän katsotaan tarvitsevan runsaasti tietoa siitä, miten hän ohjelmalle antaa implisiittistä syötettä, syötteenantomekanismin voidaan katsoa olevan epäonnistunut. Tiedon tarve viittaa nimittäin siihen, että käyttäjän on tietoisesti annettava tietynlaista syötettä, jotta ohjelma tukisi hänen tavoitteitaan. Tämä on täydellisesti vastoin epäsuoran hallinnan ideaa.

Käyttäjän toimien tarkkailuun liittyy myös tarve ottaa kantaa yksityisyyttä koskeviin kysymyksiin. Tarkkailu voi rikkoa käyttäjän yksityisyyttä tai tunnetta siitä. Siksi käyttäjällä tulisi olla mahdollisuus määritellä *saako järjestelmä havainnoida* hänen tekemisiään vai ei. Hienosyisemmin kontrollissa voidaan mahdollistaa myös se, *mitä yksittäisiä toimintoja ohjelma saa tarkkailla.* Tämä osaltaan lisää käyttäjän mahdollisuutta ohjelman kontrollointiin.

4.3.2 Käyttäjäprofiilin talletus

VAATIMUKSET KÄYTTÄJÄPROFIILIN TALLETUKSELLE

- sisällön paljastaminen
- päivitysperiaatteen julkistus
- julkistettavien tietojen hallinta

Käyttäjäprofiiliin tallennetaan sekä lyhyen että pitkän aikavälin tietoja käyttäjästä ja hänen mieltymyksistään. Käyttäjäprofiili on tyypillisesti itsenäinen ohjelmakomponentti, joka on vastuussa tiedon ylläpidosta ja tallentamisesta. Käyttäjäprofiili on siis luonteeltaan tietovarasto. Koska varaston tiedot koskevat käyttäjää itseään, hän saattaa olla kiinnostunut sen sisällöstä. Oheinen lista tiivistää käyttäjäprofiilin talletukselle asetetut vaatimukset.

Käyttäjäprofiilin luonteen perusteella eräs ilmeinen käyttöliittymävaatimus on sen *sisällön paljastaminen käyttäjälle*. Sen esittämisen käyttäjälle on katsottu edistävän järjestelmän kontrolloitavuutta [Cook and Kay 94]. Koska käyttäjäprofiilia usein rakennetaan implisiittisellä syötteellä, profiilin perusteella voi olla myös mah-

dollista nähdä, millaisia asioita järjestelmä kykenee havaitsemaan. Käyttäjäprofiilin perusteella on lisäksi mahdollista ymmärtää tarkemmin ohjelman toimintaperiaatteita, kun nähdään konkreettisia arvoja, joihin ohjelman toiminta annetulla hetkellä perustuu.

Jo käyttäjäprofiilin sisältämistä tiedoista voi olla osittain mahdollista päätellä, *millaisella periaatteella käyttäjäprofiilia päivitetään*. Tämä ei kuitenkaan aina pidä paikkaansa. Käyttäjäprofiilia voidaan päivittää monimutkaisen algoritmin perustella, jota käyttäjän voi olla mahdotonta tuntea. Tällaisia tapauksia varten voi olla aiheellista tarjota suorempaa tietoa siitä, millaiset toiminnot aiheuttavat käyttäjäprofiilin muuttamisen ja miten. Tiedon perusteella käyttäjällä on mahdollisuus ymmärtää hieman monimutkaisempaakin päivitysmekanismia.

Käyttäjäprofiilin kohdalla yksityisyyden valvonta on vielä oleellisempaa kuin käyttäjän toimien tarkkailussa. Käyttäjäprofiili sisältää konkreettisempaa ja vähemmän yhteydestä riippuvaa tietoa käyttäjästä jolloin myös väärinkäyttö voi olla helppoa. Erityisesti yhdessä yhteisöllisten tekniikoiden kanssa nousee kysymys siitä, *mitä käyttäjäprofiilin tietoja voidaan julkistaa* ja mitä ei. Tämän tiedon on oltava myös käyttäjän saatavilla.

4.3.3 Oppiminen

VAATIMUKSET OPPIMISELLE

- muuttuvien toimintojen havainnollistus
- muutoksen ajankohdan julkistus
- muutoksen perusteiden julkistus
- – muutoksesta päättäminen
- – muutostiheyden säätäminen

Toiset epäsuoraa hallintaa käyttävät ohjelmat pyrkivät tehostamaan toimintaansa havainnoimalla ympäristöään ja muuttamalla toimintaansa sen perusteella. Ohjelmia kutsutaan oppiviksi. Käyttäjälle ohjelman oppimislogiikka ei ole välttämättä selvä ja sen muuttunut toiminnallisuus voi aiheuttaa hämmennystä. Hämmennystä ehkäiseville käyttöliittymille asetettavat vaatimukset on koottu oheiseen kuvioon.

Ensimmäinen asia, jota käyttöliittymä voi oppivan ohjelman kohdalla viestiä, on *mitkä ohjelman toiminnot muuttuvat* oppimisen myötä. Jos käyttäjä tietää toiminnon olevan muuttuva, hänen mielikuvansa siitä ja ohjelman yleisestä toiminnasta ei kärsi, kun muutos tapahtuu. Muuttuvien ominaisuuksien tunnistamisen mahdollistaa vaikkapa niiden kategorisointi omaksi joukokseen. Tällaisen joukon muodostaminen voi auttaa muun ohjelman ymmärtämistä etenkin, jos muuntuvat osat tuntuvat toimivan ennustamattomalla tavalla.

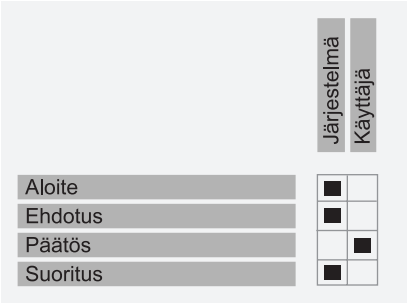
Käyttöliittymille asetettavia vaatimuksia ajatellen on tarpeen jaotella muuttuvat toiminnot tarkemmin kuin ainoastaan muuttuviin ja staattisiin. Tähän mennessä rakennetuista prototyypeistä voidaan erottaa ainakin kolmenlaisia muutoksia, joita ohjelmiin on epäsuoran hallinnan avulla toteutettu: 1) havaittavaan käyttöliittymään, 2) käyttöliittymän toimintoihin ja 3) ohjelman toiminta-algoritmiin kohdistuvat muutokset. Esimerkiksi graafista käyttöliittymää voidaan muuttaa vaihtamalla komentojen paikkoja valikoissa niiden käyttötiheyden perusteella [Sears and Shneiderman 94]. Uusia (makro)komentoja voidaan muodostaa käyttäjän suorittamien komentojen perusteella [Lieberman 93]. Useissa epäsuorasti hallituissa hakuohjelmissa hakualgoritmia tehostetaan oppimistekniikoilla [esim. Moukas 96].

Mitä suurempi vaikutus muutoksella on ohjelman havaittavaan käyttöliittymään, sitä huolellisemmin se tulee toteuttaa, sillä tällaiset muutokset rikkovat pysyvyyttä vaativaa käyttöliittymien suun-

nittelusääntöä vastaan [Nielsen 93, s. 134]. Toisaalta ohjelman toiminta-algoritmiin tehtävät muutokset ovat melko ongelmattomia, jos vain algoritmin tavoitteet pysyvät samoina. Käyttäjä huomaa tällöin muutoksen vain parantuneina (tai huonontuneina) tuloksina, ohjelman käyttötapaan tai -tarkoitukseen sillä ei ole vaikutusta.

Vaikka käyttäjän on mahdollista havaita ohjelman erilainen toiminta muutoksen jälkeen, voisi tarkemmasta tiedosta *muutoksen ajankohdasta* olla hyötyä. Sen perusteella muutos ei tulisi käyttäjälle yllätyksenä, kun hän ensimmäisen kerran muutoksen jälkeen havaitsee sen seuraukset. Toisaalta annettaessa tietoa muutoksen ajankohdasta voidaan myös ottaa kantaa siihen, kuka muutoksesta on vastuussa – ohjelma vai käyttäjä.

Ohjelman toimintojen muutosprosessi voidaan jakaa hienovaraisemmin useaan tehtävään, jotka ovat: 1) aloitteen tekeminen, 2) muutosehdotuksen laadinta, 3) muutoksen toteuttamisesta päättäminen ja 3) muutoksen toteuttaminen [Dieterich *et al.* 93]. Kunkin tehtävän voi suorittaa joko käyttäjä tai järjestelmä, kuten oheisesta kuvastakin käy ilmi (Kuva 9). Puhuttaessa epäsuorasta hallinnasta mielenkiinto kohdistuu sellaisiin tapauksiin, joissa järjestelmällä on hyvin ratkaiseva rooli muutosprosessissa. Esimerkiksi toiminnallisuus, jossa käyttäjän tekee aloitteen, ei ole tässä kohdin kiinnostava.



Kuva 9 Käyttöliittymän mukauttamisen toimijat ja tehtävät

Suurin mielenkiinto käyttöliittymän kannalta kohdistuu *muutoksen toteuttamisesta päättämiseen*, eli oheisessa kuviossa (Kuva 9) päätöstä kuvaavaan riviin. Jos toiminto annetaan käyttäjän suoritettavaksi (kuten kuvassa), on se tilaisuus antaa käyttäjälle sekä tietoa muutoksesta ja sen perusteista että antaa hänelle täysi kontrolli muutoksen suorittamisesta. Jos päätöksen tekee järjestelmä, on käyttöliittymä käyttäjän kannalta vähemmän huomiota ja työtä vaativa.

Sopiva toimintatapa riippuukin tehtävästä muutoksesta. Jos muutos koskee näkyvää käyttöliittymää, on edullista pitää käyttäjää päätöksestä vastuullisena [Sukaviriya and Foley 93]. Tällöin suuret toimintojen muutokset eivät tule yllätyksenä käyttäjälle, eikä käyttöliittymän pysyvyyttä vaativaa suunnitteluperiaatetta rikota räikeästi. Jos muutos taas koskee esimerkiksi ohjelman käyttämää hakualgoritmia, voi muutoksesta kertominen olla turhaa käyttäjän rasittamista. Tällaisten muutosten seuraukset ovat välillisiä, eivätkä siten todennäköisesti aiheuta suuria ongelmia.

Jotta käyttäjän olisi mahdollista ymmärtää ohjelman toiminnassa tehtävän muutoksen merkitys ja toisaalta niitä periaatteita, joiden perusteella muutoksesta päätetään, on hänelle *kerrottava muutoksen perusteet*. Tällainen toiminnallisuus on liitetty mm. UIDE suunnitteluympäristöllä luotuihin ohjelmiin [Sukaviriya and Foley 93].

Päätöksen teon ja muutoksen perustelujen kertomisen ohella kolmas tapa antaa kontrollia ohjelman toimintojen muuttumisesta käyttäjälle on *mahdollistaa muutostiheyden säätäminen* [Sukaviriya and Foley 93, s. 214]. Käyttäjä voisi siis asettaa esimerkiksi pienimmän ajan jonka tulee kulua muutoksesta, ennen kuin uutta voidaan ehdottaa tai suorittaa. Tämä rajoittaa suurien muutoksien vaatimat totuttelujaksot haluttuun määrään.

4.3.4 Yhteisölliset ominaisuudet

VAATIMUKSET YHTEISÖLLISILLE TEKNIIKOILLE

- käyttöajan ja kohteen julkistaminen
- käytetyn yhteisön havainnollistus
- yhteisöllisen tiedon laadun julkistus
- lupa tekniikan käyttöön
- käytetyn yhteisön hallinta

Yhteisöllisillä toiminnoilla saatetaan tehostaa ohjelmissa käytettyjä koneoppimistekniikoita tai niitä voidaan soveltaa myös suoraan, esimerkiksi käyttäjää kiinnostavan tiedon hankintaan. Yhteisölliset tekniikat perustuvat ohjelman eri instanssien väliseen kommunikaatioon. Usein ohjelmat sisältävät käyttäjäprofiilin tai oppimiskomponentin, jotka ovat yhteisöllisen tiedon lähteitä. Vaikka inhimillisessä toiminnassa yhteisölliset tavat (keskustelut, palaverit, jne.) ovat monesti selvästi nähtävillä, tietotekniikan keinoin ne voidaan toteuttaa täysin näkymättömästi. Oheinen kuvio esittää kootusti vaatimukset, jotka käyttöliittymälle asetetaan.

Yhteisöllisiin tekniikoihin liittyen käyttäjälle oleellisia asioita ovat esimerkiksi *milloin ohjelma tekniikkaa käyttää* ja *mihin tarkoitukseen*. Näiden tietojen perusteella käyttäjä pystyy ennakoimaan mahdollisia tuloksia ja toisaalta myös käsittämään tekniikan merkitystä. Implisiittisesti tämä tarkoittaa myös sitä, että käyttäjän on tiedettävä tällaisen tekniikan olemassaolosta.

Koska yhteisöllisissä menetelmissä oleellista on tiedon vaihtaminen tai kerääminen tietyn yhteisön sisällä, on oleellista tietää *mikä tämä yhteisö on*. Etenkin jos yhteisö voidaan esittää käyttäjäryhmänä, on sen perusteella mahdollista ymmärtää millaisia päämääriä yhteisöllisesti voidaan saavuttaa. Toisaalta näin voidaan myös paremmin arvioida sitä, millaista tietoa yhteisöllisen tekniikan käyttöön voidaan antaa tietoturvallisuuden siitä kärsimättä.

Jos käyttäjä kykenee arvioimaan millaista tietoa hänen mielestään yhteisöllinen tekniikka voi käyttää, on turhauttavaa, jos prosessissa aktuaalisesti käytettyä tietoa ei pystytä selvittämään. Käyttäjälle olisi siis tarjottava mahdollisuus tutkia ja kontrolloida, *millaista tietoa yhteisöllisesti käytetään*. Erityisesti tämä koskee tietoja, jotka ovat henkilökohtaisia, kuten käyttäjäprofiiliin kerätty informaatio. Käytetyn tiedon paljastamisella on suuri merkitys järjestelmän luottamuksen nostamisessa, mutta sen perusteella on mahdollista myös ymmärtää mihin sitä käytetään. Tämä edelleen auttaa ymmärtämään yhteisöllistä tekniikkaa käyttävän ohjelman osan merkitystä.

Kun puhutaan yhteisöllisten ominaisuuksien kontrolloitavuudesta, nousee päällimmäisiksi kolme asiaa: käytetäänkö tekniikkaa ylipäätään, mitä tietoa se voi käyttää ja millaisessa yhteisössä se saa toimia. Näistä ensimmäinen on yksinkertaisin. Mahdollistamalla myös käytetyn tiedon kontrolloinnin, päästään huomattavasti hienovaraisempaan kontrolliin, mikä voi joissain tapauksissa olla haluttavaa. On kuitenkin muistettava toiminnallisuus johon yhteisöllistä tekniikka käytetään ja sen asema käyttäjän toimintojen joukossa. Jos yhteisöllisellä tekniikalla tuetaan käyttäjän toissijaisia tehtäviä, on käyttöliittymän pitäminen yksinkertaisena varmaankin ensisijainen tavoite. Jos tekniikka on taas sovelluksen keskipisteessä, tarvitaan myös kontrollin mahdollistamiseen tarkempia ja siten mahdollisesti monimuotoisempia mekanismeja.

4.3.5 Päättely

Päättelytekniikat ovat varsinkin sovelluskohteidensa puolesta läheistä sukua oppimistekniikoille. Molemmilla tekniikoilla voidaan muuttaa ohjelman toiminnallisuutta vastaamaan paremmin uutta tilannetta. Toiminnan esittämiseen ja hallintaan vaikuttavia eroja-

VAATIMUKSET PÄÄTTELYLLE

- toiminnon tarkoituksen havainnollistus
- päättelyajankohdan julkistus
- päättelyperusteiden julkistus

– tekniikan käytön hallinta

kin on. Oppiminen tapahtuu pitkän ajanjakson kuluessa, kun päättely tapahtuu tietyssä tilanteessa, helposti määriteltävänä ajankohtana. Tästä johtuen päättelyllä voidaan ajatella olevan selkeä tulos, oppiminen on puolestaan jatkuvasti käynnissä oleva prosessi. Päättelystä kertovalle käyttöliittymälle asetetut vaatimukset on koottu oheiseen kuvioon.

Kuten on oppimisenkin kohdalla, myös päättelyä käyttävän ohjelman tulisi antaa käyttäjälle tietoa siitä, *mihin päättelyä käytetään*. Tämän tiedon välittäminen voidaan joissain tapauksissa yhdistää ajalliseen informaatioon siitä, *milloin ohjelma suorittaa päättelyä*. Varsinkin näiden kahden tiedon varassa käyttäjän on mahdollista käsittää mikä on päättelyn merkitys ja millaisiin asioihin päättely voi järjestelmässä vaikuttaa. Ajallinen informaatio päättelyn suorittamisesta voi osaltaan vähentää tarvetta kertoa päättelyn perusteella tehtävästä muutoksesta. Näin ainakin siinä tapauksessa, että muutos seuraa välittömästi päättelyä.

Jos päättelyn tulokset ovat käyttäjän mielestä intuitiivisesti mielekkäitä, ei suurtakaan tarvetta päättelyprosessin selittämiseen ole. Tietokoneella toteutetussa päättelyssä näin ei läheskään aina kuitenkaan käy ja käyttäjän hyväksynnän saamiseksi on oleellista *selittää mihin päättely perustuu*. Päättelyprosessin tunteminen auttaa käyttäjää arvioimaan tekniikasta mahdollisesti saatavia hyötyjä.

Kuten kontrolloinnin kanssa muutenkin, yksinkertaisin menetelmä mahdollistaa päättelyn kontrollointi on antaa mahdollisuus *kieltää ja sallia sen käyttö* ohjelmassa. Tällä tekniikalla käyttäjälle saadaan aina ylivertainen valta koko prosessin suhteen, vaikka prosessin yksityiskohtia ei voisikaan säädellä. Päättelyprosessin yksityiskohdista mielekkäästi kontrolloitavia kohteita voisi olla päättelyssä käytettyjen sääntöjen valinta. Tällöin käyttäjä voisi poistaa päättelystä esimerkiksi omaan toimintatapaansa sopimattomia sääntöjä ja näin parantaa päättelyn toimivuutta. Uusien sääntöjen lisääminen on myös eräs vaihtoehto lisätä kontrollia, jos se kyetään tekemään käyttäjälle ymmärrettävällä tavalla.

4.3.6 Tehtävien tunnistus

VAATIMUKSET TEHTÄVIEN TUNNISTAMISELLE

- käyttökohteiden julkistus
- tehtävän tunnistusajankohdan julkistaminen
- tehdyn tulkinnan julkistaminen

– lupa tehtävien tunnistamiseen

Tehtävien tunnistaminen muistuttaa käyttäjän toimien tarkkailua, mutta toiminnot eroavat tuloksen abstraktiotason suhteen. Tunnistettaessa tehtäviä tulos on huomattavasti abstraktimpi käsite kuin kerättäessä implisiittistä syötettä. Samalla myös tunnistetun käsitteen hyödyntäminen on helpompaa, koska abstrakti käsite on tyypillisesti lähempänä sovellusalan käsitteistöä. Vaikka tehtäviä tunnistamalla voidaan parantaa käyttäjän toimien tarkkailun hyödynnettävyyttä, ei siitä kertovan käyttöliittymän tekeminen ole yksinkertaista. Oheinen kuvio tiivistää käyttöliittymälle asetettavat vaatimukset.

Jotta käyttäjä voisi odottaa ohjelmalta hyödyllisiä tuloksia perustuen suoritettavan tehtävän tunnistamiseen, on hänen tiedettävä, *millaisia tehtäviä ohjelma voi tunnistaa*. Tämän tiedon välittämistä helpottaa myös ajallinen tieto siitä, *milloin ohjelma tunnistaa* käyttäjän suorittamaa tehtävää ja siitä, milloin tunnistus on itse asiassa tapahtunut. Aikaan sidottu informaatio helpottaa myös tunnistusmekanismin ymmärtämistä.

Tähdellistä tietoa käyttäjälle on myös se, *millaiseksi tehtäväksi ohjelma käyttäjän toimet tulkitsee*. Kuten niin monessa muussakin auto-

maattisessa järjestelmässä, on tässäkin potentiaalinen vaara ohjelman virheelliseen toimintaan. Palautteen perusteella käyttäjän on mahdollista arvioida, voivatko tunnistamiseen perustuvat tulokset ylipäätään olla hyödyllisiä vaiko eivät. Se myös auttaa ennakoimaan millaisiin toimiin tietoa voidaan käyttää, kun käyttäjä ymmärtää tunnistamisessa käytetyn abstraktiotason.

Tehtävien tunnistamisen yhteydessä mahdollisia kontrolloitavia kohteita on kolme: 1) onko tehtävien tunnistaminen päällä, 2) miten tehtävät tunnistetaan ja 3) mitä tehtäviä ohjelma saa tunnistaa. Yksittäisten tehtävien tunnistamisen estäminen voi olla käyttäjien yksityisyyden kannalta hyvin merkityksellistä, etenkin jos tehtävien tunnistamisesta saatuja tietoja käytetään yhteisöllisten tekniikoiden yhteydessä. Tieto käyttäjän suorittamista tehtävistä voi olla arkaluontoista tai luottamuksellista tietoa. Tehtävien tunnistamisen kontrollointi voi helpottaa ohjelman nopeaa muokkaamista käyttäjän työskentelytapoihin.

4.4 Vaatimusten ominaisuudet

Epäsuoraa hallintaa käyttävien ohjelmien käyttöliittymille asetettavia tavoitteita ja vaatimuksia on edellä lueteltu suuri joukko. Joukon hallitsemiseksi se on jaettu kategorioihin, jotka rajaavat ja selkeyttävät vaatimusten tarkoitusta.

Esitetyt vaatimukset perustuivat ensinnäkin epäsuoran hallinnan määritelmään ja sen luonteeseen. Toisin sanoen vaatimukset on johdettu epäsuoran hallinnan luonteesta ajatellen mitä sen toteuttaminen tarkoittaisi käyttöliittymän kannalta. Toinen vaatimusten laadintaan vaikuttanut seikka oli aiemmissa tutkimuksissa saadut kokemukset. Rakennettuja prototyyppijärjestelmiä testaamalla on saatu tietoa epäsuoraan hallintaan liittyvistä käytettävyysongelmista.

Oheinen kaavio (Taulukko 2) kokoaa yhteen esitetyt vaatimukset sekä luo katsauksen niiden merkitykseen. Vaatimuksia on järjestetty taulukossa kokoamalla vaatimuksia kategoriaotsikoiden alle sekä vaikutusala-ominaisuuden avulla. Vaikutusala määrittelee sen, missä yhteydessä vaatimusta olisi sovellettava. Yleiset vaatimukset koskevat käyttöliittymän kaikkia osia ja osakohtaiset käyttöliittymän jotakin tunnistettua osaa (palaute, tulokset, kontrolli, konfigurointi). Tekniikkakohtaisia vaatimuksia sovelletaan vain niihin käyttöliittymän osiin, jotka ovat kiinteässä yhteydessä johonkin epäsuoran hallinnan mahdollistavaan toteutustekniikkaan. Vaikutusalan viittaama osa tai tekniikka selviää vaatimuksen kategoriasta.

Taulukko 2 Epäsuoran hallinnan käyttöliittymälle asettamat vaatimukset ja niiden ominaisuuksia

Painoarvo:
· Pieni
▪ Kohtalainen
■ Huomattava

VAATIMUS	Vaikutusala	Yleinen	Osakohtainen	Tekniikkakohtainen	Painoarvo	Kokonaisuudelle	Vaikutusalalle
Yleiset tavoitteet							
Ei tarvetta opetteluun		■				■	■
Käyttö huomaamatonta		■				■	■
Käyttö vaivatonta		■				■	■
Tieto toiminnoista ja olemassaolosta (Palaute)							
Tietoisuus olemassaolosta		■				■	■
Häiritsemättömyys		■				■	■
Toimintaperiaatteiden selvennys		■				■	■
Tulokset							
Häiritsemättömyys			■			■	■
Hyödyllisyyden arviointi			■			■	▪
Hyödyntämisen helppous			■			■	▪
Kontrolli							
Nopeus ja helppous			■			■	■
Kompaktius			■			·	·
Ei pysyvää liikettä			■			·	▪
Konfigurointi							
Nopeasti opittava			■			▪	■
Helposti muistettava			■			·	·
Palautteen antavuus			■			▪	■
Kontrollin mahdollisuus			■			·	▪
Implisiittinen syöte							
Havaittavuus				■		▪	■
Ajankohta				■		·	▪
Käyttötarkoitus				■		▪	·
Keräyksen kontrolli				■		·	▪
Käyttäjäprofiili							
Tieto sisällöstä				■		▪	■
Tieto päivitysperiaatteesta				■		·	·
Julkistamisen kontrolli				■		·	▪
Oppiminen							
Tieto muuttuvista toiminnoista				■		▪	■
Muutoksen ajankohta				■		·	·
Perusteiden ymmärtäminen				■		▪	■
Muutoksesta päättäminen				■		·	·
Muutostiheyden säätäminen				■		·	▪
Yhteisölliset tekniikat							
Ajankohta				■		·	·
Käyttötarkoitus				■		▪	■
Yhteisö				■		·	▪
Kontrollin mahdollisuus				■		·	▪
Päättely							
Käyttötarkoitus				■		▪	▪
Ajankohta				■		·	·
Perusteiden ymmärtäminen				■		▪	■
Kontrollin mahdollisuus				■		·	▪
Tehtävien tunnistus							
Tunnistetut tehtävät				■		·	▪
Ajankohta				■		·	·
Tieto tunnistustuloksesta				■		▪	■
Kontrollin mahdollisuus				■		·	·

Vaatimusten painoarvojen tehtävänä on laittaa vaatimuksia tärkeysjärjestykseen. Kolmiportainen asteikko on ajateltava ainoastaan järjestysasteikoksi, jota voidaan hyödyntää konfliktitilanteissa. Vaatimusten painoarvoa on taulukossa arvioitu erikseen sekä kokonaisuuden (epäsuoraa hallintaa käyttävän ohjelman käytettävyyden kannalta) että vaikutusalan kannalta. Vaikutusala viittaa taulukon ensimmäisissä sarakkeissa tunnistettuihin vaikutusalueisiin.

5. KÄYTTÖLIITTYMÄRATKAISUT

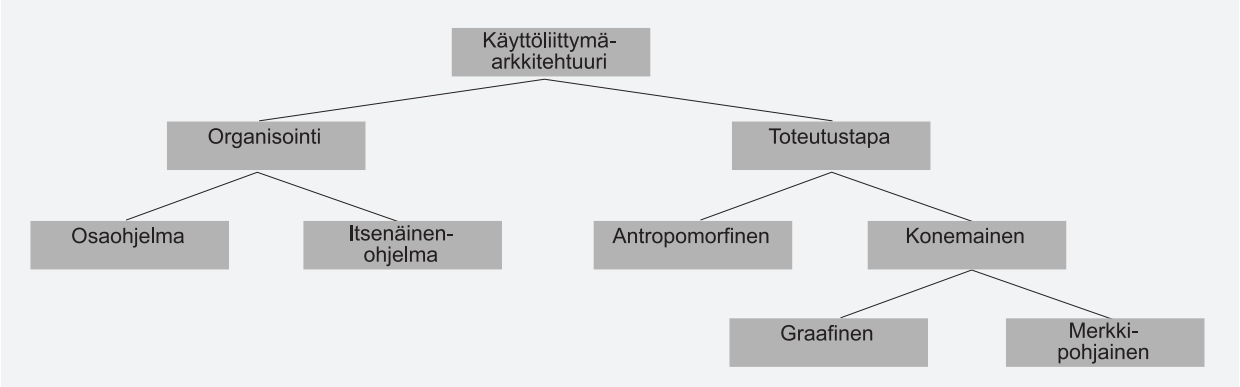
5.1 Käyttöliittymäarkkitehtuuri

Käyttöliittymäarkkitehtuuri määrittelee sen, miten ohjelmien käyttöliittymät on organisoitu ja millaisia toteutustapoja ne käyttävät. Esimerkiksi Microsoft Windows -käyttöjärjestelmän tukemassa käyttöliittymässä toisiinsa liittyvä tieto voidaan organisoida joko MDI (Multi Document Interface) tai SDI (Single Document Interface) käyttöliittymien [Windows 95] ympärille. Toteutustavatkin on määritelty: käytettävissä on suoravaikutukseen perustuvia toimintatapoja, kuten lomakkeiden käyttöä, raahausta ja dialogi-ikkunoita. Ohjelmien yhteinäinen käyttöliittymäarkkitehtuuri on osa käyttöliittymien standardointia.

Epäsuorasti hallittujen ohjelmien arkkitehtuuria ei ole mielekästä esittää samoilla primitiiveillä, kuin mitä käytetään esimerkiksi Windows käyttöliittymän yhteydessä. Seuraavassa pohditaankin millaisista osasista epäsuorasti hallittujen ohjelmien käyttöliittymäarkkitehtuuri koostuu. Osien tunnistaminen perustuu jo käytössä olevien ratkaisujen tarkasteluun. Myöhemmin tutkitaan tarkemmin näiden osien ominaisuuksia.

5.1.1 Epäsuorasti hallittujen ohjelmien käyttöliittymäarkkitehtuuri

Koska epäsuorasti hallittujen ohjelmien kohdalla käyttöliittymävaatimukset ovat osin poikkeavat tavallisista käyttöliittymävaatimuksista, valmiita käyttöliittymäarkkitehtuureita ei voida suoraviivaisesti soveltaa. On kehitettävä uusia organisoiti- ja toteutustapoja, jotka soveltuvat paremmin epäsuorasti hallittujen ohjelmien



Kuva 10 Epäsuoraa hallintaa käyttävien käyttöliittymien arkkitehtuurin osat

käyttöliittymien kuvaamiseen. Tämän työn yhteydessä käytetyt arkkitehtuuria kuvaavat käsitteet ja niiden suhteet on kuvattu oheisessa kuvassa (Kuva 10).

Organisointinsa puolesta epäsuorasti hallittu toiminto voi olla joko *itsenäinen ohjelma* tai toisen *sovelluksen osa*. Toisin kuin tavallisissa ohjelmissa, epäsuorasti hallittujen ohjelmien kohdalla ei ole itsestään selvää, että ohjelma on itsenäinen ohjelmainstanssi. Epäsuora hallinta tarvitsee tyypillisesti käyttäjän implisiittistä syötettä, jota on kätevintä kerätä jonkin työkalusovelluksen yhteydessä, jota käytetään ensisijaisten tehtävien suorittamiseen. Siksi epäsuorasti hallitut ohjelmat (tai toiminnot) ovat monesti sovelluksen osia. Kirjallisuudessa on kuitenkin nähtävissä prototyyppijärjestelmiä, joissa epäsuorasti hallittu toiminto on pääasiassa, joita siten voidaan pitää itsenäisinä ohjelmina. Tällaisia ohjelmia ovat esimerkiksi NewT [Sheth 94], Amalthaea [Moukas 96] ja Selection Recongnition Agent [Pandit and Kalbag 97].

Toinen arkkitehtuuriin liittyvä ja käyttöliittymiä organisoiva asia on käyttöliittymän toteutustapa, jolla se toimintansa ja hallintaansa esittää. Epäsuorasti hallittujen toimintojen käyttöliittymän esitystapa voi olla *antropomorfinen* (ihmismäinen) tai perinteinen *konemainen*. Konemaiset toteutustavat voidaan jakaa edelleen graafisiin (lomakkeisiin, dialogi-ikkunoihin, suoravaikutteisiin komponentteihin perustuviin) ja komentoperustaisiin.

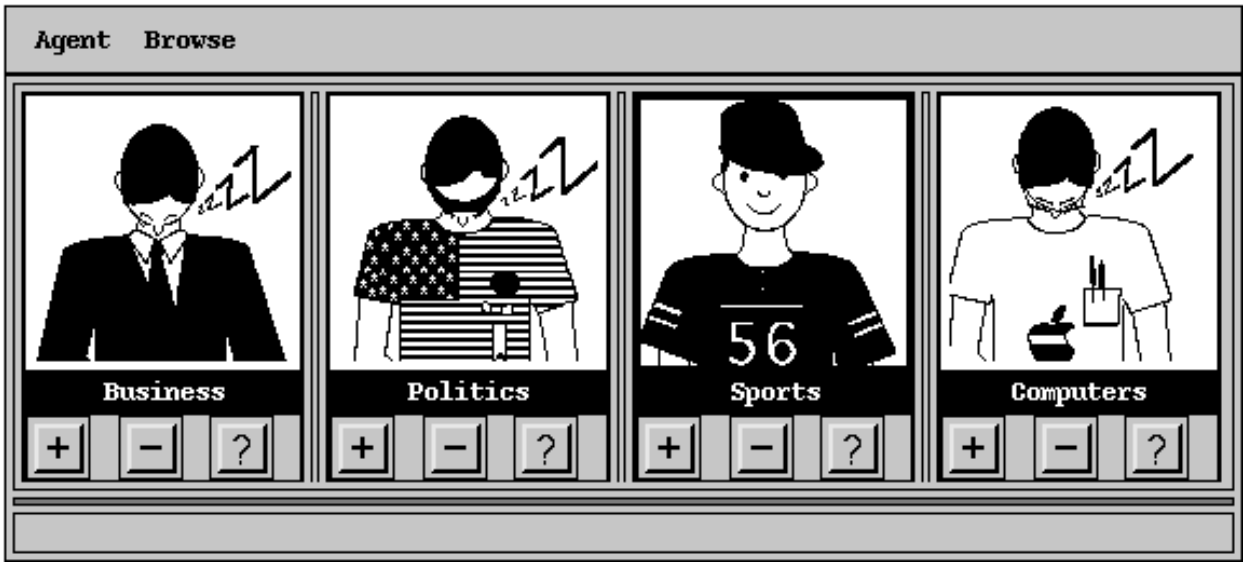
Antropomorfinen toteutustapa on tyypillinen nimenomaan epäsuorasti hallituille ohjelmille, kun suoravaikutteisuuteen perustuvat konemaiset toteutustavat ovat tyypillisiä myös muissa käyttöliittymissä. Ehkä juuri siksi antropomorfinen käyttöliittymä mahdollistaa melko eheän käyttöliittymäkonseptin muodostamisen epäsuorasti hallittujen toimintojen käyttöliittymien toteuttamiselle. Siinä suunnittelun perustana on selvä metafora, josta voidaan johtaa yhdenmukaisia käyttöliittymiä.

5.1.2 Itsenäiset ohjelmat

Itsenäisten epäsuorasti hallittujen ohjelmien käyttöliittymät muistuttavat suuresti minkä tahansa normaalin ohjelman käyttöliittymää. Graafisessa ympäristössä ne esimerkiksi koostuvat tyypillisesti yhdestä pääikkunasta ja mahdollisista apu- ja dialogi-ikkunoista.

Itsenäiseksi toteutetulla ohjelmalla on todennäköisesti laajemmat mahdollisuudet käyttöliittymän toteutuksessa kuin isäntäohjelmasta riippuvalla ohjelmalla. Käytännössä mahdollisuudet tarkoittavat enemmän kuvaruutupinta-alaa ja pienempää vaaraa häiritä tahattomasti käyttäjän muita toimia. Jos epäsuorasti hallittavana on vain sovelluksen osa, on sen noudatettava isäntäsovelluksen asettamia rajoitteita. Käyttäjän huomion keskipisteessä on tällöin tyypillisesti jokin muu asia kuin epäsuorasti hallittu toiminnallisuus, joka on tyypillisesti luonteeltaan ensisijaista toimintoa tukeva.

Itsenäinen epäsuorasti hallittu ohjelma on eräässä mielessä kummajainen. Mistä tällainen ohjelma voi saada käyttäjän implisiittistä syötettä, jota se tarvitsee epäsuoran hallinnan toteuttamiseksi. Tai miten mielekäs voi ylipäätään olla ohjelma, jonka ensisijaista toimintoa ei voi kontrolloida suoraan. On kuitenkin olemassa esimerkkejä, jotka osoittavat myös tällaisen organisoinnin toimivaksi.



Kuva 11 NewT-agenttiohjelman käyttöliittymä

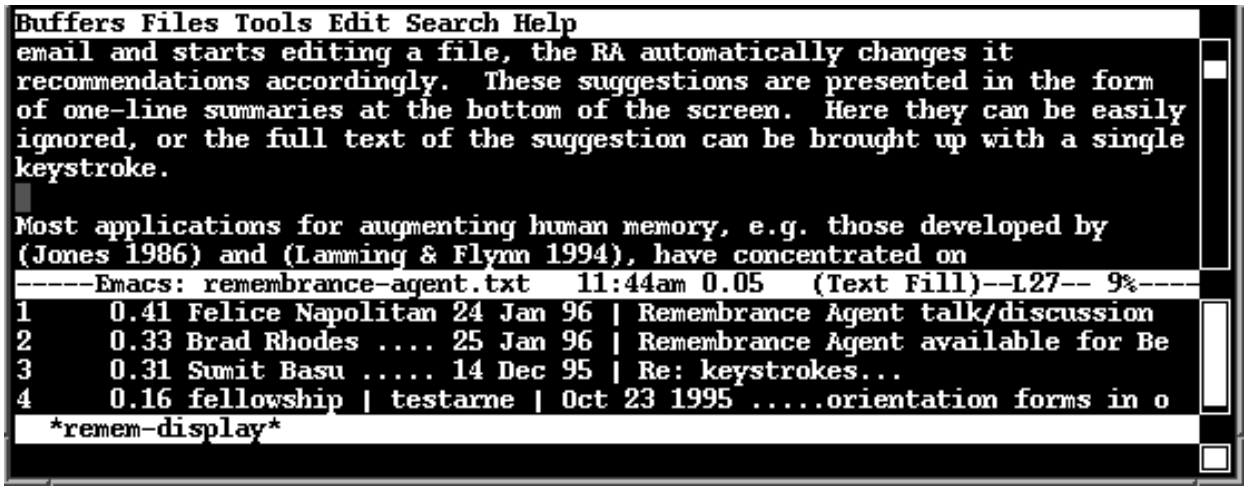
NewT [Sheth 94] on tiedonhakuohjelma (Kuva 11), jota osittain hallitaan epäsuorasti. NewTssä käyttäjä joutuu aluksi määrittelemään eksplisiittisesti haun kriteerit. Myöhemmin kriteerien parantamiseen ja muuttamiseen käytetään kuitenkin epäsuoraa menetelmää, jossa käyttäjä ainoastaan arvioi hakutulosten kiinnostavuutta. Ohjelma oppii palautteen perusteella käyttäjän kiinnostuksen kohteita ja muokkaa hakukriteerejä sen mukaan. Ohjelman hallitseminen on siinäkin mielessä epäsuoraa, että käyttäjän ei tarvitse (eikä pysty) määrittelemään hakujen ajankohtaa.

NewTn käyttöliittymä on toteutettu graafisessa ympäristössä, joten ikkunat luovat sen perusrakenteen. Ohjelman käyttö perustuu osittain lomakkeen täyttämiseen, joiden toteutuksessa on käytetty tavallisia käyttöliittymäkomponentteja. Eri hakuagenttien (hakuprofiilien) tilaa ja luonnetta kuvataa antropomorfisten hahmojen avulla, mutta nekin on NewTssä toteutettu melko perinteisesti piirroskuvin.

Itsenäisenä toteutettu epäsuoraan hallintaan perustuvan ohjelman käyttöliittymän perusrakenne ei poikkea tavallisesta ohjelmasta. Käytetyt hallinta- ja tiedonesitystavat voivat olla perinteisiä, joskin jossain tapauksissa epäsuoran hallinnan erityispiirteitä on korostettu esimerkiksi antropomorfisilla käyttöliittymän osilla.

5.1.3 Osaohjelmat

Toisen ohjelman osana toteutettavat epäsuorasti hallitut ohjelmat ovat mahdollisesti yleisemmin käytetty toteutusmuoto kuin itsenäiset ohjelmat. Sen yleisyyttä rajoittaa lähinnä toteuttamisen vaikeus ja työläys, sovelluskohteita olisi kyllä riittämiin. Käyttäjän näkökulmasta isäntäohjelman yhteyteen liitetty epäsuorasti hallittu ohjelma näyttäytyy vain yhtenä lisänä ohjelman tarjoamaan toiminnallisuuteen. Käyttöliittymää ajatellen tällainen rakenne on kuitenkin itsenäiseksi organisoitua toteutusta vaativampi, sillä nyt yhdessä ohjelmassa käytetään erilaisia vuorovaikutustapoja (esimerkiksi suoravaikutteisuutta ja epäsuoraa hallintaa). Toisin sanoen, osa ohjelman toiminnoista on automaattisia, osa ei.



Kuva 12 Remembrance Agent toiminnassa

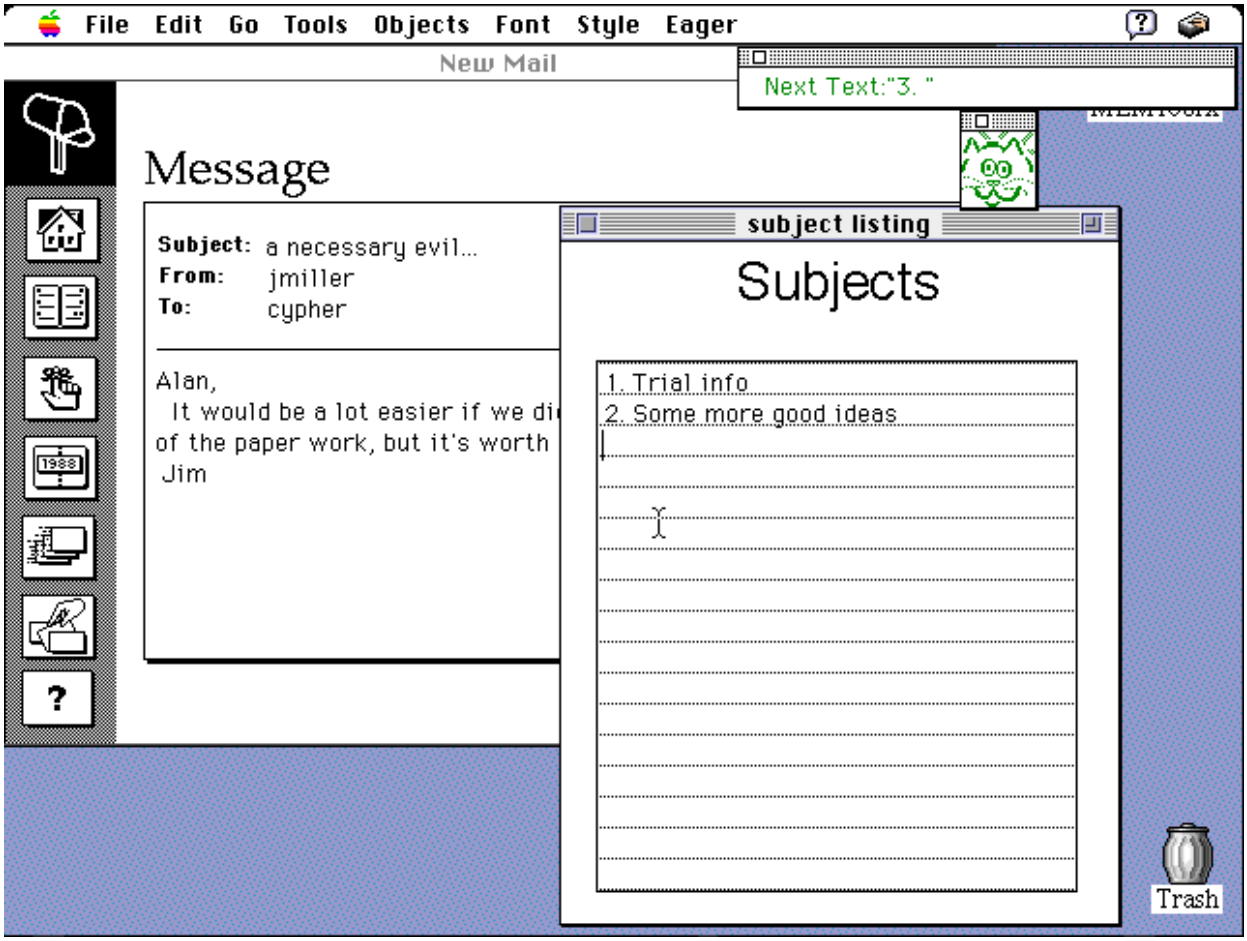
Epäsuorasti hallittu toiminnallisuus on yleensä pyritty erottamaan ohjelman muusta toiminnallisuudesta tavalla tai toisella. Tällä voidaan vähentää mahdollisuutta käyttäjän hämmentämiseen ja auttaa käyttäjää luomaan toimiva mielikuva ohjelman eri osien toiminnasta. Automaattisten ja käyttäjän hallittavissa olevien ohjelman osien tunnistaminen on tärkeää, jotta käyttäjä voisi tuntea kontrolloivansa edes sitä osaa ohjelmasta, joka on suorasti hallittavissa.

Ohjelman yhteyteen sulautetusta epäsuorasti ohjatusta toiminnallisuudesta voidaan esimerkkeinä mainita Remembrance Agent [Rhodes and Starner 96] ja Eager [Cypher 93]. Remembrance Agent (Kuva 12) muistuttaa käyttäjää muokattavaan tekstiin liittyvistä aiemmista dokumenteista ja sähköpostiviesteistä ja toimii Emacs-ohjelman yhteydessä. Eager (Kuva 13) generoi makro-ohjelmia käyttäjä suorittamien toistuvien tehtäväsarjojen perusteella HyperCard-ympäristössä.

Remembrance Agent käyttää ohjelman eri osien erottamiseen paikkaa jakaen isäntäohjelmansa editointi-ikkunan kahteen osaan. Autonomiset toiminnot tapahtuvat ainoastaan omassa eristetyssä tilassaan (kuvassa teksti-ikkunan alla oleva lista), eivätkä ne vaikuta mitenkään käyttäjän toimiin. Tulosten käyttämisestä on käyttäjä täydellisesti vastuussa.

Eagerin käyttöliittymä koostuu puolestaan animoidusta hahmosta, siihen liittyvistä pienistä paletti-ikkunoista ja muun käyttöliittymän eri osien korostuksista. Ohjelman epäsuorasti hallittujen toimintojen erottumista muusta toiminnallisuudesta tuetaan sekä mainitun hahmon että yhtenäisen värityksen avulla. Väriä käytetään ohjelman tekemien valintojen ilmaisemiseen ja animoidussa hahmossa. Ohjelman tuloksesta eli tuotetusta makro-ohjelmasta viestitetään tuomalla ohjelman suorituksen käynnistävä ikoni ruudulle.

Kun epäsuorasti hallittu toiminnallisuus on osa suurempaa ohjelma kokonaisuutta, on kiinnitettävä huomiota myös sen häiritsemättömyyteen. Kuten on aiemmin mainittu, epäsuorasti hallitut toiminnot soveltuvat parhaiten käyttäjän ensisijasta tehtävää tukeviin toimintoihin; Remembrance Agentin toiminnallisuus on hyvä esimerkki tästä.



Kuva 13 Eager-ohjelman käyttöliittymä

Remembrance Agent pyrkii välttämään käyttäjän häiritsemistä jättäen käyttämättä mahdolliset tulosten korostamiset ja värit. Lisäksi agentin varaamalle alueelle on määritelty maksimikoko, ettei se veisi liikaa käytettävissä olevasta kuvaruudun pinta-alasta. Eagerin tapauksessa häiritsemättömyyttä on lähestytty toisesta näkökulmasta.

Eager ei pakota käyttäjää arvioimaan tuotetun ohjelman soveltuvuutta missään vaiheessa suoritustaan. Tämä poikkeaa yleisestä tavasta, miten esimerkein tapahtuva ohjelmointi on toteutettu. Toisaalta tällainen proaktiivisuuden puute ei tuo mitään lisää, kun ajatellaan Eageria epäsuorasti hallittuna sovelluksena. Niinpä sen kohdalla voidaankin kysyä, onko korostukset tai ohjelman toimintavalmiutta kuvaava ikoni sittenkin liian voimakkaita keinoja kertoa ohjelman toiminnoista ja toteutuksesta.

5.1.4 Antropomorfiset käyttöliittymät

Koska epäsuora hallinta perustuu metaforaan ihmisten välisestä kommunikaatiosta, tuntuu luontevalta ja johdonmukaiselta toteuttaa tällaiseen vuorovaikutustapaan perustuvan ohjelman käyttöliittymäkin ihmishahmoiseksi eli *antropomorfiseksi*. Tällöin ohjelma antaa tulosteensa ihmismäisesti luonnollisella kielellä (puhuttuna tai kirjoitettuna), eleillä tai ilmeillä.

Oleellinen piirre antropomorfisille käyttöliittymille on elävän hahmon esittäminen osana käyttöliittymää, kuten Eager-ohjelmassa käytetty kissa (Kuva 14). Elävän ja personoidun hahmon on katsottu olevan jopa siinä määrin keskeinen toteutustapa epäsuorasti hallituille agenttiohjelmille, että sitä on ehdotettu jopa tällaisia ohjelmia määriteleväksi ominaisuudeksi [Laurel 90].

Kuva 14 Eager-ohjelman antropomorfinen hahmo

Antropomorfisia käyttöliittymiä luokitella niiden käyttämän hahmon perusteella. Antropomorfisia käyttöliittymiä on rakennettu perustuen realistiseen puhuvaan ihmisen päähän, animoituun hahmoon (ihminen, eläin, satuhahmo), staattisiin kuviin ja pelkkään tekstiin [Raisamo 99]. Tekstiin perustuvassa käyttöliittymässä lähinnä vain käytetyn kielen muoto voi viitata antropomorfiseen esitystapaan.

Inhimillisten piirteiden lisääminen ohjelmaan voi saattaa käyttäjän odottamaan ohjelman toiminnalta liikoja [Maes and Shneiderman 97, s. 56]. Ihmiset liittävät inhimillistettyyn hahmoon helposti älykkyyden ominaisuuksia, vaikka näillä ei todellisuudessa olisi ohjelmassa katetta. Älykkyys merkitsisi kykyä suorittaa monimutkaisia tehtäviä tai ymmärtää epämääräisiä komentoja. Kun ohjelma ei näitä odotuksia täytä, käyttäjät pettyvät eivätkä ehkä kykene ymmärtämään syytä ohjelman epäonnistumiseen.

Jossain määrin älykkyyden odotuksia voidaan vähentää valitsemalla antropomorfinen esitystapa sopivasti. Tutkimuksissa on todettu, että esitykseen valittu hahmo ja dialogissa käytetty kieli vaikuttavat siihen, millaisena ohjelman älykkyys havaitaan. Erityisesti se, miten ihmismäinen hahmo on, tuntuu vaikuttavan havaittuun älykkyyteen [Koda 96, s. 85].

Kuten epäsuoran hallinnan, myös antropomorfisen käyttöliittymän motivaatio ja perimmäinen hyöty on tutussa vuorovaikutustavassa. Ihmiset ovat toistensa kanssa vuorovaikutuksessa aina, eikä siihen perustuvaa tietokoneohjelman hallintatapaa tarvitsisi siis erikseen opetella. Luonnollisella kielellä pystytään ilmaisemaan myös huomattavan monimutkaisia asioita suhteellisen yksinkertaisella lauseella, jonka lähes kaikki kykenevät tuottamaan. Tämän lisäksi antropomorfinen hahmo mahdollistaa erilaisten ilmeiden ja eleiden käytön informaation välityksessä, joka voi tarjota intuitiivisen tavan viestittää tiettyjä asioita, joiden kertominen voisi muuten olla vaikeaa. Tämä voi olla arvokasta etenkin palautteen antamisessa.

Luonnollisen keskustelun kaltainen vuorovaikutustapa voi helpottaa ohjelman osien erottelua. Tällaisessa vuorovaikutustavassa voidaan tarkoitukseen käyttää erilaisia tyylillisiä keinoja, joita esimerkiksi suoravaikutteisessa vuorovaikutustavassa ei pystytä käyttämään. Näin esimerkiksi viestien muotoileminen luonnollista kieltä muistuttaviksi ja niiden esittämien puhekuplin tai peräti puheena on mahdollista. Samanlaiseen esitystyyliin perustuvat toiminnot liitetään toisiinsa, mikä auttaa ohjelman toimintojen erottelua.

Kun ohjelma esittää tuloksensa antropomorfisesti, syntyy houkutus mahdollistaa myös syötteen antaminen samanlaisessa muodossa. Sen voi ajatella tarkoittavan esimerkiksi puhutun luonnollisen kielen käyttämistä syötteenä. Nykyisellä tekniikalla ei ole kuitenkaan mahdollista tehdä ohjelmaa, joka ymmärtäisi luonnollista kieltä täysin rajoituksetta, mistä seuraa virheitä syötteen ymmärtämises-

sä. Käyttäjien on lähes mahdoton tietää tai ymmärtää, millaisia rajoituksia heidän pitäisi ilmaisuissaan huomioida.

Jos puhutun kielen muotoa taas rajoitetaan tiettyihin komentoihin, saavutettava ymmärtämisen taso paranee, mutta kommunikointi muuttuu keinotekoisen tuntuiseksi. Tutkimusten mukaan se tekee ohjelman käyttämisestä vastenmielistä [Ball *et al.* 97].

Käytettävissä oleva teknologia aiheuttaa ongelmia luonnollisen kielen käytössä, mutta myös ihmistenvälisen kommunikaation luonne voi niitä aiheuttaa. Luonnollisessa kielessä on mahdollista käyttää hyvin epämääräisiä ilmauksia, jotka eivät sovellu työkalun ohjaukseen [Shneiderman 95, s. 15]. Epämääräisistä toimintaohjeista voi olla seurauksena epähaluttuja toimintoja, jotka vähentävät käyttäjien luottamusta ohjelmaa kohtaan. Epämääräisten ohjeiden salliminen voi haitata myös ohjelman kontrolloitavuutta, koska ohjelma voi tulkita komennoksi sellaisiksi käyttäjän mielestä soveltumattomia lauseita. Siksi antropomorfisen käyttöliittymän käyttö ohjelman kontrolloinnissa tai konfiguroinnissa voi olla vaikeaa.

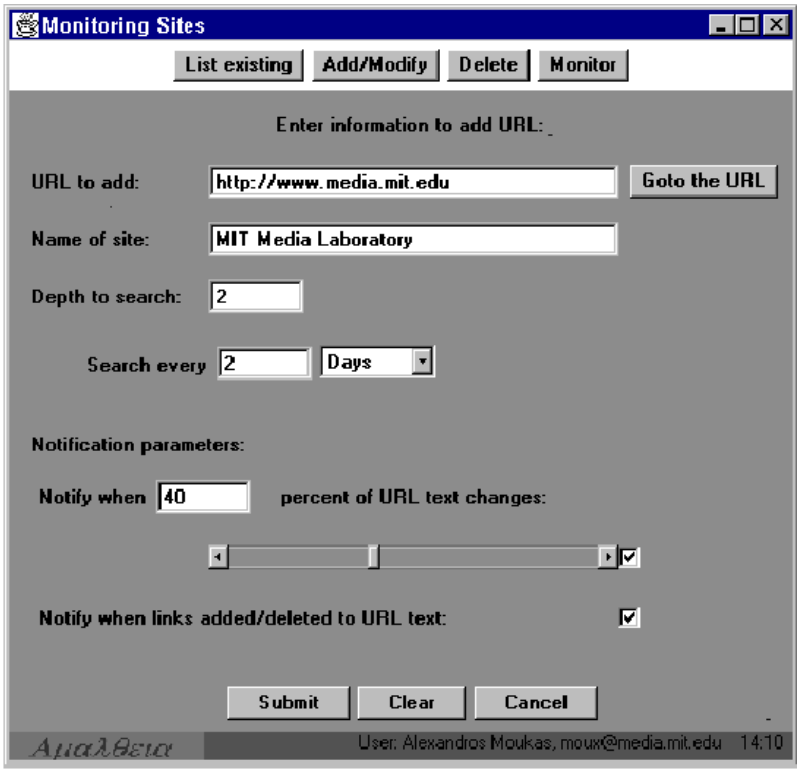
5.1.5 Graafiset käyttöliittymät

Vaikka inhimillistetty esitystapa tuntuu luontevalta valinnalta epäsuorasti ohjattujen ohjelmien käyttöliittymien toteutustavaksi, voidaan sama toiminnallisuus toteuttaa hyvin myös perinteisen graafisen käyttöliittymän keinoin. Vaikka graafisen käyttöliittymän perusidea ei soinnu epäsuoran hallinnan periaatteiden kanssa yhtä hyvin yhteen kuin antropomorfinen käyttöliittymä, sen käytöllä on kuitenkin saatavissa tiettyjä etuja. Näistä tärkeimpiä ovat ohjelman kontrollointitavan tuttuus suhteessa ohjelman toimintaympäristöön ja ohjelman toiminnallisuudesta annettavan kuvan mekanistisuus, joka yleensä heijastaa ohjelman todellista toimintatapaa ja rakennetta.

Vaikka kaikkiin epäsuorasti hallitun toiminnon osiin graafinen käyttöliittymä ei sopisikaan, osassa sen ominaisuudet ovat eduksi. Kontrollin ja konfiguroinnin mahdollistavat käyttöliittymät ovat graafiselle käyttöliittymälle ihanteellisia sovelluskohteita. Tällaisissa tarkoituksissahan sitä käytetään muissakin ohjelmissa yleisesti. Esimerkiksi Amalthaea prototyyppi käyttää hyvin perinteistä käyttöliittymää juuri tällaisten toimintojen esittämiseen (Kuva 15). Palautteen antaminen ohjelman toiminnasta, joka on epäsuoraa hallintaa käyttävässä ohjelmassa hyvin tärkeässä asemassa, voi olla kuitenkin ongelmallisempaa.

Graafisissa käyttöliittymissä on runsaasti erilaisia tapoja antaa käyttäjälle palautetta, mutta niiden soveltuvuus epäsuorasti hallitussa ohjelmassa käytettäväksi ei ole itsestään selvää. Yleensä palaute on tärkeä huomata, joten se tehdään hyvin näkyväksi. Epäsuorasti hallitun toiminnon yhteydessä tämä voi olla kuitenkin häiritsevää.

Toisaalta normaalisti graafisessa käyttöliittymässä palaute liittyy käyttäjän samalla hetkellä suorittamaan toimintoon, jolloin palautteen kohde on helposti ymmärrettävissä. Epäsuorasti hallitussa ohjelmassa tilanne on toinen, sillä ohjelman antama palaute ei välttämättä liity suoraan käyttäjän sen hetkiseen toimintaan, vaan ohjelman itsenäisestä toiminnasta aiheutuneeseen tilan muutokseen.



Kuva 15 Amalthaea-prototyypin konfigurointikäyttöliittymä

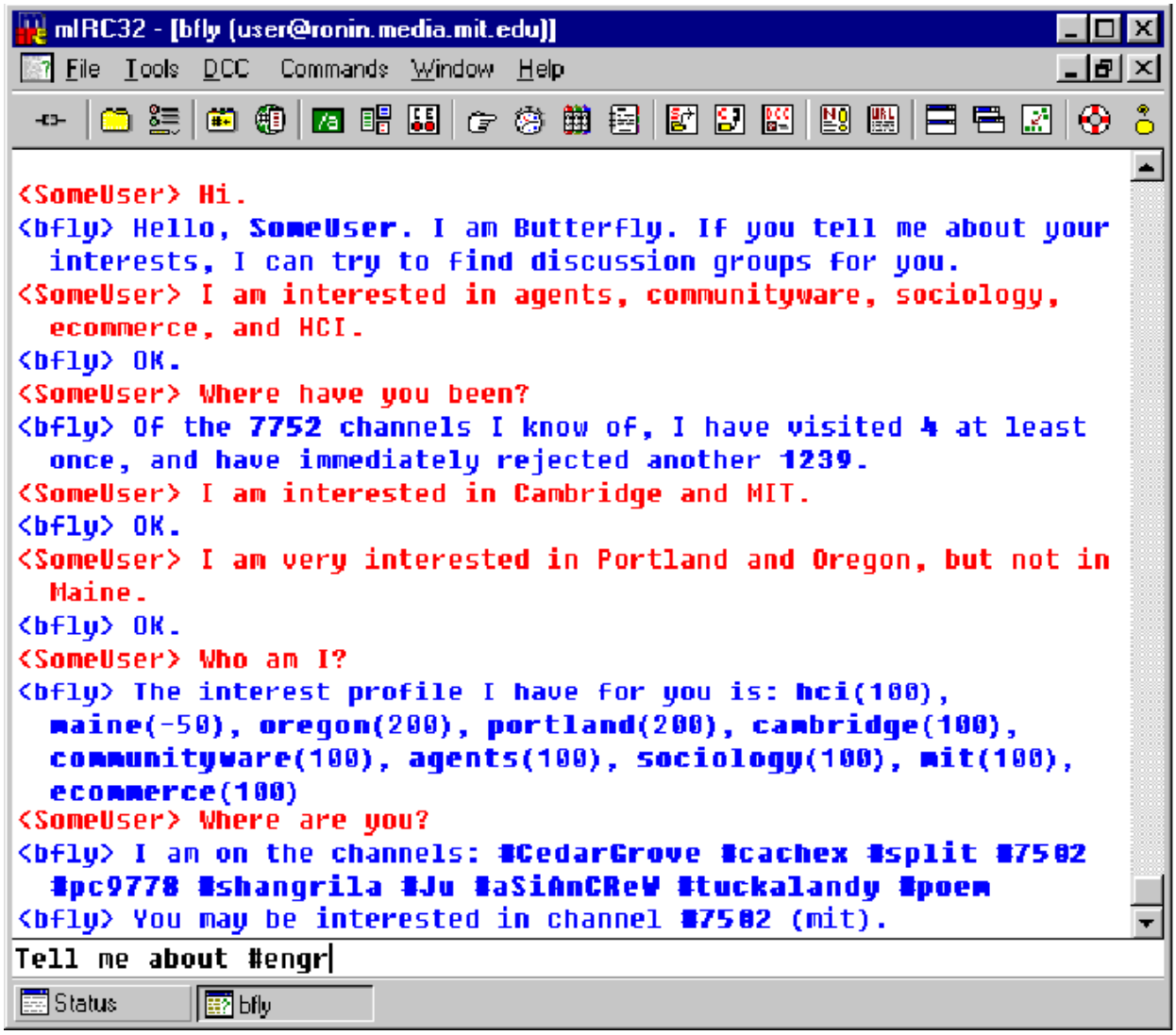
Palautteen antamisen ohella ongelmallista voi olla myös epäsuorasti hallitun ohjelman ja muun ohjelman käyttöliittymän osien erottaminen. Tämä ongelma koskee ohjelmia, joissa vain jokin toiminnallisuuden osa on toteutettu epäsuorasti hallittavana, mutta ei koko sovellus. Graafisissa ohjelmissa käyttöliittymän osia on eroteltu lähinnä niiden sisältämän tiedon tai toiminnallisuuden perusteella, eikä juurikaan toimintojen autonomisuuden perusteella. Valmiita ratkaisuja erottelun tekemiseen ei siten ole.

5.1.6 Komentopohjaiset käyttöliittymät

Epäsuorasti hallittujen ohjelmien käyttöliittymät voivat olla myös teksti- tai komentopohjaisia. Komentopohjaisissa käyttöliittymissä käyttäjä antaa ohjelmalle ennalta määriteltyjä komentoja, jotka ohjelma sitten puolestaan suorittaa. Komennot ovat luonteeltaan lyhyitä, eikä niiden olemassaolosta ole näkyviä merkkejä. Komentokielet voidaan suunnitella hyvin ilmaisuvoimaisiksi ja tehokkaiksi. Komentokäyttöliittymät pohjautuvat varhaisiin tietokoneiden käyttöjärjestelmien toteutuksiin [Shneiderman 98, s. 279].

Komentopohjaisen käyttöliittymän rakentaminen epäsuorasti hallittuun ohjelmaan perustuu monessa tapauksessa ohjelman käyttöympäristöön. Tällöin halutaan säilyttää yksi tapa kommunikoida ohjelman kanssa, ettei käyttäjän tarvitsisi opetella useampia tapoja ohjelman ohjaamiseen.

Komentopohjaisissa käyttöliittymissä palaute esitetään yleensä vastauksena käyttäjän komentoon. Tällainen mekanismi ei ole toimiva epäsuorasti ohjatussa ohjelmassa, joka voi tuottaa tuloksia muulloinkin kuin käyttäjän antaman komennon seurauksena. Palautteen esittämisen ongelma on tässä siis samanlainen kuin suora-



Kuva 16 Internetin keskustelukanavia tutkiva Butterfly-ohjelma toiminnassa

vaikutteisten käyttöliittymien kohdalla: epäsuorassa hallinnassa palautetta tarvitsee antaa muissakin tilanteissa kuin komennon antamisen yhteydessä.

Eräs mahdollisuus ratkaista palautteen antamisen ongelmia komentopohjaisessa käyttöliittymässä on varata erityinen paikka palautteen antamiselle. Näinhän oli tehty mm. Remembrance Agentissa, joka esittää tuloksensa (antaa siis palautetta tekemisistään) omassa tilassaan. Paikka ilmaisee toiminnon, joka tuloksen tai palautteen antaa. Sen lisäksi toiminnolle varattu paikka ratkaisee myös asynkronisuuden ongelman. Ongelma aiheutuu siitä, että epäsuorasti hallittu ohjelma voi tuottaa tuloksensa koska tahansa, asynkronisesti käyttäjän toimien kanssa.

Butterfly-ohjelma [Van Dyke *et al.* 99], joka etsii käyttäjää kiinnostavia Internetin keskustelukanavia, on ratkaissut synkronisuuden ongelman verraten hallitusti. Sen käyttöliittymäidea on lainattu Internetissä käytettävistä keskusteluohjelmista (Kuva 16), mikä jo sinällään vastaa ongelmaan. Keskusteluun osallistujat voivat lähettää kommentin keskusteluun koska hyvänsä, joten asynkronisuuden ongelma liittyy jo ohjelman peruskäyttöliittymään. Butter-

fly toimii yhden keskustelijan tapaan, joka voi kirjoittaa omia kommenttejaan ryhmään ja jolta voi pyytää palveluksia tietyillä komennoilla.

5.2 Tekniikoiden havainnollistaminen

Jotta käyttäjän olisi mahdollista ymmärtää epäsuoran hallinnan toimintaperiaatteet, on hänelle paljastettava miten sen toteuttavat tekniikat toimivat. Yhdessä sovelluksessa epäsuora hallinta on voitu toteuttaa usealla eri tekniikalla, joiden kaikkien toiminnasta on käyttäjän mahdollisesti saatava tietoa. Tämän lisäksi myös tekniikoiden väliset suhteet voivat olla toimintaperiaatteen ymmärtämisen kannalta oleellisia.

Seuraavassa käydään läpi kirjallisuudessa tunnettuja käyttöliittymäratkaisuja, joita on käytetty eri tekniikoiden yhteydessä. Kartoitusta ohjaa työssä aiemmin tekniikoihin liitetyt käyttöliittymävaatimukset, jotka toimivat tarkastelua ohjaavina käsitteinä. Lukemisen helpottamiseksi kutakin tekniikkaa käsittelevän alakohdan yhteyteen on liitetty siihen liittyvät vaatimukset tiivistetyssä muodossa.

Lopullisina ratkaisuina esitettyjä ideoita ei kannata pitää, mutta suuntaa antavina esimerkkeinä ne ovat hyödyllisiä. Näiden esimerkkien perusteella voidaan myöhemmin johtaa yleisempiä periaatteita siitä, millaisia toteutustapoja käyttöliittymien rakentamiseksi voidaan käyttää.

5.2.1 Implisiittisen syötteen keräys

VAATIMUKSET IMPLISIITTISEN SYÖTTEEN KERÄYKSELLE

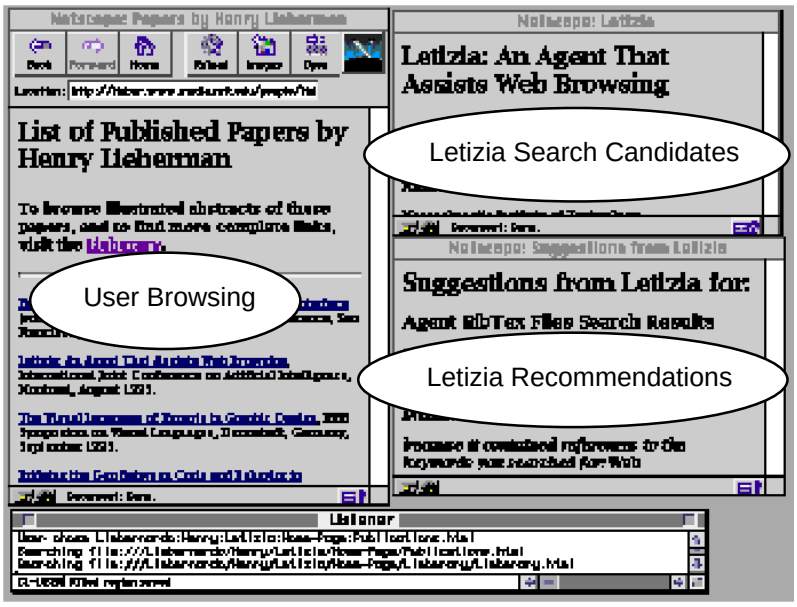
- toiminnon havaittavaksi tekeminen
- havaittujen toimien julkistus
- keräysajankohdan julkistus
- havaintojen käyttötarkoituksen julkistus

– lupa havainnointiin

Vaikka implisiittisen syötteen keräys on kaikkien epäsuorasti hallittujen ohjelmien toiminnan kulmakivi, se ei läheskään aina näy ohjelman käyttöliittymässä mitenkään. Epäsuoran hallinnan ihanteen mukaan ohjelma toimii täydellisen itsenäisesti, eikä jatkuvasti kerro toimistaan tai häiritse käyttäjän keskittymistä turhaan. Käyttäjän voi kuitenkin olla vaikea tai mahdoton ymmärtää ohjelman toimintaperiaatteita, jos hän ei voi havaita sen toimintaa tai sen toimintaan vaikuttavia asioita mitenkään.

Letizia-agentti [Lieberman 97] on eräs esimerkki siitä, miten käyttäjän toimintojen tarkkailusta on käyttäjälle ohjelmissa kerrottu. Letizian käyttöliittymä koostuu kolmesta selainikkunasta, joista kahta ohjelma käyttää itsenäisesti ja yksi on varattu käyttäjälle. Lisäksi käyttöliittymässä on näkyvillä kuuntelijaikkuna (*listener*), jossa ohjelma raportoi tekemisiään ja havaintojaan tekstimuodossa (Kuva 17).

Letizian käyttämä kuuntelijaikkuna tarjoaa tarkkaa palautetta, josta käy ilmi sekä tehty havainto että havainnointiajankohta. Toteutus on kuitenkin melko karkea, sillä ensinnäkin tieto esitetään erillisessä ikkunassa, mistä sen huomaaminen ei ole itsestään selvää. On jopa mahdollista, että ikkuna jää toisten ikkunoiden alle tai se siirretään näkyvistä pois. Vaikka ikkuna olisikin järjestetty näkyviin, se on tyypillisesti melko kaukana siitä kohteesta, johon käyttäjän huomio on kiinnittynyt esimerkiksi havainnointihetkellä. Näin ollen käyttäjä ei ehkä havaitse uutta tietoa, tai jos huomaakin, hän joutuu siirtämään katsettaan melko pitkän matkan voidakseen



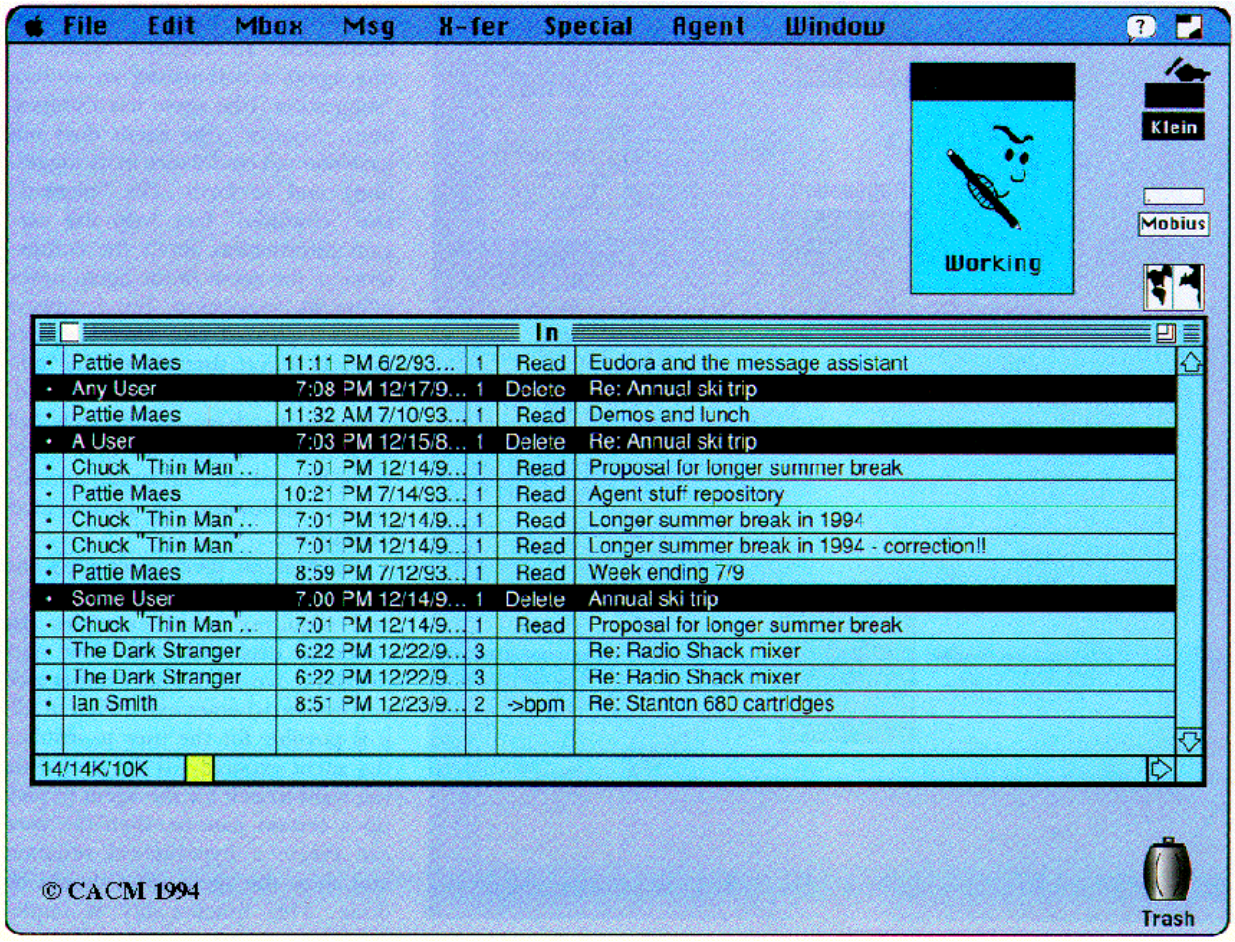
Kuva 17 Letizia-agentin käyttöliittymä

lukea viestin. Tämä haittaa käyttäjän keskittymistä ensisijaiseen tehtäväänsä.

Toiseksi havaintojen esittäminen tekstimuodossa ei tue tiedon nopeaa ja vaivatonta omaksumista. Sanoman ymmärtämiseksi on koko teksti luettava ja ymmärrettävä. Tämä voi tuntua joissain tapauksessa yksinkertaiselta ja helpolta tehtävältä, mutta jos viestejä on paljon, niiden lukeminen on vaivalloista. On muistettava, että havaintojen kuvaukset eivät edistä käyttäjän ensisijaista tehtävää, vaan ainoastaan helpottavat ohjelman autonomisen toiminnan ymmärtämistä. Monikaan ei varmasti ole valmis uhraamaan siihen paljoakaan resursseja, ei ainakaan niin paljoa kuin mitä jatkuvasti muuttuvien tekstilauseiden lukeminen vaatisi. Kolmanneksi Letizian käyttöliittymä ei mahdollista mitään kontrollia suhteessa implisiittiseen syötteen keräykseen.

Sähköpostin käsittelyagentti MAXIMS [Maes 94] käyttää toista melko yleistä menetelmää välittää tietoa käyttäjän tarkkailusta. MAXIMS-agentilla on antropomorfinen käyttöliittymä, jossa animoitu piirroshahmo kuvaa ohjelmaa ja sen toimintaa (Kuva 18). Hahmon ilmeet vaihtuvat sen mukaan, mitä ohjelma tekee ja millaisena se näkee käyttäjän toimet. Periaatteena tiedon välityksessä on siis käyttää ilmeitä ja eleitä. Tämän lisäksi MAXIMS käyttää myös teksti-ikkunoita tiedon välitykseen Letizian tapaan. Kuten varmaankin Letiziassa, tässäkin teksti-ikkunat on todennäköisesti tarkoitettu palvelemaan enemmän prototyypin rakentamista ja testaamista kuin varsinaista käyttöä.

Ilmeiden ja eleiden käyttäminen tiedon välityksessä nojaa vahvasti tapahtumien ajalliseen yhteyteen. Toisin sanoen viestin tulee seurata välittömästi tapahtumaa, johon sen välittämä viesti liittyy. Jos viesti tulee liian myöhään, käyttäjän on mahdoton liittää sitä oikeaan toimintaan. Toisaalta vaikka viestin ajoitus olisikin oikea, voi viestin viittaama asia silti olla epäselvä. Esimerkiksi MAXIMS kykenee havaitsemaan sen, että käyttäjän tuhoaa sähköpostiviestin. Eleeseen perustuva palaute tästä havainnosta ei kuitenkaan kerro sitä, mihin viestin erityisiin ominaisuuksiin ohjelma kiinnittää



Kuva 18 Sähköposteja lajitteleva MAXIMS-agentti toiminnassa

huomiota tehdessään havainnon. Todellisuudessa MAXIMS käyttää päättelyssään tietoa lähettäjästä ja viestin otsikosta. Näitä yksityiskohtia käyttäjän voi olla tarjotun palautteen avulla mahdotonta päätellä. Eleisiin perustuva palaute voi siis kärsiä tarkkuuden puutteesta.

Ajallista tapahtumien ja palautteen välistä kytkentää voidaan käyttää myös ilman ilmeitä tai eleitä. Selection Recognition Agentin [Pandit and Kalbag 97] tunnistaa tietyssä formaatissa esitettävän tekstin tyypin, kuten esimerkiksi sähköposti- tai URL-osoitteen. Tunnistettua tekstiä voidaan käyttää hyödyksi vaikkapa sähköpostin lähettämiseen. Ohjelman käyttöliittymä koostuu yhdestä ikonista, jonka paikkaa ja kokoa voidaan kontrolloida.

Kun ohjelma tunnistaa tekstin, sen ikoni muuttuu ilmaisemaan tunnistettua tyyppiä. Tekstin valitsemisen, sen tunnistamisen ja palautteen antamisen välillä on siis ajallinen suhde. Koska ohjelman toiminta on melko yksinkertaista, voidaan aika- ja tyyppiinformaation uskoa välittävän ohjelman toimintaperiaatteita käyttäjälle kohtuullisen hyvin.

VAATIMUKSET KÄYTTÄJÄPROFIILIN TALLETUKSELLE

- sisällön paljastaminen
- päivitysperiaatteen julkistus
- julkistettavien tietojen hallinta

5.2.2 *Käyttäjäprofiilin talletus*

Se, että ohjelma ylipäätään tallentaa jotain käyttäjäkohtaista informaatiota, ei ole kovin tavallista yleisimmissä nykyaikaisissa sovelluksissa. Useissa ohjelmissa on ohjelmainstanssiin kohdistuvia asetuksia, joita voidaan joissain tapauksissa myös käsitellä käyttäjäkohtaisesti. Tämä ei ole kuitenkaan sama asia kuin käyttäjän toimintahistoria tai hänen mielenkiinnonkohteidensa tallentaminen. Ne ovat käyttäjäkohtaista tietoa, jota tarvitaan usein epäsuoraa hallintaa toteutettaessa.

Let's Browse [Lieberman 99] agenttiprototyypissä on esitetty eräs sinänsä melko ilmeinen, mutta ehkä juuri siksi varsin toimivan tuntuinen tapa esittää käyttäjäprofiilin tallentaminen. Ohjelma avustaa joukkoa ihmisiä tietoverkon selailussa etsien ja esittäen sopivia sivuja. Sivujen sopivuus määritellään kaikkien läsnäolevien käyttäjien profiilien perustella.

Kuva 19 Let's Browse -ohjelma näyttää käyttäjäprofiilin painotetut avainsanat

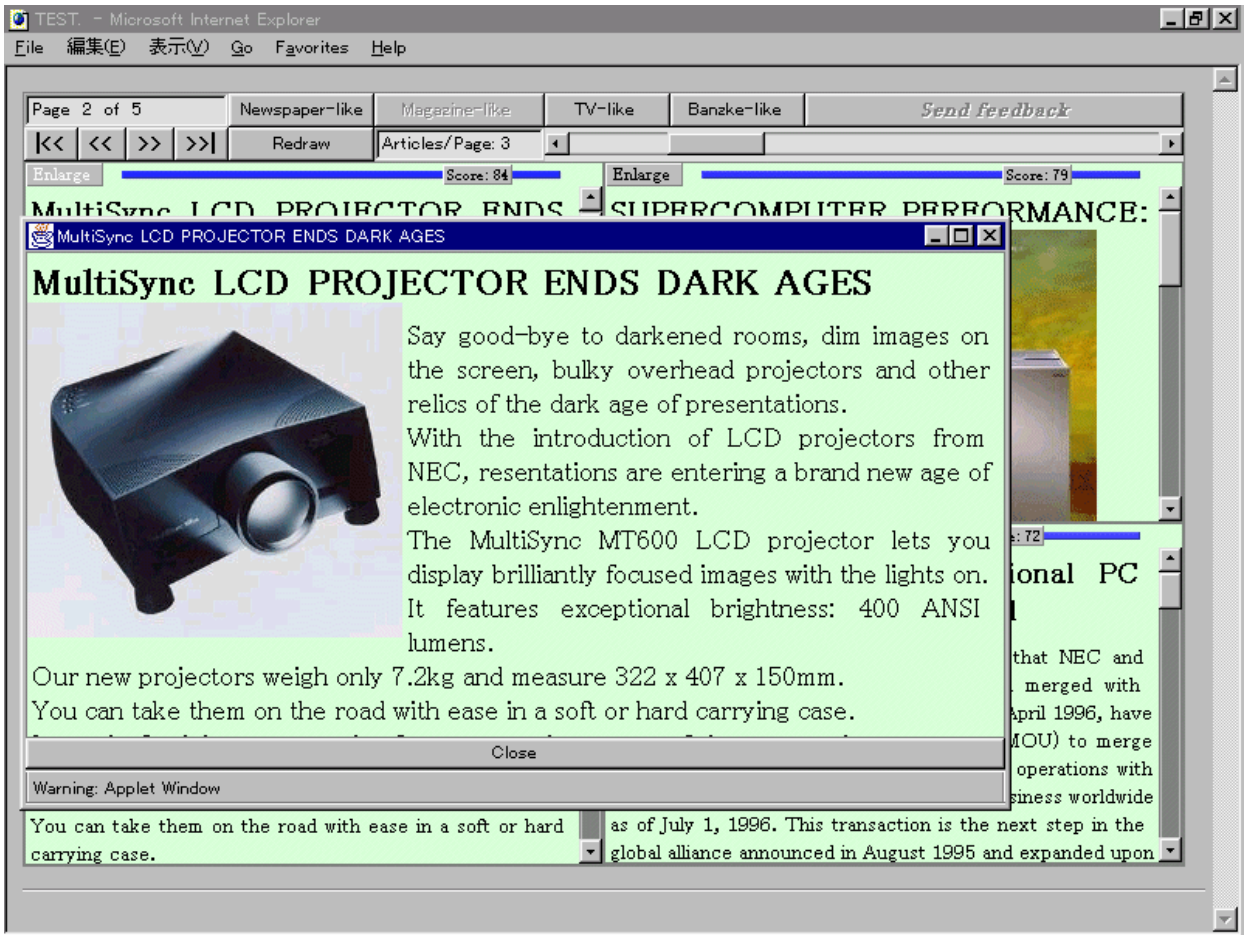
Let's Browse ohjelmassa käyttäjäprofiilit on toteutettu painotettuina avainsanavektoreina ja tämä myös näkyy käyttöliittymässä (Kuva 19). Ohjelma paljastaa käyttämänsä avainsanat painoineen käyttäjille ja pystyy myös perustelemaan esittämiään sivuvalintoja perustuen näihin käyttäjäprofiileihin. Ratkaisu julkistaa sekä käyttäjäprofiilin sisällön että sen toteutustekniikan. Ohjelman suorituksen aikana käyttäjäprofiileja ei päivitetä, joten tarvetta päivitysmekanismin esittämiseen ei ole. Käyttäjäprofiilit on alunperin koostettu käyttäjien omien kotisivujen perusteella, mikä myös käy ohjelman käyttöliittymästä ilmi.

Butterfly-ohjelmassa [Van Dyke *et al.* 99] käytetty menetelmä on Let's Browsessa esitellyn toinen muunnelma. Siinä missä Let's Browse pitää käyttäjäprofiilit näkyvillä aina, Butterfly tarjoaa näkymän niihin vain pyydettäessä. Muuten ratkaisut ovat oleellisesti samat, käyttäjäprofiilit koostuvat kummassakin painotetuista avainsanoista ja ne esitetään käyttäjälle oleellisesti samassa muodossa. Butterfly-ohjelmassa käyttäjäprofiili voi muuttua käytön aikana. Sen muuttaminen on kuitenkin käyttäjän vastuulla, joten tässäkään tapauksessa ei tarvitse selittää mahdollisesti monimutkaistakin päivitysmekanismia.

Useissa ohjelmissa on kuitenkin tyypillistä se, ettei käyttäjäprofiili ole eksplisiittisesti käyttäjälle näkyvissä. Tällaista ratkaisua vastaan on argumentoitu monesti [esim. Cook and Kay 94], mutta siitä huolimatta lähestymistapaa käytetään usein, koska monessa tapauksessa käyttäjäprofiilin esittäminen on vaikeaa. Joka tapauksessa käyttäjäprofiilin esittämisestä on lisävaivaa verrattuna siihen, että sen esittäminen jätetään kokonaan pois. Nämä ovat varmaankin tyypillisimpiä syitä siihen, miksi käyttäjäprofiilia ei aina esitetä käyttäjälle.

ANATAGONOMY on järjestelmä [Sakagami and Kamba 97], jossa käyttäjälle ei suoraan kerrota, että ohjelma sisältää käyttäjäprofiilin. ANATAGONOMY on sanomalehtiartikkeleiden lukuohjelma, joka valitsee ja järjestää artikkelit käyttäjän osoittamien mieltymysten perusteella. Esitettäväksi valitaan vain potentiaalisesti kiinnostavat artikkelit ja ne järjestetään (oletetun) kiinnostuksen mukaiseen järjestykseen. Käyttäjä näkee ensimmäisenä kiinnostavimmiksi arvioidut artikkelit.

Käyttöliittymässä on nk. arviointipalkki (*score bar*), jonka avulla ohjelma viestittää artikkelin arvioitua kiinnostavuutta ja toisaalta



Kuva 20 Käyttäjäprofiilin perusteella elektronista lehteä koostavan ANATOGONOMY-järjestelmän käyttöliittymä

antaa käyttäjälle mahdollisuuden muuttaa tätä arviota (Kuva 20). Muutokset talletetaan käyttäjäprofiiliin, mitä asiaa ei käyttäjälle erikseen kerrota. Arviointipalkki on siis näkymä käyttäjäprofiiliin ja sen olemassaolon indikaattori.

ANATAGONOMY-ohjelman toteuttama käyttäjäprofiilin käyttöliittymä on yksinkertainen toteuttaa, mutta sen riittävyydestä voidaan olla monta mieltä. Jos käyttäjä tuntee ohjelman toteutustekniikan entuudestaan ja tietää käyttäjäprofiilista, tällainen käyttöliittymä voi hyvinkin olla riittävä palaute sen tilasta – varsinkin, kun se mahdollistaa käyttäjäprofiilin muuttamisen tutulla interaktiotavalla. Jos kuitenkin ajatellaan käyttäjää, joka on ensimmäistä kertaa tekemisissä käyttäjäprofiilia keräävän ohjelman kanssa, voi hänen olla mahdotonta ymmärtää ohjelman toimintaa tällaisen palautteen perusteella. Ensinnäkin koko käyttäjäprofiilin käsite on vieras, jolloin on vaikea ymmärtää sen pysyvää luonnetta ja tarkoitusta. Merkityksen ymmärtämistä hankaloittaa erityisesti se, että käyttäjäprofiilia hyödynnetään vain uutisten valinnan yhteydessä, joten sen muuttamisella ei ole välittömiä vaikutuksia. Palautteessa merkittäväksi todettu ajallisesti välitön palaute [Shneiderman 95] toiminnosta jää saamatta.

VAATIMUKSET OPPIMISELLE

- muuttuvien toimintojen havainnollistus
- muutoksen ajankohdan julkistus
- muutoksen perusteiden julkistus
- muutoksesta päättäminen
- muutostiheyden säätäminen

5.2.3 Oppiminen

Oppimista on nykyisissä prototyypeissä käytetty lähinnä kolmenlaisten muutosten ohjaamiseen: 1) havaittavaan käyttöliittymään, 2) käyttöliittymän toimintoihin ja 3) ohjelman toiminta-algoritmiin kohdistuvat muutokset. Nämä erilaiset toiminnot asettavat erilaisia vaatimuksia niitä esittävälle ja hallitsevalle käyttöliittymälle. Vaatimukset implikoivat erilaisia käyttöliittymäratkaisuja.

UIDE on käyttöliittymien suunnitteluympäristö [Sukaviriya and Foley 93], joka mahdollistaa mukautuvien ominaisuuksien rakentamisen sillä suunniteltuihin käyttöliittymiin. UIDE-työkalulla tehtyjen käyttöliittymien muutokset voivat perustua epäsuoraan hallintaan, tarkemmin sanottuna käyttäjän suorittamien toimien tarkkailuun. Tarkkailun perusteella UIDE-työkalulla tehdyissä ohjelmissa voidaan muuttaa valikkojen ja dialogi-ikkunoiden ulkoasua, ehdottaa makrojen luomista sekä tarjota käyttäjän ja tilanteen perusteella mukautettua apua. Kaikki mukautuminen voidaan kytkeä pois päältä, joka on perustavin ja karkein tapa hallita toimintojen automaattista muuttamista UIDE-työkalulla tehdyissä ohjelmissa.

UIDE:lla toteutetuissa käyttöliittymissä käyttäjältä kysytään lupa muutoksen tekemiseen. Tämä julkistaa selvästi muutoksen ajankohdan ja mahdollistaa myös muutosehdotuksen perusteiden esittämisen käyttäjälle. Muutoksen valtuuttaminen käyttäjällä antaa hänelle hyvän kontrollin siitä, mitä käyttöliittymälle voi tehdä.

UIDE:ssa käytetty tapa kysyä lupaa muutokseen ei kuitenkaan ole täysin epäsuoran hallinnan ihanteiden mukaista. Kysymykseen vastaaminen on ylimääräinen tehtävä käyttäjälle, joka voi olla rasittava etenkin, jos muutos ajoittuu muutenkin käyttäjän kannalta kiireiseen hetkeen. Kompromissi on kuitenkin perusteltu, sillä käyttöliittymän muuttaminen käyttäjän tietämättä voisi aiheuttaa vielä enemmän ongelmia.

Myös muutoksen ajankohdan hallitsemiseen on UIDE:ssa otettu kantaa. Käyttäjän on mahdollista määritellä pienin mahdollinen aikaväli, joka kahden muutoksen väliin on jäätävä. Koska muutos vaikuttaa havaittavaan ja käyttäjän manipuloimaan käyttöliittymään, sen vaikutuksesta käyttäjän on mahdollisesti totuttava uudelleen toimintojen käyttöön. Siksi kontrollointikohde on erityisen tärkeä.

Split Menus järjestelmässä [Sears and Shneiderman 94] käyttäjän suorittamien valikkovalintojen tiheyttä käytetään syötteenä päätettäessä valikkojen järjestyksestä. Ajatuksena on nopeuttaa valinnan tekemistä nostamalla usein valitut kohteet valikon ylälaitaan. Suoritettujen testien perusteella kolmen useimmin käytetyn valinnan erottaminen omaksi ryhmäkseen vaikuttaisi toimivalta järjestelyltä. Tutkijat ehdottavat myös mahdollisuutta kontrolloida käytetäänkö tavallista vaiko valinnan tiheyteen perustuvaa tapaa järjestää valikkoalkiot.

Tutkimuksen perusteella ei kuitenkaan voida päätellä paljoakaan siitä, millaista palautetta valikon muuttamisesta pitäisi käyttäjälle antaa. Koe suoritettiin manuaalisesti siten, että ensin kerättiin tietoa käyttäjien käyttötavoista, jonka perusteella ohjelman valikot muutettiin. Uudet valikot asennettiin ja niiden käyttö tutkittiin. Käyttäjät saivat siis varsin konkreettista tietoa siitä, milloin valikot muuttuivat ja millä perusteella. Jos ohjelma olisi toiminut oikean skenaarion mukaan, tällainen palaute olisi jäänyt antamatta tai se

olisi annettu eri muodossa. Missä muodossa se olisi pitänyt antaa, jää tämän tutkimuksen valossa hämärän peittoon.

Useissa oppivissa ohjelmissa toimintojen muuttuminen ei tarkoita näin radikaaleja muutoksia ohjelman havaittavaan käyttöliittymään. Esimerkiksi MAXIMS-sähköpostin käsittelyagentti [Maes 94] oppii käyttäjän toimia seuraamalla hänen tapojaan ja muuttaa toimintaansa näiden havaintojen mukaan. Ohjelman havaittavaan käyttöliittymään ei tapahdu mitään muutoksia, mutta sen tuottamat tulokset voivat silti muuttua. Tällaisissa tilanteissa ei tyypillisesti ole kovin kriittistä viestittää käyttäjälle koska muutokset tapahtuvat ja millaisia ne täsmälleen ovat. Tärkeämpää voi olla viestittää millaisia havaintoja ohjelma kykenee tekemään.

MAXIMS-agentti viestittää oppimisprosessiaan käyttäjälle antropomorfisen käyttöliittymänsä avulla ilmein. Palautetta saadessaan se osoittaa joko tyytyväisyyttä tai hämmennystä sen mukaan, oliko se ennustanut käyttäjän reaktion oikein tai väärin. Palautteen perusteella ohjelma muuttaa toimintaansa, eli oppii. Oppimisesta annettu palaute tarkoittaa siis ohjelman toimintatavan muutoksesta viestittämistä. Kuten MAXIMS-agentinkin kohdalla, usein palaute on hyvin kevyttä, eikä kerro tarkemmin siitä *miten* oppiminen tapahtui.

Oppimisen kohdalla usein käytetään vieläkin vähäeleisempää palautetta ohjelman toimintatavan muutoksesta. Esimerkiksi Amalthaea-järjestelmässä [Moukas 96] oppimisesta ei reaaliaikaisesti viestitä mitenkään. Käyttäjän on kuitenkin mahdollista seurata oppimistapahtumaa eräänlaisen kontrollipaneelin avulla. Sen käyttämä termistö on kuitenkin hyvin teknistä eikä sovellu näin ollen loppukäyttäjälle.

5.2.4 Yhteisölliset ominaisuudet

VAATIMUKSET YHTEISÖLLISILLE TEKNIIKOILLE

- käyttöajan ja kohteen julkistaminen
- käytetyn yhteisön havainnollistus
- yhteisöllisen tiedon laadun julkistus
- lupa tekniikan käyttöön
- käytetyn yhteisön hallinta

Yhteisöllisiä tekniikoita käytetään korjaamaan muiden koneoppimistekniikoiden heikkouksia tai sinällään tuomaan lisäarvoa ohjelmaan. Yhteisölliset tekniikat perustuvat joidenkin käyttötietojen mallintamiseen ja eri käyttäjiltä saatavien mallien vertailuun. Näin löydetään mallinsa perusteella samanlaisia ihmisiä, joiden välillä voidaan vaihtaa mallien sisältämää tietoutta. Käyttäjän ja käyttöliittymän kannalta on tällöin oleellista, että käyttäjää kuvaavaa mallia siirretään, vertaillaan ja sen sisältämiä tietoja kopioidaan järjestelmän sisällä.

Firefly [Firefly 97] on internetissä oleva yhteisöllisiin tekniikoihin perustuva palvelu, joka suosittelee käyttäjilleen esimerkiksi kiinnostavia elokuvia tai saattaa yhteisiä kiinnostuksen kohteita omaavia ihmisiä yhteen. Kaikki Firefly'ssa tarjotut palvelut perustuvat yhteisölliseen suodattamiseen. Tekniikan mahdolliset käytettävyysongelmat ovat hyvin tiedostetut ja käyttöliittymässä asioihin on otettu selvästi kantaa. Käyttäjä voi nähdä oman profiilinsa ja muokata sitä mielensä mukaan. Hän voi esimerkiksi määritellä millaisia asioita hänestä voidaan julkistaa käyttäjän omalla "kotisivulla", joka on kaikille käyttäjille avoin.

Myös palvelun toimintaperiaatteista on saatavilla runsaasti tietoa. Tieto on tosin tekstimuodossa, mikä ei tue esimerkiksi tiedon nopeaa välittämistä sillä hetkellä, kun se on relevanttia. Näin ollen dynaamisen tiedon välittäminen järjestelmän tilasta tai toiminnoista voi olla vaikeaa tai mahdotonta. Firefly ei yritäkään välittää

tällaista tietoa vaan tyytyy kertomaan järjestelmän toimintaperiaatteet ohjekirjamaisessa muodossa. Konseptin voidaan kuitenkin olettaa olevan melko toimiva, sillä jo kymmenet yritykset ovat lisensoineet tekniikan käyttöönsä omille tietoverkkopalvelusivuilleen.

Myös Kasbah (myöhemmin MarketMaker) [Chavez and Maes 96] on internetissä toimiva palvelu, joka perustuu yhteisölliseen tekniikkaan. Palvelun avulla käyttäjät voivat ostaa ja myydä tavaroita puoliautomaattisesti. Kasbah etsii myyjiä ja ostajia, joiden intressit käyvät yksiin. Näin ostajien ja myyjien työläs etsintä jää pois käyttäjän tehtävistä. Käyttäjän tulee ainoastaan tehdä päätös kaupasta ja suorittaa se.

Kasbahin käyttöliittymän tarjoama tieto yhteisöllisten tekniikoiden käyttämisestä ja niiden vaikutuksista on toteutettu pitkälti Fireflyn mallin mukaisesti. Varmaankin osittain johtuen webtekniikan asettamista rajoituksista tieto annetaan tekstimuodossa. Järjestelmään rekisteröitymisen yhteydessä käyttäjälle kerrotaan käytetystä tekniikasta ja annettujen tietojen käyttötarkoituksesta. Käyttäjällä on mahdollisuus lukea teksti niin halutessaan myös myöhemmin.

5.2.5 Päättely

VAATIMUKSET PÄÄTTELYLLE

- toiminnon tarkoituksen havainnollistus
- päättelyajankohdan julkistus
- päättelyperusteiden julkistus

– tekniikan käytön hallinta

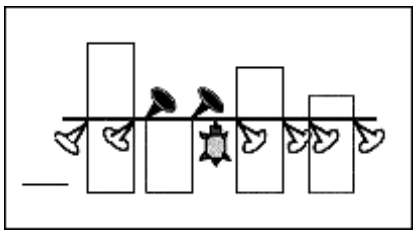
Päättelyä käytetään epäsuorasti hallitussa ohjelmassa aina tavalla tai toisella. Se voi olla yksinkertaisiin heuristisiin sääntöihin tai monimutkaiseen sääntökantaan perustuvaa. Päättelyllä voidaan nostaa implisiittisesti kerätyn syötteen abstraktiotasoa ja käyttää näin saatua tietoa vaikkapa esimerkein tapahtuvassa ohjelmoinnissa.

Päättely on kuitenkin vaikeasti havainnollistettava asia ja ehkä juuri siksi siitä on löydettävissä verraten vähän esimerkkejä. Vaikka päättely on inhimillisenä toimintana tuttua kaikille, on sille vaikea keksiä intuitiivista tapaa esittää sitä muussa kuin verbaalisessa muodossa. Esimerkiksi päättelyn esittämiseksi visuaalisessa muodossa ei ole helposti löydettävissä valmiita esikuvia. Verbaalinen esitystapa käyttöliittymissä ei kuitenkaan ole kovinkaan usein tyydyttävä, sillä se sitoo käyttäjän kognitiivista kapasiteettia monesti liiaksi. Varsinkin epäsuorasti hallitussa ohjelmassa ei ole suotavaa, että käyttäjä joutuu pohtimaan ohjelman palautetta liiaksi.

Metamouse-järjestelmään [Maulsby and Witten 93] on kuitenkin kehitetty eräs tapa esittää päättelyprosessiin vaikuttavia tekijöitä visuaalisesti. Metamouse on piirustusohjelma, jonka toimintoja on mahdollista automatisoida esimerkkiperustaisesti. Halutessaan automatisoida tietyn tehtäväsarjan, käyttäjä aktivoi opetustoiminnon ja esittää ohjelmalle esimerkillä halutun tehtävän. Ohjelma päättelee millaisen abstraktin toiminnon käyttäjä halusi suorittaa ja muodostaa tätä vastaavan toimintamallin itselleen. Tämän jälkeen toiminta voidaan suorittaa automaattisesti toisille objekteille.

Metamouse kiinnittää päättelyssä huomionsa toimintoihin, niiden toistoon ja piirrosobjektien kohdistamiseen toistensa suhteen. Se etsii kohtia, joissa piirrosobjektit koskettavat toisiaan kulmista, keskeltä tai päistä. Nämä muodostavat primitiivit, joita Metamouse kykenee havaitsemaan ja joiden perusteella se päättelynsä suorittaa. Päättelyn tuloksena saadaan toimintasarjoja, joiden tarvitsema paikkainformaatio perustuu piirrosobjektien keskinäisiin paikkoihin.

Metamouse esittää primitiivihavaintonsa kuvaruudulle piirrettyinä nupeina, jotka osoittavat piirrosobjektien yhdyskohdan, jolle ohjelma antaa erityisen merkityksen (Kuva 21). Esimerkiksi kahden objektin leikkauskohta on tällainen. Käyttäjä voi kontrolloida nupien avulla kohteita, joita päättelyssä käytetään. Kun käyttäjä osoittaa nupia, hän voi muuttaa sen huomioitavaksi tai hylättäväksi. Päättelyssä huomioidut nupit esitetään korostettuina.



Kuva 21 Metamouse-ohjelma esittää päättelyssä huomioidut asiat tummilla nupeilla

Vaikka Metamousen käyttöliittymä ei varsinaisesti esitäkään päättelyprosessia, se onnistuu antamaan päättelyn perusteista runsaasti tietoa. Käyttäjä tietää koska ohjelma päättelyä suorittaa ja mihin sen tuloksia käytetään. Kun tiedonvälitysmekanismi toimii samalla myös kanavana kontrolloida päättelyssä käytettyjä tietoja, voidaan käyttöliittymän todeta ratkaisseen useita ongelmia. Jos ohjelman käyttämä päättelyalgoritmi on kohtuullisen yksinkertainen ja käyttäjälle intuitiivinen, voi havaintojen esittäminen olla riittävä tieto ohjelman toiminnan ymmärtämiseksi.

Monissa muissa prototyypeissä on käytetty päättelyä, mutta siitä viestittäminen on niukempaa. Esimerkiksi Eager [Cypher 93] ei suoranaisesti kerro käyttäjälle millaisiin asioihin se päättelynsä perustaa vaan esittää päättelyään tulosten perusteella. Kun Eager havaitsee käyttäjän toimissa toistoa, se alkaa ennustamaan käyttäjän seuraavia toimenpiteitä perustuen toistettaviin toimintoihin. Ennustaminen tapahtuu korostamalla käyttöliittymän osia, joiden avulla toimintoja käynnistetään.

Päättelyn esittämisen kannalta Eagerissa ja Metamousessa on nähtävissä hyvin perustava ero. Metamouse esittää päättelyn perusteina käytettäviä havaintoja, kun Eager puolestaan havainnollistaa päättelyn tuotoksia. Molemmilla tiedoilla on tehtävänsä: edellinen mahdollistaa päättelyn perusteiden ymmärtämisen ja kontrolloinnin ja jälkimmäinen sen tulosten validoinnin. Myös tiedon välityksen ajankohta on näissä kahdessa tavassa oleellisesti erilainen: perusteiden havainnointi tapahtuu *ennen* päättelyä ja tulosten havainnollistus sen *jälkeen*. Tämän ajallisen erottelun havaitseminen auttaa ymmärtämään tekniikoiden mahdollisuuksia. Päättelyn jälkeen ei esimerkiksi voida enää vaikuttaa päättelyyn ja tällöin tulos on vain hyväksyttävä tai hylättävä. Ennen päättelyä tapahtuva tiedottaminen mahdollistaa myös toiminnon kontrolloinnin.

5.2.6 Tehtävien tunnistaminen

VAATIMUKSET TEHTÄVIEN TUNNISTAMISELLE

- käyttökohteiden julkistaminen
- tehtävän tunnistusajankohdan julkistaminen
- tehdyn tulkinnan julkistaminen

– lupa tehtävien tunnistamiseen

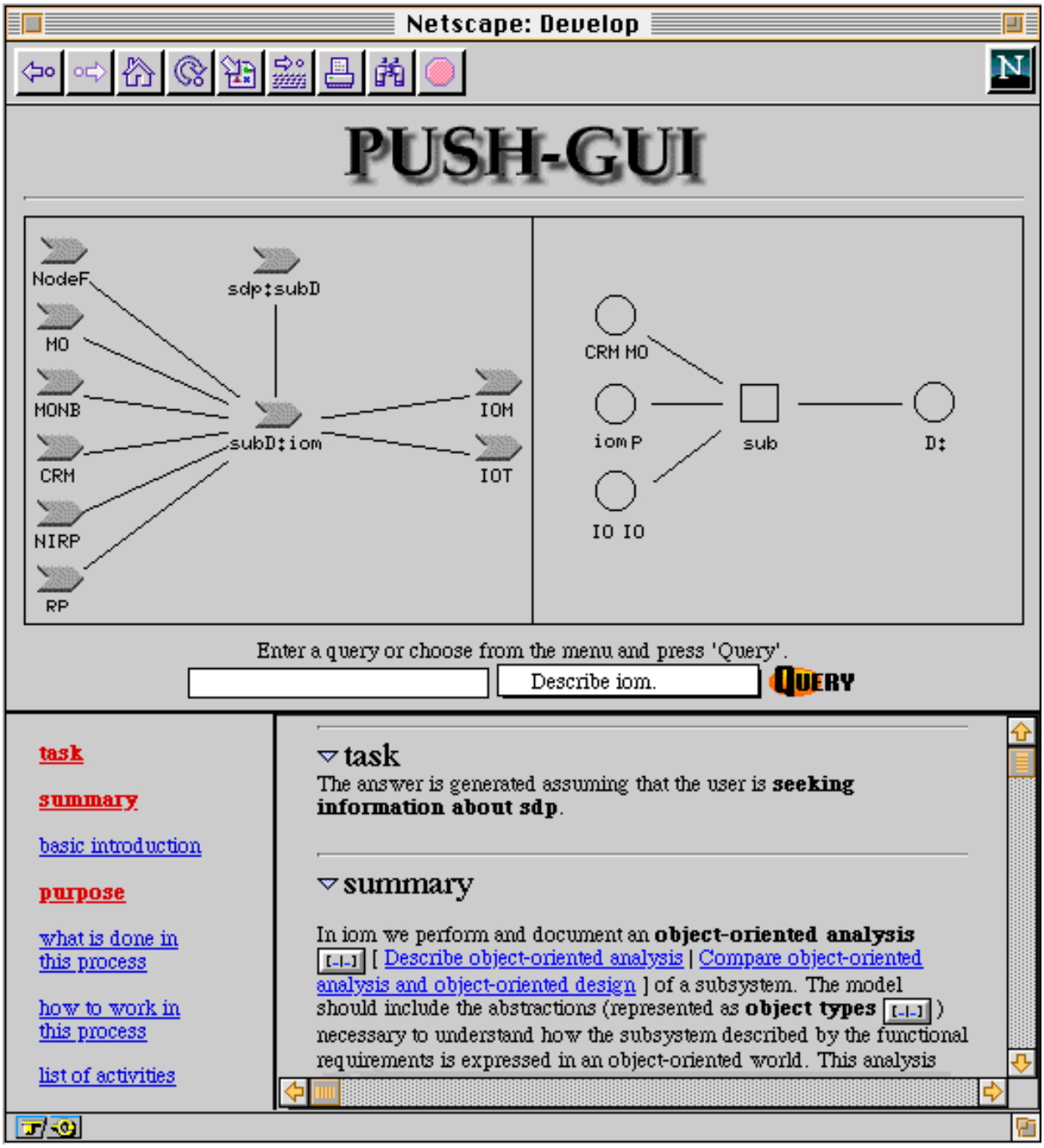
Tehtävien tunnistamisella käyttäjän toimet kyetään liittämään laajempaa yhteyteen. Laajemman yhteyden perusteella on käytettävissä enemmän tietoa käyttäjän käyttötilanteesta, mitä tietoa voidaan käyttää hyödyksi vaikkapa käytön opasteiden mukauttamisessa kulloiseenkin tilanteeseen sopiviksi.

PUSH-järjestelmä (Plan and User Sensitive Help) [Höök *et al.* 96] pyrkii mukauttamaan tarjottua informaatiota käyttäjän tehtävän perusteella. Järjestelmän prototyypissä mukauttaminen perustuu neljään stereotyyppitehtävään, jotka perustuvat yleisempään prototyypin sovellusaluetta kuvaavaan hierarkkiseen tehtävämalliin. Täydellisempi tehtävämalli on huomattavasti suurempi kuin käytettäväksi valittu neljän tehtävän osajoukko. Riittävän pieni joukko mukauttamiseen vaikuttavia tehtäviä auttaa käyttäjiä ymmärtämään tunnistetun tehtävän merkityksen ohjelman toiminnassa. Käyttäjien on myös helpompi hallita pientä määrää ohjelman toi-

mintaan vaikuttavia tehtäviä, jos heillä itsellään on mahdollisuus suoraan vaikuttaa niiden valintaan. PUSH-prototyypissä käyttäjä voi eksplisiittisesti määrittää tehtävän, jota hän on suorittamassa.

PUSH julkistaa tunnistetun tehtävän tekstinä (Kuva 22) ja samalla käyttäjälle välittyy epämäärästä tietoa myös siitä, milloin tunnistus suoritettiin. Koska PUSH on hypertekstiin perustuva sovellus, myös tehtävän tunnistustieto esitetään hypertekstinä, jossa käyttäjälle tarjotaan linkin välityksellä mahdollisuus muuttaa tunnistettua tehtävää.

Tekstimuotoinen esitystapa mahdollistaa tiedon selittämisen. PUSH käyttää tätä mahdollisuutta selvittääkseen sen, mihin tietoa käyt-

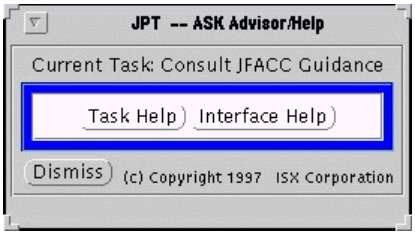


Kuva 22 PUSH-järjestelmän käyttöliittymä esittää tunnistetun tehtävän (task) tekstinä

täjän suorittamasta toimesta käytetään. Jos käyttäjä suorittaa ohjelmassa kyselyn, vastaus olettaa hänen suorittavan tiettyä tehtävää, mikä selitetään käyttäjälle. Tekstimuotoisella esitystavalla on paljon heikkouksia, jotka onkin jo aiemmin mainittu.

Sen lisäksi, että PUSH paljastaa tunnistetun tehtävän, se ei anna muuta palautetta toiminnosta. Esimerkiksi tunnistamisen perusteita tai sen tarkkaa ajankohtaa ei käyttäjälle kerrota. Ajallisen tiedon perustella käyttäjä voisi ymmärtää edellisten suorittamiensa toimintojen perusteella, mihin päättely perustuu.

ACPA-järjestelmä [Johnson *et al.* 99] on toinen esimerkkijärjestelmästä, joka käyttää tehtävien tunnistamista ohjelman toiminnan muokkaamisessa. Myös tässä tapauksessa tietoa hyödynnetään käyttäjän pyytämän tiedon valintaan, aivan kuten PUSH-järjestelmässä. ACPA ei anna käyttäjälle mitään tietoa siitä, että se tunnistaa käyttäjän tehtäviä. Vasta siinä vaiheessa, kun käyttäjä pyytää apua painamalla painiketta käyttöliittymässä, tunnistettu tehtävä paljastetaan hänelle sanallisessa muodossa (Kuva 23). Käyttäjän ei ole mahdollista muuttaa tunnistettua tehtävää, hänelle ainoastaan tarjotaan apua siihen nimenomaiseen tehtävään liittyen. Kuten niin monessa muussakin epäsuoraa hallintaa käyttävässä järjestelmässä tai prototyypissä, myös ACPA:ssa tutkimus on tuntunut keskittyvän enemmän toteutustekniikoihin kuin ohjelman käyttöliittymään.



Kuva 23 ACPA-järjestelmä kertoo tunnistetun tehtävän sanallisesti

5.2.7 Havainnollistustapojen suhde vaatimuksiin

Seuraava taulukko (Taulukko 3) esittää yhteenvedon edellä esitetyistä ratkaisuista, joilla epäsuoran hallinnan toteutuksessa käytettyjä tekniikoita voidaan havainnollistaa. Ratkaisuthan jaettiin toteutustekniikoittain kategorioihin aivan vaatimusten tapaan. Niinpä taulukon kiinnostavat solut muodostuvat vastaavien kategorioiden leikkauspisteisiin.

Ratkaisut on arvioitu heuristisella menetelmällä suhteessa vastaaviin vaatimuksiin. Arviointi on tehty yleisellä tasolla, ajatellen ratkaisun periaatteellista toimivuutta. Toisin sanoen arvioinnissa ei ole sitouduttu tiukasti mihinkään tiettyyn toteutukseen, vaan arvioinnin kohteena on ratkaisun periaatteellinen ydin. Esimerkiksi käyttäjän valtuutusta (oppiminen) ajatellessa ei kiinnitetä huomiota tapaan, jolla se toteutetaan (dialogi, tekstikehote, auditiivinen kysymys) vaan periaatteeseen siitä, että käyttäjältä kysytään kysymys.

Ratkaisujen periaatteiden toimivuus on arvioitu kolmiportaisella asteikolla heikosta hyvään. Tyhjä solu taulukossa tarkoittaa sitä, ettei ratkaisu ota vaatimukseen kantaa. Huomioitavaa asteikossa on sen ulottuminen negatiivisista arvioista positiivisiin, jossa heikko tarkoittaa epäilyttävää, hyvä suositeltavaa ratkaisua.

Kiinnostavimmat löydökset tehdään riveiltä tai sarakkeilta, joilla on ainoastaan heikkoja tai hyviä arvioita. Taulukon riveiltä voidaan nähdä miten hyviä ratkaisuja eri vaatimuksiin on esitetty. Esimerkiksi implisiittisen syötteen käyttötarkoituksen ilmaisemiseen tai sen keräämisen kontrollointiin ei prototyypeissä ole kyetty esittämään tyydyttävää ratkaisua. Toisaalta tietoa oppimisen myötä muuttuvista toiminnoista voidaan kohtuullisen hyvin toteuttaa kaikilla esitetyillä tavoilla. Eri asia toki on se, kuinka useissa ohjelmissa näitä mahdollisuuksia sitten käytetään.

Taulukko 3 Ratkaisujen arviointi epäsuoran hallinnan toteutustekniikoiden käyttöliittymävaatimuksia vastaan

· Heikko
▪ Kohtalainen
■ Hyvä

VAATIMUKSET \ RATKAISUT	Implisiittinen syöte	Tekstiraportit	Toiminnan animointi	Tilan graafinen näyttö	Käyttäjäprofiili	Profiilin tekstimuotoinen näyttö	Profiilin graafinen näyttö	Objektin arviointi mahdollisuus	Oppiminen	Käyttäjän valtuutus	Muutosvälin määrittely	Prosessin animointi	Yhteisölliset tekniikat	Tekniikan kuvaus ohjeissa	Päättely	Objektien korostus	Tuloksen havainnollistus	Tehtävien tunnistus	Tunnistettu tehtävä tekstinä
Implisiittinen syöte																			
Havaittavuus		·	■	▪															
Ajankohta		▪	▪	■															
Käyttötarkoitus		·	·	·															
Keräyksen kontrolli		·	·	·															
Käyttäjäprofiili																			
Tieto sisällöstä						▪	■	·											
Tieto päivitysperiaatteesta						▪	▪	·											
Julkistamisen kontrolli						▪	▪	·											
Oppiminen																			
Tieto muuttuvista toiminnoista										■	■	■							
Muutoksen ajankohta										■	■	▪							
Perusteiden ymmärtäminen										■	·	▪							
Muutoksesta päättäminen										■	▪	·							
Muutostiheyden säätäminen										■	■	·							
Yhteisölliset tekniikat																			
Ajankohta														·					
Käyttötarkoitus														■					
Yhteisö														■					
Kontrollin mahdollisuus														·					
Päättely																			
Käyttötarkoitus																▪	▪		
Ajankohta																■	■		
Perusteiden ymmärtäminen																▪	·		
Kontrollin mahdollisuus																■	▪		
Tehtävien tunnistus																			
Tunnistetut tehtävät																			■
Ajankohta																			■
Tieto tunnistustuloksesta																			■
Kontrollin mahdollisuus																			■
Yleiset tavoitteet																			
Ei tarvetta opetteluun		■	■	▪		■	▪	·		▪	▪	■		▪		▪	▪		■
Käyttö huomaamatonta		·	·	▪		·	■	■		·	▪	▪		·		·	▪		▪
Käyttö vaivatonta		·	▪	▪		·	■	▪		·	▪	▪		▪		▪	■		▪

Jos tarkastelu suoritetaan sarakkeittain, voidaan löytää enemmän tai vähemmän onnistuneita ratkaisuja. Varsinkin jos ratkaisun onnistumista tarkastellaan ainoastaan tekniikan havainnollistamiselle asetettujen vaatimusten valossa, esimerkiksi käyttäjän valtuutus oppimalla tehtävien muutosten yhteydessä vaikuttaa hienolta aja-

tukselta. Se näyttäisi vastaavan kaikkiin vaatimuksiin hyvin arvosanoin. Arvioinnin syventämiseksi taulukkoon onkin liitetty myös pieni yhteenveto epäsuoralle hallinnalle asetetuista yleisistä tavoitteista. Ottamalla myös ne huomioon tarkastelussa, johtopäätös ei ole enää kristallin kirkas. Sama pätee myös tunnistetun tehtävän esittämiseen tekstimuodossa.

5.3 Epäsuorasti hallitun toiminnallisuuden esittäminen

Edellä on esitelty lukuisia esimerkkejä siitä, miten epäsuoraa hallintaa toteuttavia tekniikoita on havainnollistettu käyttöliittymässä. Näiden havainnollistustapojen kautta on mahdollista antaa käyttäjälle tietoa ohjelman toimintaperiaatteista, joiden perusteella käyttäjällä on ainakin teoreettinen mahdollisuus muodostaa mielekäs mielikuvamalli ohjelman toiminnasta. Tällaisen mallin perusteella ohjelman toimintaa voidaan ennustaa, jolloin se muuttuu ymmärrettäväksi.

Näkökulma on kuitenkin ollut eräässä mielessä rajoittunut, eikä se näin ollen ole riittävä antamaan tyydyttävää kuvaa koko ilmiöstä. Tähän mennessä metsää on tutkittu tarkastelemalla yksittäisiä puita ja sanomalla jotain niiden suhteista. Nyt on kuitenkin aika ottaa hieman laajempi näkökulma epäsuorasti hallittujen ohjelmien toiminnallisuuden esittämiseen; on tarkasteltava metsää kokonaisuutena.

Epäsuoran hallinnan mahdollistavat tekniikat luovat perustan toteuttaa epäsuora hallinta käyttäjää hyödyttävää toimintoa ohjaamaan. Seuraavassa tarkastelussa käyttäjää hyödyttävä toiminto on huomion keskipisteessä. Käyttöliittymistä tarkastellaan sitä, miten ne kertovat toiminnan tarkoituksen käyttäjälle ja miten ohjelman eri osat tehtävää täydentävät. Lisäksi puututaan koko toimintaan vaikuttaviin asetusten määrittelemiseen tarkoitettujen käyttöliittymien toteuttamiseen.

5.3.1 Olemassaolo

Epäsuorasti hallitun toiminnan ollessa kyseessä edes sen olemassaolo ei ole itsestään selvä asia käyttäjälle. Koska ohjelma pyrkii käyttäjän kognitiivisen kapasiteetin minimaaliseen rasittamiseen, epäsuorasti hallittu toiminnallisuus usein piilotetaan käyttäjän näkyviltä. Toinen vaikuttava asia on se, ettei epäsuorasti hallitun toiminnon kanssa olla suorassa vuorovaikutuksessa. Ohjelman manipuloiminen selvästi edellyttää tietoa sen olemassaolosta.

Tyypillinen tapa ilmaista epäsuorasti hallitun toiminnallisuuden olemassaoloa on esittää se pienenä itsenäisenä ikkunana käyttöliittymässä. Esimerkiksi Eager, Selection Recognition Agent ja MAXIMS käyttävät tätä tekniikkaa. Eager poikkeaa linjasta sikäli, että se tuo ikoninsa näkyviin vasta, kun ohjelma voi tehdä jotain käyttäjän hyväksi. Kahdessa muussa ikkuna on aina näkösällä.

Jatkuvasti näkyvä indikaattori ohjelman läsnäolosta toiminee paremmin silloin, kun epäsuora hallinta on käyttäjälle vieras vuorovaikutustekniikka. Ainoastaan toisinaan esille ponnahtava ikkuna voi aiheuttaa kummastusta, jos syy sen ilmestymiseen ei tunnu selvältä. Toisaalta vain tarvittaessa näkyvä ikoni ei vie käyttäjän huomiota tarpeettomasti. Kun kuva sitten tuodaan näkyviin, tun-

tuu ohjelmassa todella jotain tapahtuvan. Käyttäjä huomaa varmemmin ohjelman ehdottaman toiminnon.

Jos epäsuorasti hallitun toiminnallisuuden olemassaolosta ei kerrota itsenäisellä ikkunalla, turvaudutaan usein proaktiivisesti esitettyihin viesteihin. Esimerkiksi UIDE-työkalulla toteutetut järjestelmät esittävät käyttäjälle dialogi-ikkunan, jolla kysytään hyväksyntää ehdotetulle toiminnolle. Samoin Butterfly-ohjelma esittää tuloksiaan proaktiivisesti osana tietoverkossa käytävää keskustelua. Menetelmät ovat pitkälti samoja kuin Eagerin käyttämät, mutta lähemmin ohjelmaan liittyvä esitystapa (jollainen esimerkiksi UIDE:n käyttämä tavallinen dialogi on) ei erota toiminnon erityisluonnetta kovin selvästi.

Kolmas melko yleinen menetelmä on piilottaa epäsuorasti hallittu toiminto käyttäjältä. Esimerkiksi ANATAGONOMY-palvelussa käyttäjä näkee ainoastaan epäsuorasti hallitun toiminnallisuuden tulokset. Tulokset voivat olla sen tyyppisiä, ettei käyttäjällä ole syytä olettaa hallitsevansa toimintoa epäsuorasti. ANATAGONOMY-järjestelmän toteuttama sanomalehtiartikkeleiden valitseminen ja järjestäminen on tyypillinen tällainen toiminto, jossa epäsuorasti hallitut toiminnot eivät ole mitenkään ilmeisiä. Sanomalehtihän on voitu koota jo toimituksessa käyttäjän näkemällä tavalla. Jos epäsuorasti hallittu toiminto toimii moitteetta ja tuottaa käyttäjän kannalta hyviä tuloksia, tällä ei ole merkitystä. Jos käytössä on ongelmia, eikä epäsuora hallinta tuota parhaita mahdollisia tuloksia, käyttäjän on lähes mahdotonta korjata vikaa edes manuaalisilla toimillaan.

5.3.2 Ohjelman toiminta

Hyvin oleellista ohjelman hyödyllisyyden kannalta on myös se, ymmärtääkö käyttäjä mihin epäsuorasti hallittu ohjelma todellisuudessa kykenee – toisin sanoen, millaisia tuloksia siltä on lupa odottaa. Eräs tärkeä osa ymmärrystä on tieto siitä, milloin ohjelmalta voi tuloksia odottaa. Sen tietäminen on merkki siitä, että käyttäjä pystyy ennustamaan ohjelman toimintaa ja ymmärtää sen toimintatapoja riittävällä tarkkuudella.

Ohjelman tuotosten ja siis sen merkityksen ymmärtäminen on voimakkaasti sidoksissa tietoon sen olemassaolosta. Lienee itsestään selvää, ettei ohjelman merkitystä voi ymmärtää, jos sen olemassaolosta ei ole tietoa. Asiat liittyvät kuitenkin myös toisella tavalla toisiinsa. Käytetyt menetelmät asioiden viestittämiseksi käyttöliittymässä käyttäjälle ovat nimittäin tyypillisesti samat. Viestimällä ohjelman toimintaa ja sen tuloksia viestitään samalla myös sen olemassaolosta, kuten jo edellisessä alakohdassakin nähtiin.

Epäsuorasti hallittua toiminnallisuutta ja sen tuotoksia on tähän mennessä rakennetuissa prototyypeissä havainnollistettu lähinnä kolmella eri tavalla: 1) piilottamalla ohjelman toiminta ja esittämällä ainoastaan tulokset käyttäjälle, 2) esittämällä tuloksiin johtavia toimintaehdotuksia ja kysymällä niille suorituslupaa ja 3) näyttämällä tuloksiin johtavien toimintojen reaaliaikainen suorittaminen käyttäjälle.

Toiminnon tai toimintojen piilottamisella pyritään vähentämään käyttäjän kognitiivista kuormaa. Jos käyttäjä ei tiedä tai ei näe toiminnon olemassaoloa, ei hänen myöskään tarvitse kiinnittää siihen

huomiotaan. Kaikki ohjelman tulokset tulevat käyttäjälle ikään kuin ilmaiseksi. Tämä on samalla eräänlainen puolustus sille, jos ohjelman kaikki tulokset eivät ole käyttäjälle hyödyllisiä; niiden saamiseksi ei myöskään ole tarvinnut tehdä mitään. Vaikka tämä pitääkin paikkaansa, samalla voi myös potentiaalisesti arvokkaidenkin tulosten hyödyntäminen jäädä puolitiehen. Tulosten merkityksen arvioiminen voi olla vaikeaa, jos niiden luontiprosessi on epäselvä. Tästä hyvä esimerkki ovat erilaiset hakuagentit, joiden tulosten arvioimiseksi on tärkeää tietää mistä ja mitä varten ne tietoa hakivat.

Toimintaehdotusten tekeminen tarjoaa astetta runsaammat mahdollisuudet kertoa ohjelman toiminnasta käyttäjälle. Ehdotuksen tekeminen on selkeä paikka myös toiminnon perustelujen esittämiselle, mitä esimerkiksi UIDE-järjestelmässä on hyödynnetty. Näin ohjelman tulosten merkitys ja niiden valmistumisajankohta tulee käyttäjälle selviksi. Käyttäjän on mahdollista karkealla tasolla tietää mitä ohjelma tekee tietyllä hetkellä. Ratkaisun ongelmana on ohjelman autonomisuuden väheneminen.

Käyttäjän häirintä on vaarana kolmannessa tavassa esittää ohjelman toimintaa käyttäjälle. Tulosten laatiminen voi olla monimutkainen tehtävä, joka koostuu useasta osatehtävästä. Jos nämä kaikki esitetään käyttöliittymässä reaaliaikaisesti animoiden, on vaarana käyttöliittymän muuttuminen levottomaksi. Käyttäjän huomio on vaarassa kiinnittyä epäsuorasti hallitun autonomisen ohjelman aiheuttamaan liikkeeseen näyttöruudulla.

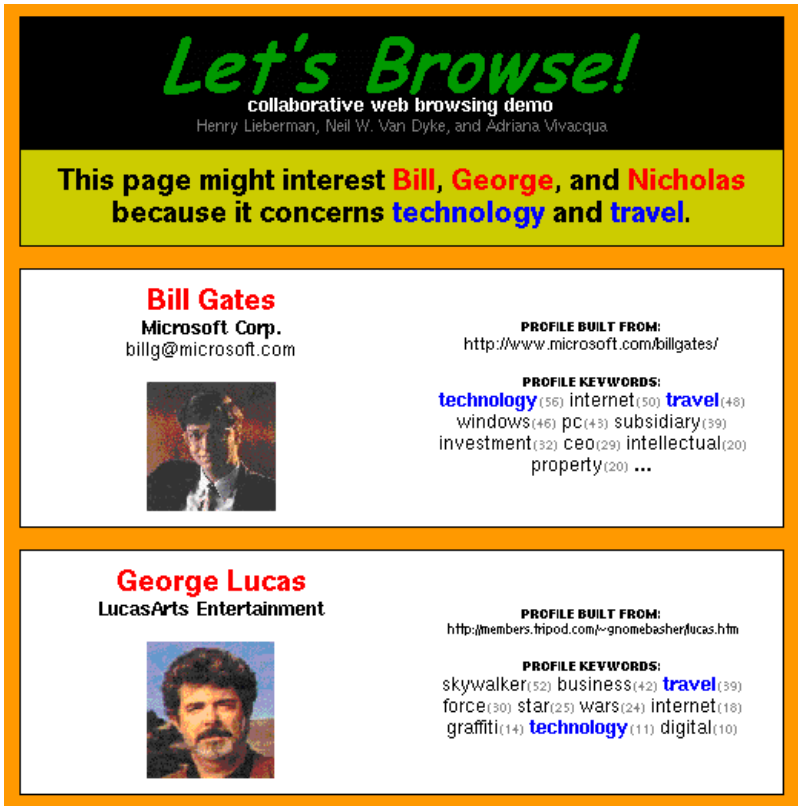
Letizia-agentti on eräs esimerkki ohjelman reaaliaikaisesta esittämisestä käyttöliittymässä. Se etsii ehdotuksia lataamalla omaan selainikkunaansa käyttäjän tarkastelemalta sivulta löytyvien linkkien kohteita. Letizian toiminta näkyy siis käyttöliittymässä samanlaisena kuin käyttäjänkin toiminta. Letizian selainikkunoiden päivitys aiheuttaa käyttöliittymään paljon käyttäjästä riippumatonta liikettä, mikä voi tehdä siitä levottoman tuntuisen. Lieberman on huomannut tämän, mutta argumentoi sen puolesta, että osalle käyttäjistä tämä ei tuota ongelmia [Lieberman 97].

5.3.3 Tulosten esittäminen

Jos käyttäjä ymmärtää miten epäsuorasti hallittu ohjelma toimii, on hänellä hyvät edellytykset myös ymmärtää sen tuottamien tulosten hyödyllisyyttä ja merkitystä. Aina ei ohjelman toiminta kuitenkaan riitä kertomaan kaikkea tarpeellista, jotta jokaisen yksittäisen tuloksen hyödyllisyys olisi arvioitavissa. Siksi myös tulosten esitysmuotoon on kiinnitettävä huomiota.

Tyypillinen tapaus, jossa ohjelman tuloksen hyödyllisyyttä on kyettävä arvioimaan, on hakuagenttien toiminta. Koska käytetty tietolähde ei aina vastaa täydellisesti haun tarkoitusta, saatuja tuloksia voidaan arvioida eli pisteyttää käyttäjäprofiilin tai muun vastaavan tiedon perusteella. Näin on mahdollista esimerkiksi tulosten järjestäminen tai numerotiedon tarjoaminen tuloksen hyödyllisyydestä. Tällaista tekniikka hyödyntää esimerkiksi NewT-hakuagentti. Se esittää yksittäisen tuloksen pisteet eli vastaavuuden hakukriteeriin graafisella pylväskuvaajalla.

Let's Browse -ohjelman käyttämä tekniikka on hieman toinen ja se on yhteydessä myös ohjelman toimintaperiaatteen esittämiseen. Jokainen tulos perustellaan esittämällä mitä avainsanoja tulos sisälsi verrattuna käyttäjiä kuvaaviin avainsanavektoreihin (Kuva 24). Tällä tavoin käyttäjät voivat arvioida tulosdokumenttia tästä näkökulmasta. Esimerkiksi sekä käyttäjiä että tulosdokumenttia kuvaaviin avainsanoihin voisi kuulua termi 'matkustaa', jonka kertominen riittää suuntaamaan käyttäjien huomiota tuloksen arvioinnissa. Jos tällaista vihjettä tuloksen hyödyllisyydestä ei olisi esitetty, tuloksen merkitys jäisi enemmänkin arvailujen varaan ja esimerkiksi täydellisesti epäonnistuneen ehdotuksen tunnistaminen olisi vaikeaa.



Kuva 24 Let's Browse -ohjelma selittää tuloksensa korostamalla oleellisia kohtia käyttäjäprofiilista

Eagerin yhteydessä tilanne on toinen, koska ohjelman tulokset ovat luonteeltaan erilaisia verrattuna hakutuloksiin. Eagerin tuloshan on ohjelma ja sen hyödyllisyyden varmistamiseksi on ymmärrettävä mitä ohjelma tekee. Eager käyttämä tapa korostaa generoidun ohjelman toimintoja käyttöliittymässä on melko nerokas. Uusia kommunikointikanavia tai -tapoja ei tarvita ja standardoidun värin ja ikonin ansiosta käyttäjän on mahdollista yhdistää koko ohjelman aktiivisuudesta kertova ikoni ehdotettuihin toimintoihin. Yleisemmin puhuen Eagerin käyttämä tekniikka on suoritettavien toimintojen luetteleminen.

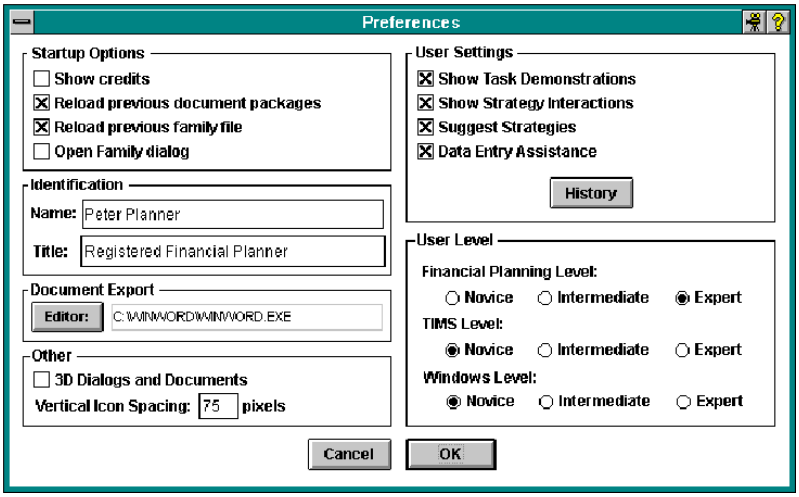
Edellisissä esimerkeissä tuli esille kaksi erilaista tapaa organisoida tulosten esittämistä. Kutsun tapoja tulosvirraksi ja kerääntyväksi listaksi. Kun tulokset esitetään tulosvirtana, ne annetaan käytettä-

väksi heti, kun tulos on tuotettu. Ohjelmissa, missä tuloksen hyödyllisyys on sidottu määrättyyn käyttötilanteeseen (esim. Eager), tällainen tulosten esitystapa on välttämätön. Kerääntyvässä listassa (esim. NewT) käyttäjällä on enemmän päätäntävaltaa tulosten hyödyntämisajankohdan suhteen. Tässä tekniikassa ohjelma kerää tuloksensa listaan, joka talletetaan. Tulokset ovat hyödynnettävissä koska tahansa niiden tuottamisen jälkeen, mutta ohjelma ei aktiivisesti tuo niitä esille.

5.3.4 Konfigurointi ja kontrollointi

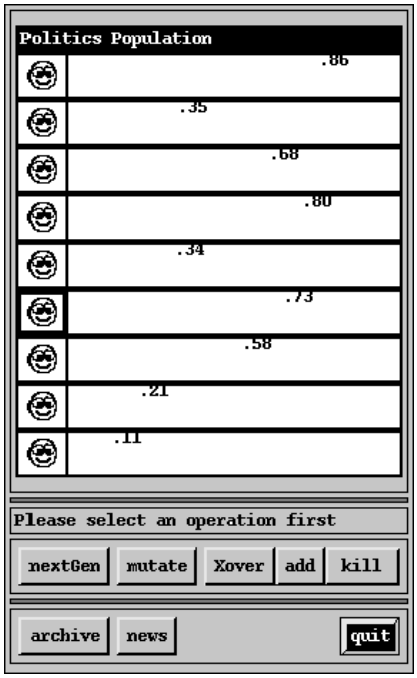
Vaikka epäsuorasti hallittua toiminnallisuutta ja sen toteutusta voidaan periaatteessa konfiguroida monin tavoin, ei monessakaan prototyypissä ole annettu käyttäjälle mahdollisuutta tähän. Tyypillisimmin käyttäjälle on annettu välineet tutkia ja muuttaa itseään koskevaa käyttäjäprofiilia.

TIMS-järjestelmä [Strachan et al. 97] on verotuksen ja talouden suunnitteluun käytettävä työkalu. Siihen on lisätty erityisiä avustajia (*assistant*), joiden tehtävänä on ohjata käyttäjää järjestelmän eri osien hyödyntämisessä. Käyttäjälle annettu avustus riippuu hänen käyttäjäprofiilistaan. TIMS-ohjelmassa käytetty käyttäjäprofiili on hyvin yksinkertainen, joten sen visualisointikin on ollut kohtuullisen helppoa. Tarkoitukseen on käytetty tavallisia graafisia käyttöliittymäkomponentteja (Kuva 25). Käyttäjä voi manipuloida ja nähdä profiilin tilan valintapainikkeina.



Kuva 25 TIMS-järjestelmän konfigurointi perustuu käyttäjäprofiilin muokkaamiseen

NewT-ohjelmassa käyttäjän vaikutusmahdollisuudet ovat jonkin verran runsaammat ja voidaan puhua toimintaperiaatteiden muuttamismahdollisuudesta. NewT toimii geneettisellä algoritmilla ja se antaa käyttäjälle mahdollisuuden manipuloida suodatuksen toteuttavaa populaatiota varsin monipuolisesti. Käyttäjän on mahdollista tarkastella populaation yksittäistä jäsentä ja muuttaa sen tilaa. Populaatiosta voidaan poistaa jäseniä tai niitä voidaan lisätä. Tämän lisäksi populaation uusiutumista voidaan kontrolloida.



Kuva 26 NewT-ohjelman konfigurointikäyttöliittymä

NewTn konfigurointikäyttöliittymä on toteutettu melko teknisellä sanastolla, mikä vaikeuttaa toimintojen ymmärtämistä sovellusalaa tuntemattomien keskuudessa. Esimerkiksi populaation elinvoimaisuutta kuvataan lukuarvolla, jonka merkityksen ymmärtäminen voi olla vaikeaa. Toisaalta toimintoja tarvitaan vain harvoin ja sikäli niiden käytettävyys ei ole kovin keskeinen tekijä koko sovelluksen käytettävyyden kannalta.

Konfigurointiin kuuluu myös ohjelman tuloksien esitysmuodon hallinta. Etenkin jos epäsuorasti hallittu toiminto esittää tuloksensa tulosvirtana, voi olla oleellista päästä vaikuttamaan millaisessa muodossa virta esitetään. Hyvin yksinkertainen ratkaisu on esitelty Remembrance Agentissa, jossa käyttäjä voi määritellä kuinka monta ehdotusta agentin tuloksista on kullakin hetkellä näkyvissä.

Epäsuorasti hallitun ohjelman kontrollointi on läheisessä suhteessa sen konfigurointiin. Näitä kahta asiaa erottaa niiden tarkoittaman toiminnon ajallinen kesto. Esitettyjen prototyyppien kontrollointitapoja on jo edellä käsitelty eri yhteyksissä melko monipuolisesti. Ohjelman kontrollointi tapahtuu tyypillisesti muiden käyttöliittymätoimintojen yhteydessä, joten niin ohjelman toimintaperiaatteiden kuin sen tulostenkin havainnollistamisen yhteydessä on puututtu myös ohjelman kontrollointiin.

Tärkeimmät kontrollointitavat ovat epäsuorasti hallitun toiminnon pysäyttäminen tai sen suorituksen estäminen. Edellinen voidaan toteuttaa Selection Recognition Agentin tapaan sulkemalla koko ohjelma ja jälkimmäinen MAXIMS-agentin tapaan pyytämällä käyttäjältä hyväksyntä toiminnon suorittamiseen.

5.3.5 Toiminnallisuuden esitystapojen suhde vaatimuksiin

Oheinen taulukko (Taulukko 4) kokoaa käyttöliittymän eri osille asetettujen vaatimusten ratkaisumallit yhteen. Taulukkoa luetaan samoin kuin aiemmin esitettyä esitystä tekniikoiden havainnollistusvaatimuksista ja -ratkaisuista. Ratkaisuja on siis arvioitu heuristisella menetelmällä suhteessa vaatimuksiin. Tulokset on esitetty kolmiportaisella asteikolla.

Koska käyttöliittymän toiminnallisuudelle asetetut tavoitteet ovat luonteeltaan yleisluontoisia, niiden toteutumista voidaan monin paikoin arvioida melko laajasti. Esimerkiksi vaatimuksiin tiedoista ohjelman toiminnoista ja olemassaolosta vastaa kaikki käsitellyt esitystavat. Osa vaatimuskategorioista on suppeita, joihin ratkaisuja ei löydy kategorian ulkopuolelta.

Muutamia keskeisimpiä löydöksiä voidaan taulukosta nostaa esiin. Tulosten esittämiseen turvautuminen ohjelman toiminnallisuudesta kertomiseksi ei vaikuta toimivalta ratkaisulta. Se ei anna käyttäjälle juurikaan tietoa ohjelman toimintaperiaatteista, mutta se on kuitenkin monesti häiritsevää. Kontrolloinnin mahdollisuudetkin ovat melko huonot. Saman suuntaiset johtopäätökset pätevät myös toimintaehdotusten tekemiseen.

Onnistuneista ratkaisuista voidaan mainita talletettu tuloslista. Se on monessa suhteessa toimiva tapa esittää epäsuorasti hallitun toiminnon tuottamia tuloksia. Onnistuneella linjalla on myös kontrollointi tavoista koko toiminnon käynnistäminen / sammuttaminen. Se on yksinkertainen ja tehokas tapa kontrollin mahdollistamiseksi, joskaan ei kovin monipuolinen.

Taulukko 4 Ratkaisujen arviointi toiminnallisuuden havainnollistamisen vaatimuksia vastaan

· Heikko
▪ Kohtalainen
■ Hyvä

VAATIMUKSET \ RATKAISUT	Olemassaolo	Itsenäinen ikkuna	Tarvittaessa näkyvä	Piilotettu	Ohjelman toiminta	Tulosten esittäminen	Toimintaehdotukset	Toiminnan animointi	Tulosten esittäminen	Hakuperusteiden korostaminen	Suoritettavien toim. esittäminen	Tulosten pisteytys	Tulosvirta	Talletettu tuloslista	Kontrolli	Ohjelman käynnistys/sammutus	Varmistusdialogi	Konfigurointi	Profiilin muuttaminen	Toim. algoritmin muuttaminen	Tulosten esitysmuodon valinta
Tieto toiminnoista ja olemassaolosta																					
Tietoisuus olemassaolosta		■	▪	·		▪	·	■		·	▪	·	■	▪		▪	▪		·	·	·
Häiritsemättömyys		▪	■	■		·	·	·		·	▪	■	·	■		■	·		■	■	■
Toimintaperiaatteiden selvennys		▪	·	·		·	▪	■		■	■	▪	▪	·		·	■		■	■	▪
Tulokset																					
Häiritsemättömyys										·	▪	■	·	■							
Hyödyllisyyden arviointi										■	■	■	▪	■							
Hyödyntämisen helppous													■	■							
Kontrolli																					
Nopeus ja helppous		■	·	·		·	·	▪								■	■				
Kompaktius		▪	■	■		▪	▪	·								■	▪				
Ei pysyvää liikettä		■	■	■		·	·	·								■	▪				
Konfigurointi																					
Nopeasti opittava																			▪	·	■
Helposti muistettava																			▪	·	■
Palautteen antavuus																			▪	▪	▪
Kontrollin mahdollisuus																			·	■	▪

5.4 Ratkaisujen käyttö

Kirjallisuudessa esiteltyjen käyttöliittymäratkaisujen hallitsemiseksi ratkaisuja on edellisessä kategorisoitu. Aluksi kartoitettiin epäsuoraan hallintaan liittyvien käyttöliittymien organisointitapoja, joiksi tunnistettiin käyttöliittymän toteuttaminen osana isäntäohjelmaa tai itsenäisenä ohjelmana. Toteutustavat saatetaan jakaa myös kahteen ryhmään: antropomorfiset ja perinteiset konemaiset toteutustavat.

Konkreettiset ratkaisut jaetaan vaatimuskategorioiden mukaan, jolloin niiden arviointi vaatimuksia vasten helpottui. Ensimmäinen kategoria on epäsuoran hallinnan toteuttavien tekniikoiden havainnollistus käyttöliittymässä. Sen yhteydessä nähtiin tapoja, joilla kutakin tekniikkaa ja sen toimintaa on havainnollistettu. Toinen kategoria on epäsuorasti hallitun toiminnallisuuden esittäminen. Siinä mielenkiinnon keskiössä ovat tavat, jolla käyttäjälle havainnollistetaan laajemmin ohjelman toimintaa, sen tuloksia ja mahdollisuuksia.

Esitetyt ratkaisut perustuivat kirjallisuudessa kuvattuihin prototyyppijärjestelmiin, mutta niiden käytöstä ei ole muodostunut yhtenäistä kuvaa. Oheinen taulukko (Taulukko 5) kerää kaikki tun-

nistetut ratkaisumallit ja esittelyssä käytetyt prototyypit yhteen. Taulukosta nähdään missä käsitellyistä prototyypeistä on käytetty mitäkin ratkaisumallia. Samalla sen avulla voidaan nähdä eri ratkaisumallien suosiota näiden ohjelmien joukossa. Ratkaisumallit on jaettu kappaleen organisoinnissa käytettyihin kategorioihin.

Taulukko 5 Käyttöliittymäratkaisujen käyttö esitellyissä järjestelmissä

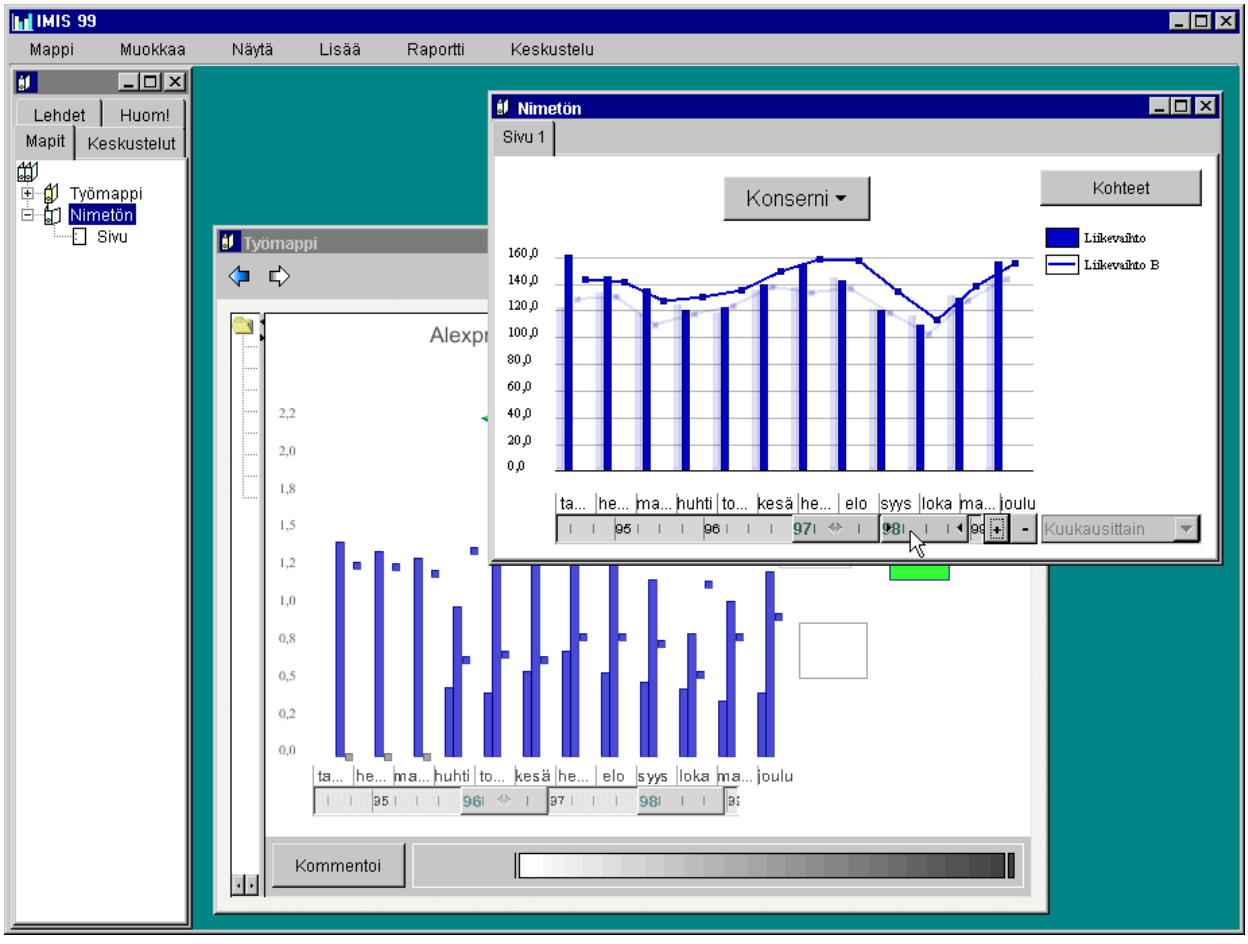
RATKAISUT / SOVELLUKSET	Tiedon suodatus / haku	NewT	Remembrance Agent	Amalthaea	Butterfly	Letizia	Let's Browse	ANATOGONOMY	Käyttäjän toimien automatisointi	Metamouse	Eager	MAXIMS	Selection Recognition Agent	Mukautuva käyttöliittymä	Split Menus	UIDE	Tims	Tehtävien tunnistus	PUSH	ACPA	Yhteisölliset	Firefly	Kasbah (Market Maker)
Olemassaolo																							
Itsenäinen ikkuna		■	■	■		■	■					■	■										
Tarvittaessa näkyvä					■					■	■					■	■		■	■			■
Piilottettu								■							■							■	
Ohjelman toiminta																							
Tulosten esittäminen		■	■	■		■	■	■			■	■			■		■		■	■		■	■
Toimintaehdotukset / selitys					■											■							
Toiminnan animointi						■				■		■											
Tulosten esittäminen																							
Perusteiden korostaminen							■												■				
Suoritettavien toim. esittäminen											■	■											
Tulosten pisteytys		■		■																			
Tulosvirta			■		■	■	■				■		■		■	■			■	■			
Talletettu tuloslista		■		■				■														■	■
Kontrolli																							
Ohjelman käynnistys/sammutus		■	■	■	■	■	■			■	■	■	■			■			■				
Varmistusdialogi										■		■				■				■			
Konfigurointi																							
Profiilin muuttaminen		■			■												■					■	
Toim. algoritmin muuttaminen		■		■																			■
Tulosten esitysmuodon valinta			■																				
Implisiittinen syöte																							
Tekstiraportit						■																	
Toiminnan animointi						■																	
Tilan graafinen näyttö		■								■		■	■										
Käyttäjäprofiili																							
Profiilin tekstimuotoinen näyttö					■		■										■						
Profiilin graafinen näyttö		■		■													■						
Objektin arviointimahdollisuus		■						■														■	
Oppiminen																							
Käyttäjän valtuutus												■				■							
Muutosvälin määrittely																■							
Prosessin animointi												■											
Yhteisölliset tekniikat																							
Tekniikan kuvaus ohjeissa																						■	■
Päättely																							
Objektien korostus										■													
Tuloksen havainnollistus											■												
Tehtävien tunnistus																							
Tunnistettu tehtävä tekstinä																			■	■			

6. ESIMERKKISOVELLUS

6.1 *IMIS-agentti*

Älykäs johdon tietojärjestelmä eli IMIS on Tekesin rahoittama Tampereen ja Helsingin yliopistojen tietojenkäsittelyopin laitoksien, Alma Median, Sitran ja PrettyBit Softwaren yhteisprojekti. Projekti on käynnistynyt vuoden 1996 syksyllä ja sen tavoitteena on ollut toteuttaa älykäs yritysjohdon tietojärjestelmä ja tutkia siihen liittyviä ongelmia. Rakennettua prototyyppiä hyödyntää Alma Media, jossa sen eri versioita on myös koekäytetty.

IMIS-sovellus on yritysjohdolle tarkoitettu tukityökalu raportointiin



Kuva 27 IMIS-järjestelmän käyttöliittymä

ja päätöksentekoon. Sen avulla käyttäjät voivat selata konsernin tulostietoja. Tietoja voidaan visualisoida useissa eri muodoissa. Ohjelma tukee myös käyttäjien synkronista ja asynkronista ryhmätyöskentelyä tarjoten tarkoitukseen erityisen keskustelutyökalun. Oheinen kuva näyttää ohjelman pääikkunan sekä muutaman raporttien tallettamiseen käytetyn mappi-ikkunan (Kuva 27).

Projektin tutkimustavoitteissa on mainittu myös automaattisten tietämyksen hallintavälineiden käyttökelpoisuuden arviointi tämän sovelluksen yhteydessä. Tarkoituksena on tutkia, miten käyttäjiä voidaan automaattisilla menetelmiä auttaa relevantin tiedon löytymisessä ja toisaalta vähentää riskiä, että jokin oleellinen tieto jäisi käyttäjältä näkemättä. Näihin tavoitteisiin vastaamaan on suunniteltu seuraavassa kuvattava IMIS-agentti.

6.1.1 Agentin tavoitteet

IMIS-agentilla pyritään tehostamaan IMIS-sovelluksen käyttöä. Siihen toteutettavalla toiminnallisuudella on tarkoitus vähentää käyttäjille muodostuvien rutiinien haittavaikutuksia. Jos käyttäjät kehittävät voimakkaan rutiinin IMIS-sovelluksen kaltaisen raportointivälineen käyttöön, on vaarana olla huomaamatta jonkin oleellisen tiedon olemassaolo. Joskus oleellinen tapahtuma sattuu ennalta odottamattomassa paikassa, eikä sitä ole mahdollista silloin huomata katselemalla vakioraportteja.

Toinen merkittävä tavoite agentin toiminnalle on tehostaa organisaatiossa olevan tietämyksen leviämistä. Suuressa organisaatiossa on tyypillisesti runsaasti epäformaalia, mutta erittäin oleellista tietämystä, jonka hallitseminen ja levittäminen on vaikeaa. Jos tietojärjestelmä onnistuu tarjoamaan välineet tällaisen tiedon helppoon syöttämiseen ja sen organisointiin, voidaan tietoa tehokkaammin levittää myös muiden käyttöön.

6.1.2 Skenaario

Johtaja saapuu työpaikalleen yhdeksän jälkeen kuten tavallista. Takana on käytäväkeskusteluja markkinointiosaston vetäjän kanssa ja aamukahvi erään taloushallintoihmisen kanssa. Keskustelut olivat herättäneet mielenkiintoisia ajatuksia, mutta miten ne saisi pysymään muistissa? Muutenkin muistettavaa on liikaa.

Työhuoneessaan Johtaja avaa tietokoneensa ja IMIS-ohjelman. Hän selaa vielä kerran yksikkönsä tulostietoja ja naapuriyksikön tulostietoja nähdäkseen oliko markkinointi-ihmisen jutuissa perää. Hänhän oli väittänyt naapurilla menneen alkuvuodesta erityisen hyvin, kun olivat lisänneet markkinoinnin määrärahoja oikein tuntuvasti. Kyllä, markkinoinnin budjetoidut ja todellisuudessakin käytetyt rahat olivat tosiaan oleellisesti nousseet. Johtaja lisää näkymään vielä naapuriyrityksen liikevaihdon kehityksen. Hieman käytettävissä olevien markkinointivarojen lisäämisen jälkeen liikevaihtoon oli tosiaan tullut voimakas nousu. Kun Johtaja rullasi näkymää hieman eteenpäin ei markkinointirahojen ja liikevaihdon yhteys ollutkaan enää niin selvä. Jostain syystä liikevaihto näytti tippuneen takaisin totutulle paikalleen muutamien kuukausien kulutta.

Tällä välin, kun Johtaja oli omatoimisesti tutkinut naapuriyrityksen tietoja, oli IMIS-agentti tehnyt omia hakujaan. Kohta näkymän viereen tuli ehdotus lehtileikkeestä. Agentti suositteli sitä kohtuullisella varmuudella ja Johtaja avasikin sen lukemista varten. Leikkeessä kerrottiin naapuriyrityksen pahimman kilpailijan kärsineen tulipalosta, joka oli sattunut jos-

kus alkuvuodesta. Johtajan tarkkaavaisuus heräsi. Tieto oli linkitetty naapuriyrityksen liikevaihtoon sille kohdalle, jossa se nousi voimakkaasti. Saattaakin siis olla, että liikevaihdon kasvun syynä olikin enemmän kilpailijan tilapäinen lamaantuminen, kuin markkinointipanoksen lisääminen.

Samaan aikaan agentti oli suositellut johtajan katsottavaksi myös naapuriyrityksen myynti- ja käyttökatetta. Niistä Johtaja ei ollut kiinnostunut, joten hän ei vaivautunut niitä katsomaan. Ehdotukset häipyivät Johtajan näkyvistä, kun hän siirtyi dokumentin toiselle sivulle, jossa oli hänen vakionäkymänsä oman yrityksen tilaan. Johtajan katseltua näkymää hetken agentti ehdotti hänelle muutamaa lehtileikettä, joiden se arvelee liittyvän näkymään. Agentti ilmaisi epävarmuutensa suositusten suhteen, mutta Johtaja päätti katsoa ne silti. Joskus niistäkin on hyötyä. Nytkin kaksi ehdotuksista oli puppua, mutta kolmannessa oli kiinnostavia uutisia pörssin käyttäytymisestä liittyen Johtajan omaan yritykseen.

6.2 Toimintaperiaatteet

Perustoiminnallisuudeltaan IMIS-agentti on epäsuorasti hallittu hakuohjelma, hakuagentti. Se suorittaa järjestelmään liittyvään tietokantaan erilaisia hakuja ja hakutulosten suodattamista. Hakuja tehdään kahdella eri menetelmällä, jotka vastaavat agentin toiminnalle asetettuja tavoitteita. Toteutustekniikoina agentissa käytetään yhteisöllistä suodattamista, käyttäjän toimien tarkkailua ja käyttäjäprofiilin tallentamista.

6.2.1 Toiminnot

IMIS-agentilla on tavoitteiden mukaisesti karkeasti jakaen kahdenlaisia tehtäviä. Ensiksi se pyrkii levittämään yhden käyttäjän tiedossa olevaa tietämystä järjestelmän muille käyttäjille ottaen kuitenkin huomioon eri käyttäjien vaihtelevat kiinnostukset. Tämä toteutetaan käyttäen hyväksi yhteisöllisiä tekniikoita.

Toiseksi se pyrkii aktiivisesti tuomaan käyttäjien ulottuville uutta tietoa, joka on vasta tullut järjestelmään ja joka on polttoainetta käyttäjien tietämyksen muodostamiselle. Tekniikkana käytetään sisältöön perustuvaa suodattamista (*content based filtering*). Tällä toiminnolla pyritään ensisijaisesti tukemaan ensimmäistä toimintoa, saattamalla käyttäjien ulottuville tietoa, jonka he voivat kiinnittää muihin tietoihin tarjotuilla välineillä. Kuten tiedon levittämisessä, myös uuden tiedon esiintuomisen perustana ovat käyttäjien henkilökohtaiset kiinnostuksen kohteet.

Käytetystä tekniikasta johtuva hienoinen epätarkkuus uuden materiaalin suosittelemisessa on siedettävää, sillä ohjelman on tarkoitus tuottaa ainoastaan ehdotuksia ja suurin riski on jonkin kiinnostavan tiedon huomiotta jääminen. Myös suosituksia tehdessään agentti ottaa huomioon käyttäjän kiinnostuksen kohteet ja juuri tässä käyttötilanteessa näkyvät tiedot. Varsinkin käyttötilanteen huomioiminen on oleellista, jotta tarjottu tieto olisi mahdollista mielekkäästi kytkeä johonkin näkymässä olevaan tietoon, joka on toiminnon pääasiallinen tavoite.

IMIS-ohjelma pyrkii tukemaan käyttäjien työskentelyä tavoilla, joiden seurauksena muodostuneita rakenteita ohjelma voi käyttää perustana tietojen levittämiselle. Perustava tekniikka tässä suhtees-

sa on tietojen linkittäminen toisiinsa. IMIS-agentti olettaa, että toisiinsa linkitetyt tiedot liittyvät jollain relevantilla tavalla toisiinsa. Tähän oletukseen perustuu agentin kyky ehdottaa käyttäjille tiettyyn aiheeseen liittyvää tietoa. Huomaa, että tietojen linkittämisen tekevät järjestelmän käyttäjät, ei kone.

6.2.2 Tietolähteet

Käyttäjälle suositeltavien aiheeseen liittyvien lisätietojen tunnistaminen on suhteellisen helppoa. Käyttäjät liittävät tietoja toisiinsa omien tarkoitustensa perusteella ja järjestelmän tarjoamilla välineillä. Välineet yhtäältä helpottavat käyttäjän tekemää tiedon organisointia ja toisaalta mahdollistavat tiedon automaattisen käytön. Koska kaikki toisiinsa linkitetyt tiedot ovat jo talletettuna järjestelmään, niihin päästään yksinkertaisesti käsiksi. Se, miten valitaan tarjottavaksi kaikille käyttäjille aivan uutta tietoa, ei ole kuitenkaan näin yksinkertaista. Kysymyksiksi nousee, mikä on uutta tietoa, miten se tunnistetaan ja mistä sitä tulee.

IMIS-järjestelmään syötetään säännöllisesti uutta sekä yrityksen sisäistä, että ulkoista tietoa. Kaikki tämä on uutta tietoa, mutta sitä on liikaa ollakseen agentin kannalta sopiva lähtökohta. Varsinkin yrityksen sisäistä, taloudellista tietoa ei kaikkea voida pitää kaikille suositeltavana. Ongelmaa voitaneen helpottaa *data mining* -tekniikoilla. Tietämyksenmuodostustekniikalla on mahdollista tunnistaa esimerkiksi poikkeavasti käyttäytyviä aikasarjoja. Jos tällainen aikasarja löydetään käyttäjää tyypillisesti kiinnostavan yrityksen tiedosta, voidaan tieto tästä nostaa esiin uutena kiinnostavana tietona.

Järjestelmään syötettävä ulkopuolinen tekstimuotoinen tieto voitaneen ottaa sinällään tietolähteeksi. Vaikka tällaistakin tietoa voi tulla järjestelmään hyvinkin paljon, voidaan sen kohdalla turvautua tiedon suodattamiseen, ettei käyttäjiä rasiteta liiallisella tietotulvalla. Tekstimuotoisen tiedon tapauksessa on analysoitava sen sisältöä, jotta suodattaminen voidaan asianmukaisesti toteuttaa. Jos tieto on jo käsiteltyä (esimerkiksi siihen on lisätty avainsanoja yms.), kaikkea siihen lisättyä tietoutta voidaan käyttää hyödyksi tiedon valikoinnissa.

6.2.3 Käyttöhistoria ja käyttäjäprofiili

Käyttäjäprofiili on agentin oppiva ja pysyvä osuus. Se koostuu käyttöhistoriasta, joka on käyttäjän näkemiä tieto-objekteja tallentava muistipankki sekä käyttäjäprofiilista, joka mallintaa käyttäjän kiinnostuksen kohteita. Käyttäjäprofiili rakennetaan käyttöhistorian perusteella.

Teknisesti ottaen IMIS-agenttiin liittyvä käyttäjäprofiili on käyttäjäkohtaisen käyttöhistorian tiivistelmä. Se kuvaa käyttäjän yleisimmin katsomia tieto-objekteja, joiden perusteella päätellään niiden kiinnostavuutta. Koska ihmisten kiinnostuksen kohteet vaihtelevat, myös käyttäjäprofiilia päivitetään säännöllisesti, jotta se pystyisi mallintamaan käyttäjäänsä kohtuullisella tarkkuudella. Tässä mielessä järjestelmä on oppiva (tai ajassa muuttuva).

Käyttäjäprofiilin tulee pystyä mallintamaan käyttäjän sekä pitkäaikaisia kiinnostuksen kohteita että nykyisessä työskentelyympäristössä kiinnostavia kohteita. Pitkäaikaisten kiinnostuksen kohteiden avulla voidaan löytää käyttäjän keskeisimpiin velvolli-

suuksiin liittyvää lisätietoa. Nykyisen työskentely-ympäristön huomioon ottaminen mahdollistaa puolestaan juuri nyt käsillä olevaan tilanteeseen liittyvän lisäinformaation hakemisen.

Käyttäjäprofiililla on vastuullaan seuraavien palveluiden tuottaminen:

- Suositella järjestelmän muille käyttäjille tiettyyn asiaan liittyviä tietoja perustuen mallintamansa käyttäjän profiiliin.
- Suodattaa tarjottuja tieto-oliota sen mukaan, onko käyttäjä nähnyt niitä viime aikoina vai ei.
- Arvioida tarjotun tiedon kiinnostavuutta mallintamalleen käyttäjälle.

Käyttöhistoriaan tallennetaan viittaus käyttäjän näkemään tieto-objektiin, sen painoarvo sekä järjestysnumero. Tapahtuman painoarvo riippuu sen tyypistä (käyttäjän aloittama, ohjelman ehdotus jne.) ja eräiden tapahtumien kohdalla myös tapahtuman kestosta (esim. katseluun käytetty aika). Tapahtumille annetaan käyttöhistoriaan tallentamisen yhteydessä järjestysnumero, jotta tapahtumien aikajärjestys säilyy.

Käyttöhistorian koko on parametroitavissa, jotta sen perusteella rakennetun käyttäjäprofiilin mallinnuskykyä voidaan säätää. Mitä suuremmaksi muistin koko kasvatetaan, sitä hitaammin käyttäjäprofiili reagoi käyttäjän toimintatapojen muutoksiin. Toisaalta, jos huomioon otettu historia on liian lyhyt, käyttäjäprofiili seuraa pienimpiäkin muutoksia käyttäjän toiminnassa eikä kykene mallintamaan käyttäjän pitkän aikavälin toimintaa.

6.2.4 Käyttäjäprofiilin rakentaminen

Käyttöhistoria on melko suuri, joten sitä ei teknisistä syistä voi sinällään käyttää käyttäjäprofiilina. Käyttöhistoria ja käyttäjäprofiili onkin erotettu toisistaan ja käyttäjäprofiili koostetaan yhteenvedoksi käyttöhistoriasta kaksivaiheisesti.

Koko käyttäjäprofiili päivitetään säännöllisesti, jotta se pystyisi seuramaan käyttäjän pitkäaikaisissa toiminnoissa tapahtuvia muutoksia. Päivityksen yhteydessä kaikki käyttöhistorian tapahtumat vaikuttavat muodostettavan käyttäjäprofiilin sisältöön. Koska koko käyttäjäprofiilin päivityksellä seurataan käyttäjän pitkäaikaisessa toiminnassa tapahtuneita muutoksia, voidaan päivitys suorittaa eräajona vaikkapa kerran vuorokaudessa. Näin harvoin suoritettavassa operaatiossa ei aikavaatimus ole kovin kriittinen, joten käyttöhistorian pituutta ei tarvitse tämän takia rajoittaa liiaksi.

Käyttäjäprofiilin on kyettävä mallintamaan myös käyttäjän nykyistä toimintaympäristöä. Tämä merkitsee sitä, että käyttäjäprofiilin on sisällettävä pitkäaikaisten käyttötietojen lisäksi tietoa myös aivan viimehetkisistä tapahtumista. Koska koko käyttäjäprofiilin päivittäminen on raskas operaatio, on sen rinnalla oltava toinen kevyempi päivitysoperaatio, joka voidaan suorittaa tarkoitusta varten riittävän usein.

IMIS-agentin käyttäjäprofiilin muokkaaminen on mahdollista. Jos käyttäjäprofiili sisältää aineksia, jotka käyttäjän mielestä eivät sinne kuulu, ne voidaan poistaa käyttöhistoriasta ja pakottaa käyttäjäprofiilin laskeminen uudelleen. Jos profiilista taas puuttuu joitain käyttäjän mielestä oleellisia kohteita, ne voidaan saada sinne li-

säämällä ne suurella painoarvolla käyttöhistoriaan ja pakottamalla jälleen käyttäjäprofiilin uudelleen laskeminen. Tämä mekanismi ei vääristä käyttäjäprofiilia kohtuuttomasti, koska lisäysten vaikutus vähenee ajan kuluessa, kuten muidenkin tapahtumien.

6.3 Epäsuoran hallinnan rooli

IMIS-agentissa sekä käyttäjäprofiilin päivittämistä että hakutoimintoa hallitaan epäsuorasti. Käyttäjäprofiilin päivittämisessä on mahdollista harkita myös käyttäjän suoran palautteen käyttämistä, mikäli se katsotaan oppimisen tarkkuuden tai käyttöliittymän kannalta tarpeelliseksi. Eksplisiittinen palaute käyttäjäprofiilin rakentamiseksi voisi olla tärkeää siksikin, että käyttäjäprofiilin perusteella ohjataan agentin hakutoimintoa. Tarkempi profiili tarkoittaa siten osuvampia hakutuloksia.

Kuten jo aiemmasta on käynyt ilmi, käyttäjäprofiilin epäsuora päivittäminen perustuu käyttäjän näkemien tieto-objektien tallentamiseen. Käyttäjän ei ole pakko mitenkään puuttua profiilinsa muokkaamiseen, vaan se tapahtuu automaattisesti käyttäjän suorittamien toimien perusteella.

Hakutoiminnon osalta epäsuora hallinta tarkoittaa sekä hakuehtojen että hakuajankohdan päättämistä käyttäjän suorasta vuorovaikutuksesta riippumatta. Tavoitteena on vapauttaa käyttäjä mahdollisen lisätiedon etsinnästä ja valjastaa kone tähän toimintaan. Toiminta on selvästi käyttäjän ensisijaista tehtävää tukeva: sillä autetaan käyttäjän tiedon hankintaa. Ensisijaisesti käyttäjä on itse vastuussa haluamansa tiedon etsimisestä.

7. EHDOTETTU KÄYTTÖLIITTYMÄ

7.1 Käyttöliittymän suunnitteluperiaatteet

Tämän työn kontekstissa esimerkiksi otetun automatisoidun toiminnallisuuden vuorovaikutustapa on selvä: epäsuora hallinta. Tutkimus on rakennettu silmällä pitäen nimen omaan tätä sovellusesimerkkiä, joten ei ole ihme, että esitellyn vuorovaikutustavan vahvuudet ja sovelluksen vaatimukset kohtaavat toisensa. Yleisessä tapauksessa näin ei tietenkään ole ja vuorovaikutustavan valintaan tulee kiinnittää tietoisesti huomiota.

IMIS-agentin on reagoitava melko nopeasti käyttäjän toimintoihin, jotta siitä olisi hyötyä. Käyttötilanteeseen liittyvän lisätiedon hankinta menettää merkityksensä, jos toiminto reagoi liian hitaasti tai ei kykene seuraamaan käyttäjän toimia riittävän tarkasti. Toisaalta käyttäjän toiminnot vaihtuvat alati ja ovat ohjelman näkökulmasta arvaamattomia, joten toiminnan ohjelmoiminen etukäteen ei onnistu. Näiden ominaisuuksien takia epäsuora hallinta soveltuu hyvin suunnitellun hakutoiminnallisuuden vuorovaikutustavaksi.

Seuraavassa esitellään suunnitteluperiaatteet, joiden avulla esimerkkikäyttöliittymä on laadittu. Suunnitteluprosessille päätettiin laatia periaatteet, jotta tulosta olisi mahdollista yleistää. Samalla suunnitteluperiaatteet konkretisoivat työn tutkivassa osassa muodostettua vaatimus-ratkaisu käsitteistöä. Periaatteet pohjautuvat esiteltyihin vaatimuksiin ja hyviksi todettuihin ratkaisuihin.

7.1.1 Yleinen suunnitteluperiaate

Suunnitteluperiaatteita on yhteensä 14, joista yksi on erityisasemassa. Ensisijainen periaate tukee epäsuoran hallinnan käyttöliittymille asetettuja yleisiä tavoitteita ja ottaa kantaa erityisesti kontrolloinnin mahdollisuuteen ja riittävän tiedon antamiseen ohjelman toiminnasta. Muut periaatteet ovat ensisijaista periaatetta tukevia.

Periaate pohjautuu keskusteluun epäsuoran hallinnan luonteesta vuorovaikutustapana, jossa sen todettiin olevan periaatteessa vaihdettava muiden vuorovaikutustapojen kanssa (katso luku ‘Epäsuoran hallinnan määritelmä ja historia’, sivu 6). Koska suoravaikutteisella ja epäsuoraan hallintaan perustuvalla vuorovaikutustavalla voidaan saada aikaan sama efekti, niitä voidaan pitää vaihdettavina. Epäsuorasti hallitun toiminnan käyttöliittymän perusperiaate onkin:

Periaate 1: *Tee epäsuorasti hallitulle toiminnolle tavallinen käyttöliittymä ja vaihdettava vuorovaikutustapa.*

Periaate kehottaa suunnittelemaan epäsuorasti hallitulle toiminnolle aluksi aivan tavallisen käyttäjän kontrolloiman käyttöliittymän. Tällainen käyttöliittymä käyttää vuorovaikutustapanaan tunnettua ja hyvin ymmärrettyä, kuten esimerkiksi suoravaikutteisuuteen tai lomakkeiden täyttämiseen perustuvaa vuorovaikutustapaa.

Näin saatuun toiminnallisuuteen lisätään mahdollisuus epäsuoraan hallintaan, joka tehdään vaihtoehtoiseksi käyttäjälähtöisen vuorovaikutustavan kanssa. Toiminnan käyttämiseksi on periaatteen mukaan rakennetussa käyttöliittymässä olemassa siis kaksi käyttäjän valittavissa olevaa vuorovaikutustapaa. Käyttäjälle annetaan valta päättää kumpaa vuorovaikutustapaa hän haluaa missäkin tilanteessa käyttää.

Suunnitteluperiaatteen tarkoituksena on varmistua siitä, että epäsuorasti hallituksi aiottu toiminto ei ryöstäydy liian monimutkaiseksi ja pysyisi siten ymmärrettävänä ja ennustettavana. Toisin sanoen suunnitteluprosessia ohjaamalla pyritään välttämään epäsuorasti hallittujen toimintojen suunnittelussa piilevä hämäryyden vaara.

Periaate tuottaa myös muutamia käyttöliittymäongelmia, jotka on ratkaistava ennen soveltamista. Ongelmia on identifioitu neljä, joista kolme ensimmäistä liittyy ratkaisun implikoimaan ohjelman tilaan. Neljäs ongelma aiheutuu ratkaisun käyttämästä käsitteistöstä, joka ei ole itsestään selvästi tyypillisille käyttäjille ymmärrettävää.

Kaksi eri käytettävissä olevaa vuorovaikutustapaa ovat ohjelman *tiloja*, joissa käyttäjän on mahdollisuus tehdä erilaisia toimia. Jos käyttäjä ei havaitse tilaa, ei osaa muuttaa sitä toiseksi tai ei ymmärrä sen merkitystä toimintojen saatavuuden kannalta, tilasta aiheutuu käytettävyysongelmia [Nielsen 93, s. 146].

Tilaongelma jakautuu kahdeksi aliongelmaksi. Ensinnäkin tarvitaan *tilan valintamekanismi*, jolla vuorovaikutustapa (eli tila) muutetaan. Toiseksi kullekin tilalle on löydettävä käyttäjän ymmärtämä *esitysmuoto*, joka mahdollistaa palautteen antamisen. Ilman kunnollista havainnollistusta ohjelman tila voi olla käyttäjälle hyvin huomaamaton käsite, jollaisena se on vaikea hallita.

Vuorovaikutustapa on tavalliselle käyttäjälle todennäköisesti *tuntematon käsite*, mikä on esitettyyn periaatteeseen perustuvan käyttöliittymän neljäs ongelma. Käsite vuorovaikutustapa on merkittävä sovelluskehittäjille ja käyttöliittymätutkijoille, joiden työssä tällaisella käsitteellä saadaan tiettyjä ilmiöitä näkyviksi ja tarkastelun kohteeksi. Tiedon ymmärtämisen kannalta on kuitenkin edullisempaa viitata käyttäjälle tuttuihin käsitteisiin [Nielsen 93, s. 123].

7.1.2 Toiminnallisuuden havainnollistamisen periaatteet

Tutkimuksen alkupuolella nähtiin, miten epäsuorasti hallitun toiminnon havainnollistaminen on haastava tehtävä. Edes toiminnallisuuden olemassaolon havaitseminen ei ole itsestään selvää ja toiminnallisuuden havainnollistamiselle onkin asetettu useita vaatimuksia, joihin hyvän käyttöliittymän olisi vastattava. Seuraavat

suunnitteluperiaatteet ottavat kantaa toiminnallisuuden havainnollistamiseen liittyviin vaatimuksiin.

Epäsuoran hallinnan koostavien osien erottelulle asetetut vaatimukset ovat laaja-alaisia, sillä mainitut osat ovat suuria ja monimuotoisia. Osien erottelulla voidaan selkeyttää ohjelman rakennetta ja auttaa käyttäjää ymmärtämään eri asetusten ja komentojen vaikutusalaa. Tavoitteiden saavuttamista palvelee useampi periaate, joista ensimmäinen on:

Periaate 2: Tee epäsuoraan hallintaan liittyvät osat havaittaviksi.

Eri osien havainnollistaminen auttaa käyttäjää ymmärtämään ohjelman toimintaperiaatteita ja käsittämään mitkä kaikki asiat ohjelman toimintaan voivat vaikuttaa. Niiden havaittavaksi tekeminen on edellytys sille, että epäsuorasti hallittujen toimntojen erottelu ohjelman muusta toiminnallisuudesta voisi onnistua.

Epäsuoraan hallintaa liittyviä osia on tyypillisesti useita ja ne voivat olla hajautettu ohjelman muun toiminnallisuuden joukkoon. Kokoavaa käsitettä siis tarvitaan, johon seuraava periaate ottaa kantaa:

Periaate 3: Epäsuoran järjestelmän kokoaa yhteen sen konfigurointikäyttöliittymä.

Konfigurointikäyttöliittymän eräs merkittävä tehtävä on siis koota epäsuoraan hallintaan liittyvät toiminnallisuudet käyttäjän silmissä yhteen paikkaan.

Kun ohjelman koostavat osat on tunnistettu ja havainnollistettu voidaan suunnitella tarkemmin jokaisen toiminnon havainnollistamista. Epäsuoran hallinnan toteuttavien komponenttien toiminnallisuus voi olla käyttäjää kiinnostavaa, mutta tärkeimpänä havainnollistettavana toimintona on käyttöliittymävaatimustenkin valossa epäsuorasti hallittu toiminto. Toiminnon havainnollistamista ohjaavan perusperiaatteen ongelmien ratkaisemiseksi on esitettävä periaate:

Periaate 4: Epäsuorasti hallitulla toiminnallisuudella on automaattinen ja manuaalinen tila.

Tämän periaatteen soveltaminen tarkoittaa hankalan käsitteen ‘vuorovaikutustapa’ unohtamista ja vastaavan asian esittämistä toiminnallisuuden automatisointina. Toisin sanoen, käyttäjän näkökulmasta kaikki epäsuorasti hallitut toiminnot voidaan normaalin käytön lisäksi asettaa myös automaattiseen tilaan, jossa ohjelma itse käyttää toimintoa oman logiikkansa perusteella.

Koska toiminnon automatisoinnissa käyttöliittymän metaforaksi tulee automaattikone, on perusperiaatteen ongelmiksi tunnistettujen tilojen ratkaisun noudatettava tätä metaforaa. Siksi:

Periaate 5: Tilan havainnollistus toteutetaan näkyvästi ja konemaisesti.

Tilan näkyvyys on ensisijaisen tärkeä vaatimus käyttöliittymälle, kun vältetään tilasta helposti aiheutuvia ongelmia. Toinen vaatimus on tilan helppo muuttaminen, johon myös seuraava periaate ottaa kantaa:

Periaate 6: *Tila muuttuu manuaaliseksi kaikista käyttäjän aloittamista toimista.*

Paitsi konemaisilla kontrollipainikkeilla, tilan muuttaminen tulee olla mahdollista myös epäsuorasti. Erityisesti automaattisessa tilassa mahdollisten käyttäjän suorittamien toimintojen määrä on tiukasti rajattu. Käyttäjä voi kuitenkin haluta suorittaa toiminnon automaattisen toiminnan innoittamana. Tällaisia tapauksia varten käyttöliittymä rakennetaan niin, että kaikki käyttäjän suorittamat toiminnot automaattisessa tilassa tulkitaan pyynnöksi siirtyä manuaaliseen tilaan.

Itse epäsuorasti hallitun toiminnallisuuden ymmärtäminen on myös oleellista, ehkä jopa oleellisempaa kuin edellä kuvatun automaattisen ja manuaalisen tilan kuvaaminen. Epäsuorasti hallittu toiminto on kuitenkin ainoa komponentti, joka tuottaa käyttäjää hyödyttäviä tuloksia. Sen toimintaperiaatteen ymmärtäminen on tulosten ymmärtämiseksi tärkeää:

Periaate 7: *Epäsuorasti hallittu toiminnallisuus havainnollistetaan animaatiolla.*

Animaatio suoritetaan samassa käyttöliittymässä, millä käyttäjä itsekin toimintoa manuaalisessa tilassa käyttää. Tällöin animointi voi myös opettaa ohjelman käyttöä ja nähtyjen toimien soveltaminen on yksinkertaista. Koska animoitaessa eli automaattisessa tilassa on käytössä sama käyttöliittymä kuin manuaalisessa tilassa, myös tulosten hyödyntäminen on tuttua. Tästä puhuu myös seuraava periaate:

Periaate 8: *Esitä tulokset sekä tutussa että kompaktissa muodossa.*

Epäsuorasti hallitun ohjelman tulosten esittäminen tapahtuu siis kahdella tavalla: 1) tulokset esitetään samanlaisessa muodossa, missä ne esitettäisiin jos ohjelmaa käyttäisi ihminen ja 2) mahdollistetaan tulosten esittäminen myös vähemmän häiritsevässä ja tilaavievässä muodossa. Ensimmäinen tapa helpottaa tulosten käsittelyä, toinen vähentää automaattisesti toimivan ohjelman häiritsevyyttä vähentämällä käyttöliittymän liikettä. Samalla käyttäjä saa lisää tilaa omien toimiensa suorittamiseen.

7.1.3 Hallinnan mahdollistamisen periaatteet

Hallinta on noussut keskeiseksi osaksi epäsuorasti hallitun käyttöliittymän käytettävyyttä. Hallinta on jaettu tässä työssä kahteen osaan: konfigurointiin ja kontrollointiin, joihin liittyviä suunnitteluperiaatteita tarkastellaan seuraavaksi.

Kaksi ensimmäistä konfigurointiin liittyvää periaatetta antavat säännöt konfiguroinnin jäsentämiselle. Ne auttavat päättämään, mitä konfigurointikäyttöliittymä sisältää ja miten tämä sisältö järjestetään:

Periaate 9: *Jokaiselle tunnistetulle osalle oltava omat konfigurointitiedot.*

Periaate 10: *Konfigurointi jäsennetään tunnistettujen osien mukaan.*

Konfigurointi perustuu vahvasti epäsuoran hallinnan toteuttavan järjestelmän eri osien tunnistamiseen. Tämä mahdollistaa osien, niiden käyttötarkoituksen ja niiden välisten suhteiden selkeämmän esittämisen myös ohjelman käyttäjälle.

Periaatteena konfiguroinnin mahdollistavassa käyttöliittymässä on jaotella jokainen konfiguroitava toiminto omaksi kokonaisuudekseen ja tarjota mahdollisimman monipuoliset konfigurointimahdollisuudet jokaiselle näin erotellulle toiminnolle. Konfiguroitavat toiminnot ryhmitellään edelleen epäsuoran hallinnan toteuttavien ohjelmakomponenttien avulla. Tämä on yhteydessä myös seuraavaan periaatteeseen:

Periaate 11: Konfigurointikäyttöliittymä havainnollistaa ohjelman toimintaperiaatteita.

Sen lisäksi, että käyttäjä voi konfiguroinnin avulla muokata ohjelman toimintaa sopimaan paremmin omaan toimintatyyliinsä, on konfiguroinnilla tärkeä tehtävä myös ohjelman toimintaperiaatteiden kertojana. Konfigurointidialogi on tässä esitetyssä käyttöliittymäkonseptissa ainoa paikka, jossa kaikki epäsuoraan hallintaan liittyvät komponentit ja toiminnot ovat kerralla näkyvissä. Se on myös ainoa paikka, jossa on tilaa kertoa kaikista toiminnoista. Siksi konfigurointidialogin sisällön valintaan tulisi vaikuttaa myös toimintaperiaatteiden esittäminen.

Konfigurointidialogille on ajatelu myös kolmas merkitys asetusten muuttamisen ja toimintaperiaatteiden esittämisen rinnalle. Sen tehtävänä on antaa käyttäjälle kontrollin tunnetta suhteessa epäsuoraan hallintaan. Siksi:

Periaate 12: Osa kontrollista toteutetaan konfigurointina.

Konfiguraation yhteydessä kaikki ohjelman säädeltävät ominaisuudet ovat keskitetysti saatavilla. Jotta kontrollin ja konfiguroinnin yhteys selkiytyisi, käyttöliittymän kontrollointiosuus voidaan ajatella konfigurointikäyttöliittymän supistetuksi versioksi. Ainakin tärkeimmät kontrollointikäyttöliittymän toiminnallisuudet löytyvät siten myös konfigurointikäyttöliittymästä. Osa kontrollista on kuitenkin oltava helpommin saatavilla:

Periaate 13: Toiminnon pois ja päälle kytkeminen sijoitetaan näkyviin.

Pääperiaate kontrollin mahdollistamisessa on tarjota jatkuvasti näkyvillä olevat välineet toiminnon käynnistämiseksi tai sammuttamiseksi. Tämän avulla käyttäjä voi esimerkiksi estää tarkkailun, jos hän kokee asian arkaluontoiseksi tai jos se tuntuu kuluttavan resursseja liiaksi. Koska toiminnan sammuttaminen ja käynnistäminen ovat vain yhden painikkeen painalluksen takana, voidaan käyttäjän odottaa myös laittavan toiminnon takaisin päälle, kun tilanne on ohi.

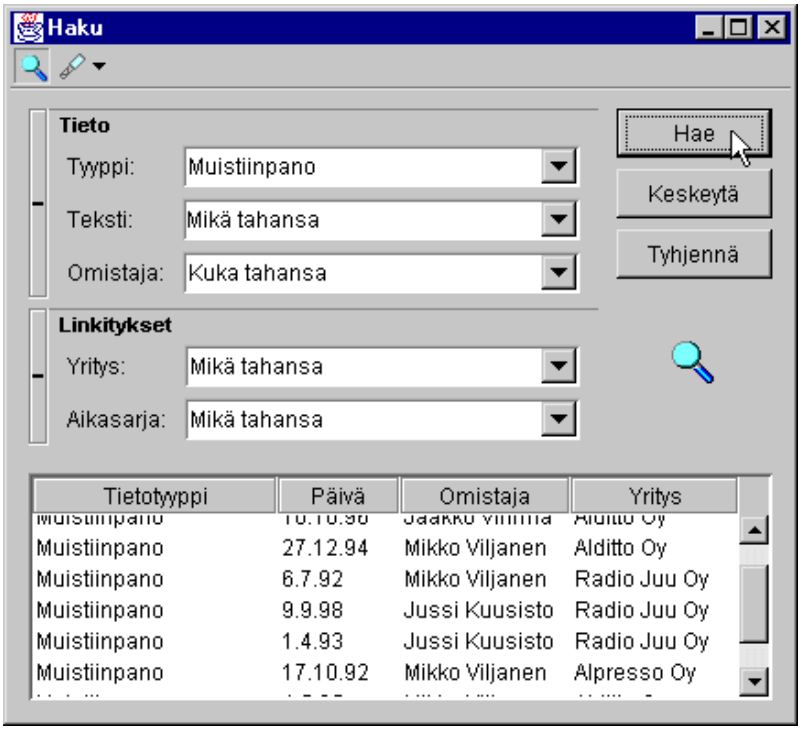
7.2 Toiminnallisuuden havainnollistus

Pääperiaatteen mukaisesti toiminnallisuuden havainnollistus perustuu pitkälti konventionaaliseen käyttöliittymään. Käyttöliittymä on toteutettu standardeilla komponenteilla, joiden merkitys on entuudestaan tuttua. Merkittävä osa toiminallisuutta on myös ratkaisussa olevan tilan hallinta, mikä on potentiaalinen käytettävyysongelmien lähde. Kolmas osa esimerkkikäyttöliittymän toiminnallisuutta on automaattinen tila.

Seuraavassa esitellään suunnitteluperiaatteiden perusteella rakennetun käyttöliittymän toiminnallisuutta. Aluksi tarkastellaan manuaalisen tilan toiminnallisuutta, joka toimii perustana muulle toiminnallisuudella. Seuraavaksi tarkastellaan siirtymää tilasta toiseen. Tarkastelu on jaettu kahteen osaan: tilan hallintaan ja sen havainnollistamiseen. Lopuksi luodaan katsaus siihen, millaisia toimintoja automaattinen tila tarjoaa.

7.2.1 Manuaalinen toiminnallisuus

IMIS-agentissa epäsuora hallinta kohdistuu hakutoimintoon. Haun käynnistyshetki valitaan tarkkailun perusteella samoin kuin hakuehdot. Tuotettujen tulosten määrää rajoitetaan suodattamalla niitä tarkkailun perusteella muodostetun käyttäjäprofiilin avulla.



Kuva 28 IMIS-sovelluksen hakutoiminnallisuus manuaalisessa tilassa

Perusperiaatteen (Periaate 1) mukaisesti epäsuorasti hallitun hakutoiminnallisuuden käyttöliittymän lähtökohtana on tavallinen käyttäjälähtöinen vuorovaikutustapa. Käyttöliittymä on toteutettu standardin mukaisilla graafisilla käyttöliittymäkomponenteilla (Kuva 28) ja se voidaan jakaa kolmeen osaan: hakutoiminnot (kentät ja painikeet), tulokset (taulukko) ja työkalupalkki (ylälaita).

Hakutoimintoihin kuuluvat haun kriteerit on jaettu kahteen kategoriaan. Ensimmäisen kategorian (Tieto) määrityksillä vaikutetaan siihen, millaista tietoa haetaan. Käyttäjä voi valita tiedon tyypin ja / tai omistajan. Etsityn tiedon kriteerinä voidaan pitää myös tieto-objektin sisältämää tekstiä.

Linkitykset-kategoriassa määritellään, millaisiin tieto-objekteihin etsityn tiedon tulee liittyä. Käyttäjän on mahdollista määritellä aikasarja ja / tai yritys, joihin yhdistetyt tiedot täyttävät haun kriteerit.

Hakukriteerit yhdistetään AND-operaattorilla, joten tulosobjektit täyttävät kaikki annetut ehdot. Kaikissa hakuehdoissa on olemassa erityinen aina todeksi evaluoituva arvo (esim. mikä tahansa), jotta vähemmän rajoittavien hakujen laatiminen olisi yksinkertaista.

Koska hakukriteerit ovat luonteeltaan lueteltuja arvoja, käyttäjän ei tarvitse muistaa tai itse kirjoittaa haluamaansa arvoa kenttiin. Jokaisen kriteerin mahdolliset arvot löytyvät valintalaatikon listasta. Ainoa poikkeus on kenttä Teksti, johon käyttäjä voi kirjoittaa merkkijonon. Kirjoitettu merkkijono tallentuu valintalaatikon listaan uutta käyttöä varten.

Haku suoritetaan Hae-painikkeella ja lopetetaan ennenaikaisesti Keskeytä-painikkeella. Tyhjennä-painike palauttaa kaikkiin kenttiin oletusarvon (kentän erityinen mikä tahansa -arvo).

Ikkunan alalaidan taulukko on käyttöliittymän tulososa. Taulukossa jokaisesta tulosobjektista esitetään sen tyyppi, luontipäivämäärä, omistaja sekä yritys, johon tieto liittyy. Tulosobjektin avaaminen tapahtuu joko kaksoisosoittamalla sitä tai raahaamalla se johonkin IMIS-dokumenttiin.

7.2.2 Tilojen hallinta

Hakutoiminnallisuutta on mahdollista hallita myös epäsuorasti, eli käyttää sitä automaattisessa tilassa (Periaate 4). Käytettävyyden kannalta tilan hallinta ja havainnollistus ovat keskeisiä tekijöitä, joten kontrollin tulee olla ilmeistä ja tiedon valitusta tilasta aina selvästi näkyvillä (Periaate 5, Periaate 13). Myös valittua metaforaa koneesta (Periaate 5) on kunnioitettava.

Koska käyttäjän valittavana on kaksi tilaa, voidaan käyttöliittymän ajatella kontrolloivan vain toisen aktiivisuutta. Käyttöliittymäkomponentteja, jotka periaatteessa soveltuvat tällaisen informaation esittämiseen ja hallintaan on useita: valinta- (*radiobutton*), asetus- (*checkbox*) ja kiikkukytkin (*togglebutton*). Kiikkukytkin voidaan toteuttaa tavallisena painikkeena, joka valittuna piirretään pohjaan painuneena. Tällaisia painikkeita käytetään yleisesti työkalupalkeissa (Kuva 29). Se on suurikokoinen ja siten näkyvä komponentti, joka vertautuu hyvin fyysisten koneiden kontrollinäppäimiin.

Kuva 29 Painonapilla toteutettuja kiikkukytkimiä (MS Word)

Painikkeen merkitystä selventää sen konemainen ulkoasu, sillä käyttäjän näkökulmasta tällä painikkeella hallitaan ohjelman automaattista toimintaa. Automaattisen toiminnan päälle kytkeminen vertautuu paremmin koneen päälle kytkemiseen kuin esimerkiksi asetusten tekemiseen. Valinta- tai asetuspainike yhdistyvät enemmän tiedon tai asetusten syöttämiseen, jollaisina ne eivät sovellu tarkoitukseen yhtä hyvin.

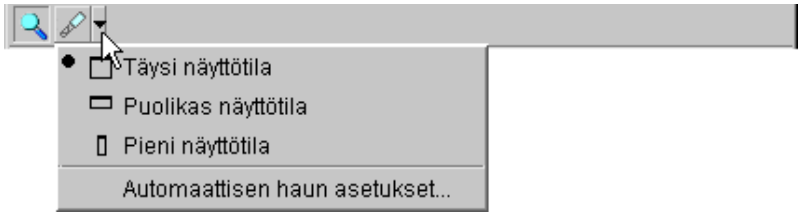
Vaikka periaatteessa tilainformaation muuttamiseen ja välittämiseen riittäisikin vain yksi painike, jolla säädeltäisiin automaattisen tilan aktiivisuutta, otettiin ratkaisuun kaksi painiketta. Näin painikkeiden merkitys saadaan selvemmäksi, kun molempiin voidaan asettaa oma työkaluvihje (*tool tip*) ja toimintaa kuvaava ikoni (Kuva 30). Painikkeet muodostavat ryhmän, jossa vain toinen voi olla kerralla valittuna.

Kuva 30 Tilan muuttamiseen käytetyt painikkeet ja työkaluvihje

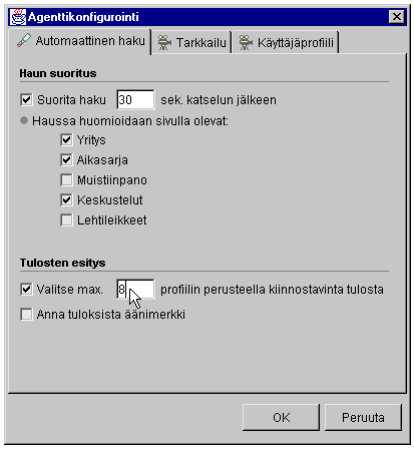
Painikkeet on sijoitettu työkalupalkkiin, joilla hallitaan automaattisen tilan päällä oloa, hakutoiminnon näyttötilaa ja epäsuoran hallinnan konfigurointidialogin avaamista. Työkalupalkki on tilan-

hallintapainikkeille hyvä sijoituspaikka, sillä se on näkyvä ja usein käytetty.

Tilanhallinta myös soveltuu hyvin työkalupalkkiin sijoitettavaksi. Työkalupalkit kontrolloivat tyypillisesti niiden alapuolella olevaa komponenttia, mikä on tilanne tässäkin tapauksessa. Yleisesti käytettyjen kontrollointisuhteiden noudattaminen helpottaa toimintojen kohteiden ymmärtämistä.



Kuva 31 Hakuohjelman työkalupalkki ja siihen liittyvä ponnahdusvalikko



Kuva 32 Työkalupalkin ponnahdusvalikosta voi avata konfigurointidialogin

Automaattisen tilan aktivoiva painike on kaksiosainen. Oikean puoleinen osa avaa ponnahdusvalikon (Kuva 31), jolla voidaan muuttaa automaattisen toiminnon näyttötilaa ja konfigurointia. Konfigurointikomento avaa konfigurointidialogin automaattisen haun välilehdeltä (Kuva 32).

Mahdollisia näyttötiloja (esitellään myöhemmin tulosten yhteydessä) on kolme ja ne ovat käytettävissä ainoastaan automaattisen toiminnon aikana. Jos käyttäjä valitsee jonkin näyttötilan manuaalissa tilassa, muutetaan tila automaattiseksi. Ohjelma muistaa edellisen näyttötilansa, joka palautetaan, jos automaattiseen tilaan siirrytään pikapainikkeen, eikä valikkovalinnan avulla.

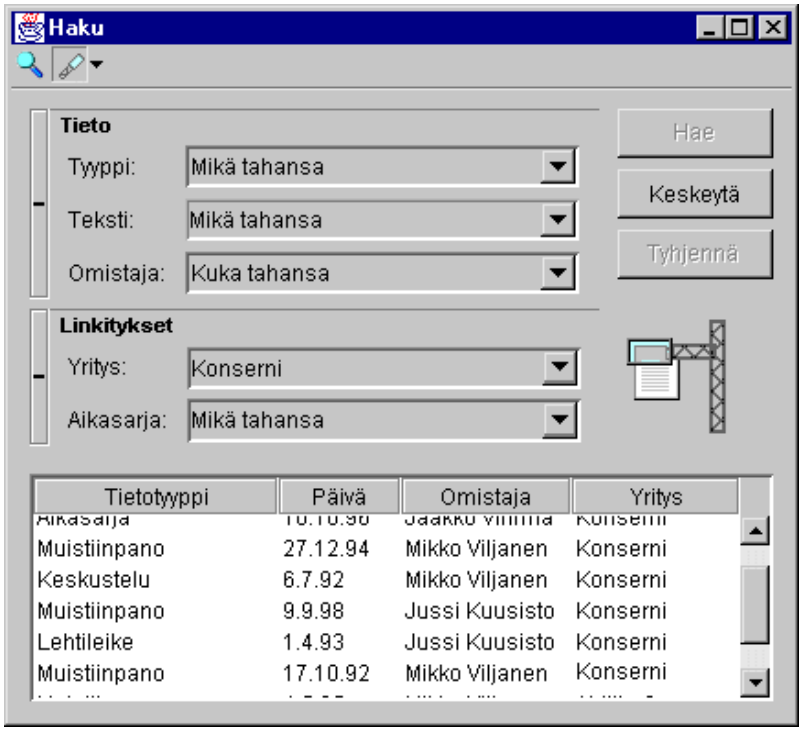
Paitsi kontrollipainikkeiden avulla, tilan muutos voi tapahtua myös epäsuorasti (Periaate 6). Automaattisessa tilassa käyttäjän toimiin kuuluu ainoastaan tulosten käsittely, joten kaikkien kenttien arvojen muuttaminen (eli haun muodostaminen) aiheuttaa tilan muuttamisen manuaaliseksi. Myös Keskeytä-painikkeen painaminen automaattisessa tilassa muuttaa tilan manuaaliseksi.

7.2.3 Tilan havainnollistaminen

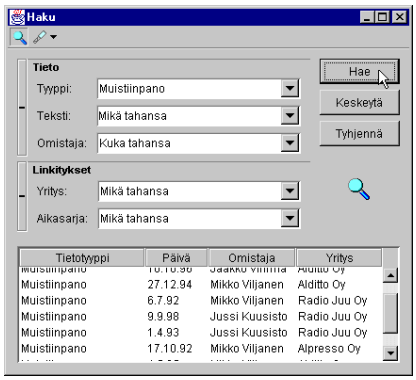
Palautetta aktiivisena olevasta tilasta annetaan neljässä muodossa. Ensimmäinen tilaindikaattori on tilan kontrollointiin tarkoitetut painikkeet, joista edellä jo puhuttiinkin. Päällä olevaa tilaa hallitseva painekehan on piirretty pohjaan painuneena.

Toiseksi tilasta viestitetään värillä (Kuva 33 ja Kuva 34). Kaikille käyttöliittymäkomponenteille, jotka esittävät toimintoa, joka automaattisessa tilassa ei ole riippuvainen käyttäjän suorasta vuorovaikutuksesta, määritellään oma värinsä (harmaa). Tätä väriä käytetään esimerkiksi hakutoimintoa esittävän dialogi-ikkunan kentissä, joilla määritellään haun kriteerit. Kun ohjelma täyttää kenttiä, niiden tausta on harmaa, mutta käyttäjä on tekemisissä valkopohjaisten komponenttien kanssa.

Kolmanneksi käyttöliittymäkomponentit, joita tarvitaan vain käyttäjälähtöisessä vuorovaikutuksessa, poistetaan käytöstä automaattisessa tilassa. Esimerkkiohjelmassa tällaisia ovat haun käynnistämiseen ja hakukriteerien nollaamiseen käytetyt painikkeet,



Kuva 33 IMIS-sovelluksen hakutoiminto automaattisessa tilassa



Kuva 34 Hakutoiminto manuaalisessa tilassa

joita automaattisessa haussa ei edes periaatteessa tarvita. Jos kontrolli on poistettu käytöstä, se esitetään harmaannutettuna.

Neljänneksi hakutoiminnon suorituksesta palautetta antavaa animoitua ikonia muutetaan tilan mukaan. Käyttäjän hallitsemassa tilassa ikonin animaatio on pyöreäliikkeinen ja noudattelee Microsoft Windows -käyttöjärjestelmän ‘Etsi’ -toiminnon käyttämää tapaa. Automaattisessa tilassa animaation liikkeet on kulmikkaat samoin kuin ikonin muodotkin.

7.2.4 Automaattinen toiminnallisuus

Epäsuorasti hallitusta toiminnosta on vaatimusten perusteella kerrottava käyttäjälle ainakin sen olemassaolosta, sen mahdollisuuksista ja toiminta-ajankohdista. IMIS-sovelluksen hakutoiminnolle on varattu käyttöliittymässä oma ikkuna, joka on näkyvissä toiminnon ollessa käyttövalmiudessa. Tämä viestittää selvästi ohjelman olemassaolosta.

Ohjelman toiminta-ajankohta on selvästi näkyvillä saman ikkunan kautta. Koska automaattisessa tilassa ohjelman toiminta esitetään ohjelman normaalin käyttöliittymän avulla animoiden, käyttäjä voi itse nähdä, milloin mitäkin hakukriteerejä muutetaan ja milloin haku suoritetaan (Kuva 33). Hakuajankohta esitetään samantyyppisellä kuva-animaatiolla, kuin manuaalisessakin tilassa.

Automaattisen haun mahdollisuuksien ymmärtäminen perustuu kahteen seikkaan. Ensinnäkin toiminnon animointi (Periaate 7) antaa käyttäjälle varsin runsaan kuvan siitä, mitä epäsuorasti hallittu toiminto kulloinkin tekee (Kuva 33). Toiseksi toiminto on myös manuaalisesti käytettävissä (Periaate 1), joten käyttäjä voi kokeilemalla todeta, millaisia tuloksia hakuohjelmalla saa aikaan

erilaisilla hakuehdoilla. Ohjelman automaattisessa tilassa ei käytetä mitään toimintoja, mihin käyttäjällä ei itsellään olisi pääsyä.

Automaattisen toiminnan esittäminen animoinnilla aiheuttaa potentiaalisesti häiritsevää liikettä. Tämän häirintäefektin poistamiseksi on käyttäjällä mahdollisuus piilottaa animointi. Tämä tapahtuu automaattisen hakutoiminnon eri näyttötilojen avulla, joiden toimintaa kuvataan tarkemmin hieman tuonnempana tulosten esittämisen yhteydessä.

Näyttömuotojen hallitsemisen lisäksi käyttäjä voi manipuloida tuloslistaa tavalliseen tapaan ja avata kiinnostavia tulosobjekteja. Kolmanneksi hän voi laittaa ohjelman manuaaliseen tilaan. Se voidaan tehdä kolmella tavalla: työkalupalkista, Keskeytä-painikkeella tai muuttamalla mitä tahansa hakuehtoa.

7.2.5 Käyttäjän tarkkailun havainnollistus

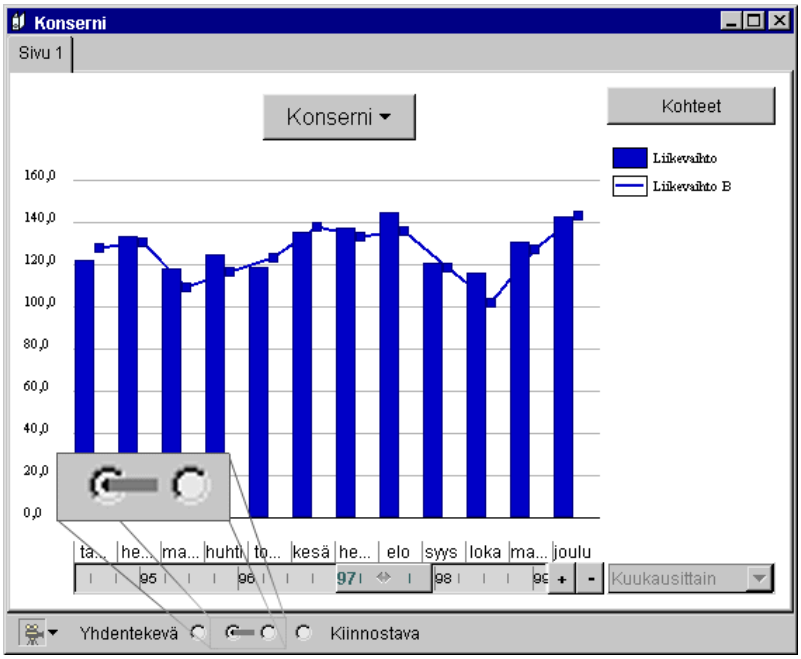
Käyttäjän toimien tarkkailu on eräs tunnistettu ja havainnollistettavaksi valittu osa (Periaate 2) IMIS-agentin epäsuoraa hallintaa. Ehkä jossain määrin erikoiseksi IMIS-agentissa suoritetun tarkkailun hyödyntämisen tekee se, että tuloksia hyödynnetään kahdella tavalla: yhtäältä tarkkailun perusteella muodostetaan käyttäjäprofiilia, joka mallintaa kiinnostuksen kohteita ja toisaalta samoja havaintoja käytetään hakuehdon muodostamiseen. Tarkkailun vaikutus kahteen eri toimintoon korostaa sen merkitystä entisestään ja asettaa näin lisää vaatimuksia sen ymmärtämiselle.

Käyttöliittymässä tarkkailu esitetään yhdellä tarkoitukseen varatulla komponentilla. Komponentti edustaa tarkkailun toteuttavaa ohjelmamoduulia käyttöliittymässä ja esittää sen tilan kullakin hetkellä. Samalla komponentti tarjoaa luontevan paikan erilaisten kontrollointivälineiden sijoittamiseen käyttöliittymässä. Etenkin tarkkailutoiminnon kytkeminen päälle ja pois on tällainen toiminto. Tämän lisäksi käyttöliittymän suunnittelussa valittiin tarkkailutoiminnon konfigurointi sellaiseksi toiminnoksi, johon käyttäjällä tulee olla yksinkertainen pääsy.

Tarkkailutoimintoa kuvaavaa komponentti toteutetaan ‘tarkkailupalkkina’, joka muistuttaa yleisesti käytössä olevia tila- tai työkalupalkkeja (Kuva 35). Käyttöliittymäkomponentti on muodoltaan pitkulainen palkki, joka ei vie näyttöruudulta kohtuuttomasti tilaa ja mahdollistaa täten melko vapaan sijoittelun. Käyttöliittymätilan säästäminen tämän komponentin kohdalla on merkityksellistä siksikin, ettei käyttäjän ensisijaiselta toiminnolta viedä liiaksi toimintatilaa.

Tarkkailupalkin muodosta johtuen siihen voidaan sijoittaa komponentteja vain yhteen riviin. Toisin sanoen komponentin sisällä olevien osien tarkoituksen spatiaaliseen merkitsemiseen on käytettävissä ainoastaan yksi ulottuvuus (vasen-oikea). Ulottuvuus on päätetty jakaa kahteen toimintoja kuvaavaan osaan vasemmalta oikealle: 1) suora kontrolli ja 2) palaute toiminnan tilasta. Osien keskinäinen sijoittelu perustuu lukemisen suuntaan länsimaisessa kulttuurissa, jonka perusteella sijoittelu muodostaa niiden välille järjestyksen vasemmalta oikealle.

Toimintojen keskinäinen järjestys on valittu niiden tärkeyden perusteella. On oletettu, että tarkkailun kytkeminen päälle ja pois on kontrolloinnin kannalta tärkein toiminto, joten se on järkevää sijoittaa lukusuunnassa ensimmäiseksi etsinnän helpottamiseksi.



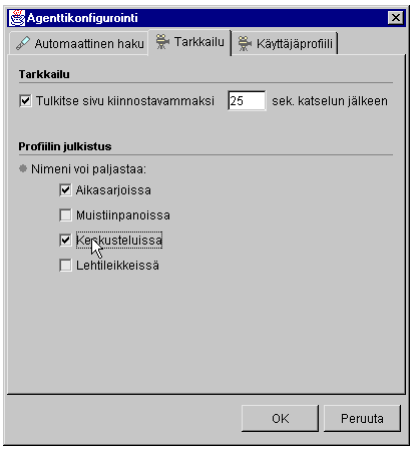
Kuva 35 Tarkkailupalkki liitettynä erääseen tieto-objektiin

Tutkittaessa silmän liikkeitä käyttöliittymien yhteydessä on todettu, että tyypillinen paikka käyttöliittymän lukemisen aloittamiseksi on vasen yläkulma [Mayhew 92, s. 466]. Usein käytetyt ja tärkeät toiminnot on tämän havainnon perusteella mielekästä sijoittaa vasemmalle ylös. Koska tarkkailupalkissa ei ole mahdollisuutta käyttää ylös-alas -akselia, jää vasen-oikea -akseli ainoaksi mahdollisuudeksi. Toisaalta myös akselilla välitetty informaatiomäärä on kohtuullisen vähäinen, joten yksi ulottuvuus riittää siihen hyvin.

Myös tarkkailupalkin sijoittelu tarkkailtavan komponentin suhteen perustuu havaintoon, jonka mukaan käyttöliittymää luetaan alkaen vasemmasta yläkulmasta. Koska tarkkailutoiminto on toissijainen tarkkailtavan komponentin toimintoon nähden, se sijoitetaan sen alapuolelle eli lukujärjestyksessä sen jälkeen. Näin käyttäjän varsinainen toiminto on myös spatiaalisesti ensisijaisessa asemassa verrattuna tarkkailuun.

Sijoittelusta seuraa se, että tarkkailukomponentin havaitseminen on vaikeampaa ja epätodennäköisempää kuin jos se olisi sijoitettu tarkkailtavan komponentin yläpuolelle. Tämä voi vaikeuttaa koko tarkkailutoiminnon ymmärtämistä, koska saattaa kestää kauankin ennen kuin käyttäjä todella huomaa komponentin olemassaolon. Toisaalta tämä on tietyssä mielessä myös haluttu vaikutus. Toissijainen toiminto ei saa häiritä ensisijaista, joten on tässä mielessä hyväksi, jos käyttäjän tarkkaavaisuus ei tahdotta eksy tarkkailupalkkiin. Joka tapauksessa käyttöliittymän suunnittelussa on tehty kompromissi ensisijaisen toiminnon hyväksi.

Tarkkailupalkki sisältää kaksi komponenttia: hallintapainikkeen sekä tarkkailupaneelin (Kuva 36). Painikkeella käyttäjä hallitsee tarkkailutoiminnon päällä oloa (Periaate 13). Tarkkailun tunnukseksi on valittu ikoni, joka kuvaa kameraa. Hallintapainike on kaksiosainen, joista oikean puoleinen avaa ponnahdusvalikon. Ponnahdusvalikko esittää painikkeen toiminnallisuuden suorasanaisessa muodossa sekä tarjoaa komennon, jolla tarkkailun konfigu-



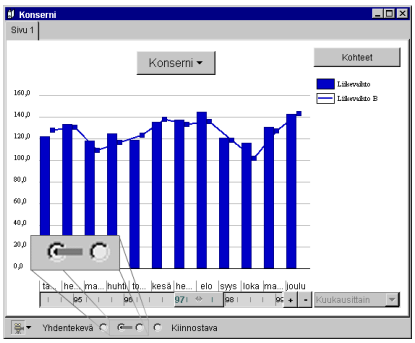
Kuva 37 Konfigurointidialogin tarkkailuvälilehti

Kuva 36 Tarkkailupalkkiin liittyvä ponnahdusvalikko

rointia päästään muokkaamaan. Komento avaa konfigurointidialogin siten, että tarkkailuvälilehti on valittuna (Kuva 37).

Palkin toinen komponentti, tarkkailupaneeli, esittää puolestaan tarkkailun toimintaa ja toimintaperiaatetta ja toisaalta mahdollistaa käyttäjän kontrollin kiinnostavuuden arviointiin. Tieto-objekteja arvioidaan neliportaisella asteikolla, jonka vastakkaisissa päissä ovat arvot yhdentekevä ja kiinnostava.

Kiinnostuksen päättelyssä käytettyä heuristiikkaa havainnollistetaan käyttäjälle, jotta hän voisi ymmärtää ohjelman toimintaperiaatteita. Kiinnostuksen arviointi perustuu aikaan, mikä näytetään paneelissa pienenä animaationa, jossa harmaa palkki kasvaa kohden seuraavaa askelta mitta-asteikolla (Kuva 38). Animaatio etenee pienin askelin välttäen suurta liikettä ja on taustansa kanssa matalakontrastinen, jottei se erottuisi käyttäjää häiritsevästi. Kun palkki saavuttaa seuraavan mitta-asteikon askeleen, askel tulee valituksi. Jos käyttäjä milloin tahansa valitsee jonkin pisteen mitta-asteikolta, automaattinen arviointi lopetetaan kyseisen tieto-objektin kohdalta. Tämä antaa käyttäjälle kontrollin toiminnon suhteen.



Kuva 38 Tarkkailua kuvaava animaatio

7.2.6 Tarkkailussa käytettyjen päättelyiden havainnollistus

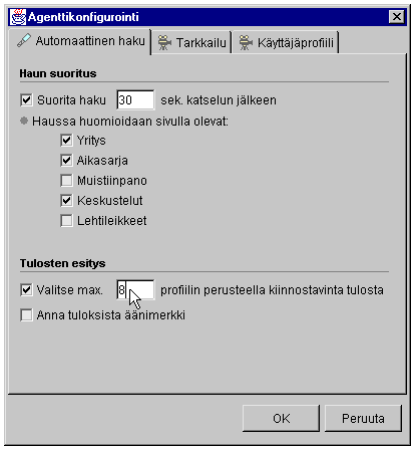
Käyttäjän toimien tarkkailussa tapahtuu kahdenlaista päättelyä: 1) käyttäjän katseluun käyttämästä ajasta päätellään miten kiinnostavana hän tieto-objekteja pitää ja 2) nähtyjen tieto-objektien päätellään olevan yhteydessä lisätietoihin, joista käyttäjä on kiinnostunut. Ensimmäisen päättelyn perusteella rakennetaan käyttäjäprofiilia ja toisen perusteella valitaan automaattisen haun parametrit.

Päättelyprosessi on molemmissa tapauksissa melko suoraviivainen. Kiinnostuksen päättelyssä katseluun käytetyn ajan oletetaan olevan suoraan verrannollinen tieto-objektin kiinnostavuuteen. Siis mitä kauemmin tieto-objektia katsotaan, sitä kiinnostavammaksi kohde tulkitaan. Tosin päättelyä on rajoitettu siten, ettei se arvioi mitään kohdetta täysin yhdentekeväksi tai erityisen kiinnostavaksi. Tällaisten virhepäätelmien todennäköisyys olisi melko suuri ja niiden vaikutus voisi olla haitallinen.

Hakukriteerien päättely on vielä yksinkertaisempi prosessi. Haun muodostamiseksi valitaan yksinkertaisesti näkymästä haun ehdoiksi soveltuvat tiedot ja suoritetaan haku niiden perusteella. Sen, mitkä tieto-objektit soveltuvat haun kriteereiksi, määrittää tiedon tyyppi. Jos tieto on hakuohjelman tunnistamaa tyyppiä, sitä voidaan käyttää haun ehtona.

Hakukriteereinä käytettyjen tietojen valintaa voitaisiin havainnollistaa korostamalla vastaavia käyttöliittymäelementtejä. Tämä tarkoittaisi kuitenkin liikkeen lisäämistä käyttäjän huomion keskipisteessä olevaan näkymään. Jo yksistään liike tällaisessa paikassa olisi todennäköisesti häiritsevää. Korostuksen informaatioarvokin on jatkuvassa käytössä kyseenalainen, sillä huomioidut tietotyypit pysyvät tyypillisesti samoina. Näistä syistä palaute on päätetty jättää antamatta tässä muodossa. Toinen mahdollisuus olisi käyttää tekstimuotoista palautetta tarkkailupalkissa, mutta myös se on hylätty huonon havaittavuutensa perusteella.

Edellä mainituista syistä johtuen käytetyt hakukriteerit julkistetaan vain konfigurointidialogissa, jonka tehtävänä on toimia myös toimintaperiaatteiden selvittäjänä (Periaate 11). Samassa yhteydessä on luontevaa ja helppoa tarjota myös mahdollisuus kontrolloida mitä asioita hakukriteerien valinnassa voidaan käyttää hyväksi. Käyttäjällä on näkyvissään yksinkertainen lista asioista, joita hakukriteerien päättelyssä voidaan käyttää hyväksi (Kuva 39).



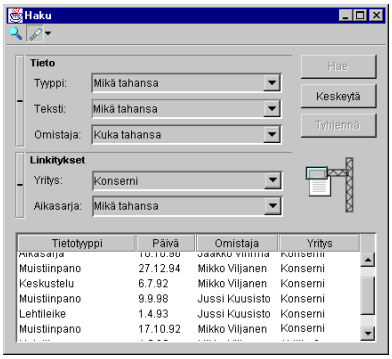
Kuva 39 Mahdollisten hakukriteerien lista

7.2.7 Tulosten esittäminen

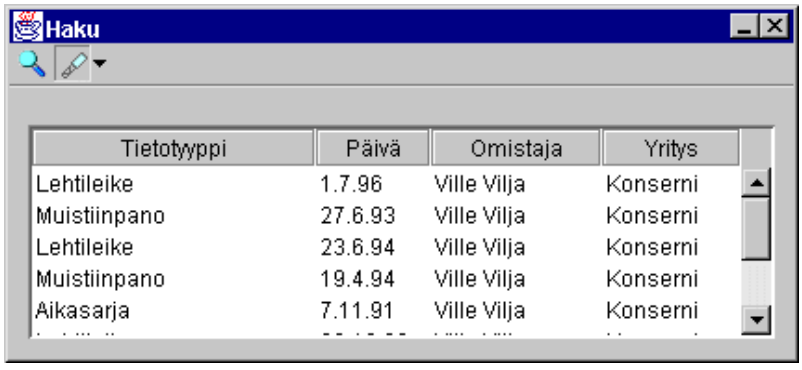
Tulosten esitystapa on merkittävä tekijä epäsuorasti hallitun ohjelman hyödyllisyyden muodostumisessa. Loppujen lopuksi kaikki epäsuoraan hallintaa liittyvät osat ja toiminnot tähtäävät vain käyttäjää hyödyttävien tulosten aikaansaamiseen. Vaikka epäsuora hallinta perustuisi kuinka hienostuneeseen päättelyyn, ovat sen perusteella saadut tulokset kuitenkin ainoastaan käyttäjän todellisten tarpeiden arvailuja. Näin ollen käyttäjälle jää aina tehtäväksi tulosten hyödyllisyyden lopullinen arviointi ja tietenkin myös tulosten hyödyntäminen.

Tulosten tuottaminen ja niiden hyödyllisyyden arviointi on selvästi näkyvillä IMIS-agentin toteuttamassa hakutoiminnossa. Siinä tulokset ovat helposti määriteltävissä ja niille on kohtuullisen helppo keksiä esitysmuoto. Tulokset ovat tieto-objekteja, joten niiden hyödyllisyys riippuu siitä, miten hyvin ne vastaavat käyttäjän mielessä oleviin kysymyksiin tai miten antoisia ideoita käyttäjälle niiden johdosta muodostuu.

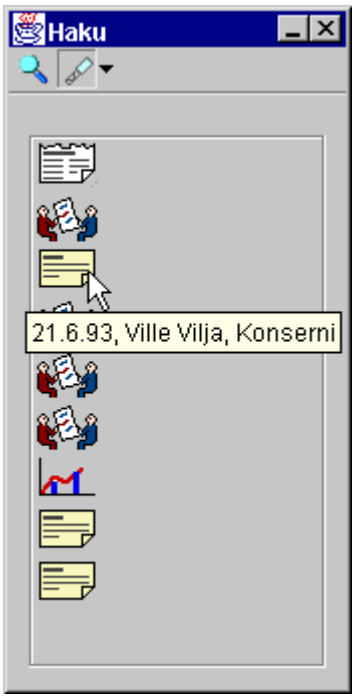
IMIS-agentin käyttöliittymässä tulosten esitysperiaatetta (Periaate 8) toteutetaan mahdollistamalla hakutoiminnallisuuden esittäminen kolmessa eri muodossa (Kuva 40, Kuva 41 ja Kuva 42). Esitysmuodot eroavat toisistaan niiden tarjoaman yksityiskohtaisen informaatiomäärän perusteella. Tarjottu informaatiomäärä vaihtelee täydellisestä haun ja sen tulosten useiden ominaisuuksien kuvaa-



Kuva 40 Automaattisen toiminnon täysi näyttötila



Kuva 41 Automaattisen toiminnon puolikas näyttötila



Kuva 42 Automaattisen toiminnon pieni näyttötila

misesta hyvin niukkaan tuloksen yhden ominaisuuden esittämiseen. Näyttötapa on käyttäjän hallittavissa ja sitä voidaan muuttaa milloin tahansa. Muuttaminen tapahtuu työkalupalkissa olevan ponnahdusvalikon näyttämien valintojen avulla.

Runsaimmassa esitysmuodossa on näkyvissä sekä hakukriteerit että tulokset kaikkine ominaisuuksineen. Ratkaisu vie melko paljon näyttöpinta-alaa, mutta tarjoaa runsaasti tietoa tuloksista, niiden merkityksestä ja niiden hankintaprosessista. Erityisesti haussa käytettyjen kriteerien näkemisen oletetaan auttavan tulosten arvioinnissa. Hakukriteerien merkityksen ymmärtämistä auttaa se, että käyttäjällä on itsellään mahdollisuus tehdä itse muotoilemiaan hakuja samojen kriteerien avulla.

Seuraavaksi suppeampi näyttömuoto on ensimmäisen kanssa muuten sama, mutta siinä hakukriteerit on piilotettu näkyvistä. Näin saadaan aikaan tilan säästöä ja kuvaruudun rauhoittumista, kun hakuprosessin animointi häviää näkyvistä. Kuitenkin käyttäjä saa tulokset samassa tutussa muodossa kuin jos hän olisi määritellyt haun itse. Tuloksista on näkyvissä useita ominaisuuksia, joiden arvellaan auttavan tuloksen hyödyllisyyden arvioinnissa.

Kolmas näyttömuoto poikkeaa kahdesta edellisestä melko radikaalisti (Kuva 42). Siinä tarkoituksena on säästää mahdollisimman paljon näyttötilaa ja rauhoittaa automaattisen toiminnon aiheuttamaa liikettä entisestään. Tulokset esitetään mahdollisimman kompaktissa muodossa ikoneina. Tulosta esittävä ikoni paljastaa tuloksen tietotyypin (muistiinpano, aikasarja jne.) ja muut tulosta kuvaavat ominaisuudet käyttäjä joutuu aktiivisesti ottamaan esille. Ne näytetään työkaluvihjeessä, kun hiiren osoitinta pidetään ikonin päällä.

Esitystavan hallintaan liittyy myös mahdollisuus tuloksista ilmoittavaan äänipalautteeseen. Äänipalautteella voidaan tuloksien tuottamista korostaa, jos käyttäjä tätä jostain syystä haluaa. Oletusarvoisesti tämä toiminnallisuus on kuitenkin vaiennettu. Äänipalautteen hallinnan on katsottu olevan sen verran harvoin tarvittu toiminto, että sen säätely on sijoitettu ainoastaan konfigurointidialogiin.

7.3 Hallinnan toteutus

Ohjelman hallinta jakautuu kahteen osaan: konfigurointiin ja kontrollointiin. Konfigurointi mahdollistaa pitkän aikavälin säätöjen tekemisen. Asetuksilla voidaan muuttaa kaikkia epäsuoraan hallintaan liittyviä toimintoja: niin käyttäjän tarkkailua, päättelyä kuin tehtävien tunnistamistakin.

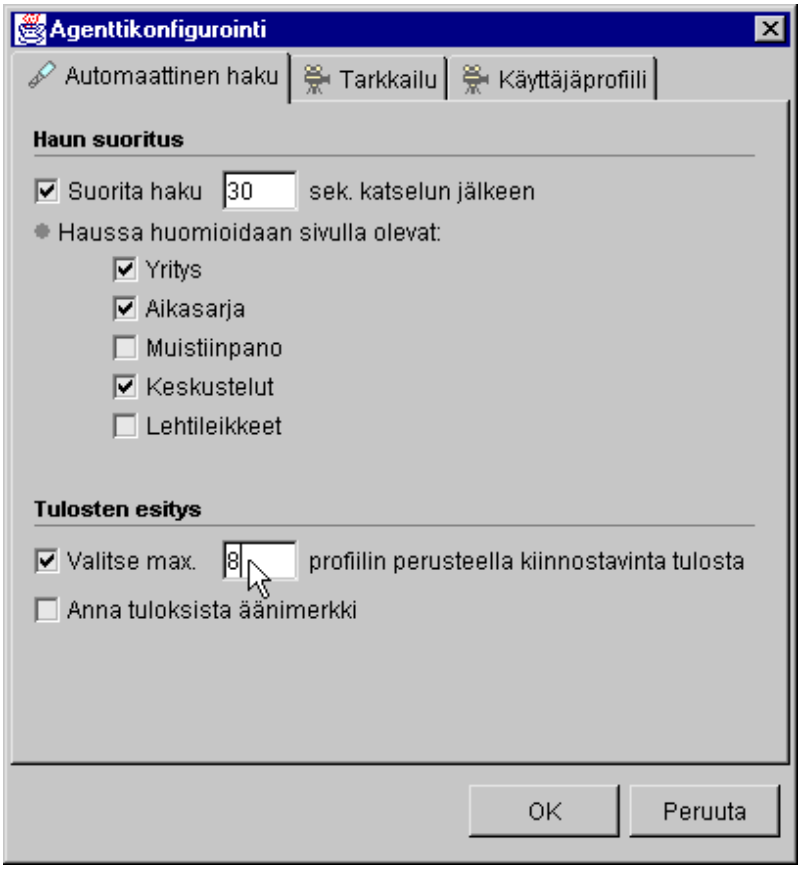
Kontrollointi tarkoittaa välittömästi vaikuttavien komentojen antamista. Edellä on jo tullut kuvatuksi ainakin epäsuorasti lähes kaikki kontrollointimahdollisuudet, joita IMIS-agentin hallitsemiseksi on olemassa. Konfiguraatio-osuuden jälkeen käyttöliittymään luodaan kuitenkin vielä lyhyt katsaus nimenomaan kontrolloinnin mahdollistamisen näkökulmasta.

7.3.1 Konfigurointidialogi

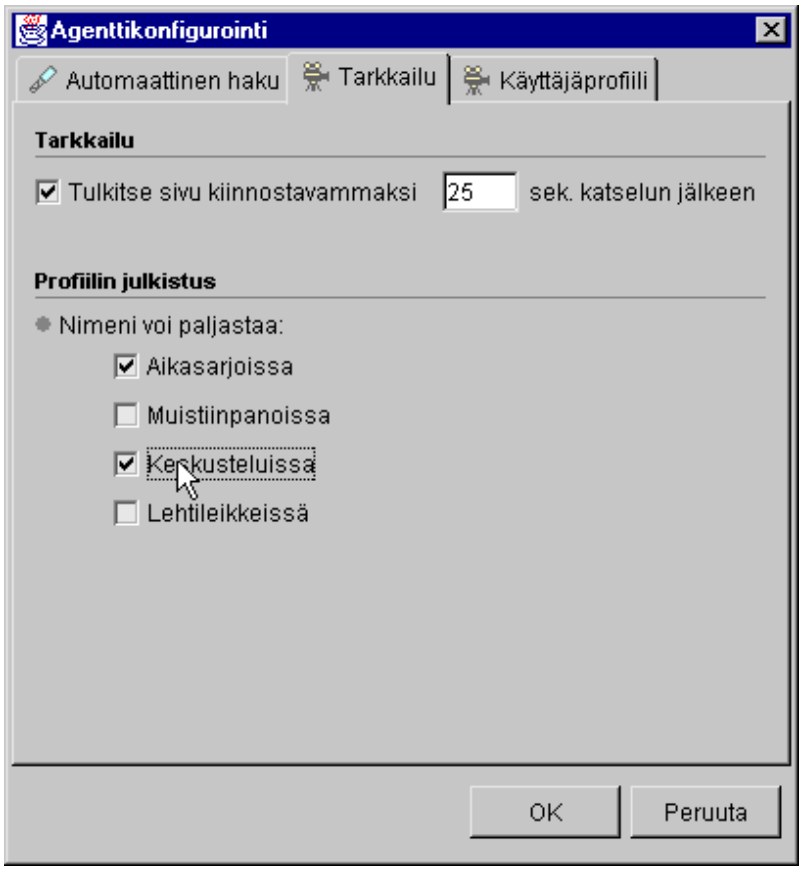
Kaikkien epäsuoraan hallintaan liittyvien toimintojen konfiguroinnin yhteen kokoaminen (Periaate 3) on esimerkkikäyttöliittymässä toteutettu dialogi-ikkunan avulla. Dialogi muodostaa paikan, jossa epäsuoraan hallintaan liittyvät ohjelmanosat ovat keskitetysti näkyvillä.

IMIS-agentissa on epäsuoraan hallintaan liittyviä keskeisiä komponentteja kaksi: käyttäjän tarkkailun toteuttava ja hakutoiminnon automatisoiva osa. Käyttöliittymän on tuettava vähintään näiden kahden osan näkymistä ja niihin liittyvien asetusten säätämistä. Molemmissa voidaan erottaa useita asetuksia, joita käyttäjän tulee voida muuttaa. Tarkkailussa voidaan hallita sekä tarkkailun suorittamista että sitä, mihin tarkkailun tuloksia käytetään. Automaattisessa haussa asetukset kohdistuvat haun parametrointiin ja tulosten esittämiseen. Tämän lisäksi on mahdollista löytää myös muita loogisesti erotettavia toimintakokonaisuuksia, jotka on järkevää esittää käyttöliittymässä omana ryhmänään.

Nykyisten graafisten käyttöliittymien rakentamiseksi on olemassa komponentteja, jotka tukevat esimerkiksi eri järjestelmän osien erottamista toisistaan. Tällaisia ovat esimerkiksi ryhmittely ja välilehtien käyttö. Epäsuoraan hallintaan liittyvien osien tunnistamisen tavoitteena on, että käyttöliittymien mahdollisuuksia myös käytetään järjestelmän todellisen rakenteen paljastamiseksi. Tämä on merkityksellistä erityisesti epäsuoran hallinnan toteuttavassa järjestelmässä, jossa komponentit ovat tyypillisesti käyttäjältä muuten piilossa.



Kuva 43 Automaattiseen hakuun liittyvä konfigurontikäyttöliittymä



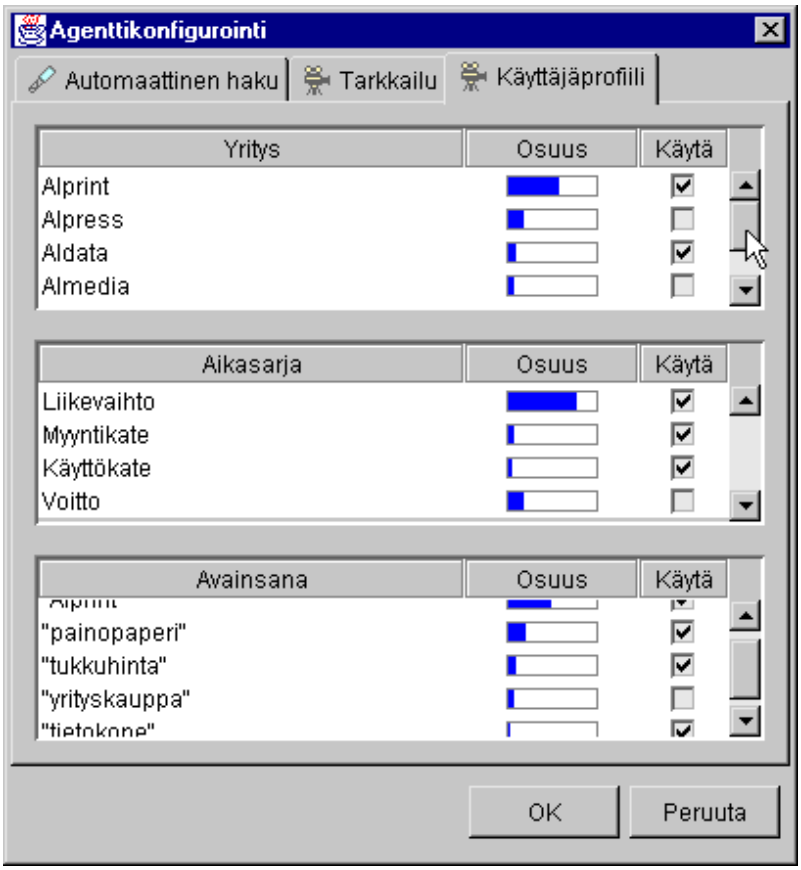
Kuva 44 Tarkkailuun liittyvä konfigurointikäyttöliittymä

Ehdotetussa käyttöliittymässä epäsuoraan hallintajärjestelmään liittyvät osat erotellaan (Periaate 9, Periaate 10) konfigurointidialogissa toisistaan välilehdin (Kuva 43, Kuva 44 ja Kuva 45). Sekä käyttäjän tarkkailusta vastaavalle komponentille että hakutoiminnon automatisoivalle ohjelman osalle on varattu konfigurointikäyttöliittymässä oma välilehti. Näiden lisäksi omalle välilehdelleen on erotettu vielä tarkkailun perusteella muodostettu käyttäjäprofiili. Tietojen suhteita ja ryhmittelyä kuvataan välilehdille sijoitetuilla ikoneilla. Kuvat ryhmittelevät ‘Tarkkailu’ ja ‘Käyttäjäprofiili’ -välilehdet samaan kategoriaan.

7.3.2 Tarkkailun konfigurointi

Ensimmäinen konkreettinen asetus, jota käyttäjän tarkkailuun liittyen on mahdollista säätää, liittyy kiinnostuksen päättelyyn. Päättely perustuu katseluun käytetyn ajan mittaamiseen ja tämän tiedon liittämiseen käyttäjän näkyvissä oleviin tieto-objekteihin. Konfigurointidialogissa käyttäjä voi muuttaa tätä aikarajaa, joka tulkitaan kiinnostuksen osoitukseksi katsottua tieto-objektia kohtaan. Esimerkkikuvassa (Kuva 44) aika on määritelty 25 sekunniksi, eli jokaisen katseluun käytetyn 25 sekunnin katsotaan merkitsevän kiinnostuksen osoittamista katsottuja tieto-objekteja kohtaan.

Käyttäjän on mahdollista kontrolloida myös sitä, millaisissa yhteyksissä hänen nimensä voidaan paljastaa järjestelmän toisille käyttäjille. Käyttäjäprofiiliin voi kerääntyä sellaista tietoa käyttäjän toimintatavoista, jonka hän ei halua leviävän yleisempään tietoon. Vaikka suoranaisia käyttötapoja tai edes käyttöhistorian osia ei



Kuva 45 Käyttäjäprofiilin esitysmuoto konfigurointidialogissa

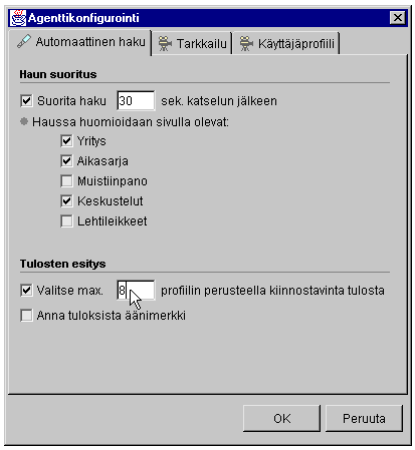
käyttäjäprofiiliin tallenneta, sen sisältämän tiedon avulla voidaan tällaisiakin asioita päätellä, jos ei ohjelmallisesti, niin ainakin ihmisten toimesta. Käyttäjän yksityisyyden suojaamiseksi on siten tärkeää mahdollistaa julkistettujen tietojen hallinta. IMIS-agentissa se tapahtuu tietotyypeittäin. Käyttäjän on mahdollista estää nimensä julkistaminen tiettyjen tietotyyppien, kuten esimerkiksi muistiinpanojen, kohdalla.

Ohjelman keräämä käyttäjäprofiili vaikuttaa kiinnostuksen kohteita jakavien käyttäjien etsintään ja tulosten suodattamiseen ja järjestämiseen. Koska käyttäjäprofiili vaikuttaa oleellisesti IMIS-agenttiin liittyvän hakutoiminnon tuloksiin, on ohjelman toiminnan ymmärtämiseksi oleellista päästä käsiksi tuohon käyttäjäprofiiliin. Eikä riitä yksin se, että käyttäjä voi profiilinsa nähdä, vaan hänelle on tarjottava myös mahdollisuus muokata sitä. Luottamus ohjelman hyödyllisyyteen nimittäin vahingoittuu varmasti, jos käyttäjä tietää sen toimivan vääristyneen käyttäjäprofiilin nojalla. Profiilin näkeminen on tärkeä tekijä luottamuksen rakentamisessa järjestelmän toimintaa kohtaan [Cook and Kay 94].

Käyttäjäprofiilin muokkaaminen tapahtuu IMIS-agentissa siten, että käyttäjällä on mahdollisuus lisätä ja poistaa profiilin huomioimia tekijöitä. Profiili esitetään käyttäjälle jaoteltuna sovellusalalla oleellisiin käsiteryhmiin, joita ovat: yritykset, aikasarjat ja avainsanat (Kuva 45). Yritysryhmään kerätään käyttäjää kiinnostavia yrityksiä ja vastaavasti aikasarjaryhmään kuuluu käyttäjän yleisimmin katsomat aikasarjat. Käyttäjän kiinnostusta kuvaavia avainsanoja kerätään järjestelmän tekstimuotoisesta tiedosta, kuten muistiinpa-

noista ja keskusteluista. Jokaiseen kiinnostukseen vaikuttavaan yksittäiseen entiteettiin (esim. yksittäinen yritys) liittyy sen paino kyseisen käsiteryhmän sisällä. Tämä kuvataan käyttäjälle graafisesti sinisellä palkilla. Konfiguroinnissa käyttäjällä on mahdollisuus määritellä mitkä entiteetit kustakin käsiteryhmästä hänen käyttäjäprofiilissaan huomioidaan.

7.3.3 Automaattisen haun konfigurointi



Kuva 46 Automaattisen haun konfiguroinnin mahdollistava dialogi

Ehkä merkittävin ja käyttäjää kiinnostavin konfigurointikohde automaattisen haun hallitsemiseksi on sen määritteleminen, mitä asioita haussa otetaan huomioon. Automaattinen haku joutuu toimimaan käyttäjän toimien muodostamassa ympäristössä, eikä tämä ympäristö välttämättä tarjoa esimerkiksi selkeää hakukriteeriä yrityksen löytämiseksi. Tällaisessa tapauksessa ohjelman on toimittava niiden tietojen varassa, jotka sillä hetkellä ovat saatavilla. Käyttäjän on kuitenkin mahdollista asettaa ohjelman toiminnalle rajat. Hän voi määritellä mitkä asiat ylipäätään otetaan haussa huomioon, jos niiden huomioiminen annetussa yhteydessä on mahdollista (Kuva 46). Haussa mahdollisesti huomioitavat kriteerit eli automaattisen haun toimintaan vaikuttavat tekijät voidaan yksinkertaisesti identifioida hakuohjelman käyttöliittymän avulla.

Toinen ominaisuus, jota käyttäjä voi muuttaa, on haun suoritusajankohta. IMIS-agentissa haun suoritusajankohta on aina sidottu käyttäjän suorittamiin toimiin, tarkemmin sanottuna siihen hetkeen, jolloin käyttäjä ottaa esille uusia tieto-objekteja. Haku voi alkaa koska tahansa tämän hetken jälkeen ja on luontevaa antaa käyttäjälle päätäntävaltaa tähän asiaan. Hakutoiminnon konfiguroinnissa käyttäjä voi määritellä tieto-objektin näkymisajan, jonka jälkeen haku suoritetaan.

Kolmanneksi käyttäjälle annetaan mahdollisuus hallita sitä, kuinka monta tieto-objektia hänelle esitetään haun tuloksena. Automaattisen haun tuloksia tuotetaan käyttäjän rinnalla väsymättä, joten ajan myötä niitä kertyy paljon. Jos käyttäjä haluaa hyödyntää tuloksia, hänen on arvioitava niiden merkitystä yksitellen. Työn rasittavuutta voidaan pienentää vähentämällä käyttäjälle esitettyjen tulosten määrää. Osa tulosten suodatustyöstä tulisi siis siirtää tietokoneen tehtäväksi. Se kykeneekin tekemään mielekästä suodattamista perustuen kerättyyn käyttäjäprofiiliin.

Myös neljäs hakutoiminnallisuuden konfigurointimahdollisuus liittyy tulosten käsittelyyn. Käyttäjien erilaisista kognitiivisista tyyleistä, työskentelytavoista ja muista yksilöllisesti vaihtelevista muuttujista johtuen on tarpeen mahdollistaa tulosten esitystavan muokkaaminen. Näin toteutettuna ohjelma tukee todennäköisesti useamman käyttäjän tapaa työskennellä. IMIS-agentissa käyttäjä voi valita annetaanko automaattisen haun tulosten valmistumisesta äänipalaute.

7.3.4 Konfigurointi toimintaperiaatteiden esittämisessä

Edellä kuvatuissa konfigurointimahdollisuuksissa on mukana muutamia sellaisia asetuksia, joita käyttäjät eivät todennäköisesti ole kiinnostuneita muuttamaan. Esimerkiksi aika, joka tulkitaan kiinnostuksen osoitukseksi tai aika jonka jälkeen haku suoritetaan ovat tällaisia. Nämä asetukset onkin otettu konfiguraatioon mukaan enemmän siksi, että ne kertovat jotain oleellista ohjelman

toiminnasta kuin niiden muuttamistarpeen vuoksi (Periaate 11). Esimerkiksi sen ymmärtäminen, että ohjelma päättelee kiinnostusta katseluun käytetyn ajan perusteella, auttaa ehkä sietämään sen toisinaan tekemiä virheitä ja antaa käyttäjälle mahdollisuuden virheiden korjaamiseen.

Asetuksista, joiden pääasiallinen tarkoitus on välittää tietoa ohjelman toimintaperiaatteista, ei ole välttämätöntä tehdä muutettavia. Olisihan yksinkertaisesti mahdollista esittää sama asia tekstinä. Tämä olisi yksinkertaista toteuttaa, mutta käyttäjälle mahdollisesti turhauttavaa. Ensinnäkin vaikuttaa siltä, etteivät käyttäjät tyypillisesti ole kiinnostuneita lukemaan opastekstejä, jollaiseksi tietyn toiminnallisuuden toiminnan selittäminen muodostuisi. Toiseksi tällaisen tiedon esittäminen käyttöliittymässä, jonka tarkoituksena on mahdollistaa asetusten muuttaminen, ei ole kovin johdonmukaista. Muutettavan tiedon merkityksen selvittäminen on käyttäjälle oletettavasti motivoivampaa kuin pelkän selitystekstin lukeminen.

7.3.5 Kontrolli

Kuten edellä nähtiin konfigurointi toteutetaan omaan dialogiikkunaansa, joka mahdollistaa kohtuullisen runsaan tilan käytön käyttöliittymän suunnittelussa. Tämä oleellisesti helpottaa käyttöliittymän suunnittelua, kun rajoittavia tekijöitä on yksi vähemmän. Tilan vapaampi käyttö mahdollistaa mm. eri komponenttien spatiaalisten suhteiden runsaamman hyväksikäytön käyttöliittymän suunnittelussa.

Jatkuvasti näkyvissä olevaksi kontrollointitavaksi valittiin koko toiminnon päälläolon hallinta. On erityisen tärkeää pitää jatkuvasti näkyvät toiminnot yksinkertaisina, jotteivät ne vie paljoa tilaa ja että niiden merkityksen opettelu ei vaatisi liikaa vaivaa. Periaatteena on pitää hallintapalkit niin yksinkertaisina kuin mahdollista ja tarjota runsaampi toiminnallisuus vasta konfigurointidialogissa.

Tarkkailupalkin tarjoamat kontrollointivälineet ovat täsmälleen periaatteen mukaiset. Palkissa on ainoastaan yksi painike, jonka yhteydessä olevalla ponnahdusvalikolla kontrollointimahdollisuuksia on lisätty. Ponnahdusvalikko toistaa napin toiminnon sekä mahdollistaa konfigurointidialogin avaamisen. Käyttäjä voi hallita myös tieto-objekteihin liittyvää kiinnostuksen arviota.

Myös hakuohjelman työkalupalkissa on ponnahdusvalikko, jolla hallitaan automaattisesti toimivan hakutoiminnon näyttötilaa. Tässä periaatetta ei ole noudatettu orjallisesti, sillä näyttötilan muuttamisen on katsottu olevan niin yleinen toiminto, että sen sijoittaminen konfigurointidialogiin olisi tuntunut sen piilottamiselta. Toiminnonhan on tarkoitus mahdollistaa hakuprosessin animoinnin nopea piilottaminen, jos se alkaa käyttäjästä tuntua häiritsevältä. Tällaisen toiminnon on oltava nopeasti saatavilla, tai muuten toiminto pysäytetään lopullisesti.

7.4 Suunnitteluperiaatteiden yleistettävyys

Suunnitteluperiaatteiden soveltuvuutta on edellä arvioitu ainoastaan yhteen epäsuorasti hallittuun toimintoon yhdellä sovellusalueella. Täten on epäselvää miten hyvin ratkaisu on yleistettävissä

muille sovellusalueille ja erilaisille epäsuorasti hallituille ohjelmille. Konsepti pitää kuitenkin sisällään joitain selviä laajenemisreittejä, joiden merkitystä seuraavassa arvioidaan.

Esitetty käyttöliittymäkonsepti perustuu siis ensinnäkin epäsuoraan hallintaan osallistuvien toiminnallisten kokonaisuuksien tunnistamiseen ja niiden tekemiseen näkyviksi. Toinen peruspilari konseptissa antaa ohjeen epäsuorasti hallitun toiminnon havainnollistamiseksi käyttöliittymässä vaatien sen toteuttamaan perinteisen käyttäjän syötteeseen perustuvan käyttöliittymän. Tämä käyttöliittymä muutetaan tiettyjen periaatteiden avulla automaattisesti toimivaksi eli epäsuorasti hallituksi.

Esitetyssä ratkaisussa konfigurointidialogi edustaa koko epäsuoran hallinnan toteuttavaa järjestelmää. Se on paikka, jossa keskitetysti on saatavissa kaikki informaatio ohjelman osista ja niiden toiminnoista. Tällainen ratkaisu kuitenkin olettaa, että yhdessä järjestelmässä on vain yksi epäsuoraa hallintaa toteuttava alijärjestelmä. Toisin sanoen tällainen käyttöliittymän järjestely ei ole käyttökelpoinen sovelluksissa, joissa on oleellista erottaa useampia epäsuoraan hallintaan liittyviä alijärjestelmiä toisistaan. Järjestelmä, jossa hallitaan esimerkiksi kahta toimintoa epäsuorasti, mutta hyvin erilaiseen logiikkaan perustuen, voisi olla esimerkki tällaisesta systeemistä.

Tämä ei kuitenkaan tarkoita sitä, etteikö esitetty konsepti yleistyisi helpostikin ohjelmiin, joissa on kaksi tai useampia epäsuorasti hallittuja toimintoja. IMIS-agentissahan vain hakutoimintoa ohjataan epäsuorasti, mutta sen rinnalle voidaan kuvitella vaikkapa automaattinen dokumenttien järjestelytoiminto. Koska toiminnot perustuvat samalle käyttäjäprofiilille ja samanlaiseen käyttäjän toimien tarkkailuun, se on helppo lisätä käyttöliittymään.

Uudelle epäsuorasti hallitulle toiminnolle lisätään vain uusi välilehti konfigurointidialogiin ja sille suunnitellaan tavallinen käyttöliittymä, jossa sen toiminnallisuus on käyttäjän itsensä käytettävissä. Kun tämä käyttöliittymä varustetaan vielä automaattisen tilan hallinnan mahdollistavilla toiminnoilla, työ on valmis. Ainoa ero esimerkkinä esitettyyn tilanteeseen on se, että nyt epäsuorasti ohjataankin kahta eikä vain yhtä toimintoa. Sama pätee myös useammalle epäsuorasti ohjatulle toiminnolle.

Jossain vaiheessa muodostuu ongelmaksi se, että vaikka automaattisesti toimivien ohjelmien käyttöliittymät voitaisiin asettaa pieneen näyttötilaan, ne veisivät liiaksi tilaa siitä huolimatta. Tällaisessa tapauksessa on kuitenkin jo aiheellista kysyä, onko niin moni automaattinen toiminto tarpeellinen. Jos on, käyttöliittymän toteuttamiseen on kehitettävä toinen menetelmä.

Toisaalta myöskään käyttäjän tarkkailun monipuolistaminen ei ole sinänsä ongelmallista. Tarkkailupalkki on paikka kaikille uusille tarkkailusta kertoville komponenteille. Riippuen siitä, mitä käyttäjä kulloinkin on tekemässä, tarkkailu kohdistuu eri asiaan. Tämä viestitetään käyttäjälle tarkkailupalkin avulla, jossa toimintaa osoittavat vain ne komponentit, joiden kuvaamaa tarkkailua kyseisessä tapauksessa suoritetaan.

Tarkkailupalkin käyttö ei kuitenkaan ole riittävä kaikille sovelluksille. Sen mahdollisuus kertoa tarkemmin siitä, mihin yksityiskohtaan käyttöliittymässä se päättelynsä perustaa on liian rajallinen. Jos ajatellaan esimerkiksi Metamouse-prototyyppiä [Maulsby and

Witten 93], lienee selvää, että tarkkailupalkin käyttäminen havaintopisteitä kuvaavien nupien tilalla selvästi heikentäisi käyttöliittymän palautetta. Tarkkailupalkki soveltuu palautteen antamiseen siis vain tapauksissa, joissa seurannan tarkka kohde ei ole ohjelman toimintalogiikan kannalta erityisen keskeinen. IMIS-agentin kohdalla hakukäyttöliittymän ja konfigurointidialogin on ajateltu riittävän yksityiskohtaisen tiedon välittämiseen siitä, mitkä asiat tarkkailun kautta hakuun vaikuttavat.

8. LOPUKSI

8.1 Yhteenveto

Arvioitaessa työn tuloksia näin jälkikäteen, on paikallaan nostaa mieliin, millainen suunta työlle alussa annettiin. Työn tavoitteiksihan asetettiin:

1. Muodostaa selkeä käsitys siitä, mitä on epäsuora hallinta.
2. Analysoida käsitettä käyttöliittymän kannalta ja esittää sen käyttöliittymälle asettamia tavoitteita ja vaatimuksia.
3. Kartoittaa nykyisiä käyttöliittymäratkaisuja ja arvioida niitä.
4. Muotoilla edellä mainitut pohdinnat ymmärrettävään ja sovellettavaan muotoon.

Listassa tavoitteet on lueteltu järjestyksessä siten, että jäljempänä mainitun tavoitteen saavuttaminen edellyttää aiempien tavoitteiden saavuttamista. Eräässä mielessä tutkimuksen tuloksen voisi siis ajatella kiteytyvän viimeiseen tavoitteeseen vastaamisessa. Viimeisen tavoitteen toteutumista ei silti pidä erehtyä nostamaan työn ainoaksi tulokseksi. Aiempien tulosten soveltaminen voidaan tehdä myös toisin, jolloin viimeisen tavoitteen vastauskin muuttuu. Vastaukset jokaiseen tavoitteeseen ovat siis koko työn tuloksia.

Jos yksinkertaisuuden vuoksi kuitenkin pitäydytään tiedon kiteytymisen ajatuksessa, työn soveltavassa osuudessa esitettiin ratkaisu epäsuoran hallinnan käyttöliittymän rakentamiseksi, joka on periaatteeltaan yksinkertainen: epäsuoraa hallintaa tukevan käyttöliittymän tulee perustua konventionaaliseen käyttöliittymään, joka mahdollistaa myös toiminnon manuaalisen käytön.

Ratkaisumalli ehkäisee sellaisia toiminnallisuuksia, joita käytettäisiin ainoastaan epäsuorasti halliten. Käyttäjältä piilotetut toiminnot muodostavat suurimmat käytettävyysongelmat epäsuorasti hallittujen ohjelmien kohdalla. Jos ohjelman toiminnallisuutta ei ole selvästi havainnollistettu esimerkiksi visualisoinnilla, sen toimintaperiaatteita on hyvin vaikea oppia ja omaksua, jolloin myös ohjelman kontrollointi vaikeutuu.

Konventionaaliseen käyttöliittymään perustuva ratkaisu paljastaa paitsi ohjelman toimintaperiaatteet myös epäsuoran vuorovaikutustavan hallintatavan. Epäsuora hallinta nähdään ratkaisumallissa vaihtoehtoisena vuorovaikutustapana, jonka käyttäjä voi niin halutessaan kytkeä toimintaan. Epäsuoran hallinnan käyttöönotto esitetään käyttäjälle automaattisen toiminnon käynnistämisenä, jolla on selvät visuaaliset seuraukset ohjelman käyttöliittymälle.

Näin käyttäjän on helpompi hahmottaa toiminnon käyttötarkoitus ja siihen liittyvän autonomisuuden toiminta-ala.

Käyttöliittymän on otettava kantaa epäsuoran hallinnan autonomiseen luonteeseen. Epäsuora hallinta tarkoittaa oleellisesti sitä, että ohjelma suorittaa *itseäisesti* toimintoja *käyttäjän rinnalla*, hänen epäsuoran hallinnan alaisuudessa. Ohjelmassa suoritetaan siis samanaikaisesti vähintään kahta toimintoa: käyttäjän sekä ohjelman itsenäisesti suorittamaa. Käyttäjän toiminnot on tässä työssä luettu primäärisiksi toiminnoiksi, joita ohjelman toiminta pyrkii tukemaan.

Ohjelman itsenäisesti suorittamat toiminnot voivat häiritä käyttäjän keskittymistä, jos ne havainnollistetaan liian suurieleisesti. Ongelmallista tämä on etenkin silloin, kun ohjelma suorittaa toimintonsa käyttäjän kanssa samanaikaisesti. Työssä ehdotettu ratkaisumalli ottaa ongelman huomioon tarjoamalla epäsuorasti hallitulle toiminnolle useita vaihtoehtoisia näyttömuotoja. Informatiiviset näyttömuodot vievät paljon näyttöpinta-alaa, mutta kykenevät näyttämään monipuolisesti ohjelman toimintatavan.

Suuri näyttöpinta-ala yhdistettynä monipuoliseen toiminnallisuuden visualisointiin muodostaa toisiin tilanteisiin liian levottoman käyttöliittymän, mitä ehkäistään pienemmillä näyttömuodoilla. Niissä näyttöpinta-alan käytön optimoinnin lisäksi onkin otettu huomioon käyttöliittymässä näkyvän liikkeen vähentäminen. Näin ohjelman itsenäinen toiminta ei riko käyttäjän huomiota tarpeettomasti. Käyttäjä itse kontrolloi sitä tapaa, jolla automaattinen toiminta esitetään.

Esimerkkikäyttöliittymä perustuu neljääntoista suunnitteluperiaatteeseen, jotka soveltuvat suunnittelutyön avuksi. Periaatteet on muodostettu työ aikaisempien vaiheiden tulosten perusteella. Niissä yhdistyy sekä vaatimusanalyysin että nykyisten ratkaisumallien kartoituksesta saatu tieto. Periaatteet esittävät tiedon rajoittuneessa, mutta suunnitteluprosessia ajatellen riittävän konkreettisessa muodossa.

Työn soveltavaa osuutta pohjustettiin laajalla tutkivalla osuudella, jossa muodostettiin teoriaa siitä, mitä epäsuora hallinta on ja millaisia tavoitteita ja vaatimuksia se käyttöliittymälle asettaa. Näiden lisäksi tutkittiin jo rakennettujen prototyyppijärjestelmien avulla ratkaisuja, joita epäsuorasti hallittujen toimintojen käyttöliittymissä on käytetty.

Tutkivan osuuden tärkeimpänä tehtävänä oli muodostaa yhtenäinen ja ymmärrettävä käsitys epäsuorasta hallinnasta ja siihen liittyvästä käyttöliittymästä. Kirjallisuudesta ei ole löydettävissä vastaavaa kattavaa kokonaisnäkemystä epäsuorasta hallinnasta käyttöliittymän näkökulmasta. Alalla tehtävä tutkimus on ainakin toistaiseksi hajanaista, pääosin erilaisten prototyyppien kuvausta.

8.2 Työn asettaminen kontekstiin

Työn tuloksia on kiinnostava tarkastella muiden soveltuvien tutkimusalueiden käsitteiden valossa, mikä asettaisi ratkaisun kontekstiin muun tutkimuksen kanssa. Tarkoitukseen soveltuvia tutkimussuuntauksia ovat jo työn alussakin esitellyt älykkäät käyttöliittymät ja agenttiohjelmat. Molempien tarjoamat jäsennystavat

auttavat arviomaan käyttöliittymiä, joissa käytettävyyttä pyritään parantamaan lisäämällä automaattisen toiminnallisuuden osuutta.

Ensimmäinen tutkimussuuntaus, jota vertailussa käytetään on älykkäät käyttöliittymät. Mayburyn [Maybury and Wahlster 98] näkemyksessä älykkään käyttöliittymän osat perustuvat eksplisiittisiin malleihin käyttäjästä, sovellusalueesta, vuorovaikutuksesta, tehtävistä ja käytettävissä olevista medioista. Ajatuksen mukaan mallien perusteella voidaan jokaisessa tilanteessa hyödyntää parhaalla mahdollisella tavalla käytettäviä resursseja suhteessa annettuun ongelmaan. Malleihin perustuvaa käyttöliittymää ei siten rakenneta staattiseksi, vaan se muuntuu tilanteiden mukaan.

Se, ettei tämä työ anna suurta panosta älykkäiden käyttöliittymien tutkimukseen Mayburyn esittämässä mielessä, johtuu painotuserosta. Mayburyn malli korostaa voimakkaasti käyttöliittymän toiminnallisuuden mahdollistavia tekniikoita. Tämän tutkimuksen ytimessä on puolestaan ollut erityisesti havaittava käyttöliittymä ja sen dynaaminen toiminta. Toisin sanoen, tämä tutkimus vastaa kysymykseen, jota Maybury ei suoraan nosta esiin: miten älykkään käyttöliittymän mahdollistama tekniikka ja sen toimintaperiaatteet havainnollistetaan käyttäjälle.

Toinen mahdollinen tulkinta on se, että älykkäiden käyttöliittymien malli on itseasiassa toinen ja vaihtoehtoinen teoria siitä, mitä epäsuora hallinta on. Näin ajatellen sekä älykkäiden käyttöliittymien tutkimus että tämä työ vastaavat ainakin osittain samaan kysymykseen: mitä on epäsuora hallinta. Tarjotut näkemykset ovat hyvin erilaisia ja soveltuvat eri tapauksissa käytettäviksi.

Agenttiohjelmien tutkimus on toinen keskeinen vertailukohta tälle työlle. Agenttitutkimus ei tunnu yhtä selvästi jäsentyneeltä, kuin älykkäiden käyttöliittymien tutkimus, koska tutkijayhteisö ei ole tuottanut samanlaista yleistä näkemystä sen suunnasta. Niinpä tutkimusalan suunta on nähtävillä ainoastaan pieninä sirpaleina eri artikkeleissa ja tutkimusraporteissa. Tämä myös näkyy sen tutkimuksen kirjona, jota nimikkeen agenttitutkimus alla tehdään.

Kokonaisnäkemyksen jonkinlaisesta jäsentymättömyydestä huolimatta jotain yleistä agenttitutkimuksestakin voidaan sanoa. Yleisen suuntauksen mukaan ohjelmilta vaaditaan tiettyjä ominaisuuksia, jotta niitä voidaan kutsua agenteiksi. Esimerkiksi autonomisuus, kommunikointi, oppiminen ja tapahtumiin reagointi ovat sellaisia.

Ominaisuusluettelo paljastaa tutkimusalan suunnan ja menetelmät. Tavoitteena on tehdä tietokoneohjelmista itsenäisempiä ja siis käyttäjästä riippumattomampia. Keinot tavoitteen saavuttamiseksi keskittyvät tässäkin, kuten älykkäissä käyttöliittymissäkin, erityisesti ominaisuuksia toteuttavien tekniikoiden kehittämiseen ja niiden uudelleen soveltamiseen. Tämä pitää tyypillisesti paikkansa, vaikka agenttiohjelman tavoitteena olisikin ohjelman ja käyttäjän vuorovaikutuksen vähentäminen ohjelman autonomisuutta lisäämällä.

Tämä tutkimus pureutuu juuri tuohon agenttiohjelmien tyypilliseen ongelmaan. Niiden kehitys on tähän mennessä keskittynyt tekniikan hiomiseen käyttöliittymän suunnittelun kustannuksella. Myös johtava agenttitutkija Pattie Maes yhdessä tunnetun käyttöliittymätutkijan (Shneiderman) kanssa ovat tätä mieltä [Maes and Shneiderman 97]. Heidän näkemyksessään agenttiohjelmien käyt-

töliittymät ovat keskeisessä asemassa ohjelmien hyödyllisyyden muodostumisessa.

Agenttitutkimuksen kannalta tämä työ vastaa siis agenttiohjelman käyttöliittymän suunnittelua koskeviin kysymyksiin, mutta varsinaisten agenttitekniikoiden kehittämiseen ei oteta kantaa. Toisaalta voidaan ajatella koko agenttiohjelman käsitteen olevan vailla merkitystä, jos sellaisten ohjelmien käyttäminen on vaikeaa tai hämmentävää.

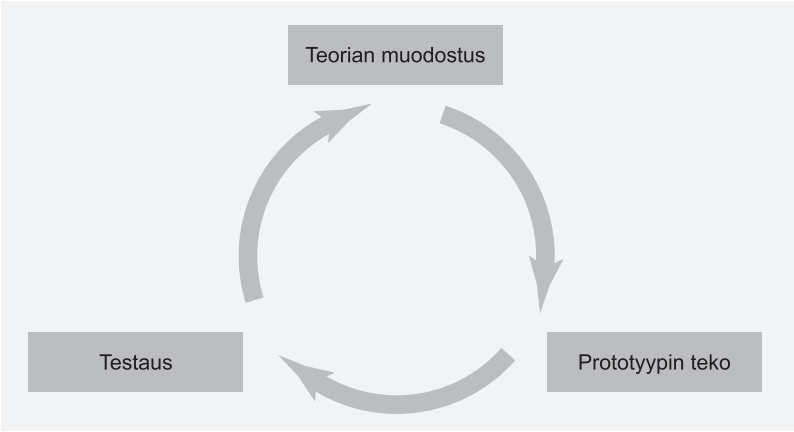
Agenttiohjelman erottaa tavallisesta nimenomaan se kieli tai tässä yhteydessä mieluummin vuorovaikutustapa, jolla ohjelmaa käytetään. Agenttitekniikat pyrkivät mahdollistamaan epäsuoran ja käyttäjän tietoisiin toimintoihin vähemmän perustuvan ohjelman hallinnan. Tämän näkemyksen mukaan agenttiteknologian perimmäisenä tavoitteena olisi ohjelman ja käyttäjän välisen vuorovaikutuksen helpottaminen, mistä näkökulmasta on outoa, että agenttiohjelmien käyttöliittymien suunnittelu ei ole merkittävä osa agenttitutkimusta.

Sen lisäksi, että tämä työ antaa osansa agenttiohjelmien käyttöliittymien tutkimukseen, tässä esitelty näkökulma voi palvella myös uutena tapana ajatella agenttiohjelmia. Työn keskeinen käsite, epäsuora hallinta vuorovaikutustapana, tarjoaa mahdollisuuden muodostaa uudenlaisen näkemyksen agenttiohjelmista. Sen mukaan agenttiohjelmaksi voidaan määritellä sellainen ohjelma, joka käyttää vuorovaikutustapanaan epäsuoraa hallintaa. Tämä muuttaisi agenttitutkimuksen luonnetta siirtämällä sen painopistettä lähemmäksi käyttäjää.

8.3 Jatko

Käyttöliittymätutkimuksen teorian muodostusta voitaisiin ajatella kolmeportaisena iteratiivisena prosessina (Kuva 47), jossa alustavaa teorian muodostuksen vaihetta seuraa prototyypin toteutus. Kolmannessa vaiheessa teorian toimivuutta testataan prototyypin avulla. Testauksen tulokset ovat polttoainetta seuraavalle iteraatiokierrokselle antaen palautetta ja ajatuksia teorian muokkaamista varten.

Jos tämä työ asetetaan tällaiseen malliin, lienee melko selvää, että



Kuva 47 Eräs näkemys käyttöliittymäteorian muodostuksesta

edes yhtä iteraatiokierrosta ei ole vielä saatettu päätökseen. Työssä on kuvattu alustava teoreettinen malli ensin epäsuoraa hallintaa ja siihen liittyvää käyttöliittymää kuvaavan käsitteistön muodossa. Toiseksi käsitteistöä tai teoriaa konkretisoitiin esittämällä se suunnitteluperiaatteina. Teorian muuttaminen prototyypin muotoonkin on alkanut. Esitelty prototyyppi ei kuitenkaan ole testausta ajatellen riittävä, vaan vaatisi lisätoiminnallisuutta.

Niinpä tutkimuksen jatko sisältää aluksi prototyypin saattamisen testattavaan kuntoon, jolloin päästään prosessin kolmanteen vaiheeseen. Käyttäjäkokeiden tulosten perusteella voitaisiin sitten harkita uuden iteraatiokierroksen tarpeellisuutta.

9. LÄHTEET

[André *et al.* 98]
André, E., Rist, T. and Müller, J.: Guiding The User Through Dynamically Generated Hypermedia Presentations with a Life-Like Character. In: *Proceedings of the International Conference on Intelligent User Interfaces* (IUI'98), pp. 21–28. San Francisco, California, January 1998.

[Ball *et al.* 97]
Ball, G., Ling, D., Kurlander, D., Miller, J., Pugh, D., Skelly, T., Stankosky, A., Thiel, D., Van Dantzich, M. and Wax, T.: Lifelike Computer Characters: The Persona Project at Microsoft Research. In: Bradshaw, J. (ed.): *Software Agents*, pp. 191–222. The MIT Press, Cambridge, Massachusetts, 1997.

[Bolt 80]
Bolt, R.: "Put-That-There": Voice and Gesture at the Graphics Interface. In: *ACM Computer Graphics*, pp. 262–270, Vol. 14, No. 3, July 1980.

[Brenner *et al.* 98]
Brenner, W., Zarnekow, R. and Wittig, H.: *Intelligent Software Agents: Foundations and Applications*. Springer-Verlag, Berlin, 1998.

[Broadbend 87]
Broadbend, D.: *Perception and Communication*. Oxford University Press, New York, 1987.

[Caglayan and Harrison 97]
Caglayan, A. and Harrison, C.: *Agent Sourcebook*. John Wiley & Sons, Inc., New York, 1997.

[Card *et al.* 83]
Card, S., Moran, T. and Newell, A.: *The Psychology of Human-Computer Interaction*. Lawrence Erlbaum Associates, Inc. Publishers, Hillsdale, New Jersey, 1983.

[Chavez and Maes 96]
Chavez, A. and Maes, P.: Kasbah: An Agent Marketplace for Buying and Selling Goods. In: *Proceedings of the First International Conference on the Practical Application of Intelligent Agents and Multi-agent Technology* (PAAM'96). London, April 1996.

[Cook and Kay 94]
Cook, R. and Kay, J.: The Justified User Model: A Viewable, Explained User Model. In: *Proceedings of the Fourth International Conference on User Modeling*. The Mitre Corp., Hyannis, Massachusetts, 1994.

[Cypher 93]
Cypher, A.: Eager: Programming Repetitive Tasks by Demonstration. In: Cypher, A. (ed.): *Watch What I Do: Programming by Demonstration*, pp. 205–219. The MIT Press, Cambridge, Massachusetts, 1993.

[Dieterich *et al.* 93]
Dieterich H., Malinowski, U., Kühme, T. and Schneider-Hufschmidt, M.: State of Art in Adaptive User Interfaces. In: Schneider-Hufschmidt, M., Kühme, T. and Malinowski, U. (eds.): *Adaptive User Interfaces: Principles and Practice*, pp. 13–48. North-Holland Elsevier Science Publishers, Amsterdam, 1993.

[Firefly 97]
Firefly Network, Inc.: Collaborative Filtering Technology: An Overview, 1997. Published in WWW at: http://www.firefly.net/CollaborativeFiltering.fly (checked 19.4.-99).

[Franklin and Graesser 96]
Franklin, S. and Graesser, A.: Is in an Agent, or just a Program?: A Taxonomy for Autonomous Agents. *Proceedings of the Third International Workshop on Agent Theories, Architectures, and Languages*. Springer-Verlag, 1996.

[Gillan 98]
Gillan, D.: *The Psychology of Multimedia: Principles of Perception and Cognition*. Tutorial notes for ACHCI 98, Tampere, May 1998.

[Hix and Hartson 93]
Hix, D. and Hartson, R.: *Developing User Interfaces: Ensuring Usability Through Product & Process*. John Wiley & Sons Inc., New York, 1993.

[Hoppe 93]
Hoppe, U.: Intelligent User Support Based on Task Models. In: Schneider-Hufschmidt, M., Kühme, T. and Malinowski, U. (eds.): *Adaptive User Interfaces: Principles and Practice*, pp. 167–183. North-Holland Elsevier Science Publishers, Amsterdam, 1993.

[Höök *et al.* 96]
Höök, K., Karlgren, J., Wærn, A., Dahlbäck, N., Jansson, C., Karlgren, K. and Lemaire, B.: A Glass Box Approach to Adaptive Hypermedia. In: *International Journal on User Modeling and User-Adaptive Interaction*, special issue on intelligent hypermedia. 1996.

[Höök 97]
Höök, K.: Steps to Take Before IUI Becomes Real. Presented in: The Reality of Intelligent Interface Technology, Edinburgh, March 25th, 1997. Available at: http://www.sics.se/~kia/papers/reality_of_II.html (checked 22.4.-99).

[Höök 99]
Höök, K.: *Designing and Evaluating Intelligent User Interfaces*. Tutorial in International Conference on Intelligent User Interfaces. Redondo Beach, Los Angeles, California, January 5th, 1999.

[Johnson 98]
Johnson, L.: Learning from Agents. In: *Proceedings of the First International Workshop on Interaction Agents* (IA'98), pp. 1–7. L'Aquila, Italy, May 1998.

[Johnson *et al.* 99]
Johnson, C., Brinbaum, L., Bareiss, R. and Hinrichs, T.: Integrating Organizational Memory and Performance Support. In: *Proceedings of the International Conference on Intelligent User Interfaces* (IUI'99), pp. 127–134. Redondo Beach, Los Angeles, California, January, 1999.

[Kay 90]
Kay, A.: User Interface: A Personal View. In: Laurel B. (ed.): *The Art of Human-Computer Interface Design*, pp. 191–208. Addison Wessley, Reading, Massachusetts, 1990.

[Koda 96]
Koda, T.: *Agents with Faces: A Study on the Effects of Personification of Software Agents*. Master's Thesis at the Massachusetts Institute of Technology, 1996. Available at:
http://tomoko.www.media.mit.edu/people/tomoko/
(checked 22.7.1999).

[Kühme and Schneider-Husfschmidt 93]
Kühme, T. and Schneider-Hufschmidt, M.: Introduction. In: Schneider-Hufschmidt, M., Kühme, T. and Malinowski, U. (eds.): *Adaptive User Interfaces: Principles and Practice*, pp. 1–10. North-Holland Elsevier Science Publishers, Amsterdam, 1993.

[Lashkari *et al.* 94]
Lashkari, Y., Metral, M. and Maes, P.: Collaborative Interface Agents. In: *Proceedings of AAAI'94 Conference*, Seattle, Washington, August 1994.

[Laurel 90]
Laurel, B.: Interface Agents: Metaphors with Character. In: Laurel B. (ed.): *The Art of Human-Computer Interface Design*, pp. 355–366. Addison Wesley, Reading, Massachusetts, 1990.

[Lieberman 93]
Lieberman, H.: Mondrian: A Teachable Graphical Editor. In: Cypher, A. (ed.): *Watch What I Do: Programming by Demonstration*, pp. 341–360. The MIT Press, Cambridge, Massachusetts, 1993.

[Lieberman 97]
Lieberman, H.: Autonomous Interface Agents. In: *Proceedings of the Conference on Human Factors in Computing Systems* (CHI'97), pp. 67–74. Atlanta, May 1997.

[Lieberman 99]
Lieberman, H.: Let's Browse: A Collaborative Web Browsing Agent. In: *Proceedings of the International Conference on Intelligent User Interfaces* (IUI'99), pp. 65–68. Redondo Beach, Los Angeles, California, January , 1999.

[Maes 94]
Maes, P.: Agents that Reduce Work and Information Overload. In: *Communications of the ACM*, pp. 31–40, Vol. 37, No. 7, July 1994.

[Maes and Shneiderman 97]
Maes, P. and Shneiderman, B.: Direct Manipulation vs Interface Agents: Excerpts from debates at IUI 97 and CHI 97. In: *Interactions of the ACM*, pp. 42–61, Vol. 4, No. 6, November / December 1997.

[Maulsby and Witten 93]
Maulsby, D. and Witten, I.: Metamouse: An Instructible Agent for Programming by Demonstration. In: Cypher, A.: *Watch What I Do: Programming by Demonstration*, pp. 155–182. The MIT Press, Cambridge, Massachusetts, 1993.

[Maybury 99]
Maybury, M.: *Intelligent User Interfaces: An Introduction*. Tutorial in International Conference on Intelligent User Interfaces. Redondo Beach, Los Angeles, California, January 5th, 1999.

[Maybury and Wahlster 98]
Maybury, M. and Wahlster, W.: Intelligent User Interfaces: An Introduction. In: Maybury, M. and Whalster, W. (eds.): *Readings in Intelligent User Interfaces*, pp. 1–13. Morgan Kaufmann Publishers, Inc., San Francisco, California, 1998.

[Mayhew 92]
Mayhew, D.: *Principles and Guidelines in Software User Interface Design*. Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1992.

[Morita and Shinoda 94]
Morita, M. and Shinoda, Y.: Information Filtering Based on User Behavior Analysis and Best Match Text Retrieval. In: *Proceedings of the 17th Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval*, pp. 272–281. 1994.

[Moukas 96]
Moukas, A.: Amalthaea: Information Discovery and Filtering using a Multiagent Evolving Ecosystem. In: *The Proceedings of the First International Conference on the Practical Application of Intelligent Agents and Multi Agent Technology* (PAAM'96), London, 1996.

[Myers 93]
Myers, B.: Peridot: Creating user Interfaces by Demonstration. In: Cypher, A. (ed.): *Watch What I Do: Programming by Demonstration*, pp. 125–154. The MIT Press, Cambridge, Massachusetts, 1993.

[Neal and Shapiro 91]
Neal, J. and Shapiro, S.: Intelligent Multi-media Interface Technology. In: Sullivan, J. and Tyler, S. (eds.): *Intelligent User Interfaces*, pp. 11–44. ACM Press, New York, 1991.

[Nielsen 93]
Nielsen, J.: *Usability Engineering*. Academic Press, Inc., San Diego, California, 1993.

[Norman 94]
Norman, D.: How Might People Interact with Agents. In: *Communications of the ACM*, pp. 68–71, Vol. 37, No. 7, July 1994.

[Pandit and Kalbag 97]
Pandit, M. and Kalbag, S.: The Selection Recognition Agent: Instant Access to Relevant Information and Operations. In: *Proceedings of the International Conference on Intelligent User Interfaces* (IUI'97), pp. 47–52. Orlando, Florida January 6–9, 1997.

[Pohl *et al.* 95]
Pohl, W., Kobsa, A. and Kutter, O.: User Model Acquisition Heuristics Based on Dialogue Acts. In: *International Workshop on the Design of Cooperative Systems*, pp. 471–486. Antibes-Juan-les-Pins, France, 1995.

[Raisamo 99]
Raisamo, R.: Agentit käyttöliittymässä. Esitelmä seminaarissa: Agenttiohjelmointi ja sen uuden sovellutukset. Hanasaari, Espoo, tammikuu 1999.

[Rhodes and Starner 96]
Rhodes, B. and Starner, T.: Remebrance Agent: A Continuously Running Automated Information Retrieval System. In: *Proceedings of the First International Conference on the Practical Application of Intelligent Agents and Multi Agent Technology* (PAAM'96), pp. 487–495. 1996.

[Roth *et al.* 91]
Roth, S., Mattis, J. and Mesnard, X.: Graphics and Natural Language as Components of Automatic Explanation. In: Sullivan, J. and Tyler, S. (eds.): *Intelligent User Interfaces*, pp. 207–240. ACM Press, New York, 1991.

[Sakagami and Kamba 97]
Sakagami, H. and Kamba, T.: Learning Personal Preferences on Online Newspaper Articles From User Behaviors. In: *Hyper Proceedings of the Sixth International World Wide Web Conference*. 1997. Available at: http://www6.nttlabs.com/HyperNews/get/PAPER142.html (checked 21.4.-99).

[Salton 89]
Salton, G.: *Automatic text processing: the transformation, analysis, and retrieval of information by computer*. Addison-Wesley, Reading, Massachusetts, 1989.

[Schubert *et al.* 98]
Schubert, C., Zarnekow, R. and Brenner, W.: A Methodology for Classifying Intelligent Software Agents. In: Baets, W. (ed.): *Proceedings of the Sixth European Conference on Information Systems*, pp. 304–316. Aix-en-Provence France, June 4–6, 1998.

[Sears and Shneiderman 94]
Sears, A. and Shneiderman, B.: Split Menus: Effectively Using Selection Frequency to Organize Menus. In: *ACM Transactions on Computer-Human Interaction*, pp. 27–51, Vol. 1, No. 1, March 1994.

[Sheth 94]
Sheth, B.: *A Learing Approach to Personalized Information Filtering*. Master's Thesis at the Massachusetts Institute of Technology, February 1994.

[Shneiderman 93]
Shneiderman, B.: Beyond Intelligent Machines: Just Do It! In: *IEEE Software*, pp. 100–103. Vol. 10, No. 1, January 1993.

[Shneiderman 94]
Shneiderman, B.: Dynamic Queries for Visual Information Seeking. In: *IEEE Software*, pp. 70–77. Vol. 11, No. 6, November 1994.

[Shneiderman 95]
Shneiderman, B.: Looking for the Bright Side of User Interface Agents. In: *Interactions of the ACM*, pp. 13–15. Vol. 2, No. 1, January 1995.

[Shneiderman 98]
Shneiderman, B.: *Designing the User Interface: Strategies for Effective Human-Computer Interaction*, 3rd ed. Addison Wesley, Reading, Massachusetts, 1998.

[Strachan *et al.* 97]
Strachan, L., Anderson, J., Sneesby, M. and Evans, Mark.: Pragmatic User Modeling in a Commercial Software System. In: Jameson, A., Paris, C. and Tasso, C. (eds.): *User Modeling: Proceedings of the Sixth International Conference* (UM97), pp. 189–200. Vienna, Austria. Springer, New York, 1997.

[Sukaviriya and Foley 93]
Sukaviriya, P. and Foley, J.: A Build-in Provision for Collecting Individual Task Usage Information in UIDE: the User Interface Design Environment. In: Schneider-Hufschmidt, M., Kühme, T. and Malinowski, U. (eds.): *Adaptive User Interfaces: Principles and Practice.* North-Holland Elsevier Science Publishers, Amsterdam, 1993.

[Towns *et al.* 98]
Towns, S., Callaway, C., Voerman, J. and Lester, J.: Coherent Gestures, Locomotion, and Speech in Life-Like Pedagogical Agents. In: *Proceedings of the International Conference on Intelligent User Interfaces* (IUI'98), pp. 13–20. San Francisco, California, January 1998.

[Van Dyke *et al.* 99]
Van Dyke, N., Lieberman, H. and Maes, P.: Butterfly: A Conversation-Finding Agent for Internet Relay Chat. In: *Proceedings of the International Conference on Intelligent User Interfaces* (IUI'99), pp. 39–46. Redondo Beach, Los Angeles, California, January 1999.

[Webster 81]
Webster's Third New International Dictionary, Encyclopædia Britannica, Inc., 1981.

[Wiio 94]
Wiio, O.: *Johdatus viestintään*, kuudes laitos. WSOY, Porvoo, 1994.

[Windows 95]
The Windows® Interface Guidelines for Software Desing. Microsoft Press, Redmond, Washington, 1995.