7

Käytettävyys ja suunnittelumenetelmät

Olemme aikaisemmin todenneet, että keskustelujen sisällön määrittäminen ja keskustelujen toteutuksen suunnittelu on hyvä erottaa toisistaan. Keskustelun sisällön määritys on osa systeemisuunnittelua, kun taas keskustelujen toteutuksen suunnittelu kuuluu projektin myöhempiin vaiheisiin.

Tässä luvussa katsomme, mitä eväitä käytön suunnittelijoilla pitäisi olla siinä vaiheessa, kun keskustelujen toteutusta lähdetään suunnittelemaan. Edellisissä luvuissa on nähty, miten sovellusalueen käsitteiden ja prosessien parempi ymmärrys auttaa suunnittelijaa tekemään käyttäjäystävällisempiä sovelluksia. Tässä luvussa tarkastellaan, että miten nämä asiat vaikuttavat suunnittelun menetelmien soveltamiseen.

Unified Modeling Language eli UML on viime vuosina noussut selkeästi johtavaksi sovelluskehityksen kuvaustekniikoiden työkalupakiksi. Tässä luvussa tarkastelemme, miten UML:n tekniikoita pitäisi soveltaa ja täydentää.

Käsitemalli käyttäjän näkökulmasta

Olemme jo nähneet, että sovelluksen tulisi keskustella käyttäjän kanssa tutuilla ja hänen tarpeisiinsa liittyvillä käsitteillä. Käyttäjän tehtäviin liittyvien käsitteiden hakeminen ei ole lainkaan vierasta systeemityölle. Systeemisuunnittelijat ovat jo ikiajat tehneet käsiteanalyysejä sekä piirrelleet ER-kaavioita (Entity/Relationship). Nykyisin luokkakaaviot tarjoavat samaan tarkoitukseen paljon rikkaamman kuvaamisen välineistön.

Ohjelmistotekniikan menetelmien työkalupakista löytyy siis nykyisin hyviä ja hyödyllisiä tekniikoita käyttäjän käsitteiden mallintamiseen. Niitä on vain osattava käyttää oikealla tavalla. Kirjassaan ”UML Distilled” Martin Fowler erottaa UML:n luokkakaavioille kolme erilaista käyttötarkoitusta:

- **Käsitemalli** kuvaa ne käyttäjän kielen käsitteet, joilla käyttäjä ajattelee ja keskustelee sovellusalueesta.
- **Määrittelymalli** kuvaa systeemin arkkitehtuurin toiminnalliset osat sekä niiden väliset ohjelmointirajapinnat (interfaces).
- **Toteutusmalli** kuvaa ne luokat, joilla määrittelymallin rajapinnat toteutetaan.

Käytettävyyden näkökulmasta meitä kiinnostaa lähinnä sovellusalueen käsitemalli. Fowler korostaa, että se ei mallinna tietokannan sisältöä tai ohjelmakoodin olioita, vaan nimenomaan *käyttäjän sovellusaluetta koskevaa ymmärrystä.*

Vastaavasta mallista puhutaan useissa UML-tekniikoita soveltavissa kirjoissa. Se esiintyy mm. nimillä problem domain model, domain object model tai problem domain object model. Lähes kaikissa kirjoissa tälle mallille on kuitenkin varattu varsin pieni rooli käyttäjien kanssa tapahtuvan Use Case –työskentelyn valinnaisena apuvälineenä.

Olen nähnyt esimerkiksi kirjanpito-ohjelmia, jotka keskustelevat käyttäjien kanssa tietueitten luomisesta tilanteissa, jossa pitäisi keskustella tositteen kirjaamisesta. Toteutusmalliin liittyvät käsitteet eivät saisi ryömiä sovelluksen ja käyttäjän väliseen viestintään! Käyttöliittymän suunnitte-

lun onnistumisen kannalta on tärkeää tehdä käsitemalli huolella – sekä pitää tämä malli puhtaana järjestelmän ratkaisuun liittyvistä käsitteistä!

ER-kaaviot sekä luokkakaaviot kuvaavat sovelluksen käsittelemän tiedon pysyvää rakennetta. Käsitemalli kuvaa ne substantiivit, joilla käyttäjä ja sovellus keskustelevat keskenään. Käyttäjän toimintaan liittyvät käsitteet ovat kuitenkin yhtä tärkeitä ymmärtää. Niiden selvittämiseen täytyy käyttää muita keinoja.

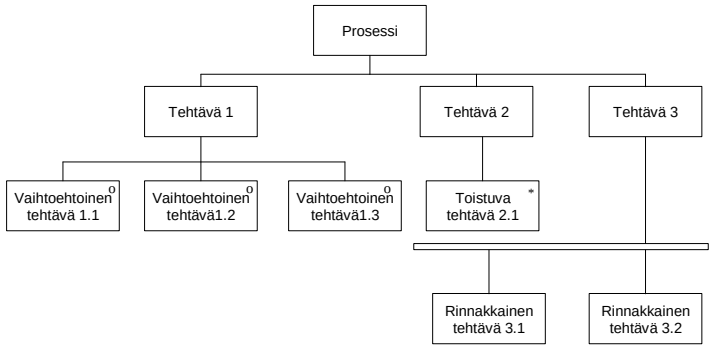
Toiminnan käsitteet elinkaarimallista

UML ei ole tietojärjestelmien suunnittelumenetelmä vaan ohjelmistojen suunnittelumenetelmä. Se tosin sisältää tietojärjestelmien suunnittelussa käyttökelpoisia kuvaustekniikoita. Esimerkiksi luokkakaaviot ovat käyttökelpoisia tietokannan suunnittelussa, ja aktiviteettikaavioita voidaan soveltaa prosessien kuvaamiseen.

UML:n työkalupakista puuttuu eräs erityisen hyödyllinen kuvaustekniikka. Elinkaarimallit ovat erinomainen väline, kun pyritään ymmärtämään sekä käyttäjän että liiketoiminnan prosesseja.

Tunnetuin elinkaarien kuvaustekniikka on jo 80-luvulla kehitetty Jackson Structured Design eli JSD. Tätä ei pidä sekoittaa Jackson Structured Programming –menetelmään, joka on ohjelmoinnin menetelmä. Hämmennystä voi lisätä se, että kummassakin käytetään aivan saman näköistä kaaviotekniikkaa, mutta aivan eri tarkoituksiin.

JSD kuvaa prosesseja neljän perusrakenteen avulla:



- Peräkkäisrakenne (kuvassa Tehtävä 1 - Tehtävä 3)
- Vaihtoehdot (kuvassa Vaihtoehtoinen tehtävä 1.1 - Vaihtoehtoinen tehtävä 1.3)
- Toistorakenne (kuvassa Toistuva tehtävä 2.1)
- Rinnakkasirakenne (kuvassa Rinnakkainen tehtävä 1 - Rinnakkainen tehtävä 2)

Samat rakenteet voidaan esittää myös tekstimuotoisena jäsennyksenä:

Prosessi

Tehtävä 1: Vaihtoehtoisesti

Vaihtoehto 1.1

Vaihtoehto 1.2

Vaihtoehto 1.3

Tehtävä 2: Toistuvasti

Tehtävä 2.1

Tehtävä 3: Rinnakkain

Rinnakkainen tehtävä 3.1

Rinnakkainen tehtävä 3.2

Kaaviot ovat ehkä parhaimmillaan havainnollisempia ja helpompia lukea, mutta tekstimuotoisen kuvauksen muokkaus on paljon helpompaa kuin kaavioiden muokkaus. Kaavion pitäisi mahtua yhdelle paperiarkille, kun taas tekstimuotoista kuvausta voi jatkaa rajatta.

Kirjallisuudessa JSD:tä käytetään lähinnä sovelluksen tietokantaan sisältyvien olioiden elinkaarien kuvaamiseen. Esimerkiksi webbikaupan järjestelmän tietokannassa on tietenkin tiedot tehdyistä tilauksista. Tilauksen JSD-malli kuvaa tilaukselle tapahtuvat asiat syntymästä arkistointiin.

Tällaiset mallit kuvaavat siis asioita tietojärjestelmän näkökulmasta. Prosesseja voidaan kuitenkin kuvata erilaisista näkökulmista. Olettakaamme esimerkiksi, että olemme tekemässä tietojärjestelmää kurssimuotoista koulutusta ja seminaareja tarjoavalle yritykselle. Eri sidostahot näkevät samaan kurssiin liittyvän prosessin hyvin erilaisena.

Esimerkiksi kurssin suunnittelijan näkökulmasta kurssin pitämiseen voisi liittyä seuraavia tapahtumia ja toimintoja:

Alustavan suunnitelman tekeminen
Suunnitelman prosessointi: Rinnakkaisesti
Kommentit markkinoinnista
Kommentit johdolta
Kommentit asiantuntijoilta
Suunnitelman muokkaus
Pitopäätös: Vaihtoehtoisesti
    Ei kurssia
    Kurssin pito
Suunnitelman muutokset
Palaute kurssilaisilta
Uusinnat: Toistuvasti
    Uusintakurssi

Sama kurssi voisi osallistujan näkökulmasta näyttää seuraavalta

Esitteen saanti
Osallistumispäätös: Vaihtoehtoisesti
    Ei osallistumista
    Osallistuminen
        Ilmoittautuminen
        Vahvistus
        Vaihtoehtoisesti
            Peruutus
            Kurssipäivä
            Ilmoittautuminen
            Materiaalin vastaanotto
            Luennot
            Palautteen anto

Vastaavasti esimerkiksi kurssin järjestelyvastaavalla, kurssin isännällä, puhujalla sekä laskutuksen hoitajalla on kaikilla erilainen näkökulma tähän samaan kurssiin.

Kun kurssinpidon prosessi on ensin tällä tavoin jäsennetty eri näkökulmista, niistä on helppoa koota kuvaus siitä, mitä tapahtumia ja toimintoja tietokannan ja ohjelmiston täytyy tukea. Näin syntyvä kuvaus muodostaa sitten hyvän lähtökohdan näiden olioiden ohjelmoinnin suunnittelulle.

Tällainen jäsennys eri näkökulmista auttaa myös systeemin tekijöitä ymmärtämään käyttäjien ja muiden sidostahojen käsitteitä, pyrkimyksiä, prosesseja ja odotuksia.

Käyttötilannemalleilla keskustelun sisältö

Käyttötilannemallit (englanniksi Use Case) ovat UML:n keskeisimpiä työvälineitä. Se kuvaa tavan, jolla käyttäjä saa jonkun tavoitteensa toteutetuksi. Esimerkiksi rahan nostaminen pankkiautomaatista tai majoituksen varaaminen hotellista ovat tyypillisiä käyttötilannemallin käyttäjätavoitteita.

Käyttötilannemalli voidaan myös tehdä jostain käyttäjän osatavoitteesta. Esimerkiksi marginaalin asettaminen voisi olla sopivan käyttömallin aihe, kun suunnitellaan tekstinkäsittelyohjelmaa.

Käyttötilannemallien tarkoitus on kuvata keskustelun sisältöä - ei sen toteutusta. Käyttötilanteen kuvaus kertoo, että

- mitä tietoja keskustelu käsittelee, ja että
- missä järjestyksessä ja
- minkälaisten sääntöjen mukaan keskustelu etenee.

Käyttötilannemalli ei siis kerro mitään valikoista, kentistä, painikkeista tai muista keskustelun toteutuksen välineistä.

Käyttötilannemallien läpikäynnissä käyttäjien edustajien kanssa saatetaan tarvita paperille tai tietokoneen ruudulle toteutettuja protoja sovelluksen ikkunoista ja lomakkeista. Tässä vaiheessa näiden protojen tarkoitus on kuitenkin ainoastaan auttaa keskustelun sisällön määrittelemisessä.

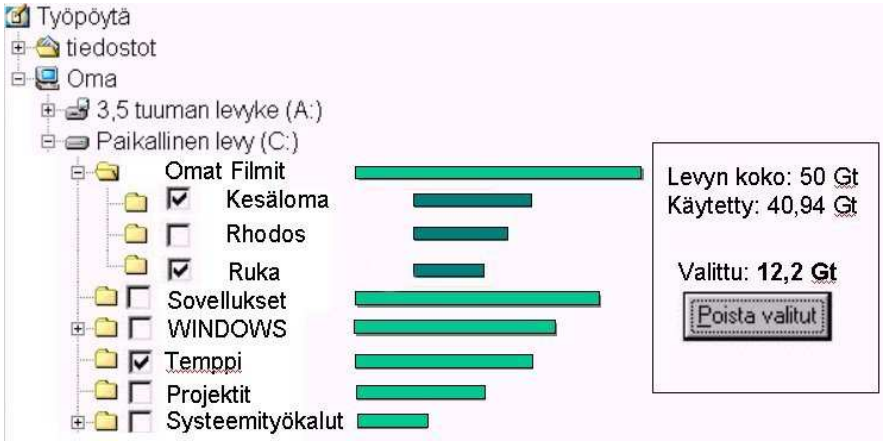
Käyttötilannemallin tarkoitus on kuvata ne erilaiset tavat, joilla käyttäjä pääsee tiettyyn tavoitteeseen. Yleensä kuvataan ensin normaali tapahtumien kulku, ja sitten erilaiset muunnelmat ja poikkeustilanteet.

Käyttötilannemalli tehdään siis yleispäteväksi. Se saattaa muodostua niin mutkikkaaksi ja abstraktiksi, että sitä ei kannata käyttää keskusteluissa käyttäjien kanssa. Heille käyttötilannemalleja voidaan konkretisoida esimerkkitapauksilla eli skenaarioilla (englanniksi scenario). Olio-ohjelmoinnin kielellä sanottuna skenaario kuvaa käyttötilannemallin kuvaaman keskusteluluokan ilmentymän.

Käyttötilannemalli kuvaa siis yleispätevästi, mitä voi tapahtua, kun käyttäjä esimerkiksi haluaa nostaa rahaa pankkiautomaatista. Skenaario taas voi esimerkiksi kertoa, mitä Matti Virtanen kokee, kun hän haluaa palkkapäivänä nostaa tilin tyhjäksi.

Mallit käyttäjän pyrkimyksistä

Palaamme aikaisempaan esimerkkiin Windows 98:n Resource Managerin käytöstä levytilan vapauttamiseen. Nykyisellään se on hyvin vaivalloinen käyttää tilanteessa, jossa käyttäjä haluaa vapauttaa suuren määrän levytilaa. Parempi ratkaisu voisi olla vaikkapa seuraavanlainen:



Jos olemme suunnittelemassa uutta File Manageria, niin mikä tässä on se käyttäjän tavoite, josta pitäisi tehdä käyttötilannemalli?

Käyttäjällä on yleensä joku syy poistaa hakemistoja. Se voi olla joko

- tilan raivaaminen johonkin tiettyyn tarkoitukseen kuten esimerkiksi uuden ohjelman asentamiseen,
- tai jatkuva pyrkimys pitää levy vapaana tarpeettomaksi käyneistä tiedostoista.

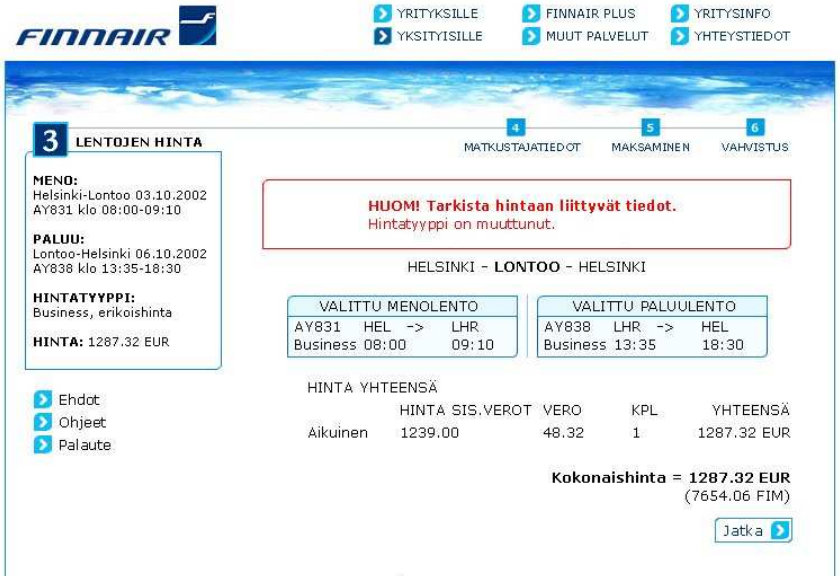
Tilan raivaaminen ja tarpeettomien tiedostojen siivoaminen ovat käyttäjän pyrkimyksiin ja prosesseihin liittyviä tavoitteita, kun taas hakemiston

poisto on systeemitekninen tavoite. Jos otamme käyttötilannemallin aiheeksi tilan raivaamisen tavoitteen, lopputulos on varmasti kovin erinäköinen kuin jos tavoite on vain poistaa hakemistoja.

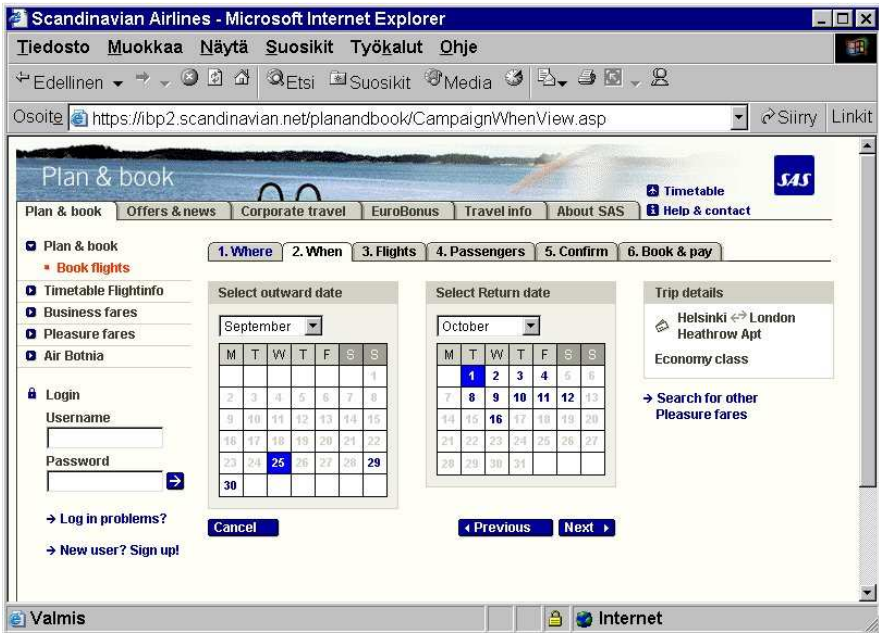
Tämän esimerkin opetus on se, että käyttötilannemallit pitäisi tehdä käyttäjän prosesseihin liittyvistä tavoitteista eikä esimerkiksi systeemiteknisistä tavoitteista.

Sama tavoite – eri käyttötilannemalli

Aikaisemmissa esimerkeissä olemme nähneet yhden ja saman palvelun – lentolipun oston – käyttöön kaksi eri ratkaisua. Seuraava sopii käyttäjälle, joka haluaa matkustaa tiettynä päivänä:



Seuraava taas sopii matkustajalle, joka haluaa edullisen lennon ja jolla on pelivaraa matkan ajankohdan suhteen:



Tästä näkee yhdellä silmäyksellä, milloin edullisia lippuja on tarjolla.

Miten näissä tapauksissa pitäisi valita käyttötilannemallin käyttäjätavoite? Jos asiaa katsoo lentoyhtiön liiketoimintaprosessin näkökulmasta, kummallakin tapauksella on yksi ja sama tavoite – lipun varaus. Olisi helppoa ajatella, että tässä on yksi käyttäjän tavoite, jolle tehdään yksi käyttötilannemalli, joka toteutetaan yhdellä käyttöliittymän dialogilla.

Ja kyllähän käyttäjänkin näkökulmasta käyttötilanteen lopullinen tavoite on kummassakin tapauksessa se yksi ja sama lentolippu. Mutta jos katsomme asiaa käyttäjien pyrkimysten näkökulmasta, niin ensimmäisessä tapauksessa käyttäjän ensisijainen tavoite on lentolippu tietylle päivälle, kun taas toisessa tapauksessa käyttäjän ensisijainen tavoite on edullinen lentolippu joidenkin aikarajojen puitteissa.

Tämän esimerkin opetus on, että yhdelle ja samalle käyttäjän tavoitteelle on joskus syytä tehdä määrittelyvaiheessa useita käyttötilannemalleja. Ennen tilanteen mallinnusta kannattaa pohtia, että voidaanko käyttäjät

jaotella erilaisten prioriteettien mukaan eri ryhmiin, joille kullekin tehdään omat mallinsa. Esimerkiksi lentolipun varauksessa käyttäjät voidaan jaotella liikematkailijoihin sekä omalla rahalla lentäviin, kustannustietoisiin lomamatkailijoihin.

Käyttötilannemallin laatiminen

Käyttötilannemallit voivat palvella ainakin kahta hyvin erilaista tarkoitusta:

1. Keskustelujen sisällön määrittämisen alkuvaiheissa ne toimivat suunnittelutyöhön osallistuvien keskinäisen viestinnän välineenä.
2. Viimeistellyt käyttötilannemallit ovat lähtötietoja käyttöliittymän suunnittelijoille ja toteuttajille.

UML-kirjallisuuden ohjeet käyttötilannemallien kirjoittajille ovat yleensä aika ympäripyöreitä. Yleinen henki tuntuu olevan, että käyttö kuvataan vapaamuotoisen, kertovan tekstin avulla. Hieman tarkempia ohjeita tarjoaa Frank Armourin ja Granville Millerin kirja ”Advanced Use Case Modeling”. Tekijät erottavat kolme kuvaustapaa:

1. Vapaamuotoinen kertova teksti
2. Numeroiduilla toimenpideaskeleet
3. Käytön kuvaus tila-automaattina. Teksti kuvaa kunkin käyttötilan sekä tavat siirtyä tilasta toiseen.

Viimeksi mainittu tapa soveltuu sellaisiin sovelluksiin, joilla on selvästi toisistaan erottuvia toimintatiloja. Tavallinen lankapuhelin on tästä hyvä esimerkki. Se voi olla suljettu, hälyttämässä tulevaa puhelua, numeronottotilassa, hälyttämässä soitettavaa puhelua jne.

Tällaisen tila-automaatin kuvauksen ongelma on, että se ei kuvaa tapahtumia siinä järjestyksessä kuin käyttäjä tyypillisesti ne kokee. Tapahtumien kulun hahmottaminen vaatii lukijalta melkoista ajatustyötä.

Alkuvaiheissa on syytä lähteä siitä, että asioita iteroidaan, ja malleja kirjoitetaan uusiksi. Mallien pitäisi tässä vaiheessa olla helppoja lukea sekä helppoja kirjoittaa. Tässä vaiheessa haetaan asioiden yleiskuvaa. Liiat yksityiskohdat sekä poikkeustilanteiden kuvaukset eivät saisi päästä hukuttamaan tätä yleiskuvaa. Niiden aika tulee sitten, kun malleja viimeistellään käyttöliittymän suunnittelua varten.

Käyttötilanne ja käyttäjän tarpeiden näkökulma

Edellisessä luvussa opimme, että käyttöliittymässä liiketoimintaprosessin tarpeet ja käyttäjän pyrkimyksiin liittyvät tarpeet kohtaavat. Kirjallisuudessa näkee usein käyttötilannemallien esimerkkejä, jotka kertovat lähinnä vain sen, miten liiketoimintaprosessi saa käyttäjältä tarvitsemansa tiedot. Mallien pitäisi myös huomioida käyttäjän ko. tilanteeseen liittyvät tarpeet.

Tarkastelkaamme esimerkiksi lentolipun ostoa netistä (esimerkillä, joka yksinkertaistaa opetuksellisista syistä hieman lennon valinnan todellisuutta). Käyttötilannemallin selostus voisi mennä näin:

Käyttäjä valitsee kohdekentän. Hän kertoo päivämäärän jolloin hän haluaa matkustaa. Hänelle esitetään lista sen päivän lennoista, joilla on tilaa. Käyttäjä valitsee tästä listasta lennon. Jne…

Jos katsomme asioita käyttäjän tarpeiden näkökulmasta, niin huomaamme, että käyttäjällä on ensisijaisesti tarve lentää johonkin tiettyyn matkakohteeseen (esimerkiksi Lontooseen). Tällä matkakohteella saattaa olla useita lentokenttiä (esimerkiksi Lontoon Heathrow, Gatwick ja Stanstead). Käyttäjää täytyy auttaa valitsemaan itselleen sopiva lento itselleen sopivalle kentälle. Siis uusi yritys:

Käyttäjä kertoo halutun lentopäivän. Käyttäjä joko

- valitsee matkakohteen ja saa listan kaikista ko. päivän lennoista kohteen eri kentille, tai
- nimeää yhden tai useamman kentän ja saa listan lennoista ko. päivänä nimetyille kentille.

Käyttäjä valitsee lennon kellonajan mukaan järjestetystä listasta. Valintapäätöstä varten kustakin lennosta esitetään listat saatavissa olevista hintaluokista sekä näiden hinnoista. Lentokentistä käyttäjä voi pyytää kenttäinfon, joka kertoo ainakin

· etäisyyden kaupungin keskustasta sekä ilmansuunnan kaupungin keskustasta katsoen, sekä

· matka-ajan kaupunkiin autolla/taksilla sekä saatavissa olevilla eri julkisilla liikennevälineillä.

Usein käyttäjä tarvitsee eniten apua päätöstilanteissa (esimerkiksi: Minkä kentän valitsen? Minkä lennon valitsen?). Käyttötilannemallin laatijan on syytä kiinnittää erityistä huomiota siihen, kuinka käyttäjää autetaan tekemään tilanteen vaatimat päätökset. Sen pitäisi kertoa, mitä tietoa käyttäjä päätökseen tarvitsee, ja että tämä tieto on järjestetty.

Joidenkin UML-oppaiden mukaan käyttötilannemallien tehtävä on kertoa, mitä luokkakaavioiden määrittelemille olioille tapahtuu käytön tilanteissa. Tämän ohjeen ahdas noudattaminen saattaa johtaa käyttäjän näkökulman hukkaamiseen malleilla, jotka kertovat vain, mitä liiketoimintaprosessi saa käyttäjältä.

Tässä luvussa tarkasteltiin aikaisemmin tilan raivaamista PC:n kovalevyltä. Poistettavien hakemistojen valinta on selvästikin päätöstilanne. Käyttöliittymämallin tekijä joutuu pohtimaan, mitä tietoa ja missä järjestyksessä käyttäjälle tarjotaan tämän päätöksen avittamiseksi. Käyttötilannemallin selostus voi olla esimerkiksi seuraavanlainen:

Käyttäjälle tarjotaan suuruusjärjestykseen lajiteltu lista ylimmän tason hakemistoista. Kustakin hakemistosta esitetään nimi ja koko. Jokainen hakemisto voidaan avata vastaavaksi uudeksi listaksi. Käyttäjä voi monivalita yhden tai useampia hakemistoja. Valittujen hakemistojen yhteiskoko on jatkuvasti näkyvillä. Kun käyttäjä saanut valittua itseään tyydyttävän yhdistelmän hakemistoja, hän käynnistää niiden poiston.

Sidostahot ja –intressit

UML-kirjallisuudessa ei juuri näe käsitettä ”sidostaho” (stakeholder). Käyttötilannemallien yhteydessä käytetään termiä ”actor”. Se on sidostahoa kapeampi käsite, jolle ”osapuoli” lienee sopiva käännös. Sillä tarkoitetaan tahoja, jotka antavat käyttötilanteessa välittömästi tai välillisesti tekemisissä sovelluksen kanssa.

Sidostaholla tarkoitetaan tässä kirjassa laajemmin kaikkia tahoja, joilla on jokin intressi käyttötilanteeseen. Kun käyttötilanteen mallia aletaan tehdä, on hyvä listata kaikki sidostahot sekä heidän odotuksensa tilanteen suhteen.

Esimerkiksi lentolipun varauksen sidostahoja liiketoimintaprosessin puolella selviä sidostahoja voivat olla esimerkiksi

- lennon suunnittelu, joka tarvitsee arviota kokonaiskuormasta,
- muonitus, joka tarvitsee tietoja matkustajamääristä sekä erikoisruokavalioista,
- laskutus, joka tarvitsee yhteystietoja sekä luottokortin tietoja
- markkinointi tarvitsee yhteystietoja sekä tilastoja.

Sidostahoja on parempi etsiä liian isolla kuin liian pienellä haavilla. Esimerkiksi lentoyhtiöllä saattaisi olla monenlaisia liiketoimintaan, markkinointiin tai operatiiviseen toimintaa liittyviä pyrkimyksiä, joita lipun myyntitilanteessa voitaisiin edistää sopivalla informaatiolla. Esimerkiksi kenttäoperaatioilla voisi olla intressi opastaa matkustajia välttämään ruuhkia ja pullonkauloja kentällä. Markkinoinnilla taas saattaisi olla intressi kehua yhtiön kanta-asiakasohjelmaa. Jne.

Asiakaspalvelutilanteiden suunnittelussa kannattaa usein tutkia, voidaanko käyttötilanteessa

- myydä jotain,
- saada jotain hyödyllistä tietoa,
- tai neuvoa käyttäjiä asioissa, jotka tuottavat usein ongelmia ja pullonkauloja.

Lopulliset käyttötilannemallit

Kun suunnittelun osapuolet ovat saaneet aikaan keskustelujen sisällöstä yhteisen näkemyksen, on aika täydentää käyttötilannemallit lähtökohdaksi käyttöliittymän suunnittelijoille.

Kirjallisuudessa käyttötilannemallin sisältö jaotellaan yleensä seuraavasti:

- Osapuolet
- Ennakkoehdot (preconditions)
- Normaali tapahtumien kulku
- Vaihtoehtoiset tapahtumankulut sekä poikkeukset
- Lopputilanne (postconditions).

Ehdotan, että osapuolten luettelemisen sijasta tehdään mahdollisimman laaja luettelo sidostahoista ja heidän odotuksistaan!

Tapahtumien kulun kuvaus muokataan ja täydennetään. Ehdotan että tapahtumien kulku jaotellaan numeroiduiksi askeliksi. Jokaisesta askeleesta kerrotaan seuraavat asiat:

- Ennakkoehdot
- Kehote: miten käyttäjälle kerrotaan, mitä häneltä odotetaan.
- Mahdollisessa käyttäjän päätöstilanteessa tarvittavat tiedot: mitä tietoa missä järjestyksessä käyttäjä tarvitsee.
- Mahdollinen palaute: miten käyttäjälle ilmaistaan, mitä hänen toimenpiteensä seurauksena on tapahtunut
- Lopputilanne

On syytä vielä muistuttaa, että käyttötilannemalli ei kuvaa keskustelun toteutuksen keinoja. Toteutuksen puolelle on eksytty, jos mallissa näkee sellaisia termejä kuin esimerkiksi ”tietokenttä”, ”valintapainike” tai ”komentopainike”.

Käyttäjän toimenpiteiden ennakkoehdot muodostavat ketjuja, jotka määrittävät käyttöliittymän toteutuksen rakenteet. Esimerkiksi lentolipun varauksen vahvistuksen ennakkoehtoina on, että

- lento on valittu,
- lennolla on tilaa, ja että
- käyttäjä on antanut hyväksyttävät yhteys- ja luottokorttitiedot.

Lennon valinnan ennakkoehtoina taas on, että lentopäivä on annettu ja lennon määränpää on valittu.

Tällaiset ketjuuntuvat ennakkoehdot pystyvät kuvaamaan varsin mutkikkaidenkin keskustelujen loogisia rakenteita varsin tyhjentävästi. Joskus kuitenkin sovelluksen käytössä on erotettavissa selvästi eri tiloja. Tällöin aikaisemmin mainittu tila-automaattiin perustuva esitystapa saattaa olla soveliaampi.

Yhteenveto

Käytön keskustelujen suunnittelussa tarvitaan kohdealueen käsitemalli, joka kuvaa käyttäjän sovellusaluetta koskevaa ymmärrystä. Tekniseen toteutukseen liittyvät käsitteet eivät saisi päästä saastuttamaan tätä mallia.

Toiminnan ymmärtämisessä prosessin elinkaarimalli on hyödyllinen väline, joka UML:ssä loistaa poissaolollaan. Elinkaarimalleja on yleensä käytetty tietokannan kuvaamien olioiden analysoimiseen. Tekniikkaa voi kuitenkin yhtä hyvin soveltaa kaikkien käyttäjä- ja sidostahojen pyrkimyksiin liittyvien prosessien kuvaamiseen.

UML:n käyttötilannemallit voivat

- keskustelujen sisällön määrittelyn alkuvaiheessa toimia suunnittelun osapuolten viestinnän apuvälineenä, ja
- määrittelyn lopuksi välittää työn tulokset käyttöliittymän suunnittelijoille.

Käyttötilannemalli tehdään yhdestä käyttäjän tavoitteesta. On syytä olla tarkkana, että mallien aiheeksi valitut tavoitteet ovat käyttäjän pyrkimyksiin ja prosesseihin liittyviä tavoitteita, eivätkä esimerkiksi systeemiteknisiä tavoitteita. Joskus käyttäjien erilaiset tarpeet ja prioriteetit aiheuttavat, että yhdestä ja samasta tavoitteesta on syytä tehdä useita eri käyttötilannemalleja.

Käyttötilanteessa liiketoimintaprosessi kohtaa käyttäjän pyrkimykset. Niinpä käyttötilannemallin pitäisi kertoa,

- mitä liiketoimintaprosessi saa käyttäjältä, ja
- miten sovellus auttaa käyttäjää tässä tilanteessa.

Käyttäjän päätöstilanteisiin on syytä kiinnittää erityistä huomiota: miten sovellus voi auttaa käyttäjää tekemään päätöksen? Mitä tietoja käyttäjä tarvitsee, ja miten järjestettynä?