Opastusjärjestelmän rakentaminen

Eero Kettunen

13.5.2004

Joensuun yliopisto
Tietojenkäsittelytiede
Pro gradu -tutkielma

Tiivistelmä

Opastusjärjestelmä on tiettyä sovellusta varten tehty ja sen käyttöä helpottamaan tarkoitettu ohjelmisto, joka tarjoaa sovelluksen käyttäjälle käytönaikaista opastusta kyseisen sovelluksen käytössä. Tällaiset suoraohjeet ovat valtaosin korvanneet perinteiset painetut käyttöohjeet nykyaikaisissa sovelluksissa. Kaikkia tämä siirtymä ei kuitenkaan miellytä - tutkimusten mukaan valtaosa käyttäjistä suosii painettua käyttöohjetta opastusjärjestelmään nähden ja monilla on vaikeuksia opastusjärjestelmien käytössä. Tässä pro gradu -tutkielmassa perehdytään olemassa oleviin opastusjärjestelmiin, niissä havaittuihin ongelmiin sekä tutustutaan opastusjärjestelmän suunnitteluun ja rakentamiseen liittyviin seikkoihin niin kirjallisuuden pohjalta kuin myös käytännön tasolla.

ACM-luokat (ACM Computing Classification System, 1998 version): H.5.2

Avainsanat: opastusjärjestelmä, suoraohje

Sisältö

1 Johdanto

90-luvun alkupuolella julkistetun Word for Windows -tekstinkäsittelyohjelman ensimmäisen version mukana toimitettiin muutama disketti ja melkein tuhatsivuinen käyttöopas. Tänä päivänä samaisen ohjelman uusin versio toimitetaan CD-ROM:lla ohuen vihkosen kera. Vaikka kyseisen ohjelman koko sekä sen sisältämien toimintojen määrä ovat kasvaneet kymmenessä vuodessa räjähdysmäisesti, on sen paperidokumentaatio kutistettu kustannussyistä olemattomiin. Kyseinen trendi on ollut nähtävissä yleisesti jo pitempään ohjelmistoissa: ohjelmistojen koot kasvavat huimasti ja uusia versioita ilmestyy markkinoille tiheään. Ohjelmistosta puuttuvaa paperimanuaalia korvaa nykyisin lähes poikkeuksetta ohjelmistoon itseensä integroitu opastusjärjestelmä (on-line help). Paperidokumentaation puuttuessa korostuu opastusjärjestelmien rooli käyttäjien apuna alati kasvavien ja monimutkaistuvien ohjelmistojen viidakossa.

Opastusjärjestelmällä tarkoitetaan yhdestä tai useammasta ohjelmasta koostuvaa järjestelmää, jonka tarkoituksena on avustaa käyttäjää jonkin suuremman ohjelman tai tietokonejärjestelmän käytössä (Kearsley, 1988). Opastusjärjestelmästä käytetään myös nimitystä *käytönaikainen ohje* tai *suoraohje*. Molemmat termit kuvaavat hyvin opastusjärjestelmien luonnetta nopeasti saatavilla olevana, ohjelman käytönaikaisena opastuskeinona.

Opastusjärjestelmiä pidetään yleensä ylivoimaisina paperidokumentaatioon verrattuna, koska ne voivat hyödyntää tietokoneen laskentakapasiteettia ja tarjota mm. säilytys-, etsintä- ja vuorovaikutusominaisuuksia (Duffy & al., 1992). Opastusjärjestelmän etu on mm. se, että se on yleensä integroitu itse sovellukseen, jolloin se on aina käytettävissä siellä missä itse ohjelmakin. Sitä ei tarvitse lähteä hakemaan kirjahyllystä, kuten tiiliskiven kokoista painettua käyttöopasta. Opastusjärjestelmä ei myöskään katoa muutossa, eikä kukaan lainaa sitä, kuten opaskirjalle saattaa käydä. Silti useat tutkimukset osoittavat, että käyttäjät, etenkin vasta-alkajat, löytävät tietoa käyttämästään ohjelmasta helpoiten paperidokumentaation avulla (Kearsley, 1988; Schneiderman, 1998).

Vaikka opastusjärjestelmät ovatkin kehittyneet huimasti viimeisen kahdenkymmenen vuoden kuluessa, on niiden käytettävyydessä vielä runsaasti parantamisen varaa. Tämän tutkielman tarkoituksena on valottaa opastusjärjestelmän suunnitteluun liittyviä seikkoja ja kompastuskiviä alaan liittyvän tutkimustiedon pohjalta ja toisaalta arvioida kahta erilaista opastusjärjestelmän toteutusympäristöä sekä kuvata järjestelmän ra-

kentamisprosessi käytännössä. Tutkielma keskittyy nimenomaan työasematietokoneiden opastusjärjestelmiin, erilaisten pienten laitteiden, kuten matkapuhelinten ja pda-laitteiden opastusjärjestelmät eivät kuulu tutkielman piiriin.

Tutkielman rakenne on seuraavanlainen. Luvussa 2 perehdytään opastusjärjestelmien historiaan ja kehitykseen sekä kuvataan nykyisiin opastusjärjestelmiin osittain jo vakiintuneet osat ja toiminnot. Luvun lopuksi tutustutaan opastusjärjestelmistä tehtyihin tutkimuksiin. Luvussa 3 tarkastellaan opastusjärjestelmän käyttöliittymän sekä opastustekstien sisällön suunnittelemiseen liittyviä seikkoja. Tässä luvussa kuvataan myös opastusjärjestelmän rakentamisprosessi alusta loppuun saakka. Luvussa 4 tutustutaan opastusjärjestelmän rakentamiseen käytännössä kahden eri toteutusympäristön kautta ja pyritään käsittelemään käytännön toteutustyössä esiintulevia seikkoja lukujen kaksi ja kolme teoriapohjaa vasten. Lopuksi luvussa 5 tehdään yhteenveto havaituista asioista sekä arvioidaan opastusjärjestelmien nykytilaa.

2 Opastusjärjestelmät ja niiden kehitys

Monet nykyisten opastusjärjestelmien ominaisuuksista ovat peräisin perinteisestä kirjamuotoisesta dokumentaatiosta. Tässä luvussa kuvataan opastusjärjestelmien yleiset ominaisuudet ja opastusjärjestelmien kehittyminen nykyiseen muotoonsa. Luvun lopuksi käydään läpi opastusjärjestelmistä tehtyjä tutkimuksia, joille tutkielman teoriaosuus perustuu. Ensin luodaan katsaus opastusjärjestelmien kehitykseen.

2.1 Opastusjärjestelmien kehitys

Johnson-Eilola (2001) kuvaa dokumentaation historiallista evoluutiota sosiaalisen, suullisen ja fyysisen dokumentaation muuttumisena yksityiseksi, kirjalliseksi ja näkymättömäksi. Aiemmin tietokone-ekspertit työskentelivät suurissa tietokonesaleissa keskenään, ja dokumentaatio oli pitkälti suullista ja sosiaalista. Myöhemmin dokumentaatio siirtyi suullisesta muodosta paksuihin painettuihin dokumentteihin ja nykyisin dokumentaatio on yksilöllistä (jopa personoitua) ja kirjallista, mutta painetun sijasta jopa näkymätöntä, käytettävään sovellukseen liitettyä tai sen käyttöliittymään upotettua.

Kun paperidokumenttien painaminen, jakelu sekä päivittäminen huomattiin hankalaksi ja kalliiksi, alettiin etsimään keinoja jakaa dokumentit varsinaisten tuotteiden kanssa sähköisinä. Aluksi painetut dokumentit (hardcopy) siirrettiin sellaisenaan sähköiseen muotoon (softcopy) ja jaettiin yleensä CD-ROM:lla. Vähitellen kirjamuotoisen tekstin rinnalle lisättiin hypertekstilinkkejä sekä hakutoiminto. Näin syntyivät sähköiset dokumentit (online documents), joissa itse opastusteksti oli identtinen painetun dokumentin sisällön kanssa. Myöhemmin dokumentit liitettiin osaksi sovelluksen käyttöliittymää: näin saivat alkunsa opastusjärjestelmät, joissa myös dokumenttien sisältöä oli muokattu tiivimmäksi ja tarkemmaksi (Turk & Nichols, 1996).

Sähköiset dokumentit (mm. pdf-tiedostot) sekä suuri osa vielä nykyisistäkin opastusjärjestelmistä ovat luonteeltaan kirjamaisia: ne sisältävät sisällysluettelon, hakemiston sekä selaustoiminnon. Nämä kirjaparadigmaan pohjautuvat opasteet nojautuvat muotonsa puolesta käyttäjän aikaisempiin kokemuksiin, jolloin käyttäjän on helpompi käyttää niitä tehokkaasti. Kirjamainen muoto mahdollistaa sisällön selaamisen usealla eri tavalla, kuten (Kantner & al., 2002):

- kannesta kanteen (alusta loppuun)
- sisällysluettelosta tiettyyn aiheeseen
- ristiviittauksesta viitattavaan tietoon
- hakemiston asiasanasta tiettyyn aiheeseen
- käyttäjän asettamaan kirjanmerkkiin mistä tahansa paikasta.

Kun painettuja dokumentteja alettiin tuottamaan ja jakamaan sähköisessä muodossa, pääpaino oli aluksi sillä, kuinka dokumentit saatiin varastoitua mahdollisimman pieneen tilaan. Myöhemmin käytettävyysvaatimusten korostuminen aiheutti mielenkiinnon siirtymisen dokumenttien luettavuuteen ja käyttökelpoisuuteen. Aluksi painotettiin sellaisia suunnittelun elementtejä kuten tekstin muotoilu (listat ja taulut) sekä tekstin esitystapa (fontit ja korostus) (Turk & Nichols, 1996).

Tietokoneiden kehittyneitä ominaisuuksia alettiin käyttää hyväksi myös opastusjärjestelmien tekemisessä: erilaisia haku- ja selaustoimintoja liitettiin opastusjärjestelmien yhteyteen. Hypertekstin yleistyminen 1980-luvun loppupuolella merkitsi vallankumousta myös opastusjärjestelmille. Hyperteksti (hypertext) mahdollistaa toisiinsa liittyvän informaation linkittämisen toisiinsa asiayhteyksien perusteella. Hyperteksti koostuu lyhyistä tekstin osista, tekstikatkelmista, joista on viittauksia toisiin tekstikatkelmiin. Tekstikatkelmia voidaan kutsua tietosoluiksi (nodes) ja tietosolujen välisiä kytkentöjä linkeiksi (links) (Nielsen, 1990). Perinteisessä, kirjamaisessa tiedon esitystavassa dokumentin fyysinen ja looginen rakenne ovat liittyneet läheisesti toisiinsa: teksti koostuu peräkkäisistä osioista, kuten sanoista, lauseista, kappaleista ja luvuista. Peräkkäiset tekstin osat muodostavat tekstin sisällön, jota voi selata eteen- tai taaksepäin. Hypertekstissä tekstin asiasisältö on irrotettu tekstin rakenteesta, ts. sen lineaarinen rakenne on rikottu (Heimbürger & al., 1990). Näin hypertekstimuotoista tietoa voidaan selata toisiinsa liittyvien asioiden välillä, asiayhteyksiin nojaten. Hypertekstin lisäksi joissakin opastusjärjestelmissä käytetään myös hypermediaa, joka on hypertekstin laajennos sisältäen hypertekstin lisäksi ääntä, videokuvaa ja animaatiota (Heimbürger & al., 1990).

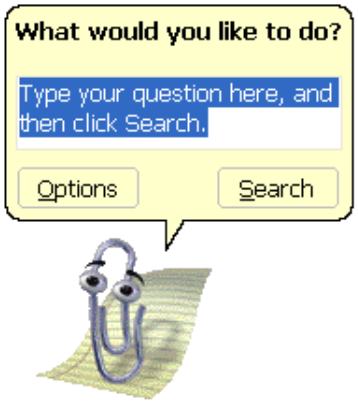
Varhaisimmat opastusjärjestelmät kehitettiin komentokieliä varten ja niiden tehtävänä oli helpottaa komentojen nimien ja niiden syntaksin muistamista. Esimerkkeinä tällaisista opastusjärjestelmistä ovat mm. komentorivipohjaisten käyttöjärjestelmien, kuten

UNIX-käyttöjärjestelmän sisältämät komento-opasteet sekä mm. varhainen tietoliikenneohjelma Crosstalk (Kearsley, 1988). Komentorivipohjaiset opastusjärjestelmät olivat yleensä staattisia, mikä tarkoittaa sitä, että järjestelmän tarjoama opaste ei ole riippuvainen käyttäjän sijainnista ohjelmassa tai tämän aikaisemmin suorittamista toiminnoista.

MicroPro-yhtiön Wordstar tekstinkäsittelyohjelma oli yksi varhaisimmista ohjelmista, joihin rakennettiin monitasoinen opastusjärjestelmä, joka on käyttäjän muokattavissa. Kyseisen ohjelman käyttäjä voi valita opasteen laajuuden oman tietämystasonsa mukaiseksi ja säätää sitä oppimisen kuluessa (Kearsley, 1988). Taulukkolaskentaohjelma Lotus 1-2-3 sisälsi jo ensimmäisessä versiossaan kontekstisidonnaisen ohjeen, mikä mahdollistaa kuhunkin käyttötilanteeseen sopivan avun saamisen yksinkertaisesti F1-painiketta painamalla (Kearsley, 1988). Kyseinen ominaisuus on olemassa vielä nykyaikaisissakin opastusjärjestelmissä ja on käyttäjien suosiossa edelleenkin (Purchase & Worrill, 2002). Apple Guide, Macintosh käyttöjärjestelmään integroitu opastusjärjestelmä, on järjestelmä joka antaa interaktiivista käyttäjäopastusta ja vastaa käyttäjän kysymyksiin, kuinka tietty tehtävä voidaan suorittaa. Se tarjoaa apua ottaen huomioon käyttäjän kontekstin järjestelmässä ja pystyy synkronoimaan opasteen niiden tehtävien mukaan, joita käyttäjä on parhaillaan suorittamassa (Hefley, 1995).

Microsoftin vuonna 1993 lanseeraama Bob ja siitä myöhemmin kehitetty toimistoapulainen (Office Assistant) poikkeavat perinteisistä kirjamaisista opastusjärjestelmistä (Randall & Pedersen, 1998). Toimistoapulainen on esitetty kuvassa 1. Kyseessä on sarjakuvamainen, animoitu ”olio”, jonka ulkoasun käyttäjä voi valita ja joka tarjoaa käyttäjälle kontekstisidonnaisia ohjeita tämän pyytämättäkin. Toimistoapulainen tuli markkinoille Microsoft Office 97:n sekä Office 98 for Macintosh -toimistopakettien mukana ja sai aikaan melkoisen vastustuksen käyttäjien keskuudessa (Shroyer, 2000). Ihmiset eivät pitäneet toimistoapulaisen tavasta keskeyttää heidän työntekonsa tarjoamalla jotakin vinkkiä. Myös toimistoapulaisen levoton animointi aiheutti monissä käyttäjissä ärtymystä.

Varhaisemmat opastusjärjestelmät olivat yleensä dedikoituja, yhtä sovellusta varten kehitettyjä ohjelmia. Nykyiset opastusjärjestelmät rakennetaan usein käyttäen apuna erilaisia opastusjärjestelmän rakentamistyökaluja (authoring tools) tai ainakin hyödyntämällä standarditekniikoita, kuten HTML Help tai JavaHelp. Opastusjärjestelmän rakentaminen on kuvattu käytännön tasolla luvussa 4. Seuraavaksi käydään läpi opastusjärjestelmien eri lajit sekä niihin tyypillisesti kuuluvat osat ja toiminnot.



Kuva 1: Microsoftin Office Assistant.

2.2 Opastusjärjestelmien eri lajit sekä niiden ominaisuudet

Opastusjärjestelmät voidaan jakaa *aktiivisiin* ja *passiivisiin* (Fischer & al., 1985). Passiivisia opastusjärjestelmiä ovat suurin osa sovelluksissa käytössä olevista hakemiston ja sisällysluettelon sisältävistä, kirjamaisista ohjeista. Yhteistä passiivisille opastusjärjestelmille on mm. se, että niiden käynnistäminen vaatii käyttäjältä toimenpiteen. Passiivisesta opastusjärjestelmästä voidaan näin ollen käyttää myös nimeä *käyttäjäaloitteinen opastusjärjestelmä* (user-initiated help). Aktiivisia opastusjärjestelmiä ovat mm. edellä kuvattu toimistoapulainen sekä yleisemmin kaikki järjestelmän toimesta käynnistyvät opastusjärjestelmät (system-initiated help). Aktiviisille opastusjärjestelmille ominaista on niihin liittyvä tekoäly, ne pitävät yleensä kirjaa käyttäjän kontekstista varsinaisessa sovelluksessa ja samalla valvovat käyttäjän toimia tai jopa pyrkivät ennustamaan ja mallintamaan mitä tehtävää käyttäjä kulloinkin yrittää suorittaa.

Eräs aktiivisten opastusjärjestelmien alalaji on *sulautettu opastusjärjestelmä*. Sulautettu opastusjärjestelmä toimii suoraan käytettävän ohjelmiston käyttöliittymän integroituna ja tarjoaa käyttäjälle opastusta ilman tämän aloitetta ja joskus jopa käyttäjän huomaamatta. Sulautetuille opastusjärjstelmille yhteistä on se, että ne pyrkivät tarjoamaan käyttäjälle proaktiivista apua, ts. opastusta ennen kuin käyttäjä edes huomaa tarvitsevansa apua tai joutuu virhetilanteeseen. Esimerkki sulautetusta opastusjärjestelmästä on *velho* (wizard), jota käytetään jonkin tietyn toiminnallisuuden (esim. graafin piirtäminen datasta) suorittamiseksi. Velho esittää käyttäjälle sarjan dialogeja, joissa kussakin on yksinkertaisia kysymyksiä, ja johdattaa käyttäjän näin askelittain kohti koko

tehtävän suoritusta.

Opastusjärjestelmät voidaan luokitella toisaalta myös *staattisiin* ja *dynaamisiin*. Staattiset opastusjärjestelmät eivät ole riippuvaisia käyttäjän sijainnista ohjelmassa tai tämän suorittamista toiminnoista. Dynaamisten opastusjärjestelmien antama tuloste taas riippuu käyttäjän sijainnista ohjelmassa tai tämän tekemistä toiminnoista (Kearsley, 1988).

Nykyisistä opastusjärjestelmiin vakiutuneista toiminnoista yleisin lienee *hakemisto*, joka mahdollistaa tietyn hakusanan etsimisen aakkosjärjestykseen järjestetystä listasta aivan kuten perinteisen kirjan lopusta löytyvä hakemisto. *Sisällysluettelo* (contents) on sekin kirjoista tuttu asia, jonka avulla käyttäjä voi selailla hierarkkisesti järjestettyjä aihealuita ja saada siten yleiskuvan ohjeen sisällöstä.

Hakutoiminto (find) on opastusjärjestelmien kirjaparadigman laajennos, joka antaa mahdollisuuden etsiä tietyn sanan esiintymiä opastustekstistä.

Hyperlinkit (hyperlinks) mahdollistavat toisiinsa liittyvän informaation linkittämisen toisiinsa samoin kuin Internetin WWW-sivut. Niiden avulla opastusjärjestelmän lukija voi selata ohjetta toisiinsa liittyviä aihealueiden kesken. Normaalissa paperidokumentissa käyttäjä on sidottu selaamaan ohjetta sivu sivulta siinä järjestyksessä, kuin se on kirjoitettu.

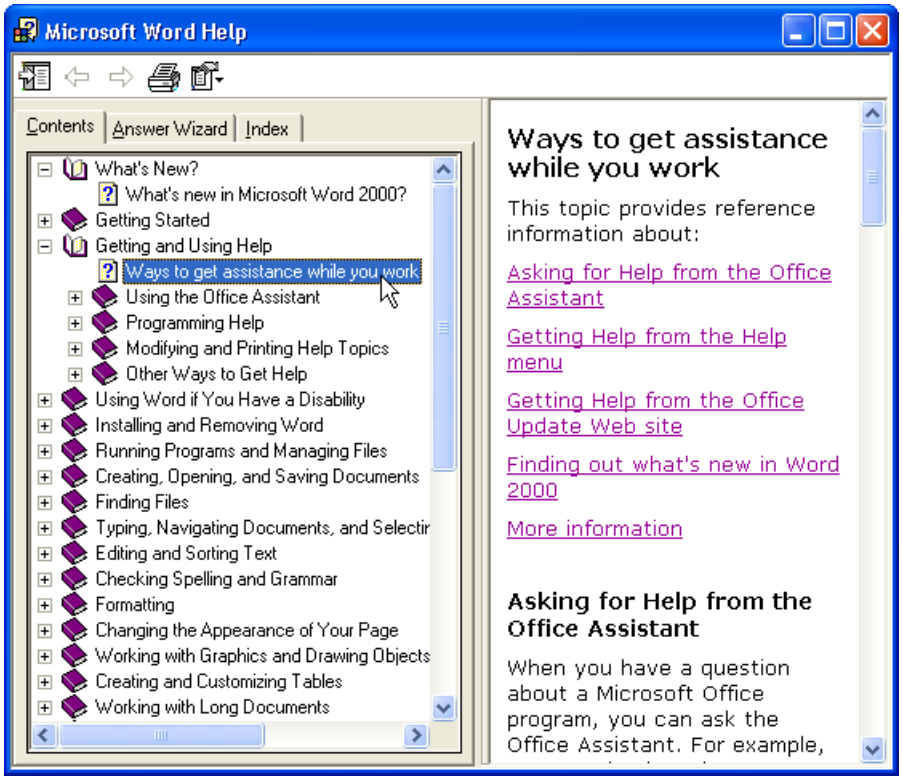
Kontekstisidonnaisuus (context-sensitivity) on joistakin opastusjärjestelmistä löytyvä ominaisuus, jonka avulla käyttäjä saa apua parhaillaan suorittamansa tehtävän tekemiseen. Jos käyttäjä on esim. tallentamassa dokumenttia, saa hän apua pyytäessään nimenomaan tallentamiseen liittyvää opastusta, eikä hänen tarvitse etsiä haluamaansa opastetta alusta alkaen.

Kuplaohje eli *vihjeteksti* (balloon help, tooltip) tarkoittaa lyhyttä selitystekstiä sisältävää puhekuplaa, joka ilmestyy näytölle kun käyttäjä siirtää hiiren kohdistimen jonkin käyttöliittymäelementin päälle. Kuplaohjeen avulla voidaan käyttäjälle antaa tietoa esimerkiksi tietyn käyttöliittymän kentän hyväksymistä arvoista, jolloin opastus toimii ennakoivasti, eikä käyttäjän tarvitse avata mitään erillistä opastetta sovelluksen päälle. Kuplaohje edustaa aktiivista opastusta: käyttäjä saa apua vaikkei pyydä sitä itse. Hyvin toteutettu kuplaohje on kytkettävissä pois helposti, mikäli käyttäjä kokee sen häiritseväksi.

Muita hieman harvinaisempia opastusjärjestelmien ominaisuuksia ovat mm. *selausnuolet* eteen- ja taaksepäin, joista esiintyy opastusjärjestelmissä kahdenlaista versiota: yleensä käyttäjä voi selailla nuolilla aikaisemmin katsomiaan opastussivuja, mutta joistakin järjestelmistä löytyy myös toteutus, jossa nuolilla voi selailla läpi kaikki opastusjärjestelmän sisältämät opastussivut.

Kirjanmerkkien (bookmarks, favorites) tallentamismahdollisuus on selaimista tuttu toiminto, joka mahdollistaa usein katsottujen opastussivujen tallettamisen erityiseen suosikkilistaan, josta ne ovat nopeasti löydettävissä. Myös *haku luonnollista kieltä käyttämällä* on mahdollista joissakin opastusjärjestelmissä. Kyseisen ominaisuuden avulla käyttäjä voi hakea haluamaansa opastetta käyttämällä kokonaisia lauseita hakusanojen sijaan. Näiden lisäksi useissa opastusjärjestelmissä on mahdollisuus tulostaa opastussivuja paperille.

Edellä luetellut toiminnot ovat nykyisistä opastusjärjestelmistä löytyviä yleisimpiä toimintoja. Kuvassa 2 on esitetty tyypillinen, kirjamainen opastusjärjestelmä.



Kuva 2: Microsoft Word 2000 -tekstinkäsittelyohjelman opastusjärjestelmä esimerkkinä tyypillisestä opastusjärjestelmästä.

2.3 Opastusjärjestelmistä tehdyt tutkimukset

Seuraavassa kuvaillaan joitakin opastusjärjestelmien piirissä tehtyjä empiirisiä tutkimuksia ja niistä saatuja tuloksia. Osa viitatuista tutkimuksista mittaa käyttäjien tehokkuutta ja suoritusaikaa, osa taas käyttäjien mieltymyksiä ja asenteita opastusjärjestelmiä kohtaan.

Kearsleyn (1988) luettelema opastusjärjestelmistä empiirisesti mitattujen asioiden lista on edelleenkin validi ja kuuluu seuraavasti:

1. Tarvitaanko vasta-alkajille ja edistyneille erilaisia opastusjärjestelmiä?
2. Kumpi on tehokkaampi opastusjärjestelmän laji: käyttäjä- vai järjestelmäaloitteinen (passiivinen vai aktiivinen)?
3. Ovatko opastusjärjestelmät hyödyllisempiä kuin paperimanuaalit?
4. Mitkä opastusjärjestelmien ominaisuudet ovat kaikkein hyödyllisimpiä?

Kearsley huomauttaa, että empiiristen tutkimusten käytännön arvo on yleensä pieni, koska tutkimukset mittaavat joko liian montaa tai liian harvaa muuttujaa. Saadut tulokset soveltuvat Kearsleyn tulkinnan mukaan suunnittelun ohjenuoriksi sekä tutkimuksen lähtökohdiksi. Kearsleyn (1988) mukaan Relles (1979) suoritti yhden varhaisimmasta formaaleista tutkimuksista opastusjärjestelmien saralla. Relles testasi vasta-alkajien ja edistyneiden käyttäjien suoriutumista tehtävistä yksinkertaistetulla tilienhoitojärjestelmällä. Testiin osallistujat jaettiin neljään ryhmään: a) kontrolliryhmä, jolla oli painettu manuaali mutta ei opastusjärjestelmää; b) toinen ryhmä, jolla oli käytössään painettu manuaali ja yksinkertainen (yksitasoinen) opastusjärjestelmä; c) kolmannella ryhmällä oli käytössään sekä painettu manuaali että kehittynyt (kontekstisidonnainen, monitasoinen) opastusjärjestelmä; ja d) neljännellä ryhmällä oli käytössään sama kehittynyt opastusjärjestelmä, mutta ei painettua manuaalia. Relles suoritti kokeen kahdesti: ensin kuudella vasta-alkajalla ja sitten 30:llä edistyneellä. Kokeen tuloksena huomattiin, että vasta-alkajat suoriutuivat pelkkää paperimanuaalia käyttänyttä kontrolliryhmää huonommin käyttäessään opastusjärjestelmää. Toisaalta edistyneet käyttäjät, joilla oli käytössään kehittynyt opastusjärjestelmä suoriutuivat tehtävistä paremmin kuin kontrolliryhmä.

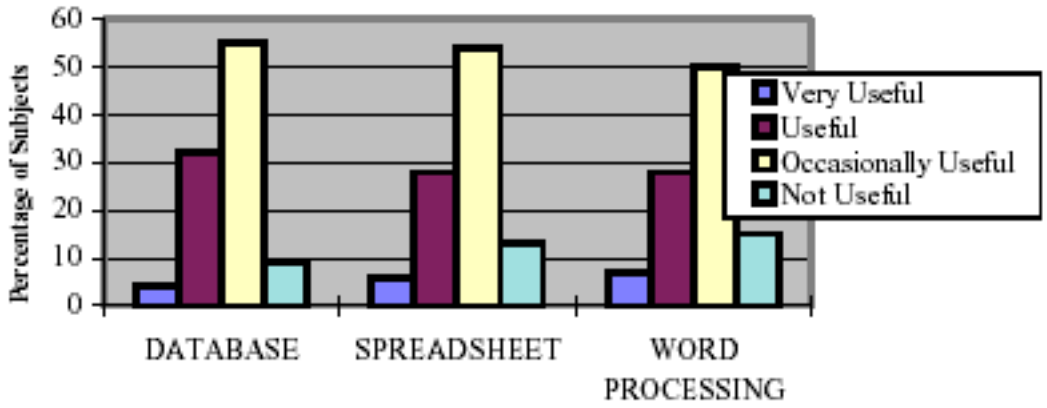
Taulukko 1: Borensteinin testin tulokset (Kearsley, 1988, mukaillen).

Testi	*Edistyneet*	*Vasta-alkajat*
UNIX:in vakio-opastusjärjestelmä	168	167
Muunnellut opastustekstit	116	115
ACRONYM	139	103
Ihmisopettaja	103	60

Magers (1983) testasi VAX/VMS -käyttöjärjestelmän paranneltua versiota 30:llä vasta-alkajatason käyttäjällä. Testiä varten käyttöjärjestelmän opastuskomentoa muutettiin siten, että opastuskomento korvattiin opastuspainikkeella, tarjottiin kontekstisidonnaista apua hakusanapohjaisen opastuksen sijaan, parannettiin apukomentojen syntaksia ja poistettiin tietokoneslangia opastusteksteistä sekä lisättiin esimerkkejä selittämään komentojen syntaksia abstraktin notaation sijasta. Tutkimuksen tuloksista havaittiin, että parannetun järjestelmän käyttäjiltä meni vähemmän aikaa tehtävien tekemiseen ja he myöskin tekivät vähemmän virheitä sekä käyttivät opastusjärjestelmää enemmän ja paperimanuaalia vähemmän kuin alkuperäistä järjestelmää käyttäneet.

Kearsley (1988) raportoi myös Borensteinin (1985) tekemästä edellä kuvatun Magersin tutkimuksen tapaisesta kokeesta, jossa käytettiin UNIX-käyttöjärjestelmän parannettua versiota. Borenstein vertasi UNIX-käyttöjärjestelmän vakio-opastusjärjestelmää samankaltaiseen järjestelmään, johon oli kirjoitettu vaihtoehtoiset opastustekstit. Lisäksi hän kehitti vakio-opastusjärjestelmän rinnalle kontekstisidonnaisen ACRONYM-opastusjärjestelmän. Borensteinin tutkimuksessa kahta käyttäjäryhmää (toisella ryhmällä oli kokemusta UNIX-käyttöjärjestelmästä, toisella puolestaan ei) pyydettiin suorittamaan 24 tehtävää, joista 12 vakio-opastusjärjestelmää käyttäen ja 12 joko parannettuja opastustekstejä, ACRONYM-järjestelmää tai ihmisopettajaa apuna käyttäen. Borensteinin tutkimuksen tulokset keskiarvosuoritusaikoineen (sekunteina) on esitetty taulukossa 1. Tutkimuksen tuloksista nähdään, että pelkästään opastustekstien laatua parantamalla saatiin aikaan huomattava tehokkuuden parantuminen vakio-opastusjärjestelmään verrattuna. ACRONYM-järjestelmä ei ollut staattisia, muunneltuja opastustekstejä merkittävästi parempi (edistyneiden käyttäjien tulokset olivat jopa huonompia ACRONYMiä käytettäessä). Testin tuloksista ilmeni myös se, että ihmisopettajan antama opastus oli ylivertaista muihin tutkittuihin opastuksen muotoihin verrattuna.

Abdullahi ja Alty (1998) tutkivat opastusjärjestelmien käyttötiheyttä verrattuna painetun dokumentaation käyttöön ja käyttäjien tyytyväisyyttä opastusjärjestelmiin. Tutkimustapana käytettiin kyselylomaketta ja kysymykset koskivat Microsoftin toimistosovellusten, tekstinkäsittelyn, taulukkolaskennan ja tietokantaohjelmiston, käyttöä sekä niiden opastusjärjestelmien hyödyllisyyttä. Tutkimukseen osallistuneilta tiedusteltiin heidän kokemuksiaan opastusjärjestelmien käyttökelpoisuudesta. Saadut tulokset on esitetty kuvassa 3. Esimerkiksi tietokantaohjelmiston osalta hyvin hyödylliseksi opastusjärjestelmän koki vajaa 5%, hyödylliseksi reilut 30%, ajoittain hyödylliseksi noin 55% ja hyödyttömäksi noin 10% vastaajista. Muiden ohjelmistojen osalta tulokset ovat saman suuntaiset.



Kuva 3: Käyttäjien mielipiteet toimistosovellusten opastusjärjestelmien hyödyllisyydestä (Abdullahi & Alty, 1998).

Tutkimusta varten vastaajien tuli sijoittaa itsensä vasta-alkajien, keskiverto tai edistyneiden käyttäjien ryhmään. Taulukossa 2 on esitetty vastaajien tyytyväisyys opastusjärjestelmiin heidän osaamistasonsa huomioituna. Abdullahin ja Altyn tulkinnan mukaan edistyneet käyttäjät saavat opastusjärjestelmistä enemmän hyötyä, koska he pystyvät aikaisemman kokemuksensa pohjalta muodostamaan vasta-alkajia parempia hakukyselyjä. Samoin todettiin sellainen seikka, että edistyneet käyttäjät syyttivät opastusjärjestelmää kunnollisen opastuksen puutteesta, kun taas vasta-alkajat ja keskivertokäyttäjät syyttivät itseään opastusjärjestelmän hyödyttömyydestä.

Tutkimuksessa pyrittiin selvittämään myös käyttäjien mielipiteitä opastusjärjestelmien tehokkuudesta suhteessa painettuun dokumentaatioon. Kuten taulukosta 3 on nähtävissä valtaosa (75%) vastaajista suosi painettuja dokumentteja, ja vähemmistö (18%) pi-

Taulukko 2: Käyttäjien arviot opastusjärjestelmien hyödyllisyydestä käyttäjätason suhteen kuvattuna (Abdullahi & Alty, 1998, mukaillen).

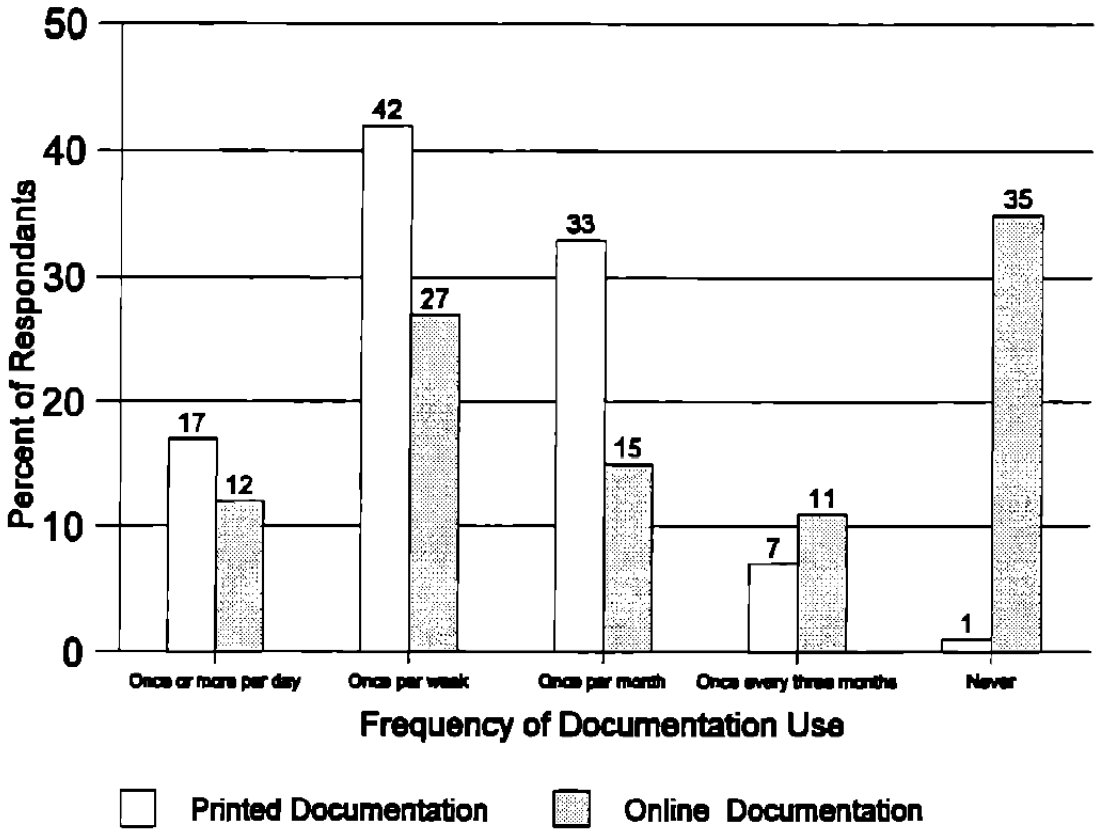
	Hyödyllisyys		*Todellinen käyttö*		
	Hyödyllinen	Hyödytön	Usein	Joskus	Harvoin / Ei koskaan
Edistyneet + Keskitasoiset	36%	64%	7%	45%	48%
Vasta-alkajat	29%	71%	3%	39%	58%

Taulukko 3: Opastusjärjestelmien tehokkuus suhteessa painettuun dokumentaation käyttäjien arvioimana (Abdullahi & Alty, 1998, mukaillen).

Painettu manuaali	*Opastusjärjestelmä*	*Molemmat*
75%	18%	7%

ti opastusjärjestelmiä tehokkaampina. Saman suuntaisia tuloksia ovat esittäneet myös Corrigan ja Kennard (1997) sekä Smart, DeTienne ja Whiting (1998), jotka tekivät puhelinhaastattelun 400:n erään tekstinkäsittelyohjelmiston koti- ja toimistokäyttäjän kesken. Tutkimuksessa verrattiin mm. kyseisen tekstinkäsittelyohjelmiston opastusjärjestelmän ja painetun dokumentin käytön tiheyttä. Tulokset on esitetty kuvassa 4. Kuten tuloksista voidaan havaita, painetun dokumentaation käyttö oli haastateltujen kesken selvästi yleisempää. 35 prosenttia haastatelluista jopa ilmoitti, ettei käytä opastusjärjestelmää koskaan, kun vain 1 prosentti ilmoitti, ettei käytä painettua dokumentaatiota ollenkaan.

Edellä kuvatut empiiriset tutkimukset antavat karun kuvan opastusjärjestelmien käyttökelpoisuudesta. Jos nykyiset opastusjärjestelmät ovat niin huonoja, etteivät käyttäjät halua niitä käyttää, niin minkälainen on käyttäjien mielestä hyvä opastusjärjestelmä? Tähän kysymykseen etsivät vastausta Purchase ja Worrill (2002), joiden tuoreessa empiirisessä tutkimuksessa selvitettiin mitä ominaisuuksia käyttäjät odottivat opastusjärjestelmiltä. Tutkimukseen osallistuneille annettiin lista, joka sisälsi yhteensä 18 opastusjärjestelmiin jo toteutettua tai kuviteltavissa olevaa ominaisuutta, jotka pisteytettiin tutkittavien tekemien valintojen perusteella. Kuusi haastateltujen mielestä käyttökelpoisinta toimintoa olivat



Kuva 4: Painetun dokumentaation ja opastusjärjestelmän käyttötiheys tutkimuksessa haastateltujen kesken (Smart & al., 1998).

- hakemisto
- hakutoiminto
- sisällysluettelo
- kuplaohje (balloon help, tooltip)
- hyperlinkit
- kontekstisidonnaiset ohjeet.

Kaikki listassa olevista toiminnoista löytyvät yleisesti nykyisistä opastusjärjestelmistä. Purchase ja Worrill päättelevät, että kyseiset ominaisuudet valittiin, koska haastatellut ihmiset tunsivat ne. Jatkotutkimuksessa kuitenkin selvisi, että mainitut ominaisuudet

eivät suinkaan ole optimaalisesti toteutettuja nykyisin, vaan niihin olisi käyttäjien mielestä tehtävissä selkeitä parannuksia. Purchasen ja Worrillin huoli onkin, että ilman empiiristä tutkimusta käytettävyydeltään huonoista ominaisuuksista voi tulla yhdenmukaisuuden nimissä ylläpidetty defacto standardi.

Samaisessa tutkimuksessa tutkittiin myös mitä periaatteita käyttäjät arvostivat eniten opastusjärjestelmän suunnittelussa. Periaatteet valittiin kirjallisuuden pohjalta ja käyttäjien mielestä parhaat ohjenuorat opastusjärjestelmien suunnitteluun ovat seuraavat:

1. Ohjeiden tulisi olla helposti ymmärrettäviä.
2. Ohjeiden tulisi olla proseduraalisia: askel askeleelta kuvattuja ohjeita, jotka kuvaavat tarkasti mitä käyttäjän tulee tehdä.
3. Opastuksen tulisi olla ei-tungettelevaa. Ohje tulee voida poistaa tieltä, mikäli se häiritsee.
4. Opastuksen tulisi olla tarkkaa, kattavaa ja johdonmukaista.
5. Opastuksen tulisi olla käyttäjän kielellä kirjoitettua. Käsittämätön opaste on hyödytön.

Listoja vertaamalla nähdään nopeasti, että opastusjärjestelmien ominaisuudet liittyvät opasteiden esittämiseen käytettyyn käyttöliittymätekniikkaan, kun taas suunnittelussa käytettävät periaatteet liittyvät pääosin opasteiden sisältöön, siis itse opastusteksteihin. Tätä kahtiajakoa hyödynnetään seuraavassa luvussa, jossa pohditaan opastusjärjestelmien suunnitteluun liittyviä seikkoja.

3 Opastusjärjestelmän suunnittelu

Opastusjärjestelmien suunnittelun ongelmat voidaan jakaa kahteen eri luokkaan: dokumenttien ja käyttöliittymän suunnitteluun (Duffy & al., 1992). Dokumenttien suunnittelun ongelma on kuinka tuottaa kattavia, käytettäviä ja houkuttelevia dokumentteja – käyttökelpoista sisältöä. Käyttöliittymän suunnittelun perusongelma puolestaan on se, kuinka saada tuotettu teksti helposti käytettävään esitysmuotoon. Kohdassa 3.2 pohditaan mielekkäiden dokumenttien suunnittelun perusasioita ja kohdassa 3.3 käsitellään opastusjärjestelmän käyttöliittymän suunnitteluun liittyviä seikkoja. Kohdassa 3.4 kuvataan opastusjärjestelmän koko suunnitteluprosessi vaihe vaiheelta. Aluksi luodaan katsaus olemassaolevien opastusjärjestelmien ongelmiin.

3.1 Olemassa olevien opastusjärjestelmien ongelmat

Edellisessä luvussa kuvattiin joitakin opastusjärjestelmien tehokkuutta ja niiden käytön mielekkyyttä mittaavia tutkimuksia. Näistä tutkimuksista selvisi mm. se, että etenkin vasta-alkajat löytävät apua huonommin opastusjärjestelmiä hyödyntämällä kuin paperidokumentaatiota käyttäen (Relles, 1979; Abdullahi & Alty, 1998). Käyttäjät myös käyttäisivät mieluummin paperidokumentaatiota kuin opastusjärjestelmiä (Abdullahi & Alty, 1998; Corrigan & Kennard, 1997; Smart & al., 1998). Abdullahi ja Alty (1998) identifioivat kolme ongelma-aluetta nykyisissä opastusjärjestelmissä:

- Ontologinen ongelma, eli tiedon löytämisen ongelma. Käyttäjät (etenkin vasta-alkajat) eivät löydä oikeita termejä, joilla ilmaista avun tarpeen aihe järjestelmälle. Näin heidän kyselynsä johtavat joko epäolennaisen tiedon löytämiseen tai tyhjään vastaukseen.
- Tiedon lokeroituminen (compartmentalisation). Hakupolut johtavat yksittäisiin tiedon lohkoihin syvällä hakupuun hierarkkiassa. Jos käyttäjä ajautuu jossakin harhaan, on hänen vaikeaa palata taaksepäin, ja hän on yleensä pakotettu aloittamaan alusta. Vaikka opastustekstissä olisikin ristiviittauksia jonkin verran, ne usein vain hämmentävät.
- Kokonaisvaltaisen tietopohjan puuttuminen (lack of an integrated base of information). Tätä ongelmaa ei esiinny paperimanuaaleja käytettäessä. Kun käyttäjä

ei tiedä tarkkaan mitä hän haluaa tietää, hän voi selata paperimanuaalia etsien sopivia ongelmaan liittyviä avainsanoja. Opastusjärjestelmässä kokonaisvaltaisen tietopohjan soveltaminen tarkoittaisi sitä, että järjestelmän tulisi näyttää käyttäjälle kulloiseenkin kontekstiin liittyvien termien joukko, joita käyttäjä voisi työstää, lukea uudelleen tai ohittaa kunnes etsittävä tieto löytyy.

Muita opastusjärjestelmien käyttöön liittyviä ongelmia on mm. vaikeus vaihtaa sovelluksen ja ohjeen välillä (Duffy & al., 1992; Ray & Ray, 2001), joka aiheutuu kun käyttäjä pyrkii yhtä aikaa lukemaan opasteita toisesta ikkunasta ja toimimaan ohjeiden mukaan toisessa ikkunassa olevassa sovelluksessa. Ohjetekstien harhaanjohtavuus tai puutteellisuus, navigointivaikeudet sekä esimerkkien puute vaivaavat myös yleisesti käyttäjiä (Purchase & Worrill, 2002). Eräs opastusjärjestelmien käytettävyyden kannalta kriittinen havainto on se, etteivät käyttäjät yleisesti joko halua tai edes osaa käynnistää opastusjärjestelmää sovelluksesta käsin (Grayling, 1998). Myös hyperavaruuteen eksyminen on yleistä hypertekstiä hyödyntävissä opastusjärjestelmissä (Mueller, 2003; Grayling, 1998).

Kirjallisuudessa esiintyy paikoin mielipide (mm. Cooper, 1995), jonka mukaan opastusjärjestelmän liittäminen sovelluksen yhteyteen on merkki huonosta käyttöliittymän suunnittelusta. Tällaisten mielipiteiden taustalla lienee ajatus, että käyttöliittymän pitäisi itsessään olla riittävän intuitiivinen ja helppo käyttää – ilman erillistä opastusjärjestelmää. On myönnettävä, ettei opastusjärjestelmällä koskaan saisi paikata itse sovelluksen käyttöliittymän puutteita. Silti voidaan pitää mahdottomana ajatusta siitä, että nykyaikaiset sovellukset voitaisiin tehdä ymmärrettäväksi pelkästään käyttöliittymän avulla huomioon ottaen nykyisten sovellusten sisältämien toimintojen määrä. Cooperin mielipiteen kaltaiset ajatukset voivat osaltaan olla vaikuttamassa opastusjärjestelmien arvostuksen puutteeseen ohjelmistoja tuottavissa organisaatioissa: kuten Duffy, Palmer ja Mehlenbacher (1992) toteavat, opastusjärjestelmän tekeminen on usein viimeinen osa ohjelmistoprojektia, pakkopulla, joka kaiken lisäksi jätetään usein ohjelmoijien vastuulle. Tästä on pahimmillaan seurauksena se, että opastusjärjestelmän teksti on loppukäyttäjän kannalta käsittämätöntä tietokoneslangia.

3.2 Dokumenttien suunnittelu

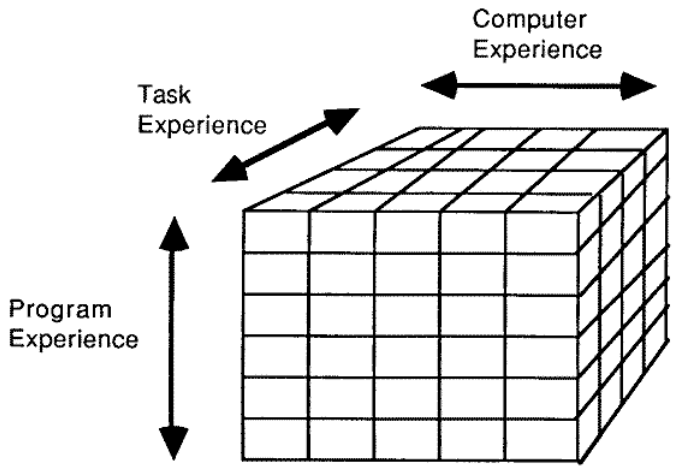
Kuten kohdassa 2.3 kuvatussa Borensteinin (1985) tutkimuksessa todettiin, pelkän opastustekstin muokkaamisella on merkittävä vaikutus opastusjärjestelmän tehokkuuteen. Samoin käyttäjien mieltymyksiä mitanneessa Purchasen ja Worrillin (2002) tutkimuksessa odotetusti todettiin, että opasteen helppo ymmärettävyys oli käyttäjien toivelistalla ylimpänä. Seuraavassa käsitellään dokumenttien luettavuuteen ja ymmärettävyyteen vaikuttavia seikkoja.

Opastusjärjestelmien ohjetekstien kirjoittamiseen liittyy kaksi vaativaa tehtävää. Näistä ensimmäinen on loppukäyttäjän profilointi, ts. sen määrittely, kenelle tuotettavaa tekstiä kirjoitetaan. Toinen ongelma on selvittää tekstin käytön konteksti, eli missä tilanteessa käyttäjä tarvitsee tekstiä.

Loppukäyttäjän tietämystasoa voidaan mallintaa lukuisilla eri tavoilla. Kearsley (1988) esittää erään mallin, jolla voidaan havainnollistaa käyttäjän tietämystasoa kolmella eri ulottuvuudella: *tehtävätietämyksen* (kuinka hyvin käyttäjä tuntee suoritettavan tehtävän), *ohjelmatietämyksen* (kuinka hyvin käyttäjä tuntee ko. ohjelman) sekä *tietokonetietämyksen* (käyttäjän yleinen tietokonetietämys ja taidot) avulla. Esimerkiksi uutta pankkisovellusta kokeileva pankkivirkailija voisi olla tehtävätietämyksen osalta edistynyt, tietokonetietämyksen osalta keskitasoinen ja uuden ohjelman ollessa kyseessä ohjelmatietämyksen osalta vasta-alkaja. Näin laskemalla saadaan lukuisia eri käyttäjätyyppejä, joista kukin tarvitsisi omalle tyypilleen vartavasten suunnitellun ohjeensa. Käytännössä tämä on kuitenkin mahdotonta tai ainakin liian kallista toteuttaa. Sen sijaan voidaan kerätä tietoa, jonka avulla yleisimmät käyttäjätyypit voidaan huomioida opastustekstejä kirjoitettaessa. Kearsleyn loppukäyttäjän tietämystasoja on havainnollistettu kuvassa 5.

Eräs viime aikoina suosiota saavuttanut käytettävyyden suunnittelutekniikka on etnograafinen haastattelu [1]. Etnograafisella haastattelulla tarkoitetaan immersiivisen tarkkailun ja ohjatun haastattelun yhdistelmää, jonka pyrkimyksenä on kerätä laadullista tietoa käyttäjistä ja heidän tavoitteistaan (Cooper & Reimann, 2003). Eräs etnograafisen haastattelun muoto on Beyerin ja Holtzblattin kehittämä *kontekstuaalinen tutkimus*

[1] Termi *etnograafinen haastattelu* on vapaa suomennos englanninkielen ilmauksesta *ethnographic interview*. Englanninkielinen termi *ethnography* puolestaan on antropologian termi, jolla tarkoitetaan immersiivistä ja systemaattista kulttuurien tutkimusta. Antropologiassa etnograafiset tutkijat viettävät vuosia eläen osana tutkimaansa kulttuuria (Cooper & Reimann, 2003).



Kuva 5: Käyttäjän osaamisen mallintaminen (Kearsley, 1988, s. 4).

(contextual inquiry). Kontekstuaalinen tutkimus on menetelmä loppukäyttäjien mallintamiseksi ja heidän tarpeittensa kartoittamiseksi. Menetelmän mukainen haastattelu tulisi suorittaa noudattaen seuraavia periaatteita (Beyer & Holtzblatt, 1998):

- Tietoa kerätään loppukäyttäjän normaalissa (työ)ympäristössä, eli siinä kontekstissa missä käyttäjä yleensä tekee tehtäviään.
- Tutkija ja käyttäjä pohtivat asioita yhdessä, haastattelun luonteen tulisi olla tutkijan ja käyttäjän välistä yhteistyötä: työn seuraamista ja siitä esille nousevista asioista keskustelua.
- Tulkinnalla on merkittävä osuus siinä, kuinka käyttäjän käytös, toimintaympäristö ja hänen mielipiteensä otetaan kokonaisuutena huomioon ja analysoidaan suunnitteluperiaatteiksi.
- Haastattelulla tulee olla selkeä fokus, ts. tutkija ja käyttäjä keskustelevat selvästi määriteltyjen aihealuiden pohjalta.

Päinvastoin kuin useat formaalit tutkimusmenetelmät, kontekstuaalinen tutkimus ei pyri suodattamaan tutkimustilanteesta pois mitään ennalta asetettuja muuttujia, vaan huomioi käytön kontekstin mahdollisimman autenttisena.

3.3 Käyttöliittymän suunnittelu

Käyttöliittymän suunnittelusta ja käytettävyydestä on olemassa paljon tutkimustietoa. Koska opastusjärjestelmä on itsessään käyttöliittymä, voidaan tämän tutkimuksen tuloksia soveltaa myös opastusjärjestelmien käyttöliittymien suunnitteluun.

Kuten kohdassa 2.3 viitatuissa tutkimuksissa todettiin, käyttäjät käyttäisivät mieluummin paperidokumentaatiota kuin opastusjärjestelmää. Osasyynä tähän voi olla se, että sähköisessä muodossa oleva teksti on Schneidermanin (1998) mukaan jo sinänsä huonommin luettava kuin paperille painettu teksti: sähköisen dokumentin luettavuutta vähentäviä seikkoja ovat mm. fonttien huonous, pieni kontrasti tekstin ja taustan välillä, näytön alttius heijastuksille tai välkkymiselle sekä näytön pinnan kuperuuden aiheuttama haitta. Lisäksi pientä näyttöä käytettäessä joudutaan sivua vaihtamaan usein, lukuetäisyys voi olla suurempi kuin paperilla, ja näytön sijoittelu väärä. Toisaalta Schneidermanin (1998) viittama Jornan (1991) tutkimus osoittaa, ettei lukunopeudessa tai havaitussa kuvan laadussa ole eroja, mikäli näytön resoluutio vastaa painetun paperin resoluutiota. Koska näytöt eivät vieläkään yllä resoluutioltaan paperin tasolle, on näytöltä lukeminen edelleenkin vaikeampaa kuin paperilta.

Edellä mainitut seikat huomioiden on selvää, ettei tietokonelaitteisto ole optimaalinen tapa esittää käyttäjälle tietoa ymmärrettävässä muodossa. Jotta laitteiston rajoitukset voitaisiin minimoida tulee opastusjärjestelmän käyttöliittymän suunnittelussa ottaa huomioon seuraavassa käsiteltäviä seikkoja. Huomionarvoisia seikkoja ovat mm. tekstin ladonta, kuvien ja värien käyttö, sekä kuvien ja tekstin keskinäinen suhde. Vaikka alan kirjallisuus sisältää paljon eri tutkijoiden ja käyttöliittymägurujen esittämiä (osin keskenään ristiriidassa olevia) ohjeita ja parhaita menetelmiä, on seuraavassa pitäydytty esittämään vain sellaisia seikkoja, joita voidaan tukea empiirisin havainnoin.

Käyttöliittymän tekstin ulkoasun osalta tulisi huomioida mm. seuraavat seikat. Kokonaan isoin kirjaimin kirjoitettua tekstiä tulisi käyttää harkiten: isolla kirjoitetut sanat on vaikeampi erottaa kuin pienellä kirjoitetut ja niiden lukemiseen kuluu enemmän aikaa (Horton, 1990). Tekstin rivit pitäisi pitää kohtuullisen lyhyinä, koska pitkän rivin lukeminen vaatii useita silmän liikkeitä ja vaikeuttaa uuden rivin alun löytämistä (Horton, 1990).

Värit vetävät huomiota puoleensa, helpottavat navigointia ja selausnopeutta sekä ilmentävät suhteita asioiden välillä (Cooper & Reimann, 2003). Värejä voidaan toisaal-

ta helposti käyttää väärin. Liika värien käyttö hämää ja hidastaa hakujen tekemistä. Komplementtivärejä ei tulisi käyttää yhdessä: niiden muodostamia kuvioihin on vaikeaa tarkentaa ja niitä on hankala havaita oikein (Cooper & Reimann, 2003). Samoin spektrin vastakkaisissa päissä olevien värien yhteiskäyttöä tulisi välttää: näin yhdistettynä värit näyttävät ”värähtelevän” – ilmiö tunnetaan nimellä *chromostereopsis* (Cooper & Reimann, 2003). Esimerkkejä vältettävistä väriyhdistelmistä ovat mm. sininen ja punainen, keltainen ja violetti, sekä magenta ja vihreä (Schneiderman, 1998). Käyttöliittymään ei koskaan pitäisi laittaa punaista väriä sinisellä taustalla tai päinvastoin. Värien käyttöä suunniteltaessa tulisi myös huomioida, että noin kahdeksalla prosenttilla Pohjois-Amerikan ja Euroopan väestöstä on jonkin asteinen puute värinäössä. Puutteista yleisin on puna-vihersokeus, jossa ihminen näkee sekä punaisen että vihreän värin harmaana (Schneiderman, 1998).

Opastusjärjestelmän käyttöliittymän suunnittelussa voidaan käyttää hyväksi kognitiivista mallintamista, josta esimerkkinä mainittakoon Cardin, Moranin ja Newellin (1983) kehittämä GOMS-analyysi. Johnin ja Kierasin (1996) mukaan GOMS (Goals, Operators, Methods & Selection rules) on malli, jonka avulla pyritään ennustamaan käyttäjän suoriutumista käyttöliittymän käytössä identifioimalla käyttäjän tavoitteet, operaattorit, metodit sekä valintasäännöt. Tavoitteella (goal) tarkoitetaan tässä käyttäjän lopullista toiminnan tavoitetta, esim. kirjeen kirjoittaminen tekstinkäsittelyohjelmalla. Tavoitteet ovat yleensä jaettavissa pienempiin osatavoitteisiin (subgoals). Operaattorit (operators) puolestaan ovat niitä toimintoja, joita ohjelmisto mahdollistaa käyttäjälle. Komentojonopohjaisissa käyttöliittymissä operaattoreilla tarkoitetaan komentojen ja parametrien yhdistelmää, graafisessa käyttöliittymässä esimerkiksi menuvalintoja, painikkeiden painamista tai suoraa muokkausta. Metodit (methods) ovat sarja hyvin opittuja operaattoreita ja osapäämääriä, joiden avulla voidaan päästä tavoitteeseen. Jos tiettyyn tavoitteeseen voidaan päästä useampaa eri metodia käyttämällä, tarvitaan valintasääntöjä (selection rules). Valintasäännöt ovat sellaisia henkilökohtaisia sääntöjä, joiden mukaan käyttäjä valitsee tietyssä tilanteessa käytettävän metodin. Yhdessä tavoitteet, operaattorit, metodit ja valintasäännöt tuottavat proseduraalista tietoa siitä, mitä käyttäjän tulee tietää tietyn tehtävän suorittamiseksi. Kognitiivisen mallintamisen etu on, että sen avulla voidaan arvioida loppukäyttäjän suorituskykyä ilman prototyyppiä tai käyttäjätestausta (John & Kieras, 1996).

Käyttöliittymän käytettävyyden mittaamiseen ja arviointiin on GOMS-analyysin lisäksi tarjolla lukuisia muitakin menetelmiä. Mack ja Nielsen (1994) luettelevat neljä ta-

paa käyttöliittymien arvioimiseksi: automaattinen-, empiirinen-, formaali- sekä epäformaali arviointi. *Automaattisessa arvioinnissa* käyttöliittymälle voidaan laskea käytettävyysarvio antamalla käyttöliittymän spesifikaatio syötteenä erityiselle arviointiohjelmalle. *Empiirisessä arvioinnissa* tehdään käytettävyystutkimus oikeilla käyttäjillä. *Formaalissa arvioinnissa* käytetään hyväksi tarkkoja malleja ja kaavoja, joilla käytettävyysarviolle voidaan laskea numeerinen arvo. Edellä kuvattu GOMS-analyysi ja sen lukuisat variantit lukeutuvat tähän kategoriaan. *Epäformaali arviointi* puolestaan perustuu nyrkkisääntöihin sekä arvioijan taitoihin ja kokemuksiin.

Mitä tekniikkaa sitten käytetäänkään, tulee opastusjärjestelmän helppokäyttöisyydestä ja selkeydestä varmistua. Koska opastusjärjestelmän käyttäjä on jo alkaessaan käyttää opastusjärjestelmää apua vailla tai kokonaan hukassa varsinaisen sovelluksen kanssa, ei itse opastusjärjestelmä saisi enää lisätä käyttäjän hämmennystä.

3.4 Opastusjärjestelmän rakentamisprosessi

Edellä perehdyttiin opastusjärjestelmien sisällön ja käyttöliittymän suunnitteluun liittyviin seikkoihin. Seuraavassa on tarkoitus käsitellä opastusjärjestelmien rakentamista kokonaisvaltaisemmin – projektinäkökulmasta.

Duffy, Palmer ja Mehlenbacher (1992) pyrkivät kuvaamaan opastusjärjestelmän rakentamisprosessin keräämällä tietoja 20:ltä alan ammattilaiselta. Opastusjärjestelmien suunnittelijoilta saamiensa tietojen perusteella Duffy, Palmer ja Mehlenbacher jakoivat opastusjärjestelmän rakentamisprossessin yhdeksään eri vaiheeseen:

1. Analysointi ja käytettävyyden testaussuunitelman tekeminen
2. Hallinnollinen suunnittelu
3. Sisällön suunnittelu
4. Käyttöliittymän suunnittelu
5. Protoilu
6. Ohjetekstin tuottaminen
7. Ohjeen linkittäminen sovellukseen

8. Tuotantotestaus ja laadunvarmistus

9. Tuotannonjälkeinen testaus ja seuranta.

Seuraavassa on kuvattu yllä mainittujen yhdeksän vaiheen tärkeimmät tehtävät sekä niiden suorittamiseen liittyvät yleisimmät ongelmat.

Ensimmäiseen vaiheeseen, *analysointiin ja käytettävyyden testaussunnitelman laatimiseen*, kuuluu mm. tutustuminen tuotteeseen, sen oletettuihin käyttötarkoituksiin sekä käyttäjiin. Vaiheen tärkein seikka on ymmärtää sovellus sen käyttäjän näkökulmasta. Loppukäyttäjien tarpeet, kyvyt ja odotukset voidaan selvittää analysointivaiheessa suorittamalla haastatteluita tai focus-group -tutkimuksia oletettujen loppukäyttäjien kanssa. Analysointivaiheessa kohdattu yleinen ongelmia ovat mm. se, että tuote on vielä kesken, eikä siitä ole olemassa dokumentaatiota ja vaatimusmäärittelyt ovat nekin mahdollisesti muutostilassa. Muita tyypillisiä analysointivaiheen ongelmia ovat puutteet tuotteen tiedoista, vaarimusmäärittelyn vajavuudet sekä vaikeudet saada tietoa tuotteen oletetuilta käyttäjiltä.

Toisessa vaihessa, *hallinnollisessa suunnittelussa*, määritellään vaatimukset opastusjärjestelmän esitysmuodolle ja käytettävälle ympäristölle, valitaan tai kehitetään tarvittavat työvälineet, jaetaan vastuualueet tiimin jäsenten kesken sekä arvioidaan resurssien (aika, raha, henkilöstö) tarve. Hallinnollisessa suunnittelussa eniten ongelmia tuottavat resurssien tarpeen tarkka arviointi etukäteen, sopiminen aikataulusta ja vastuualueista tiimin jäsenten kesken, muutoksiin sopeutuminen sekä tuotteen kehitystiimin kanssa työskentely – tuotantotiimi saattaa jäädä aikataulusta tai jättää sovelluksen käyttöliittymän tekemisen viimeiseksi. Myös resurssien hankinta voi olla ongelma, sillä sekä sovelluskehittäjät että johto näkevät opastusjärjestelmän rakentamisen usein alemman prioriteetin tehtävänä.

Hallinnollisen suunnittelun valmistuttua voidaan siirtyä seuraavaan, *sisällön suunnittelun*, vaiheeseen. Tässä vaiheessa suunnitellaan opastuksen kulku, linkit aihealuiden välillä sekä kirjoitetaan kehittäjille ohjeet sisällön tyylin, laajuuden, yksityiskohtaisuuden ja ymmärrettävyyden suhteen. Samoin tässä vaiheessa määritellään ohjeen tarkoitus, eli käytetäänkö sitä vasta-alkajien apuna vai referenssimanuaalina tai kenties opetuksessa. Myös grafiikan käytöstä sovitaan ja päätetään yhtenäisestä yrityksen tyylioppaasta tyylin ja muodon suhteen. Suurimpina sisällön suunnittelun ongelmina opastusjärjestelmien kehittäjät näkivät erottelun sähköisen dokumentaation ja opastusjär-

jestelmien välillä: kehittäjät kokivat vaikeaksi vakuuttaa muut siitä, että (a) opastusjärjestelmän sisältö ei saa olla yksinkertaisesti lyhennetty versio painetusta manuaalista tai (b) ettei se saa olla järjestetty täysin sovelluksen rakenteen mukaan. Muita yleisiä ongelmia ovat mm. puutteellisten tai virheellisten vaatimusmäärittelyiden löytäminen ja korjaaminen, joka on seurausta siitä, ettei loppukäyttäjien tarpeita ole ymmärretty. Samoin hankalaa on perehtyä sovellukseen perinpohjaisesti ja samalla pyrkiä säilyttämään vasta-alkajan näkökulma. Myös sovelluksen muuttuvat toiminnalliset vaatimukset aiheuttavat ongelmia sisällön suunnittelun vaiheessa.

Käyttöliittymän suunnittelun tärkeimpiä osatehtäviä on osajärjestelmän luonnostelu, jonka avulla voidaan havainnollistaa navigointia, ohjeiden tasoja, näytön asettelua yms. Muita tärkeitä tehtäviä ovat dialogi- ja esitystekniikan valitseminen, käytettävyystavoitteiden määrittely sekä mallien tekeminen opastusjärjestelmän näytöille. Yleinen käyttöliittymän suunnitteluun liittyvä ongelma on ymmärtää, kuinka oletettu käyttäjäryhmä todellisuudessa käyttää sovellusta. Toinen yleinen ongelma on opastusjärjestelmän toiminnallisuuden sovittaminen yhteen sovelluksen toiminnallisuuden kanssa siten, etteivät käyttäjät ”eksy” opastusjärjestelmään. Yleistä on myös vaikeudet sovittaa ideaalinen käyttöliittymä (eli tutkimuksen ja käyttäjäanalyysin perusteella paras mahdollinen käyttöliittymä) ja se mitä voidaan toteuttaa sovelluksen ja / tai sovellusalustan rajoitukset huomioiden. Joskus voi olla myös vaikeaa päästä yli subjektiivisistä käsityksistä koskien sitä, mikä on hyvää.

Protoiluvaihe on osatehtävien osalta suoraviivainen: prototyyppi tehdään ja se integroidaan sovellukseen. Tämän jälkeen tehdään vielä kaksi tai useampia prototyyppejä, joita testataan ja arvioidaan suhteessa ensimmäiseen prototyyppiin. Lopuksi muokataan suunnittelussa tehtyjä määrityksiä saatujen kokemusten perusteella. Käytännössä opastusjärjestelmien kehittäjät kohtaavat suuria ongelmia protoiluvaiheessa. Ongelmat liittyvät suurimmalta osin resurssien saamiseen: prototyypin tekemiseen tai valmiin prototyypin evaluointiin ei riitä aikaa tai rahaa, ei löydy riittävän kokeneita ihmisiä suunnittelemaan ja toteuttamaan prototyypin evaluointia ja testausta tai ei löydy sopivaa työvälinettä prototyypin tekemistä ja evaluointia varten. Samoin voi olla vaikeuksia löytää loppukäyttäjiä proton arviontia varten.

Ohjetekstin tuottamisvaiheessa määritellään linkit ohjeen ja sovelluksen eri tilojen välille, katselmoidaan sovellusta ajoittain, jotta mahdolliset muutokset saadaan selville, koordinoidaan sovelluksen ja painetun manuaalin dokumentaation välillä, hahmotellaan opastusjärjestelmän sisältö aihealueittain ja proseduureittain. Ohjetekstin tuotta-

misvaiheen lopuksi paketoidaan teksti soveltuvaan formaattiin ja määritellään hakusanat ohjeesta etsimistä varten. Suurin ongelma ohjetekstin tuottamiseen liittyen on sellaisen tekstin kirjoittaminen, joka on ymmärrettävää ja auttaa käyttäjää ratkaisemaan tietyn ongelman, muttei hämmennä tätä liialla informaatiolla. Hankaluuksia voi tuottaa myös muuttuvaan käyttöliittymään tai toiminnallisuuteen mukautuminen, mikä voi aiheuttaa muutoksia myös ohjetekstissä. Myös loppukäyttäjän tiedontarpeen ennustaminen voi olla vaikeaa samoin kuin aikataulussa pysyminen. Muita opasteiden kehittäjien huolia ovat mm. versionhallintaongelmat, tekstin muuttaminen haluttuun muotoon, tehokkaan yhteistyön ja koordinaation ylläpitäminen painetun dokumentaation tekijätiimin kanssa sekä se, kuinka välttää kiusaus kirjoittaa uutta materiaalia silloin kun se ei ole välttämätöntä.

Ohjeen linkittäminen sovellukseen voi tapahtua jo prototyyppivaiheessa tai vasta kun opastusjärjestelmä on kokonaan valmis. Tässä vaiheessa todetaan, onko kaikki opastusjärjestelmälle tarkoitettu toiminnallisuus todella käytettävissä. Samoin suoritetaan testaus koko sovellukselle käyttämällä myös opastusjärjestelmää, jolloin nähdään kuinka tehokas opastusjärjestelmä oikein on. Yleisesti kohdattuja ongelmakohtia ovat mm. kuinka ohje linkitetään sovellukseen, tekniset ongelmat, jotka liittyvät ikkunointijärjestelmän mahdollisuuksiin ja näytön käsittelyyn, sekä käytetyn käyttöjärjestelmän ominaisuuksiin sopeutuminen. Hankaluuksia voi aiheutua myös opastustiedostojen formaatista, kun siirrytään tekstinkäsittely- tai kehitysmuodosta tuotantoformaattiin. Joskus myös ohjelmointiresurssien saaminen voi olla vaikeaa.

Kun opastusjärjestelmä on valmis ja integroitu sovellukseen, on aika suorittaa *tuotantotestaus ja laadunvarmistus*. Tässä vaiheessa tehdään suorituskykymittauksia, joilla selvitetään onko asetetut tavoitteet saavutettu. Käyttöliittymän yhdenmukaisuus ja yleinen ulkonäkö tarkastetaan. Samoin suoritetaan lopullisen tuotteen asennustestaus ja tehdään käyttäjätestaus ohjetekstin luonnoksella. Tässä vaiheessa myös varmistetaan, että opasteen aihealueet vastaavat sovelluksen aihealueita. Opastustekstiä voidaan vielä muokata ja lopuksi viimeiset versiot lisätään järjestelmään.

Opastusjärjestelmän rakentamisprosessin viimeinen vaihe on *tuotannonjälkeinen testaus ja seuranta*. Tähän vaiheeseen kuuluu sen varmistaminen, että käyttäjiltä saadaan palautetta opastusjärjestelmästä sen julkaisun jälkeen. Samoin kenttätestausta jo julkaistun tuotteen osalta voidaan tehdä. Tässä vaiheessa tulisi myös suunnitella miten seurataan tuotteen uuden version kehittämistä, jotta uusi opastusjärjestelmä voidaan julkaista sen mukana. Suunnitella tulee myös se seikka, kuinka opastustekstit muoka-

taan tarvittaessa tuotteen julkaisun jälkeen. Yleisin ongelma tuotteen ollessa jo tuotannossa on laadukkaan tiedon saaminen tuotteessa mahdollisesti olevista puutteista. Voi olla myös vaikeaa löytää henkilö vastaamaan opastusjärjestelmän päivityksestä sen jälkeen kun projekti on valmis. Kolmas ongelma on uudelleenjulkaisujen rajoittaminen – päivitykset ovat yleensä luonteeltaan ylläpitojulkaisuja, joilla on tiukat aikarajoitukset.

Edellä kuvattu rakentamisprosessin kuvaus syntyi Duffyn, Palmerin ja Mehlenbacherin (1992) yhdeksänkymmentäluvun alussa opastusjärjestelmien suunnittelijoiden keskuudessa tekemästä haastattelututkimuksesta. Kuvaus ei siis edusta mitään ideaalia tavoitetta prosessin kulusta, vaan pikimminkin tuona aikana vallinnutta käytäntöä – haastattelututkimuksen mahdolliset vääristymät huomioituna.

Opastusjärjestelmän rakentamisprojekti ei eroa merkittävästä itse sovelluksen rakentamisprojektista. Nykyaikana, sulautettujen opasteiden suosion kasvaessa, opastusjärjestelmän rakentaminen integroituu entistä enemmän osaksi varsinaisen ohjelmiston rakentamisprosessia. Siinä missä edellä kuvattu rakentamisprosessi oli selkeästi erotettu oma kokonaisuutensa, saattaa opastusjärjestelmän rakentaminen nykyään olla pikemminkin osa ohjelmistoprojektia kuin oma projektinsa.

4 Opastusjärjestelmän rakentaminen

Tässä luvussa perehdytään opastusjärjestelmän rakentamiseen käytännössä. Aluksi esitellään yleisimmät nykyisin markkinoilla olevat tehtävään soveltuvat tekniikat ja työvälineet sekä sopivan vaihtoehdon valintaan vaikuttavat seikat. Tämän jälkeen tutustutaan lyhyesti tutkielmaa varten tehtyyn esimerkkisovellukseen PingPaliin, jonka jälkeen kuvataan kolmen eri opastusjärjestelmän rakentamisprosessit PingPal-sovellukselle. Lopuksi vielä vertaillaan rakentamisprosessissa esiin tulleita asioita ja arvioidaan työkalujen soveltuvuutta ja niillä aikaansaatujen opastusjärjestelmien laatua.

4.1 Markkinoilla olevat kehitysympäristöt

Kirjoitushetkellä markkinoilla on lukuisia työvälineitä opastusjärjestelmien rakentamiseen, jotka eroavat ominaisuuksiltaan ja kohdejärjestelmiltään. Valtaosa työvälineistä on HTML-pohjaisia. HTML:ään perustuvat opastusjärjestelmät koostuvat HTML-kielellä kirjoitetuista opastussivuista sekä GIF-, JPG- tai PNG -muotoisista kuvatiedostoista. Yleensä järjestelmiä voidaan laajentaa mm. Java-, JavaScript-, ActiveX- tai VBScript -tekniikoita käyttämällä.

HTML-pohjaisia opastusjärjestelmien rakennusvälineitä ovat mm. WinHelp, HTML Help, WebHelp, JavaHelp sekä Oracle Help for Java. Näistä ensin mainittu on jo siirtymässä pois käytöstä ja Microsoftin tuki painottuu HTML Help -tekniikalle Windows-sovellusten opastusjärjestelmien rakentamisalustana. WebHelp on eHelp Corporationin kehittämä työkalu alustariippumattomien web-pohjaisten opastusjärjestelmien rakentamista varten (eHelp Corporation, 2004). JavaHelp ja Oracle Help for Java ovat molemmat Java-kieleen pohjautuvia tekniikoita, joiden avulla rakennettuja opastusjärjestelmiä voidaan käyttää kaikissa järjestelmissä, joihin on saavilla Java-tulkki (Java Runtime Environment). Käytännössä Java-pohjaisten opastusjärjestelmien käyttö rajoittuu Javalla tehtyjen sovellusten yhteyteen, vaikkei mikään periaatteessa estä Java-pohjaisen opastusjärjestelmän käyttöä esim. C++:lla tai Visual Basicilla tehdyn sovelluksen opasteena. Java-pohjaisten opastusjärjestelmien etuna on mm. alustariippumattomuus – ne toimivat myös muissa kuin Windows-käyttöjärjestelmissä.

Ennen tekniikan valitsemista, tulisi opastusjärjestelmän suunnittelijan pohtia mm. seuraavia loppukäyttäjän tekniseen ympäristöön liittyviä asioita (James-Tanny, 2002):

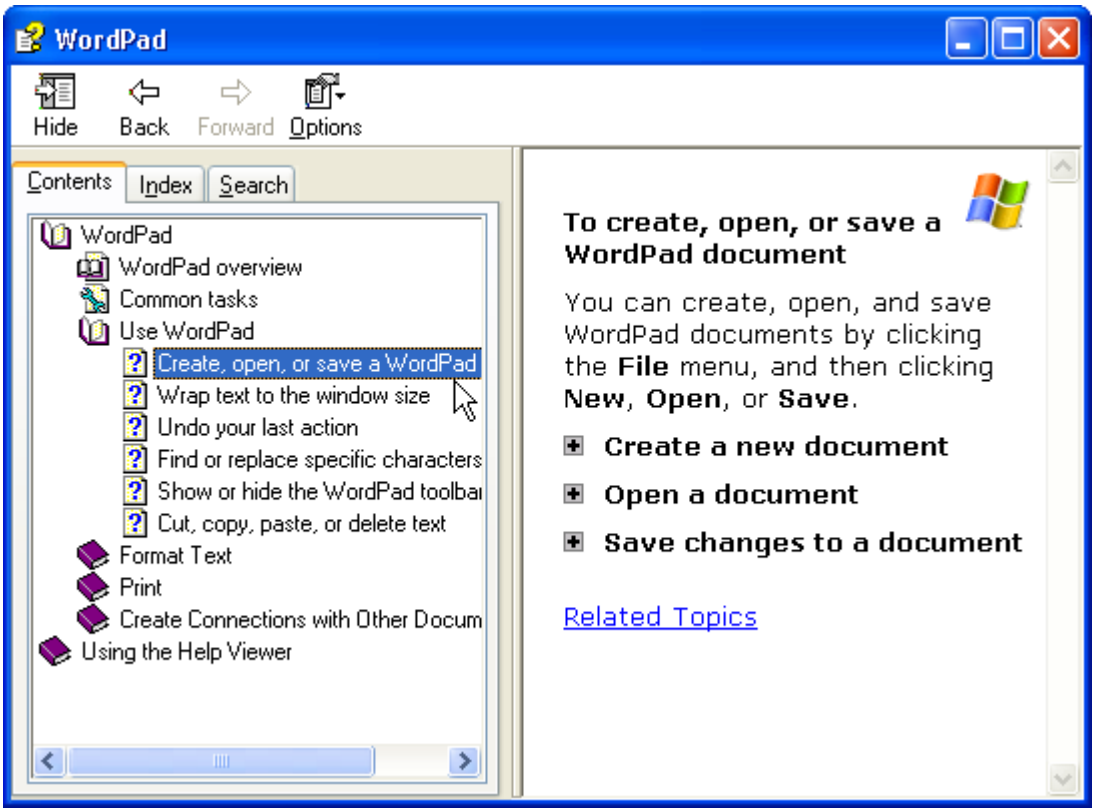
- Millä alustalla opastusjärjestelmän tulee toimia (Windows, Mac OS, jokin UNIXin versio)? Tuleeko ohjelmistosta versioita mahdollisesti useammalle alustalle, jolloin opastusjärjestelmän olisi toivottavaa toimia niissä kaikissa?
- Mitä Internet-selaimia loppukäyttäjällä on käytössä (Internet Explorer, Netscape, Opera)? Pitäisikö opastujärjestelmän toimia selaimen avulla, vai tuleeko sen olla itsenäinen sovellus?
- Onko käyttäjillä pääsy Internetiin, ja kuinka nopea yhteys heillä on käytössään? Tämä vaikuttaa mm. siihen, voidaanko opastusjärjestelmää päivittää netin kautta tai jopa säilyttää kaikki opastustekstit käyttäjän kovalevyn sijasta palvelimella.
- Tietoturvanäkökohdat, kuten voiko käyttäjän koneelle ladata ja suorittaa esim. appletteja tai ActiveX-kontrolleja, vai pitääkö opasteiden perustua puhtaaseen HTML:ään?

Opastusjärjestelmän käyttöympäristön lisäksi myös kehitystyössä käytetyillä työkaluilla on merkitystä sopivaa tekniikkaa valittaessa. Opastusjärjestelmää suunniteltaessa kannattaa selvittää onko ohjetekstiä tuotettu jo Microsoft Wordillä, Adobe FrameMakerilla tai HTML-muodossa tai onko tekstiä tallennettu tietokantaan (James-Tanny, 2002)?

Varsinaisten toteutustekniikoiden lisäksi markkinoilla on olemassa lukuisia opastusjärjestelmän laatimistyökaluja, joita käyttämällä voidaan luoda HTML Helpin, JavaHelpin tai muun vastaavan tekniikan mukaisia ohjetiedostoja pelkistä opastusteksteistä. Näiden työkalujen perusideana on, että yhdestä opastemateriaalista voidaan tuottaa opastusjärjestelmä useisiin eri kohdeympäristöihin helposti, ilman tarvetta ohjelmointiin. Tällöin opastusjärjestemän tekijän tehtäväksi jää huolehtia opastusjärjestelmän sisällöstä laatimistyökalun generoidessa halutun kohdejärjestelmän opastusjärjestelmän. Seuraavassa perehdymme muutamaan yleiseen opastusjärjestelmän rakentamista varten tehtyyn tekniikkaan: HTML Helpiin, JavaHelpiin, sekä WebHelpiin.

HTML Help

HTML Help 1.4 on kirjoitushetkellä Microsoftin uusin tuotantokäytössä oleva opastusjärjestelmien kehitysympäristö Windows-käyttöjärjestelmälle. HTML Help toimii kaikissa 32-bittisissä Windows-käyttöjärjestelmissä, joihin on asennettu Internet Explorer



Kuva 6: Esimerkki HTML Help -opastusjärjestelmä.

-selain. Opastustekstitiedostot käännetään ja pakataan .chm-tiedostoiksi, jolloin ne saadaan mahtumaan huomattavasti pienempään tilaan kuin pakkaamattomat .html-tiedostot. Navigointitavoista HTML Help tarjoaa sisällysluettelon, monitasoisen hakemiston, kokotekstihaun (full-text search) sekä suosikit-välilehden (DeLoach, 2001).

Ikkunoiden osalta HTML Help tarjoaa kolmiosaisen ikkunan (esitetty kuvassa 6), työkaluvalikon sekä tekstimuotoiset pop-up-ikkunat. Ohje on jaettu kolmeen paneeliin, joista suurin on opastustekstipaneeli (topic pane), jossa varsinaiset ohjetekstit ovat. Tästä vasemmalla on on navigointipaneeli (navigation pane), josta voidaan valita näytettäväksi joko sisällysluettelo, hakemisto tai tekstihaku. Ikkunan yläosassa on työkalupaneeli, jonka sisältämät toiminnot on mahdollista valita opastusjärjestelmän luontivaiheessa.

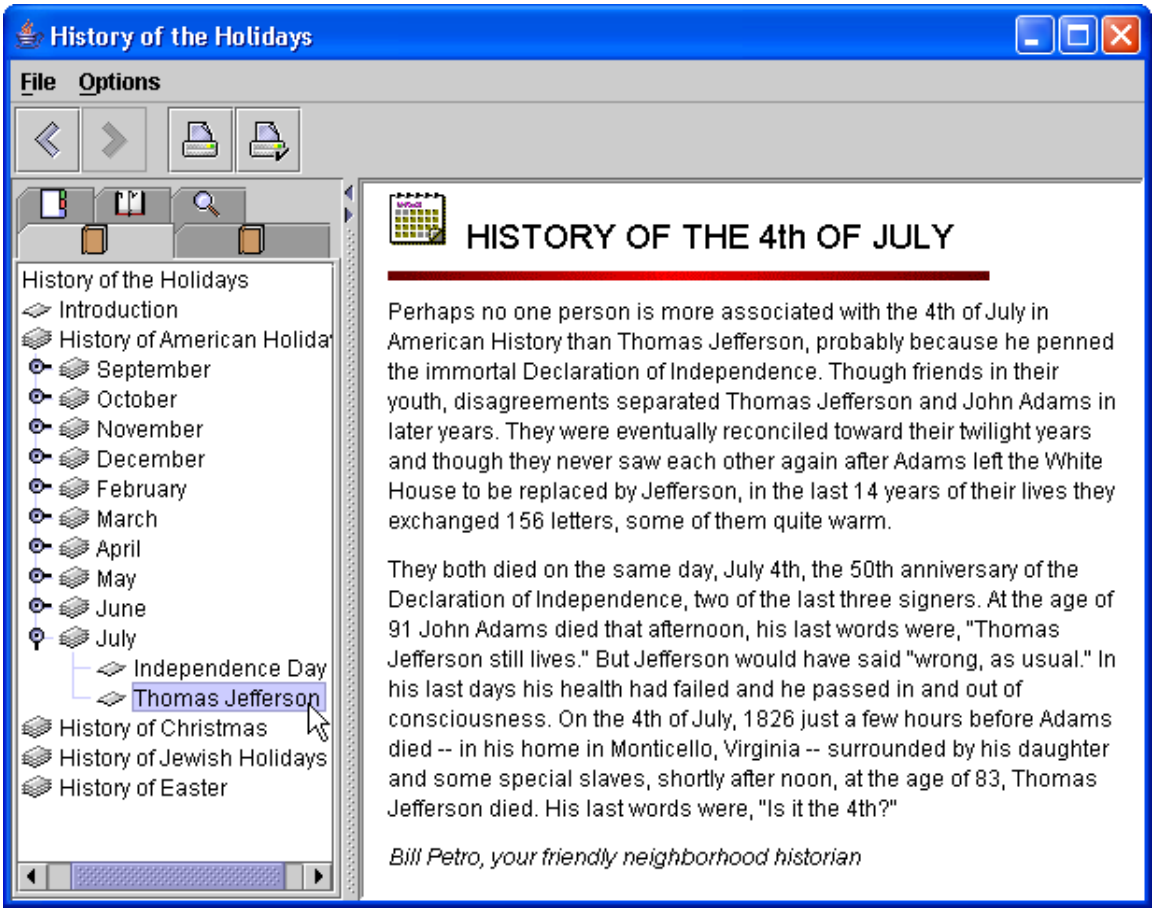
Kontekstisidonnaisuus on toteutettu numeroiden ja aliasten avulla ja myös sivujen dynaaminen linkittäminen on mahdollista. Rajoituksena voidaan mainita, että HTML Help vaatii kohdejärjestelmään asennetuksi vähintään Internet Explorer version 3.02 (versio 4.0 tarvitaan kehittyneempien ominaisuuksien kuten DHTML hyödyntämisek-

si) (DeLoach, 2001).

JavaHelp

JavaHelp on Sun Microsystemsin Java-teknologiaan perustuva tekniikka alustariippumattoman opastusjärjestelmän kehittämistä varten. JavaHelp tukee HTML 3.2 -muotoisten opastustiedostojen näyttämistä. Se ei tue DHTML:ää, JavaScriptiä, VBScriptiä, ActiveX:ää, kuvakarttoja (image maps), sähköpostilinkkejä, avi-videokuvaa eikä wav-äänitiedostoja (DeLoach, 2001). JavaHelpin etu on se, että se on kehitetty Java-kielellä, minkä vuoksi opastusjärjestelmän käyttöliittymä ja toiminnallisuus on helposti kustomoitavissa kulloiseenkin tarkoitukseen soveltuvaksi. Toinen kiistaton etu on Javan mukanaan tuoma alustariippumattomuus: jos Java-sovellus toimii jossakin tietyssä ympäristössä, myös JavaHelpillä toteutettu opastusjärjestelmä toimii samassa ympäristössä. JavaHelp ei myöskään ole riippuvainen mistään kolmannen osapuolen tarjoamasta selaimesta: JavaHelp käyttää omaa selaintaan opastustekstien näyttämiseen. JavaHelp tarjoaa lisäksi useita esitysmuotoja ohjeelle: opastusjärjestelmä voidaan tehdä erilliseksi järjestelmäksi (standalone), kontekstiriippuvaiseksi tai sulautetuksi. Myös ponnahdusikkunoiden, sekundaaristen ikkunoiden sekä multimedian integrointi on mahdollista (Lewis, 2000).

JavaHelpin ensimmäinen versio julkaistiin maaliskuussa 1999. Tätä seurasi versio 1.1.3 ja myöhemmin vuonna 2003 versio 2.0 beta (Leritz-Higgins, 2003). JavaHelp versio 2.0 betan lisäominaisuuksia edeltäjiinsä nähden ovat mm. sanasto (glossary) sekä suosikit (favorites) -välilehdet navigointipaneelissa ja mahdollisuus liittää yhteen (merge) useampia opastusjärjestelmäprojekteja. Mittava parannus on myös tuki palvelinpohjaisen (server based) opastusjärjestelmän rakentamista varten. Palvelinpohjaisessa opastusjärjestelmässä opastussivut sijaitsevat Internet-palvelimella, josta client-sovellukset hakevat sivut dynaamisesti käyttäjän koneeseen aina tarvittaessa (Leritz-Higgins, 2003). Mikäli JavaHelp-opastusjärjestelmä on tarkoitus asentaa paikallisesti käyttäjän työasemalla, voidaan se toimittaa joko joukkona kääntämättömiä lähdetiedostoja tai yhtenä pakattuna jar-tiedostona (Java Archive File) (Deloach, 2001). Kuvassa 7 on esimerkki JavaHelp-opastusjärjestelmästä.



Kuva 7: JavaHelp versio 2.0 Beta.

WebHelp

WebHelp on eHelp-nimisen yrityksen kehittämä opastusjärjestelmäformaatti, joka on alusta- ja selainriippumaton. WebHelp-opastusjärjestelmä voidaan tehdä RoboHelp-nimistä opastusjärjestelmän rakentamistyökalua käyttäen ja varsinaiset opastustekstit voidaan kirjoittaa joko RoboHelpin omaa editoria, Microsoft Wordia tai mitä tahansa HTML-editoria käyttäen. WebHelp-opastusjärjestelmä voidaan asentaa intranettiin, extranettiin tai Internetiin ja Windows-, Linux- tai Mac -käyttöjärjestelmällä varustettuihin koneisiin (eHelp Corporation, 2004).

eHelp Corporationin (2004) mukaan WebHelp-opastusjärjestelmä tukee HTML:n lisäksi perusnavigointiominaisuuksia kuten sisällysluettelo, monitasoinen hakemisto, kokotekstihaku (mukaan lukien boolean operaattorit), kontekstisidonnainen opastus, sekä muut aiheeseen liittyvät aiheet (related topics) -kontrolli. WebHelp-

opastusjärjestelmän opastustekstejä ei käännetä tai pakata, vaan ne toimitetaan sellaisenaan, jolloin yksittäisen opastustekstin päivittäminen on yksinkertaista, joskin opastusjärjestelmä vie pakkaamattomana enemmän levytilaa. WebHelp Pro tarjoaa edellä mainittujen lisäksi mahdollisuuden kerätä reaaliaikaista tietoa käyttäjän tavasta käyttää sovellusta ja opastusjärjestelmää, haun luonnollista kieltä käyttäen, sekä monen kehittäjän tuen (eHelp Corporation, 2004).

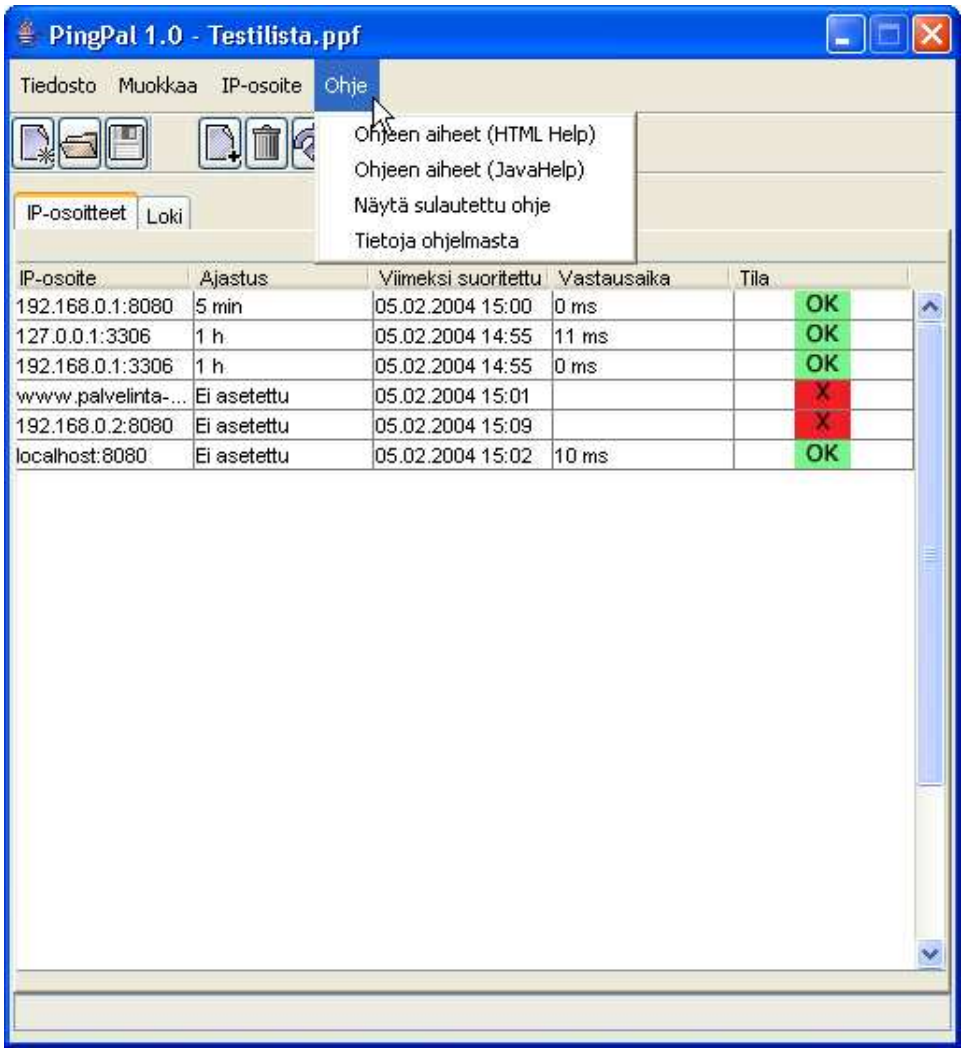
Edellä kuvatuista tekniikoista HTML Help sekä JavaHelp ovat opastusjärjestelmien kehittäjille ilmaisia. Viimeksi kuvattu WebHelp on kaupallinen opastusjärjestelmäformaatti, jota tuotetaan RoboHelp-nimistä, maksullista, työkalua käyttäen. HTML Help ja WebHelp valittiin esiteltäviksi niiden suuren käyttäjämäärän takia: WinWritersin (2003) kyselyn mukaan 64 prosenttia käytti WebHelp-formaattia tuottavaa RoboHelpiä ensisijaisena työkalunaan ja HTML Help Workshopia puolestaan 35 prosenttia vastanneista. JavaHelp ei nauti alan ammattilaisten keskuudessa ainakaan vielä aivan yhtä suurta suosiota. Kyseinen tekniikka otettiin mukaan tutkielmaan, koska alustariippumattomana se tarjoaa vastakohdan HTML Helpille.

4.2 PingPal-esimerkkisovellus

Opastusjärjestelmän rakentamista varten rakennettiin esimerkkisovellus PingPal, joka on Java-pohjainen, Internet-palvelinten tilanhallintaan tarkoitettu työkalu. PingPalin avulla verkon ylläpitäjä tai muu henkilö voi pitää yllä listaa palvelimista, joita sovellus ”pingaa” eli testaa yhteyden toimivuuden säädetyin väliajoin ja ilmoittaa ylläpitäjälle, mikäli jokin annetuista palvelimista ei vastaa lähetettyyn pyyntöön. Kuvassa 8 on esitetty PingPalin käyttöliittymä.

PingPal versio 1.0 sisältää mm. mahdollisuuden tallentaa mielivaltaisen monta erillistä palvelinlistaa. Listoihin voidaan lisätä palvelinten osoitteita ja niitä voidaan myöskin poistaa listoilta. Osoitteita voidaan testata joko yksitellen listalta valitsemalla tai ajastetusti, jolloin jokaiselle osoitteelle voidaan määritellä aikaväli, jonka puitteissa palvelimen tila tarkistetaan. Palvelimen osoitteeseen voidaan myös lisätä testattava portti, joka on oletuksena WWW-portti numero 80. PingPal-sovellus testaa käyttäjän antamia palvelinosoitteita ja ilmoittaa, mikäli jokin palvelimista ei vastaa.

PingPalin ohje-valikko eroaa hiukan tyypillisestä sovelluksen ohje-valikosta (kuva 8). Tutkielmaa varten PingPal-sovellukselle tehtiin yhteensä kolme eri opastusjärjestel-



Kuva 8: PingPal-sovelluksen käyttöliittymä.

mää: kohta kuvattava HTML Help -pohjainen toteutus, JavaHelp-toteutus, sekä JavaHelpillä tehty sovellukseen sulautettu opastusjärjestelmä.

PingPal on kirjoitettu Java-ohjelmointikielellä NetBeans IDE 3.5.1 -kehitysympäristöä käyttäen. Sovellus on testattu Windows XP -käyttöjärjestelmässä.

4.3 HTML Help -opastusjärjestelmän rakentaminen

Seuraavassa kuvataan opastusjärjestelmän rakentamisprosessi käytännössä Windows-ympäristöön. Työkaluksi tehtävään on valittu Microsoftin HTML Help Workshop, koska se on ilmainen ja yleisesti käytetty. Aluksi kuvataan lyhyesti tehtävässä käytettävä

työkalu, jonka jälkeen esitellään itse rakentamisprosessi pääpiirteissään.

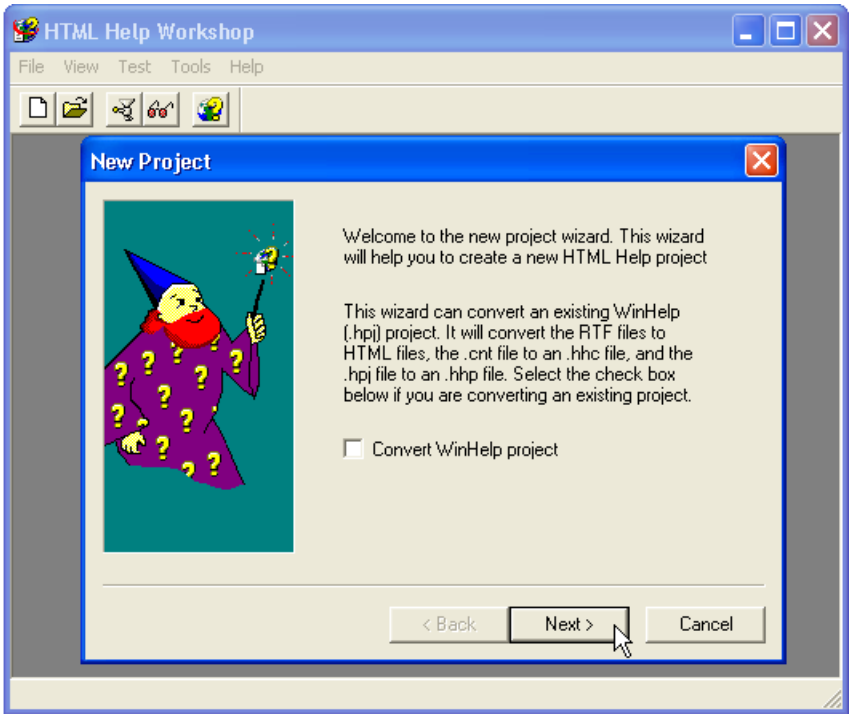
HTML Help Workshop -työkalu

HTML Help -opastusjärjestelmä voidaan rakentaa käyttämällä Microsoftin ilmaista HTML Help Workshopia. Kyseessä on opastusjärjestelmän laatimistyökalu, joka sisältää seuraavat komponentit (Microsoft, 2004):

- Editori HTML-muotoisten opastussivujen luomista varten.
- ActiveX-kontrolli (Hhctrl.ocx), joka on pieni modulaarinen ohjelma navigointipaneelin ja sekundaaristen ikkunoiden toteuttamista varten. Ohjelma mahdollistaa laajennettavan sisällysluettelon, sanahaun, pikalinkit (short-cuts) sekä ponnahdusikkunat (pop-up help topics).
- HTML Help Java Applet, joka on ActiveX-kontrollia vastaava Java-kielinen toteutus. Appletti mahdollistaa sisällysluettelon, hakemiston sekä liittyvien aiheiden (related topics) liittämisen opastusjärjestelmään.
- HTML Help Viewer on selain, jolla valmiita opastusjärjestelmiä katsotaan. Selain tarvitsee toimiakseen vähintään Internet Explorer versio 3:n.
- HTML Help Compiler, kääntäjä jolla tehdyt opastustekstit voidaan kääntää ja pakata pienempään tilaan.
- HTML Help Image Editor, kuvankaappauksiin ja kuvamuotojen vaihtamiseen soveltuva työkalu.
- HTML Help suoritustiedosto (Hh.exe), jonka avulla käännetty opastusjärjestelmätiedosto (.chm) voidaan suorittaa. HTML Help suoritustiedosto kutsuu HTML Help ActiveX-kontrollia, joka avaa opastustiedoston ja toteuttaa navigoinnin ja muun toiminnallisuuden käyttäjälle.

HTML Help Workshop sisältää kaiken tarvittavan yksittäisten opastussivujen (.html), grafiikan (.gif, .jpg, .png), sisällysluettelon (.hhc), hakemiston (.hhk) luomista varten ja tallentamista yhdeksi projektitiedostoksi (.hhp) ja lopulta kääntämistä valmiiksi opastusjärjestelmäksi (.chm). HTML Help Workshop voidaan ladata Microsoftin kotisivuilta ilmaiseksi. Asennuspaketti on kooltaan n. 3,5 Mt ja vaatii asennettaessa kiintolevyltä

tilaa n. 6 Mt. Itse asennusprosessi on suoraviivainen ja yksinkertainen. Kuvassa 9 on esitetty HTML Help Workshopin version 4.74 käyttöliittymä.



Kuva 9: HTML Help Workshop käynnistyksen jälkeen. Velho johdattelee käyttäjää luomaan uutta opastusjärjestelmäprojektia.

Opastusjärjestelmän rakentaminen

Opastusjärjestelmän (samoin kuin itse sovelluksen) rakentamista tulisi aina edeltää kattava loppukäyttäjän mallintaminen esimerkiksi kohdassa 3.2 kuvattua kontekstuaalista tutkimusmenetelmää käyttäen. Tutkielman aikarajoitukset huomioiden ei PingPal-sovellusta tai seuraavassa kuvattavien opastusjärjestelmien rakentamista varten voitu tehdä kunnollista käyttäjän mallintamista. Koska kysessä on uusi tuote, eikä käyttäjäpalautetta aikaisemmista versioista täten ole, päädyttiin loppukäyttäjän määrittelyssä oletukseen, että loppukäyttäjien ryhmä koostuu pääosin hyvin teknisistä, eksperttitason tietokonekäyttäjistä, jotka hallitsevat myös PingPalin sovellusalueeseen liittyvät tiedot ja taidot. Tarvittaessa loppukäyttäjä osaisi hoitaa PingPalin mahdollistamat toiminnot vaikkapa Windowsin tai UNIX:n komentotyökaluja käyttäen. PingPal-sovelluksen perimmäisenä tarkoituksena on yksinkertaistaa ja automatisoida näiden toimintojen suo-

rittamista. Näin sen käytettävyyden lopullisena mittarina voidaan pitää sitä, että sovellus on helpompi ja nopeampi käyttää kuin erillisen komentojonoskriptin kirjoittaminen ja suorittaminen.

Edellä mainitut seikat huomioiden opastusjärjestelmän rakentamisessa edetään seuraavaan vaiheeseen, jonka tarkoituksena on määritellä ohjelmassa olevat opastusta kaipaavat toiminnot, eli opastusjärjestelmään tulevat opastusaiheet. Koska kyseessä on suhteellisen yksinkertainen sovellus, saatiin tämän vaiheet tulokset tiivistettyä taulukkoon 4. Opastusaiheet jakautuvat luontevasti seuraaviin aihealueisiin

- osoitelistat
- osoitteiden käsittely
- loki.

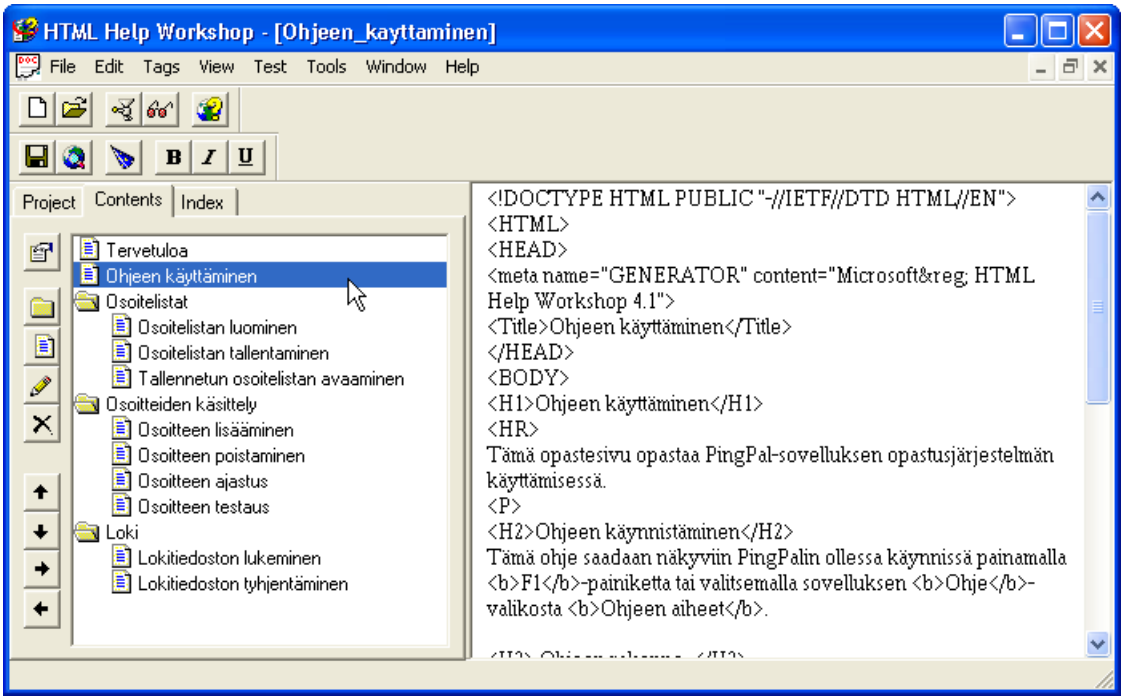
Edellä mainittujen opastusaiheiden lisäksi opastusjärjestelmän alkuun lisätään vielä *Ohjeen käyttäminen* -aihe, jollaisen tulisi olla helposti löydettävissä jokaisessa opastusjärjestelmässä (Mazur, 1995).

Edellä kuvattujen suunnitteluvaiheiden jälkeen siirrytään itse opastusjärjestelmän toteuttamiseen HTML Help Workshop -työkalulla. Opastusjärjestelmää varten luodaan uusi projekti, jonka jälkeen kirjoitetaan HTML-pohjaiset opastussivut taulukossa 4 listattuja opastusaiheita varten. Jokaista opastusaihetta varten kirjoitetaan oma sivu. HTML Help Workshop tekee uudelle HTML-sivulle mallipohjan, josta opastusjärjestelmän tekijä voi jatkaa lisäämällä otsikon, leipätekstin sekä muut ohjeessa tarvittavat elementit. Opastus pyritään pitämään mahdollisimman selkeänä ja tehtäväorientoituneena, askel askeleelta etenevänä ohjeena. Pitkiä tekstilohkoja pyritään välttämään ja asiat esitetään mahdollisuuksien mukaan listoina. Purchasen ja Worrillin (2002) kehoitusta noudattaen opastusaiheet pyritään tekemään mahdollisimman syklisiksi lisäämällä opastussivujen alaosaan sisäisiä linkkejä muihin aiheeseen liittyviin sivuihin, esimerkiksi *Osoitteen lisääminen* -opastusaiheeseen liitetään linkit aiheisiin *Osoitteen poistaminen*, *Osoitteen testaaminen* sekä *Osoitteen ajastus*. Kaikki esimerkkisovelluksen opastussivut on esitetty liitteessä 2.

Kun HTML-sivut ovat valmiit linkitetään ne opastusjärjestelmän sisällysluetteloon. Käytännössä tämä tarkoittaa, että jokaiselle sivulle määritellään sisällysluettelossa näy-

Taulukko 4: PingPal-sovelluksen opastusta tarvitsevat ominaisuudet.

Toiminto	*Toiminnon kuvaus*
Luo uusi osoitelista.	Luo uuden listan palvelinosoitteiden syöttämistä varten.
Avaa aikaisemmin luotu osoitelista.	Avaa näytölle tiedosto-dialogin, jolla voidaan avata sovelluksen käsittelemiä osoitelistatiedostoja käyttäjän määräämästä hakemistosta.
Talleta luotu osoitelista.	Avaa näytölle tiedosto-dialogin, josta käyttäjä voi valita tiedostolistalleen haluamansa nimen sekä hakemiston johon tiedosto tallennetaan.
Lisää osoite listaan.	Lisää yksittäisen osoitteen osoitelistaan.
Poista osoite listalta.	Poistaa osoitteen osoitelistalta.
Testaa yhden osoitteen toiminta.	Testaa valitun osoitteen toimivuuden (ts. vastaako palvelin annetussa osoitteessa).
Ajasta osoitteen testaus.	Käyttäjä voi määritellä jokaiselle osoitteelle oman aikajakson (10:stä sekunnista yhteen viikkoon), jonka välein kyseinen osoite testataan.
Käynnistä kaikkien osoitteiden ajastettu testaaminen.	Testaa kaikki osoitteet, niille säädetyin väliajoin toistuvasti.
Pysäytä ajastettu testaaminen.	Pysäyttää ajastetun testaamisen.
Näytä loki.	Näyttää lokitiedoston sisällön näytöllä (lokissa näytetään kaikki yhteydenmuodostusyritykset, niin onnistuneet kuin epäonnistuneetkin).
Tyhjennä loki.	Tyhjentää lokitiedoston.



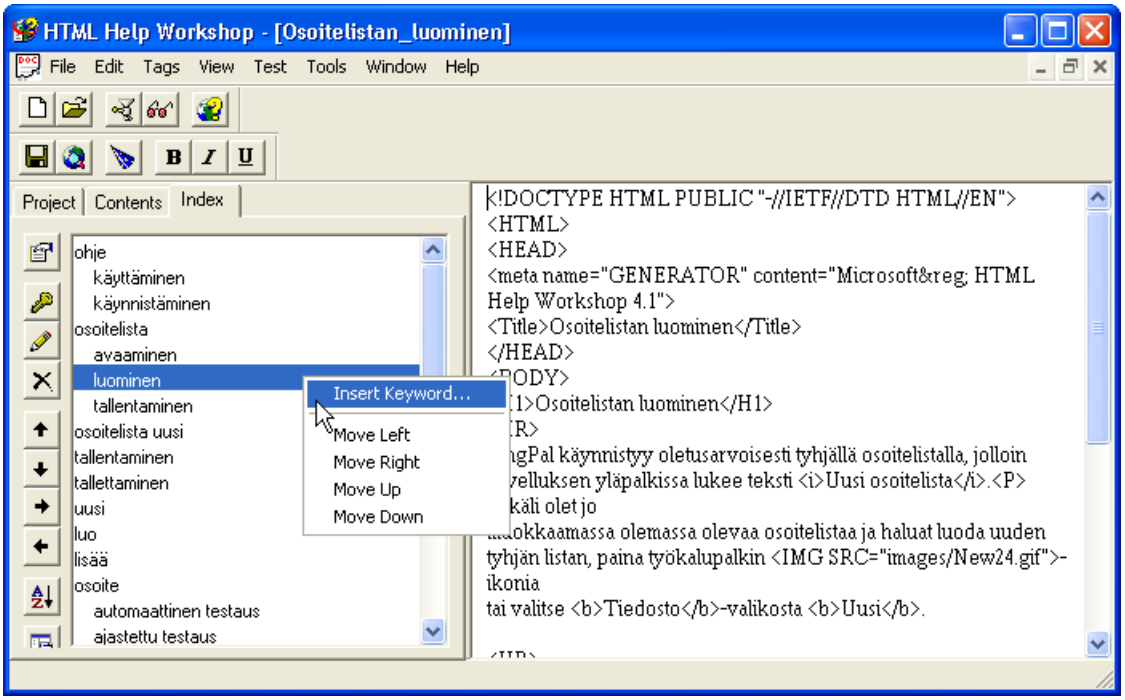
Kuva 10: Opastusjärjestelmä sisällysluettelo valmiina.

tettävä nimi. Koska sisällysluettelosta voidaan tehdä hierarkkinen, jaetaan opastussivut edellä mainitusti eri otsikoiden alle. Valmis sisällysluettelo on esitetty kuvassa 10.

Sisällysluettelon luomisen jälkeen on hakemiston vuoro. Koska hakemiston on todettu olevan sisällysluetteloakin suositumpi hakutapa käyttäjien keskuudessa (Purchase & Worrill, 2002), tulee hakusanojen laatimiseen kiinnittää erityistä huomiota. Suorien hakusanojen lisäksi hakemistoon lisätään myös mahdollisuuksien mukaan hakusanojen synonyymejä. Hakemistoon lisättävät sanat linkitetään niihin liittyviin opastussivuihin. Hakusanan lisääminen opastusjärjestelmän hakemistoon on esitetty kuvassa 11.

Kun yksittäiset ohjesivut, sisällysluettelo sekä hakemisto on luotu, voidaan opastusjärjestelmän lähdetiedostot kääntää yhdeksi opastusjärjestelmätiedostoksi. HTML Help Workshop osaa itse tehdä kokotekstihakutietokannan kääntämisen yhteydessä, joten käyttäjän tehtäväksi jää tämän jälkeen enää opastusjärjestelmän testaaminen ja linkkien toimivuuden sekä tekstin oikeinkirjoituksen tarkistaminen. Käännetty HTML Help -opastusjärjestelmä tekstinhakupaneeli aktiivisena on esitetty kuvassa 12.

Perinpohjaisen testauksen jälkeen valmis opastusjärjestelmä voidaan linkittää sovellukseen. HTML Help -pohjaisen opastusjärjestelmän linkittäminen Java-kieliseen



Kuva 11: Hakemiston luominen opastusjärjestelmään.

sovellukseen onnistuu ainakin luomalla uusi prosessi, joka kutsuu HTML Help -suoritustiedostoa (Hh.exe) parametrinaan käynnistettävän opastusjärjestelmän nimi sijaintipolkuineen. Kyseisen toiminnallisuuden toteuttava Java-metodi on listattu kuvassa 13. Menetelmän haittana on se, että joka kutsulla luodaan uusi prosessi opastusjärjestelmää varten, ts. vaikka opastusjärjestelmälle voidaankin antaa parametrina näytettävän opastussivun nimi (alias), ei kontekstisidonnaisen opasteen rakentaminen onnistu, koska jokaista apupyyntöä varten avautuu uusi opastusjärjestelmä. Visual Basic ja Visual C++ -kielet voivat hyödyntää suoraan HTML Help API:a, jolloin ohjeen linkittäminen sovellukseen käy yksinkertaisesti ja kontekstisidonnaisuuden toteuttaminen on mahdollista. PingPal-sovelluksen osalta tyydyttin opastusjärjestelmän pääsivun linkittämiseen Ohje-valikkoon. Seuraavassa kuvataan vastaavan kaltaisen opastusjärjestelmän rakentaminen JavaHelp-teknologiaa käyttäen, jolloin myös kontekstisidonnaisuuden lisääminen opastusjärjestelmään on toteutettavissa.

Kuva 12: Valmiin HTML Help -opastusjärjestelmän kokotekstihaku.

```
private void showHtmlHelp() {
   try {
       Runtime rt = Runtime.getRuntime();
       String[] callAndArgs = { ”Hh.exe”, ”PingPal.chm” };
       Process child = rt.exec(callAndArgs);
   }
   catch(Exception ex) {
            ex.printStackTrace();
   }
}
```

Kuva 13: HTML Help -opastusjärjestelmän käynnistävä Java-kielinen metodi.

4.4 JavaHelp-opastusjärjestelmän rakentaminen

JavaHelp-opastusjärjestelmän rakentamista varten tarvitaan Java 2 Software Development Kit (J2SDK) sekä JavaHelp Technology -paketti, joka on saatavilla zip-tiedostona

Sun Microsystemsin Internet-sivuilta. Kirjoitushetkellä uusin saatavilla oleva versio on 2.0_01. Edellä mainittujen lisäksi tarvitaan jokin tekstieditori tai kehitysympäristö JavaHelp-projektissa tarvittavien Java-, HTML-, sekä XML-tiedostojen luomista ja muokkaamista varten. JavaHelpin mukana ei tule mitään aiemmin kuvatun HTML Help Workshopin kaltaista integroitua kehitysympäristöä. Sen sijaan JavaHelp kehittäjien oletetaan käyttävän samaa kehitysympäristöä, mitä he käyttävät muussakin Java-kehitystyössä. Seuraavassa on kuvattu JavaHelp-projektiin kuuluvat osat ja opastusjärjestelmän rakentaminen käytännössä.

JavaHelp-projektin komponentit

JavaHelp-projektin ydintä kuvaa englanninkielinen termi *HelpSet*, joka tarkoittaa opastusjärjestelmän kuuluvia olennaisia tiedostoja. HelpSet-tiedostoihin kuuluvat (Lewis, 2000)

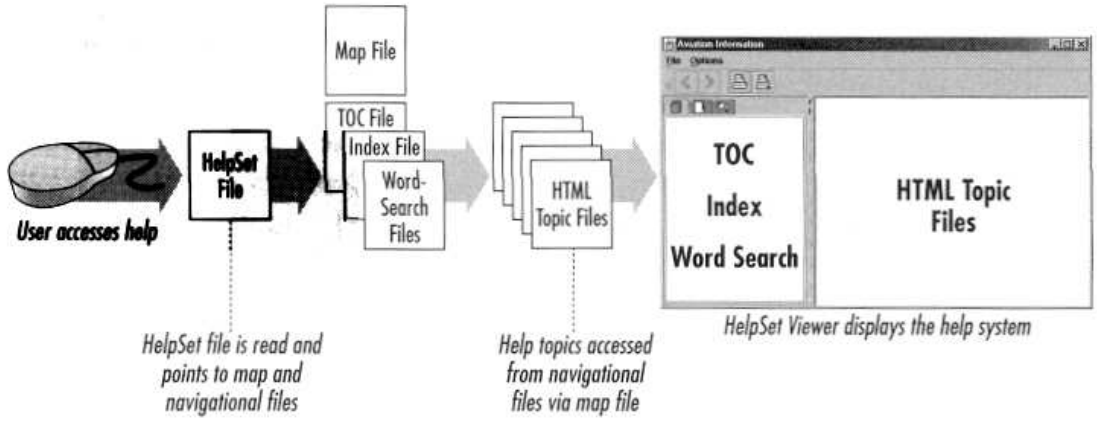
- HelpSet data-tiedostot (data files)
- Navigointitiedostot (navigation files)
- Opastusaihetiedostot (topic files).

HelpSet *data-tiedostot* sisältävät yleistä tietoa HelpSetin rakenteesta ja sisällöstä. Ne ovat XML-muotoisia tekstitiedostoja. HelpSet data-tiedostoja on kahdentyyppisiä: .hs-päätteinen HelpSetin ”päätiedosto”, sekä .jhm-päätteinen HelpSetin map-tiedosto. Päätiedosto määrittelee käytettävät navigointikomponentit sekä tiedostot, jotka konfiguroivat em. komponentit. HelpSetin päätiedostossa määritellään myös opastusjärjestelmän käyttämä ns. map-tiedosto. Map-tiedostossa on liitetty tunniste (id) jokaiseen opastusaiheeseen (topic) tai tarkemmin ilmaistuna opastusaihetiedoston URL-osoitteeseen. HelpSetin tiedostot ja navigointitiedosto käyttävät aina map-tiedostossa määriteltyjä tunnisteita viittaamaan opastuaihesivuihin, eivätkä ne koskaan viittaa suoraan sivujen URL-osoitteisiin.

HelpSetin *navigointitiedostot* määrittelevät HelpSetin sisällysluettelon (TOC), hakemiston (Index), sekä sanahakuhakemiston (Word-search index). Sisällysluettelo ja hakemisto ovat molemmat rakenteeltaan samanlaisia ja ne ovat XML-muotoisia. Sanahakuhakemisto on hiukan monimutkaisempi: se viittaa hakemistoon, joka sisältää useita

tiedostoja. Nämä tiedostot luodaan käyttäen apuna JavaHelpin mukana tulevaa indeksointityökalua (jhindexer). Kyseinen työkalu muodostaa koko tekstin kattavan hakutietokannan HelpSetille. Tätä tietokantaa käytetään tekstihakuihin, kun käyttäjä käyttää navigointipaneelin tekstihakua (Lewis, 2000).

Itse *opastustekstisivut* (topic files) ovat HTML-tiedostoja ja ne ovat myös osa HelpSetiä. Yhteen HTML-tiedostoon voi kirjoittaa ainoastaan yhdestä aiheesta, koska edellä kuvattu map-tiedosto määrittelee yhden tunnisteen vastaamaan yhtä HTML-tiedostoa. Opastustekstisivut voivat sisältää kuvia ja linkkejä aivan kuten tavallisetkin HTML-sivut. Lisäksi niissä voidaan käyttää ponnahdusikkunoita (pop-up window), sekundaarisia ikkunoita sekä multimedia-leikkeitä (Lewis, 2000). JavaHelp-opastusjärjestelmän HelpSet-tiedostojen keskinäiset suhteet on esitelty kuvassa 14.



Kuva 14: HelpSet-tiedostojen yhteistoiminta (Lewis, 2000, s. 7).

Opastusjärjestelmän rakentaminen

JavaHelp-opastusjärjestelmän rakentamisessa käytettiin hyödyksi kohdassa 4.3 HTML Help -opastusjärjestelmää varten laadittuja (HTML-muotoisia) opastussivuja. Koska JavaHelp-teknologian mukana ei toimiteta mitään erityistä työkalua opastusjärjestelmän kokoamiseksi, täytyy opastusjärjestelmän HelpSetiin kuuluvat tiedostot luoda käsin. Työ aloitettiin luomalla HelpSetin päätiedosto helpset.hs. HelpSet-tiedostossa määritellään opastusjärjestelmän muiden tieto- sekä navigointitiedostojen (sisällysluettelo ja hakemisto) nimet. XML-muotoinen tiedosto on esitetty kokonaisuudessaan kuvassa 15.

```
<?xml version='1.0' encoding='ISO-8859-1' ?>
<!DOCTYPE helpset
  PUBLIC "-//Sun Microsystems Inc.//DTD JavaHelp HelpSet Version 2.0//EN"
         "http://java.sun.com/products/javahelp/helpset_2_0.dtd">

<helpset version="2.0">
     <title>PingPal-sovelluksen opastusjärjestelmä (JavaHelp)</title>
     <maps>
          <mapref location="Map.jhm"/>
          <homeID>etusivu</homeID>
     </maps>

     <!-- Määritellään sisällysluettelo -->
     <view>
         <name>TOC</name>
         <label>Sisällysluettelo</label>
         <type>javax.help.TOCView</type>
         <data>TOC.xml</data>
     </view>

     <!-- Määritellään hakemisto -->
     <view>
          <name>Index</name>
          <label>Hakemisto</label>
          <type>javax.help.IndexView</type>
          <data>Index.xml</data>
     </view>

     <!-- Määritellään sanahakutoiminto -->
     <view>
          <name>Search</name>
          <label>Sanahaku</label>
          <type>javax.help.SearchView</type>
          <data engine="com.sun.java.help.search.DefaultSearchEngine">
                JavaHelpSearch
          </data>
     </view>
</helpset>
```

Kuva 15: JavaHelp-opastusjärjestelmän HelpSet-tiedosto XML-muodossa.

Seuraavaksi luodaan Map-tiedosto, jossa määritellään jokaiselle opastusjärjestelmän (HTML-muotoiselle) aihetiedostolle nimi (alias), jolla tiedoston suhteelliseen polkuun viitataan. Näitä nimiä voidaan sitten käyttää varsinaisessa sovelluksessa viittaamaan tiettyihin ohjeen sivuihin. Näin voidaan toteuttaa kontekstisidonnainen opastusjärjestelmä. Map-tiedoston määritys XML-muodossa on esitetty kuvassa 16.

Map-tiedoston luonnin jälkeen voidaan siirtyä kirjoittamaan navigointitiedostoja si-

```
<?xml version='1.0' encoding='ISO-8859-1' ?>
<!DOCTYPE map
  PUBLIC "-//Sun Microsystems Inc.//DTD JavaHelp Map Version 2.0//EN"
         "http://java.sun.com/products/javahelp/map_2_0.dtd">

<map version="2.0">
<!-- Tähän määritellään kaikille opastusjärjestelmän html-sivuille
     yksinkertaisempi aliasnimi
  -->

<mapID target="etusivu" url="Ohjeet/Tervetuloa.htm" />
<mapID target="ohjeen_kayttaminen" url="Ohjeet/Ohjeen_kayttaminen.htm" />

<mapID target="osoitelistan_luominen" url="Ohjeet/Osoitelistan_luominen.htm" />
<mapID target="osoitelistan_avaaminen" url="Ohjeet/Osoitelistan_avaaminen.htm" />
<mapID target="osoitelistan_tallentaminen"
       url="Ohjeet/Osoitelistan_tallentaminen.htm"
/>

<mapID target="osoitteen_lisaaminen" url="Ohjeet/Osoitteen_lisaaminen.htm" />
<mapID target="osoitteen_poistaminen" url="Ohjeet/Osoitteen_poistaminen.htm" />
<mapID target="osoitteen_ajastus" url="Ohjeet/Osoitteen_ajastus.htm" />
<mapID target="osoitteiden_testaus" url="Ohjeet/Osoitteiden_testaus.htm" />

<mapID target="lokitiedoston_lukeminen" url="Ohjeet/Lokitiedoston_lukeminen.htm" />
<mapID target="lokitiedoston_tyhjentaminen"
       url="Ohjeet/Lokitiedoston_tyhjentaminen.htm"
 />

</map>
```

Kuva 16: Map-tiedosto (Map.jhm) XML-muodossa.

sällysluetteloa ja hakemistoa varten. Sisällysluettelo on tässä toteutettu tiedostoon TOC.xml. Tiedostossa määritellään sisällysluettelossa näkyvät opastussivut, niiden nimet, sekä hierarkkinen suhde muihin sivuihin. Sisällysluettelon määrittelyyn käytettävä tiedosto on esitetty kuvassa 17.

Hakemistoa varten luodaan tiedosto Index.xml, johon määritellään hakusanat ja niitä vastaavat opastussivut. Hakemistoon voidaan määritellä useamman tasoisia hakusanoja käyttämällä hierarkioita. On huomioitava, ettei JavaHelp-opastusjärjestelmän käyttämä selain osaa järjestää hakusanoja aakkosjärjestykseen vaan opastusjärjestelmän rakentajan on huolehdittava itse siitä. Hakemiston määrittelevä XML-tiedosto on esitetty kokonaisuudessaan kuvassa 18.

Koska itse opastussivut luotiin jo HTML Help -opastusjärjestelmän rakentamisen yh-

```
<?xml version='1.0' encoding='ISO-8859-1' ?>
<!DOCTYPE toc
  PUBLIC "-//Sun Microsystems Inc.//DTD JavaHelp TOC Version 2.0//EN"
         "http://java.sun.com/products/javahelp/toc_2_0.dtd">

<toc version="2.0">
<!-- Tässä määritellään sisällysluettelon rivit siinä järjestyksessä
     kuin niiden halutaan esiintyvän opastusjärjestelmässä.
-->

<tocitem target="etusivu" text="Tervetuloa"/>
<tocitem target="ohjeen_kayttaminen" text="Ohjeen käyttäminen"/>

<tocitem text="Osoitelistat">
   <tocitem target="osoitelistan_luominen" text="Osoitelistan luominen"/>
   <tocitem target="osoitelistan_tallentaminen" text="Osoitelistan tallentaminen"/>
   <tocitem target="osoitelistan_avaaminen" text="Tallennetun osoitelistan avaaminen"/>
</tocitem>

<tocitem text="Osoitteiden käsittely">
   <tocitem target="osoitteen_lisaaminen" text="Osoitteen lisääminen"/>
   <tocitem target="osoitteen_poistaminen" text="Osoitteen poistaminen"/>
   <tocitem target="osoitteen_ajastus" text="Osoitteen ajastus"/>
   <tocitem target="osoitteiden_testaus" text="Osoitteen testaus"/>
</tocitem>

<tocitem text="Loki">
   <tocitem target="lokitiedoston_lukeminen" text="Lokitiedoston lukeminen"/>
   <tocitem target="lokitiedoston_tyhjentaminen" text="Lokitiedoston tyhjentäminen"/>
</tocitem>
</toc>
```

Kuva 17: Sisällysluettelo tiedosto (TOC.xml).

teydessä ei niitä aleta tässä kirjoittamaan uudelleen, vaan kohdassa 4.3 luodut opastussivut kopioidaan sellaisenaan opastusjärjestelmäprojektin alihakemistoon *Ohjeet*. Tämän jälkeen voidaan luoda tekstihaun käyttämä hakutietokanta JavaHelpin mukana tullutta *jhindexer*-indeksointityökalua käyttäen. Työkalu käynnistetään komentokehotteen avulla ja sille annetaan parametrina indeksoitavien HTML-sivujen hakemisto, tässä tapauksessa: `c:\javahelp\bin\jhindexer Ohjeet`. Työkalu käy läpi kaikki hakemiston sisältämät HTML-tiedostot (ja rekursiivisesti myös kaikki sen alihakemistot) ja luo opastehakemistoon *JavaHelpSearch*-nimisen alihakemiston, joka sisältää tekstihakuun tarvittavan tietokannan. Esimerkissä rakennetun JavaHelp-opastusjärjestelmän koko hakemistorakenne on esitetty liitteessä 1.

Kun sanahakutoimintokin on luotu, voidaan opastusjärjestelmän toimintaa kokeilla

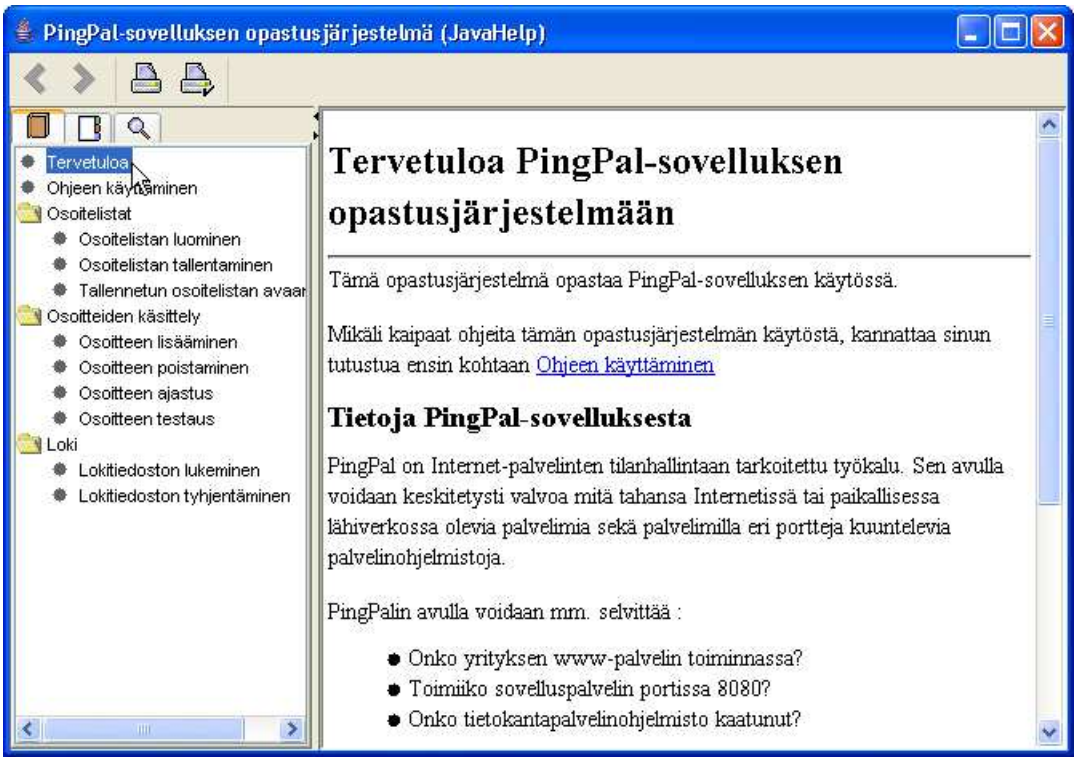
```
<?xml version='1.0' encoding='ISO-8859-1' ?>
<!DOCTYPE index
  PUBLIC "-//Sun Microsystems Inc.
            //DTD JavaHelp Index Version 2.0//EN"
          "http://java.sun.com/products/javahelp/index_2_0.dtd">

<index version="2.0">
  <indexitem text="ohje">
      <indexitem target="ohjeen_kayttaminen" text="käyttäminen"/>
      <indexitem target="ohjeen_kayttaminen" text="käynnistäminen"/>
  </indexitem>

  <indexitem text="osoitelista">
      <indexitem target="osoitelistan_avaaminen" text="avaaminen"/>
      <indexitem target="osoitelistan_luominen" text="luominen"/>
      <indexitem target="osoitelistan_tallentaminen" text="tallentaminen"/>
  </indexitem>

  <indexitem text="osoite">
     <indexitem target="osoitteiden_testaus" text="automaattinen testaus"/>
     <indexitem target="osoitteiden_testaus" text="ajastettu testaus"/>
     <indexitem target="osoitteen_lisaaminen" text="lisääminen"/>
     <indexitem target="osoitteen_ajastus" text="ajastus"/>
     <indexitem target="osoitteiden_testaus" text="testaus"/>
     <indexitem target="osoitteen_poistaminen" text="poistaminen"/>
  </indexitem>

  <indexitem target="tervetuloa" text="tietoja ohjelmasta"/>
  <indexitem target="tervetuloa" text="yleiskuvaus"/>
</index>
```

Kuva 18: Hakemiston määrittelevä XML-tiedosto Index.xml

JavaHelp-paketin mukana tulevalla *hsviewer*-työkalulla. Kuvassa 19 on esitetty edellä luotu JavaHelp-opastusjärjestelmä PingPal-sovellukselle.

Kun opastusjärjestelmä on testattu ja kaikki linkit, otsikot ja hakemiston sanat todettu toimiviksi, voidaan aloittaa opastusjärjestelmän linkittäminen itse sovellukseen. Sovellukseen lisättiin seuraavat linkit opastusjärjestelmään:

- *Ohje*-valikon valinnalle *Ohjeen aiheet (JavaHelp)* lisättiin linkki opastusjärjestelmän etusivulle.
- Työkaluvalikon *Ohje*-painikkeeseen lisättiin näyttötasoinen kontekstisidonnainen ohje.
- *F1*-painikkeeseen linkitettiin myös näyttötasoinen kontekstisidonnainen ohje.



Kuva 19: JavaHelp-teknologialla toteutettu opastusjärjestelmä PingPal-sovellukselle.

Näyttötasoinen kontekstisidonnainen ohje tarkoittaa PingPal-sovelluksen kohdalla sitä, että jos käyttäjä painaa *IP-osoite*-välilehdellä ollessaan *Ohje*- tai *F1*-painiketta, opastusjärjestelmä avaa tuota näyttöä varten kirjoitetun ohjeen. Vastaavasti jos käyttäjä on *Loki*-välilehdellä, avautuu ohjeeseen lokitiedostoon liittyvä sivu. *Ohje*-valikon valinta *Ohjeen aiheet (JavaHelp)* avaa aina ohjeen etusivun näkyville. Ohjeen linkittäminen sovellukseen tehtiin kuvassa 20 esitettyä koodia käyttäen.

Tuotantoonsiirtoa varten kannattaa valmiin opastusjärjestelmän tiedostot paketoida yhteen jar-tiedostoon. Paketointi onnistuu Javan mukana tulevalla *jar*-työkalulla siirtymällä ensin opastusjärjestelmän sijaintihakemistoon ja antamalla komentokehotteessa komento: `jar -cf HelpSet.jar *`. Komento luo tiedoston *HelpSet.jar*, joka sisältää kaikki hakemistossa olevat tiedostot alihakemistoineen.

Sulautetun JavaHelp-opastusjärjestelmän rakentaminen

Luvussa 2 todettiin, että opastusjärjestelmän käynnistäminen ja tilanteeseen sopivan ohjeen löytäminen voi olla monelle vasta-alkajalle ylivoimainen tehtävä. Edistyneet

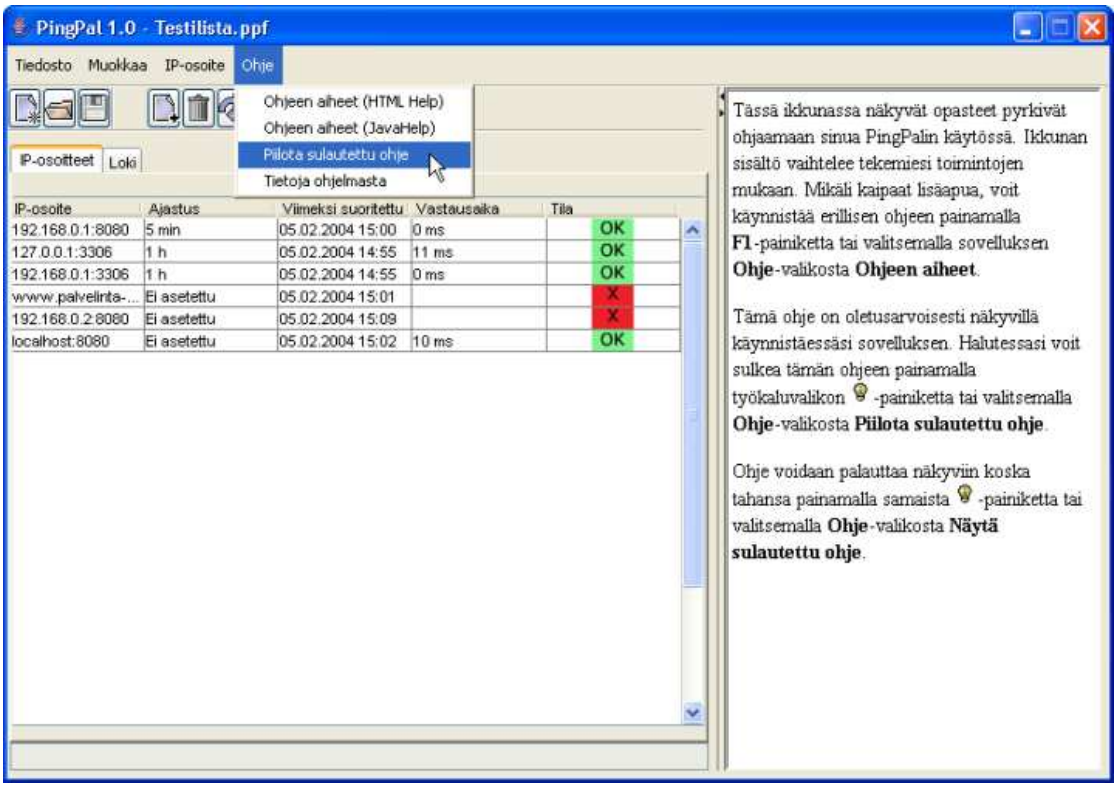
```
import javax.help.*;
...

HelpSet helpSet = null;
HelpBroker helpBroker = null;

// HelpSet.hs-tiedosto sijaitsee samassa hakemistossa
// kuin itse sovelluskin jar-tiedostoon pakattuna
String helpSetAddress = ”jar:file:HelpSet.jar!/HelpSet.hs”;
...

try {
    // luodaan String-osoitteesta URL-muotoinen osoite
    URL helpSetURL = new URL(helpSetAddress);
    // yritetään alustaa helpset
    helpSet = new HelpSet(null, helpSetURL);
}
catch(Exception e) {
    System.out.println(”HelpSet not found”);
    e.printStackTrace();
}

// alustetaan helpBroker helpSetin avulla
helpBroker = helpSet.createHelpBroker();

// linkitetään opasteen etusivu F1-painikkeeseen
helpBroker.enableHelpKey(getRootPane(), ”etusivu”, helpSet);

// luodaan tapahtumankuuntelija luokan CSH staattisella metodilla
// tapahtumankuuntelija hoitaa opastusjärjestelmän käynnistämisen
ActionListener helper = new CSH.DisplayHelpFromSource(helpBroker);

// lisätään edellä luotu tapahtumankäsittelijä Ohje-valikon JavaHelp-valintaan...
jMenuItem11.addActionListener(helper);

// sekä työkaluvalikon Ohje-painikkeeseen
jButton10.addActionListener(helper);
```

Kuva 20: JavaHelp-opastusjärjestelmän linkittäminen sovellukseen. Koodi linkittää opastusjärjestelmän etusivun avautumaan *F1*-painikkeesta, työkaluvalikon *Ohje*-painikkeesta (jButton10) sekä *Ohje*-valikon *Ohjeen aiheet (JavaHelp)* -valinnasta (jMenuItem11).

käyttäjät taas saattavat tuntea epäonnistumisen tunteita, mikäli he joutuvat turvautumaan opasteeseen. Sulautettu opastusjärjestelmä vastaa näihin haasteisiin siten, että se on integroitu sovelluksen käyttöliittymään ja on siis oletusarvoisesti näkyvillä. Sulautettu opastusjärjestelmä on myös kontekstisidonnainen: se tarjoaa apua niistä aiheista,

joista käyttäjä kussakin tilanteessa tarvitsee apua.

Edellä kuvattua JavaHelp-opastusjärjestelmää päätettiin muokata siten, että siitä kehitettiin PingPal-sovelluksen käyttöliittymään sulautettu opastusjärjestelmä. Opastepaneeli päätettiin kokeilun tuloksena sijoittaa itse sovelluksen oikealle sivulle. Navigointipaneeli sisällysluetteloineen on oletusarvoisesti piilotettuna, mutta sen saa näkyviin yhdellä hiiren napsautuksella. Samoin työkaluvalikkoon lisättiin painike ja sovelluksen *Ohje*-valikkoon valinta sulautetun opastusjärjestelmän piilottamista ja esiin tuomista varten. Sulautettu opastusjärjestelmä on oletusarvoisesti näkyvillä sovelluksen käynnistyttyä. Edellä kuvattujen parannusten jälkeen aikaansaatu PingPal-sovellus on esitetty kuvassa 21.



Kuva 21: PingPal-sovellus sulautetulla opastusjärjestelmällä varustettuna.

Sulautetun opastusjärjestelmän suunnittelukriteereiksi määriteltiin se, että opastustekstien pitäisi olla mielellään olla näkyvillä kokonaisuudessaan ruudulla – ilman, että käyttäjä joutuu vierittämään ikkunaa. Toinen kriteeri oli se, ettei sulautetun ohjeen toteuttaminen saa pienentää itse sovellukselle varattua näyttöalaa.

Sulautetun opastusjärjestelmän ja sovelluksen interaktion toteutuksessa sovellettiin

osin kenttätasoista kontekstisidonnaisuutta. Käytännössä tämä ilmenee esim. käyttäjän napsauttaessa hiirellä osoitelistaa. Tällöin näytön oikeassa reunassa olevaan opasteeseen ladataan ohje, jossa neuvotaan osoitelistan muokkaamiseen liittyvissä toiminnoissa. Samoin käyttäjän vaihtaessa *Loki*-välilehdelle, opaste reagoi tapahtumaan näyttämällä lokitiedostoon liittyvää opastetta. Kuvassa 22 on esitetty esimerkkikoodi, joka mahdollistaa PingPal-sovelluksen ohjeen kontekstisidonnaisuuden. Käyttäjän pyytäessä apua ohjeessa näytetään etusivun sijasta osoitteen lisäämisestä kertovaa opastussivua. Lainausmerkeissä annettu merkkijono ”osoitteen_lisaaminen” on alias opastussivuun, jotka määriteltiin kuvassa 16 esitetyssä Map-tiedostossa. Tämän avulla opastusjärjestelmä osaa näyttää oikean opastussivun, joka on *Ohjeet/Osoitteen_lisaaminen.htm*.

```
// lisätään näytön välilehtikomponenttiin (jTabbedPane1)
// kuuntelija, joka reagoi käyttäjän vaihtaessa välilehteä

jTabbedPane1.addChangeListener(new ChangeListener() {
      // Tätä metodia kutsutaan kun käyttäjä vaihtaa välilehteä
      public void stateChanged(ChangeEvent evt) {

          if (sel==0) { // käyttäjä vaihtoi IP-osoitteet -välilehdelle

             // asettaa F1-painikkeesta avautuvan opastesivun
             CSH.setHelpIDString(MainH.this.getRootPane(),"osoitteen_lisaaminen");

             // asettaa työkaluvalikon Ohje-painikkeesta avautuvan opastesivun
             CSH.setHelpIDString(jButton10,"osoitteen_lisaaminen");

             // päivittää sulautetussa ohjeessa näytettävän opastesivun
             hp.viewer.setCurrentID("osoitteen_lisaaminen");
          }
          else if (sel==1) { // käyttäjä vaihtoi Loki-välilehdelle
             // asetetaan avautuvat opastesivut kuten edellä, mutta viittaamaan
             // eri opasteisiin
             CSH.setHelpIDString(MainH.this.getRootPane(),"lokitiedoston_lukeminen");
             CSH.setHelpIDString(jButton10,"lokitiedoston_lukeminen");
             hp.viewer.setCurrentID("lokitiedoston_lukeminen");
          }
      }
});
```

Kuva 22: Ohjeen linkittäminen sovellukseen mahdollistaen kontekstisidonnaisuuden.

4.5 Arviointi

Edellä kuvattiin käytännön tasolla kaksi erilaista opastusjärjestelmien rakentamiseen saatavilla olevaa tekniikkaa: HTML Help ja JavaHelp. Tarkoitusta varten on saatavilla runsaasti muitakin samankaltaisia työkaluja, mutta edellä mainittuihin päädyttiin, koska molemmat teknologiat ovat ilmaisia ja yleisesti käytettyjä. HTML Help edustaa kirjoitushetkellä Windows-ympäristön opastusjärjestelmän standardia rakentamistekniikkaa. JavaHelp puolestaan tarjoaa alustariippumattoman tavan rakentaa opastusjärjestelmä. Java-pohjaisista tekniikoista JavaHelp valittiin Oracle Help for Javan sijasta sen suuremman käyttäjämäärän (WinWriters, 2003) ja kenties vankemman aseman vuoksi.

Käytännön rakentamisprosessin aikana huomattiin mm. seuraavia seikkoja. Sekä JavaHelpillä että HTML Helpillä toteutetut opastusjärjestelmät olivat ulkoasultaan hyvin samankaltaisia – molemmat tarjoavat kirjamaisen perusvaikutelman. Toimintojen osalta kumpaankin onnistuttiin toteuttamaan sisällysluettelo, hakemisto sekä sanahaku. Molemmissa opastusjärjestelmissä navigointipaneeli (sisällysluettelo) päivittyy sisältöpaneelin muutoksista, eli käyttäjä näkee aina sisällysluettelosta mitä ohjeen kohtaa hän lukee, riippumatta pääsikö hän kyseiselle sivulle sisällysluettelosta vaiko opastussivun sisäisestä linkistä käsin. Tämä seikka osaltaan ehkäisee ”hyperavaruuteen eksymisen” vaaraa, joka tuli esille tutkielman teoriaosuudessa hypertekstijärjestelmiin liittyvänä ongelmana.

Eroavaisuuksista mainittakoon mm. se, että JavaHelp-opastusjärjestelmä käynnistyy oletusarvoisesti sisällysluettelon hierarkkiset opastusaiheet avattuina kun taas HTML Help -opastusjärjestelmällä ne ovat suljettuina. Hakemistossa useampitasoisia hakusanoja voi JavaHelpissä sulkea ja avata, HTML Helpissä ne ovat aina kaikki näkyvillä. Sanahaun osalta JavaHelp näyttää haetun sanan esiintymien lukumäärän kutakin löydettyä opastussivua kohti, järjestää sivut sanojen esiintymistiheyden mukaan laskevaan järjestykseen ja lataa sisältöpaneeliin automaattisesti eniten haetun sanan esiintymiä sisältävän opasteen. HTML Help -opastusjärjestelmä näyttää vain kaikkien niiden opastussivujen nimet, joilta haettu sana löytyy. JavaHelpin versio 2.0 mahdollistaa myös sanaston, mikä puuttuu HTML Helpistä.

JavaHelpin osalta valitettavaa on sisäänrakennetun selaimen tuki vain HTML 3.2 -standardille, minkä takia JavaHelpillä toteutettuun opastusjärjestelmään ei kannata linkittää valmiita, uusille HTML 4.0 versiota tukeville selaimille tarkoitettuja WWW-

sivuja, koska ne eivät toimi JavaHelp-selaimessa kunnolla. Sama puute pätee tietyin varauksin myös HTML Help -opastusjärjestelmiin – kaikki linkitetyt sivut eivät välttämättä toimi kunnolla, koska HTML Help -opastusjärjestelmä käyttää käyttäjän koneelle asennettua Internet Explorer -selainta sivujen näyttämiseen ja tuettu HTML-kielen versio riippuu asennetun selaimen versiosta.

HTML Helpin puutteeksi on laskettava myös se, ettei kontekstisidonnaisen opasteen toteuttaminen onnistunut Java-kielisen sovelluksen kanssa. Ongelmaan on luultavasti olemassa ratkaisu, mutta se tuskin on täysin Java-pohjainen. Sama ongelma pätee myös päinvastaiseen tilanteeseen: kontekstisidonnaisen JavaHelp-opastusjärjestelmän linkittäminen C++ -kielellä tehtyyn sovellukseen ei varmasti ole triviaali asia. Näin ollen opastusjärjestelmän toteutustekniikkaa valittaessa eräs tärkeimmistä valintakriteereistä on itse sovelluksen toteutuskieli.

Rakennusprosessi oli molemmissa tapauksissa melko suoraviivainen. HTML Helpin mukana tuleva HTML Help Workshop helpotti työtä jonkin verran – opastusjärjestelmän toteuttajan työksi jää lähinnä HTML-muotoilu ja tietenkin itse opastustekstin kirjoittaminen. JavaHelpin käyttämät XML-muotoiset määrittelytiedostot vaativat hiukan enemmän teknistä asiantuntemusta.

Molemmat tekniikat ovat pohjimmiltaan HTML:ään perustuvia, eli niiden tulostamat opasteet on joko kirjoitettu tai ainakin muutettu HTML-muotoon ennen niiden näyttämistä. Eduistaan ja käytön yleisyydestä huolimatta myös HTML-kieleen liittyy omat ongelmansa. Niistä suurin lienee se, että HTML-dokumentin sisältö ja (ainakin suuntaa antava) ulkoasu on määritelty samassa tiedostossa. Näin opastustekstin kirjoittaja joutuu tekstisisällön lisäksi ottamaan kantaa myös tekstin esitysmuotoon.

Microsoft pyrkii ratkaisemaan edellä kuvatun HTML-pohjaisiin opastussivuihin liittyvän ongelman tulevassa Windowsin versiossa, joka tunnetaan koodinimellä ”Longhorn” (Ellison, 2004). ”Longhorniin” sisältyvä opastusjärjestelmätekniikka mahdollistaa mm. opasteiden sisällön ja ulkoasun erottamisen XML:ään pohjautuvalla MAML-kielellä (Microsoft Assistance Markup Language). Uuden Windowsin ja sen mukana kokonaan uudelleen muokatun opastusjärjestelmän tuotantoon saaminen voi kuitenkin kestää vielä pari vuotta.

Sulautettua opastusjärjestelmää rakennettaessa tulivat ilmi JavaHelp-tekniikan mahdollisuudet opastusjärjestelmän kustomoinnissa – JavaHelp API tarjoaa erinomaiset

mahdollisuudet oletustoteutuksena annetun opastusjärjestelmän muokkaamista, sulautetun opastuksen toteuttamista tai kokonaan oman opastusjärjestelmän tekemistä varten. Ajatus aina näkyvillä (joskin helposti piilotettavissa) olevasta ohjeesta tuntuu sikälikin mieleiseltä, ettei käyttäjän tarvitse myöntää edes itselleen tarvitsevansa opastusta, puhumattakaan siitä että keskeyttäisi parhaillaan suorittamansa tehtävän ja alkaisi etsimään omatoimisesti tehtävään liittyvää opastusta. Toteutuksen kannalta ongelmallista sen sijaan on, etenkin laajemmissa sovelluksissa, käyttäjän kulloisenkin päämäärän identifioiminen – ts. sen ennustaminen, mitä käyttäjä yrittää sovelluksella parhaillaan aikaansaada.

Rakennettujen opastusjärjestelmien käytettävyyden mittaaminen olisi vaatinut empiirisen tutkimuksen järjestämistä, johon ei tämän tutkielman puitteissa valitettavasti ollut mahdollisuutta. Tällainen tutkimus vaatisi myös hiukan laajemman ja monimutkaisemman testisovelluksen toteuttamisen.

5 Yhteenveto

Tässä tutkielmassa käsiteltiin opastusjärjestelmiä, niistä tehtyjä tutkimuksia ja pohdittiin niiden käytettävyyttä loppukäyttäjän näkökulmasta sekä rakentamista kehittäjän näkökulmasta.

Luvussa 2 tutustuttiin opastusjärjestelmien kehitykseen, niiden eri lajeihin sekä ominaisuuksiin. Lisäksi luotiin katsaus aiheesta kirjoitettuun kirjallisuuteen sekä tutkimustuloksiin. Luvussa 3 identifioitiin opastusjärjestelmiin liittyvät yleisimmät ongelmat, käsiteltiin opastusjärjestelmien suunnittelua sekä käyttöliittymän että sisällön osalta, ja käytiin opastusjärjestelmän rakentamisprosessi läpi kirjallisuuden pohjalta. Luvussa 4 sama tehtiin käytännössä esittelemällä tutkielmaa varten tehty esimerkkisovellus sekä kuvaamalla kolmen eri opastusjärjestelmän rakentaminen kyseiselle sovellukselle kahta eri toteutusvälinettä käyttäen. Samassa luvussa luotiin myös katsaus markkinoilla oleviin tarkoitusta varten kehitettyihin välineisiin ja tekniikoihin.

Opastusjärjestelmistä tehtyihin tutkimuksiin perehdyttäessä kävi ilmi, että nykyisin käytössä olevat opastusjärjestelmät eivät tyydytä käyttäjiä. Syitä opastusjärjestelmien puutteisiin etsittiin mm. ohjelmistojen kehittäjien asenteista ja arvostuksen puutteesta opastusjärjestelmiä kohtaan. Huomion kiinnitti myös se seikka, että nykyisistä opastusjärjestelmistä valtaosa on vielä pitkälti kirjaparadigmaan perustuvia. Tutkimusten mukaan kirjamaisista ohjeista löytyvät toiminnot ovat juuri niitä toimintoja, joita käyttäjät eniten arvostavat. Toisaalta voidaan olettaa, että käyttäjät arvostavat ehkä juuri niitä toimintoja, jotka ovat heille ennestään tuttuja. Myös opastusjärjestelmien kehittäjille kirjamaisen ohjemuodon noudattaminen on helppoa – kaikkea ei tällöin tarvitse keksiä itse alusta alkaen. Tutkielman kokeellisessa osuudessa havaittiin, että nykyiset opastusjärjestelmien rakentamiseen käytetyt tekniikat, kuten HTML Help ja JavaHelp, keskittyvät nekin kirjamuotoisten opasteiden tuottamiseen. Näistä seikoista seuraa se vaara, että kirjaparadigmaan perustuva opastusjärjestelmä vakiintuneine toimintoineen voi saavuttaa eräänlaisen standardin aseman, vaikkei sen käytettävyys sitä välttämättä edellyttäisikään.

Toisaalta sulautettujen opastusjärjestelmien määrän lisääntyminen antaa toivoa paremmasta – ehkäpä meneillään on paradigman muutos.

Viitteet

Abdullahi, U.G., Alty, J.L. (1998) How Useful is On-line Help? An Observational Study, *Proceedings of the Australasian Conference on Computer Human Interaction*, IEEE Computer Society, Washington, DC, USA, 94–101.

Beyer, H., Holtzblatt, K. (1998) *Contextual Design*, Morgan Kaufmann Publishers, Los Altos, California, USA.

Borenstein, N.S. (1985) *The Design and Evaluation of On-line Help Systems*. Unpublished doctorial dissertation, Carnegie-Mellon University, Pittsburgh, USA.

Card, S., Moran, T., Newell, A. (1983) *The Psychology of Human-Computer Interaction*, Erlbaum, Hillsdale, New Jersey, USA.

Cooper, A. (1995) *About Face. The Essentials of User Interface Design*, Programmers Press, Foster City, California, USA.

Cooper, A., Reimann, R. (2003) *About Face 2.0: The Essentials of Interaction Design*, Wiley Publishing, Inc., Indianapolis, Indiana, USA.

Corrigan, D., Kennard, R. (1997) Computer-Mediated Learning Systems: A New Perspective. *Computing & Control Engineering Journal* **8**(3), 100–106.

DeLoach, S. (2001) An Overview of HTML-based Help. *Proceedings of the 48th Annual Conference of the Society for Technical Communication*. Society for Technical Communication, Chicago, USA, 537–542 (Saatavana myös: `http://www.stc.org/confproceed/2001/PDFs/STC48-000120.PDF`, (29.1.2004)).

Duffy, T., Palmer, J., Mehlenbacher, B. (1992) *On-line Help: Design and Evaluation*. Ablex Publishing Corporation, Norwood, New Jersey, USA.

eHelp Corporation (2004) *The Power of Browser-Based Help WebHelp - The standard in browser-based Help*. `http://www.ehelp.com/products/robohelp/whitepapers.asp` (29.1.2004).

Ellison, M. (2004) *Microsoft "Longhorn" Help Highlights*. `http://www.winwriters.com/articles/longhorn/` (26.1.2004) .

Fischer, G., Lemke, A., Schwab, T. (1985) Knowledge-based Help Systems. *Procee-*

dings of the SIGCHI conference on Human Factors in Computing Systems, ACM Press, New York, USA, 161–167.

Grayling, T. (1998) Fear and Loathing of the Help Menu: A Usability Test of Online Help. *Technical Communication* **5**(2), 168–179.

Heimbürger, A., Alkula, R., Kuhanen, T. (1990) *Hyperteksti ja hypermedia*. Valtion teknillinen tutkimuskeskus, Espoo.

Hefley, W. (1995) Helping users help themselves. *IEEE Software* **12**(2), 93–95.

Horton, W.K. (1990) *Designing and writing online documentation: help files to hypertext*, John Wiley & Sons, Inc., New York, USA.

James-Tanny, C. (2002) Choosing Help Authoring Tools: What Factors Affect Your Decision? *Boston Broadside Online* **60**(2). `http://www.stc-boston.org/broadside/11_2002/v60_no2_james-tannyprint.shtml` (20.2.2004).

John, B.E., Kieras, D.E. (1996) Using GOMS for user interface design and evaluation: Which technique? *ACM Transactions on Computer-Human Interaction*, **3**(4), 287–319.

Johnson-Eilola, J. (2001) Little Machines: Understanding Users Understanding Interfaces. *ACM Journal of Computer Documentation* **25**(4), 119–127.

Jorna, G.C. (1991) Image quality determines differences in reading performance and perceived image quality with CRT and hard-copy displays. *Proceedings of the 35th Annual Meeting of the Human Factors Society*, Human Factors Society, Santa Monica, California, USA, 1432–1436.

Kantner, L., Shroyer, R., Rosenbaum, S. (2002) Structured Heuristic Evaluation of Online Documentation. *Proceedings of International Conference on Professional Communication*, IEEE International, 331–342.

Kearsley, G. (1988) *Online help systems : design and implementation*. Ablex Publishing Corporation, Norwood, New Jersey, USA.

Leritz-Higgins, S. (2003) *Sun Officially Unveils JavaHelp 2.0 Beta*. `http://www.winwriters.com/articles/javahelp/` (18.11.2003).

Lewis, K. (2000) *Creating Effective JavaHelp*. O'Reilly & Associates, Inc., Sebasto-

pol, California, USA.

Mack, R.L., Nielsen, J. (1994) Usability Inspection Methods: Executive Summary teoksessa Baecker R, Grudin, J, Buxton, W., Greenberg S. (eds) (1995) *Readings in Human-Computer Interaction: Toward the Year 2000*, 2nd ed. Morgan Kaufmann Publishers, Inc., San Francisco, California, USA, 170–181.

Magers, C.S. (1983) An Experimental Evaluation of Online HELP for Non-Programmers. *ACM CHI'83 Proceedings*, ACM Press, New York, USA 277–281.

Mazur, B. (1995) Helping New Users Use Help. *Proceedings of International Conference on Professional Communication*, IEEE International, New York, USA, 89–92.

Microsoft (2004) *The HTML Help Components*. WWW-sivusto, `http://msdn.microsoft.com/library/default.asp?url=/library/en-us/htmlhelp/html/vsconwhatis.asp` (30.1.2004).

Mueller, P. (2003) Developing an Embedded Help Solution. *Technical Communication* **50**(1), 24–32.

Nielsen, J. (1990) *Hypertext and Hypermedia*. Academic Press, Inc., San Diego, California, USA.

Purchace, H., Worrill, J. (2002) An empirical study of on-line help design: features and principles. *International Journal of Human-Computer Studies* **56**(5), 539–567.

Ray, D. S., Ray, E. J. (2001) Embedded Help: Background and Applications for Technical Communicators. *Technical Communication* **48**(1), 105–115.

Randall, N., Pedersen, I. (1998) Who Exactly is Trying to Help Us? The Ethos of Help Systems in Popular Computer Applications. *Proceedings of the 16th annual international conference on Computer documentation*, ACM Press, New York, USA, 63–69.

Relles, N. (1979) *The Design and Implementation of User-Oriented Systems*. Unpublished doctorial dissertation, University of Wisconsin, Madison, USA.

Schneiderman, B. (1998) *Designing the User Interface (3rd ed)*. Addison-Wesley, Reading, Massachusetts, USA.

Shroyer, R. (2000) Actual Readers Versus Implied Readers: Role Conflicts in Office

97. *Technical Communication* **47**(2), 238–240.

Smart, K. L., DeTienne, K. B., Whiting, M. (1998) Customers' Use of Documentation: The Enduring Legacy of Print, *Proceedings of the 16th annual international conference on Computer documentation*, ACM Press, New York, USA, 23–28.

Turk, K.L., Nichols, M.C. (1996) Online Help Systems: Technological Evolution or Revolution? *Proceedings of the 14th annual international conference on Systems documentation*, ACM Press, New York, USA, 239–242.

WinWriters, Inc. (2003)*The 2003 WritersUA Skills and Technologies Survey*. `http://www.winwriters.com/surveys/skillstech03/skillstech03.htm` (5.5.2004).

Liite 1: JavaHelp-opastusjärjestelmän hakemistorakenne

```
F:\PROJECTS\PINGPALJAVAHELP
|   HelpSet.hs
|   Index.xml
|   Map.jhm
|   TOC.xml
|
+---JavaHelpSearch
|       DOCS
|       DOCS.TAB
|       OFFSETS
|       POSITIONS
|       SCHEMA
|       TMAP
|
\---Ohjeet
    |   Lokitiedoston_lukeminen.htm
    |   Lokitiedoston_tyhjentaminen.htm
    |   Ohjeen_kayttaminen.htm
    |   Osoitelistan_avaaminen.htm
    |   Osoitelistan_luominen.htm
    |   Osoitelistan_tallentaminen.htm
    |   Osoitteen_ajastus.htm
    |   Osoitteen_lisaaminen.htm
    |   Osoitteen_poistaminen.htm
    |   Osoitteiden_testaus.htm
    |   Tervetuloa.htm
    |
    \---images
            Add24.gif
            book_icon.gif
            Delete24.gif
            ErrorDialog.gif
            failed_mine.jpg
            Log.gif
            New24.gif
            Ok_mine.jpg
            Open24.gif
            Refresh24.gif
            refresh_all24.gif
            Remove24.gif
            Save24.gif
            Stop24.gif
```

Liite 2: JavaHelp-opastustiedostot

Tervetuloa.htm

```
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML//EN">
<HTML>
<HEAD>
<meta name="GENERATOR" content="Microsoft&reg; HTML Help Workshop 4.1">
<Title>Tervetuloa</Title>
</HEAD>
<BODY>

<H1>Tervetuloa PingPal-sovelluksen opastusjärjestelmään</H1>
<HR>
Tämä opastusjärjestelmä opastaa PingPal-sovelluksen käytössä.
<P>
Mikäli kaipaat ohjeita tämän opastusjärjestelmän käytöstä, kannattaa sinun tutustua
ensin kohtaan <a href="Ohjeen_kayttaminen.htm"> Ohjeen käyttäminen</a>

<H2> Tietoja PingPal-sovelluksesta </H2>
PingPal on Internet-palvelinten tilanhallintaan tarkoitettu työkalu. Sen avulla
voidaan keskitetysti valvoa mitä tahansa Internetissä tai paikallisessa lähiverkossa
olevia palvelimia sekä palvelimilla eri portteja kuuntelevia palvelinohjelmistoja.

<P>
PingPalin avulla voidaan mm. selvittää :
<UL>
   <LI>Onko yrityksen www-palvelin toiminnassa?
   <LI>Toimiiko sovelluspalvelin portissa 8080?
   <LI>Onko tietokantapalvelinohjelmisto kaatunut?
</UL>

PingPal suorittaa palvelimille kyselyt ajastetusti ja ilmoittaa käyttäjälle,
mikäli jokin määritetyistä osoitteista ei toimi. Palvelimen osoitteena voidaan antaa
joko palvelimen selkokielinen osoite tai IP-osoite (www.yritys.com tai 192.168.0.1).
Osoitteen perään voidaan määritellä testattava portti kaksoispisteen jälkeen
(www.yritys.com:1204), oletuksena käytetään WWW-porttia numero 80.

<P>
PingPalin avulla on mahdollista ylläpitää rajaton määrä palvelinten osoitelistoja.
Osoitelistat voidaan tallentaa tiedostoihin ja haluttu lista voidaan avata
sovelluksen käyttöön. Sovellus tukee myös osoitteiden testauksen ajastusta:
jokaiselle palvelimelle voidaan erikseen määritellä aikaväli, jonka kuluttua
osoitteeseen yritetään muodostaa yhteys.

</BODY>
</HTML>
```

```
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML//EN">
<HTML>
<HEAD>
<meta name="GENERATOR" content="Microsoft&reg; HTML Help Workshop 4.1">
<Title>Ohjeen käyttäminen</Title>
</HEAD>

<BODY>
<H1>Ohjeen käyttäminen</H1>

<HR>
Tämä opastesivu opastaa PingPal-sovelluksen opastusjärjestelmän käyttämisessä.
<P>
<H2>Ohjeen käynnistäminen</H2>
Tämä ohje saadaan näkyviin PingPalin ollessa käynnissä painamalla <b>F1</b>-painiketta
tai valitsemalla sovelluksen <b>Ohje</b>-valikosta <b>Ohjeen aiheet</b>.

<H2> Ohjeen rakenne  </H2>

Ohjeen näyttö on jaettu kolmeen paneeliin: <b>ylävalikko</b>, <b>navigointi</b> sekä
<b> sisältö -paneeleihin</b>.

<P>
<b>Ylävalikosta</b> voit mm. palate edelliseen katsomaasi opasteeseen,
valita näytetäänkö sisällyspaneelia tai tulostaa opastussivu.

<P>
<b>Navigointipaneelissa</b> (ikkunan vasen laita) voit selata ohjeen sisällysluetteloa
ja valita sieltä haluamasi opastusaiheen.
Vaihtoehtoisesti voit myös käyttää sisällyspaneelin välilehdiltä löytyviä hakemistoa
tai sanahakutoimintoa.

<P>
<b>Sisältöpaneelissa</b> (ikkunan oikea laita) näytetään varsinaiset ohjeet.
Ohjeen alalaidassa voi olla linkkejä kyseiseen ohjeeseen liittyviin aiheisiin.

</BODY>
</HTML>
```

```
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML//EN">
<HTML>
<HEAD>
<meta name="GENERATOR" content="Microsoft&reg; HTML Help Workshop 4.1">
<Title>Osoitelistan luominen</Title>
</HEAD>

<BODY>
<H1>Osoitelistan luominen</H1>

<HR>
PingPal käynnistyy oletusarvoisesti tyhjällä osoitelistalla, jolloin sovelluksen
yläpalkissa lukee teksti <i>Uusi osoitelista</i>.<P>
Mikäli olet jo muokkaamassa olemassa olevaa osoitelistaa ja haluat luoda
uuden tyhjän listan, paina työkalupalkin <IMG SRC="images/New24.gif">-ikonia
tai valitse <b>Tiedosto</b>-valikosta <b>Uusi</b>.

<HR>
<A HREF="Osoitelistan_tallentaminen.htm"><IMG SRC="images/book_icon.gif" border=0>
 Osoitelistan tallentaminen
</A>

<BR>
<A HREF="Osoitelistan_avaaminen.htm"><IMG SRC="images/book_icon.gif" border=0>
 Tallennetun osoitelistan avaaminen
</A>

<BR>
<A HREF="Osoitteen_lisaaminen.htm"><IMG SRC="images/book_icon.gif" border=0>
 Osoitteen lisääminen</A>
<BR>

</BODY>
</HTML>
```

```
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML//EN">
<HTML>
<HEAD>
<meta name="GENERATOR" content="Microsoft&reg; HTML Help Workshop 4.1">
<Title>Osoitelistan tallentaminen</Title>
</HEAD>

<BODY>
<H1>Osoitelistan tallentaminen</H1>

<HR>
Kun olet syöttänyt osoitelistan haluamasi määrän osoitteita, voit tallentaa listan
seuraavasti:
<OL>
   <LI>Paina työkalupalkin <IMG SRC="images/Save24.gif">-ikonia tai valitse
   <b>Tiedosto</b>-valikosta <b> Tallenna</b>.
   <LI>Näytölle avautuu dialogi-ikkuna, josta voit valita haluamasi
   tallennushakemiston.
   <LI>Syötä tiedostonimi sille varattuun kenttään.
   <LI>Tallenna painamalla <b>Save</b>-painiketta tai peruuta painamalla
   <b>Cancel</b>-painiketta.
</OL>

<P>
Mikäli valitsit Save, tiedosto tallennetaan haluamaasi paikkaan antamallasi
tiedostonimellä. Tiedoston päätteksi tulee <b>.ppf.</b>
Parhaillaan avoimena olevan osoitelistan nimi näkyy sovelluksen yläpalkissa.

<HR>
<A HREF="Osoitelistan_luominen.htm"><IMG SRC="images/book_icon.gif" border=0>
 Osoitelistan luominen
</A>

<BR>
<A HREF="Osoitelistan_avaaminen.htm"><IMG SRC="images/book_icon.gif" border=0>
 Tallennetun osoitelistan avaaminen
</A>
<BR>

</BODY>
</HTML>
```

```
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML//EN">
<HTML>
<HEAD>
<meta name="GENERATOR" content="Microsoft&reg; HTML Help Workshop 4.1">
<Title>Tallennetun osoitelistan avaaminen</Title>
</HEAD>

<BODY>
<H1>Tallennetun osoitelistan avaaminen</H1>

<HR>
Mikäli haluat avata aikaisemmin tallentamasi osoitelistan, toimi seuraavasti:
<OL>
   <LI>Paina työkalupalkin <IMG SRC="images/Open24.gif">-ikonia tai valitse
   <b>Tiedosto</b>-valikosta <b>Avaa</b>.
   <LI>Näytölle avautuu dialogi-ikkuna, josta voit valita haluamasi hakemiston
    sekä tiedoston.
    PingPalin osoitelistat tallennetaan oletusarvoisesti <b>.ppf</b>-päätteellä.
   <LI>Paina <b>Open</b>-painiketta.
</OL>

<P>
Haluamasi osoitelista avataan PingPal-sovellukseen.
Osoitelistan nimi näkyy sovelluksen yläpalkissa.

<HR>

<A HREF="Osoitelistan_luominen.htm"><IMG SRC="images/book_icon.gif" border=0>
 Osoitelistan luominen
</A>

<BR>

<A HREF="Osoitelistan_tallentaminen.htm"><IMG SRC="images/book_icon.gif" border=0>
 Osoitelistan tallentaminen
</A>
<BR>

</BODY>
</HTML>
```

```
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML//EN">
<HTML>
<HEAD>
<meta name="GENERATOR" content="Microsoft&reg; HTML Help Workshop 4.1">
<Title>Osoitteen lisääminen</Title>
</HEAD>
<BODY>
<H1>Osoitteen lisääminen</H1>
<HR>
Osoitteita voidaan lisätä seuraavasti:
<OL>
   <LI>Paina työkalupalkin <IMG SRC="images/Add24.gif">-ikonia tai valitse
       <b>IP-Osoite</b>-valikosta <b>Lisää osoite</b>.
   <LI>Osoitelistaan ilmestyy tyhjä rivi.
   <LI>Voit nyt syöttää rivin ensimmäiseen sarakkeeseen haluamasi palvelimen osoitteen.
       Osoite voi olla joko selkokielinen tai IP-osoite. Osoitteen perään voit lisätä
       kaksoispisteen jälkeen haluamasi porttinumeron. Oletusportti on 80.
   <LI>Listan toisesta sarakkeessa olevasta pudostusvalikosta (Ajastus) voit valita
       osoitteelle testausvälin (10 sekunnista yhteen viikkoon saakka). Tämä arvo on
       väliaika, jonka kuluttua osoite testataan aina uudelleen.
</OL>

<HR>
<A HREF="Osoitteen_poistaminen.htm"><IMG SRC="images/book_icon.gif" border=0>
 Osoitteen poistaminen
</A>

<BR>
<A HREF="Osoitteen_ajastus.htm"><IMG SRC="images/book_icon.gif" border=0>
 Osoitteen ajastus</A>
<BR>
<A HREF="Osoitteiden_testaus.htm"><IMG SRC="images/book_icon.gif" border=0>
 Osoitteen testaus</A>
<BR>

</BODY>
</HTML>
```

```
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML//EN">
<HTML>
<HEAD>
<meta name="GENERATOR" content="Microsoft&reg; HTML Help Workshop 4.1">
<Title>Osoitteen poistaminen</Title>
</HEAD>

<BODY>
<H1>Osoitteen poistaminen</H1>

<HR>
Osoitteita voidaan poistaa seuraavasti:
<OL>
   <LI>Valitse poistettava osoite napsauttamalla osoitteen riviä. Valittu rivi eroaa
       muista taustaväriltään.
   <LI>Paina työkalupalkin <IMG SRC="images/Delete24.gif">-ikonia tai valitse
      <b>Muokkaa</b>-valikosta <b>Poista</b>.
</OL>

<P>
Voit myös poistaa kaikki listan rivit seuraavasti:
<OL>
   <LI>Valitse <b>Muokkaa</b>-valikosta <b> Poista kaikki</b>. Näytölle ilmestyy
       varmistusdialogi, jossa kysytään haluatko varmasti poistaa kaikki osoitteet
       listalta.
   <LI>Paina <b>OK</b>. Lista tyhjenee.

</OL>

<HR>
<A HREF="Osoitteen_lisaaminen.htm"><IMG SRC="images/book_icon.gif" border=0>
 Osoitteen lisääminen
</A>
<BR>

<A HREF="Osoitteen_ajastus.htm"><IMG SRC="images/book_icon.gif" border=0>
 Osoitteen ajastus
</A>
<BR>

<A HREF="Osoitteiden_testaus.htm"><IMG SRC="images/book_icon.gif" border=0>
 Osoitteen testaus</A>
<BR>

</BODY>
</HTML>
```

```
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML//EN">
<HTML>
<HEAD>
<meta name="GENERATOR" content="Microsoft&reg; HTML Help Workshop 4.1">
<Title>Osoitteen ajastus</Title>
</HEAD>
<BODY>
<H1>Osoitteen ajastus</H1>
<HR>
Voit asettaa yksittäisen osoitteen testausvälin seuraavasti:
<OL>
   <LI>Valitse testattava osoite napsauttamalla osoitteen riviä. Valittu rivi eroaa
       muista taustaväriltään.
   <LI>Valitse rivin toisessa sarakkeessa (Ajastus) olevasta pudostusvalikosta
       haluamasi arvo. Testausväli voi vaihdella 10 sekunnista yhteen viikkoon.
   <LI>Osoitteen ajastus on nyt asetettu. Voit menetellä samoin kaikkien osoitteiden
       kohdalla.
</OL>

Käynnistääksesi osoitteiden ajastetun testauksen paina työkalupalkin
<IMG SRC="images/refresh_all24.gif">-ikonia tai valitse  <b> IP-osoite</b>-valikosta
<b>Testaa kaikki</b>. Järjestelmä testaa jokaisen osoitteen sille määritellyin
väliajoin.

<P>
Epäonnistuneista yhteyksistä järjestelmä ilmoittaa sovelluksen alalaidan tilarivillä.
Listalla epäonnistunut yhteys ilmenee
<IMG SRC="images/failed_mine.jpg">-kuvana tila-sarakkeessa.

<P>
Mikäli sovelluksen ikkuna on minimoitu (ts. jokin toinen sovellus on aktiivisena
työpöydällä), ilmoittaa järjestelmä epäonnistuneesta yhteydestä alla kuvatun
kaltaisella virhedialogilla.
<BR><IMG SRC="images/ErrorDialog.gif">

<P>
Jos haluat lopettaa osoitteiden ajastetun testauksen, paina työkalupalkin
<IMG SRC="images/Stop24.gif">-ikonia tai valitse <b>IP-osoite</b>-valikosta
<b>Pysäytä testaus</b>.

<HR>
<A HREF="Osoitteen_lisaaminen.htm"><IMG SRC="images/book_icon.gif" border=0>
 Osoitteen lisääminen </A><BR>

<A HREF="Osoitteen_poistaminen.htm"><IMG SRC="images/book_icon.gif" border=0>
 Osoitteen poistaminen</A><BR>

<A HREF="Osoitteiden_testaus.htm"><IMG SRC="images/book_icon.gif" border=0>
 Osoitteen testaus </A><BR>
</BODY>
</HTML>
```

```
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML//EN">
<HTML>
<HEAD>
<meta name="GENERATOR" content="Microsoft&reg; HTML Help Workshop 4.1">
<Title>Osoitteiden testaus</Title>
</HEAD>

<BODY>
<H1>Osoitteen testaus</H1>
<HR>
Voit testata yksittäisen osoitteen seuraavasti:
<OL>
   <LI>Valitse testattava osoite napsauttamalla osoitteen riviä. Valittu rivi eroaa
       muista taustaväriltään.
   <LI>Paina työkalupalkin <IMG SRC="images/Refresh24.gif">-ikonia tai valitse
       <b>IP-osoite</b>-valikosta <b>Testaa yhteys</b>.
   <LI>Valitun rivin Tila-sarakkeesen ilmestyy <IMG SRC="images/Ok_mine.jpg">-kuva,
       mikäli yhteys toimii, vastausaika sarakkeessa näkyy osoitteen vastausaika
       millisekunteina. Mikäli yhteys ei toiminut Tila-sarakkeeseen ilmestyy
       <IMG SRC="images/failed_mine.jpg">-kuva ja Vastausaika-sarake on valitulla
       rivillä tyhjä.
</OL>

<HR>
<A HREF="Osoitteen_lisaaminen.htm"><IMG SRC="images/book_icon.gif" border=0>
 Osoitteen lisääminen
</A>
<BR>

<A HREF="Osoitteen_poistaminen.htm"><IMG SRC="images/book_icon.gif" border=0>
 Osoitteen poistaminen
</A>
<BR>

<A HREF="Osoitteen_ajastus.htm"><IMG SRC="images/book_icon.gif" border=0>
 Osoitteen ajastus
</A>
<BR>

</BODY>
</HTML>
```

```
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML//EN">
<HTML>
<HEAD>
<meta name="GENERATOR" content="Microsoft&reg; HTML Help Workshop 4.1">
<Title>Lokitiedoston käsittely</Title>
</HEAD>

<BODY>
<H1>Lokitiedoston lukeminen</H1>
<HR>
Lokitiedostoa pääsee selaamaan valitsemalla sovelluksen pääikkunassa
<b>Loki</b>-välilehden.
<br>
Lokiin tallentuu päivämäärä, kellonaika, osoite johon yhteyttä yritettiin,
vasteaika sekä tieto siitä onnistuiko yhteys vai ei.
Lokitiedostoa voi selata edestakaisin vierityspalkkia hyväksikäyttäen.
<P>

<IMG SRC="images/Log.gif">
<br>
<b>Kuva Loki-välilehdestä.</b>
<HR>
<A HREF="Lokitiedoston_tyhjentaminen.htm"><IMG SRC="images/book_icon.gif"
 border=0>Lokitiedoston tyhjentaminen
</A>

</BODY>
</HTML>
```

```
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML//EN">
<HTML>
<HEAD>
<meta name="GENERATOR" content="Microsoft&reg; HTML Help Workshop 4.1">
<Title>Lokitiedoston tyhjentäminen</Title>
</HEAD>

<BODY>
<H1>Lokitiedoston tyhjentäminen</H1>
<HR>
Lokitiedosto voidaan tyhjentää <b>Loki</b>-välilehdellä olevaa
<b>Tyhjennä</b>-painiketta painamalla.
Huomaa, että lokin tyhjentämisen jälkeen sen sisältöä ei voi enää palauttaa.
<P>
<HR>
<A HREF="Lokitiedoston_lukeminen.htm"><IMG SRC="images/book_icon.gif" border=0>
Lokitiedoston lukeminen
</A>
<BR>

</BODY>
</HTML>
```