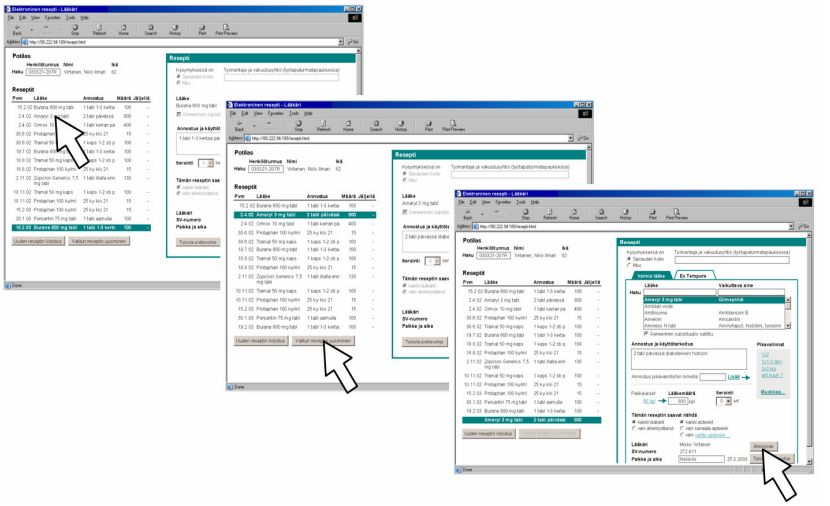
Käyttöliittymät

Sari A. Laakso
Antti Latva-Koivisto



Helsingin yliopisto
Tietojenkäsittelytieteen laitos
Sarja D-2006-1

Helsinki 5.3.2006

Esipuhe

Hyvä käyttöliittymä saadaan aikaan selvittämällä käyttäjien tavoitteet ja käyttötilanteet ja tekemällä tilanteiden hoitamisesta mahdollisimman suoraviivaista ja itsestään selvää. Nämä käyttäjille päivittäin eteen tulevat tilanteet ovat testitapauksia, joiden suorittamista tarkastelemalla hyvät käyttöliittymäratkaisut erottuvat huonoista (käytettävyys), tarvittava tietosisältö erottuu tarpeettomasta ja hyödylliset toiminnot hyödyttömistä.

Tällä kurssilla harjoitellaan kahden käyttöliittymän arviointimenetelmän käyttöä, simulointitestausta ja käytettävyystestausta. Molempien menetelmien syötteeksi opitaan laatimaan käyttötilannekuvauksia.

Samoja käyttötilanteita voidaan hyödyntää testauksen lisäksi käyttöliittymän suunnittelussa. Tällä kurssilla luodaan katsaus systemaattisiin suunnitteluprosesseihin ja tutustutaan lähemmin yhteen prosessiin, tietojenkäsittelytieteen laitoksella ja Interacta Designissa kehittämäämme simulointipohjaiseen suunnitteluun (Goal-Derived Design, GDD), joka tuottaa käyttötilanteista käyttöliittymän näyttökuvat ja toimintalogiikan sekä käyttäjälle näkyvät tietosisältövaatimukset ja toiminnalliset vaatimukset. GDD-suunnitteluprosessissa on yhteisiä aineksia mm. Søren Lauesenin Virtual Windows -menetelmän, Alan Cooperin Goal-Directed Design -menetelmän sekä skenaariopohjaisen suunnittelun kanssa.

GDD-prosessin ja muiden suunnitteluprosessien perusteellisempi tarkastelu ja käyttöliittymäsuunnittelun vaikutukset vaatimusmäärittelyyn on sijoitettu Käyttöliittymät II -jatkokurssin puolelle. Peruskurssilla testauksen ja suunnittelun syötteenä olevat käyttötilanteet määritellään itse, mutta jatkokurssilla opitaan käyttötilanteiden selvittämistä varten kenttätutkimus-menetelmiä. Jatkokurssilla myös tutustutaan laajemmin erilaisiin arviointi- ja testausmenetelmiin ja opitaan sijoittamaan testaus- ja suunnittelumenetelmiä todellisten ohjelmistoprojektien vaiheisiin.

Tämä peruskurssi sisältää runsaasti konkreettisia esimerkkejä ja demoja, joiden avulla harjoitellaan parempien ja huonompien käyttöliittymäratkaisujen erottamista toisistaan ja opitaan soveltamaan joitain suoraviivaisia ratkaisuja tyypillisimpiin käyttötilanteisiin. Kurssin tavoitteena on antaa käytännön suunnittelutaitoja tulevien opinnäytetöiden ja työpaikan projektitöiden käyttöliittymien laatimista varten ja valmiudet asettaa eri ratkaisuvaihtoehtoja paremmuusjärjestykseen simuloimalla käyttöä ja käytettävyystestaamalla niitä.

Tätä kurssimateriaalia laatiessamme toivomme, etteivät tällä kurssilla hankitut suunnittelutaidot jäisi elämään vain tämän kurssin sisälle, vaan että muuttaisit maailmaa viemällä yritysten ohjelmistoprojekteihin uusia tapoja käyttäjille hyödyllisten ohjelmistojen suunnittelemiseksi. Teknisten määrittelyjen tai taloudellisen hyödyn tavoittelun ehdoilla suunniteltujen laitteiden ja ohjelmistojen sijaan tarvitsemme järjestelmiä, jotka on rakennettu ihmisten tavoitteiden ja ihmisille eteen tulevien tilanteiden ehdoilla. Se ei ole pelkkää käytettävyyttä, vaan se johtaa paljon syvemmälle. Usko itseesi ja tee se, mikä on ilmeistä mutta mitä niin kovin harvat vasta tekevät.

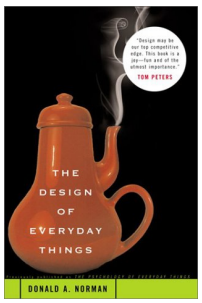
Helsingissä 5.3.2006

Sari A. Laakso

Sisältö

Sisältö

Itsestäänselvyyden periaatteita: opittavuus ja virheiden välttäminen



[Norman90]

Normanin periaatteita

- **Mahdollisuudet** (affordances)
- **Rajoitteet** (constraints)
- **Sama järjestys** (mapping)
- **Näkyvyys** (visibility)

Mahdollisuudet ja rajoitteet
Esimerkki: TKTL:n hissinovet

Mahdollisuudet (affordances)

Rajoitteet (constraints)

Mahdollisuudet ja rajoitteet
Lisää oviesimerkkejä

Hyvät mahdollisuudet ja rajoitteet

Konferenssikeskus Seattlessa

Harhaanjohtavat mahdollisuudet ja rajoitteet

Helsingin Kluuvikadun Fazerin kahvila

Sama järjestys
Esimerkki: Auton sähköikkunat

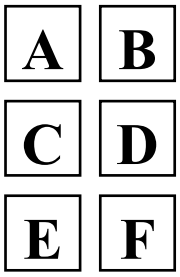
Kuvan autossa sähköikkunoiden säätökytkimet ovat **samassa järjestyksessä** (mapping) kuin fyysiset ikkunat

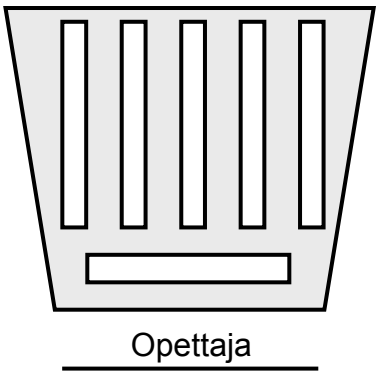
(Mazda 626)

Ei samaa järjestystä
Esimerkki: Automaattivaihteet

Ei samaa järjestystä
Esimerkki: Auditorion valokatkaisijat

Vanhan tietojenkäsittelytieteen laitoksen (Teollisuuskatu 23) auditorion valokatkaisijat





Ei samaa järjestystä
Esimerkki: Stockmannin hissi

Eri järjestys:

Sama järjestys:

Näkyvyys
Esimerkki: Sokeripalat

Tuo vihjeet *näkyville!*

Sokeripalan tarjoamien mahdollisuuksien **näkyvyys** (visibility) on niin heikko, etteivät ihmiset osaa avata niitä oikein.

Ohjetekstejä ei tarvita, jos suunnittelija osaa luoda visuaaliset vihjeet, joiden avulla käyttäjälle on itsestään selvää, miten pakkaus toimii.

Tavoittele suunnittelijana ensisijaisesti itse käyttöliittymän parantamista. Ohjeiden lisääminen on vasta viimeinen keino, jos suunnittelija ei saa mitään parempaa keinoa toimimaan.

Näkyvyys
Esimerkki: Käsipyyhkeet

Ongelma: Käsipyyhetelineen tarjoamien mahdollisuuksien **näkyvyys** on heikko. Tässäkin varsinaista ongelmaa on yritetty korjata ohjetekstillä.

Miten itse ongelma voitaisiin korjata tässä? Miten ohjetekstille silloin käy?

Ohjelmiston käyttöönotossa eteen tulevat ongelmat

Viimeistään ohjelmiston käyttöönottovaiheessa järjestelmän avulla ryhdytään tekemään oikeita työtehtäviä ja suorittamaan todellisia käyttötilanteita. Tällöin puutteet ohjelmiston tietosisällössä, toiminnallisuudessa ja käytettävyydessä tulevat viimeistään esille.

Ohjelmiston käyttöönotto
Ongelmat tulevat esille

- Monesti ohjelmisto toteutetaan ja otetaan käyttöön ilman minkäänlaista käyttötilanteiden suorittamiseen perustuvaa arviointia tai testausta.
- Tällöin ohjelmiston käyttökelpoisuus tavallaan testataan vasta kentällä, jolloin ongelmien korjaaminen on kallista.

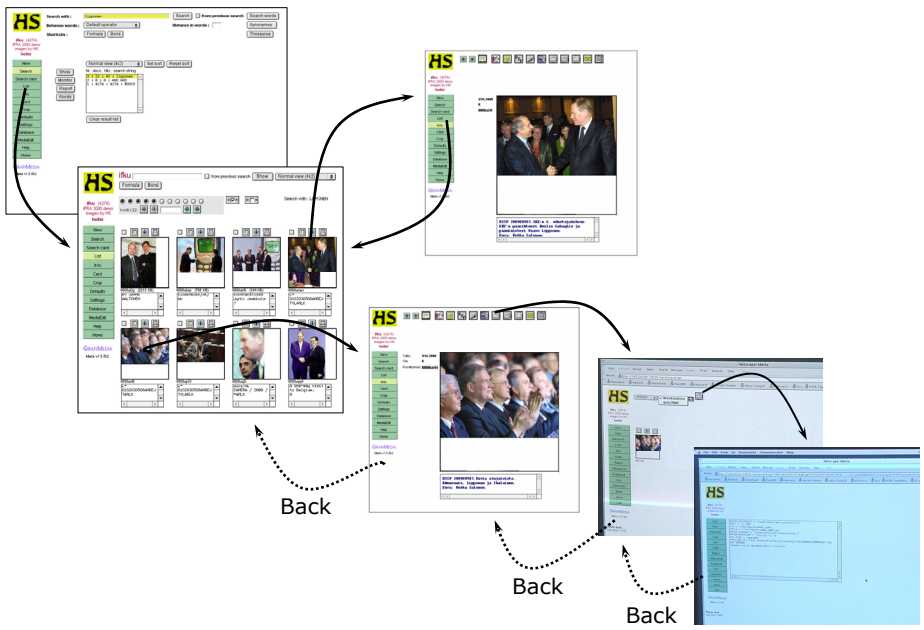
Ohjelmiston käyttöönotto
Ongelmat tulevat esille

Esimerkkejä ongelmista:

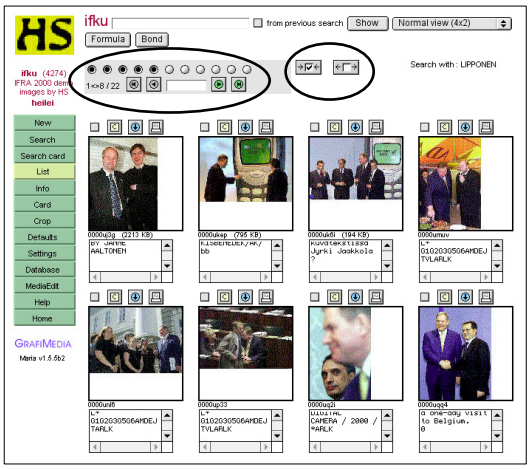
- Työtehtävien suorittaminen on vaivalloista ja aikaavievää.
- Käyttäjät eivät keksi, miten järjestelmää pitäisi käyttää. Käyttäjät tekevät turhaan virheitä, ja koulutukseen kuluu paljon resursseja.
- Kaikki työnkulut eivät edes 'mene läpi', vaan osa vaiheista on suoritettava kiertoteitse paperilla tai muilla ohjelmistoilla.
  - Järjestelmästä puuttuu **toimintoja**.
  - Järjestelmästä puuttuu työtehtävissä tarvittavaa **dataa**, tai data on organisoitu käyttäjän työtehtävän kannalta väärin.
- Järjestelmässä ylläpidetään toimintoja tai dataa, joita kukaan käyttäjä ei oikeasti koskaan tarvitse.

Ohjelmiston käyttöönotto
Esimerkki: Kuva-arkisto

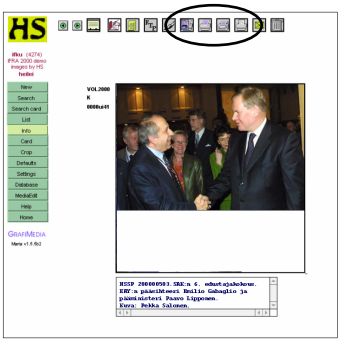
- Kuvatoimittaja täydentää Paavo Lipposesta kertovaa lehtijuttua sopivilla kuvilla.



Opittavuus: Käyttäjä ei ymmärrä ohjelman toimintalogiikkaa eikä keksi, mitä ohjelman toiminnot tarkoittavat ja miten niitä käytetään.

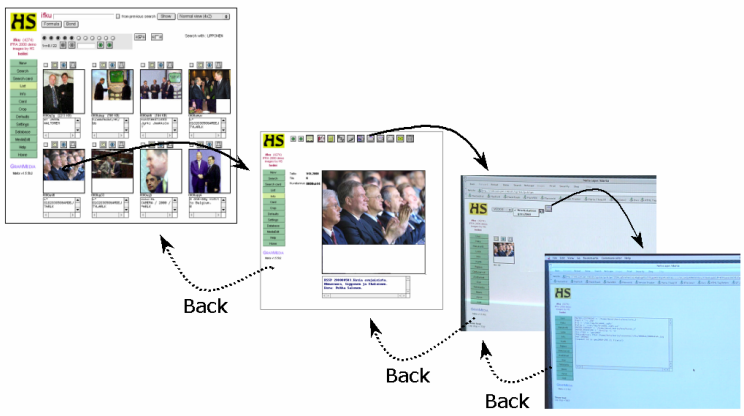


Virheet: Käyttäjä tekee tarpeettomia virheitä epäselvien tai harhaanjohtavien toimintojen takia.



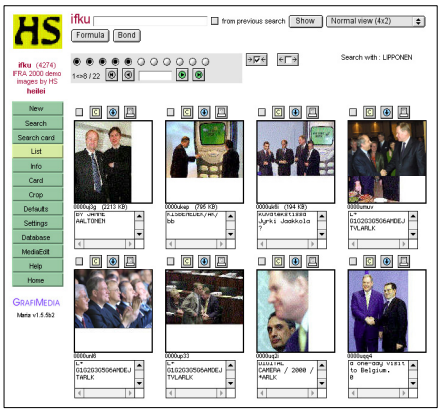
Tehokkuus: Käyttäjä joutuu tekemään turhaa työtä tavoitteensa saavuttamiseksi.

Tulostamisen monivaiheinen polku:



Puuttuva toiminnallisuus: Ohjelmasta puuttuu toimintoja tai tietoja, joita käyttäjä tarvitsisi työnkulkunsa osaksi.

Kuvien keräämistoiminto (ns. ostoskori) puuttuu:



Tietosisältö, toiminnallisuus ja käytettävyys

Käytettävyyttä ei voida lisätä väärän tietosisällön tai toiminnallisuuden päälle. Hyvä käyttöliittymä sisältää ensisijaisesti sellaisen tietosisällön ja toiminnallisuuden, että käyttäjien on mahdollista saavuttaa tavoitteensa (työtehtävät ylipäätään 'menevät läpi').

Kun työt on järjestelmällä mahdollista tehdä, voidaan parantaa niiden tekemisen tehokkuutta ja opittavuutta ja yrittää estää tarpeettomat virheet.

Hyvä käyttöliittymä
Hyödyllisyys ja käytettävyys

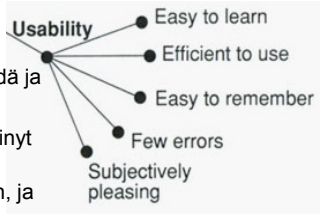
1. Hyödyllisyys (utility): Voiko tehdä oikeaa asiaa?

Onko järjestelmässä sellaiset **toiminnot** ja **tietosisältö**, että oikeiden työtehtävien tekeminen 'menee läpi' – vaikka vaikeasti ja vaivalloisestikin?

Jos oikean asian pystyy tekemään jotenkin:

2. Käytettävyys (usability): Onko tekeminen sujuvaa?

Tehokkuus (efficiency)	Onko turhia vaiheita
Opittavuus (learnability)	Keksiikö käyttäjä, mitä pitäisi tehdä ja mitä tiedot tarkoittavat
Muistettavuus (memorability)	Onko selvää, kun on kerran keksinyt
Virhealttius (errors)	Houkuttaako käli virhetoimintoihin, ja miten käyttäjä selviytyy virheistä
Tyytyväisyys (satisfaction)	Kokeeko käyttäjä käytön miellyttävänä



[Nielsen93, s. 25; Nielsen03]

Sisältö

Käyttöliittymän testaus

Testauksen tarkoituksena on löytää mahdollisimman paljon käyttöliittymän ongelmakohtia tai esimerkiksi asettaa kaksi erilaista käyttöliittymäratkaisua paremmuusjärjestykseen.

Sekä simulointitestauksessa että käytettävyystestauksessa testitapauksena on realistinen käyttötilanne (työtehtävä), jota lähdetään suorittamaan.

Käyttötilanteet testitapauksina

Testitapaus kuvaa käyttötilanteen lähtöasetelman mutta ei vielä järjestelmän käyttöä (mitä käyttäjä tekee ja miten järjestelmä siihen reagoi), koska nimenomaan tätä on tarkoitus ryhtyä testaamaan. Testitapaus on syöte testaukselle.

Käyttötilanteet testitapauksina
Lyhyt skenaario

- Hyvä testitapaus sisältää taustatilanteen kuvauksen hieman vastaavalla tavalla kuin Hackosin *lyhyt skenaario* (*brief scenario):*

> Joan Gaynor, the travel agent, takes a call from Mrs. Reed. The Reed family is going on a vacation and they want Joan to arrange for airline tickets and to reserve a rental car for them. When Joan talks about it, she says she is booking a car for the Reeds. The family includes two adults and three children: 11, 7, and 3. They need a car that will hold five and all their luggage. They need a car seat for the toddler. They want to pick up the car when they fly into the airport in Salt Lake City and drop it off at the same airport a week later just before their flight back. They want to rent from an agency that has cars at the airport so they don't have to carry their luggage. They want the cheapest rental they can get for the week with no mileage charges.

[Hackos98, s. 324]

Käyttötilanteet testitapauksina
Lähtötilanne, ei järjestelmän käyttöä

- *Lyhyen skenaarion* kuvaama lähtötilanne pysyy samana, vaikka tehtävän suoritustapaa ja käyttöliittymää muutetaan [Hackos98, s. 324]: Lyhyt skenaario...
  - kuvailee tilanteen, joka käyttäjien pitää pystyä hoitamaan käyttöliittymän avulla, mutta
  - ei kuvaa sitä, miten käyttäjät tekevät tämän tehtävän nykyisellä käyttöliittymällä (mitä toimintoja käynnistävät, mitä tietoja katsovat käyttöliittymästä, missä järjestyksessä jne.).
- (Edellisestä esimerkistä kannattaa huomata, että vaikka siinä onkin konkreettinen tilanne pohjana, sitä voisi vielä parantaa paljon. ”Käyttäjien haluamisen”, kuten *they want to rent, they want the cheapest,* sijaan olisi parempi kuvata tilanneasetelmaa ja haluamisen syitä.)

Käyttötilanteet testitapauksina
Lähtötilanne, ei järjestelmän käyttöä

Lähtötilanne:
Petteri on yliopistolla osoitteessa Teollisuuskatu 23 koodaamassa demo-ohjelmaa kahden työkaverinsa kanssa. On jo ilta klo 21.15, ja kaikilla alkaa olla hirveä nälkä. Koodaus on vielä ihan kesken, ja sen kanssa näyttää menevän myöhään. Kellään ei ole mukana mitään syötävää.

Ei järjestelmän käyttöä:
Petteri avaa hampurilaisravintolan tilauspalvelun ja valitsee 6 megaburgeria ja kolmet isot ranskalaiset. Sitten hän syöttää osoitteen ja painaa tilauspainiketta. Näytön alareunaan ilmestyy viesti: ”Tilaus lähetetty, saapuu puolen tunnin kuluttua.”

- Testitapauksen sisältämä lähtötilannekuvaus on **syöte** testaukselle.
- Vasta testauksen aikana selvitetään, miten käyttäjä pääsee tavoitteeseensa käyttöliittymän avulla. Käytön kuvaus on siis testauksen **tuotos**.

Käyttötilanteet testitapauksina
Lähtötilanne, ei järjestelmän käyttöä

Lähtötilanne (syöte testaukselle):
Helsinkiläinen lääkäri Terhi on menossa ensi viikon torstaina Turussa klo 12 alkavaan sydäntautiseminaariin. Seminaari alkaa TYKSin johtavan sydäntautilääkärin tilannekatsauksella (klo 12-13). Viimeinen esitys pidetään klo 17.15-17.45. Terhi tietää, että seminaaripaikka on keskustassa osoitteessa Brahenkatu 10 noin viiden minuutin kävelymatkan päässä linja-autoasemasta.

Järjestelmän käyttöä (vasta testauksen tuloksena):
Terhi avaa Matkahuollon sivuston. Etusivulta hän klikkaa *Aikataulut* ja kirjoittaa näppäimistöltä lähtöpaikaksi Helsinki ja määräpaikaksi Turku. Lähtöpäivän hän katsoo kalenteristaan, kirjoittaa kenttään 24.3. ja painaa *Hae*. Järjestelmä löytää useita paikkoja ja kehottaa jatkamaan ehdotetuilla paikoilla. Terhi painaa uudestaan *Hae.* Hakutuloksista hän löytää bussin, joka lähtee Helsingistä klo 9.30. Sitten hän painaa *Paluumatka*-linkkiä ...

Käyttötilanteet testitapauksina
Käyttäjän tavoite (motiivi)

- Hyvään lähtötilannekuvaukseen sisältyy käyttäjän tavoite (motiivi): mitä hän on yrittämässä tehdä ja miksi?
  - Testauksen kannalta ei ole juurikaan merkitystä, mikä motiivi tilanteessa on, kunhan se on realistinen ja riittävän tyypillinen.
  - Jos motiivi puuttuu, testaaja ei pysty ottamaan huomioon tilanteeseen tyypillisesti vaikuttavia tekijöitä. Tästä seuraa, ettei testauksella saada esiin kaikkia keskeisiä, todellisessa tilanteessa eteen tulevia ongelmia. Esimerkki:

    Käyttötilanne: *Ensi viikon torstaina Turkuun*
    Helsinkiläinen lääkäri Terhi on menossa ensi viikon torstaina Turkuun. Perillä pitää olla klo 12, ja paluu on klo 18.

  - Puuttuvan motiivin takia aikatauluhaun testauksesta menisi ongelmitta läpi huono käyttöliittymäratkaisu, joka näyttäisi kerralla vain ensimmäisen paluubussin (esim. klo 18 lähtevän), vaikka monissa todellisissa tilanteissa Terhin kannattaisikin valita myöhemmin (esim. klo 18.30 tai klo 19) lähtevä bussi.

Käyttötilanteet testitapauksina
”Käyttäjä haluaa” -ongelma

- Testitapausta laatiessa kannattaa välttää ylimalkaista ilmausta ”Käyttäjä haluaa/haluaisi/tahtoo”, koska tällöin testitapauksen motiivi jää epäselväksi eivätkä kaikki todellisen tilanteen ongelmat tule esille (vrt. edellinen kalvosivu).
- Koska testitapauksesta ei selviä, mihin haluaminen perustuu, osa ”haluamisesta” voi olla epärealistista tai koko testitapaus voi osoittautua virheelliseksi tai niin epätyypilliseksi, ettei sillä kannata testata.
- Haluamisen takana olevan informaation saat esille yleensä kysymällä ”**miksi**”.

Ei käyttäjän haluamista:
Helsinkiläinen lääkäri Terhi on menossa ensi viikon torstaina Turkuun. Hän haluaa olla perillä klo 12 ja haluaa lähteä takaisin klo 18.

Kuvaa tilannetta ja syitä haluamisen takana:
Helsinkiläinen lääkäri Terhi on menossa ensi viikon torstaina Turussa klo 12 alkavaan sydäntautiseminaariin. Seminaari alkaa TYKSin johtavan sydäntautilääkärin tilannekatsauksella (klo 12-13). Päivän viimeinen esitys pidetään klo 17.15-17.45...

Simulointitestaus

Käytön simulointiin perustuvissa menetelmissä ei ole testikäyttäjiä, vaan arvioija testaa käyttöliittymää simuloimalla itse käyttötilanteen suorituspolkua (käyttäjän syötteitä, painikkeiden painalluksia, näytöltä luettuja tekstejä jne.). Simuloinnin aikana arvioija ennustaa käyttäjän toimintaa ja ajattelua.

Käytön simulointiin perustuvia menetelmiä

- Käyttöliittymän **simulointitestauksessa** [Laakso02] arvioija selvittää ensin parhaan lopputilan käyttötilanteen suorittamiseksi ja sitten jäljittelee käyttäjän toimenpiteitä ja ajatuksia vaihe vaiheelta (simulointi). Näin paljastuu puuttuvaa tietosisältöä ja toiminnallisuutta sekä tehokkuusongelmia ja jossain määrin opittavuusongelmiakin.
- **Kognitiivisella läpikäynnillä** (cognitive walkthrough; ks. esim. [Lewis97]) arvioidaan ensisijaisesti jokaisen toimenpideaskelen opittavuutta: keksiikö käyttäjä, mikä toimenpide seuraavaksi pitäisi tehdä. Arvioija simuloi käyttäjän toimenpiteitä vaihe vaiheelta ja vastaa jokaisen toimenpiteen yhteydessä neljään apukysymykseen.
- Myös **heuristisessa läpikäynnissä** (heuristic walkthrough) [Sears97] arvioija tarkastelee käyttäjän toimenpidesekvenssiä, mutta neljän apukysymyksen lisäksi hän etsii ongelmia käyttämällä Nielsenin tarkistuslistaa [Nielsen94].

Simulointitestauksen tekeminen

Simulointitestauksessa [Laakso02] asiantuntija selvittää testitapauksen suorittamiseen tarvittavan käyttösekvenssin jäljittelemällä vaihe vaiheelta käyttäjän toimenpiteitä ja ajatuksia. Sen jälkeen hän paikantaa ongelmakohdat simuloimalla käyttösekvenssin suorittamista useaan kertaan ja tarkastelemalla käyttäjältä vaadittavaa työtä.

Simulointitestaus
Menetelmän vaiheet

- Lähtökohdaksi arvioija selvittää jonkin tosielämässä käyttäjälle eteen tulevan **konkreettisen käyttötilanteen**.
- Arvioija arvioi ensin, mikä olisi valitussa käyttötilanteessa käyttäjälle **paras loppuratkaisu** (esimerkiksi huoneen varaaminen 2 yöksi hotelli Ilveksestä taikka Michelinin Lontoo-matkaoppaan tilaaminen kirjakauppapalvelusta).
- Seuraavaksi arvioija selvittää simuloimalla, mikä on käyttöliittymän tarjoama ensisijainen **oikea polku** loppuratkaisun saavuttamiseksi:
  - mitkä toimenpiteet käyttäjän pitää tehdä ja
  - missä kohdissa käyttäjä lukee tietoja näytöltä, vertailee vaihtoehtoja ja tekee päätöksiä.
- Arvioija jäljittelee askel askeleelta käyttäjän ajatuksia ja toimenpiteitä, kuten painikkeiden painalluksia ja syötteiden kirjoittamista, ja **simuloi useaan kertaan** samaa sekvenssiä.

Simulointitestaus
Vihjeitä simulointivaiheeseen

- Simuloi käyttäjän toimenpiteet niin **yksityiskohtaisesti** kuin mahdollista. Ylimalkaisen luonnehdinnan (esim. ”Tästä se sitten katsoisi sopivan bussin”) sijaan simuloi
  - jokainen käyttäjän **toimenpide** (esim. vierittää puoli näyttöä alaspäin, siirtyy seuraavalle sivulle, siirtyy takaisin) sekä
  - **tietojen lukeminen** näytöltä (käyttäjä lukee, että ”perillä vasta 14.40, liian myöhään”, sitten katsoo edellistä: ”perillä jo 13.00, mutta tässä on hidas vaihto” jne.)
- Käytä koko ajan **todellisia syötteitä**. Esim. käyttäjän kotiosoite voi olla ”Maaherrankatu 25 D 9”, mutta ei ”asdf” tai ”xxxxx”.

Simulointitestaus
Esimerkki: Matkahuolto ja VR

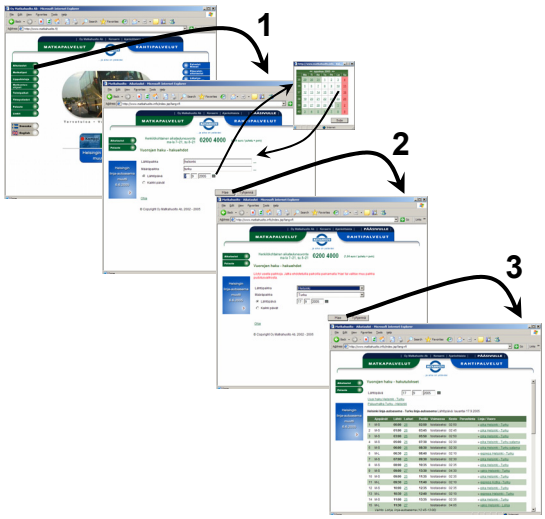
- Hyvä testitapaus on **riippumaton testattavasta järjestelmästä**: testitapauksessa ei kuvata vielä järjestelmän käyttöä, vaan käyttöä edeltävää tilanneasetelmaa.
- Seuraavassa esimerkissä testitapaukseksi on valittu todellinen tilanne, jossa Markus oli menossa kummipoikansa syntymäpäiville Turkuun ja yritti selvittää, miten pääsisi sinne.

> ***Käyttötilanne:*** *Kummipojan syntymäpäiville Turkuun*
>
> *Marko, 27-vuotias ravintolakokki, on maanantai-iltana kotonaan Helsingin Kalliossa osoitteessa Kolmas linja 23. Hän on menossa ensi lauantaina 4-vuotiaan kummipoikansa syntymäpäiväjuhliin Turkuun. Juhlat alkavat klo 14 kummipojan kotona Rauhankadulla, joka sijaitsee Turun keskustassa rautatie- ja linja-autoaseman lähellä. Marko arvelee lähtevänsä sieltä kotiin joskus kuuden maissa.*

- Testataan nyt käyttöä simuloimalla, miten hyvin Matkahuollon ja VR:n web-sivustot tukevat Markon tilannetta.

VR:n ja Matkahuollon sivustojen simulointitestaus
Sivusiirtymät menomatkan aikataulujen selviämiseen asti

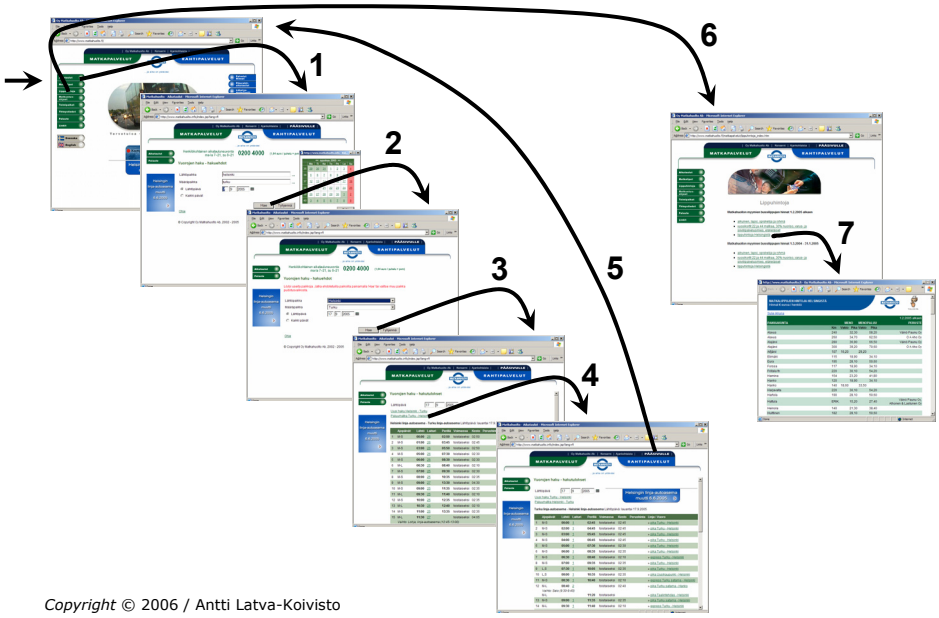
Matkahuolto



VR

VR:n ja Matkahuollon sivustojen simulointitestaus

Matkahuollon sivusiirtymät

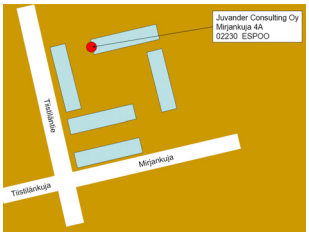


Simulointitestaus

Simulointi loppuun asti

- Simulointia ei pidä lopettaa vielä tietokoneohjelman **käytön päättymiseen** (esim. *kun käyttäjä on saanut varattua hotellin web-palvelusta huoneen ja sulkee ohjelman*), vaan...
- ...se tulee viedä **loppuun** aina **tavoitteen saavuttamiseen asti** (esim. *kunnes käyttäjä on ajanut autolla perheineen hotelli Mesikämmeneen ja mennyt huoneeseensa*).
- Käyttötilanteen loppupään simulointi voi paljastaa yllättäviä ongelmakohtia käyttöliittymästä. Esimerkiksi ajokartan avulla ei oikeasti löydäkään perille, tai hotellihuone ei vastaakaan sitä, mitä varausjärjestelmä käyttäjälle lupasi.

Lähde: www.scandic-hotels.fi



Lähde: www.juvander.fi/yhteystiedot/Company_Location.pdf

Simulointitestaus
Käyttäjän tietämys ja paras ratkaisu

- Arvioijan oma tietämys on eri asia kuin käyttäjän tietämys. Arvioija voi tietää, että käyttäjän kannattaisi tällä kertaa esimerkiksi mennä hotelli Ilvekseen (paras loppuratkaisu), mutta käyttäjä ei välttämättä aluksi tiedä, mikä hotelli olisi hänen tilanteessaan paras tai missä olisi tilaa (käyttäjän tietämys).
- Kiinnitä käyttötilannetta simuloidessasi huomiota käyttäjän tietämykseen: mitä käyttäjä tässä käyttötilanteessa tietää ja mitä hän aluksi ei tiedä?
- Esimerkki: VR/Matkahuolto
  - Ensin arvioija selvittää Markon tilanteeseen parhaan loppuratkaisun:
    - menomatka klo 11.30 lähtevällä bussilla ja
    - paluumatka klo 18.30 tai klo 19.00 lähtevällä bussilla.
  - Tässä tilanteessa Marko ei aluksi tiedä em. parasta ratkaisua. Seuraavaksi arvioija selvittää simuloimalla Markon ajatuksia ja toimenpiteitä, miten Marko saisi parhaiten tämän tiedon selville järjestelmää käyttämällä (löytäisi sopivimmat bussit).

Simulointitestaus
Hyvä päätöksenteon testisekvenssi

- Kun testataan käyttäjän päätöksentekoa sisältäviä vertailutilanteita, arvioijan tulee simuloida riittävän kehittyneellä päätöksentekoprosessilla, jotta **mielessä pitämisen ongelmat** saadaan esiin:
  - Tarjontaa kuvaavassa esimerkkidatassa on useita **hyviä vaihtoehtoja**: käyttäjän täytyy punnita näitä keskenään ja arvioida, mikä olisi paras tässä tilanteessa.
  - Esimerkkidata sisältää myös käyttäjälle **huonoja vaihtoehtoja**: käyttäjän pitäisi pystyä nopeasti havaitsemaan nämä huonoiksi.
- Jos arvioija testaa epärealistisen yksinkertaista päätöksentekoa (esim. käyttäjä menee Turkuun aamun ensimmäisellä junalla, oli se mikä tahansa) tai käyttää pelkästään helppoja erikoistapauksia (junia menee Kemijärvelle vain yksi päivässä), testauksella ei tule esiin mielessä pitämisen ongelmia.

Käyttösekvenssit kuvasarjoina ja skenaarioina

Simulointitestauksessa selville saaduista käyttösekvensseistä kannattaa usein laatia **kuvasarjoja**, joiden avulla arvioija voi toistaa sekvenssiä läpi helposti useita kertoja. Samalla sekvenssi tulee dokumentoitua talteen.

Simulointisekvenssit voidaan tallentaa myös tekstimuotoisina **skenaarioina**, joissa arvioija kuvaa käyttäjän toimenpiteet ja järjestelmän reaktiot niihin.

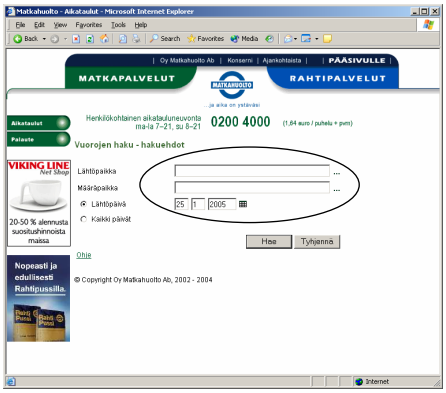
Käyttösekvenssit kuvasarjoina
Sekvenssin dokumentointi

- Simulointitestauksen sisältämistä vaiheista kannattaa koota **kuvasarja**, josta näkyy käyttäjän jokainen toimenpide. Kuvasarja auttaa arvioimaan käyttöliittymän ongelmakohtia, ja sen avulla on havainnollista esittää simuloinnin tuloksia.
- Kuvasarjan ensimmäisenä kuvana on **järjestelmän alkutila**, jossa käyttäjä ei ole vielä tehnyt ensimmäistäkään toimenpidettä (esim. etusivu www.matkahuolto.fi).
- Jokaiseen kuvaan merkitään **yksi käyttäjän toimenpide**, joka on tyypillisesti hiirellä klikkaaminen (merk. nuolikursorin kuvalla) tai tiedon syöttäminen näppäimistöltä (ympyröi syötteet).
- **Toimenpiteen seuraus** eli järjestelmän reaktio esitetään seuraavassa kuvassa. Jos käyttäjä esim. täyttää peräkkäin monta syötekenttää, joihin ohjelma ei reagoi mitenkään, ne voidaan merkitä samaan kuvaan. Aina, kun ohjelma reagoi käyttäjän toimenpiteeseen jotenkin, tehdään uusi kuva.

Esimerkkikatkelma kuvasarjasta: www.matkahuolto.fi

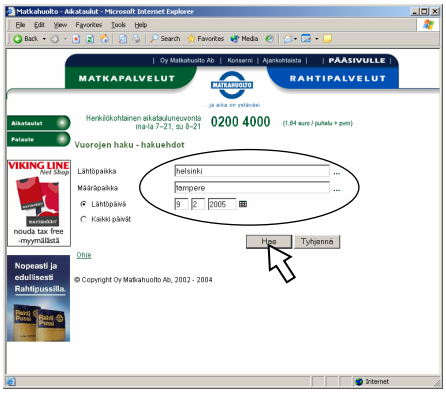
Huom. Alareunan tekstit ovat ohjeita, jotka eivät tule mukaan itse kuvasarjaan.

Alkutilakuvana on sellainen näyttökuva, jossa käyttäjä ei ole vielä tehnyt ensimmäistäkään toimenpidettä. (Alkutilakuvaan voit merkitä ensimmäisen **hiiren painalluksen nuolikursorin kuvalla**.)

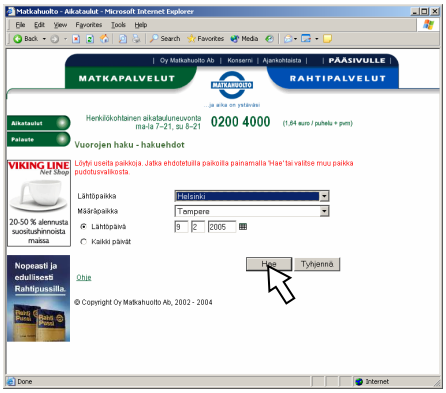


Näyttö avautuu tämän näköisenä, kun käyttäjä on tehnyt edellisen kuvan toimenpiteen eli painanut hiirellä *Aikataulut*-linkkiä vasemmasta reunasta. Tähän kuvaan merkitään **ympyröinnillä kentät**, joihin käyttäjä seuraavaksi **syöttää tietoja**.

Esimerkkikatkelma kuvasarjasta (sivu 2/3)

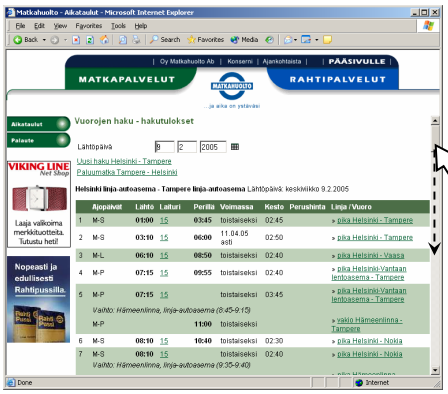


Tässä käyttäjä on syöttänyt ympyröityihin kenttiin *"helsinki"* ja *"tampere"* ja vaihtanut päivämäärän. (Tähän kuvaan voit merkitä myös *Hae*-painikkeen, koska ohjelma ei reagoi ennen sitä. Yhtä hyvin voit merkitä *Hae*-painikkeen vasta seuraavaan kuvaan.)

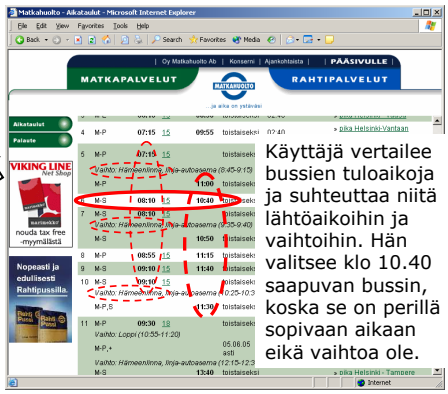


Kun käyttäjä on edellisessä kuvassa painanut *Hae*-painiketta, näytölle ilmestyy yllä näkyvä punainen teksti ja pari pudotusvalikkoa. Seuraavaksi käyttäjä painaa uudelleen *Hae*-painiketta.

Esimerkkikatkelma kuvasarjasta (sivu 3/3)



Seuraavaksi käyttäjä vierittää sivua alaspäin nähdäkseen ennen puoltapäivää perillä olevat bussit. Merkitse **raahaus nuolikursorilla ja katkoviivalla**, joka kuvaa raahaamisen suuntaa ja pituutta.



Käyttäjä vertailee bussien tuloaikoja ja suhteuttaa niitä lähtöaikoihin ja vaihtoihin. Hän valitsee klo 10.40 saapuvan bussin, koska se on perillä sopivaan aikaan eikä vaihtoa ole.

Tässä käyttäjä vertailee eri busseja ja yrittää päättää, millä bussilla hänen kannattaisi mennä. Merkitse tärkeimmät **vertailussa vaikuttavat tiedot** kuvaan **punaisella katkoviivalla**. **Kirjoita viereen tekstillä**, mitä käyttäjä vertailee, minkä hän valitsee ja miksi. Merkitse **valittu bussi** punaisella ympyröinnillä.

Käyttösekvenssit kuvasarjoina
Vertailumerkinnöillä valintapäätökset

- Vertailumerkintöjen tekeminen auttaa löytämään sellaisia käyttöliittymän ongelmakohtia, jotka liittyvät **käyttäjän mielessä** tapahtuvaan vaihtoehtojen vertailemiseen ja päätöksentekoon.
- Vertailumerkinnöillä ympäröidään ne **yksittäiset tiedon palaset**, joiden perusteella käyttäjä päättelee, että kyseinen vaihtoehto on hyvä tai huono. Älä ympäröi kaikkea dataa, jota käyttäjä voisi näytöltä lukea tai katsoa, vaan ainoastaan ne datapalaset, jotka *vaikuttavat* juuri tässä tapauksessa juuri tämän käyttäjän valintapäätökseen.

Käyttösekvenssit skenaarioina
Käyttöä kuvaavat skenaariot

- Kuvasarjan sijaan simulointisekvenssi voidaan kirjata talteen myös tekstimuotoisena skenaariona. Tällöin itse testaukseen tarvitaan aina skenaarion lisäksi alkuperäinen ohjelma, jolla arvioija voi suorittaa skenaariossa kuvatut toimenpiteet.
- Nykyisen järjestelmän käyttöä kuvaava *tehtäväskenaario* (*task scenario* [Hackos98, s. 322-324]) voidaan kirjoittaa erilaisilla yksityiskohtaisuuden tasoilla [Hackos98, s. 326-329]:
  - *complete task scenario* (eniten yksityiskohtia, käyttäjän yksittäiset toimenpiteet näkyvissä)
  - *elaborated scenario* ja
  - *vignette* (vähiten yksityiskohtia)
- Käyttöä kuvaavaan skenaarioon on hyödyllistä lisätä näyttökuvia testitehtävän suorituksen varrelta, kuten lähteessä [Cato01, s. 224-237] on tehty käytettävyystestin tuloksia raportoitaessa. Tällöin lähestytään jo kuvasarjan laatimista.

Simulointituloksina käyttöliittymäongelmia

Tähän osioon on kerätty esimerkkejä simulointitestauksen tyypillisesti tuottamista käyttöliittymäongelmista. Nämä luettelot on tarkoitettu simulointitestauksen opiskeluvaiheen avuksi, jotta arvioijan olisi helpompaa päästä jyvälle siitä, miten simulointitestaus tuottaa ongelmahavaintoja ja millaisia löytyneet ongelmakohdat voivat olla.

Simulointituloksina käliongelmia
Ensin läpimeno, sitten käytettävyys

- Ensin testataan, meneekö tehtävän suorittaminen läpi:
  - Aukot tietosisällössä
  - Aukot toiminnoissa
- Jos tehtävän suoritus menee läpi, arvioidaan käytettävyyttä:
  - Tehokkuusongelmat (käyttäjältä vaadittu turha työ)
    - Turhat toimenpiteet
    - Turha mentaalityö
  - Opittavuusongelmat (käyttäjä ei keksi)
  - (Opittavuusongelma voi aina olla myös muistettavuusongelma)

Simulointituloksina käliongelmia
Aukot tietosisällössä ja toiminnoissa

Kun arvioija on selvittänyt optimaalisen lopputilan, hän arvioi käyttötilanteen läpimenoa muodostamalla simulointisekvenssin:

- Onko tavoite **ylipäätään mahdollista saavuttaa** järjestelmän avulla, vai puuttuuko tarvittavia tietoja tai toimintoja:
  - Tehtävän suorittamisessa tarvittavan tiedon esille kaivamiseen tarvitaan ulkoista lähdettä, kuten toista ohjelmaa tai paperilähdettä, tai muuten suoritus ei etene (esim. Stockmannin myymälän katuosoitteen selvittäminen Reittiopasta varten Googlella; potilaan edellisten hoitotietojen etsiminen arkistomapista).
  - Tavoitetta ei voi saavuttaa loppuun asti järjestelmän avulla, vaan suoritus katkeaa kesken kokonaan tai epäoptimaalisesta kohdasta (esim. hotellihuoneiden hinnat ja varaustilanteen voi selvittää webistä, mutta huoneen varaaminen on tehtävä puhelimella tai menemällä paikan päälle; ajo-ohjetta hotelliin ei ole lainkaan).
  - Tehtävän suorittamisessa tarvitaan ulkoista apuvälinettä, kuten kynää ja paperia tai itse tehtyä Excel-taulukkoa.
  - Tehtävää ei voi tehdä ollenkaan, ei edes osittain.

Simulointituloksina käliongelmia
Tehokkuusongelmat (turha työ)

- Käytön tehokkuutta heikentävät **turhat toimenpiteet**, esimerkkejä:
  - Järjestelmä pakottaa käyttämään käsillä olevan tilanteen kannalta turhia toimintoja tai pakottaa navigoimaan turhan mutkan kautta.
  - Toimintoja on käynnisteltävä vuorotellen eri paikoista, mikä aiheuttaa edestakaisen navigoinnin kahden tai useamman näytön välillä.
  - Käyttäjän on syötettävä sama tieto kahteen tai useampaan kertaan eri paikkoihin taikka tehtävä samat säädöt monta kertaa, kun ohjelma unohtaa ne välillä.
  - Käyttäjän on syötettävä käsin sellainen tieto, jonka järjestelmä voisi hakea automaattisesti toisesta järjestelmästä.

Simulointituloksina käliongelmia
Tehokkuusongelmat (turha mentaalityö)

- Käytön tehokkuutta heikentävä **turha mentaalityö** eli ’miettimistyö’, esimerkkejä:
  - Käyttäjän on pideltävä mielessään vertailtavia vaihtoehtoja, koska vaihtoehdot on esitetty käyttöliittymässä eri paikoissa eikä niitä saa samanaikaisesti näkyville.
  - Kun käyttäjä löytää itselleen sopivia vaihtoehtoja (esimerkiksi sopivia hotellivaihtoehtoja Tukholmasta), hänen on yritettävä pitää löytämänsä hyvät vaihtoehdot mielessään sen sijaan, että ohjelma muistaisi ne.
  - Käyttäjän pitää laskea päässä jotakin sen perusteella, mitä näytöllä näkyy (esim. elokuvan päättymisajan laskeminen lisäämällä kesto alkamisaikaan tai ensi viikon torstain päivämäärän laskeminen näytöllä näkyvän tämän päivän päiväyksen perusteella).

Simulointituloksina käliongelmia
Opittavuusongelmat (vaikea keksiä)

- Mitkä toimenpiteet tai niiden seuraukset ovat **vaikeita keksiä tai ymmärtää**? Esimerkkejä:
  - Toimenpiteitä, joita on vaikea keksiä:
    - Kohde, johon liittyvät toimenpiteet eivät ole näkyvissä, esim. kuva tai tekstinpätkä, josta ei voi tajuta, että sitä voisi klikata.
    - Toimintopainikkeen tai linkin harhaanjohtava otsikko, joka...
      - ei vaikuta oikealta, vaikka todellisuudessa on se oikea, tai
      - vaikuttaa houkuttavalta, mutta johtaa väärään paikkaan.
    - Käyttöliittymän komponentin sisältöä huonosti kuvaava otsikko: käyttäjä ei keksi, mitä esimerkiksi syötekenttään pitäisi kirjoittaa ja missä muodossa.
  - Toimenpiteiden seurauksia, joita on vaikea ymmärtää:
    - Ohjelman antama vaikeaselkoinen palaute, josta käyttäjä ei ymmärrä, mitä ohjelma teki, esimerkiksi lähtikö viesti nyt vai ei.
    - Huono virheilmoitus, josta käyttäjä ei ymmärrä, mitä tapahtui tai mikä on ongelmana tai miten ongelmaa pääsee korjaamaan.

Käytettävyystestaus

Käytettävyystestissä (usability test) testikäyttäjille annetaan oikeita työtehtäviä, joista he yrittävät suoriutua järjestelmän avulla. Näin saadaan selville ensisijaisesti käytettävyysongelmia.

Kirjallisuutta käytettävyystestauksesta: [Nielsen93, luku 6], [Rubin94], [Lauesen05, luvut 1.4 ja 13]

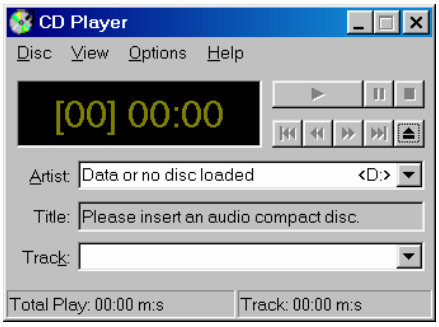
Kuva:
Käytettävyystestausta Helsingin yliopiston tietojenkäsittelytieteen laitoksen *Käyttöliittymät*-kurssilla keväällä 1999.

Käytettävyystestaus
Menetelmä

- Käytettävyystestissä jäljitellään todellista työntekotilannetta: testikäyttäjä etsii sopivaa suoritustapaa, jolla hän saisi testitehtävässä kuvatun tilanteen hoidettua.
  - Testikäyttäjät valitaan järjestelmän **todellisesta kohderyhmästä**.
  - Testin ohjaaja antaa testikäyttäjälle **oikeita käyttötilanteita** vastaavia testitehtäviä.
  - Testikäyttäjä suorittaa tehtäviä itsenäisesti parhaaksi katsomallaan tavalla **ilman testin ohjaajan apua**.
- Testin ohjaaja tai joku toinen testiä seuraava henkilö tekee havaintoja käyttäjän toimenpiteistä ja ajatuksista sekä tulkitsee niistä käyttöliittymän ongelmakohtia.

Käytettävyystestaus
Demo: CD-soittimen testaus

- Selvitetään parin testikäyttäjän avulla, kuinka hyvin CD-soitinohjelma selviytyy muutamasta testitehtävästä.



Testitehtävät
Tehtävät todellisia käyttötilanteita

- Käytettävyystestin testitehtävät vastaavat pitkälti simulointitestauksen testitapauksia, vaikka ne kirjoitetaankin hieman erilaiseen muotoon.
- Testitehtävät ovat realistisia käyttötilanteita, joita järjestelmän oikeille käyttäjille tulee eteen heidän työssään [Rubin94, s. 179].
- Käyttäjälle annetaan testitehtävässä vain tavoite (lähtötilanne). Tehtävänkuvauksessa ei pidä antaa minkäänlaisia vihjeitä toiminnoista, joiden avulla tehtävää suoritetaan [Rubin94, s. 180].

Testitehtävät
Tehtävien valinta ja muu aineisto

- Testitehtävien tulisi koostua käyttäjien yleisimmistä tavoitteista ja käyttötilanteista [Rubin94, s. 102].
- Testi kannattaa aloittaa lyhyellä ja suoraviivaisella tehtävällä, jotta käyttäjä kokee saavansa jotain tehdyksi ja pääsee hyvin alkuun [Nielsen93, s. 186]. Monivaiheiset ja hankalat tehtävät kannattaa sijoittaa testin loppuosaan, jotta niitä on mahdollista pudottaa pois, jos aika ei riitä.
- Testitehtävien yhteydessä tulee käyttää todellisuutta vastaavaa aineistoa (dokumentteja, taulukkoja ym.). Epärealistisen yksinkertaisella aineistolla tehty testi ei paljasta kaikkia ongelmia, joita esimerkiksi hyvin laajan aineiston käsittelyssä voi olla.

Testitehtävät
Esim: VR:n junalippuautomaatti 1/2

Hyödyllinen tehtävä:

Olet pääsiäislomaa edeltävänä perjantaina Helsingin rautatieasemalla. Ensi keskiviikkona aiot lähteä pääsiäisen viettoon vanhempiesi luokse Pieksämäelle viimeisen luentosi jälkeen, joka päättyy klo 16 keskustassa Porthaniassa. Ajattelit hankkia itsellesi junalipun pääsiäistä varten jo nyt, koska tiedät junien olevan silloin usein täynnä.

Ongelmallisia tehtäviä:

Selvitä, mihin aikaan Pieksämäelle menee junia pääsiäislomien aikoihin. Selvitä myös hinnat sekä opiskelijalipuista että normaalihintaisista aikuisten ja lasten lipuista. Hanki lopuksi lippu johonkin haluamaasi junaan.

Selvitä, montako junaa menee arkisin Helsingistä Pieksämäelle. Kuinka monta niistä on InterCity-junia?

Testitehtävät
Esim: VR:n junalippuautomaatti 2/2

Hyödyllinen tehtävä:

Asut Riihimäellä, mutta olet ollut tänään käymässä Helsingissä. Tapasit kavereitasi ja kävitte ostoksilla. Nyt kello on 18.45, kun saavut väsyneenä Helsingin rautatieasemalle. Tiedät, että junia Riihimäelle menee tosi usein, ja ajattelit mennä kotiin seuraavalla sopivalla junalla. Hanki itsellesi junalippu.

Ongelmallisia tehtäviä:

Osta junalippu tänään klo 19.04 Helsingistä Riihimäelle lähtevään junaan.

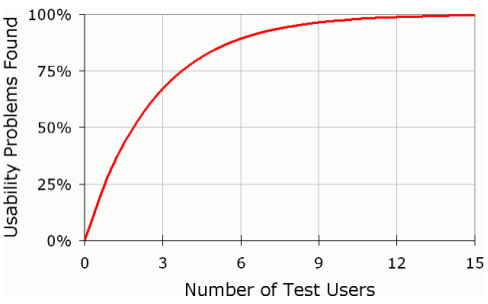
Osta opiskelijalippu ensimmäiseen klo 18.45 jälkeen Helsingistä Riihimäelle lähtevään junaan.

Selvitä, mikä on nopein juna Helsingistä Riihimäelle arkisin ma-pe. Mitkä ovat lippuhinnat nopeimpaan junaan?

Testikäyttäjät
Käyttäjät oikeasta kohderyhmästä

- Testikäyttäjät on tärkeää valita testattavan tuotteen todellisesta kohderyhmästä ([Rubin94, s. 119], [Nielsen93, s. 169]). Väärän kohderyhmän käyttäjillä testaaminen voi johtaa siihen, ettei todellisen käytön kannalta keskeisimpiä ongelmia saada selville.
  - Softan kehittäjän 25-vuotias nörttikaveri voi nauttien selviytyä käsittämättömästä toimintoviidakosta, jossa "luodaan hierarkkisia profiileja" ja "säädetään IMAP-asetuksia", mutta tuotteen ensisijaiseen kohderyhmään kuuluva käyttäjä ei välttämättä onnistu lukemaan sähköpostejaan ollenkaan.
  - Kohderyhmään kuulumattomalta testikäyttäjältä voi puuttua työn tekemiseen tarvittavia tietoja ja taitoja. Esimerkiksi kuva-arkiston käytettävyystesteihin tarvitaan testikäyttäjiksi kuvatoimittajia, jotka tietävät, millä perusteilla kuvia todellisessa työssä lehteen valitaan.

Testikäyttäjät
Käyttäjien lukumäärä



Lähde: [Nielsen00]

Jos testaat yhtä käyttöliittymää, 3-5 käyttäjää on yleensä riittävä määrä.

Mitä useammalla käyttäjällä testaat, sitä useammin samat ongelmat toistuvat. Testaukseen käytetyt lisäresurssit tuovat enää vain vähän lisää hyötyä.

Muut testivalmistelut
Testitila ja pilottitesti

- Erillistä käytettävyyslaboratoriota ei tarvita, vaan testi voidaan järjestää esim. tavallisessa työhuoneessa [Nielsen93, s. 200].
- Testihuone lavastetaan oleellisilta osin todellista käyttöympäristöä vastaavaksi. Esimerkiksi matkatoimistossa käytettävän järjestelmän testauksessa voisi olla hyödyllistä lavastaa jotakin seuraavanlaista:
  - asiakaspalvelutiski ja asiakkaiden käyntejä,
  - puhelin ja asiakkaiden puhelinsoittoja ja
  - matkatoimistossa käytettäviä paperiesitteitä ja muuta materiaalia.
- Ennen varsinaisia testejä kannattaa järjestää ns. **pilottitesti** eli harjoittelutesti, jonka tuloksia ei oteta huomioon varsinaisissa testituloksissa. Pilottitestin tarkoituksena on harjoitella testin läpivientiä ja korjata jäljellä olevat ongelmat testitehtävistä, testin ohjauksesta ja testattavan prototyypin toiminnasta [Nielsen93, s. 174-175].

Testitilanteen ohjaaminen
Ennen testin aloittamista

- Yritä saada testin alku rennoksi ja luontevaksi. Käyttäjät ovat usein hermostuneita aloittaessaan testin, kun he eivät tiedä, mitä tuleman pitää. Usein he kokevat myös valtavia suorituspaineita, joita tarkkailun kohteena oleminen vielä lisää entisestään [Nielsen93, s. 181].
- Kerro testikäyttäjälle ennen testin aloittamista seuraavat toimintaperiaatteet [Nielsen93, s. 188; Rubin94, s. 148-149]:
  - ***Emme testaa sinua**, vaan tätä ohjelmaa. Tarkoituksena on löytää ohjelmasta niin paljon ongelmia kuin mahdollista.*
  - *Jos haluat, **voit lopettaa** testin milloin tahansa.*
  - *Jos testi videokuvataan: Emme ole kuvaamassa sinua, vaan kamera on sitä varten, jotta muistaisimme jälkeenpäin testitilanteessa esille tulleet asiat.*
  - *Pyytäisin **ajattelemaan ääneen**, mitä olet tekemässä [Nielsen93, s. 195].*

Testitilanteen ohjaaminen
Testitilanteen alussa

- Kerro käyttäjälle lyhyesti, millaisesta ohjelmasta testissä on kyse, ja kuvaile testitehtävien taustatilanne lyhyesti:
  - Järjestelmän tarkoitus noin yhdellä lauseella, esim. *"Tällä ohjelmalla käsitellään kodinkoneliikkeeseen tulevia asiakasvalituksia."*
  - Kuka käyttäjä on ja missä hän kuvitellusti on, esim. *"Olet juuri tullut ensimmäisenä työpäivänäsi kodinkoneliikkeen valitustiskille töihin. Jotkut asiakkaasi tulevat tähän tiskille esittämään valituksensa ja osa soittaa sinulle puhelimella."*
  - Jos testiympäristöön kuuluu esim. puhelin, kerro siitä käyttäjälle ennen testin alkua, esim. *"Tuo puhelin on tässä testissä oma työpuhelimesi. Jos se soi, voit vastata siihen omalla nimelläsi."*
- Tarkoituksena on saada käyttäjän sovellusalueeseen ja testiasetelmaan (mutta ei itse käyttöliittymään!) liittyvä tietämys todellista tilannetta vastaavaksi.
- Järjestelmän toimintaa ei pidä esitellä käyttäjälle etukäteen, jos käyttäjä ei tositilanteessakaan saisi koulutusta ennen käyttöä.

Testitilanteen ohjaaminen
Testitehtävien antaminen käyttäjälle

- Anna testitehtävät käyttäjälle yksi kerrallaan.
- Kuvaile testitehtävän sisältämä esimerkkitilanne käyttäjälle suullisesti vapaamuotoisesti.
- Yleensä testitehtävä kannattaa antaa käyttäjälle myös paperilapulla [Nielsen93, s. 186], jottei hänen tarvitsisi pitää mielessään tehtävässä annettuja tietoja.
- Siirry tehtävästä toiseen huomaamattomasti kuvailemalla uutta tilannetta. Älä kommentoi käyttäjän suoriutumista tehtävästä [Nielsen93, s. 190], koska käyttäjän kykyjä ei olla nyt testaamassa.

Testitilanteen ohjaaminen
Ulkopuoliset tarkkailijat

- Testitilassa voi olla testikäyttäjän ja testin ohjaajan lisäksi muita henkilöitä (esim. 1-2 henkilöä tarkkailemassa tai tekemässä muistiinpanoja), mutta he eivät saa häiritä testiä tai osallistua testin kulkuun [Nielsen93, s. 190].
- Yleensä ulkopuolisten tarkkailijoiden on parempi seurata testiä testihuoneen ulkopuolella esim. TV-ruudulta tai yksisuuntaisen peilin läpi (jos testi järjestetään käytettävyyslaboratoriossa).

Testitilanteen ohjaaminen
Neuvomisen välttäminen

- Älä neuvo tai auta käyttäjää tehtävän tekemisessä äläkä anna minkäänlaisia vihjeitä tehtävän suorittamisesta [Nielsen93, s. 183; Rubin94, s. 216, 222].
  - Jos käyttäjä esimerkiksi keksii **uusia tai yllättäviä toimintatapoja**, anna hänen tehdä toimenpiteet itse loppuun asti.
    - Älä 'vedätä' tai johdattele ennalta suunniteltuun suoritustapaan.
    - Älä näytä yllättyneeltä tai ala hämmästellä tilannetta, vaikka käyttäjä tekisi jotain täysin odottamatonta [Rubin94, s. 220].
  - Jos käyttäjä kysyy sinulta suoraan ohjelman toiminnasta, esim. "Tulostaako tämä, jos painan nyt tästä?", esitä **vastakysymys** [Nielsen93, s. 196], esimerkiksi "Mitä arvelisit? Miltä se sinusta vaikuttaa? Voit kokeilla vapaasti kaikkea."
- Testin ohjaajalla on monesti suuri houkutus käyttäjän neuvomiseen testin aikana, mutta neuvominen pilaa helposti testin tulokset (vrt. [Nielsen93, s. 183]).

Testitulokset
Ongelmien havaitseminen

- Käyttöliittymäongelmia löytyy hyvin, kun testaaja lähtee siitä oletuksesta, että testikäyttäjä toimii aina järkevästi. Kaikki testin aikana havaitut käyttäjän virheet ja ongelmat tehtävissä johtuvat lähtökohtaisesti **käyttöliittymän** ongelmista ja puutteista, eivätkä käyttäjästä. Tavoitteena on löytää tuotteesta syy käyttäjän ongelmille [Rubin94, s. 276].
- Käyttäjän kohtaamien ongelmien syitä eli käyttöliittymän vikoja voidaan päätellä
  - käyttäjän tekemistä **toimenpiteistä,**
  - käyttäjän **reaktioista** ja
  - **ääneenajattelusta** [Nielsen93, s. 195].

Testitulokset
Ongelmien havainnointi

- Mahdollisia oireita käyttöliittymän ongelmista:
  - Käyttäjä ei onnistu tekemään testitehtävää.
  - Käyttäjä luulee tehneensä tehtävän kokonaan, mutta todellisuudessa suoritus on keskeneräinen tai lopputulos on väärä.
  - Käyttäjä tekee suorituksessa virheen, kuten syöttää dataa väärään kenttään [Rubin94, s. 276], esim. etunimen sukunimikenttään, tai hävittää erehdyksessä aiemmin luomaansa dataa, esim. unohtaa tallentaa tiedoston.
  - Käyttäjä tekee epäoptimaalisia toimenpiteitä, jotka poikkeavat odotetusta oikeasta suoritustavasta [vrt. Rubin94, s. 276].
  - Käyttäjä käynnistää turhaan väärän toiminnon, avaa väärän ikkunan tai menee väärälle sivulle, mutta huomaa virheensä saman tien ja korjaa sen (esim. sulkee ikkunan, painaa *Back*).
  - Käyttäjä epäröi pitkään jossakin vaiheessa tehtävän suoritusta, mutta suorittaa lopulta tehtävän oikein. (Pitkän epäröinnin aikana käyttäjää kannattaa muistuttaa ääneenajattelusta.)
  - Käyttäjä valittaa ääneen jostakin ongelmasta.

Testitulokset
Ongelmien havainnointi ja muistiinpanot

- **Kevyt tapa:** Jos käytettävyystestaukseen on vain vähän aikaa,
  - pyydä käytettävyystestausta osaava kaveri joko vetämään testiä tai tekemään muistiinpanoja (ja ota itse näistä toinen rooli), ja
  - huolehdi siitä, että muistiinpanojen tekijä osaa katsoa testissä ilmenevät ongelmakohdat ja merkitsee ne muistiin testin kuluessa.
- **Perusteellisempi tapa:** Jos tarvitset luotettavampia ja täsmällisempiä tuloksia tai jos olet epävarma omista taidoistasi ongelmakohtien tulkitsemisessa, testin aikana kannattaa ottaa talteen käyttäjän täsmälliset toimenpiteet (käyttösekvenssi) ja keskeisimmät suulliset kommentit (ääneenajattelu).
  - Käyttäjän toimenpiteistä tai ääneenajattelusta on mahdollista löytää ja analysoida jälkikäteen sellaisiakin käyttöliittymän ongelmia, joita on vaikea nähdä testin aikana suoraan.
  - Muistiin kirjatuista käyttäjän toimenpidesekvensseistä saa vielä jälkikäteen selville, mihin löydetyt ongelmat perustuivat ja olivatko testin aikana ongelmista tehdyt tulkinnat oikeita.

Testitulokset
Käyttösekvenssin taltioiminen 1/2

- Jos tallennat testistä käyttösekvenssin:
  - Yritä saada toimenpiteet talteen tekemällä heti testin aikana käsin muistiinpanoja.
    - Käyttösekvenssin voi kirjoittaa muistiin yksinkin, mutta sitä on kuormittavaa tehdä testin ohjaamisen ohella.
    - Pyydä kaveria avuksi ohjaamaan testiä tai tekemään muistiinpanoja. Hyvien muistiinpanojen tekeminen on vaativampaa kuin testin ohjaus.
    - Videokamera on hyvä lähinnä varajärjestelynä: videonauhalta voit jälkeenpäin tarkistaa käyttäjän tekemisiä tai puheita, jos et ehtinyt nähdä, mitä käyttäjä teki jossakin tärkeässä kohdassa.
  - Merkitse muistiin ensisijaisesti se, mitä käyttäjä oikeasti teki (todelliset tapahtumat). Testin aikana tehdyt tulkinnat voivat osoittautua myöhemmin virheellisiksi tai hyödyttömiksi. Jos alkuperäistä tapahtumadataa ei ole tallessa, tulkintoja ei voi myöhemmin muuttaa vastaamaan testissä ilmennyttä uutta dataa.
  - Ks. Lauesenin esimerkki käytettävyystestin muistiinpanoista [Lauesen05, s. 440].

Testitulokset
Käyttösekvenssin taltioiminen 2/2

Esimerkki YTV:n Reittioppaan käytettävyystestin muistiinpanoista:

- Ote käyttösekvenssistä ja käyttäjän ääneenajattelusta:
  …Käyttäjä painoi Hae. Avasi Mistä-pudotuslistan, jossa annettu kaksi eri Unikkotietä. *"Ahaa, kaks Unikkotietä, Hesassa ja Vantaalla. Mä vaihdan tuoksi Vantaan Unikkotieksi."* Valitsi Vantaan Unikkotien, klikkasi Vaihda, järjestelmä tyhjensi kentän. *"Mitä ihmettä, mitä nyt tapahtu?"* Kirjoittaa uudelleen Unikkotie 6, painaa Hae, valitsee Vantaan Unikkotien, klikkaa Hae.

Testitulokset
Harjoitus: Silja Line

- Tässä harjoituksessa käytettävyystestataan Silja Linen web-sivustoa, tehdään muistiinpanoja käyttäjän toimenpiteistä ja kerätään testituloksiksi käyttöliittymän ongelmakohtia.

Testitulokset
Ongelmien vakavuuden arviointi

1. **Puuttuva toiminnallisuus**: Käyttöliittymällä ei pysty suorittamaan testitehtävää. Esim: *Käyttäjä yritti tulostaa alennuskoodit tehdäkseen niihin omia merkintöjään, mutta järjestelmästä ei voi tulostaa koodeja.*
2. **Epäonnistunut tehtävän suoritus**: Testikäyttäjä ei pystynyt suorittamaan tehtävää itsenäisesti tai luuli (virheellisesti) tehneensä tehtävän jo. Esim: *Käyttäjä luuli tehneensä tehtävän ja tuloksen olevan tallessa, mutta hänen olisi pitänyt vielä painaa Update-painiketta.*
3. **Selvästi häiritsevä ongelma**: Käyttäjä ei osannut toimia optimaalisella tavalla tai hän valitti suullisesti järjestelmän olevan hankala tai kömpelö. Esim: *Käyttäjä sanoi, että on täysin älytöntä kulkea kuuden näytön läpi, jotta saisi täytettyä kymmenen kenttää.*
4. **Melko häiritsevä ongelma**: Käyttäjä sai tehtävän suoritettua pitkällisten yritysten jälkeen. Esim: *Käyttäjä yritti haun käynnistämistä monella eri tavalla ennen kuin keksi näytöltä opasteen ja painoi F10.*
5. **Vähäinen ongelma**: Käyttäjä löysi ratkaisun muutaman lyhyen yrityksen jälkeen. Esim: *Käyttäjä yritti käynnistää haun ensin enterillä, mutta huomasi sitten painaa F10-näppäintä.*

[Lauesen05, s. 12-13]

Yhteenveto
Kustannustehokas testaus

- Videoidut käytettävyystestit suurilla käyttäjämäärillä ovat kalliita ja vievät paljon aikaa. Muu projekti voi ehtiä edetä jo pitkälle, kun testejä vasta analysoidaan.
- Monesti yhden laajan testin sijaan kannattaa kerätä nopeasti tietoa pahimmista ongelmista järjestämällä monta kevyttä ja edullista testiä. Testien välillä korjataan pahimmat ongelmat.
  - Järjestä testejä esim. tavallisessa työhuoneessa tai neukkarissa.
  - Valitse 3-4 **todellista käyttäjää**, mutta älä koodaaja-työkavereita.
  - Laadi testitehtäviksi **aitoja tilannekuvauksia**.
  - Tee ongelmakohdista **muistiinpanot testin kuluessa**, ja pura muistiinpanot heti testin jälkeen. Voit hoitaa testit yksinkin, mutta jos mahdollista, pyydä työkaveria avuksi joko muistiinpanojen tekijäksi tai testin ohjaajaksi.

Sisältö

Käyttöliittymän suunnitteluprosesseja

Käyttöliittymän systemaattista tuottamista on toistaiseksi tutkittu varsin vähän. Esimerkkejä systemaattisista suunnitteluprosesseista ovat mm. Lauesenin ja Cooperin prosessit sekä Carrollin skenaariopohjainen suunnittelu. Tällä kurssilla käsitellään yksityiskohtaisemmin GDD-suunnitteluprosessia, jota on kehitetty mm. tietojenkäsittelytieteen laitoksella ja jossa on yhtymäkohtia kaikkien edellä mainittujen prosessien kanssa.

Suunnitteluprosessit
Lauesen, Carroll ja Cooper

- Hyvien käyttöliittymäratkaisujen syntymistä ja systemaattista luomista on tutkittu todella vähän. Tällä hetkellä kirjallisuudessa kuvattuja menetelmiä ovat lähinnä seuraavat:
  - Søren Lauesenin järjestelmällisesti etenevässä *Virtual Windows* -menetelmässä [Lauesen01, Lauesen05] käyttöliittymä suunnitellaan ensin pelkän tietosisällön avulla ja toiminnallisuus lisätään näyttökuviin vasta suunnittelun loppuvaiheessa.
  - *Scenario-Based Design* [Carroll00] -menetelmässä käyttöliittymää suunnitellaan kirjoittamalla tekstimuotoisia skenaarioita eli tarinoita siitä, miten kuviteltu käyttäjä toimisi järjestelmän kanssa.
  - Alan Cooperin *Goal-Directed Design* [Cooper03] –menetelmässä kuvataan tulevia käyttäjiä ja heidän korkean tason tavoitteitaan. Sen jälkeen näiden käyttäjien näkökulmasta kirjoitetaan tekstimuotoisia käyttöskenaarioita, joiden sisältämät tiedot ja toiminnot lopulta organisoidaan käyttöliittymän näkymiksi ja komponenteiksi.

Suunnitteluprosessit
Lauesenin *Virtual Windows*

- Lauesenin *Virtual Windows* –menetelmässä käyttöliittymä suunnitellaan ensin pelkän tietosisällön avulla, ja toiminnot lisätään vasta suunnittelun loppuvaiheessa.
  - Ensin selvitetään käyttäjän tehtävät (*tasks*), jotka järjestelmällä pitäisi voida tehdä, ja niistä kirjoitetaan tekstimuotoiset kuvaukset.
  - Sitten tietosisältö jaetaan ns. virtuaali-ikkunoihin simuloimalla käyttäjän tehtäviä ja soveltamalla Lauesenin antamia suunnittelusääntöjä. Tietosisältöä voidaan katsoa tässä vaiheessa myös esimerkiksi tietomallin kuvauksesta.
  - Virtuaali-ikkunoista piirretään visualisoituja oikean näköisiä näyttökuvia. Näissä graafisissa virtuaali-ikkunoissa ei kuitenkaan ole vielä lainkaan toimintoja eikä niitä ole sovitettu mahtumaan lopulliseen näyttökokoon (voivat olla liian pieniä tai isoja näytölle).
  - Lopuksi virtuaali-ikkunoihin lisätään toiminnot ja virtuaali-ikkunat sovitetaan lopulliseen näyttökokoon. Tämä tehdään taas simuloimalla käyttäjän tehtäviä.

Suunnitteluprosessit
Carrollin skenaariomenetelmä

- Carrollin *skenaariomenetelmässä* [Carroll00] käyttöliittymää suunnitellaan ensisijaisesti tekstimuodossa, ja siirtyminen tekstiskenaarioista näyttökuviin jää osin avoimeksi.
  - Aluksi tehdään skenaarioiden laatimisen pohjaksi kenttätutkimusta: käyttäjien haastatteluja ja heidän nykyisen toimintansa tarkkailua.
  - Tekstimuotoisten skenaarioiden avulla tehdään rinnan käyttöliittymäsuunnittelua ja työnkulkujen uudelleensuunnittelua. Käyttäjän työnkulun etenemistä ja käyttöliittymän toimintalogiikkaa ideoidaan ja kuvataan skenaarioina, joita kirjoitetaan moneen kertaan (iteratiivinen suunnittelu). Käyttöliittymän tietosisältöä tai toiminnallisuutta ei systemaattisesti visualisoida näyttökuvina, kuten Lauesenin prosessissa, mutta apuna voidaan käyttää kuviakin.
  - Kun skenaarioiden kirjoittamisen yhteydessä tulee esille erilaisia suunnitteluratkaisuja, niitä verrataan ns. väittämillä (*claims*). Väittämiin kirjataan ratkaisuvaihtoehtojen hyvät ja huonot puolet, jotta niitä voidaan verrata keskenään.

Suunnitteluprosessit
Cooperin *Goal-Directed Design*

- Cooperin *Goal-Directed Design* -menetelmässä [Cooper03, s. 16-20, 39-90] kuvataan tulevia käyttäjiä ja heidän tavoitteitaan sekä laaditaan keskeisimmistä tilanteista skenaariokuvauksia.
  - Aluksi tehdään tulevista käyttäjistä ns. persoonakuvauksia mm. käyttäjähaastattelujen ja –tarkkailujen perusteella. Jokaiselle persoonalle selvitetään/luodaan korkean tason tavoitteet.
  - Seuraavaksi laaditaan käyttöskenaarioita (*context scenarios*) hieman vastaavaan tapaan kuin Carrollinkin menetelmässä. Skenaariossa kuvataan käyttäjän kannalta mahdollisimman hyvä toteuma tulevalla järjestelmällä, mutta se myös kiinnittää jo karkealla tasolla uuden järjestelmän toimintoja ja käyttöliittymän suunnitteluratkaisuja.
  - Skenaarioista tulkitaan käyttöliittymässä tarvittavat toiminnot ja tiedot. Sen jälkeen ne organisoidaan käyttöliittymän näkymiksi ja paneeleiksi ”top-down” eli aloittaen karkeista näkymistä ja siirtyen vähitellen kohti näytön yksityiskohtia.

Suunnitteluprosessit
Simulointipohjainen GDD-prosessi

- Simulointipohjaisessa GDD-suunnitteluprosessissa (*Goal-Derived Design*), joka on GUIDe-prosessimallin ydin [Laakso04], käyttöliittymää piirretään näyttökuviksi simuloimalla yhden käyttötilanteen suorittamista kerrallaan.
  - Aluksi selvitetään kenttätutkimusten avulla käyttötilanteet. Nykymenettelyistä erotellaan niiden takana olevat tavoitteet.
  - Simuloitavaksi valitaan yksi käyttötilanne kerrallaan, jota ryhdytään piirtämään auki käyttöliittymäksi (tietosisältö ja toiminnot). Piirtämisen aikana kiinnitetään käyttötilanteelle mahdollisimman hyvä toteuma: sekä käyttöliittymä että työnkulku. Toteuma kiinnitetään näyttökuvien piirtämisen avulla, toisin kuin Carrollin ja Cooperin prosesseissa, joissa se tehdään tekstiskenaarion pohjalta.
  - Kun muutaman käyttötilanteen suorittaminen ja niissä tarvittu toimintalogiikka ja tietosisältö on piirretty samaan käyttöliittymään, jonka näyttökuvat alkavat muuttua sotkuiseksi, kuvista piirretään puhtaaksi siistimmät versiot (ns. refaktorointivaihe).

Suunnitteluprosessin valinta
Lopputulos testataan käyttötilanteilla

- Kaikille suunnittelumenetelmille yhteistä on se, että niiden tuottama lopputulos aina viime kädessä testataan todellisilla käyttötilanteilla, kun järjestelmä otetaan käyttöön ja käyttäjät ryhtyvät ensimmäistä kertaa tekemään sillä todellista työtään.
- Kaikki edellä kuvatut prosessit ottavat suunnittelussa huomioon käyttäjien työtehtävät tai käyttäjille eteen tulevat käyttötilanteet.
- (Käyttäjien työnkulkuja optimoivat GDD-prosessi, Carrollin skenaariomenetelmä sekä Cooperin prosessi.)

Suunnitteluprosessin valinta
Käyttöliittymäratkaisun kiinnitys

- Käyttöliittymäratkaisun kiinnittäminen eri menetelmissä:
  - Carrollin skenaariomenetelmässä ja Cooperin prosessissa käyttöliittymän tietosisältöä ja toimintalogiikkaa kiinnitetään **tekstimuodossa skenaarioiden** avulla.
  - Lauesenin Virtual Windowsissa ja GDD-prosessissa tietosisältö ja toimintalogiikka kiinnitetään **piirtämällä näyttöjä**.
    - Virtual Windowsissa kiinnitetään ensiksi tietosisältö, lopuksi toiminnot.
    - GDD-prosessissa kiinnitetään sekä tietosisältö että toiminnot yhtä aikaa.
- Käyttöskenaarioiden ja näyttökuvien etuna on se, että järjestelmän vaatimuksia voidaan arvioida/testata jo varhain.
  - **Skenaarioiden** avulla käyttäjät saavat paremmin selville todellisia tarpeitaan, koska he voivat simuloida järjestelmän toimintaa mielessään [Go04]. Suunnittelijat voivat arvioida skenaarioiden avulla suunnitteluratkaisuja ja paikantaa ongelmakohtia [Carroll94].
  - **Näyttökuvien** avulla järjestelmän toimintaa ja tietosisältöä voidaan testata varhaisessa vaiheessa (esim. simulointitestaus tai käytettävyystestaus).

Simulointipohjainen GDD-suunnitteluprosessi

GDD-simulointisuunnittelussa (*Goal-Derived Design*) kantavana ajatuksena on käyttää suunnittelussa samaa menettelyä kuin testauksessa ja järjestelmän todellisessa käytössä: käyttäjä ryhtyy hoitamaan todellista käyttötilannetta vaihe vaiheelta.

Tällä kurssilla opetellaan GDD-prosessista yksinkertaistettua versiota, jossa mm. *tavoitepohjaiset käyttötapaukset* on korvattu yksinkertaisemmilla käyttötilanteilla.

Käyttöliittymän piirtäminen

Käyttöliittymä rakentuu pala kerrallaan näyttökuviksi, kun suunnittelija piirtää käyttötilanteessa tarvittavaa tietosisältöä ja toimintoja käyttäjän toimintaa simuloiden.

Käyttöliittymän piirtäminen
Käyttötilanteet vastaavat testitapauksia

- Suunnittelussa tarvittavat käyttötilanteet ovat vastaanlaisia kuin simulointi- tai käytettävyystestauksen testitapauksetkin:
  - Käyttötilanteet ovat realistisia tilanteita, joita järjestelmän oikeille käyttäjille tulee eteen.
  - Käyttötilanteessa kuvataan vain käyttäjän tavoite - ei ohjelman käyttöä tai niitä toimintoja, joiden avulla käyttäjä voisi päästä tavoitteeseensa.
- Jos käyttötilanteessa kuvattaisiin järjestelmän käyttöä, käyttöliittymän suunnittelua olisi tehty osittain jo ennen käyttöliittymän piirtämistä näyttökuviksi.

Käyttöliittymän piirtäminen
Piirrä vaihe vaiheelta näkyviin

- Suunnitteluun otetaan yksi konkreettinen käyttötilanne kerrallaan.
- Valitun käyttötilanteen suorittamista ryhdytään **simuloimaan käyttäjän näkökulmasta**. Simuloinnin jokaisessa vaiheessa tarvittavat toiminnot ja tiedot piirretään näkyviin käyttöliittymään sitä mukaa kuin niiden tarpeeseen törmätään.
- Aloita piirtäminen katsomalla käyttötilannetta käyttäjän näkökulmasta, ja hae lyhintä polkua parhaaseen lopputulokseen.
  - Piirrä näkyviin se, mitä käyttäjälle ensimmäiseksi kannattaisi näyttää ja mitä hänen ensiksi kannattaisi tehdä.
  - Simuloi käyttäjän ensimmäinen toimenpide ja piirrä sen seuraukset käyttöliittymään näkyviin.
  - Jatka samalla tavalla askel askeleelta, kunnes käyttäjä on saavuttanut käyttötilanteensa kannalta parhaan ratkaisun.

GDD-suunnitteluprosessin vaiheet
Käyttötilanne kerrallaan simuloiden

Toimenpide a. Piirrä yksi käyttötilanne kerrallaan dataa ja toimintoja näkyviin 'yhteen suureen ikkunaan'.

Käyttötilanne 1
1.1 Piirrä kt 1 käyttöliittymäratkaisuksi simuloimalla käyttösekvenssiä vaihe vaiheelta.
1.2 Simuloi ilmeisimmät variaatiot.

Käyttötilanne 2
2.1 Editoi kt 2 edelliseen käyttöliittymään mukaan simuloimalla sitä vaihe vaiheelta.
2.2 Simuloi ilmeisimmät variaatiot.
2.3 Testaa käyttöliittymää simuloimalla edellinen käyttötilanne 1.

Käyttötilanne 3
3.1 Editoi kt 3 mukaan simuloimalla.
3.2 Simuloi ilmeisimmät variaatiot.
3.3 Testaa käyttöliittymää simuloimalla edelliset käyttötilanteet 1 ja 2.

Käyttötilanne 4
...

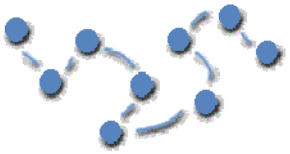
Toimenpide b. Testaa käyttötilanneketjut (vain mielekkäät tositilanteet).
Simuloi esim. ketju 2, 1, 2, 4, 2, 2, 4.
Simuloi esim. ketju 1, 1, 3.

Toimenpide c. Refaktoroi käyttöliittymää tarvittaessa.

Käyttöliittymän piirtäminen
Suoraviivainen käyttöliittymä

- Kun piirrät vaihe vaiheelta näkyviin, mitä käyttäjä näkee ja mitä hän tekee, tavoittele **suoraviivaista** toimintaa.
  - Näytä mahdollisimman **paljon hyödyllistä tietoa kerralla**. Älä toistaiseksi huolehdi siitä, mahtuuko kaikki näytölle vai ei. Käytä riittävän suurta piirtoarkkia, esim. A3.
  - Piirrä aluksi suurin piirtein **kaikki yhteen ikkunaan** (yhdelle sivulle, yhteen näkymään). Älä pakota käyttäjää siirtyilemään paikasta toiseen, vaan anna hänen tehdä tehtävänsä yhdessä paikassa aina kun voit.

Mutkikas ja pitkä toimintoketju



Suoraviivainen toiminta

Käyttöliittymän piirtäminen
Yksi tilanne kerrallaan

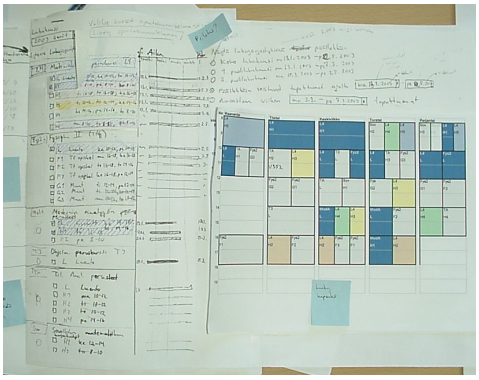
- Pysyttele tiukasti vain **yhden käyttötilanteen** piirtämisessä kerrallaan. Kerää **ongelmalistaa**, jos kesken piirtämisen mieleesi tulee…
  - muita tärkeitä käyttötilanteita, jotka pitäisi huomioida,
  - toimintoja, joita saatetaan tarvita muissa tilanteissa, tai
  - muissa tilanteissa ongelmalliseksi osoittautuvia ratkaisuja.
- Jos suunnittelija kesken piirtämisen huolestuu muiden käyttötilanteiden ongelmista tai ryhtyy miettimään muissa käyttötilanteissa tarvittavia toimintoja, piirtäminen etenee hitaasti ja käyttöliittymästä tulee helposti sellainen, ettei sen avulla olekaan suoraviivaista hoitaa yhtään käyttötilannetta.

Käyttöliittymän piirtäminen
Ongelmalistan käsittely

- Piirtämisen jälkeen käy ongelmalista läpi. Jos listalla on...
  - …**uusi toiminto**, jota arvelet jonkun käyttäjän tarvitsevan: Etsi yksi konkreettinen käyttötilanne, jossa kyseistä toimintoa voitaisiin tarvita. Jos sellaista ei löydy, jätä toiminto sivuun. Jos löytyy, katso nyt hetken ajan pelkästään löytämääsi käyttötilannetta ja piirrä sen tukemiseksi mahdollisimman suoraviivainen ratkaisu. Paras ratkaisu ei välttämättä olekaan alunperin mieleesi tullut toiminto.
  - …**uusia käyttötilanteita**: Kokeile ensin simuloimalla, onnistuuko niidenkin tekeminen käyttöliittymälläsi. Jos ei onnistu, mutta tilannetta olisi syytä tukea, ota uusi käyttötilanne suunnitteluun mukaan.
  - …**käyttöliittymän aiheuttama ongelma** jonkin muun kuin suunnittelussa mukana olevan käyttötilanteen kannalta: Selvitä ensiksi, mikä tämä muu tilanne on. Sen jälkeen arvioi, onko haitta todellinen. Jos sen korjaaminen heikentää muiden käyttötilanteiden suorittamista, haarukoi sopiva kompromissiratkaisu simuloimalla kaikkia ongelmaan liittyviä käyttötilanteita nopeasti peräkkäin.

Käyttöliittymän piirtäminen
Näyttökuvat

Näyttökuvia kannattaa piirtää nopeasti kynällä paperille. Ne osat, joita on nopeampaa tehdä esimerkiksi PowerPointilla kuin käsin piirtämällä, kannattaa tehdä tietokoneella ja tulostaa piirrosten joukkoon.



Esimerkki paperille laaditun näyttökuvan yhdestä sivusta: WWWTopi (kurssi-ilmot ja opintojen suunnittelu)

Demo:
Hampurilaisravintolan tilauspalvelu

Tässä luentodemossa suunnitellaan käyttöliittymää kuvitellun hampurilaisravintolan web-tilauspalvelua varten.

Suunnittelun lähtökohtana käytetään kahta käyttötilannetta.

Suunnittelun eteneminen:

- Valitaan keskeisin käyttötilanne 1. Piirretään tähän tarvittavat tiedot ja toiminnot vaihe vaiheelta näkyviin.
- Valitaan seuraava käyttötilanne 2 ja yritetään suorittaa sitä aiemmalla käyttöliittymällä. Korjaillaan ja lisätään puuttuvat tiedot ja toiminnot.
- Palataan testaamaan käyttötilannetta 1: varmistetaan, että sen tekeminen onnistuu edelleen vaivattomasti.
- Valitaan käyttötilanne 3… jne.

Käyttötilanteet
Hampurilaisravintola: Käyttötilanne 1

Käyttötilanne 1: Heti ruokaa koodaajille

Torstai-iltana nuoret tutkijat Petteri, Ilkka ja Asko ovat nälissään yliopiston tietojenkäsittelytieteen laitoksella Helsingin Vallilassa (Teollisuuskatu 23). He ovat viimeisenä yönä koodaamassa demoprotoa, jonka parissa menee vielä myöhään. Nyt kello on jo 21.15, mutta koodaus on vielä kesken.

Kaikilla on hirveä nälkä muttei mitään syötävää. Ainoastaan limua ja kahvia löytyy laitoksen kahvihuoneesta ja opiskelijahuoneesta. Kaikki tietävät pääkaupunkiseudun hampurilaisvalikoimat hyvin, koska he käyvät usein syömässä hampurilaisia eri paikoissa. Nyt kuitenkaan ei olisi aikaa lähteä minnekään syömään.

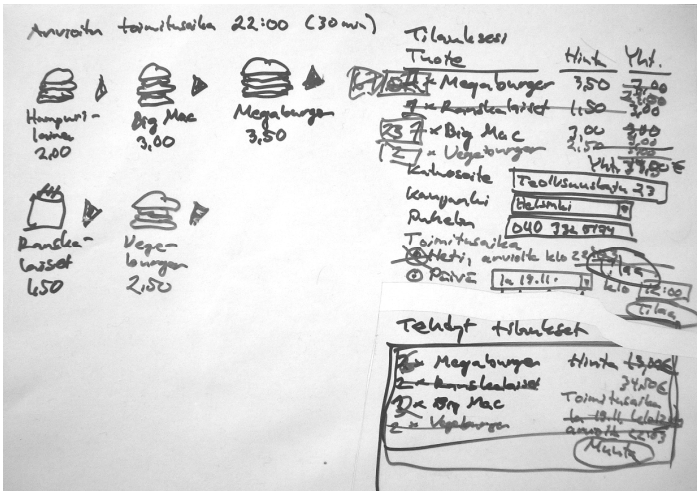
Käyttötilanteet
Hampurilaisravintola: Käyttötilanne 2

Käyttötilanne 2: Tekes-projektipäivän lounas lauantaina

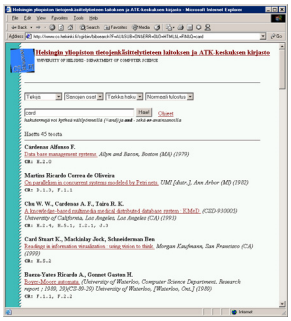
Yliopiston tutkija Petteri pitää ensi lauantaina tietojenkäsittelytieteen laitoksella (Teollisuuskatu 23) Tekes-projektinsa epävirallisen suunnittelupäivän klo 9-15. Nyt on keskiviikko klo 9.45, ja Petteri on työpaikallaan suunnittelemassa lauantain tapahtumaa. Suunnittelupäivään on tulossa hänen itsensä lisäksi 8 projektiin enemmän tai vähemmän löyhästi kuuluvaa työkaveria, ja tilaisuuteen pitäisi saada jotakin lounaaksi kelpaavaa syötävää.

Petteri on miettinyt, että hampurilaiset tai esimerkiksi pizza voisivat olla sopivia vaihtoehtoja tähän tarkoitukseen, koska työkaverit syövät niitä aina muutenkin. Hän tuntee pääkaupunkiseudun hampurilais- ja pizzatarjonnan hyvin, koska hän käy usein syömässä hampurilaisia eri paikoissa.

Käyttöliittymä
Luonnos käyttötilanteista 1 ja 2



Esimerkki:
Kirjastohakujärjestelmä



TKTL:n vanha kirjastohakujärjestelmä

Tässä esimerkissä suunniteltavana käyttöliittymänä on yliopiston tietojenkäsittelytieteen laitoksen (TKTL) kirjastojärjestelmä.

Suunnittelun lähtökohtana on konkreettisia käyttötilanteita, joiden pohjalta käyttöliittymää piirretään GDD-suunnitteluprosessin mukaisesti tilanne kerrallaan.

Kirjastoesimerkki
Käyttötilanne 1

Käyttötilanne 1: Cardin visualisointikirja omasta lähikirjastosta

Maanantaina 1.9. klo 9.30 tutkija Hannu Toivonen on laatimassa *Tutkimustiedonhallinnan peruskurssin* seuraavaa luentoaan, jonka hän pitää ti 9.9. klo 10-12.

Hannu tietää, että Cardin visualisointikirjassa olisi hyviä glyyfiesimerkkejä luennolle, koska hän on aiemminkin lukenut kyseistä kirjaa, mutta hänellä itsellään ei ole sitä. Hän ei myöskään tiedä, onko kirjaa kirjastossa. Hannu muistaa, että kirjan nimi on jokin *'Information Visualization'* ja yksi tekijöistä on Card.

Kirjan todelliset tiedot: Card Stuart, MacKinlay Jock, Shneiderman Ben, *Readings in Information Visualization: Using Vision to Think*.

Todellisuudessa omassa tietojenkäsittelytieteen laitoksen (TKTL) lähikirjastossa kirjoja on 3 kpl, joista 1 on lainassa Inkeri Verkamolla ja 2 hyllyssä.

Käyttötilanne 1 (Cardin kirja lähikirjastosta)

Kirjan haku

Kansi	Tekijä(t)	Kirjan nimi	Vuosi	Hyllyssä
Hae				
	Aho Alfred, Sethi Ravi, Ullman Jeffrey	Compilers: principles, Techniques and tools	1986	6
	Aho Alfred, Kernighan Brian, Weinberger Peter	The AWK Programming Language	1988	
	Alexander, Christopher	The Timeless Way of Building	1979	2
	Beck Kent, Fowler Martin	Extreme Programming Explained: Embrace Change	1999	2
	Beck Kent, Fowler Martin	Planning Extreme Programming	2000	
	Brooks Frederick	The Mythical Man-Month: Essays on Software Engineering	1999	1

Kirjan haku

Kansi	Tekijä(t)	Kirjan nimi	Vuosi	Hyllyssä
Hae	card			
	**Card** Remy, Dumas Eric, Mevel Franck	The Linux Kernel Book	1998	1
	**Card** Stuart, Mackinlay Jock, Shneiderman Ben	Readings in Information Visualization: Using Vision to Think	1999	2
	**Card** Stuart, Moran Thomas, Newell Allen	The Psychology of Human-Computer Interaction	1983	1
	Cardelli Luca	Typeful Programming	1989	
	Cardenas Alfonso	Research Foundations in Object-Oriented and Semantic Database Systems	1990	1
	Iyer Balakrishna, Ri**card** Gray, Varman Peter	An Efficient Percentile Partitioning Algorithm for Parallel Sorting	1989	2

Kirjastoesimerkki
Käyttötilanne 2

Käyttötilanne 2: Kaikki Cardin visualisointikirjat lainassa

Maanantaina 1.9. klo 9.30 tutkija Hannu Toivonen on laatimassa *Tutkimustiedonhallinnan peruskurssin* seuraavaa luentoaan, jonka hän pitää ti 9.9. klo 10-12.

Hannu tietää, että Cardin visualisointikirjassa olisi hyviä glyyfiesimerkkejä luennolle, koska hän on aiemminkin lukenut kyseistä kirjaa, mutta hänellä itsellään ei ole sitä. Hän ei myöskään tiedä, onko kirjaa kirjastossa. Hannu muistaa, että kirjan nimi on jokin *'Information Visualization'* ja yksi tekijöistä on Card.

Kirjan todelliset tiedot: Card Stuart, MacKinlay Jock, Shneiderman Ben, *Readings in Information Visualization: Using Vision to Think*.

Todellisuudessa omassa tietojenkäsittelytieteen laitoksen (TKTL) lähikirjastossa kirjoja on 3 kpl, jotka kaikki ovat lainassa: Esko Ukkosella, Inkeri Verkamolla ja Hannu Erkiöllä.

Ero kt 1:een

Käyttötilanne 2 (kaikki lainassa)

Kirjan haku

Kansi	Tekijä(t)	Kirjan nimi	Vuosi	Hyllyssä	Lainassa Pvm	Lainaaja	Puhelin
Hae	card						
	**Card** Remy, Dumas Eric, Mevel Franck	The Linux Kernel Book	1998	1			
	**Card** Stuart, Mackinlay Jock, Shneiderman Ben	Readings in Information Visualization: Using Vision to Think	1999		18.3.03 31.3.03 15.5.03	Esko Ukkonen Inkeri Verkamo Hannu Erkiö	44 226 44 219 44 253
	**Card** Stuart, Moran Thomas, Newell Allen	The Psychology of Human-Computer Interaction	1983	1			
	Cardelli Luca	Typeful Programming	1989		10.6.03 12.7.03	Petri Myllymäki Mikael Jokela	44 173 44 628
	Cardenas Alfonso	Research Foundations in Object-Oriented and Semantic Database Systems	1990	1			
	Iyer Balakrishna, Ri**card** Gray, Varman Peter	An Efficient Percentile Partitioning Algorithm for Parallel Sorting	1989	2			

Kirjastoesimerkki
Käyttötilanne 3

Käyttötilanne 3: Waren visualisointikirjaa ei kirjastossa

Maanantaina 1.9. klo 9.30 tutkija Hannu Toivonen on laatimassa *Tietokonegrafiikan* kurssille luentojaan. Ensimmäinen luento on ti 16.9. klo 10-12.

Hannu tietää, että Waren kirjassa olisi hyviä esimerkkejä rinnakkaiskoordinaattien käytöstä *Tietokonegrafiikan* kurssille, koska hän on aiemminkin lukenut kyseistä kirjaa. Hänellä itsellään ei kuitenkaan ole kirjasta omaa kappaletta, eikä hän tiedä, onko kirjaa kirjastossa. Hannu muistaa, että kirjan tekijä on Ware ja kirjan nimi on *'Information Visualization'.*

Kirjan todelliset tiedot: Ware Colin, *Information Visualization: Perception for Design*.

Todellisuudessa kirjasta ei ole lainakappaleita kirjastossa ollenkaan.

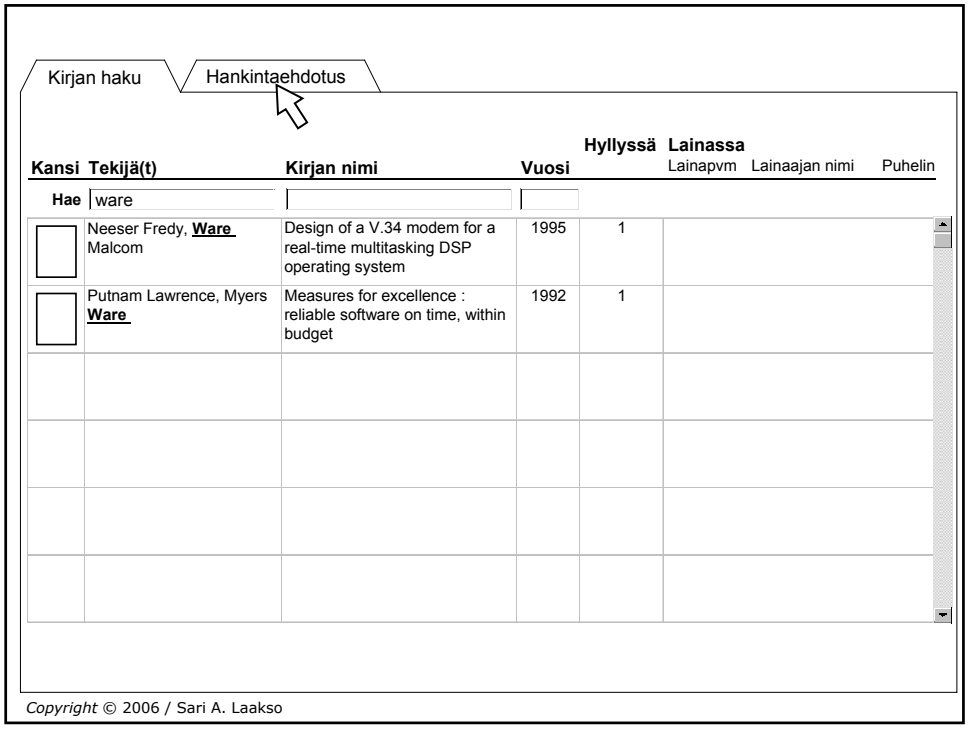
Hannu tietää, että hänen opiskelijansa todennäköisesti käyttävät kirjaa myöhemminkin, koska kirja on mm. hänen *Tietokonegrafiikka*-kurssinsa oheismateriaalina.

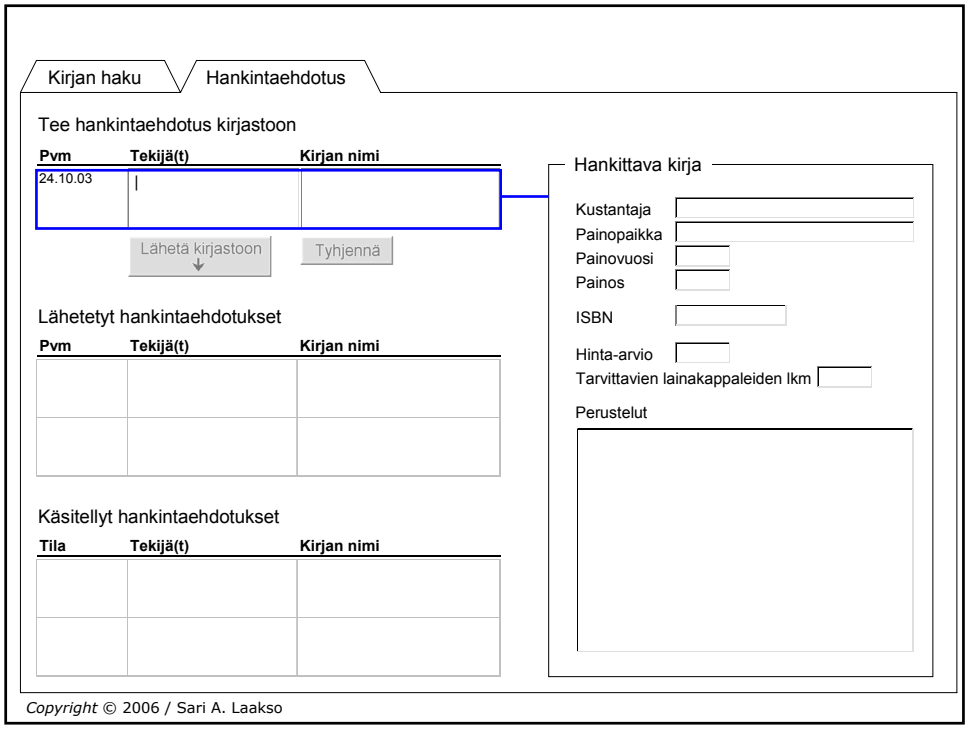
Erot edellisiin tilantei-siin

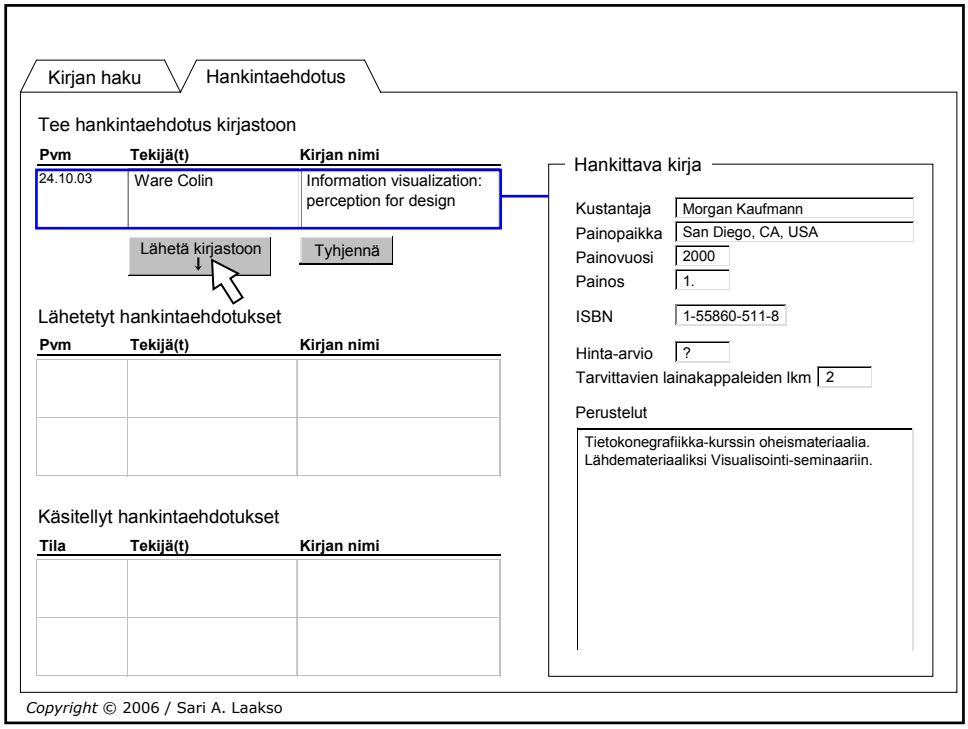
Käyttötilanne 3 (kirjaa ei ole kirjastossa ollenkaan)

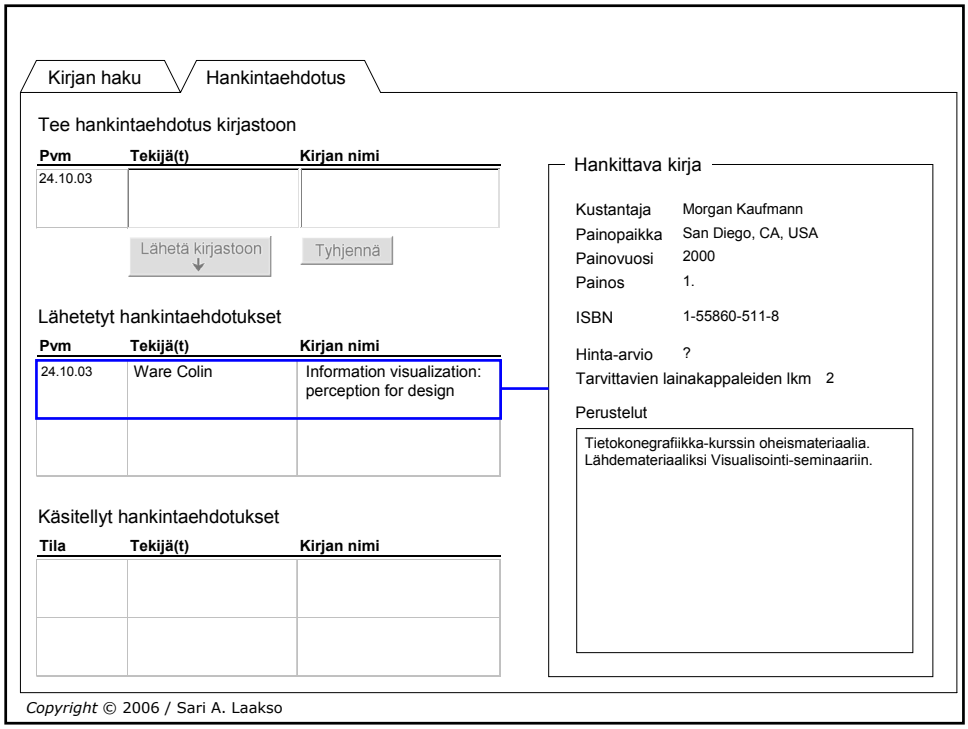
Kirjan haku | Hankintaehdotus

Kansi	Tekijä(t)	Kirjan nimi	Vuosi	Hyllyssä	Lainassa Lainapvm	Lainaajan nimi	Puhelin
Hae	ware						
☐	Neeser Fredy, **Ware** Malcom	Design of a V.34 modem for a real-time multitasking DSP operating system	1995	1			
☐	Putnam Lawrence, Myers **Ware**	Measures for excellence : reliable software on time, within budget	1992	1			





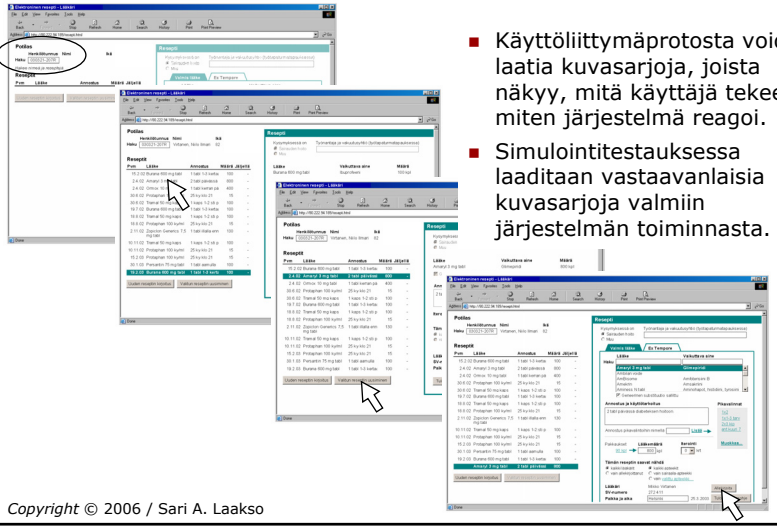




Kuvasarjat toimintalogiikasta

Käyttöliittymän toimintalogiikka voidaan esittää ja dokumentoida kuvasarjana, jossa käyttäjä suorittaa käyttötilannetta. Kuvasarjasta näkyy, mitä käyttäjä tekee ja miten järjestelmä vastaa käyttäjän toimenpiteisiin.

Kuvasarjat toimintalogiikasta
Käyttäjä tekee ja järjestelmä reagoi



- Käyttöliittymäprotosta voidaan laatia kuvasarjoja, joista näkyy, mitä käyttäjä tekee ja miten järjestelmä reagoi.
- Simulointitestauksessa laaditaan vastaavanlaisia kuvasarjoja valmiin järjestelmän toiminnasta.

Ongelmallisia käyttöliittymän suunnittelutapoja

Käyttöliittymä ei tue kovin hyvin tosielämän käyttötilanteista selviytymistä, jos sen suunnittelu perustuu...

1. irrallisiin toimintoluetteloihin,
2. tietomalliin tai
3. järjestelmän käyttöä kuvaaviin UML-käyttötapauksiin.

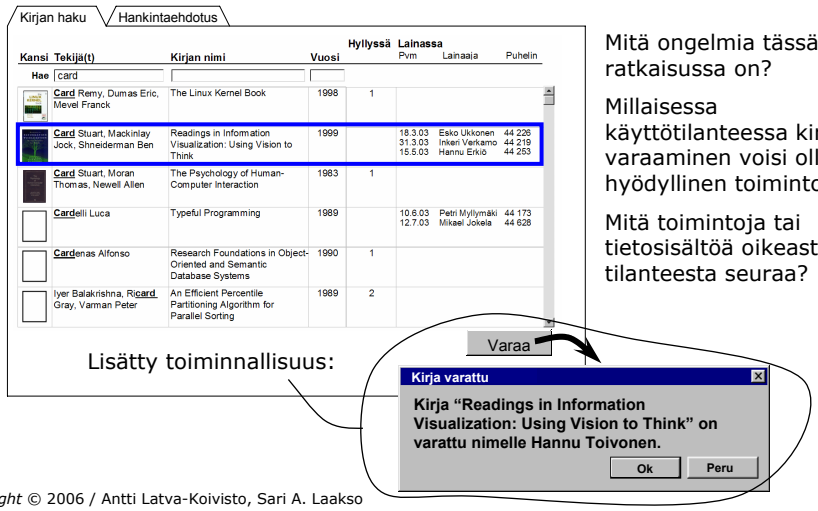
Tässä tarkastellaan ongelmallisten suunnittelutapojen seurauksia niistä syntyvän käyttöliittymäratkaisun kannalta.

1. Irralliset toimintoluettelot

- Jos keksittyjä ominaisuuksia (toimintoja tai tietosisältöä) ripotellaan käyttöliittymään ilman käyttötilanteita, ne ehkä...
  - eivät yksinään riitä tukemaan oikeata käyttötilannetta, vaan tueksi tarvittaisiin muitakin toimintoja/tietosisältöä
  - ovat oikean käyttötilanteen tukemiseen epäoptimaalisia, koska parempiakin ominaisuuksia olisi olemassa
  - eivät olekaan tarpeellisia missään oikeassa käyttötilanteessa [Lauesen02, s. 87]
  - ovat tarpeellisia vain todella harvoissa tilanteissa tai niistä on vain vähän hyötyä suhteessa toteutusvaivaan
  - päätyvät käyttötilanteen kannalta väärään paikkaan, josta käyttäjä ei niitä keksi tai josta niiden käyttäminen on vaikeaa ja tehotonta
- Joskus erikseen määritellyt toiminnot voivat vastata oikeassa käytössä esiintyviä tilanteita. Näin voi käydä esim. silloin, jos toiminnot määritellään hyvin suunnitellun vanhan järjestelmän pohjalta. Tulos jää kuitenkin sattumanvaraiseksi.

Esimerkki 1: Irrallinen Varaus-toiminto

- Tässä esimerkissä määrittelyvaiheessa on esitetty, että kirjastojärjestelmässä pitäisi voida tehdä varaus tällä hetkellä lainassa olevasta kirjasta. Selvitetään, mitä tapahtuu, kun varaus-toiminto lisätään käyttöliittymään toiminnallisen määrittelyn pohjalta ilman käyttötilannetta.



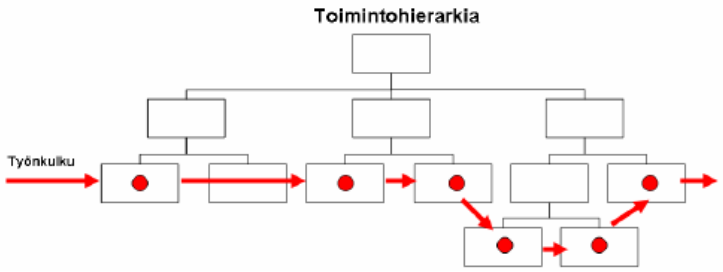
Mitä ongelmia tässä ratkaisussa on?

Millaisessa käyttötilanteessa kirjan varaaminen voisi olla hyödyllinen toiminto?

Mitä toimintoja tai tietosisältöä oikeasta tilanteesta seuraa?

Esimerkki 2: Erillinen hierarkkinen toimintokuvaus

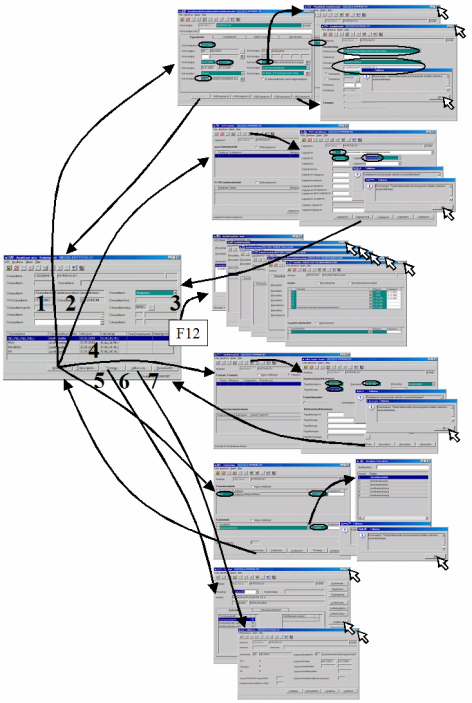
- Tämän esimerkkijärjestelmän (asiaskasrekisteröinnin sovellus) *kaikki* toiminnot suunniteltiin ilman käyttötilanteita. Osa toiminnoista saatiin vanhasta merkkipohjaisesta järjestelmästä.
- Aluksi laadittiin hierarkkisesti organisoitu toimintoluettelo, jonka pohjalta käyttöliittymä suunniteltiin. Eri hierarkiahaarojen toiminnot päätyivät tyypillisesti omiin ikkunoihinsa.
- Tämän tuloksena yhden tehtävän suoritus (työnkulku) kulkee monen eri ikkunan kautta. Suorituspolku on pitkä ja monimutkainen (sekä käyttöliittymän tehokkuus että opittavuus ovat huonoja).



Lähde: Anna Kolehmaisen pro gradu –tutkielma (työn alla)

Esimerkki 2 (jatkuu)

- Näyttökuvat ja näyttöjen välinen navigointi yhden tyypillisen tehtävän suorittamisesta.
- Kuvan tilanteessa käyttäjä lisää uuden asiakkaan asiakasrekisteriin.



Lähde: Anna Kolehmaisen pro gradu –tutkielma (työn alla)

2. Käyttöliittymä tietomallista

- Käyttöliittymän suunnittelemisessa tietomallin (esim. ER-mallin tai UML-luokkakaavion) pohjalta on se ajatus, että jos käyttäjä voi nähdä ja muuttaa kaikkea tietokannassa olevaa tietoa, hän voi *periaatteessa* tehdä mitä tahansa [Lauesen05, s. 208].
- Jos käyttöliittymä suunnitellaan tietokannan pohjalta ilman käyttötilanteita, käyttöliittymään syntyy tyypillisesti seuraavia ongelmia (vrt. [Lauesen05, s. 208-212, 121-123]):
  - Järjestelmässä on paljon erillisiä ikkunoita, esim. jokaisesta tietokantataulusta rivilistaus ja yhden tietueen kaikki tiedot sisältävä ikkuna sekä joitain useita tauluja yhdistäviä ikkunoita.
  - Lukuisien ikkunoiden mielessä pitäminen on käyttäjälle vaikeaa, ja mentaalimallin muodostamisesta tulee monimutkaista.
  - Yhden käyttötilanteen suorittaminen vaatii lukuisten eri ikkunoiden käyttöä ja paljon vaiheita, mistä seuraa tehokkuusongelmia.

Esimerkki: Hotellijärjestelmä
Tietokannan pohjalta paljon näyttöjä

- Tässä esimerkissä [Lauesen05, s. 208-212] hotellijärjestelmän käyttöliittymää on suunniteltu tietokannan pohjalta ja tulosta on verrattu Lauesenin oman menetelmän käyttöön.
- Tuloksena tietokantapohjaisesta suunnittelusta on enemmän erilaisia näyttöjä (9 vs. 6), ja kunkin tehtävän suorittaminen vaatii useammalla eri näytöllä käymistä.

Database-driven design: Windows used	Room availability	Guests	Stays	Guest details	Stay details	Rooms	Services	Service list	Service prices
Reception									
T1.1 Book guest	X	X		x	X	X			
T1.2 Check-in	x	x	X	X	X	X			
T1.3 Check-out			X		X	x	X		
T1.4 Change stay	X		X	x	X	X			
T1.5 Record/change services			X		X		X	x	
T1.6 Breakfast list								X	
T1.7 Room repair	X		x		x	x			
Management									
T2.1 Price changes	X								X
T2.2 Add/change rooms	X								

Complete version: Virtual windows used	Rooms	Floor map	Find guest	Stay	Service prices	(Comboboy)	Breakfast
Reception							
T1.1 Book guest	X	x	X	X			
T1.2 Check-in	x	x	X	X			
T1.3 Check-out			X	X		x	
T1.4 Change stay	x	x	X	X			
T1.5 Record/change services			X	X		X	
T1.6 Breakfast list							X
T1.7 Room repair	X		x	x			
Management							
T2.1 Price changes	X				X		
T2.2 Add/change rooms	X	x					

Legend: X = Window always used during this task. x = Window sometimes used.

Kuvat: [Lauesen05, s. 213 ja 209]

3. Käli UML-käyttötapauksista

- UML-käyttötapauksissa (*use cases*) [Jacobson92] kuvataan käyttäjän tekemät toimenpiteet ja järjestelmän reaktiot niihin.
- UML-käyttötapausten käyttäminen käyttöliittymäsuunnittelun lähtökohtana tuottaa helposti huonon lopputuloksen [Lauesen05, s. 304]. UML-käyttötapausten ongelmia:
  - Ne tukevat askel askeleelta etenevän (wizard-tyyppisen) käyttöliittymän suunnittelemista, jonka käytettävyys on heikko [Lauesen05, s. 164].
  - Niissä kiinnitetään etukäteen suunniteltavan järjestelmän toimintaa, kuten käyttäjän ja järjestelmän välistä työnjakoa, vaikka tätä on tarkoitus vasta suunnitella [Lauesen05, s. 164].
  - Ne kuvaavat tyypillisesti matalan tason tehtäviä, joista monet voitaisiin automatisoida kokonaan.
  - Ne kiinnittävät tehtäville tietyn suoritustavan, minkä seurauksena suunnitteluvaiheessa ei enää ole mahdollisuuksia parempien ja uusien suoritustapojen kehittelyyn [Lauesen05, s. 230].

Esimerkki 1: Kurssi-ilmo
a) Esimerkki UML-käyttötapauksesta

Kurssille ilmoittautuminen

Tavoite: Käyttäjä ilmoittautuu valitsemalleen kurssille.

Käyttäjärooli: Tuleva kurssilainen

Esiehdot: Käyttäjällä on järjestelmään toimiva käyttäjätunnus.

Tyypillinen tapahtumien kulku (T)

1. Käyttäjä antaa käyttäjätunnuksen ja salasanan.
2. Järjestelmä tunnistaa käyttäjän. (A1)
3. Käyttäjä antaa kurssitunnuksen.
4. Järjestelmä näyttää listan kurssin toteutuspäivistä.
5. Käyttäjä valitsee toteutuspäivän ja käynnistää ilmoittautumistoiminnon.
6. Järjestelmä tarkistaa, onko käyttäjä ilmoittautunut samalle kurssin toteutuspäivälle jo aiemmin. Järjestelmä ilmoittaa käyttäjälle, että ilmoittautuminen onnistui, ja lisää ilmoittautumisen tietokantaan. (A2)
7. Jos käyttäjä haluaa jatkaa ilmoittautumista, jatketaan kohdasta 3.

Vaihtoehtoiset tapahtumien kulut (A)

- A1: Käyttäjätunnus tai salasana on väärin. Järjestelmä antaa tästä ilmoituksen. Kolmen yrityskerran jälkeen järjestelmä palaa alkutilaan.
- A2: Käyttäjä on ilmoittautunut jo aiemmin samalle kurssin toteutukselle. Järjestelmä antaa tästä viestin käyttäjälle eikä päivitä tietokantaa.

Esimerkki 1: Kurssi-ilmo
b) Esimerkki käyttötilanteesta

Käyttötilanne: Käyttöliittymän suunnittelu Intranet-projektissa

Nyt on maanantai 5.4. ja Taina on työpaikallaan. Juuri käynnistyneessä Soneran sisäisessä Intranet-projektissa Tainan vastuulla on laatia intranetiin hyvä käyttöliittymä, mutta hän ei ole aiemmin suunnitellut käyttöliittymiä eikä hän tiedä aiheesta juuri mitään ennestään. Aiemmissa projekteissa hän on koodannut web-sovelluksia. Koulutukseltaan hän on tietotekniikan DI.

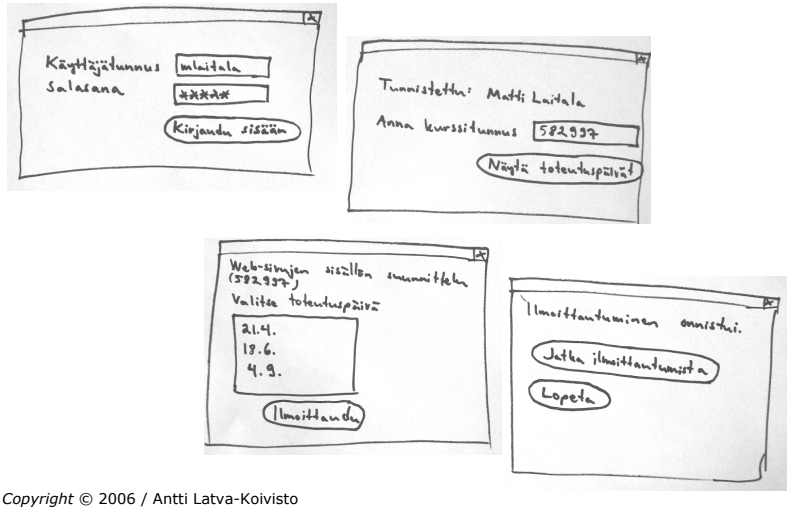
Intranetin käyttöliittymää aletaan koodata ma 14.6. lähtien, ja valmiin järjestelmän käyttöönottopäivä on 16.8.

Taina ei tunne yliopiston kaupallisia kursseja, mutta hän tietää, että kollegat ovat joskus osallistuneet niille.

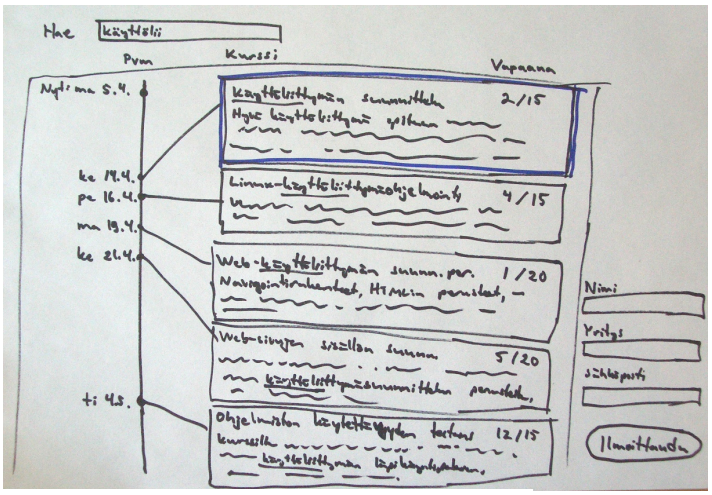
Todellisuudessa yliopisto tarjoaa yrityksille mm. seuraavia kursseja:

- Käyttöliittymän suunnittelu *ke 14.4.*
- Ohjelmiston käytettävyyden testaus *ti 4.5.*
- Web-käyttöliittymän suunnittelun perusteet *ma 19.4.*
- Web-sivujen sisällön suunnittelu *ke 21.4.*

Esimerkki 1: Kurssi-ilmo
a) UML-käyttötapauksesta tehtyä käliä

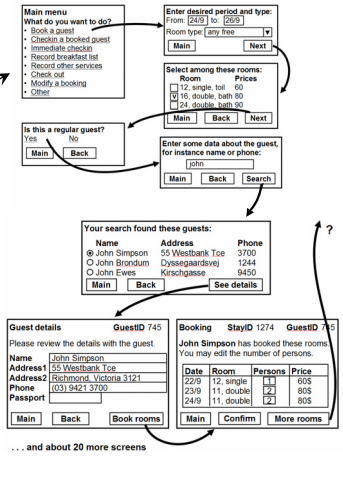


Esimerkki 1: Kurssi-ilmo
b) Käyttötilanteesta tehtyä käliä



Esimerkki 2: Hotellijärjestelmä
UML-käyttötapauksista näyttöketjuja

- Lauesen antaa toisenlaisen esimerkin siitä, minkälaisen ongelmallisen käyttöliittymän suunnitteluun UML-käyttötapaukset voivat houkutella [Lauesen05, s. 215].
- Tässä UML-käyttötapauksen pohjalta laaditussa hotellijärjestelmän päänäkymässä näytetään listaa tehtävistä, joista jokainen johtaa omaan näyttöketjuunsa (vrt. [Lauesen05, s. 212-215, 164]).
- Tuloksena on tehotonta käyttöä ja päätöksentekovaiheen ongelmia, koska samoissa tehtävissä tarvittavia tietoja on ripoteltu eri paikkoihin.



Kuva: [Lauesen05, s. 215]

Käyttöliittymäratkaisun demoaminen

Havainnollinen ja ymmärrettävä demo perustuu käyttötilanteisiin, joiden avulla demon pitäjä näyttää, miten järjestelmää käytetään oikeissa tilanteissa.

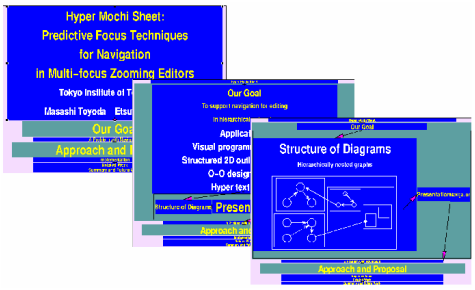
Myös keskeneräisiä paperiprotoja voidaan demota havainnollisesti jäljittelemällä näytön päivittymistä paperilappuja tai piirtoheitinkalvoja siirtelemällä tai kynällä piirtämällä.

Demotekniikka
Perustana käyttötilanne

- Käyttöliittymää kannattaa esitellä demoskenaarioiden (esimerkkitilanteiden) avulla. Demoskenaario kuvaa todellisen käyttötilanteen ja näyttää, miten käyttäjä etenee vaihe vaiheelta kohti tavoitettaan.
- Esimerkki hyvästä demosta, joka on kuvattu käyttötilanteen avulla: Applen *Knowledge Navigator* –video vuodelta 1987 [Dubberly87].

Demotekniikka
Ongelmana toimintoesittely

- Demon katsojan on vaikea hahmottaa järjestelmän toimintaa, jos demossa vain esitellään toimintoja yleisellä tasolla.
- Esimerkki demosta, josta käyttötilanteet puuttuvat kokonaan ja jota on sen vuoksi hyvin vaikea ymmärtää ja seurata: *Hyper Mochi Sheet* [Toyoda99]



Demotekniikka
Etukäteisvalmistelut

Ennen demoa:

- Valitse demottavat käyttötilanteet.
- Valmistele käyttöliittymään realistinen esimerkkiaineisto, jolla demoat.
- Jos järjestelmällä tyypillisesti käsitellään laajoja aineistoja, voit suunnitella demon niin, että aloitat tyhjästä ja sopivassa kohdassa kesken demon hyppäät ajassa eteenpäin ja jatkat laajan aineiston käsittelyllä.
- Harjoittele demo etukäteen, jotta voit varmistua siitä, että kaikki demossa tarvittavat käyttöliittymän osat ovat mukana (paperiprotot) ja että esimerkkidata on kunnossa. Muutoin katsojien aikaa haaskaantuu siihen, kun demon pitäjä yrittää suunnitella käyttöliittymän puuttuvia kohtia tai vaivalloisesti lisätä esimerkkidataa demon aikana lennossa.

Demotekniikka
Demotilanne

Demon aikana:

- Kun demoat paperiprotoa, näytä selvästi käyttäjän **toimenpiteet** (klikkaa esim. kynällä paperiproton painikkeisiin) ja kirjoita protoon **oikeat syötteet** (paperiprotoon kynällä).
- Demoa yhden käyttötilanteen kohdalla **vain ko. tilanteeseen osuvien toimintojen käyttäminen**. Jos poikkeat käyttötilanteesta ja ryhdyt kuvailemaan toimintoja tai käyttöliittymän osia yleisellä tasolla, katsojan on vaikea saada käsitystä käyttöliittymän toimintalogiikasta.
- Demoa käyttäjän **päätöksentekokohdat** (=kun käyttäjä yrittää valita esim. sopivaa kirjaa tai elokuvaa tai hotellia) lukemalla käyttöliittymästä ääneen ne tiedot, joiden avulla käyttäjä päättää, minkä kohteen hän valitsee. Tässä ei kannata lukea ääneen kaikkea näytöllä näkyvää dataa, vaan nostaa esiin vain tässä tilanteessa keskeiset vertailuun tarvittavat tiedot.

Demotekniikka
Vältä tyypilliset demo-ongelmat

- Yritä demon alussa päästä mahdollisimman nopeasti itse asiaan eli demoamaan käyttöliittymää. Katsojat odottavat demoa. Lisäselitykset on parempi kertoa demon jälkeen kuin ennen demoa, koska demon jälkeen katsojilla on jonkinlainen käsitys ohjelman toiminnasta.
- Älä esittele käyttöliittymää toimintojen pohjalta käymällä läpi valikkoja tai painikerivistöjä tyypiin *”Tällä Print-napilla voisi tulostaa...”*, koska katsoja ei tällöin saa käsitystä, mihin tilanteisiin nuo toiminnot liittyvät ja miten niitä silloin käytetään.
- Älä spekuloi demossa suullisesti, mitä ”voisi tehdä”, vaan tee. Katsojat ymmärtävät parhaiten, kun valitset sopivan käyttötilanteen ja näytät, miten järjestelmää käytetään.

Sisältö

Tiedon visualisointi

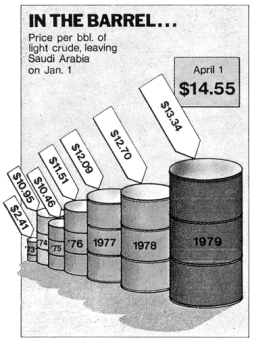
Visualisointien suunnitteluperiaatteita

1. Yksinkertainen koodaus
2. Turha roskamuste pois
3. Selitykset kontekstiinsa
4. Kohde esille kontekstista
5. Datarikas konteksti

1. Yksinkertainen koodaus
Esim. a) 3D-pylväsdiagrammi

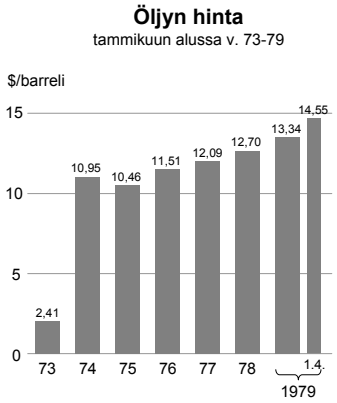
Turhia koodaustapoja, jotka vääristävät tietoa:

- kolmiulotteisuus (tynnyrit)
- perspektiivi



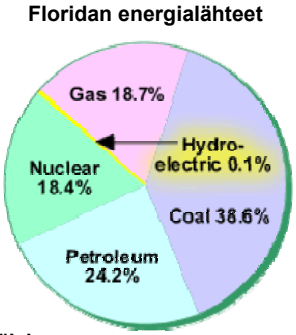
[Tufte83, s. 71]

Turhat koodaustavat poistettu:



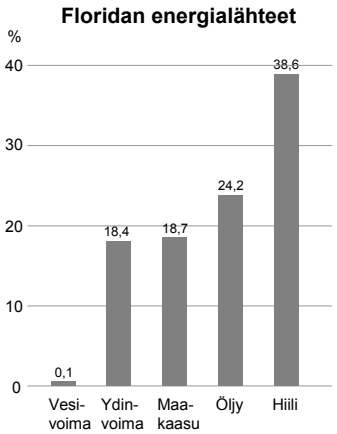
Esim. b) Piirakkakaavio

Turha kaksiulotteisuus vaikeuttaa yksiulotteisten määrien vertaamista:

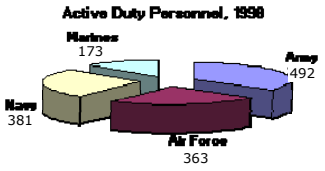


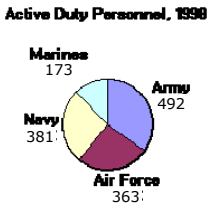
Lähde:
www.sunsmart.org/Green/

Turha kaksiulotteisuus poistettu:

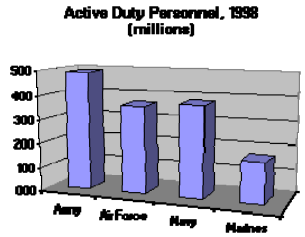


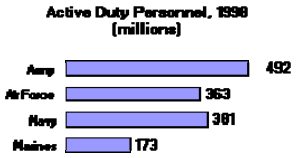
Esim. c) Perspektiivi ja piirakka





Mistä kaaviosta on vaivattominta nähdä esim. suurimman ja pienimmän ero tai vierekkäisten erot taikka saada kokonaiskuva?



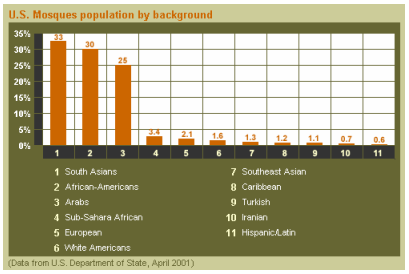


Kuvat lähteestä: http://lilt.ilstu.edu/gmklass/pos138/datadisplay/sections/goodcharts.htm

2. Turha 'roskamuste' pois
Esim. a) Pylväsdiagrammit

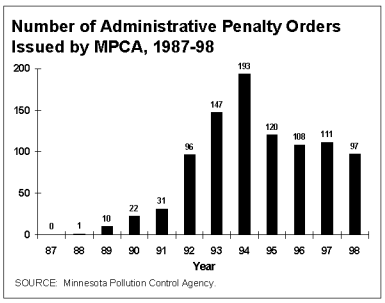
Turhaa roskamustetta ainakin

- valtaosa taustaruudukosta
- tummat laatikot pylväiden alla
- tummat laatikot vasemmalla
- väri ilman informaatiosisältöä



Lähde:
www.opendialogue.com/english/facts.html

Todella vähän roskamustetta - ainoastaan x-akselilla pylväiden väleissä olevat pienet poikkiviivat?



Lähde:
http://www.auditor.leg.state.mn.us/ped/1999/cu99.htm

Esim. b) Pylväsdiagrammit

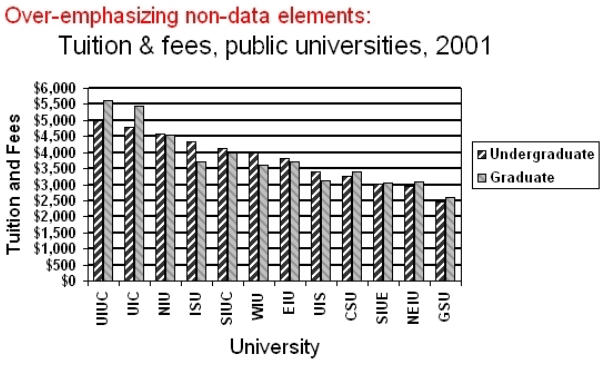
Oheinen pylväsdiagrammi näyttää kisoihin hyväksyttyjen toimittajien lukumääriä. Pylväsdiagrammissa on ylimääräisen roskamusteen lisäksi muitakin ongelmia.

Lähde:
defitv.genesys.com/minisite/uk/aboutCup.htm

Esim. c) Moiré-ilmiön roskamuste

Moiré-ilmiössä viivoitus tai taustakuvio 'hyppii silmissä' ja vaikeuttaa tiedon hahmottamista [Tufte83, s. 108].

Tästä syystä kohteita ei kannata värittää Moiré-raidoilla.



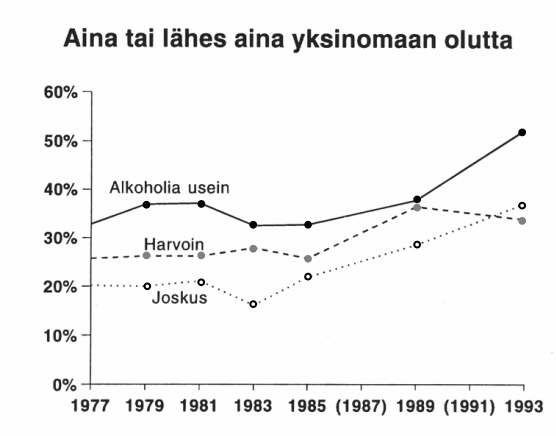
 http://lilt.ilstu.edu/gmklass/pos138/datadisplay/sections/goodcharts.htm

3. Selitykset kontekstiinsa
Esim. a) Käyrien yhteydessä

Hyvä graafi.

Selitykset kontekstissaan käyrien yhteydessä.

Gestalt-periaate: *Lähekkäisyys (Proximity)*

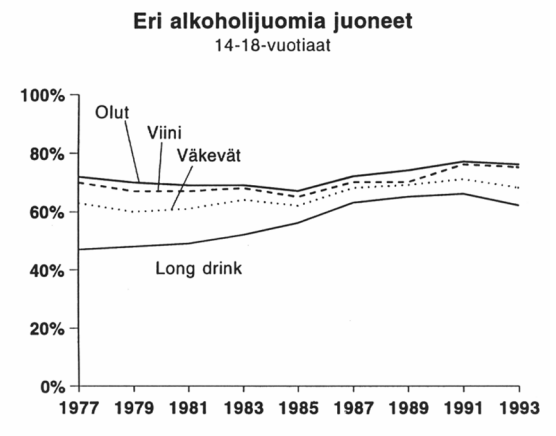


Esim. b) Viivoilla käyriin

Hyvä kompromissi.

Jos tilaa ei ole riittävästi, selitykset voi yhdistää viivoilla käyriin.

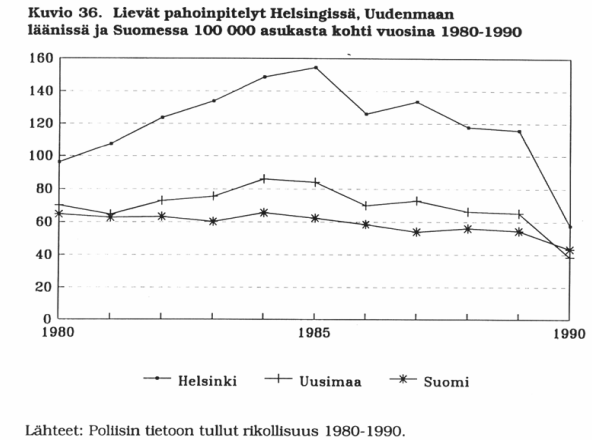
Gestalt-periaate: *Viivalla yhdistäminen (Connectivity)*



Esim. c) Erillään käyristä

Ongelmia:

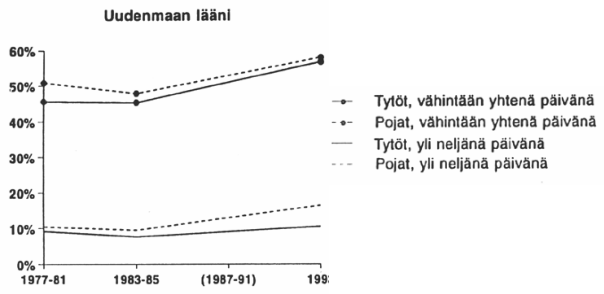
Erilliset selitystekstit pakottavat käyttäjän tekemään työtä selitysten yhdistämiseksi käyriin ja lisäävät mielessä pitämisen tarvetta (työmuistin kuormitusta).



Esim. d) Erillään käyristä

Paljon ongelmia:

Tässä selitystekstit ovat erillään käyristä ja tekstien järjestyskin (*mapping*) on virheellinen käyrien järjestykseen nähden.



Esim. e) Selitykset kartalla

Selitykset hyvin kontekstissaan:

Reittiin liittyvät selitykset on merkitty suoraan kontekstiinsa kartalle. (Karttapohjaa sen sijaan voisi vielä parantaa.)

Helsingin Sanomat 17.1.2001

4. Kohde esille kontekstista
Esim. a) Reitti esille kartalta

Ongelmia: Tässä reitti (kohde) ei erotu kartalta (kontekstistaan) kunnolla, koska molemmat on piirretty yhtä vahvoina.

Tässä on yhtä paljon tietoa kuin viereisessä kuvassa. Reitti erottuu paremmin, koska se on korostettu ja tausta on himmennetty.

Kuvat: http://home.comcast.net/~jdewolf/Tero/Tero_WC05_GB_Middle.pdf

5. Datarikas konteksti
Esim. a) Reitti ja karttakonteksti

Reitin seuraamista ja poikkeustilanteista selviytymistä auttaa himmeämpi mutta datarikas karttakonteksti. (Simuloi!)

http://home.comcast.net/~jdewolf/Tero/Tero_WC05_GB_Middle.pdf

Ongelmia: Tässä reitti on esitetty todella vähäisen karttakontekstin avulla. Mistä syntyy ongelmia?

http://graphics.stanford.edu/papers/routemaps/threemaps.jpg

> "Push back visually a good contextual map and then overlay the route and instructions on top."
>
> Edward Tufte, June 12, 2002
>
> http://www.edwardtufte.com/bboard/q-and-a-fetch-msg?msg_id=0000Bs&topic_id=1&topic=Ask+E%2eT%2e

Esim. b) Karttakontekstin esittäminen

Ongelmia: Kaikki alueiden rajat, tiet, joet ja nimet on esitetty samalla värillä ja vahvuudella. Karttaa on hidasta ja vaikeaa tulkita.

Tässä vielä suurempi määrä tietoa on jäsennetty muodon, tummuuden, koon ja värin avulla. Kartan luettavuus on paljon parempi.

Kuvat: [Tufte90, s. 58]

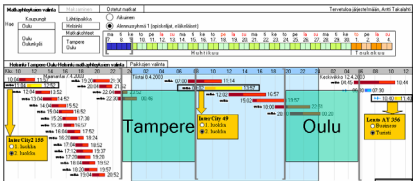
Värit ja värisokeus

Käyttöliittymää voidaan aluksi suunnitella kokonaan ilman värejä ja lisätä värejä vasta sitten, kun väri ratkaisee jonkin ongelman selvästi paremmin kuin muiden visualisointikeinojen käyttäminen.

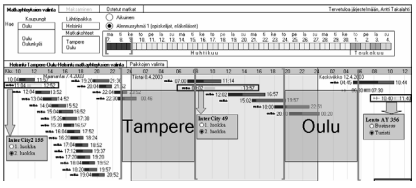
Käyttäjien värisokeudesta riittää käytännössä huomioida tyypillisin värinäköongelma, osittainen tai täydellinen *punavihersokeus*, josta kärsii noin 8 % miehistä. Naisilla värisokeus on huomattavasti harvinaisempaa kuin miehillä.

Värit
Käyttöliittymän harmaasävyversio

- Käyttöliittymäsuunnittelua kannattaa tehdä mahdollisimman pitkälle pelkillä harmaasävyillä ja käyttää muita visualisointikeinoja värien sijaan. Siten värejä säästyy käytettäviksi niihin kohtiin, joissa väri on selvästi parempi visualisointikeino kuin muut.
- Jos värejä on jo käytetty liikaa, korjaaminen on helpointa aloittaa poistamalla käyttöliittymästä ensin kaikki värit ja lisäämällä sitten värejä vain niihin kohtiin, joihin värien käyttö tuo paljon hyötyä.



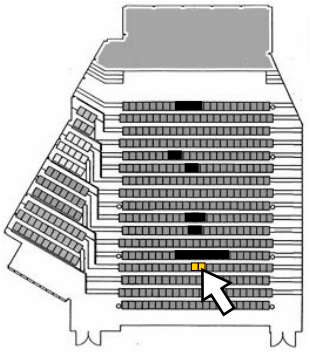
Kuva: Juna-ohjelmistotuotantoprojektin laatima käyttöliittymäluonnos (kesä 2003)



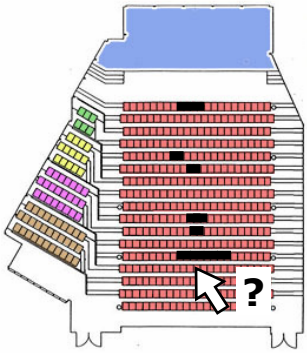
Sama käyttöliittymäluonnos ilman värejä

Värien säästeliäs käyttö
Esimerkki 1: Tampere-talon paikat

- Vain käyttäjän valitsemat paikat näytetään värillisinä. Vaikka ne ovat kooltaan pieniä, ainoana värillisenä kohteena ne erottuvat:

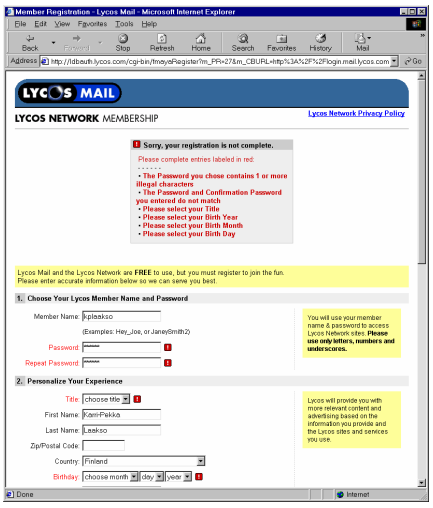


- Tässä eri värejä on jo käytetty niin paljon, että käyttäjän valinnalle on vaikea löytää uutta erottuvaa väriä:



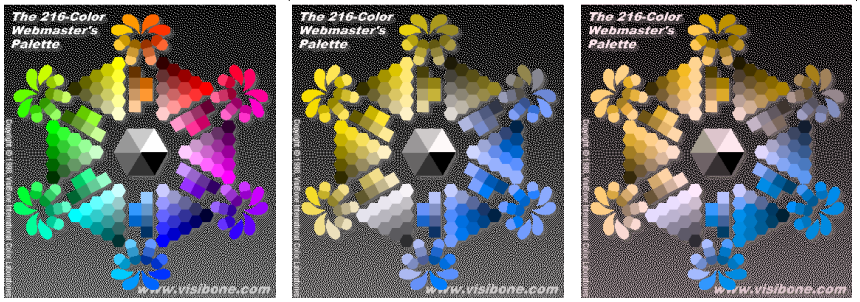
Värien säästeliäs käyttö
Esimerkki 2: Virheilmoitukset

Punaiset tekstit ja ikonit yrittävät vetää käyttäjän huomiota virheellisiin kenttiin, mutta muut värilliset alueet (sininen ja tässä erityisesti keltainen) heikentävät virheilmoitusten nousemista esille.



Värisokeus
Kahdenlaisia punavihersokeita

Miehistä noin 8 % on osittain tai täysin värisokeita, tyypillisesti punavihersokeita.



Normaali värinäkö | Protanopia | Deuteranopia

www.labs.bt.com/people/rigdence/colours/ColChoice.html

Värisokeus
Dikromaatit (1 tappisolutyyppi ei toimi)

Normaali värinäkö

Täysin punaviher- tai keltasinisokeiden henkilöiden määriä:

Punavihersokeus

	Miehet	**Naiset**
Protanopia	1 %	0.02 %
Deuteranopia	1 %	0.01 %

Keltasinisokeus

Tritanopia	0.005 %	0.005 %

www.InternetTG.org/newsletter/mar99/accessibility_color_challenged.html

Värien valinta
Värisokeuden huomiointi

Visualisoinnin suunnittelijalla on kolme päävaihtoehtoa:

1. Käytä väriä **redundanttina koodaustapana** (toissijaisena tapana jonkin muun koodauksen lisäksi)
2. Tarjoa värisokeille **vaihtoehtoisia väripaletteja**.
3. Käytä värejä, jotka **kaikki erottavat**, myös värisokeat.

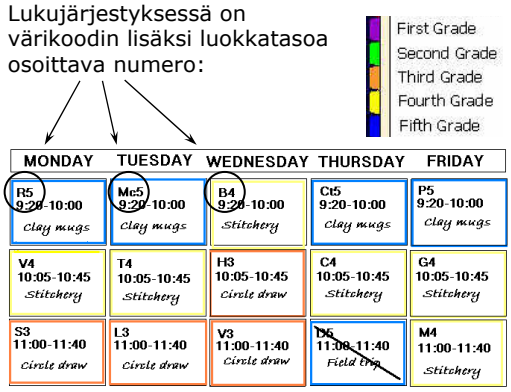
1. Redundantti koodaustapa
Värin lisäksi toinenkin koodaustapa

Esimerkki 1:
Jauhesammuttimien käyttökohteet on koodattu värin lisäksi kirjaimella ja (huonolla) ikonilla.

A	Common Combustibles	Wood, Paper, Cloth, Etc.
B	Flammable Liquids & Gases	Gasoline, Propane other Solvents
C	Live Electrical Equipment	Computers, Fax Machines, Etc.
D	Combustible Metals	Magnesium, Lithium, Titanium
K	Cooking Media	Oils, Lards, Fats

www.illinoisfire.com/commercial.shtml

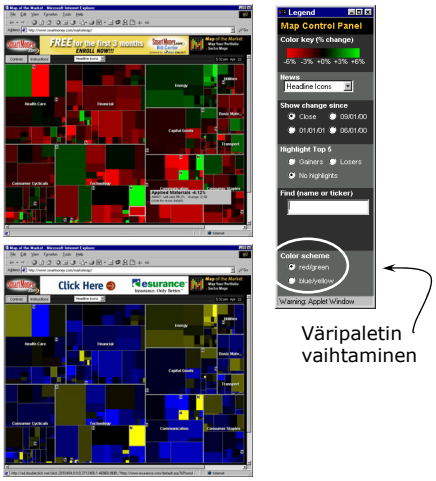
Esimerkki 2:
Lukujärjestyksessä on värikoodin lisäksi luokkatasoa osoittava numero:



www.dreamscape.com/ymrugg/COLOR_CODING_THE_ART_ROOM.htm

2. Vaihtoehtoiset väripaletit
Käyttäjä voi vaihtaa väripaletin

- Voit käyttää värisokeille ongelmallisia väriyhdistelmiä (esim. punainen+vihreä), jos tarjoat käyttäjille mahdollisuuden vaihtaa toiseen värikoodaukseen.
- Yksi vaihtoehtoinen paletti yleensä riittää. Vain todella harvoilla käyttäjillä on ongelmia kaikkien väripalettien kanssa.
- Älä pakota käyttäjää valitsemaan yksittäisiä värejä, vaan tarjoa valmis paletti.



Väripaletin vaihtaminen

www.smartmoney.com/marketmap/

3. Kaikille erottuvat värit
Värierot näkyvissä myös värisokeille

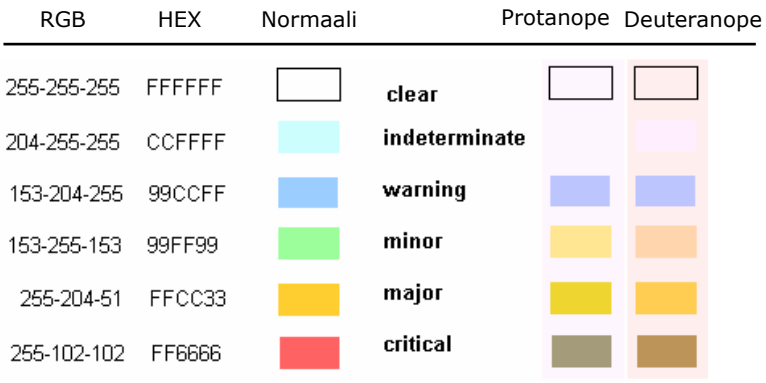
www.labs.bt.com/people/rigdence/colours/ColChoice.html

Esimerkki: 10 erottuvaa väriä

Normaali Protanope Deuteranope

www.labs.bt.com/people/rigdence/colours/ColChoice.html

Esimerkki: Erottuvat varoitusvärit



www.labs.bt.com/people/rigdence/colours/ColChoice.html

Tiedon visualisointi
Hahmolakeja

Ihmisen aistijärjestelmä tulkitsee ja ryhmittelee kohteita. Ihminen muodostaa osista kokonaisuuksia hahmolakien (*Gestalt Laws*) mukaan. Käyttöliittymän suunnittelussa useimmin tarvittuja hahmolakeja ovat seuraavat:

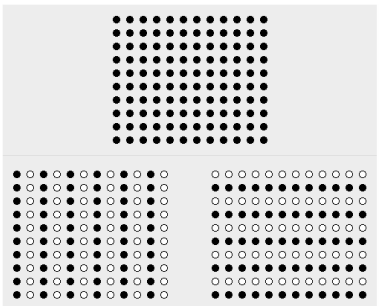
Samanlaisuus
Lähekkäisyys
Viivalla yhdistäminen
Jatkuvuus
Yhteinen alue

[Goldstein02]
[Ware00]

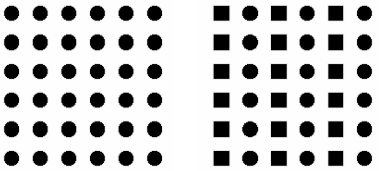
Hahmolakeja
1. Samanlaisuus (similarity)

Samanväriset, -kokoiset ja -muotoiset kohteet ryhmittyvät yhteen.

Sama väri:

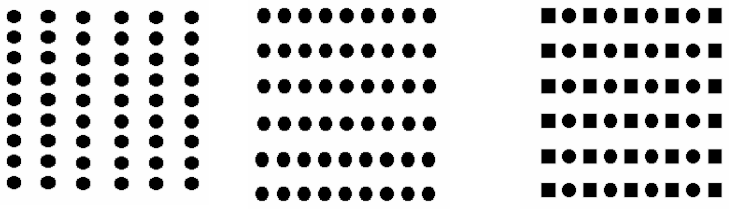


Sama muoto:

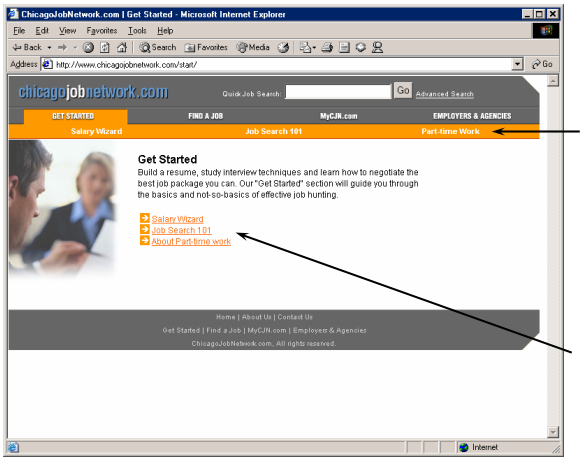


Hahmolakeja
2. Lähekkäisyys (proximity)

Lähekkäisyys on vahvempi kuin samanlaisuus (tässä sama muoto):



Esimerkki 1: Lähekkäisyys

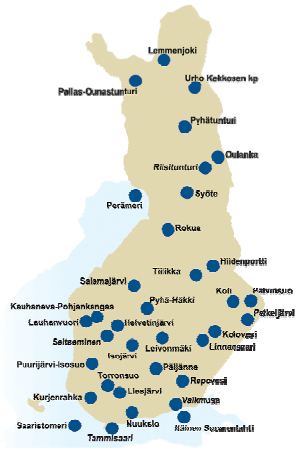


GET STARTED -välilehdellä on kolme vaihtoehtoa, joita samanvärinen taustakaan ei saa ryhmittymään yhteen, koska kohteet ovat liian kaukana toisistaan.

Tässä samat kolme vaihtoehtoa ryhmittyvät yhteen.

www.chicagojobnetwork.com/start/

Esimerkki 2: Lähekkäisyys

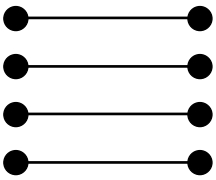


Kansallispuistojen nimet ryhmittyvät yhteen niiden sijaintia kuvaavien pallukoiden kanssa lähekkäisyyden perusteella, vaikka pallukka on eri värinen (sininen) kuin alueen nimi (musta).

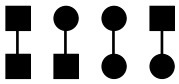
www.metsa.fi/luo/kpuistot/kpkartta.htm

Hahmolakeja
3. Viivalla yhdistäminen (connectivity)

Viivalla yhdistäminen on vahvempi kuin lähekkäisyysperiaate:



Viivalla yhdistäminen on vahvempi kuin samanlaisuus:

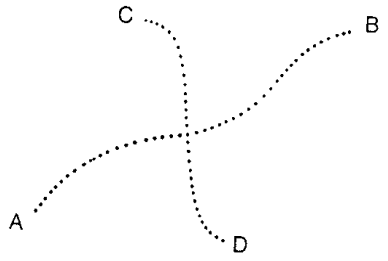


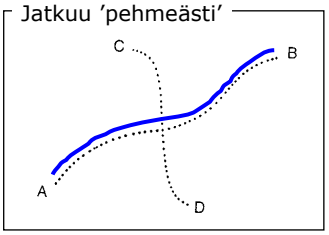
Esimerkki: Viivalla yhdistäminen

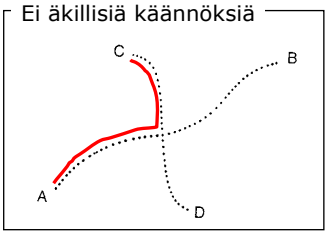
Eteläisten luonnonsuojelualueiden *Laajalahti* ja *Espoonlahti* nimet ryhmittyvät yhteen vastaavien sijaintipallukoiden kanssa viivalla yhdistämisen ansiosta:

www.metsa.fi/luo/lsa/muulsamap.htm

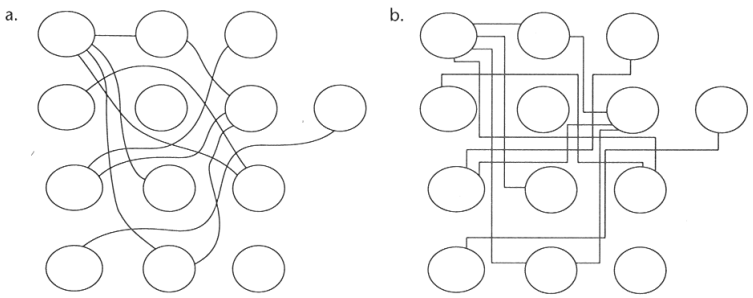
Hahmolakeja
4. Jatkuvuus (continuity)







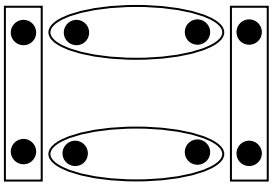
Hahmolakeja
Esimerkki: Jatkuvuus



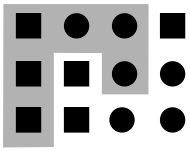
[Ware00, s. 207]

Hahmolakeja
5. Yhteinen alue (common region)

Yhteinen alue on vahvempi periaate kuin läheisyys:



Yhteinen alue on vahvempi periaate kuin samanlaisuus:



Esimerkki: Yhteinen alue

Useita **yhteisen alueen** avulla eroteltuja ryhmiä

Sama data ilman yhteisiä alueita

Lähde:
http://www1.cs.columbia.edu/~paley/spring03/assignments/HWFINAL/es481/principlestable.html

Datan vertaileminen käyttäjän päätöksenteossa

Tiedon organisointi näytölle

Käyttäjän valintatilanteissa tarvitsema data **jäsennetään** ja ryhmitellään käyttöliittymään käyttötilannetta tukevalla tavalla.

Numeerinen tieto kannattaa yleensä myös **visualisoida**, jotta kokonaisuuksien hahmottaminen ja vertailujen tekeminen olisi nopeampaa ja vaivattomampaa.

Datan organisointi
Ongelmat esiin simulointitestauksella

- Simulointitestauksella saadaan selville myös datan organisoinnin ongelmia.
  - Valitse konkreettinen käyttötilanne ja katso sitä käyttäjän näkökulmasta.
  - Selvitä, miten käyttäjä tulkitsee datanäkymää saadakseen käyttötilanteensa suoritetuksi: mitä hän katsoo ensiksi, mitä sen jälkeen. Joutuuko hän esimerkiksi etsimään samaa tietoa silmillään näytöltä moneen eri kertaan?
- Seuraavissa esimerkeissä simulointitestataan...
  - Helsingin Sataman laivojen aikatauludatan organisointia,
  - kalenteritapahtumien organisointia,
  - konserttisalin vapaista paikoista kertovan tiedon organisointia ja
  - hampurilaisten hintatietojen organisointia.

Datan organisointi
Esimerkki 1: Tallinnan laivat

Käyttötilanne: Asiakaspalaveriin Tallinnaan

Nyt on perjantaiaamu 28.5.2004. Taneli on töissä mainostoimistossa, jolla on useita asiakkaita Tallinnassa. Hän ei ole vielä kovin monta kertaa käynyt palaverissa Tallinnassa, mutta hän tietää, että laivoja menee sinne melko tiuhaan. Seuraava palaveri on ensi viikon tiistaina klo 14-16, mutta hän ei tiedä laivojen aikatauluja eikä sitä, millä laivavuorolla hänen kannattaisi mennä palaveriin.

Datan organisointi
Esimerkki 1: Tallinnan laivat

Helsingistä Tallinnaan lähtevät laivat voidaan organisoida esim. *laivayhtiön* tai *lähtöajan* mukaan.

Lähtöajan mukaan organisoitu data tukee tätä käyttötilannetta paremmin.

Laivayhtiöittäin:

Helsinki-Tallinna joka päivä

Linda Line

08.00	>>	09.25
10.00	>>	11.25
12.00	>>	13.25
14.00	>>	15.25
17.00	>>	18.25
19.00	>>	20.25

SILJA LINE

HELSINKI Eteläsatama/ Makasiiniterminaali		TALLINNA Reisisadam/ D-term.
H1 **08.00**	=>	09.40
H2 **10.55**	=>	12.35
H3 **12.30**	=>	14.10
H4 **15.50**	=>	17.30
H5 **18.30**	=>	20.05

TALLINK

07.45-09.15
10.00-11.40
12.00-13.40
15.00-16.40
17.00-18.40
20.00-21.40
22.00-23.40

Lähde: www.hel.fi/port/

Lähtöaikojen mukaan:

Helsinki-Tallinna joka päivä

Lähtö	*Laivayhtiö*
7.45 - 9.15	Tallink
8.00 - 9.25	Linda Line
8.00 - 9.40	Silja Line
10.00 - 11.25	Linda Line
10.00 - 11.40	Tallink
10.55 - 12.35	Silja Line
12.00 - 13.25	Linda Line
12.00 - 13.40	Tallink
12.30 - 14.10	Silja Line
14.00 - 15.25	Linda Line
15.00 - 16.40	Tallink
15.50 - 17.30	Silja Line
17.00 - 18.25	Linda Line
17.00 - 18.40	Tallink
18.30 - 20.05	Silja Line
19.00 - 20.25	Linda Line
20.00 - 21.40	Tallink
22.00 - 23.40	Tallink

Datan organisointi
Esim. 2: Kalenteritapahtumat 1/3

- Yritä muodostaa itsellesi kokonaiskuva seuraavista kalenteritapahtumista. Testaa erilaisia kalenteritiedon esitystapoja vaikkapa seuraavan esimerkkitilanteen avulla:

  Tapasit pitkästä aikaa entisen opiskelukaverisi, jonka kanssa olisi mukava jutella lisää paremmalla ajalla. Hän kysyy, lähtisitkö lounaalle keskiviikkona tai torstaina. Ehditkö?
- Kahdessa ensimmäisessä kalenterin esitystavassa (erityisesti ensimmäisessä) käyttäjä joutuu tekemään ylimääräistä ajattelutyötä yrittäessään muodostaa oikean mielikuvan viikon aikataulustaan.

Datan organisointi
Esim. 2: Kalenteritapahtumat 2/3

Organisointi tapahtumien mukaan (aakkosjärjestys):

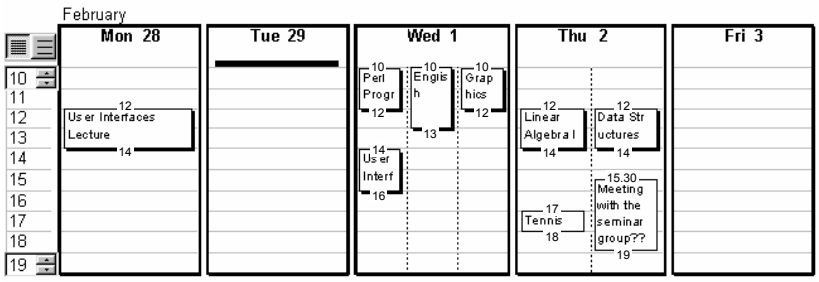
Data Structures	Thu 12-14
English	Wed 10-13
Graphics	Wed 10-12
Linear Algebra I	Thu 12-14
Meeting	Tue 8-9
Meeting with the seminar group??	Thu 15.30-19
Perl Programming	Wed 10-12
Tennis	Thu 17-18
User Interfaces	Mon 12-14 Wed 14-16

Organisointi ajankohdan mukaan:

Mon	12-14	User Interfaces
Tue	8-9	Meeting
Wed	10-12	Perl Programming
	10-13	English
	10-12	Graphics
	14-16	User Interfaces
Thu	12-14	Linear Algebra I
	12-14	Data Structures
	15.30-19	Meeting with the seminar group??
	17-18	Tennis

Datan organisointi
Esim. 2: Kalenteritapahtumat 3/3

Tässä tiedot on organisoitu ajan mukaan ja lisäksi visualisoitu, minkä seurauksena niitä on huomattavasti nopeampaa hahmottaa kuin pelkästä tekstimuotoisesta luettelosta:



Datan organisointi
Esim. 3: Teatterin istumapaikat 1/2

Tässä dataa vapaista ja varatuista istumapaikoista ei näytetä ollenkaan:

la 4.10. TAPIOLAN KUORO 40 vuotta
Tampere-talo, pieni sali

Lippujen lukumäärä 2 Varaa

Olet varannut 2 paikkaa:
Rivi 8, paikka 18
Rivi 8, paikka 19

Tässä näytetään data vapaista paikoista ilman visualisointia:

la 4.10. TAPIOLAN KUORO 40 vuotta
Tampere-talo, pieni sali

Vapaat paikat, permanto

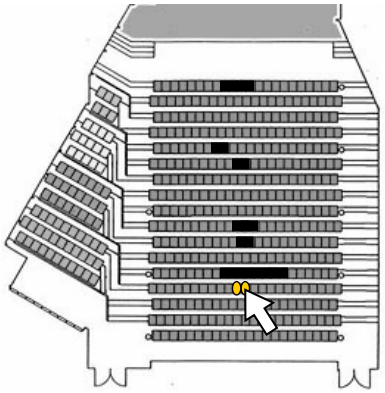
	Rivi	Paikka
☐	1	12
☐	1	13
☐	1	18
☐	2	3
☐	2	4
☑	2	5
☑	2	6
☐	2	22
☐	2	23
☐	3	5

Varaa

Datan organisointi
Esim. 3: Teatterin istumapaikat 2/2

Tässä näytetään data sekä vapaista että varatuista paikoista, ja molemmat on visualisoitu:

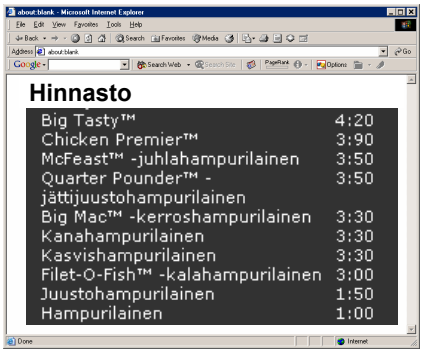
la 4.10. TAPIOLAN KUORO 40 vuotta
Tampere-talo, pieni sali
Valitse paikat:



Kuva:
www.tampere-talo.fi/talo/index.php?alaosio=istumapaikkakartat

Datan organisointi
Esim. 4: Hampurilaishinnasto

Ongelma: Vertailudata on hajallaan kahdessa paikassa.



Datan vertaileminen käyttäjän päätöksenteossa
Vertailtavien kohteiden näyttäminen

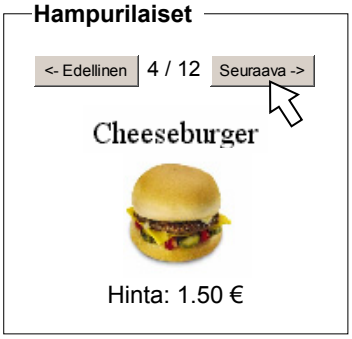
Vertailudata kannattaa pitää käyttöliittymässä näkyvillä siten, ettei käyttäjän tarvitse valita samoja kohteita näkyviin moneen kertaan (palata aiempiin kohteisiin uudelleen) eikä yrittää pitää vertailutietoja mielessään tai kirjoittaa niitä muistiin paperille tai toiseen ohjelmaan.

Käyttöliittymän tulisi helpottaa vertailua eikä vaikeuttaa sitä.

Vertailukohteiden näyttäminen
Vältä mielessä pitämisen ongelma

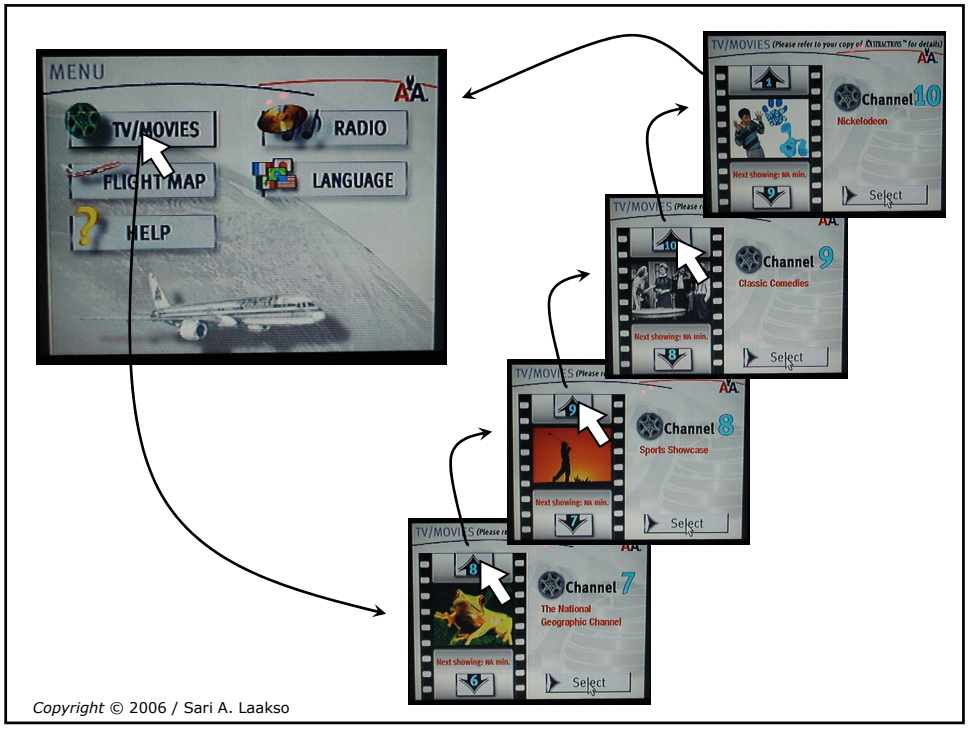
Vertailudata näkyvissä kerralla:

Ongelma: Vertailudata piilotettu.



Vertailukohteiden näyttäminen
Esimerkki: Elokuvat lennolla

American Airlines



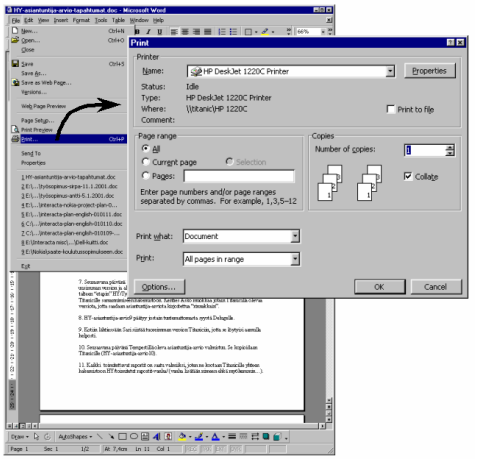
Sisältö

Navigoinnin minimointi

Ikkuna- ja näyttöketjut teettävät käyttäjällä turhaa navigointityötä ja aiheuttavat joissain tilanteissa myös päätöksentekoprosessiin ongelmia.

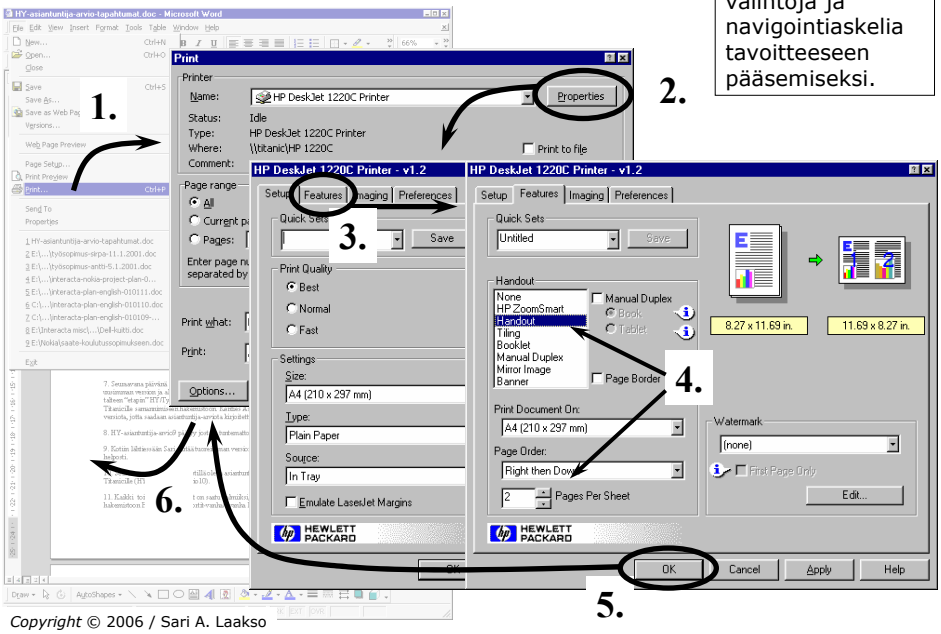
Näistä ongelmista päästään eroon sijoittamalla kerralla tarvittavat tiedot ja toiminnot samaan näkymään ja hyödyntämällä tarvittaessa *Overview beside Detail* –rakennetta.

Ikkuna- ja näyttöketjut
Esimerkki 1: Word-tulostus

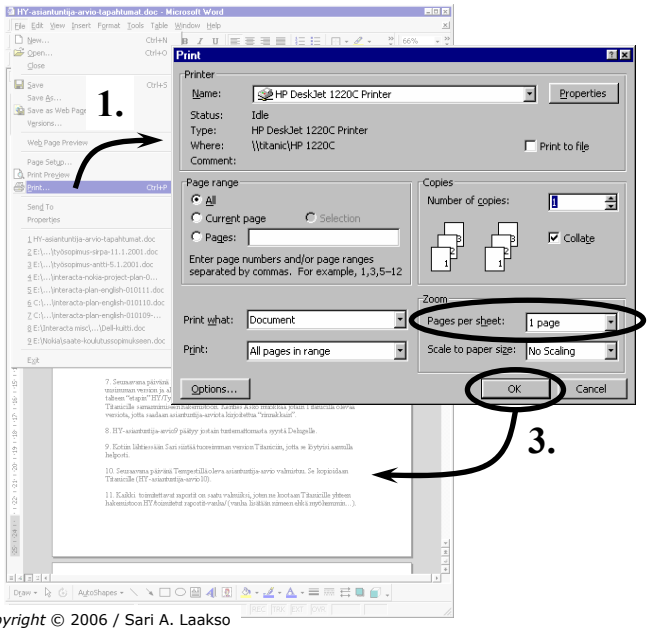


Tehtävä: Dokumentin tulostaminen 2 sivua rinnan.
Tehtävän suorittaminen: **6 vaihetta**

Tässä käyttäjältä vaaditaan paljon valintoja ja navigointiaskelia tavoitteeseen pääsemiseksi.



Tehtävä: Dokumentin tulostaminen 2 sivua rinnan.
Tehtävän suorittaminen: **3 vaihetta**

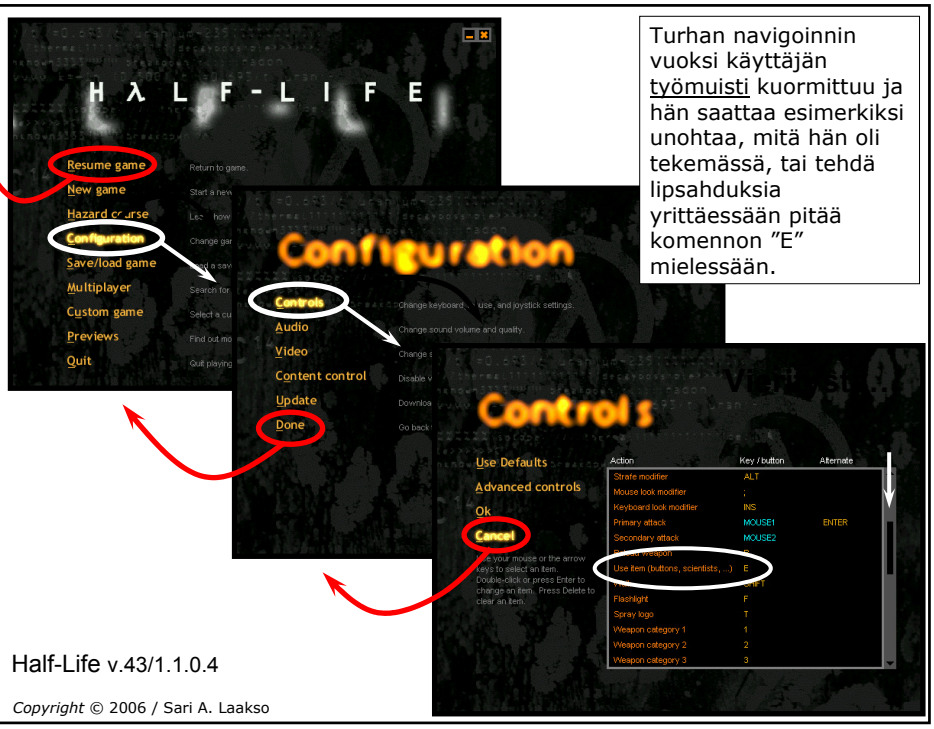


Käyttäjältä vaadittava turha työ vähenee (käytön tehokkuus paranee), kun tarpeettomia valintoja ja navigointiaskeleita poistetaan. Tässä esimerkissä myös opittavuus paranee selvästi.

Ikkuna- ja näyttöketjut
Esimerkki 2: *Half Life* -peli

Tilanne:
Päästäkseen pelissä eteenpäin pelaajan on tuhottava hirviö. Hirviö tuhotaan painamalla TEST FIRE -painiketta, mutta hän ei muista, millä näppäinyhdistelmällä sitä tässä pelissä painetaan.

Half-Life v.43/1.1.0.4

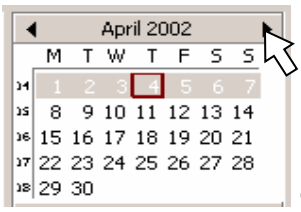


Turhan navigoinnin vuoksi käyttäjän työmuisti kuormittuu ja hän saattaa esimerkiksi unohtaa, mitä hän oli tekemässä, tai tehdä lipsahduksia yrittäessään pitää komennon ”E” mielessään.

Half-Life v.43/1.1.0.4

Ikkuna- ja näyttöketjut
Esim. 3: Jatkuva kalenteri

Erillisiin kuukausipätkiin paloiteltu kalenteri aiheuttaa navigointiketjuja kuukausien välillä ja tuottaa ongelmia kuukausirajalle osuvien viikkojen käsittelyssä.



Microsoft Outlook

Jatkuvassa kalenterissa kuukaudet seuraavat saumattomasti toisiaan ja navigointi on suoraviivaisempaa.

2003	Ma	Ti	Ke	To	Pe	La	Su
	25	26	27	28	29	30	31
Syys	1	2	3	4	5	6	7
	8	9	10	11	12	13	14
	15	16	17	18	19	20	21
	22	23	24	25	26	27	28
	29	30	1	2	3	4	5
Loka	6	7	8	9	10	11	12
	13	14	15	16	17	18	19
	20	21	22	23	24	25	26
	27	28	29	30	31	1	2
Marras	3	4	5	6	7	8	9
	10	11	12	13	14	15	16
	17	18	19	20	21	22	23
	24	25	26	27	28	29	30
Joulu	1	2	3	4	5	6	7
	8	9	10	11	12	13	14
	15	16	17	18	19	20	21
	22	23	24	25	26	27	28
	29	30	31	1	2	3	4
Tammi	5	6	7	8	9	10	11
2004	12	13	14	15	16	17	18
	19	20	21	22	23	24	25

Overview beside Detail -rakenne
Vertailu *Overview*-osassa

- Jos kohteista on olemassa niin paljon dataa, ettei kaikki mahdu samanaikaisesti näkyviin (esim. pitkiä tekstikuvauksia tai suuria kuvia), kokeile *Overview beside Detail* -rakennetta (*ObD*) [Card99, s. 285-286].
  - Sijoita **yleiskuva** (*Overview*) vasempaan reunaan tai yläreunaan. Näytä valitun kohteen **yksityiskohtia** (*Detail*) yleiskuvaosion vieressä tai alapuolella.
  - Korosta **valittuna oleva** kohta yleiskuvanäkymästä, myös alkutilassa. *Detail*-näkymä ei koskaan ole tyhjä.
  - Nosta käyttäjän **päätöksenteon vertailussa** tarvittavat keskeiset tiedot *Overview*-osaan, jotta käyttäjä näkisi vertailtavat tiedot samanaikaisesti:
    + Hänen ei tarvitse pitää vertailtavia tietoja mielessään.
    + Hänen ei tarvitse käydä katsomassa kovin monen kohteen yksityiskohtatietoja (*Detail*).

Overview beside Detail -rakenne
Esimerkki: MMM-tiedotteet

www.mmm.fi/tiedotteet
(Editoitu näyttökuva)

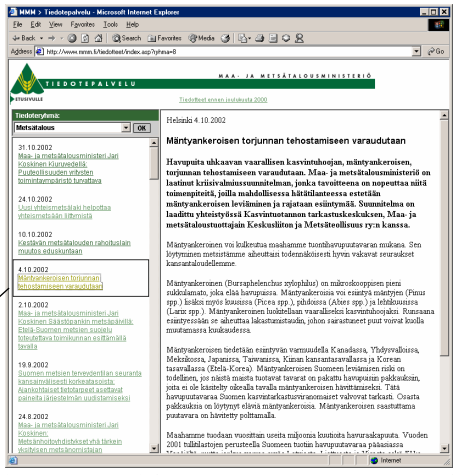
Yleiskuva ja yksityiskohdat:

1. Vaihtoehtojen vertailemisessa tarvittava data sijoitetaan *Overview*-osaan (yleiskuva).

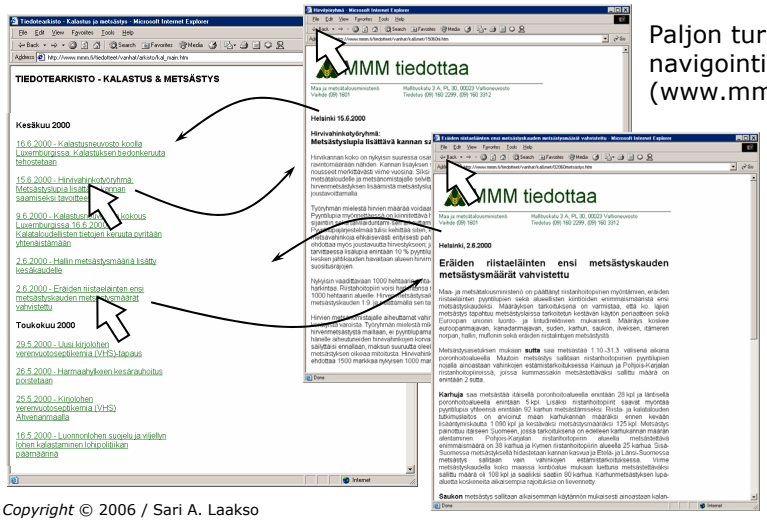
2. Vertailutilanteen jälkeen tarvittava data tai toiminnallisuus sijoitetaan *Detail*-osaan (yksityiskohdat).

Overview-osassa aina yksi on valittuna, myös alkutilassa.

Korosta valittuna oleva vaihtoehto (tässä kehys).



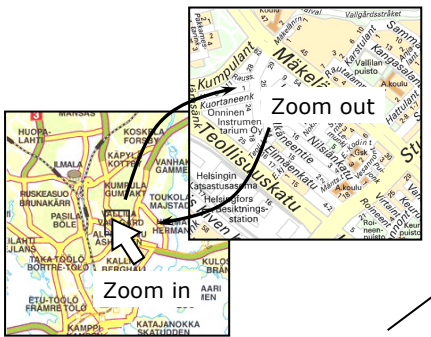
Overview beside Detail -rakenne
Esimerkki: MMM-tiedotteet



Paljon turhaa navigointia (www.mmm.fi)

Overview beside Detail -rakenne
Esimerkki 3: Karttapalvelut

Paljon edestakaista navigointia:



Navigointitarvetta vähennetty *Overview beside Detail* -rakenteella:

Valittu alue

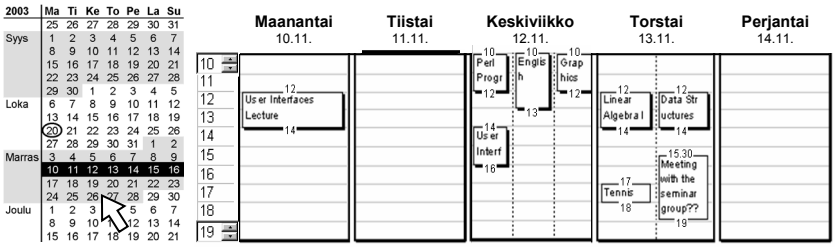
Overview

Detail

Karttaesimerkit: kartta.hel.fi/opas/fi/

Overview beside Detail -rakenne
Esimerkki 4: Kalenterikomponentit

Jatkuva kalenteri ja viikkonäkymä



[Laakso00]

Jokaiseen kalenteriviikkoon on suora pääsy Overview-kalenterista. Tällä vähennetään turhaa navigointia ja monissa käyttötilanteissa esiintyvää tarvetta hahmottaa kalenteria mielessä.

Web-navigointi
Navigointipalkki

- Vältä **sivuketjuja**, joissa käyttäjä 'sukeltaa' sivulta toiselle linkkejä pitkin:
  - Käyttäjän muistikuorma kasvaa, kun hän yrittää mielessään muodostaa käsitystä sivuston rakenteesta ja omasta sijainnistaan sivustossa.
  - Navigointi vaatii paljon toimenpiteitä.
  - Käyttäjä tekee helposti navigointivirheitä (lähtee väärään haaraan) ja saattaa eksyä syviin sivurakenteisiin.
- Käytä **navigointipalkkia** tai edes leivänmurupolkua (bread-crumb trail), koska tällöin…
  - sivuston rakenne ja käyttäjän sijainti ovat koko ajan näkyvissä, ja
  - jokaiselle sivulle on suora pääsy (minimaalinen navigointi).

Web-navigointi
Navipalkin 1-tasoiset välilehdet

Navigointipalkki on sijoitettu yläreunaan.

Nykysijainti on korostettu tummemmalla taustavärillä.

www.eigenvalue.com

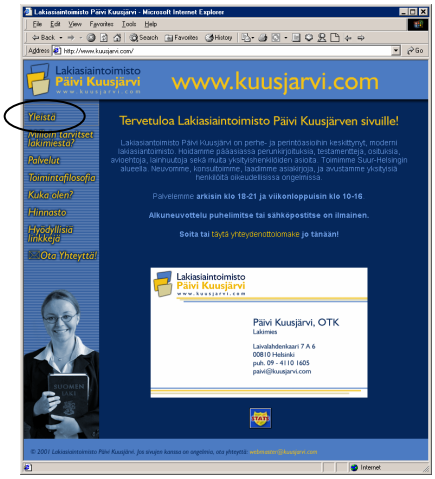
Web-navigointi
4-tasoinen navigointipalkki

Tällä sivulla on 4-tasoinen navigointirakenne, joka on jaettu kahteen palkkiin:
- yläreunassa päätasot ja
- vasemmalla alitasot.

www.nokia.fi/puhelimet/viihde/pelit/extrat.html

Web-navigointi
Etusivu mukaan navigointipalkkiin

Tässä etusivun linkki on sijoitettu navigointipalkin ylimmäiseksi vaihtoehdoksi (*Yleistä*).

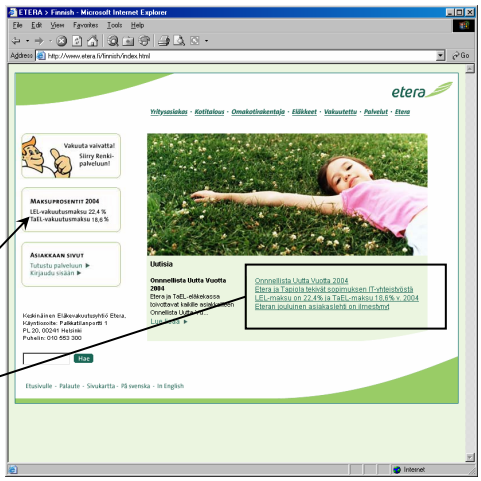


www.kuusjarvi.com

Web-navigointi
Etusivulle tietosisältöä: Etera

Etusivulle kannattaa sijoittaa sellaista hyödyllistä tietoa, jonka avulla osa tyypillisistä käyttötilanteista päätyy suoraan lopputilaan tai nopeasti kohti lopputilaa.

Esimerkiksi Eteran etusivulla on sivuston käyttäjien usein tarvitsemat maksuprosentit sekä linkit uusimpiin tiedotteisiin.



www.etera.fi

Web-navigointi
Navipalkin suunnittelunäkökohtia

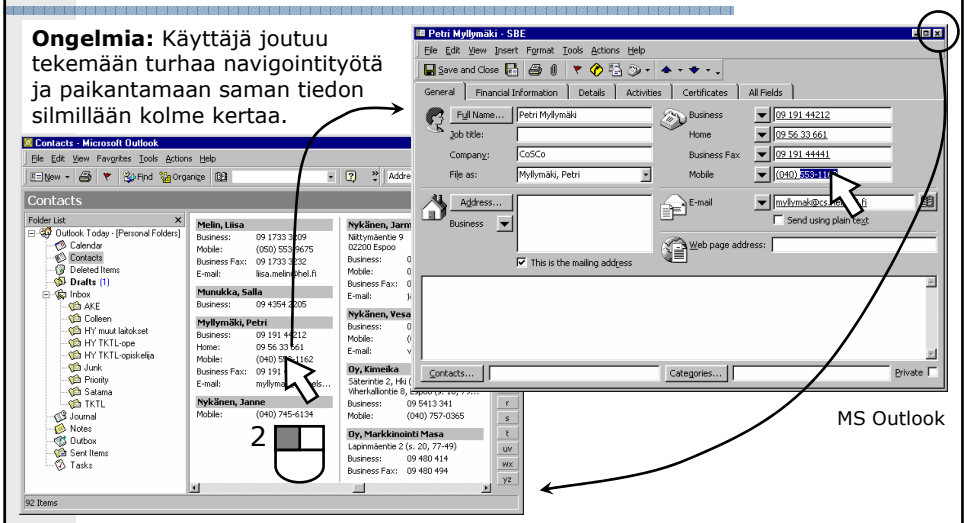
- Korosta **valittuna oleva vaihtoehto** navigointipalkista esimerkiksi lihavoimalla sen teksti tai muuttamalla taustaväriä. Dynaamista valikkovaihtoehtojen korostusta (mouseover) ei tarvita, mutta voi sitä käyttää valittuna olevan vaihtoehdon korostuksen lisäksi.
- Merkitse käyttäjän **vierailemat linkit** eri visualisointitavalla (esim. väri) kuin ne linkit, joissa käyttäjä ei vielä ole käynyt.
- Sisällytä navigointipalkkiin myös **etusivu**.
- Navigointipalkki kannattaa sijoittaa sivun **vasempaan reunaan tai yläreunaan**. Käyttäjä valitsee ensin navigointipalkista (*Overview*) linkin ja sen jälkeen katsoo sivun sisältöä (*Detail*). Länsimaisittain luonteva kontrollisuhteen suunta (valinta ja valinnan seuraus) on sama kuin luonteva lukusuunta.

Tietojen suorakäsittely
Editointi paikallaan

Aina kun mahdollista, anna käyttäjän editoida tietoja samassa paikassa, jossa näytät niitä [Cooper95, s. 177]. Editoitavien taulukkojen käyttö on suoraviivaista (tehokkuus) ja helposti ymmärrettävää (opittavuus).

Editointi paikallaan
Esimerkki 1: Kontaktikortit 1/2

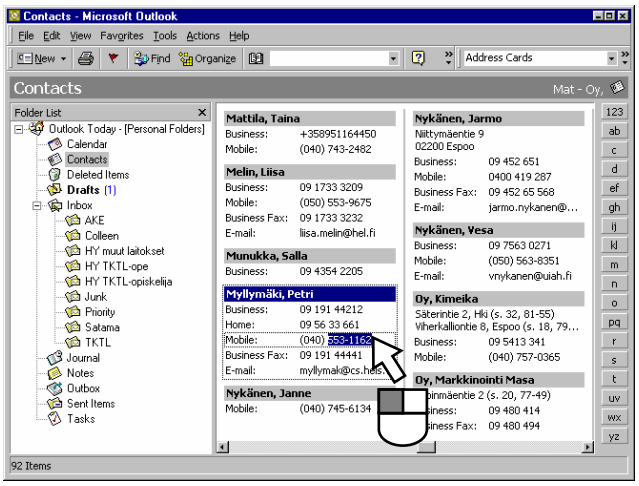
Ongelmia: Käyttäjä joutuu tekemään turhaa navigointityötä ja paikantamaan saman tiedon silmillään kolme kertaa.



MS Outlook

Editointi paikallaan
Esimerkki 1: Kontaktikortit 2/2

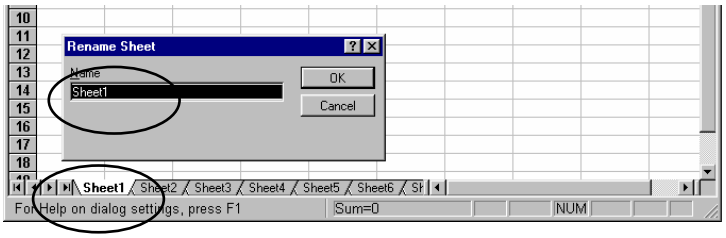
Tehokas suorakäsittely-ratkaisu, jossa edelliset ongelmat on ratkaistu:



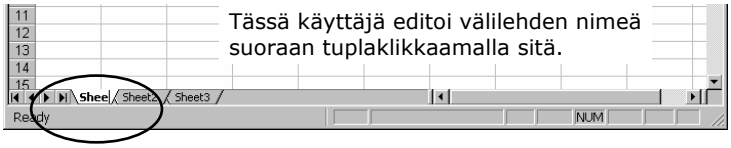
MS Outlook

Editointi paikallaan
Esimerkki 2: Excel-välilehdet

Tämä editointi sisältää turhia vaiheita:



Vastaava suorakäsittelyratkaisu ilman turhia vaiheita:

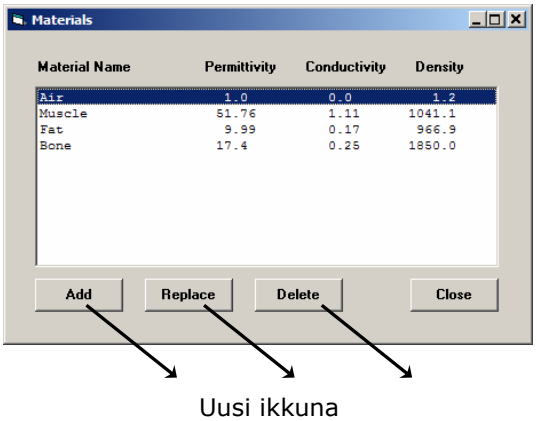


Tässä käyttäjä editoi välilehden nimeä suoraan tuplaklikkaamalla sitä.

Editointi paikallaan
Esimerkki 3: Materiaalitaulukko 1/2

Ongelmia: Tämä käyttöliittymä pakottaa käyttäjän siirtymään muihin ikkunoihin tietojen editointia ja uuden materiaalin lisäystä varten.

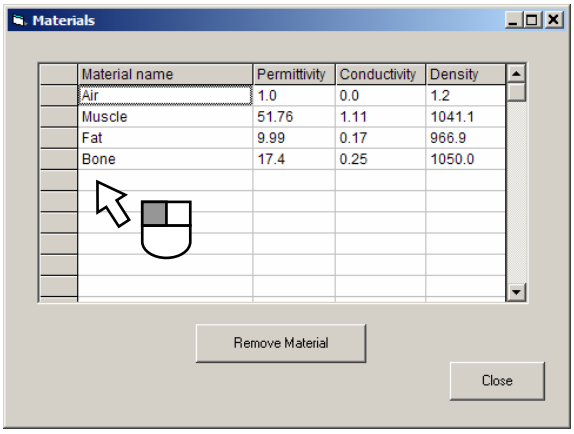
Yksinkertaisesta tietosisällön käsittelystä syntyy neljä erillistä ikkunaa (*Materials*, *Add*, *Replace* ja *Delete*), joiden välillä siirtyileminen on työtä, joka voidaan poistaa käyttäjältä ja siten parantaa työn tehokkuutta.



Editointi paikallaan
Esimerkki 3: Materiaalitaulukko 2/2

Tämä tehokkaampi suorakäsittely-ratkaisu sisältää saman toiminnallisuuden kuin edellinenkin ratkaisu, mutta tässä kaikki turha työ on poistettu käyttäjältä.

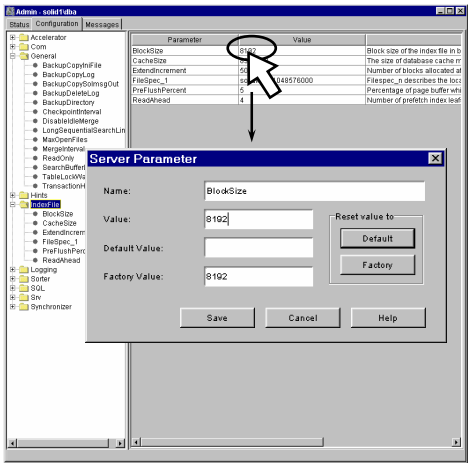
- Editointi suoraan paikallaan
- Uusien lisääminen lopussa oleville tyhjille riveille



Editointi paikallaan
Esimerkki 4: Solid FlowControl

Ongelmia: Toinen esimerkki käyttöliittymästä, jossa käyttäjä ei voi muuttaa parametrin arvoja suoraan paikallaan, vaan muokkaus on mentävä tekemään erilliseen ikkunaan.

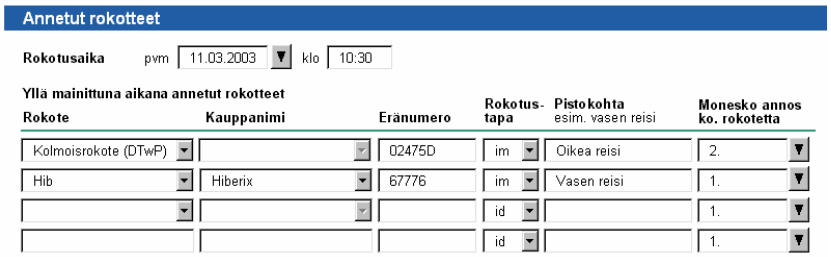
Seurauksena on vastaavanlaisia ongelmia kuin edellisessäkin esimerkissä.



Solid FlowControl

Editoitava web-taulukko
Koko taulukko kerralla editoitavissa

Tässä editoitavassa web-taulukossa terveydenhoitaja syöttää potilaan rokotteet taulukossa valmiiksi näkyville tyhjille riveille. Kaikki rivit ovat jatkuvasti editoitavissa, ja uudet lisätään loppuun.



Rokotusten haittavaikutusilmoitukset

Käyttöliittymä Interacta Design Oy,
toteutus Mediweb Oy

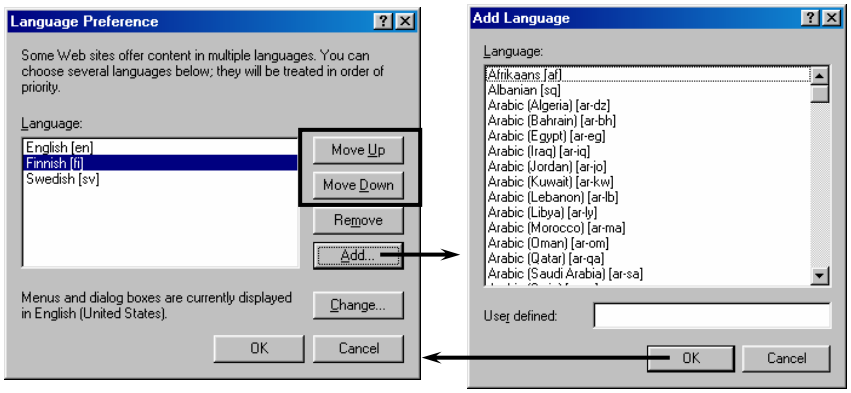
Tietojen suorakäsittely
Kohteen raahaaminen ja kahvat

Kohteen raahaaminen hiirellä on suoraviivainen tapa valintojen tekemiseen ja kohteiden järjestämiseen. Käsiteltävään kohteeseen liitettyjen **kahvojen** (handles) avulla pystyt vihjaamaan käyttäjälle, miten hän voi suorakäsitellä kohdetta.

Erityisesti kuvien ja karttojen yhteydessä raahaaminen on käyttökelpoinen suorakäsittelyratkaisu. Tarjoa lisäksi näppäinoikoteitä tarpeen mukaan.

Kohteen raahaaminen
Esimerkki 1: Kielivalinta 1/2

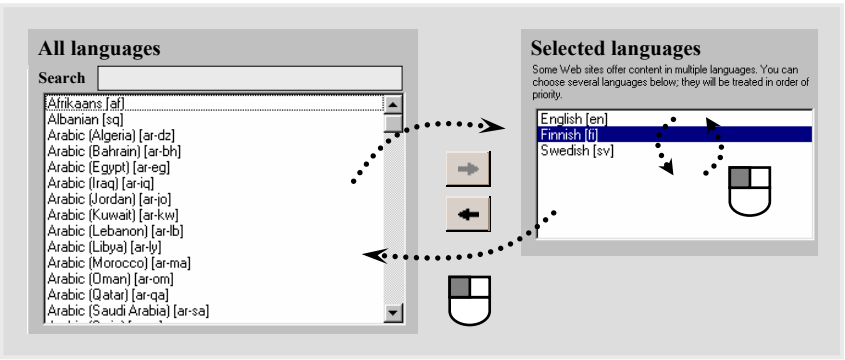
Ongelmia: Tässä käyttäjä joutuu navigoimaan kahden ikkunan välillä ja järjestämään kielet monen erillisen toimenpiteen avulla.



Internet Explorer 5.5: *Internet Options*

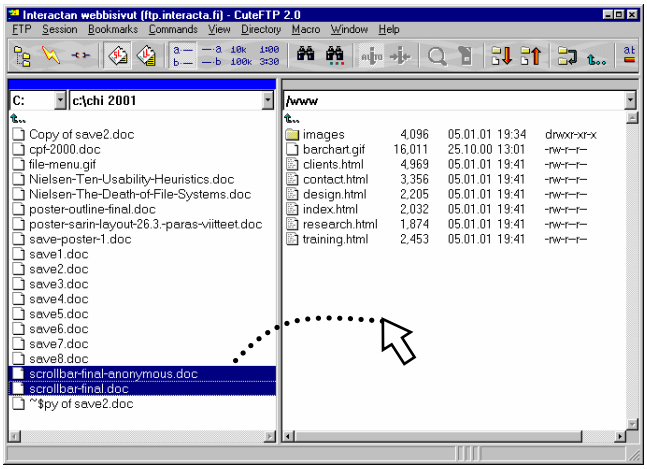
Kohteen raahaaminen
Esimerkki 1: Kielivalinta 2/2

Tässä tehokkaammassa suorakäsittelyratkaisussa käyttäjä voi siirtää kohteita suoraan hiirellä raahaamalla. (Opittavuutta voidaan parantaa vielä kahvoilla, ja raahaamisen lisäksi voidaan tarjota näppäinoikotiet.)



Kohteen raahaaminen
Esimerkki 2: Tiedostojen siirto

Esimerkki tehokkaasta suorakäsittely-ratkaisusta tiedostojen siirrossa:



CuteFTP 2.0

Kohteen raahaaminen
Esimerkki 3: Valokuvien henkilöt 1/2

Ben Shneidermanin PhotoFinderissa käyttäjä nimeää valokuvissa näkyviä ihmisiä raahaamalla listasta nimen valokuvaan ko. henkilön kohdalle:

Henkilön valitseminen nimilistasta

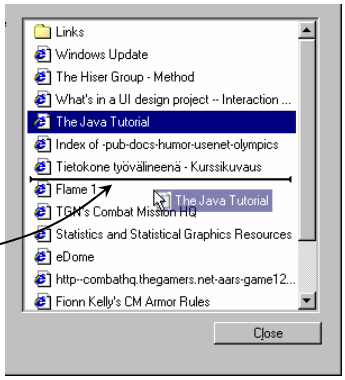
[Shneiderman00]

Kohteen raahaaminen
Esimerkki 3: Valokuvien henkilöt 2/2

[Shneiderman00]

Kohteen raahaaminen
Yksityiskohtia 1/2

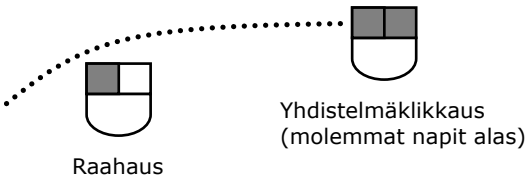
- Raahattava kohde kannattaa näyttää hiirikursorin yhteydessä joko sellaisenaan tai pienennettynä [Cooper95, s. 253-254]. Kuvassa raahattava kohde (The Java Tutorial) näytetään kursorin yhteydessä puoliläpinäkyvänä.
- Kuvassa näkyy myös pudotuskursori [Cooper95, s. 259-260] eli se kohta, johon kohde putoaisi, jos käyttäjä nyt päästäisi irti. Kuvassa pudotuskursori näytetään mustana vaakapalkkina.



Internet Explorer 5.5:
Organize Favorites

Kohteen raahaaminen
Yksityiskohtia 2/2

- Hyviä keinoja raahauksen keskeyttämiseen [Cooper95, s. 233-234] ovat ainakin
  - yhdistelmäklikkaus (hiiren toinenkin nappi alas),
  - Esc-näppäin ja
  - kohteen raahaaminen alueen ulkopuolelle (ei aina mahdollista; sopii joskus paremmin poisto-operaatioksi).



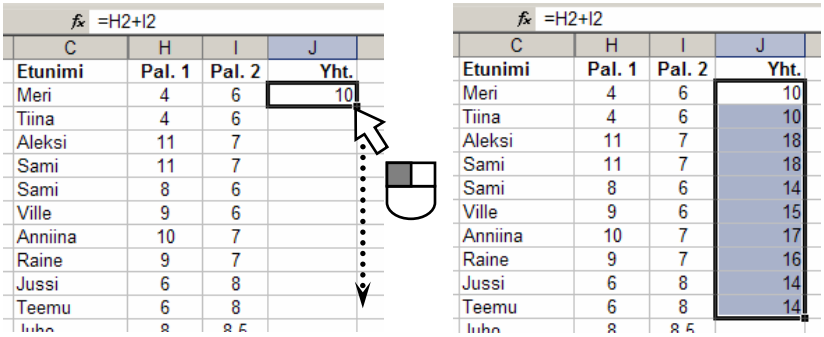
Kahvojen käyttö
Esimerkki 1: PowerPointin kuvakahvat

- Kohteen yhteydessä olevat kahvat tuovat näkyvyyttä (*visibility*) toimintojen käyttämiseen. Toiminnot näytetään kontekstissaan kahvoina, joiden avulla käyttäjä voi suorakäsitellä kohdetta.
- Viereisessä esimerkissä PowerPoint XP tarjoaa kahvat
  - kuvan koon muuttamiseen (*resize*), vrt. [Cooper95, s. 241], ja
  - kuvan kääntämiseen (*rotate*).

PowerPoint XP
(kuva Dürnsteinista, Itävallasta)

Kahvojen käyttö
Esimerkki 2: Excelin solukahva

- Excelissä käyttäjä voi kopioida laskentakaavan muihin soluihin raahaamalla solun kulmakahvasta:



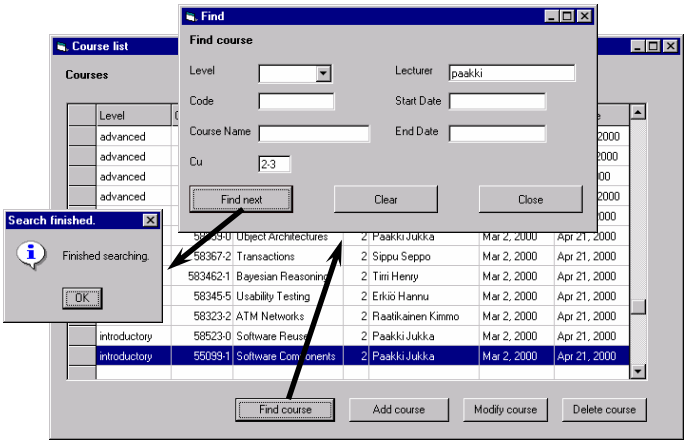
Haku

Kun ohjelma filteröi hakutuloksia käyttäjän syötteiden mukaan jatkuvasti, haku on tehokasta ja käyttäjän riittää syöttää minimaalinen hakuehto. Virheistä toipuminen on nopeaa.

”Näitä dynaamisia hakuja (dynamic queries) voidaan sanoa suorakäsittelyksi, koska niissä on sama idea. Tavoitteena on saada liukusäätimet ... ja muut valintakomponentit yhdistettyä nopeaan näytönpäivitykseen (alle 0,1 sekuntia) jopa silloin, kun näytöllä pitäisi pystyä näyttämään kymmeniätuhansia kohteita kerralla.” [Shneiderman98, s. 541]

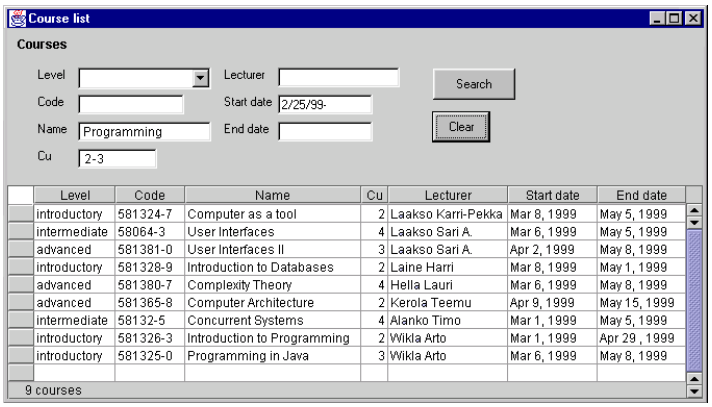
Jatkuva haku
Esimerkki 1: Kurssihaku 1/3

Ongelmallinen ratkaisu, jossa ei ole filteröintiä eikä synkronointia:



Jatkuva haku
Esimerkki 1: Kurssihaku 2/3

Hieman parempi ratkaisu, jossa on filteröinti, muttei synkronointia:



Level	Code	Name	Cu	Lecturer	Start date	End date
introductory	581324-7	Computer as a tool	2	Laakso Karri-Pekka	Mar 8, 1999	May 5, 1999
intermediate	58064-3	User Interfaces	4	Laakso Sari A.	Mar 6, 1999	May 5, 1999
advanced	581381-0	User Interfaces II	3	Laakso Sari A.	Apr 2, 1999	May 8, 1999
introductory	581328-9	Introduction to Databases	2	Laine Harri	Mar 8, 1999	May 1, 1999
advanced	581380-7	Complexity Theory	4	Hella Lauri	Mar 6, 1999	May 8, 1999
advanced	581365-8	Computer Architecture	2	Kerola Teemu	Apr 9, 1999	May 15, 1999
intermediate	58132-5	Concurrent Systems	4	Alanko Timo	Mar 1, 1999	May 5, 1999
introductory	581326-3	Introduction to Programming	2	Wikla Arto	Mar 1, 1999	Apr 29 , 1999
introductory	581325-0	Programming in Java	3	Wikla Arto	Mar 6, 1999	May 8, 1999

Jatkuva haku
Esimerkki 1: Kurssihaku 3/3

Tehokas jatkuva haku (*Filter Table*):

Courses

Search	Level	Code	Name	Cu	Lecturer	Start date	End date
				2-4	la	2/25/99-	
	introductory	581324-7	Computer as a Tool	2	Laakso Karri-Pekka	Mar 8, 1999	May 5, 1999
	intermediate	58064-3	User Interfaces	4	Laakso Sari A.	Mar 6, 1999	May 5, 1999
	advanced	581381-0	User Interfaces II	3	Laakso Sari A.	Apr 2, 1999	May 8, 1999
	introductory	581328-9	Introduction to Databases	2	Laine Harri	Mar 8, 1999	May 1, 1999
	advanced	581380-7	Complexity Theory	4	Hella Lauri	Mar 6, 1999	May 8, 1999
	advanced	581365-8	Computer Architecture	2	Kerola Teemu	Apr 9, 1999	May 15, 1999
	intermediate	58132-5	Concurrent Systems	4	Alanko Timo	Mar 1, 1999	May 5, 1999
	introductory	581326-3	Introduction to Programming	2	Wikla Arto	Mar 1, 1999	Apr 29 , 1999
	introductory	581325-0	Programming in Java	3	Wikla Arto	Mar 6, 1999	May 8, 1999

9 courses

Taulukon ylimmälle riville kirjoitettavat hakuehdot ovat jatkuvasti synkassa hakutulosten kanssa. Kuvassa käyttäjä on juuri syöttänyt kolme hakuehtoa: *Näytä ne **25.2. jälkeen** alkavat **2-4 opintoviikon** kurssit, jotka **luennoi "la"**-jotain.* Alla oleva taulukko päivittyy (filteröityy) jatkuvasti sitä mukaa kuin käyttäjä syöttää hakuehtoa.

Jatkuva haku
Filteröinnin toimintaperiaatteita

- **Alkutilassa** näytetään **kaikki data**, koska
  - esimerkkidata auttaa käyttäjää muodostamaan käsityksen siitä, mitä ohjelmistolla voi tehdä,
  - esimerkkidata antaa malleja kelvollisista hakuehdoista (erillisten opastetekstien tarve vähenee), ja
  - käyttäjä saa välittömästi käsityksen tarjonnasta ja pystyy päättämään mahdollisimman optimaalisen hakumenettelyn: mistä kohdista datanäkymää kannattaa ryhtyä säätämään.
- Kun käyttäjä alkaa **filteröidä dataa** kirjoittamalla hakuehtoa, vain osuman saaneet kohteet jäävät näkyviin.
  - Hakutuloksia päivitetään välittömästi jokaisen näppäinpainalluksen jälkeen (synkronoinnin idea).
  - Osumakohdat korostetaan hakutuloksesta esimerkiksi alleviivaamalla, lihavoinnilla tai erivärisellä taustalla.

Jatkuva haku
Esimerkki 2: Kirjastohaku 1/2

Alkutilassa käyttäjä ei ole vielä syöttänyt mitään hakuehtoa. Kaikki kirjat ovat listassa näkyvissä.

Kirjan haku

Kansi	Tekijä(t)	Kirjan nimi	Vuosi	Hyllyssä
Hae				
	Aho Alfred, Sethi Ravi, Ullman Jeffrey	Compilers: principles, Techniques and tools	1986	6
	Aho Alfred, Kernighan Brian, Weinberger Peter	The AWK Programming Language	1988	
	Alexander, Christopher	The Timeless Way of Building	1979	2
	Beck Kent, Fowler Martin	Extreme Programming Explained: Embrace Change	1999	2
	Beck Kent, Fowler Martin	Planning Extreme Programming	2000	
	Brooks Frederick	The Mythical Man-Month: Essays on Software Engineering	1999	1

Kirjoja 18105 / 18105 kpl

Jatkuva haku
Esimerkki 2: Kirjastohaku 2/2

Käyttäjä on syöttänyt tekijäkenttään nimeä ”card”. Alla oleva tuloslista päivittyy jokaisen näppäinpainalluksen (kirjaimen) jälkeen.

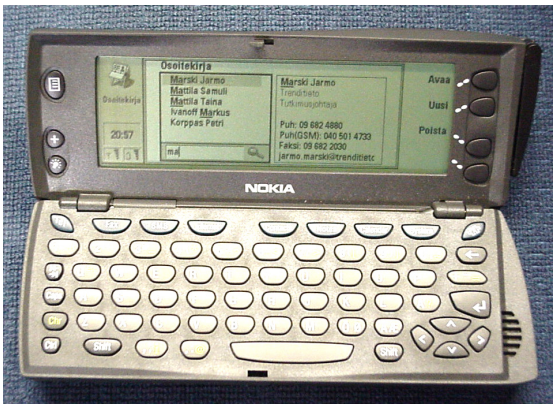
Kirjan haku

Kansi	Tekijä(t)	Kirjan nimi	Vuosi	Hyllyssä
Hae	card			
	**Card** Remy, Dumas Eric, Mevel Franck	The Linux Kernel Book	1998	1
	**Card** Stuart, Mackinlay Jock, Shneiderman Ben	Readings in Information Visualization: Using Vision to Think	1999	2
	**Card** Stuart, Moran Thomas, Newell Allen	The Psychology of Human-Computer Interaction	1983	1
	Cardelli Luca	Typeful Programming	1989	
	Cardenas Alfonso	Research Foundations in Object-Oriented and Semantic Database Systems	1990	1
	Iyer Balakrishna, Ri**card** Gray, Varman Peter	An Efficient Percentile Partitioning Algorithm for Parallel Sorting	1989	2

Kirjoja 13 / 18105 kpl

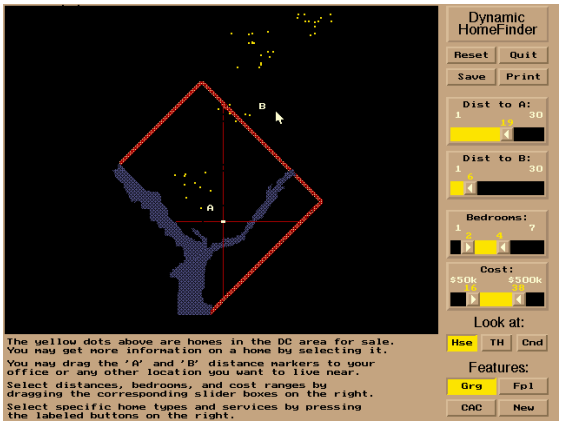
Jatkuva haku
Esimerkki 3: Kontaktikorttihaku

Tehokas jatkuva filter-haku
Nokia Communicator 9110/9210:ssa:



Jatkuva haku
Esimerkki 4: HomeFinder-haku

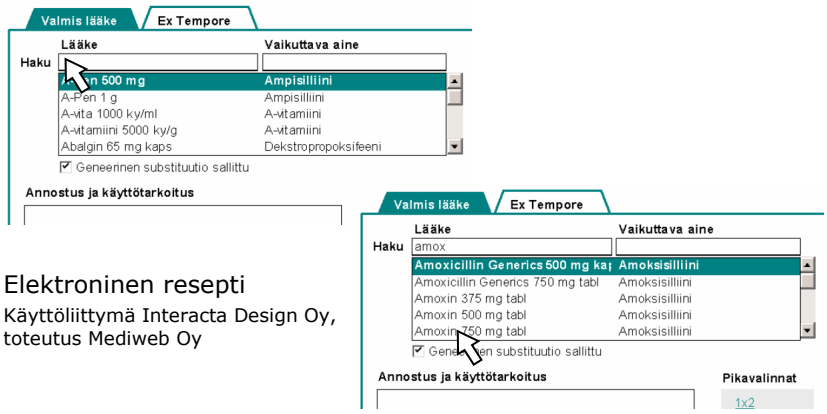
Jatkuva haku (*dynamic query*) HomeFinder-asuntohakujärjestelmässä [Williamson92]:



Asuntotarjonta päivittyy kartalla sitä mukaa kuin käyttäjä raahaa säätimiä (*sliders*):

Jatkuva haku
Web-toteutus

Esimerkki webiin toteutetusta jatkuvasta filter-hausta:



Elektroninen resepti

Käyttöliittymä Interacta Design Oy,
toteutus Mediweb Oy

Jatkuva haku
Käytettävyystestauksen tuloksia

- Lauesen on havainnut hotellin vastaanottoon suunnitellun varausjärjestelmän käytettävyystestauksessa lukuisia käytettävyysongelmia, jotka korjaantuivat, kun haku muutettiin synkronoiduksi (jatkuvaksi hauksi) [Lauesen05, s. 286].
- Esimerkkejä ongelmista, jotka jatkuva haku korjasi:
  - Käyttäjä täytti aina turhaan kaikki hakuehdot, vaikka vain pari hakuehtoa olisi riittänyt rajaamaan tuloksen.
  - Käyttäjä ei huomannut haun käynnistävää *Find*-painiketta eikä onnistunut käynnistämään hakua.
  - Syötettyään hakuehdot käyttäjä painoi Enter-näppäintä saadakseen hakutulokset näkyville, mutta Enterin painallus avasikin eri ikkunan.
  - Joissain tilanteissa haun käynnistämisen jälkeen näytölle ilmestyi tyhjä hakutuloslista. Todellisuudessa käyttäjä oli tehnyt hakuehtoon kirjoitusvirheen. Käyttäjä ei ymmärtänyt, miksi lista oli tyhjä (ei keksinyt, että oli tehnyt kirjoitusvirheen).

Keräilykori päätöksenteon tukena

Jos tarjontaa ja vertailtavia kohteita on paljon, mielessä pitämisen ongelmia voidaan ratkaista tarjoamalla käyttäjälle keräilykori. Myös keräilykorin sisällä on pystyttävä vertailemaan kohteita.

Keräilykori
Esimerkki 1: Kuva-arkisto

Jotta käyttäjän ei tarvitse vertailutilanteessa pitää hyviksi havaitsemiaan kohteita mielessään ja hakea niitä moneen kertaan esille, käyttöliittymä voi tarjota keräilykorin, johon hän voi siirtää löytämänsä kohteet talteen.

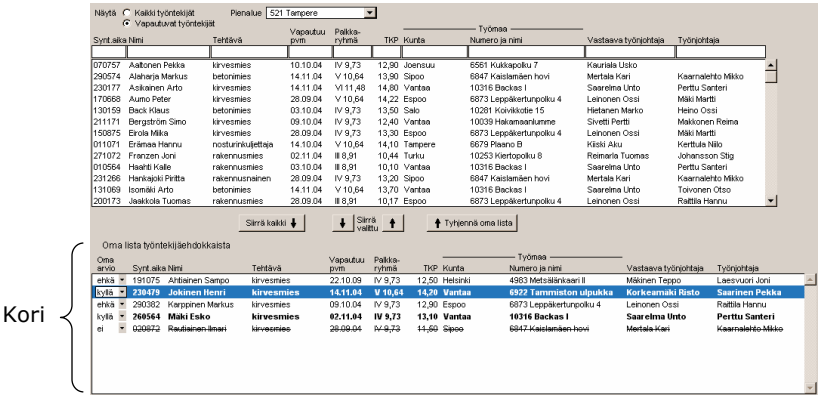
Myös korin sisällä on syytä tarjota mahdollisuus kohteiden vertailuun, koska hyvien kohteiden vertailemista usein kannattaa jatkaa vielä korissa.

Joissain tapauksissa keräilykori toimii ostoskorina/varauskorina, josta käyttäjä voi edetä ostamiseen, tilaamiseen tai varaamiseen.

Esimerkki 1: Kuva-arkiston kuvakori.

Keräilykori
Esimerkki 2: Työntekijöiden valinta

Tässä esimerkissä käyttäjä valitsee ylemmästä kaikkien työntekijöiden taulukosta sopiviksi katsomiaan ehdokkaita alempaan keräilykoriin:



Palaute (feedback)

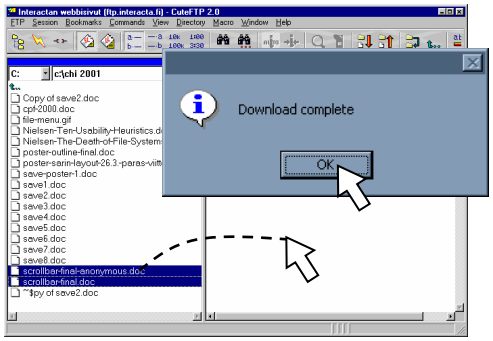
Suorakäsittelyoperaation palaute (*feedback*) kannattaa näyttää kohteen tai toiminnon välittömässä läheisyydessä. Lähettämiseen liittyville toiminnoille hyvä palaute on kerryttää lähteneet viestit tai tilaukset lähetettyjen listaan.

Pitkäkestoisten operaatioiden kohdalla animoitu edistymispalkki auttaa käyttäjää ymmärtämään, miten operaation suoritus etenee.

Suorakäsittelyn palaute
Vältä erillisiä ilmoitusikkunoita

Tässä esimerkissä käyttäjä on siirtämässä tiedostoja levyltä toiselle, ja ohjelma näyttää palautteen erillisessä ikkunassa.

Ongelmia: Erillisessä viesti-ikkunassa näytetty palaute keskeyttää käyttäjän työnteon, ja ikkuna on aina kuitattava erikseen [Cooper95, s. 441-443]. Tästä syntyy turhaa työtä ja joissain tilanteissa myös työmuistiongelmia, kun käyttäjän tavoite tai tehtävän tekemiseen liittyviä tietoja putoaa keskeytyksen takia pois työmuistista.

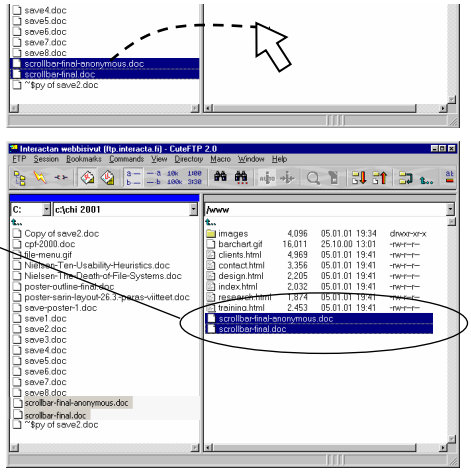


CuteFTP 2.0

Suorakäsittelyn palaute
Palaute näkyviin kontekstiinsa

Tässä edellä kuvatut ongelmat on korjattu ratkaisulla, jossa ohjelma näyttää palautteen kontekstissaan korostamalla siirtyneet tiedostot. Käyttäjä saa palautteen tiedostojen siirtymisestä, mutta hänen ei tarvitse erikseen kuitata palautteen lukemista.

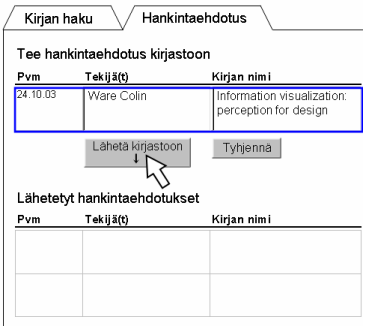
(Miten tätä ratkaisua voitaisiin parantaa edelleen?)

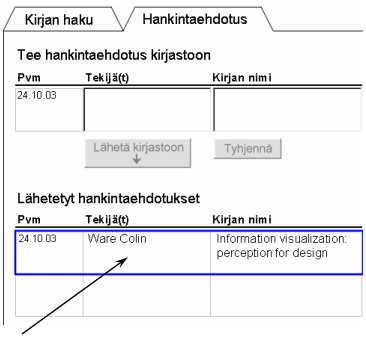


CuteFTP 2.0

Lähettämistoiminnon palaute
Lähetetyt kohteet kertyvät listaan

Hyvä palaute tilausten, varausten ja erilaisten viestien lähettämiselle on kerryttää lähetetyt kohteet lähetettyjen listaan (taulukkoon). Juuri lähetetty kohde kannattaa korostaa listassa valintapalkilla.

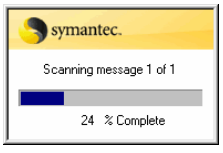




Lähettämistoiminnon palaute näkyy täällä listassa, jonne lähetetyt hankintaehdotukset kertyvät.

Pitkään kestävät operaatiot
Palautteen suunnittelunäkökohtia

- Kun ohjelma juuttuu tekemään jotakin, sen on syytä informoida käyttäjäänsä [Cooper95, s. 316]:
  - Aikaavievä operaatio on työn alla, ja asiat ovat täysin normaalisti.
  - Kuinka paljon aikaa vielä menee.
  - Käyttäjälle annetaan mahdollisuus keskeyttää operaatio.
- Animoitu edistymispalkki (*progress indicator*) luo käyttäjälle mielikuvan siitä, että ohjelma on todella tekemässä jotakin ja etenee.
- Esimerkkejä aikaavievistä operaatioista:
  - tiedostojen kopiointi ja siirto
  - tulostaminen
  - sähköpostiviestin virusskannaus (kuvassa)
  - laajan hakutuloksen laskeminen



Norton AntiVirus 2003

Pitkään kestävät operaatiot
Esim. Photoshopin kuvankäsittely

- Kuvan tilanteessa käyttäjä on muuttamassa kuvaa maalauksen näköiseksi.
- Hän on valinnut *Allemaalaus*-suotimen, jonka vaikutusten laskenta vie jonkin aikaa.
- Järjestelmä näyttää aikaavievän operaation palautteen kuvan alareunassa mustana edistymispalkkina (*progress indicator*).

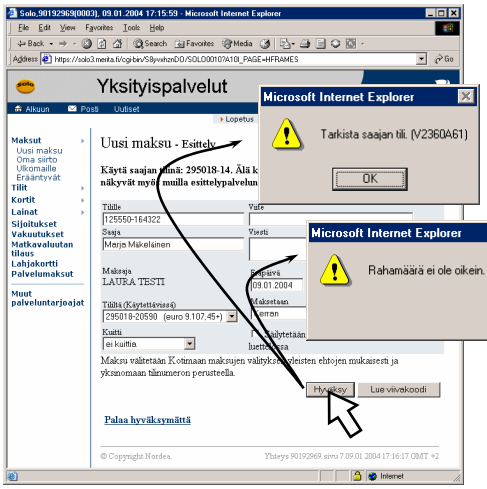
Adobe Photoshop 5.0
(Valokuva Haagista, Hollannista)

Virheiden käsittely

Yritä ensin estää ne virheet, jotka voit [Nielsen93, s. 145-146]. Näissä tapauksissa virheilmoitusta ei tarvita ollenkaan.

Kun virheilmoitus tarvitaan, kaikki virheelliset kohdat kannattaa näyttää kerralla virhekohtien välittömässä läheisyydessä.

Virheiden käsittely
Ongelmalliset erilliset virheilmoitukset



Ongelmia: Nordean Solo-pankkipalvelu näyttää vain yhden virhekohdan kerrallaan ja senkin erillisessä ikkunassa. Virheilmoituksen tekstikään ei ole kovin informatiivinen.

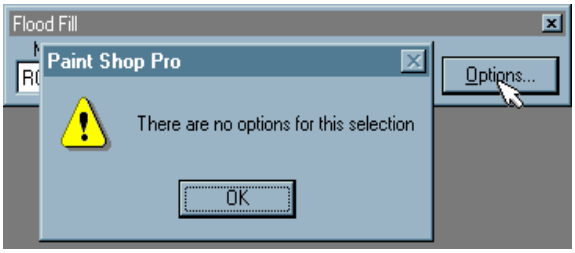
Virheiden käsittely
Virheiden syntymisen estäminen

Virheitä voidaan estää mm. harmaannuttamalla toimintoja ja tarjoamalla rakenteisia valintakomponentteja, joissa data voi olla visualisoitu nopeasti hahmotettavaan muotoon.

Virheiden estäminen
Toimintojen harmaannuttaminen

Ongelmia: Tässä ohjelma harhauttaa käyttäjää antamalla hänen valita toiminnon, joka ei juuri nyt ole käytettävissä.

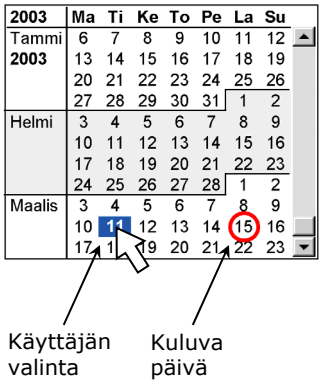
Parempi ratkaisu: Kun tällaiset toiminnot ja valintavaihtoehdot harmaannutetaan (*disabled*), käyttäjä ei turhaan erehdy yrittämään niiden käynnistämistä.



Virheiden estäminen
Rakenteiset komponentit

Turhia virheitä voidaan estää myös visualisoimalla syötteet havainnolliseen muotoon ja tarjoamalla käyttäjälle vapaan syötekentän sijaan valintakomponentin.

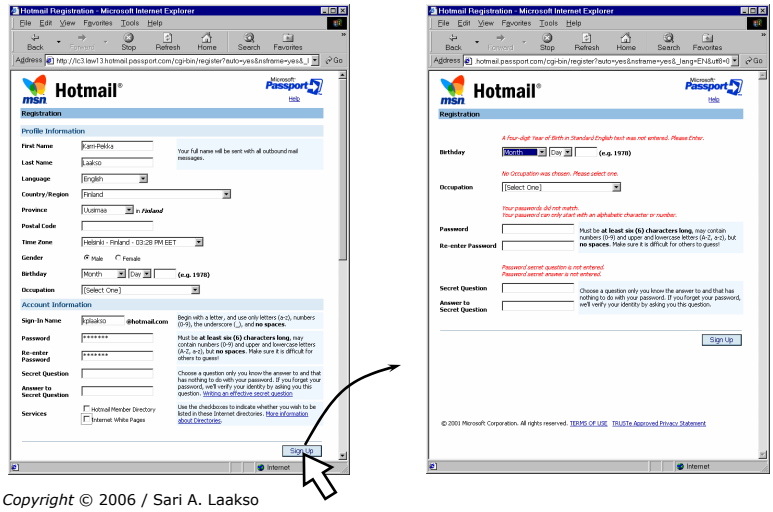
Esimerkiksi päivämäärien syöttövirheitä voidaan monissa tilanteissa estää kalenterikomponenteilla. Kuvan esimerkissä kuluva päivä (’tänään’) on merkitty ympyröinnillä ja käyttäjän valitsema päivä (tiistai) sinisellä taustalaatikolla.



Virheiden käsittely
Virheet kerralla kontekstissaan

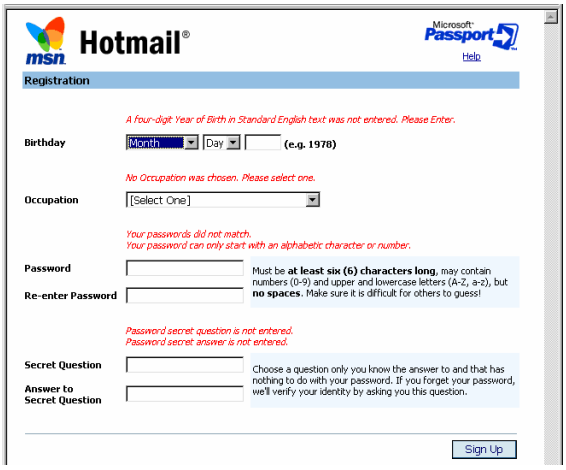
Kaikki virhekohdat kannattaa näyttää kerralla ja virheilmoitukset kontekstissaan virheellisen kentän välittömässä läheisyydessä.

Virheiden käsittely
Esimerkki 1: Rekisteröityminen 1/2



Virheiden käsittely
Esimerkki 1: Rekisteröityminen 2/2

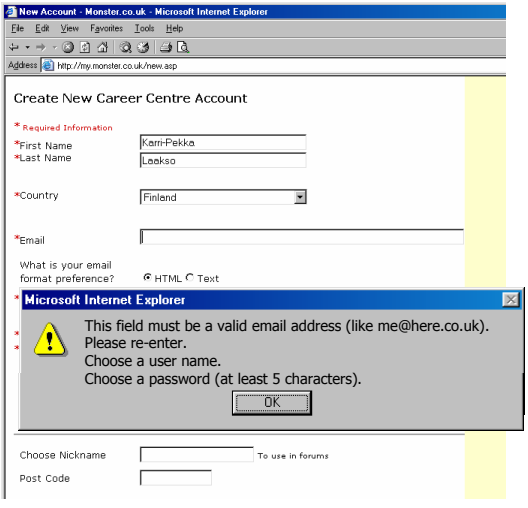
+ Kaikki virheet näkyvissä kerralla.
+ Virheilmoitukset näytetään samassa ikkunassa, jossa virheelliset kentätkin ovat.
+ Jokainen virheilmoitus on virheellisen kohdan lähellä.



Virheiden käsittely
Esimerkki 2: Omien tietojen syöttö

Ongelmia: Erillinen virheilmoikkuna.

- Virheilmo katoaa näkyvistä, kun käyttäjä kuittaa ikkunan OK:lla.
- Kuittauksen jälkeen käyttäjän täytyy etsiä virhekohdat lomakkeelta yksitellen ja yrittää pitää mielessään virheilmo-ikkunassa olevat tiedot virheellisistä kohdista ja virheiden syistä (työmuistin kuormitus).
- Käyttäjä joutuu navigoimaan alkuperäisen ikkunan ja virheilmoikkunan välillä.



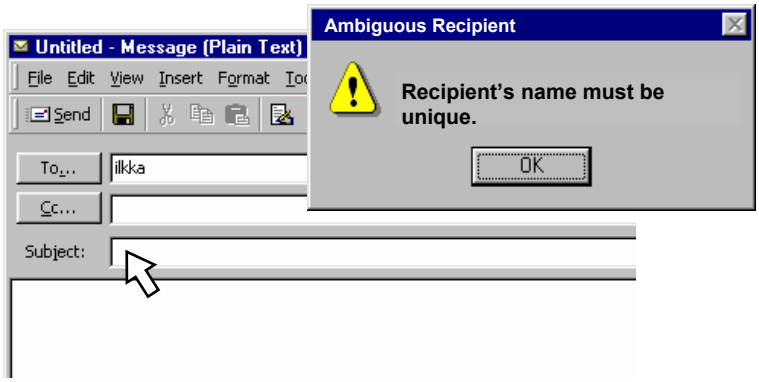
Virheiden käsittely
Virheilmoitusten jatkuva päivitys

Ohjelman on hyvä reagoida virheeseen välittömästi ja näyttää virheilmoitusta heti, kun mahdollista (jatkuva päivitys).

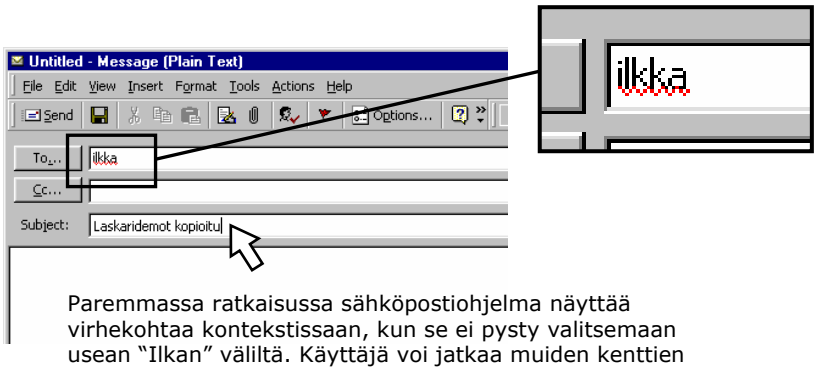
Käyttäjää ei kuitenkaan pidä pakottaa juuttumaan virhekohdan korjaamiseen. Hyvä käyttöliittymä pitää virheelliset kohdat näkyvillä ja antaa käyttäjälle vapauden korjata ne parhaaksi katsomassaan järjestyksessä.

Virheilmoitusten päivitys
Esim. 1: Vastaanottajan valinta 1/3

Ongelmia: Ohjelma näyttää virheilmoituksen erillisessä ikkunassa eikä päästä käyttäjää virheellisestä kohdasta eteenpäin, ennen kuin hän on korjannut sen.

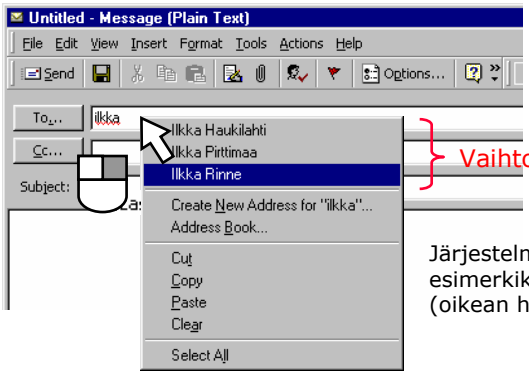


Virheilmoitusten päivitys
Esim. 1: Vastaanottajan valinta 2/3



Paremmassa ratkaisussa sähköpostiohjelma näyttää virhekohtaa kontekstissaan, kun se ei pysty valitsemaan usean "Ilkan" väliltä. Käyttäjä voi jatkaa muiden kenttien täyttämistä, vaikka vastaanottajakenttä olisikin vielä virheellisessä tilassa. (Sen sijaan virheilmon esitystavassa olisi vielä parantamista.)

Virheilmoitusten päivitys
Esim. 1: Vastaanottajan valinta 3/3

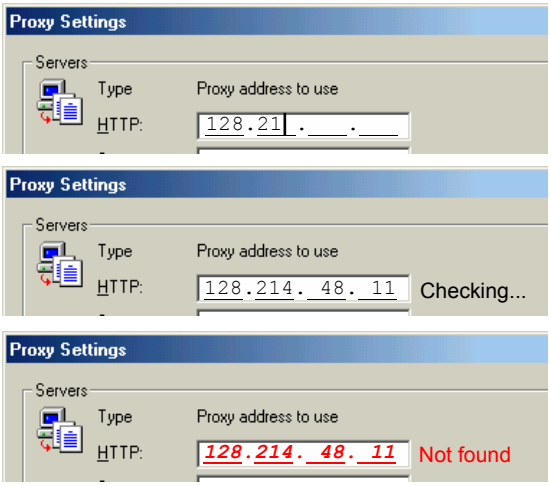


Vaihtoehdot nimelle "ilkka"

Järjestelmä voi näyttää vaihtoehdot esimerkiksi kontekstivalikossa (oikean hiirennapin valikossa).

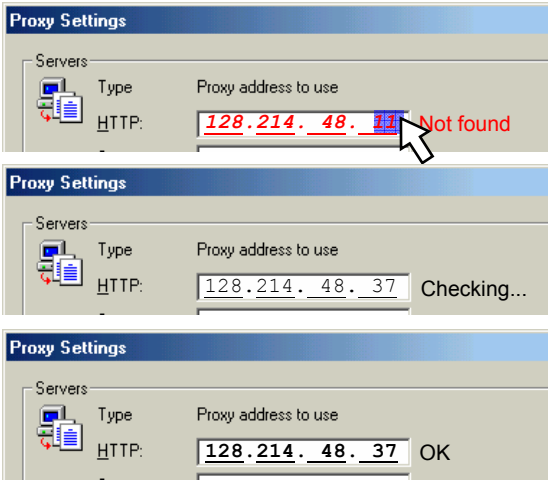
Virheilmoitusten päivitys
Esim. 2: IP-osoite 1/2

Tässä ohjelma tarkistaa käyttäjän syöttämän IP-osoitteen kelvollisuutta ja löytymistä jatkuvasti taustaprosessina ja näyttää tarkistuksen tuloksia käyttäjälle kontekstissaan.



Virheilmoitusten päivitys
Esim. 2: IP-osoite 2/2

Käyttäjä korjaa virheellistä kohtaa, ja ohjelma jatkaa jatkuvaa tarkistustaan taustaprosessina.



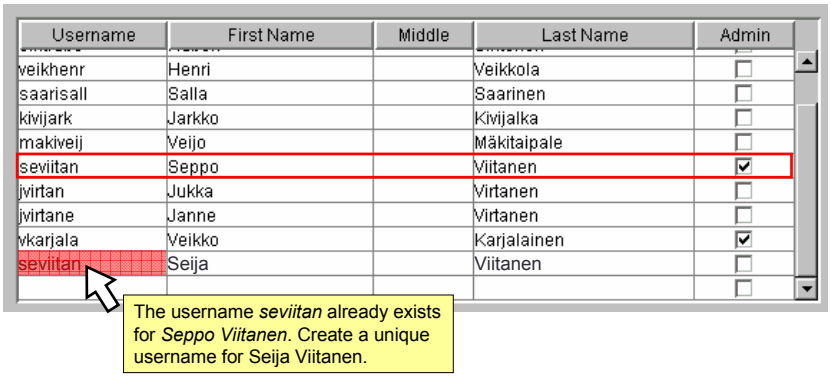
Virheilmoitusten päivitys
Esim. 3: Käyttäjätunnus 1/2

Toinen esimerkki jatkuvasta virheilmoituksen päivityksestä:

Username	First Name	Middle	Last Name	Admin
veikhenr	Henri		Veikkola	☐
saarisall	Salla		Saarinen	☐
kivijark	Jarkko		Kivijalka	☐
makiveij	Veijo		Mäkitaipale	☐
seviitan	Seppo		Viitanen	☑
jvirtan	Jukka		Virtanen	☐
jvirtane	Janne		Virtanen	☐
vkarjala	Veikko		Karjalainen	☑
seviitan	Seija		Viitanen	☐
				☐

Käyttäjä on syöttänyt viimeiselle riville käyttäjätunnukseksi *seviitan*. Järjestelmä antaa välittömästi palautetta ja korostaa virheellisen syötteen punaisella taustalla. Käyttäjä voi kuitenkin jatkaa työtään syöttämällä etunimen (Seija) ja sukunimen (Viitanen).

Virheilmoitusten päivitys
Esim. 3: Käyttäjätunnus 2/2

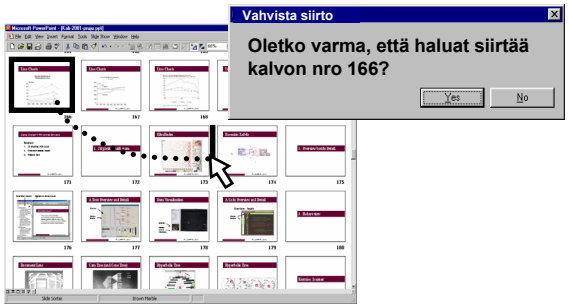


Käyttäjä vie fokuksen virheellisen kentän kohdalle.
Tässä järjestelmä näyttää virheilmoituksen tooltip-opasteen avulla.

Virheiden käsittely
Varmistuskysymykset

Varmistuskysymykset (*confirmation boxes*) teettävät käyttäjällä turhaa työtä silloin, kun virhettä ei ole tapahtumassa. Virhetilanteenkaan sattuessa varmistuskysymyksistä ei ole juurikaan apua, koska monissa tilanteissa käyttäjät ovat jo rutinoituneet kuittaamaan ne.

Ongelmalliset varmistusikkunat
Esim. 1: Powerpoint-kalvot

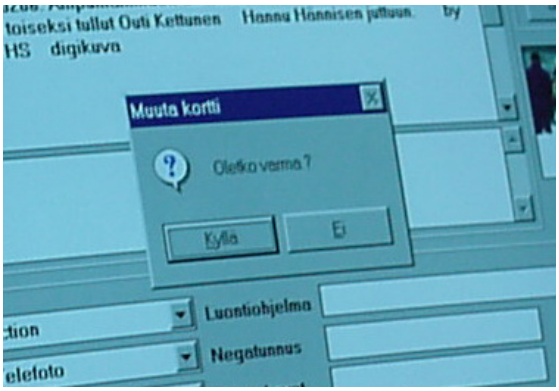


Ongelmia:

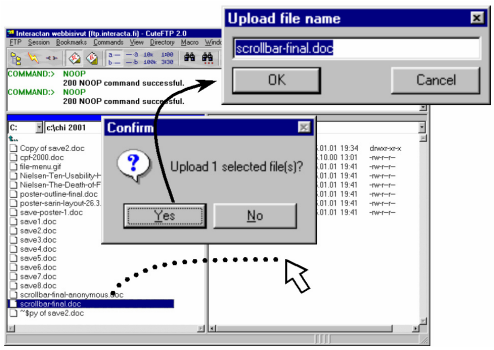
- Käyttäjät rutinoituvat kuittaamaan varmistuskysymyksiä (automatisoitunut prosessi) ja tekevät yhtä paljon virheitä kuin muutenkin. Kriittisissä tilanteissa niistä ei ole enää hyötyä [Cooper95, s. 445-446].
- Varmistuskysymykset teettävät käyttäjillä turhaa työtä.

Ongelmalliset varmistusikkunat
Esim. 2: Kuva-arkisto

Esimerkki Helsingin Sanomien kuva-arkiston varmistuskysymyksestä, jonka käyttäjä oli rutinoitunut kuittaamaan Enter-näppäimellä vilkaisematta edes ikkunan sisältöä. (Vrt. luennolla näytetty videopätkä.)



Ongelmalliset varmistusikkunat
Esim. 3: Tiedoston siirto



CuteFTP 2.0

Ongelmia: Tässä tiedoston siirrosta tulee peräti kaksi varmistusta peräkkäin: siirretäänkö todella tämä tiedosto ja tällä nimelläkö. Käyttäjä oppii pian siirtämään tiedostoja raahaamalla ja painamalla sitten ajattelematta 2xEnter.

Parempi ratkaisu olisi antaa käyttäjän siirtää tiedosto kyselemättä ja tarjota mahdollisuus...

- keskeyttämiseen ja perumiseen siirron aikana sekä
- tiedostonimen editointiin siirron jälkeen.

Varmistuskysymysten sijaan
Paremman ratkaisun aineksia

- Poista mahdollisimman moni varmistuskysymysikkuna, mutta suunnittele tilalle parempi ratkaisu (vrt. [Cooper95, s. 446-448]):
  - Anna käyttäjän editoida dataa. Älä kysy. Älä keskeytä.
  - Suunnittele ja otsikoi toiminnot niin, että käyttäjä tajuaisi jo ennakkoon toimenpiteen seurauksen eikä käynnistäisi erehdyksessä vääriä toimintoja.
  - Anna *palautetta jatkuvana päivityksenä jo ennakkoon,* ja tuo käyttäjän toimenpiteet *näkyviin*. Esimerkki: Käyttäjä yrittää siirtää poikkeuksellisen suurta rahasummaa yrityksen tililtä toiselle.
  - Tarjoa *perumistoiminto (undo)* aina, kun mahdollista. Varmistuskysymys on yleensä täysin turha sellaisten operaatioiden kohdalla, jotka on mahdollista perua perumistoiminnolla.

Sisältö

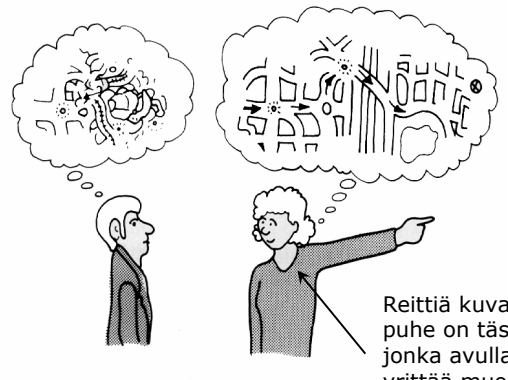
Kognitiivisia peruskäsitteitä

Mentaalimallit

Ongelmanratkaisutilanteessa ihminen rakentaa ja työstää mielessään mentaalimalleja (*mental models*), jotka ovat tiedoista ja tapahtumista jäsennettyjä tietorakenteita. Kirjallisuudessa mentaalimallista käytetään myös termiä ‘sisäinen malli’.

Tässä mentaalimallien muodostamista tarkastellaan muutaman esimerkin avulla.

Mentaalimallit
Esimerkki 1: Reitin kuvaaminen



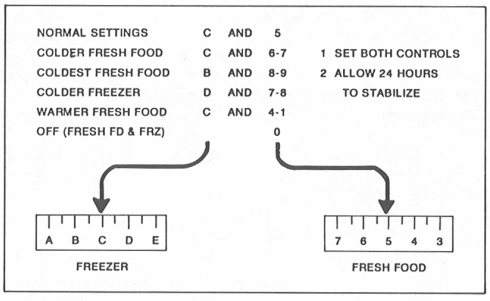
Kuva: [Preece94, s. 131]

Reittiä kuvailevan henkilön puhe on tässä **käyttöliittymä**, jonka avulla toinen kaveri yrittää muodostaa oikeaa mentaalimallia reitistä.

Mentaalimallit
Esimerkki 2: Normanin jääkaappi

Käyttäjä luo mentaalimallin laitteen tai ohjelman toiminnasta käyttöliittymän perusteella.

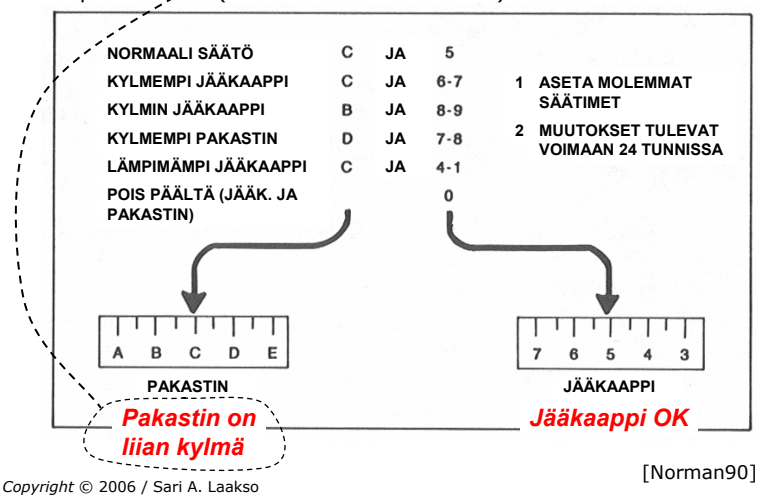
Esimerkki: Normanin jääkaappi-pakastin



[Norman90]

Esimerkki 2 (jatkuu)
Lähtötila

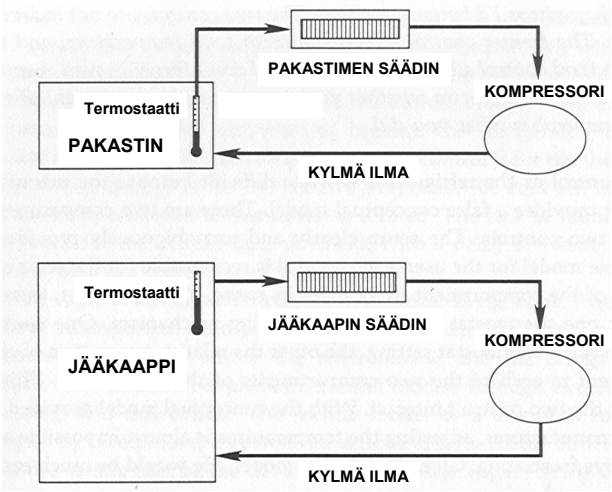
Tehtävä: Korjaa tämä ongelma säätämällä yhtä säädintä tai molempia säätimiä (PAKASTIN tai JÄÄKAAPPI) :



[Norman90]

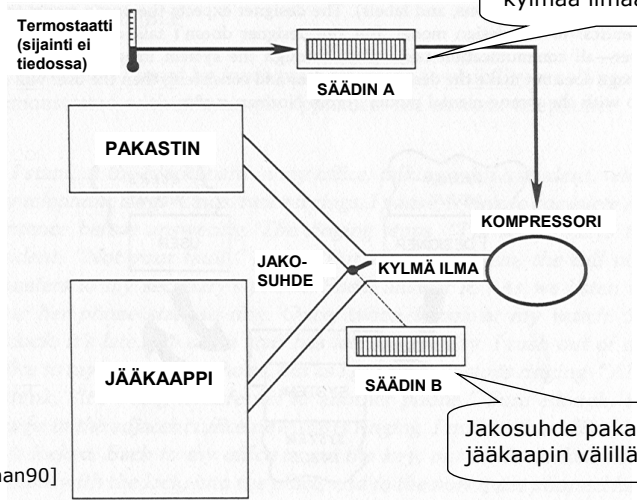
Esimerkki 2 (jatkuu)
Virheellinen mentaalimalli

Käyttöliittymä houkuttaa käyttäjää muodostamaan tämäntyyppisen virheellisen ja käyttökelvotto-man mallin:



[Norman90]

Esimerkki 2 (jatkuu)
Riittävä mentaalimalli

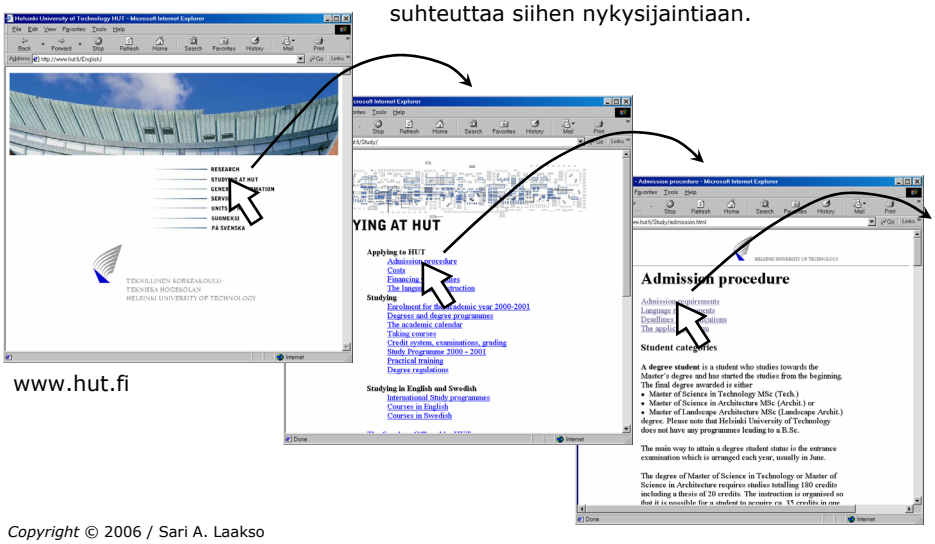


[Norman90]

Esimerkki 3a: TKK:n web-sivut

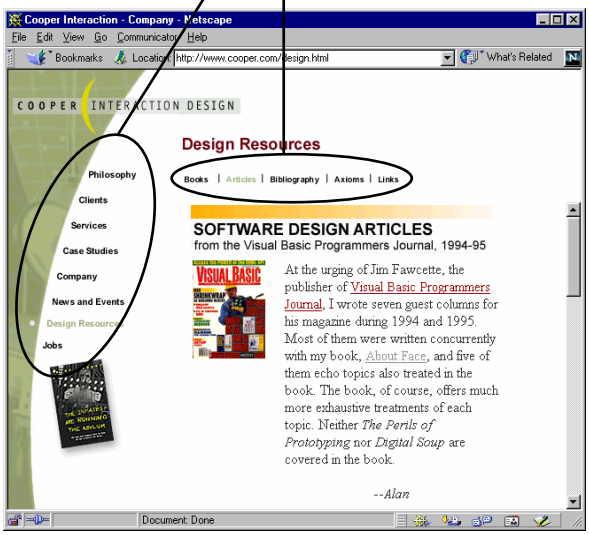
Ongelmia: Käyttäjän on työlästä luoda pala palalta mentaalimallia sivuston rakenteesta.

Kun hän on tehnyt valintansa etusivulta, päävalikko katoaa näkyvistä, eikä hän voi enää suhteuttaa siihen nykysijaintiaan.



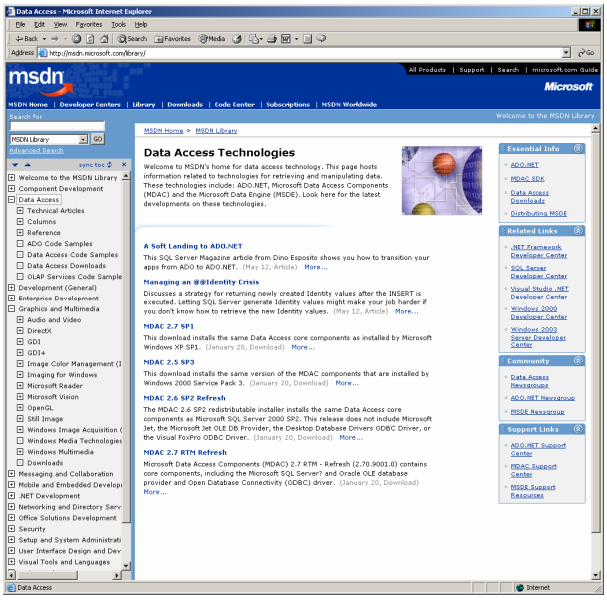
Esimerkki 3b: Cooperin entiset web-sivut

Näiden sivujen rakenne on paremmin näkyvillä:



Esimerkki 3c: MSDN Library

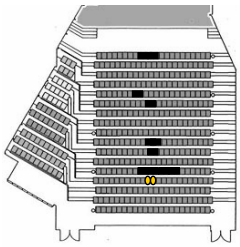
Näinkin laajan sivuston rakennetta on mahdollista näyttää navigointipalkissa:



Esimerkki 4:
Mentaalimalli tietosisällöstä

	Rivi	Paikka
☐	1	12
☐	1	13
☐	1	18
☐	2	3
☐	2	4
☑	2	5
☑	2	6
☐	2	22
☐	2	23

- Tampere-talon istumapaikkaesimerkissä huonompi käyttöliittymä pakotti käyttäjän tekemään paljon työtä tarkoituksenmukaisen mentaalimallin (mielikuvan) muodostamiseksi tarjolla olevasta tietosisällöstä: salin rakenteesta ja istumapenkkien riveistä sekä mm. vapaiden paikkojen sijainnista.
- Parempi käyttöliittymä säästää käyttäjältä tiedon rakenteen hahmottamiseen liittyvän 'miettimistyön' ja tarjoaa rakenteen valmiina (tässä salikartan kuvan avulla).
- Nyt käyttäjä voi käyttää miettimistyötä käyttötilanteeseensa sopivan paikan valitsemiseen sen sijaan, että hän yrittää hahmottaa salin rakennetta ja penkkirivejä.

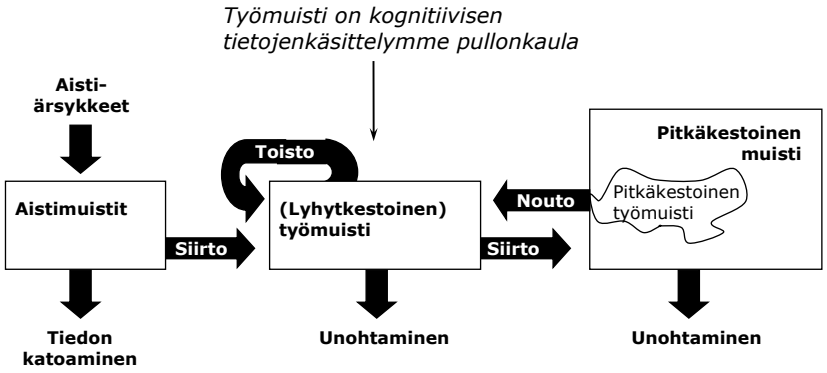


Kognitiivisia peruskäsitteitä
Työmuisti

Työmuisti (*working memory*) on ihmisen tiedonkäsittelyn keskus. Työmuistissa käsitellään sekä aistien kautta tulevaa tietoa (näkö, kuulo) että pitkäkestoisesta muistista haettua tietoa, ja tuloksista osa tallennetaan pitkäkestoiseen muistiin.

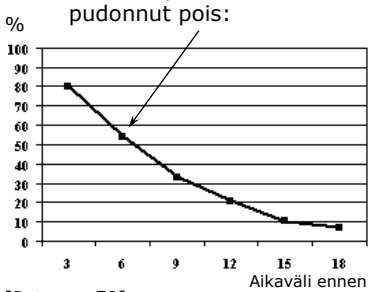
Työmuisti on keskeinen pullonkaula ihmisen ongelmanratkaisussa ja tiedon työstämisessä: ongelmina ovat lyhyt tallennusaika ja rajallinen kapasiteetti.

Työmuistin kuormittuminen
Ihmisen muistijärjestelmä



Työmuistin ongelmat
Tallennusaika ja kapasiteetti

- **Lyhyt tallennusaika**
  - Tiedot pysyvät työmuistissa vain vähän aikaa, alle 20 sekuntia. Jo noin 6 sekunnin kuluttua puolet tiedoista on pudonnut pois:



[Peterson59]

- **Rajallinen kapasiteetti**
  - Alunperin arvioitu 7±2 mieltämisyksikköä (*chunk*) [Miller56],
  - myöhemmin uudempana arviona mm. **noin 4 mieltämisyksikköä** [Broadbent75, Cowan01].

Mieltämisyksikkö: Tiedon palanen tai sääntö, jonka avulla tietoa rakennetaan

Työmuisti
Esimerkki: Mieltämisyksiköt (chunks)

NTH EDO GSA WTH ECA TRU

- Tämä data voi viedä jopa 18 mieltämisyksikköä. Jotkut ihmiset yrittävät muistaa kokonaisia kolmen kirjaimen ryhmiä, minkä tuloksena kirjainjono voi ryhmittyä esim. 6 mieltämisyksiköksi.

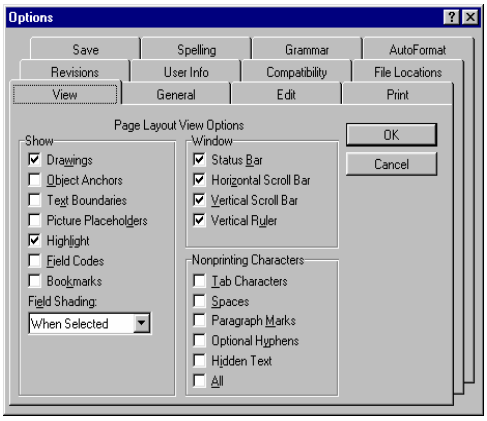
THE DOG SAW THE CAT RUN

- Tässä dataa on yhtä paljon kuin edellä (18 kirjainta, jotka ovat täsmälleen samat), mutta tämä on mahdollista tulkita noin 1-3 mieltämisyksiköksi.

Työmuisti
Esim. 1a: Moniriviset välilehdet

Ongelmia: Välilehtien rivit vaihtavat paikkaa käyttäjän valinnan jälkeen, mikä tuo lisää tiedonkäsittelytarvetta myös työmuistiin.

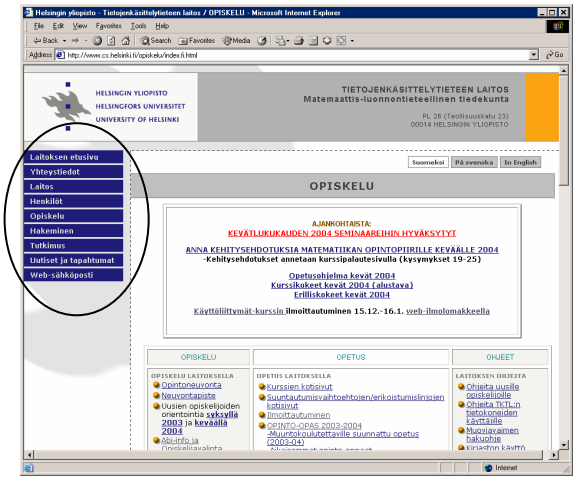
Käyttäjän työmuistin kapasiteetti ja tallennusaikakin alkavat ylittyä hänen yrittäessään pitää mielessä, mitkä välilehdet hän jo katsoi ja mistä hän löysi tai ei löytänyt tavoitteeseensa mahdollisesti soveltuvia asetuksia.



MS Word 97

Työmuisti
Esim. 2: Välilehdet ilman korostusta

Ongelmia: Valittuna olevaa sivua ei ole korostettu navigointipalkista, minkä seurauksena käyttäjän on yritettävä pitää tätä tietoa mielessään (sen sijaan, että ohjelma pitäisi tietoa näkyvillä koko ajan).



www.cs.helsinki.fi

Työmuisti
Esim. 3: Opastus kontekstissaan

Ben Shneidermanin **PhotoFinder**

Käyttäjän ei tarvitse yrittää pitää opasteiden sisältöjä työmuistissaan siirtyessään erillisten opastussivujen ja varsinaisen ohjelman välillä, vaan tässä hän näkee ohjetekstit samalla, kun on tekemässä varsinaista tehtäväänsä. Työmuistin kapasiteettia vapautuu opastetekstin ja käyttäjän tehtävän väliseen tulkintaan ja ongelmanratkaisuun.

Sisältö

Käyttöliittymän opittavuus vs. tehokkuus

Opittavuutta vaikuttaa olevan helpompi lisätä tehokkaaksi suunnitellun käyttöliittymän päälle kuin päinvastoin.

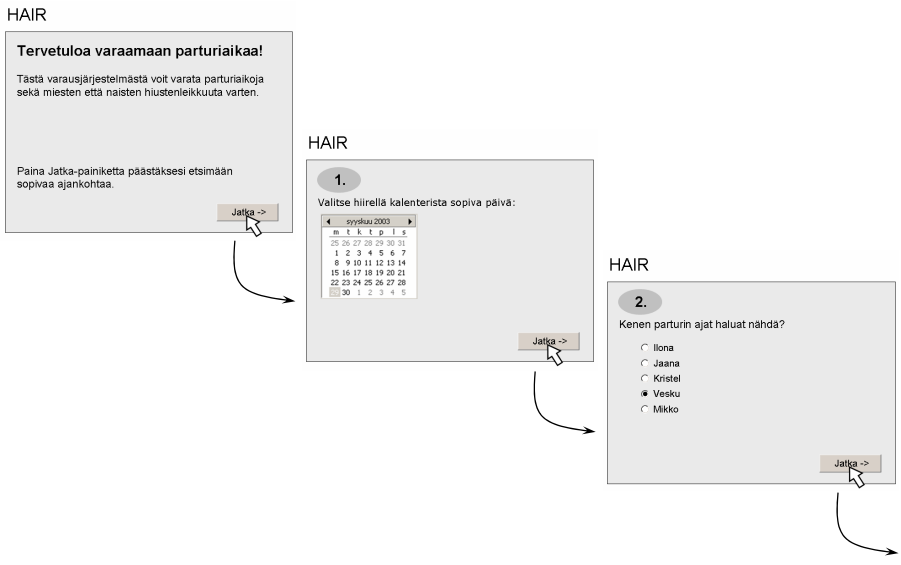
Kannattaa siis ensin keskittyä tavoittelemaan tehokkaita ratkaisuja ja parantaa niiden opittavuutta vasta lopuksi. Opittavuutta on myös helppoa testata käyttäjien avulla (käytettävyystestaus).

Käyttöliittymän laatiminen
Ensin tehokkuus

- Monissa tapauksissa käytön tehokkuus (*task efficiency*) ja ratkaisun opittavuus/ymmärrettävyys konfliktoivat keskenään, eikä molempia voida saavuttaa optimaalisesti [Lauesen01].
- Tavoittele **ensin käytön tehokkuutta**.
  - Piirrä käyttötilanteessa tarvittavat tiedot samaan ikkunaan kerralla näkyviin.
  - Pyri siihen, että käyttäjä voisi hoitaa aina yhden käyttötilanteen samassa paikassa siirtyilemättä eri puolille järjestelmää.
- Lisää tehokkaaseen käyttöliittymäratkaisuun vasta suunnittelun **lopuksi opittavuutta**:
  - Yritä saada käyttöliittymän rakenteesta ja toimintalogiikasta 'itsestään selvä' (vrt. esim. itsestään selvät ovet). Lisää visuaalisia vihjeitä ja paranna otsikkotekstejä sekä näytön osien ryhmittelyä.

Esimerkki: Parturin ajanvarausjärjestelmä

Käyttötilanne: Petterin seuraava käynti Veskulla

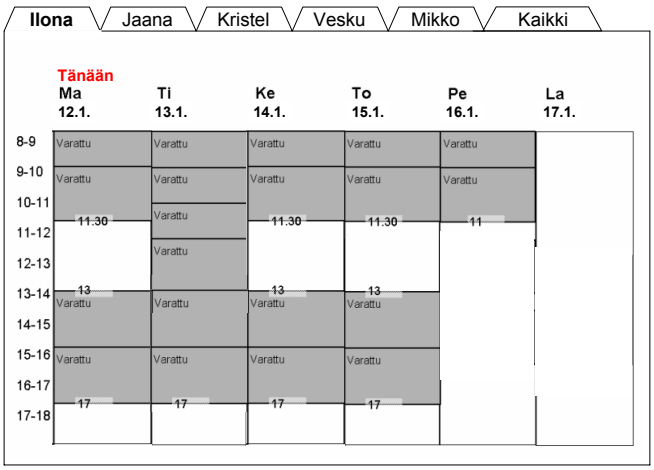


Esimerkki: Parturin ajanvarausjärjestelmä

Käyttötilanne: Petterin seuraava käynti Veskulla

Tarkasteltava viikko

Viikon varaustilanne



Varaukset p. (09) 724 2268

Käyttöliittymän laatiminen

Opittavuus alisteista tehokkuudelle

- Hyvin suunnitellun **tehokkuuden** (tarvittavat tiedot näkyvillä kerralla ja suoraviivaiset interaktiotavat) päälle pystyt aina lisäämään **opittavuutta**, mutta päinvastainen ei onnistu yhtä helposti.
  - Huonon tehokkuuden korjaaminen yleensä ‘rikkoo’ koko käyttöliittymäratkaisun, ja sen jälkeen myös opittavuus on suunniteltava alusta lähtien.
  - Opittavuuden lisääminen tehokkaaksi suunniteltuun käyttöliittymäratkaisuun ei yleensä heikennä ratkaisun tehokkuutta tai se heikentää sitä vain todella vähän.
- Tällä kurssilla painotetaan käyttöliittymäratkaisujen tehokkuutta, koska sen suunnittelu on vaikeaa ja kannattaa tehdä ensin. Opittavuuskin on tärkeää, mutta sitä voidaan parantaa vielä testausvaiheessakin.

Lähteitä

Bias91	Bias R. G., Walkthroughs: efficient collaborative testing. IEEE Software, Vol. 8, No. 5, 1991, s. 94-95.
Broadbent75	Broadbent D., The magic number seven after twenty years. Teoksessa Kennedy R., Wilkes A. (toim.), Studies in long term memory. Wiley, New York, 1975, s. 253-287.
Card99	Card S. K., Mackinlay J., Shneiderman B. (toim.), Readings in Information Visualization: Using Vision to Think. Morgan Kaufmann, San Francisco, CA, USA, 1999.
Carroll94	Carroll J. M., Making Use A Design Representation. Communications of the ACM, Vol. 37, No. 12, December 1994, s. 29-35.
Carroll00	Carroll J.M., Making Use. Scenario Based Design of HumanComputer Interactions. The MIT Press, USA, 2000.
Cato01	Cato, J. User-Centered Web Design. Pearson Education Ltd, 2001.
Cooper95	Cooper A., About Face: The Essentials of User Interface Design. IDG Books, USA, 1995.
Cooper03	Cooper A., Reimann R. M., About Face 2.0. The Essentials of Interaction Design. Wiley Publishing, Inc., Indianapolis, Indiana, 2003.
Cowan01	Cowan N., The magical number 4 in short-term memory: A reconsideration of mental storage capacity. Behavioral and Brain Sciences, 24, 2001, s. 87-185. www.bbsonline.org/documents/a/00/00/04/46/bbs00000446-00/bbs.cowan.html
Dubberly87	Dubberly H., Mitch D., The Knowledge Navigator. Apple Computer, videonauha. Esiintyy videolla Myers B. A. (toim), HCI'92 Special Video Program.
Go04	Go K., Carroll J. M., The Blind Men and the Elephant: Views of Scenario-Based System Design. interactions, November/December 2004, s. 44-53.

Goldstein02 Goldstein E. B.,
Sensation and Perception.
Wadsworth Publishing, 6th edition, 2002.

Hackos98 Hackos J., Redish J. C.,
User and Task Analysis for Interface Design.
John Wiley & Sons, USA, 1998.

Jacobson92 Jacobson I. et al.,
Object-oriented software engineering: a use case driven approach.
Addison-Wesley, 1992.

Laakso00 Laakso S. A., Laakso K.-P., Saura A.,
Improved Scroll Bars.
ACM CHI 2000 conference proceedings, Extended Abstracts, s. 97-98.
www.cs.helsinki.fi/u/salaakso/papers/CHI2000-Improved-Scrollbars.PDF

Laakso02 Laakso S. A.,
Käyttöliittymien arviointimenetelmät.
Käyttöliittymät II –kurssin luentomateriaalia, Helsingin yliopisto, Tietojenkäsittelytieteen laitos, päivitetty 21.10.2002.
www.cs.helsinki.fi/u/salaakso/kl2-2002/lahteet/arviointi.html

Laakso04 Laakso S. A., Laakso K.-P.,
Hyvän käyttöliittymän varmistaminen GUIDe-prosessimallilla.
Helsingin yliopisto, Tietojenkäsittelytieteen laitos, julkaisematon artikkeli, päivitetty 3.10.2004.
www.cs.helsinki.fi/u/salaakso/papers/GUIDe-suomeksi.pdf
www.cs.helsinki.fi/u/salaakso/papers/GUIDe-suomeksi.html

Lauesen01 Lauesen S., Harning M.B.,
Virtual Windows: Linking User Tasks, Datamodels, and Interface Design.
IEEE Software, July/August 2001, s. 67-75.
www.itu.dk/people/slauesen/Papers/VirtualWindowsIEEE.pdf

Lauesen02 Lauesen S.,
Software Requirements. Styles and Techniques.
Pearson Education Ltd, 2002.

Lauesen05 Lauesen S.,
User Interface Design. A Software Engineering Perspective.
Pearson Education Ltd, 2005.

Lewis97 Lewis C., Wharton C.,
Cognitive Walkthroughs.
Teoksessa Helander M., Landauer T., Pradhu P. (toim.), Handbook of Human-Computer Interaction. Elsevier Science B.V., Netherlands, 1997, s. 717-732.

Miller56 Miller G. E.,
The Magical Number Seven, Plus or Minus Two: Some Limits on Our Capacity for Processing Information.
The Psychological Review, 1956, vol. 63, s. 81-97
www.well.com/user/smalin/miller.html

Nielsen93 Nielsen J.,
Usability Engineering.
Academic Press, New York, 1993.

Nielsen94 Nielsen J.,
Heuristic evaluation.
Teoksessa Nielsen J., Mack R.L. (toim.), Usability inspection methods.
Wiley, New York, USA, 1994, s. 25-62.

Nielsen00 Nielsen J.,
Why You Only Need to Test With 5 Users.
Alertbox, March 19, 2000.
www.useit.com/alertbox/20000319.html

Nielsen03 Nielsen J.,
Usability 101: Introduction to Usability.
Alertbox, August 25, 2003.
www.useit.com/alertbox/20030825.html

Norman90 Norman D.,
The Design of Everyday Things.
Doubleday, New York, 1990.
(Alkuperäinen painos nimellä *The Psychology of Everyday Things*, Basic Books, New York, 1988; suomenkielinen versio nimellä *Miten avata mahdottomia ovia*)

Peterson59 Peterson L. R., Peterson M. J.,
Short-term retention of individual verbal items.
Journal of Experimental Psychology, Vol. 58, 1959, s. 193-198.

Preece94 Preece J.,
Human-Computer Interaction.
Addison-Wesley, Wokingham, UK, 1994.

Rubin94 Rubin J.,
Handbook of Usability Testing: How to Plan, Design, and Conduct Effective Tests.
John Wiley & Sons, 1994.

Sears97 Sears A.,
Heuristic Walkthroughs: Finding the Problems Without the Noise.
International Journal of Human-Computer Interaction, Vol. 9, No. 3, 1997, s. 213-234.

Shneiderman98 Shneiderman B.,
Designing the User Interface.
Addison-Wesley, 3rd edition, 1998.

Shneiderman00 Shneiderman B., Kang H.,
Direct Annotation: A Drag-and-Drop Strategy for Labeling Photos.
International Conference on Information Visualisation (IV2000), London, England, 2000.
www.cs.umd.edu/hcil/photolib/paper/DirectAnnotation7.doc

Toyoda99 Toyoda M., Shibayama E.,
Hyper Mochi Sheet: A Predictive Focusing Interface for Navigating and Editing Nested Networks through a Multi-focus Distortion-Oriented View.
ACM CHI'99 conference proceedings, 1999, s. 504-511.

Tufte83 Tufte E. R.,
The Visual Display of Quantitative Information.
Graphics Press, USA, 1983.

Tufte90 Tufte E. R.,
Envisioning Information.
Graphics Press, USA, 1990.

Ware00 Ware C.,
Information Visualization: Perception for Design.
Morgan Kaufmann, San Francisco, CA, 2000.

Williamson92 Williamson C., Shneiderman B.,
The dynamic HomeFinder: Evaluating dynamic queries in a real-estate information exploration system.
ACM SIGIR'92 conference proceedings, 1992, s. 338-346.