Graafisen käyttöliittymän suunnittelu ja käytettävyys tietojärjestelmän kehittämisessä

Tietojärjestelmän kehittäminen – kurssin raportti

TURUN YLIOPISTO
Informaatioteknologian laitos
Tietojärjestelmätiede
Jyri Lehtonen, Juha Sinisalo
Syksy 2007

TURUN YLIOPISTO
Informaatioteknologian laitos

JYRI LEHTONEN (72039)
JUHA SINISALO (71654)

Tutkimusraportti

Tutkimus, 17 s, 2 liitesivua
Tietojärjestelmätiede
Lokakuu 2007

Raportin alkuosan tarkoituksena on määritellä, mitä on tietojärjestelmien kehittäminen. sekä rakentaa määritelmästä pintaraapaisu aiheen kokonaiskuvaan. Tietojärjestelmän kehityksen yhteydessä käydään läpi siihen liittyvät oleelliset termistöt ja mallit, mukaan lukien järjestelmän elinkaari ja lisäksi käsittelemme ongelmien hallintaa yleisellä tasolla.

Alkuosion määrittelyn jälkeen on käsitelty tutkimuksen kannalta oleelliset tekijät, käytettävyys, ihmisen ja tietokoneen vuorovaikutus, graafisen käyttöliittymän suunnittelu ja luova suunnittelu. Näiden päätekijöiden on oltava tiukasti yhteydessä toisiinsa raportin aiheen kokonaisvaltaisen kuvan saamiseksi. Raportti keskittyy PIOCO:n tasoista eniten IO tasolle, raapaisten hieman P ja CO tasoja muutamilla tilannekohtaisilla kuvauksilla.

Raportti esittelee metodologian, Multiview. Tämän ja creative desining pohjalta esitetään kotitaloudenhallinta esimerkki, jossa käytetään raportissa esitettyjä aihepiirejä.

Avainsanat: tietojärjestelmän kehittäminen, metodologia, graafinen käyttöliittymä, käytettävyys, Human-Computer Interaction, graafisen käyttöliittymän suunnittelu.

SISÄLLYSLUETTELO

1 JOHDANTO

Tietojärjestelmä on tiettyä toimintaa palveleva kokonaisuus. Tämä kokonaisuus koostuu tiedoista, toimintaohjeista, ohjelmista, tietojenkäsittely- ja tiedonsiirtolaitteista sekä näitä laitteita ja ohjelmia käyttävistä ihmisistä. Tietojärjestelmä voidaan jakaa kahteen osa-alueeseen, automaattiseen ja manuaaliseen. Automaattisella tarkoitetaan järjestelmää, jossa tietojenkäsittelyä hoitavat tietokoneet ja ohjelmat. Kun taas, manuaalisessa järjestelmässä tietoja käsittelevät ihmiset. Nykyaikaisissa tietojärjestelmissä molemmat osat ovat mukana toiminnassa.

Tietojärjestelmien kehittäminen on osa liikemaailmaa. Kehittäminen auttaa organisaatiota suuntautumaan paremmin tavoitteisiinsa, mahdollistaa uudet toiminnot tai kehittää jo olemassa olevia toimintamalleja. Kaiken kehittämisen tarkoituksena on organisaation toiminnan parantaminen. Kehittämisen kohteena ovat ihmiset, teknologia tai organisaation toiminnot. Näitä on yleensä vaikeata erottaa toisistaan kehityksessä. Moderneissa kehityshankkeissa kehitetään pääsääntöisesti teknologiaa, mutta sen kehittäminen vaikuttaa väistämättä myös ihmisiin ja toimintoihin.

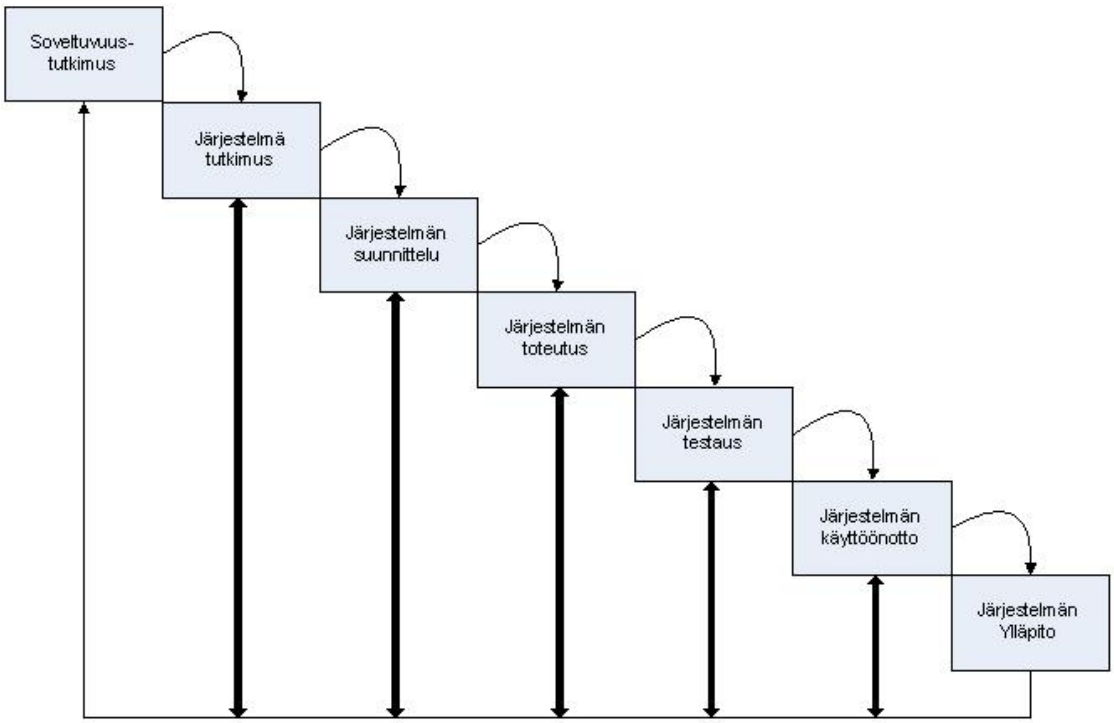
Metodologia määritellään yleisellä tasolla sarjana suositeltavia vaiheita ja toimintoja, joita tulee seurata tietojärjestelmää kehitellessä. Metodologia on siis kokoelma toimintoja, tekniikoita, työkaluja ja dokumentteja jotka auttavat järjestelmien kehitystyössä.

Tietojärjestelmän kehittäminen on nuori ala, mutta sen kehityksen voidaan jakaa neljään osa-alueeseen. Ensimmäisessä vaiheessa (Pre Metodology era) teknologian kehitys oli tärkeätä, ja ongelmien ratkaisu oli teknistä. Tämä muuttui näiden varhaisten vuosien jälkeen teknologian hurjan kasvun myötä. Toisessa vaiheessa (Early Metodology era) tietojärjestelmän elinkaaren käsite otettiin mukaan määritelmään. Elinkaari kuvastaa toimintaa, jossa edeltävä vaihe on oltava suoritettuna ennen seuraavaan siirtymistä. Kolmannessa vaiheessa (Metodology Era) tietojärjestelmän kehitys jakautui kahteen. Yritykset ja organisaatiot loivat käytännön ja käytännöstä rakennettiin tietojärjestelmien

kehitysteoria. Viimeinen vaihe on tämänhetkinen tilanne. Nykyään organisaatiot kehittävät uusia metodologioita, tai hylkäävät kokonaan metodologioiden käytön (Avison D, Fitzgerald G, 2003, VII). Tämän kehityksen voimme tiivistää seuraavasti. Oli aika, jolloin tietojärjestelmän kehitys sisälsi ohjelmoijan, joka kirjoitti koodia ongelman ratkaisuksi. Nykyään järjestelmämme voivat olla niin kompleksisia, että tarvitaan erilaisia tiimejä, (analyytikot, ohjelmoijat, testaajat, käyttäjät, jne.) joiden on tultava toimeen keskenään luodakseen miljoonia rivejä koodia, jotka tukevat yhtiöitämme.

1.1 Suunnittelun elinkaari

Tietojärjestelmän kehittämisen elinkaarta kuvataan SDLC (System Development Life Cycle) mallilla, jota myös kutsutaan vesiputousmalliksi. Vesiputousmallissa tietojärjestelmän kehittämisprosessi on jaettu selkeisiin vaiheisiin. Vaiheet toteutetaan järjestyksessä. Mallista löytyy useita eri versioita, mutta perusvaiheet ovat kaikissa samat. Seuraavaksi esittelemme seitsemän vaiheisen vesiputousmallin.



Kuva 1: Klassinen elinkaari

1.1.1 Esitutkimus

Esitutkimus vaiheessa kartoitetaan yrityksen tai organisaation tietojärjestelmä tarpeet. Mitä uuden järjestelmän pitäisi tehdä ja mitkä ovat sille asetetut tavoitteet? Selvitetään myös edellytykset hankkeen toteuttamiselle ja miksi hankkeeseen pitäisi ryhtyä. Näiden kysymysten vastauksien perusteelta päätetään ryhdytäänkö hankkeeseen vai ei.

1.1.2 Järjestelmäanalyysi

Seuraavassa vaiheessa selvitetään mitä rakennettavan järjestelmän tulee tehdä. Analysoidaan esitutkimus vaiheessa tunnistettuja asiakasvaatimuksia ja johdetaan niistä järjestelmän toiminnallinen määrittely. Järjestelmän rajoitteiden määrittely on myös analyysivaiheen keskeinen tehtävä ja järjestelmäanalyysi vaiheessa luodaan myös loogisen tason kuvaus kohdejärjestelmän toiminnoista.

1.1.3 Suunnittelu

Suunnitteluvaiheessa tarkoituksena on muuntaa analyysivaiheessa tehty toiminnallinen määrittely teknilliseksi määrittelyksi, joka sisältää kuvauksen järjestelmän toteutuksesta. Suunnitteluvaihe jaetaan yleensä arkkitehtuuri- ja moduulisuunnitteluun. Arkkitehtuurisuunnittelussa järjestelmä jaetaan moduuleihin ja moduulisuunnittelussa suunnitellaan moduulien sisäiset rakenteet.

1.1.4 Toteutus

Toteutus on suunnittelun kanssa hyvin läheinen vaihe. Kaikki tärkeimmät ohjelmiston rakennetta ja toimintaa määrittelevät päätökset on jo tehty aikaisemmissa vaiheissa. Näin ollen toteutus on varsin suoraviivainen vaihe, jossa ohjelmisto tai sen osat toteutetaan ohjelmointikielillä tai sovelluskehittimellä, ja valmiit ohjelmistomoduulit integroidaan toimivaksi kokonaisuudeksi.

1.1.5 Testaus

Ennen järjestelmän käyttöönottoa tulee järjestelmän toiminta testata. Testivaiheessa kartoitetaan järjestelmän eri osien toimivuus ja katsotaan vastaako järjestelmä suunnittelu- ja analyysivaiheissa luotuja määrittelyjä ja tavoitteita.

1.1.6 Käyttöönotto

Testauksen jälkeen vuorossa on järjestelmän käyttöönotto. Käyttöönotossa tulee ottaa huomioon muun muassa seuraavia seikkoja: mahdollisten olemassa olevien tietojen ja tietokantojen siirtäminen uuteen järjestelmään sekä mahdollisten rinnakkaisten järjestelmien olemassaolo. Henkilöstön ja ylläpitohenkilökunnan kouluttaminen on myös yksi keskeisistä tehtävistä.

1.1.7 Ylläpito

Ylläpito vaiheessa keskitytään järjestelmän ylläpitoon muun muassa virheiden korjauksella, jatkokehityksellä ja toimintakyvyn ylläpidolla. Ylläpitovaihe kestää käytännössä järjestelmän koko eliniän.

1.2 Prototyyppilähestymistapa

Vesiputousmallin rinnalta löytyy useita muita malleja. Näistä yleisimmässä käytössä on prototyyppilähestymistapa. Prototyyppilähestymistapaa käytetään tilanteissa, joissa järjestelmän tilaaja haluaa nopeasti päästä kokeilemaan järjestelmän toimintoja, mutta ei välttämättä osaa määritellä haluamaansa järjestelmää. Järjestelmästä pyritään tuottamaan mahdollisimman nopeasti karkea prototyyppi, joka sisältää järjestelmän päätoiminnot, mutta ei juurikaan yksityiskohtia. Tämän jälkeen asiakas arvioi prototyypin ja palautteen perusteelta tehdään parannuksia. Kun prototyyppi vastaa asiakkaan toiveita, toteutetaan varsinainen järjestelmä prototyypin pohjalta.

Prototyyppimallin ongelmana ovat suuret resurssikustannukset ja pelkistyneisyytensä takia prototyyppi ei välttämättä paljasta kaikkia järjestelmässä piileviä ongelmia sekä mahdolliset huonot ratkaisut aikaisemmissa prototyypeissä saattavat päästä mukaan lopulliseen järjestelmään. (Avison D, Fitzgerald G, 1995, 76-80)

1.3 Ongelmienhallinta

On osoitettu, että ei ole sellaista viisastenkiveä, joka ratkaisisi edes merkittävän osan tietojärjestelmänkehityksen ongelmista. Toisaalta systemaattisella toimintatapojen kehittämisellä voidaan päästä erinomaisiin tuloksiin.

Ohjelmistoprosessi on kokonaisuus, joka kattaa koko elinkaaren. Tätä menetelmää voidaan käyttää eräänä ongelmienhallintajärjestelmänä. Se tarjoaa tietojärjestelmänkehityksen kehityshankkeiden hallinnalle viitekehyksen, jonka avulla laajat hankkeet kyetään jakamaan pienempiin osiin. Kyseinen toiminta mahdollistaa kehityshankkeiden, tehtävien ja resurssien hallinnan. Tällöin elinkaareen liitetään laadunvarmistus, dokumentointi ja riskienhallinta. (Avison D, Fitzgerald G, 2003, VII)

1.4 Tutkimusalueen rajaus

Miksi monet käyttöliittymät ovat niin hankalia käyttää? Miksi nämä käyttöliittymät ovat vaikeita suunnitella ja toteuttaa? Miten voisi olla mahdollista parantaa käyttöliittymien suunnittelua? Näitä kysymyksiä pohditaan raportissa eri näkökulmista. Raportissa on myös pyritty löytämään kehittämisen haasteet ja mahdollisuudet.

Tutkimuksemme kohteena ovat sovellusten graafiset käyttöliittymät ja niiden kehitys kurssin viitekehyksien sisällä. Keskitymme raportissamme yleisesti käytettävyyden ja ihmisen ja koneen välisen kanssakäymiseen liittyviin kysymyksiin, PIOCO mallin IO tasolla. Nostamme luovuuden käsiteltäväksi valittujen aineistojen kohdalla. Käyttöliittymäkeskeiset asiat eivät tule esille monissa tutkituissa metodologioissa ja menetelmissä. Koska käyttöliittymä ”kytkee” ihmisen tietokoneeseen, on sen merkitys kiistaton.

2 ARTIKKELIT JA RAPORTIN NÄKÖKULMA

2.1 Esittely

2.1.1 Käytettävyys yleisesti

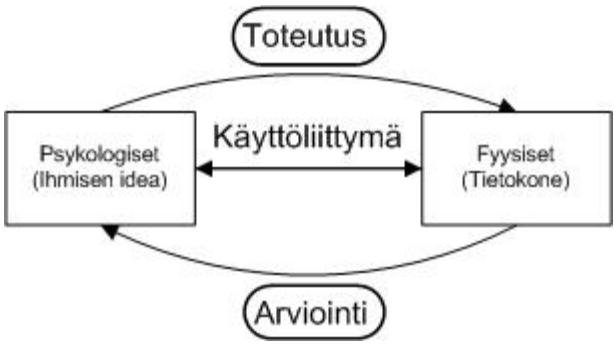
Käytettävyys on osa ihmisen ja ympäristön vuorovaikutus tutkimusta. Yleisesti käytettävyydessä tutkitaan ihmisen ja koneen tai työvälineen välistä yhteyttä. Käytettävyys on artefaktin, ihmisen ja näiden interaktion määritelmä. Mukaan voidaan ottaa myös tutkimus itse toiminnasta, jolloin käytettävyyden käsite on laajempi. Koska käytettävyyttä on vaikea mitata, sitä pitää kuvailla. Tämä johtaa ominaisuuksien hakuun, joista syntyy johtopäätöksiä. Hyvä käytettävyys on määritelty miellyttävyyden ilmenemisellä, toimintojen helppoutena ja turvallisuutena. Jos artefaktista saa halutun lopputuloksen ulos ilman, että käyttäjä joutuu ongelmanratkaisu tilanteeseen, se luo miellyttävyyttä. Korkea käytettävyys on määritelty siten, että artefakti ei aiheuta riskejä, eikä sen tavoitepohjainen peruskäyttö vie liikaa aikaa. (Norman D, 1990, III)

Optimoitu käyttöliittymä vaatii systemaattisen lähestymisen suunnitteluprosessiin. Varmistaakseen käyttöliittymän optimaalisen tehokkuuden, käytettävyyden testaus on välttämätön osa suunnitteluprosessia. Tämä testaus luo kokonaiskuvaa sille, mitkä toiminnot toimivat niin kuin oli suunniteltu ja mitkä eivät. Vasta käytettävyyskorjausten jälkeen voidaan käyttöliittymää kutsua käyttäjille optimoiduksi.

2.1.2 HCI yleisesti

Ihmisen ja koneen vuorovaikutuksessa (Human-Computer Interaction) keskeisenä määrittelyssä on Normanin toiminnan teoria. Tämän mukaan ihmiset omaavat ideoinnin taidon ja tietokone on vain kone. Ihmiset käyttävät tietokoneita työkaluina. He tekevät aloitteen, joka sisältää päämäärän. Tämän jälkeen tehdään toiminto, johon järjestelmä vastaa tietyllä tavalla. Vastausta arvioidaan siten, että liittyikö se siihen, mitä käyttäjä halusi (Redmond-Pyle D, Moore A, 1995, 29-43). Käyttöliittymä on tässä keskeinen käsite. Se on laitteen, ohjelmiston tai minkä tahansa muun tuotteen osa, jonka kautta käyttäjä käyttää tuotetta. Graafinen käyttöliittymä, jota me käsittelemme tässä

tutkielmassa, tarkoittaa tekstiin, kuviin ja käyttöliittymäelementteihin perustuvaa metaforista ihmisen ja tietokoneen välistä vuorovaikutusta. Graafinen käyttöliittymä koostuu graafisista käyttöliittymäelementeistä. Kuvassa 2 mallinnetaan yksinkertaista vuorovaikutus tilannetta.



Kuva 2: HCI

Toiminnan teoriaan liittyy myös ihmisen luoma henkinen malli. Mitä parempi henkinen malli on kyetty rakentamaan, sitä tehokkaammin kyseinen käyttäjä tulee toimeen järjestelmän kanssa (Redmond-Pyle D, Moore A, 1995, 29-43). Täten ongelmanratkaisuissa on mahdollista soveltaa tietoa, ja löytää ratkaisu. Henkinen malli on pidettävä mielessä graafisen käyttöliittymän suunnittelussa. Mitä helpommin käyttäjä kykenee luomaan kyseisen mallin mieleensä, sitä suurempi on hänen tehokkuutensa järjestelmän parissa. Graafisen käyttöliittymän pitäisi siis olla selvä, yhdenmukainen, yksinkertainen ja sen objekteilla (ja ikoneilla) on oltava yhteys reaalimaailmaan. Varma käytettävyysongelma saadaan aikaiseksi, jos objekti näyttää samalta, mutta toimii eritavalla kuin reaalimaailman vastaava objekti. Käyttöliittymän suunnittelija auttaa käyttäjää luomaan henkisen mallin järjestelmästä. Käyttäjä luo myös muiden järjestelmien kokemusten pohjalta omaa malliaan. Tästä voidaan päätellä, että henkinen malli järjestelmistä on monen tekijän vaikutuksen kohteena (kts. kuva 3).

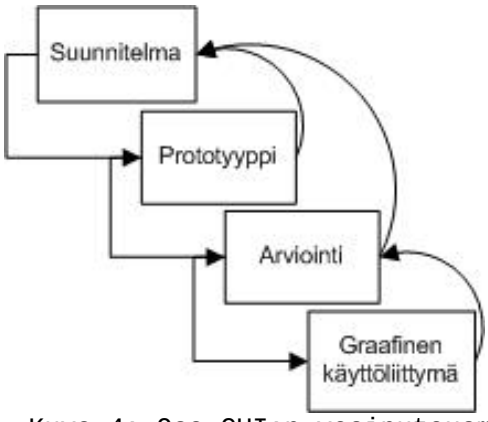
Kuva 3: Henkiseen malliin vaikuttavat tekijät

Tämän pohjalta voisi olettaa, että on helppoa käyttää henkistä mallia apuna graafisen käyttöliittymän suunnittelussa. Mallin käytössä kuitenkin esiintyy ongelmia. Miksi emme siis voi suoraan toimia mallien kanssa? Mallit ovat aineettomia, käyttäjän pään sisällä olevia käyttäjäkeskeisiä rakenteita. Täten harvoin myös kyetään kertomaan tai selvittämään mallin rakenne ja sisältö täydellisesti. Psykologisesti arvioituna malli on osittain alitajunnainen, eikä siis helposti piirrettävissä paperille.

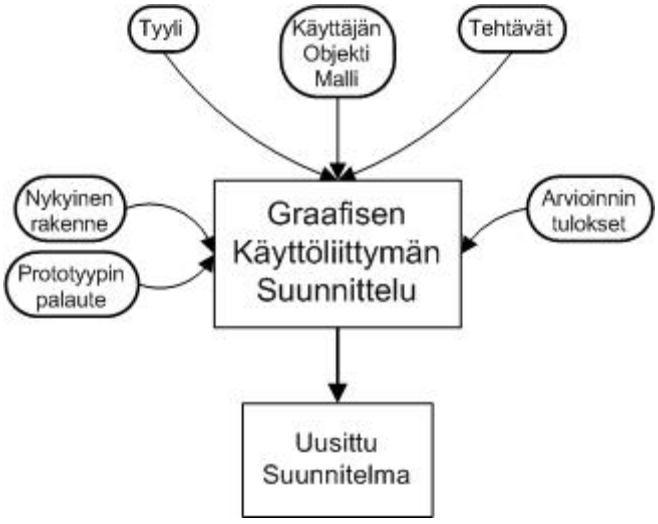
2.1.3 GUI suunnittelu yleisesti

Yllä on mainittu jo tekijöitä, joita on otettava huomioon graafisen käyttöliittymän suunnittelussa. Itse toteutusta ajatellen, on otettava huomioon vielä kognitiivisia tekijöitä. Käyttäjää ei saa laittaa ”muistamaan” asioita, on siis käytettävä tunnistusta, mieleen palauttamisen sijasta. Prosessien päättyessä on ilmoitettava toiminnan loppumisesta. Nämä seikat liittyvät ihmisten muistinkäyttöön. Toisena on otettava huomioon fakta, että ihmiset tekevät virheitä. Käyttöliittymän on siis kyettävä ennakoimaan virheidenteko, esimerkiksi estämällä mahdollisten tilanteiden synty. Tilanteen kuitenkin syntyessä, on annettava hyödyllinen virheilmoitus. Koska ihmisten vahvuutena on erilaisten tuntemattomien objektien tutkiminen ja sopeutuminen sen käyttöön, on käyttöliittymän tuettava tätä tutkimisen halua. Integroituna on siis oltava prosessi, joka pystyy purkamaan viimeiseksi tehdyn toiminnon ((Re)Do – Undo).

Graafisen käyttöliittymän suunnittelu on kehittynyt kolmesta tekijästä, käyttäjän objektimallista, tehtävienmallista ja sovelluksen tyylistä (Redmond-Pyle D, Moore A, 1995, 180-215). Nämä liittyvät ensimmäiseen suunnitelmaan, prototyyppiin. Prototyypin luomisen jälkeen, suoritetaan arviointi, ja tämän onnistuessa luodaan graafinen käyttöliittymä. Vesiputousmallin mukaisesti voidaan joutua palautumaan takaisin suunnitelmaan tai arviointiin, jos käyttöliittymä on todettu vajaaksi kuvan 4 mukaisesti. Kun tämä ensimmäinen vedos on tuotu julki, alkaa käyttöliittymän kehitys. Tässä vaiheessa on jo enemmän kuin kolme tekijää luomassa kokonaisuutta, jota kutsumme graafiseksi käyttöliittymäksi. Tässä kehityksessä on otettava huomioon jo valmis käyttöliittymä, prototyypin palaute, ja arvioinnin tulokset. Graafisen käyttöliittymän suunnittelua käytetään siis käyttöliittymän prototyypissä, käyttöliittymän arvioinnissa ja itse koko järjestelmän toteutuksessa ja testauksessa. Näitä tekijöitä on mallinnettu kuvassa 5.

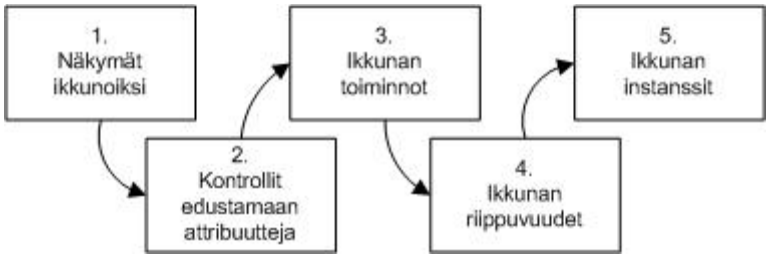


Kuva 4: Osa GUI:n vesiputousmallia



Kuva 5: GUI:n kehittämisen tekijät

Tarkastellaan viimeiseksi itse toteuttamisen prosesseja, jotka voisivat olla kehityksen alaisena. Prosessiluokkia on kolme. Ensimmäinen määrittelee käyttäjän objektien näkymät. Toinen kuvastaa tietoa ja toimintoja, jotka käyttäjä tarvitsee tietyissä tehtävissä. Viimeinen prosessi on olennainen interaktiivisuuden luontiin käyttäjälle. Olemme eniten kiinnostuneita tämän tutkimuksen ohessa prosessiluokasta kaksi (Redmond-Pyle D, Moore A, 1995, 180-215). Tätä luokkaa on mallinnettu kuvassa 6.



Kuva 6: Suunnitelman prosessit toteutukseen

Prosessi etenee seuraavasti: luodaan pääikkuna, ja mahdolliset lisänäkymät tarvittaessa (liiallisen tiedon kertyminen yhdelle näkymälle on vältettävä). Muutetaan attribuutit käyttäjän käyttämiksi kontrolleiksi kysymyksellä: miten attribuutti tulisi esittää käyttäjälle kontrollina ja myös silloin, kun sitä ei ole mahdollista käyttää? Tarkistetaan, että käyttäjällä on työkalut tehdä sitä, mitä heidän tarvitsee ohjelmalla milläkin hetkellä suorittaa. Jotta käyttäytyminen olisi hyvin määritelty, turhat ikkunoiden suhteet toisiinsa tulee poistaa. Esimerkkinä, jos sulkee ikkunan A, niin mitä tapahtuu jo auki olevalle ikkunalle B? Tämä prosessin vaihe nostaa käyttöliittymän käytettävyyttä. Viimeisenä

tässä prosessissa tulee määritellä voiko käyttäjällä olla enemmän kuin yksi instanssi auki. Määrittely on tärkeätä, koska se vaikuttaa kehittämiseen. Vaikutus näkyy käyttöliittymän kompleksisuuden vähenemisenä.

2.1.4 Luova suunnittelu

Puhuttaessa Creative Designista on syytä myös mainita Engineering Design ja kertoa hieman näkökulmien eroista. Engineering Design olettaa ongelman olevan selkeästi määritelty ja rajattu ja pyrkii löytämään mahdollisimman taloudellisen ja tehokkaan ratkaisun, jonka mukaan valmis järjestelmä tuotetaan. Creative Design taas yrittää ongelman ratkaisun lisäksi myös ymmärtää ongelmaa ja etsii useampia vaihtoehtoisia ratkaisuja Engineering Designin yhtä ratkaisua vastaan. Creative Designin suunnittelutyössä suunnittelija on siis keskeisemmässä roolissa kuin Engineering Designissa. (Löwgren J, 1995, 87-94)

Tietojärjestelmän elinkaari Engineering Designin näkökulmasta suunnitellulla järjestelmällä on suppea mallintaa. Järjestelmä suunnitellaan vastaamaan tiettyyn ongelmaan, joten uusien ongelmien ilmetessä järjestelmän muuttaminen on jähmeää. Lisäksi täysin uuden kehitysprojekti luonti on hidasta ja kallista. Creative Design taas on mukautuvampi ja kehitysvaiheessa vapaampi lähestymistapa tarjoaa monipuolisemman lopputuloksen. Taulukossa 1 on esiteltynä näiden eroavaisuudet.

	Engineering Design	Creative Design
Prosessi	Suppeneva kokonaisuus.	Hajaantuvia näkökulmia
Avainkysymys	Miten: ratkaista ongelma?	Miksi: tämä on ratkaistava ongelma?
Toteutuksen vaiheet	Peräkkäinen: yksi ratkaisuvaihtoehto toteutetaan	Rinnakkainen: monta vaihtoehtoa pohditaan.
Prosessin luonne	Analyyttinen: voidaan jäsentää	Luova: luonnostaan ennustamaton
Tarkoitus	Yksi tyydyttävä ratkaisu.	Tutkitaan vaihtoehtoja ennen toteutusta
Omistus	Persoonation: suunnittelija on tavoitteen omaava työkalu	Persoonallinen: suunnittelija on osana suunnitelmaa, kantaa sosiaalisen vastuun

Taulukko 1: Engineering Design ja Creative Desingin eroavaisuudet

2.2 Suunnittelu ihmisille

Itselleni tulee graafisesta käyttöliittymästä mieleen ikkuna, jossa on nappuloita, valintalaatikoita, värejä, kuvia ja muita objekteja, joita voi painaa hiirellä. Graafisen käyttöliittymän kautta voin siis ohjata ohjelmaa hiirenpainalluksilla. Tämän hallinnan tuntemuksen aikaansaaminen on yleisesti kuva onnistuneesta suunnittelusta. Vastakohtaisesti tunteen puuttuminen heikentää ihmisen ja koneen välistä kanssakäymistä, ja kun kommunikaatio ontuu, laskee koko prosessin tehokkuus.

Tietyn sovelluksen satunnaiset käyttäjät, kuten organisaation johtoporras, käyttävät ohjelmaa harvoin. Täten heillä ei usein ole aikaa perehtyä ohjelman käyttöön. Käyttöliittymän on siis otettava tämänkaltaisen kohderyhmän huomioon. Siirryttäessä askeleen alaspäin PIOCO:lla, P tasolta eli asiantuntemustasolta IO eli käyttäjätasolle, päästään käyttäjien pariin, jotka ovat oleellisin ryhmä tässä tutkimuksessa. Heidän osuutensa käyttöliittymän suunnittelussa on oltava suurempi, kuin vain monivalintakyselyyn vastaaminen. Jos käyttäjän osallistuvat aktiivisesti käyttöliittymän suunnitteluun, he voisivat yleisesti olla tyytyväisempiä lopputulokseen. Tällöin ekspertit eivät ole tehneet pimeissä huoneissaan koko järjestön toimintaa muuttavaa tietojärjestelmää, ja usuttaneet sen käyttäjiensä päälle, vaan tämä järjestelmä olisi myös käyttäjien ehdotuksia sisältävä. Käyttäjien (sovellukseen liittyvä käytännön tieto ja taito) ja järjestelmäsuunnittelijoiden (sovellukseen liittyvä tekninen tieto ja taito) on toimittava tiiminä, jolloin käyttäjät ovat tietoisia järjestelmän toiminnasta sen implementoinnin tapahtuessa. Epäonnistuminen toimimaan yhdessä IO tasolla aiheuttaa kokonaisvaltaisen ja välittömän toiminnan tehokkuuden laskun. Jos käyttäjille tuodaan täysin erilainen järjestelmä verrattuna edeltäjään, on takuuvarmaa, että jokainen käyttäjä on sen kanssa hukassa enemmän tai vähemmän. Tässä raportissa emme laajasti huomioi PIOCO-mallin CO tasoa.

Yllämainittujen perusteluiden pohjalta on helppo todeta käyttöliittymän kiistaton merkitys metodologioissa ja tietojärjestelmän kehittämisessä. Siihen on välttämätöntä kiinnittää huomiota heti alusta alkaen, jotta projektin kokonaisvaltaisuus säilyisi ehjänä.

3 METODOLOGIAN KÄSITTELY

3.1 Mikä on Multiview?

Hyödyllisempänä tutkimuksellemme on kurssin aineistojen ulkopuolella oleva metodologia. Tämä metodologia on Multiview. Metodologia käsittää tietojärjestelmän kehittämisen käyttäjien ja kehittäjien yhteistyönä. Metodologia siis sisältää käytännön tiedon ja teknisen tiedon järjestelmien kehittämisessä. Multiview ei ole määräävä metodologia, vaan ehdollinen. Tilanteet ja ympäristöt jossa projekteja toteutetaan, vaihtelevat järjestelmistä riippuen. Metodologia on tutkimusmatka tietojärjestelmän kehittämiseen. Lähtökohtana on, että Multiview on joustava metodologia, tietty tekniikka tai näkökulma sopii vain tiettyyn tilanteeseen. (Avison D, Fitzgerald G, 1995, 375-386)

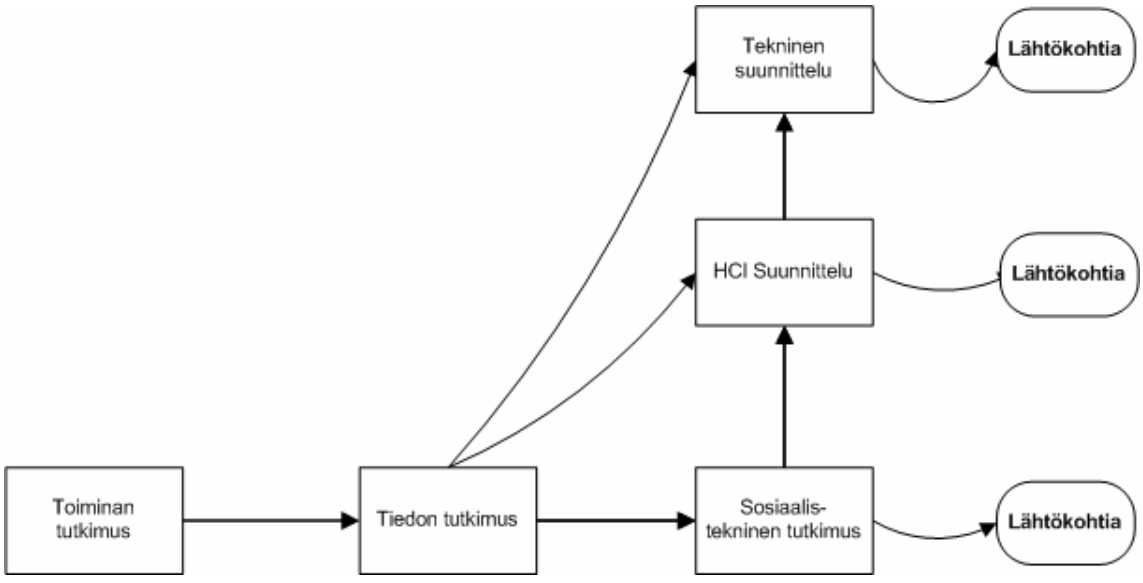
Metodologian vaiheet ovat seuraavat (kts. kuva 7):

1. Ihmisen toiminnan tutkimus
2. Informaation tutkimus
3. Sosiaalis-tekninen tutkimus ja suunnittelu
4. Ihmisen ja tietokoneen kanssakäymisen suunnittelu
5. Teknisten vaiheiden suunnittelu

Nämä vaiheet ovat erilaisia näkökulmia, joiden pohjalta on tarkoitus käydä läpi kaikki tarvittavat käyttäjien toimintaan liittyvät kysymykset. Kokonaisvaltaisuus tietojärjestelmän kehittämisestä tulee tällöin paremmin ilmi. Vaiheet etenevät yleisestä suunnittelusta yksityiskohtaisempaan suunnitteluun. Vaiheiden lopputulokset ovat joko seuraavan vaiheen lähtökohta tai ovat koko metodologian lopputuloksia (kts. taulukko 2). Vaiheiden mahdolliset tulosteet ovat esiteltyinä alla olevassa taulukossa. Rajaamme tutkimuksemme kohdistumaan vaiheeseen 4, joka käsittelee HCI:tä.

Lähtökohdat	Kysymykset
Sosiaaliset näkökulmat	Miten kehitys vaikuttaa minuun?
Roolien asettelu	Muuttuuko työni? Miten?
Ihmisten tehtävät	Mitä minun pitää tehdä?
HCI käyttöliittymä	Miten toimin tietokoneen kanssa?
	Mitä syötteitä/tulosteita?
Tietokanta	Mitä tietoa on mukana?
Tietokannan ylläpito	Tiedon rakenteellinen hallinta?
Palautus	Mitä tapahtuu jos käy huonosti?
Seuranta	Toimiiko järjestelmä vaaditusti?
Hallinta	Turvallisuus ja yksityisyys?
Tiedon haku	Mitä tietoa saan?
Sovellus	Mitä järjestelmä tekee?
Syötteet ja tulosteet	Vaikuttaako muuhunkin?

Taulukko 2: Metodologiaan kuuluvat tarkastelut



Kuva 7: Metodologian vaiheet

3.2 Luova suunnittelu graafisessa käyttöliittymässä

Esittelemme kokoamamme aineiston pohjalta havainnollistavan esimerkin, miten näitä kaikkia on mahdollista ja tarpeellista käyttää yhdessä tietojärjestelmän kehityksessä.

Valitsimme havainnollistavaksi esimerkiksi kurssilla esitellyn naapuriston kotitaloudenhoidon kehittämisen. Ympäristössämme on 5 kotitaloutta, joissa asuu sinkkuja, lapsiperheitä ja pariskuntia. Kotitaloudet ovat pyytäneet luomaan järjestelmän, jolla on mahdollista hallita tuloja ja menoja. Sovellus toimii kotitietokoneympäristössä.

Käytämme lähestymistapana Creative Desingia, joka suosii variaatiota. Projektin kohderyhmän ollessa kuitenkin pienehkö, on joissain osissa suositeltavampaa käyttää hyödyksi myös Engineering Designiä. Multiview metodologia liittyy hyvin vahvasti tutkimuksessa esiteltyihin HCI:hin ja käytettävyyteen. Hyödynnämme näitä yhdessä ylläolevaan esimerkkiin.

Koska asukkaiden taustat ovat erilaiset, voidaan tehdä oletus, että laitteisto ja tietotaito ovat eritaisoiset. Ensimmäinen siirto olisi esitutkinta, millä kartoitetaan järjestelmän minimivaatimukset sekä saada yleiskäsitys asiakkaiden taidoista, toiveista ja vaatimuksista järjestelmän suhteen. GUI:n vesiputousmallin mukaisesti siirrytään tekemään järjestelmäanalyysi esikartoituksen jälkeen. Tässä vaiheessa kehittäjät selvittävät itselleen esitutkinnan tiedoista loogisen tason kuvauksen järjestelmän toiminnasta.

Kehitysryhmän seuraava siirto on suunnitelman luominen. Suunnitteluvaiheessa tarkoituksena on muuntaa analyysivaiheessa tehty toiminnallinen määrittely teknilliseksi määrittelyksi, joka sisältää kuvauksen järjestelmän toteutuksesta. Suunnittelussa otamme huomioon vain raportin rajauksen käsittelyn, eli emme keskity teknisen puolen toteutukseen, vaan keskitymme seuraavaksi itse GUI:n suunnitteluun.

Suunnitteluvaiheessa otetaan huomioon jo aiemmin käsitelty GUI:n suunnittelu (2.1.3) ja HCI:n perusteet (2.1.2). Käyttöliittymän on siis otettava huomioon ihmisen psykologisia ja kognitiivisia ominaisuuksia. Esimerkkeinä: käyttäjien muistinkäyttö on minimoitava havainnollistuksen avulla. On otettava huomioon se, että käyttäjän on kyettävä muodostamaan avullamme henkinen malli järjestelmästä (dokumentoinnin ja tiedotuksemme tärkeys). Faktaa on kuitenkin se, että ihmiset tekevät virheitä. He myös lähes varmasti tutkivat järjestelmää, mitä he käyttävät. Tästä johtuen järjestelmän palautusominaisuus on välttämätön ((re)do/undo). Kun virhetilanne kuitenkin ilmentyy, on sen annettava havainnoiva virheilmoitus. Graafisen käyttöliittymän pitäisi siis olla selvä, yhdenmukainen, yksinkertainen ja sen objekteilla (ja ikoneilla) on oltava yhteys reaalimaailmaan.

Suunnittelusta seuraava vaihe on ensimmäisen prototyypin hahmottaminen GUI:n suunnittelun kuvan 4 mukaan. Kehittäjäryhmä on tässä vaiheessa saanut kokonaiskuvan kohderyhmästä ja luonut sen pohjalta suunnitelman, miten toteuttaa järjestelmä. Prototyyppi on hyvin pelkistetty versio, ja sisältää vain perustoiminnot. Tätä ensimmäistä prototyyppiä annetaan asiakkaille testikäyttöön, ja suoritetaan käytettävyystestaus. Käytettävyystestauksen lopputuloksista ja asiakkaiden palautteiden perusteella palataan joko suunnittelu vaiheeseen, tai siirrytään lopullisen version muodostamiseen. Prototyyppi-suunnittelu vaihetta toistetaan kunnes se vastaa riittävän hyvin asetettuja tavoitteita.

Perusoletuksena voidaan pitää, että ensimmäinen prototyyppi palaa ainakin kerran takaisin suunnitteluvaiheeseen. Tässä vaiheessa otetaan huomioon GUI:n suunnittelun kuvan 5 tapahtumaketju. Seuraavan prototyypin suunnitteluvaiheessa on siis otettava huomioon prototyypin palaute, nykyinen järjestelmän rakenne ja käytettävyysarvioinnin tulokset. Näiden käsittelyjen jälkeen siirrytään prototyypin toisen version luomiseen ja palataan GUI:n vesiputousmallin osan prototyyppivaiheeseen.

Prototyyppivaiheen valmistuttua, siirrytään elinkaarimallin mukaan käyttöönottovaiheeseen. Graafiselle käyttöliittymälle suoritetaan tässä vaiheessa vielä viimeinen käytettävyystestaus. Palautteesta riippuen jatketaan joko mallin viimeiseen

vaiheeseen eli ylläpitoon tai palataan prototyyppivaiheen suunnitteluun. Ylläpidossa korjataan mahdolliset vähäiset käytettävyystestissä havaitut heikkoudet ja tekniset virheet. Lisäksi ylläpidetään järjestelmää päivityksillä ja varmistetaan sen sujuva toiminta.

4 YHTEENVETO

Graafisen käyttöliittymän suunnittelussa on otettava huomioon lukuisia asioita ennen käyttöönottovaihetta: käyttäjien psykologiset ja kognitiiviset (human-computer interaction, henkinen malli) ominaisuudet, järjestelmän yhtenäisyys ja toiminnallisuus (käytettävyys), suunnittelusuunta (engineering design / creative design) ja koko operaatiota ohjaava metodologia (multiview).

Suunnittelussa on syytä ottaa huomioon seuraavien avainkohtien sisällyttäminen tietojärjestelmään:

- Yhtenäisyyden pyrkimys.
- Kokeneille käyttäjille etenemismahdollisuuksia.
- Aina prosessin päättyessä ilmoitus käyttäjälle.
- Virheiden käsittely ja selkeä viestintä.
- Mahdollisuus kumota tehty toiminto.
- Käyttäjien muistamistarpeen vähentäminen.
- Koordinointi käyttäjäkannan kanssa.
- Henkisen mallin luonti.

Ensimmäiset kokemuksemme käyttöliittymiin ovat DOS - pohjaiset järjestelmät. Muistamme hyvin ensimmäisten graafisten käyttöliittymien ilmestymisen markkinoille. Microsoft Windows 3.X tuoteperheen vaihtoehtoinen tapa hallita koko järjestelmää oli mullistava siirtyminen DOS:in käskyjen ulkoa muistamisesta, järjestelmän navigointiin hiirellä. Koska oma sukupolvemme muistaa tämän siirtymävaiheen, on siis ala hyvin nuori ja kehitys ollut huimaa. Graafisten käyttöjärjestelmien kehittämisellä on mahdollisuus siirtyä vain eteenpäin.

LÄHTEET

Redmond-Pyle D. & Moore A. (1995): “Graphical user interface design and evaluation (guide): a practical process”, London: Prentice Hall. Luku 3 (29-43) ja Luku 10 (180-215)

Iivari, J. and Koskela E. (1987): “The PIOCO Model for Information Systems Design”

Jonas Löwgren (1995): “Applying Design Methodology to Software Development”, Department of Computer and Information Science, Ljnköping University, Sweden.

Avison, D.E. and Fitzgerald, G. (2003) Information Systems Development: Methodologies, Techniques and Tools, Third Ed., McGraw-Hill. Luku 7 (527-574)

Avison, D.E. and Fitzgerald, G. (1995) Information Systems Development: Methodologies, Techniques and Tools, Second Ed., Luku 6 (76-80, 375-386)

Donald Norman (1990) The design of everyday things