Teksti: Karri-Pekka Laakso

Kokemuksia GUIDe-käyttöliittymäsuunnittelun ja Scrum-menetelmän yhdistämisestä

Karri-Pekka Laakso suunnittelee ja toteuttaa käyttöliittymiä Reaktor Innovationsissa.

Reaktor Innovationsissa on jo vuosien ajan käytetty ketterää Scrum-menetelmää käytännössä kaikissa ohjelmistoprojekteissa. Vuodesta 2005 lähtien ohjelmistojen käyttöliittymäratkaisuja on alettu parantaa laatimalla ne GUIDe-käyttöliittymäsuunnittelumenetelmää noudattaen. Näiden menetelmien yhdistäminen onnistuneesti ei ole ollut aivan itsestään selvää ja vaivatonta, ja siksi tässä artikkelissa jaetaan Reaktorissa kerättyjä kokemuksia.

Miksi ketterät menetelmät tarvitsevat systemaattista käyttöliittymäsuunnittelua?

Käyttöliittymäsuunnittelu on monitulkintainen termi, jolla joskus tarkoitetaan esimerkiksi pelkkää visuaalista suunnittelua tai vaatimusmäärittelyvaiheessa päätettyjen toimintojen asettelemista näytölle. GUIDe-mallin [GUIDe04] mukainen käyttöliittymäsuunnittelu on kuitenkin pikemminkin vaatimusmäärittelyn työvaihe, joka tuottaa vastaukset seuraaviin kysymyksiin:

mitä tietoja järjestelmä käyttäjälleen näyttää (tietosisältö käyttäjän kannalta),

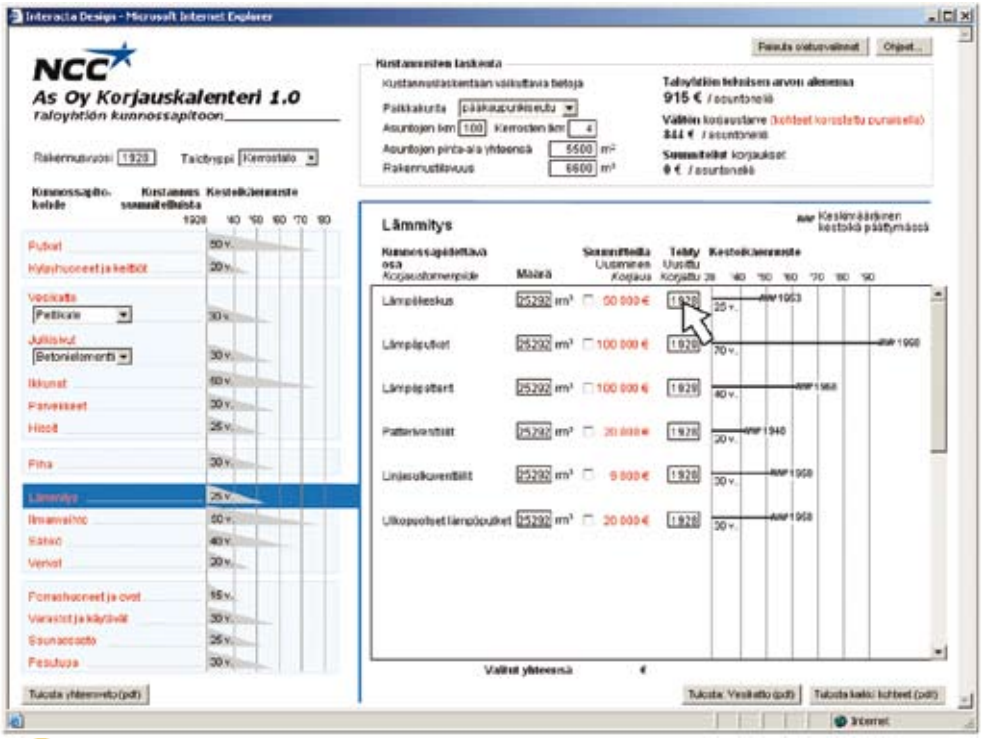
miten se **näyttää** nämä tiedot (tietojen organisointi ja visualisointi näytöllä),

mitä käyttäjä voi tehdä järjestelmällä (toiminnallisuus) ja

miten järjestelmä reagoi, kun käyttäjä tekee nämä asiat (toimintalogiikka).

Tällaisessa käyttöliittymäsuunnittelussa on siis kyse yksityiskohtaisesta tietosisällön ja toiminnallisuuden määrittelystä, joka tehdään käyttötilanteiden suorittamista simuloimalla. GUIDe-mallissa se kulminoituu yksityiskohtaiseen käyttöliittymäspeksiin, joka koostuu lopullisen ohjelmiston kaltaisista näyttökuvista (vrt. kuva 1, ensimmäinen näyttökuva). GUIDen käyttöliittymäsuunnittelua ja sen seurauksia on kuvattu Systeemityö-lehden numerossa 2/2006 [Sytyke06].

Kuva 1. NCC:n Korjauskalenterin alkuperäinen käyttöliittymäkuva (Interacta).



Ketterissä menetelmissä vastaava työ hoidetaan yleensä kirjoittamalla käyttäjätarinoita, mutta niihin sisältyy käsitteenä paljon huonosti määriteltyjä kohtia: Kuka tarinat tekee, asiakas vai toimittaja? Kuka on vastuussa tarinoista? Kuvaako käyttäjätarina ongelman (mitä järjestelmällä pitäisi saada aikaan) vai ratkaisun (mitä toiminnallisuutta järjestelmään tulee)? Miten käyttäjätarinoista saadaan aikaan käyttöliittymäratkaisu (esimerkiksi tietojen organisointi ja toimintalogiikka) ja kuka sen laatii?

Käytännössä ketterää menetelmää käyttävä toimittaja työntää yleensä perimmäisen vastuun määrittelystä sekä sen kattavuudesta ja oikeellisuudesta asiakkaalle. Asiakas ohjataan kirjoittamaan käyttäjätarinoina ratkaisuista, jolloin itse asiassa asiakas yrittää suunnitella järjestelmän toiminnallisuuden itse. Tästä seuraa usein kömpelö järjestelmä, koska asiakas ei tyypillisesti osaa käyttöliittymäsuunnittelua.

GUIDen mukaisessa toimintatavassa asiakkaiden toiminnallisuustoiveita (ratkaisuja) ei oteta toteutettavaksi sellaisinaan, vaan ensin käyttöliittymäsuunnittelija selvittää, mikä on asiakkaan tarvetta vastaava käyttötilanne (ongelma), joka pitäisi ratkaista. Esimerkiksi ammattikorkeakoulun opetuksen suunnittelujärjestelmän asiakas taannoin pyysi, että järjestelmään tehtäisiin itsepalveluraportointi, jossa käyttäjä voisi hakea opetustapahtumia millä tahansa kentillä. Osoittautui, että konkreettisia hakutilanteita, joita tämä haku palvelisi, oli karkeasti ottaen kaksi: 1) opettaja kirjoittaa omista kursseistaan sisältökuvauksen ja aikataulun opiskelijoille, ja 2) hallinnon työntekijä tekee tilastokeskukselle raportin edellisen vuoden opetuksesta. Kumpaankaan näistä ongelmista itsepalveluraportointi ei ollut kovin hyvä ratkaisu: opettajalle kannattaa mielummin näyttää lista hänen omista kursseistaan, ja hallinnon työntekijälle luoda suoraan tilastokeskuksen tarvitsema raportointimuoto.

GUIDe-mallin mukainen käyttöliittymäsuunnittelu täydentää hyvin ketterien menetelmien määrittelyvaiheen aukkoja, ks. laatikko.

GUIDe-mallin mukainen käyttöliittymäsuunnittelu täydentää hyvin ketterien menetelmien määrittelyvaiheen aukkoja:

- Käyttötilanteet kuvaavat käyttäjän ongelman eivätkä lainkaan ratkaisua (niissä ei esiinny toteutettavaa järjestelmää lainkaan).
- Käyttöliittymäratkaisu kuvaa yksikäsitteisen ja suoraviivaisen ratkaisun em. ongelmiin.
- Ratkaisu esitetään kuvina ja kuvasarjoina, joita lukijan on helppo ymmärtää.
- Käyttöliittymäratkaisun laatii käyttöliittymäsuunnittelija (vastuu ratkaisusta on siis toteuttajalla, ei asiakkaalla).
- Käyttötilanteet selvitetään ensisijaisesti järjestelmän tulevilta käyttäjiltä, ja asiakkaan rooli on ohjata suunnittelijat oikeiden käyttäjien luo tutkimaan heidän työtään.
- Ratkaisun oikeellisuutta ja kattavuutta voidaan testata jo ennen toteutusta.

Käyttöliittymästä toteutukseksi

Reaktorin projekteissa käyttöliittymäsuunnittelu sijoitetaan aivan projektin alkuun, mieluiten siten, että toteutustyö aloitetaan vasta käyttöliittymän valmistumisen jälkeen. Jos projektissa ollaan tekemässä kokonaan uutta järjestelmää, alku etenee GUIDea noudattaen: ensin tehdään käyttötilanneselvityksiä käyttäjien luona, ja sitten suunnitellaan niiden pohjalta ratkaisut, jotka testataan käyttäjien kanssa.

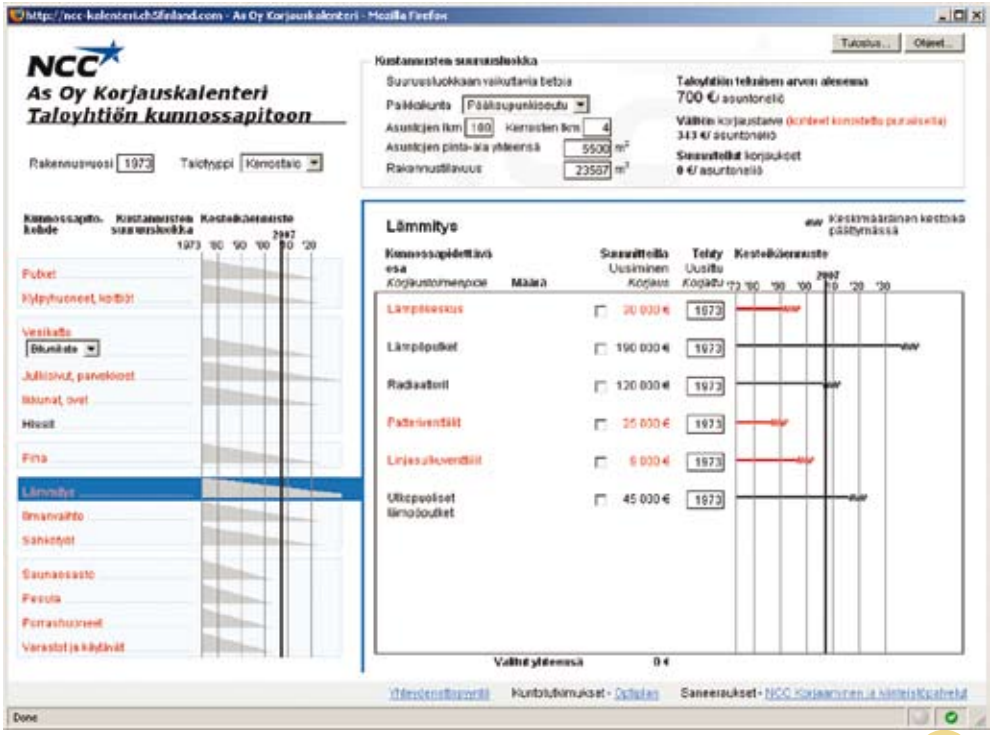
Käyttöliittymäratkaisut kommunikoidaan toteutustiimille käyttöliittymäkuvina tai -kuvasarjoina. Tämän lisäksi käyttöliittymäsuunnittelija demoaa ratkaisun toiminnan toteuttajille realistisilla käyttötilanteilla, jotta toteuttajat varmasti ymmärtäisivät ratkaisun toimintalogiikan oikein. Reaktorissa on pyritty kirjoittamaan mahdollisimman vähän dokumentaatiota ja panostamaan sen sijaan kommunikointiin: ideaalitilanteessa käyttöliittymäsuunnittelija istuu toteutustiimin kanssa koko projektin ajan saman pöydän ääressä tai samassa huoneessa, jolloin toteuttajat voivat milloin vain kysyä häneltä täsmennyksiä ja tarkennuksia. Käytännössä tähän päästään kuitenkin valitettavan harvoin, koska yhdellä suunnittelijalla on helpostikin käynnissä 2-5 projektia samaan aikaan.

Toteutusta varten käyttöliittymäsuunnittelija ja toteutustiimi purkavat käyttöliittymän toiminnoiksi tai käyttäjätarinoiksi, joista muodostuu projektin toimintolista (product backlog). Backlogin priorisoinnissa kiinnitetään erityisesti huomioita siihen, että käyttäjä voisi mahdollisimman varhaisessa vaiheessa suorittaa järjestelmällä kokonaisia käyttötilanteita loppuun asti.

Kun toteuttaja on koodannut omasta mielestään valmiiksi jonkin toiminnallisuuden, hän antaa sen käyttöliittymäsuunnittelijalle hyväksyttäväksi. Hyväksynnässä suunnittelija tarkistaa, että toteutus todella vastaa interaktiotavoiltaan käyttöliittymäratkaisua. Samalla hän myös testaa ratkaisua yksinkertaisilla poikkeustilanteilla. Asiakkaalle toiminnon kerrotaan olevan valmis vasta sen jälkeen, kun käyttöliittymäsuunnittelija on hyväksynyt toteutuksen.

Koettuja haasteita...

GUIDe-käyttöliittymäsuunnittelun ja Scrumin yhteenliittäminen ei ole sujunut kitkatta. Esimerkiksi projektin aloittaminen käyttöliittymäsuunnittelulla on näennäisesti ristiriidassa Agile



Kuva 2: Kuva lopullisesta toteutuksesta (Reaktor).

manifeston [Agile01] kanssa: toteutustyön pitäisi periaatteessa alkaa heti projektin ensimmäisistä päivistä lähtien. Tämän ajattelutavan vuoksi GUIDe-menetelmää käyttöönotettaessa oli vaikea perustella, miksi koodaamista kannattaisi lykätä myöhemmäksi. Todellisuudessahan järjestelmän käyttöliittymän määritteleminen koodaamalla on erittäin hidasta ja kallista verrattuna kynällä paperille piirrettyihin käyttöliittymäkuviin, ja lisäksi se tuottaa helposti huonoja ratkaisuja. Manifesto onkin tähdätty hyödytöntä dokumentaatiota vastaan. GUIDen käyttöliittymäsuunnittelu ei ole manifeston tarkoittamaa ylimääräistä dokumentointia, vaan siinä järjestelmän käyttöliittymää kehitetään hyvin konkreettisesti piirtämällä kuvia ketteriä menetelmiä vastaavissa nopeissa suunnittelu-testaus-sykleissä – ainoastaan kymmeniä kertoja koodausta nopeammin.

Erityisesti jos projektin kustannukset halutaan pitää mahdollisimman pieninä eikä projektilla ole kova kiire, koodaaminen kannattaa aloittaa vasta käyttöliittymäsuunnittelun jälkeen. Näin ei tehty vielä ensimmäisissä projekteissa. Esimerkiksi NCC:lle tehtyä Korjauskalenteri-sovellusta [NCC06] kehitettäessä Reaktor aloitti toteutustyöt samaan aikaan, kun Interacta Design Oy aloitti sovelluksen käyttöliittymäsuunnittelun. Käyttöliittymäratkaisun valmistuttua osoittautui, että kaikkea siihen mennessä tehtyä toteutustyötä ei olisikaan välttämättä tarvittu, vaan taustajärjestelmäksi olisi riittänyt yksinkertaisempikin toteutus. Tämä ei suinkaan ollut selvää projektin alkaessa, vaan se selvisi vasta käyttöliittymäratkaisun avulla. Toteutuksen työmäärässä voitaneen säästää kymmeniä prosentteja, jos käyttöliittymäratkaisu on etukäteen selvillä.

Yksinkertaisimmillaan Reaktorin ketterässä projektissa lähes kaikki työ on koodausta ja kaikki projektiin osallistuvat henkilöt osallistuvat vain tähän yhteen projektiin. Näin ei enää olekaan, kun projektiin sisältyy käyttöliittymäsuunnittelu. Silloin projektit pitäisi ensinnäkin miehittää niin, että toteuttajat tekisivät muita projekteja käyttöliittymäsuunnittelun ajan. Toiseksi projektipäällikön pitää huolehtia hyvissä ajoin siitä, että tiimillä riittää koko ajan mielekästä koodattavaa: toimintoja ei enää saadakaan toteutettavaksi ilman käyttöliittymäsuunnitteluvaihetta, vaan käyttöliittymäsuunnittelu on toteutuksen kriittisellä polulla. Jos käyttöliittymäsuunnittelijoiden muiden projektien työkuorma viivästyttää toteutusta, toteuttajat tyypillisesti kannattaa siirtää toisiin projekteihin siksi ajaksi, että käyttöliittymäratkaisu valmistuu.

Lähteet
[Agile01]
Manifesto for Agile Software Development.
http://agilemanifesto.org
[GUIDe04]
Laakso Sari A., Laakso Karri-Pekka, Hyvän käyttöliittymän varmistaminen GUIDe-prosessimallilla.
Helsingin yliopisto, tietojenkäsittelytieteen laitos, julkaisematon artikkeli, 2004.
http://www.cs.helsinki.fi/u/salaakso/papers/GUIDe-suomeksi.pdf
[NCC06]
NCC:n Korjauskalenteri-sovellus.
http://ncc-kalenteri.ch5finland.com
[Sytyke06]
Laakso Sari A., Käyttöliittymälähtöinen vaatimusmäärittely.
Systeemityö, nro 2, 2006, s. 19-21.

... ja koettuja onnistumisia

Hyväksymismenettely, jossa toteutettu toiminnallisuus palaa käyttöliittymäsuunnittelijan hyväksyttäväksi, on osoittautunut erittäin hyödylliseksi ja tärkeäksi vaiheeksi. Toteuttajat eivät ole yleensä varautuneet siihen, millaista laatua käyttöliittymätoteutuksen osalta heiltä odotetaan. Tuloksena on usein toteutus, joka pääsääntöisesti toimii jotenkuten mutta on interaktioiltaan puutteellinen tai virheellinen. On aivan tyypillistä, että uuden tiimin ensimmäisen sprintin kaikki toteutetut toiminnallisuudet palautetaan korjattaviksi. Tarvittaessa jopa kaikki ensimmäisen sprintin tehtävät saatetaan siirtää sellaisenaan toiseen sprinttiin, koska uusia ominaisuuksia ei aleta toteuttaa, ennen kuin vanhat on korjattu ja hyväksytty. Tämän myötä käyttöliittymätoteutuksen laatu yleensä paranee ensimmäisten sprinttien aikana merkittävästi ja päästään tilanteeseen, jossa kaikki järjestelmään tähän mennessä toteutetut piirteet toimivat niin kuin pitääkin ja ne on testattu todellisia käyttötilanteita simuloimalla. Tämä kasvattaa tiimin ammattiylpeyttä: järjestelmää voidaan esimerkiksi demota asiakkaalle vailla huolta ns. demoefekteistä.

Koska toteutuksen työmääräarviot tehdään vasta valmiin käyttöliittymäratkaisun perusteella, ne ovat muuttuneet olennaisesti aiempaa osuvammiksi. Tämä on tuonut lisää realismia ja näkyvyyttä projektin tilaan: sekä asiakkaalla että toimittajalla on parempi käsitys siitä, mitä milloinkin tulee olemaan valmiina ja missä tilassa ollaan suhteessa tavoitetilaan.

Alun kompuroinnin ja roolien hahmottumisen jälkeen projektin tehtäväjako on muuttunut kaikkien osapuolten kannalta luontevammaksi. Toteuttajien ei tarvitse ottaa kantaa käyttöliittymäratkaisuihin, vaan he voivat keskittyä koodaamiseen ja delegoida esiin tulleet käyttöliittymäongelmat eteenpäin. Projektipäällikön ei tarvitse enää neuvotella asiakkaan kanssa järjestelmän sisällöstä, vaan hän voi delegoida sen käyttöliittymäsuunnittelijalle ja keskittyä projektissa ilmenneiden esteiden raivaamiseen ja aikatauluihin.

Lopuksi

Käyttötilanteiden simulointiin perustuvan käyttöliittymäsuunnittelun lisääminen ketterän projektin alkuun saattaa tuoda ketterän herätyksen saaneille toteuttajille uhkakuvia paluusta kohden vesiputousmallia. Monet heistä kuitenkin näkevät viimeistään GUIDen ja Scrumin yhdistelmäprojektiin osallistumalla, miten paljon mielekkäämmäksi projekti muuttuu, kun käyttöliittymäkuvien myötä projektin tavoitetila on selvillä jo heti koodauksen alkaessa. Hyvien käyttöliittymäratkaisujen toteuttamisessa on sekin hieno puoli, että käyttäjiltä voi saada aika erilaista palautetta kuin ennen: ”Tämä on todella mainio softa.”