SYSTEEMITYÖ

Systeemityöyhdistys SYTYKE ry:n jäsenlehti N:o 2/2009

Systeemityöyhdistyksen laivaseminaari 2009:
Softaa sutjakasti – Kuinka pitää projektimopo käsissä

Tuo projektitiimisi vuosihuoltoon, viritä tiimisi tehot huippuun!

- Tekemisen kypsyys: Tarvittava ammattitaidon taso muutosten keskellä, yksilön- ja tiimin kypsyystason arviointi ja kehittäminen
- Onko organisaatiosi valmis uudenlaisiin työskentelytapoihin: Projektimallit, kommunikaatio, muutoshallinta
- Mitkä ovat onnistuneen projektin tunnusmerkit – Kerää parhaat käytännöt omaan ja tiimin tekemiseen

Olemme varanneet tiimillesi syyshuollon ajalle
9.9.2009 - 11.9.2009.

Ennakkovaraa paikkasi nyt osoitteessa
www.ttlry.fi/laivaseminaari

Hyvä tietotekniikan liiton jäsen!

TTL Tietotekniikan liitto ry

Näin päivität tietosi

Voit päivittää jäsentietosi verkkosivuillamme www.ttlry.fi. Tietojen päivittämiseen tarvitset käyttäjätunnuksen (= jäsennumerosi, merkitty jäsenlehtiin) ja salasanasi (= postinumerosi). Jos olet muuttanut salasanasi tai kirjautuminen ei muutoin onnistu, voit lähettää tunnusten tarkistuspyynnön osoitteella jasenasiat@ttlry.fi.
Toivomme sinun erityisesti varmistavan, että sähköpostiosoitteesi jäsentiedoissa on oikea.

Henkilökohtaisempaa palvelua - Sinun eduksesi

Tietotekniikan liitto jäsenyhdistyksineen, osaamisyhteisöineen ja kerhoineen haluaa palvella jäseniään henkilökohtaisemmin ja paremmin, tarjota tietoa juuri Sinua kiinnostavista aiheista. Palvelun parantamiseksi olemme uusineet verkkopalvelumme.

Päivität vain tiedot itseäsi kiinnostavista aiheista ja saat tietoa juuri niistä. Voit päivittää valintasi aina halutessasi. Tietoja ei anneta ulkopuolisille tahoille vaan niitä käytetään ainoastaan TTL:n ja sen piirissä toimivien yhteisöjen tarkoituksiin.

Tietotekniikan liitto ry

Lars Sonckin kaari 12
02600 Espoo

www.ttlry.fi
etunimi.sukunimi@ttlry.fi

jasenasiat@ttlry.fi
p. 020 741 9898
f. 020 741 9889

Julkaisija
Systeemityöyhdistys Sytyke ry
Susanna Koskinen
Talvikkitie 40 A 33, 01300 Vantaa
p. 09 5607 5363
f. 09 5607 5365
sytyke@hennax.fi

Päätoimittaja
Minna Oksanen
minna.oksanen@gmail.com
puh. 040 577 6640

Toimitussihteeri
Susanna Koskinen
sytyke@hennax.fi

Taitto
Speaking Bark Tmi
Katja Tamminen
taitto.kt@speakingbark.com

Toimituskunta 2/2009
Minna Oksanen
Matti Matikainen
Tarja Raussi
Erkki Pöyhönen
Matti Vuori

Lisätietoja lehdestä
www.sytyke.org/lehti

Tilaukset
Systeemityölehti sisältyy yhdistyksen Tietotekniikan liiton suositusten mukaiseen yhdistyksen jäsenmaksuun.
Vuositilaus 30 €
Irtonumerot 8 €
Hyvissä ajoin ennen painatusta tehty vähintään 50 kpl lisätilaus 2 €/kpl.
Tilaukset yhdistyksen toimistosta.

Kansikuva
Ilkka Pirttimaa

Seuraava numero

Toimituskunta:
PrOsy
Markku Niemi

Ilmestyy: to 28.9.2009

Painopaikka
T-Print
Ahokaari 1-3
05460 Hyvinkää
Puh. (019) 475 8500

Painos: 2500 kpl
ISSN 1237-0525
16 vuosikerta, nro. 2

Ilmoitushinnat
Takakansi A4 1200 €
Sisäkannet A4 1000 €
Sisäsivut 1/1 800 €
Sisäsivut 1/2 600 €
Sisäsivut 1/4 400 €

Arvonlisävero 0%
Vakiopaikan vähintään vuodeksi varanneille 20% alennus.

Teksti: Minna Oksanen, lehden päätoimittaja,
BI/DW asiantuntija Digiassa ja
DAMA Finlandin perustajajäsen

Pääkirjoitus:
Pohjatöitä

Äitienpäivä lähestyy ja äitiä on ilo muistaa kakkupalalla heti aamutuimaan. Ennen kuin kakku on lautasella, on tehtävä päätöksiä. Ensinnäkin, leivonko kakun itse vai ostanko valmiin leipomosta? Myös se pitää tietää, onko äidin mielestä mansikkakakku parempi kuin suklaakakku. Kakkua tehdessä teen vielä pohjatöitä esimerkiksi sen mukaan, laitanko täytteeksi banaania vai mansikkarahkaa.

Vastaavasti, ennen kuin tietojärjestelmä on käytettävissä, pitää tehdä päätöksiä ja pohjatöitä. Uuden järjestelmän hankinnan päätöksen tekemiseen tarvitaan tietoa. Liiketoiminnalla on näkemys tarpeesta, joka pitäisi saada jäsennettyä ratkaisumuotoon. Jäsentäminen on pohjatyötä onnistuneen ratkaisun saavuttamiseksi. Pohjatöihin kuuluvat esitutkimus, järjestelmä- ja arkkitehtuurivalinta sekä vaatimusmäärittely ovat toimintoja, jotka on tehtävä, ennen kuin tarpeen täyttävä ratkaisu voidaan konkreettisesti määritellä ja toteuttaa.

Jos pohjatyöt tehdään huolimattomasti tai niitä ei tehdä lainkaan, ei päästä haluttuun lopputulokseen. Varsinaisesta hankintapäätöksestä on asiaa tarkemmin lehdessä 3/2009, joka myös on SYTYKEn 30-vuotisjuhlanumero. Olemmehan nyt tehneet pohjatöitä sen eteen, että suomalaisilla ohjelmistokehittäjillä on mahdollisuus verkostoitua ja oppia toisiltaan hyviä ja vielä parempia käytäntöjä.

Oikein antoisia lukuhetkiä!

Sisällys

Teksti: Matti Matikainen

Tietojärjestelmähankintojen onnistumisen ja epäonnistumisen kriteerit

Kirjoittaja on projektipäällikkönä Digiassa ja toimii Keskuspuiston Nuorkauppakamarin hallituksessa talousvastaavana.

Yritykset hankkivat tietojärjestelmiä liiketoiminnan tarpeisiin. Yritysten tietojärjestelmähankintojen on lähdettävä liikkeelle liiketoiminnan tarpeesta, jolla kehitetään prosesseja, säästetään kustannuksia ja tuotetaan lisäarvoa käyttäjälle.

Ennen vaatimusmäärittelyä tehdään siis liiketoiminnan päätös järjestelmän hankkimisesta ja esitutkimus. Hyvin suunnitellut ja toteutetut tietojärjestelmähankinnat parantavat tehokkuutta, helpottavat työn tekemistä tai tehostavat päätöksen tekoa. Monet organisaatiot hakevat tietojärjestelmä uudistuksilla kilpailuetua tai pyrkivät kuromaan kilpailijoiden etua kiinni. Useat järjestelmähankinnat ovat epäonnistuneet väärien hankinnan lähtökohtien johdosta. Mikäli järjestelmän hankintapäätös tehdään väärin tai puutteellisin tiedoin, on todennäköistä, että järjestelmä ei vastaa liiketoiminnan tarpeita.

Vaatimusmäärittelystä puhutaan usein ohjelmiston toimitusprojektin ensimmäisenä vaiheena. Sitä ennen tehdään kuitenkin paljon. Vaatimusmäärittelyssä määritetään, mitä järjestelmä tulee tekemään. Projektin johto vastaa puolestaan projektin aikatauluttamisesta, budjetista, toiminnoista, riskien hallinnasta ja toteutuksesta. Vaikka projektin muut toteutusvaiheet tehtäisiinkin oikeaoppisesti, saattaa väärin perustein hankittu järjestelmä koitua virheeksi. Mikäli toteutus lähtee heti alussa väärille raiteille, ei lopputulos voi vastata tilaajan tarpeita. Väärin perustein tehty tietojärjestelmähankinta tuo ennakoimattomia lisäkuluja tai pahimmillaan jopa vaarantaa koko yrityksen olemassaolon.

Vaatimusmäärittelyssä tapahtuneet virheet johtavat siihen, että määrittelyn lopputuloksena toteutettava järjestelmä ei vastaa tilaajan tarpeita. Tässä lehdessä käsitellään suurelta osin juuri vaatimusmäärittelyn haasteita.

Ensiksi tulee siis tutkia, voiko tietojärjestelmillä luoda liiketoiminnalle hyötyä. Vaatimusmäärittelyssä puolestaan määritetään, miten liiketoiminnan hyödyt toteutetaan. Hankinnan analysointi saattaa myös pitää sisällään useita vaihtoehtoisia toteutustapoja.

IT-projektien hankinnan onnistumisen ja epäonnistumisen kriteerejä

Suurten tietojärjestelmien hankinta vaatii ammattitaitoa, taitoa ja tietämystä. Suurin osa onnistuneista tietojärjestelmähankkeista käyttää samanlaisia suunnittelun apuvälineitä suunnittelussa, arvioinnissa ja laadun hallinnassa.

Vaikka tietojärjestelmähankintojen tarpeet vaihtelevatkin suuresti eri liiketoiminnan osa-alueilla, hankintaprosessista on identifioitavissa yleisiä onnistumisen ja epäonnistumisen kriteerejä. Näitä kriteerejä on koottu taulukkoon 1.

”Järjestelmähankinnassa tulee varmistaa, että kaikki aspektit on huomioitu”

Mikäli tietojärjestelmä päätetään hankkia, tietojärjestelmähankinta tulee linjata organisaation liiketoimintastrategian, liiketoiminnan vision ja tietojärjestelmästrategian kanssa. Liiketoimintastrategian tulee ensisijaisesti määritellä hankintapäätöksiä ja järjestelmien rakenteita.

Hallinnolliset säännöt on myös syytä huomioida suunnittelun yhteydessä. Esimerkiksi liiketoiminnan, rahoituksen, riskien hallinnan, lainsäädännön tai tekniikan rajoitteet saattavat vaikuttaa projektin kustannuksiin merkittävästi.

Esitutkimus

Ensiksi tutkitaan, mitä hyötyä järjestelmästä on liiketoiminnalle. Tämän jälkeen voidaan tehdä esitutkimus ja tarvekartoitus. Puhun tässä artikkelissa esitutkimuksesta kun puolestaan tarvekartoitus toteutetaan nykyoppien mukaan osana vaatimusmäärittelyä.

Esitutkimuksen lopputuloksena ja ennen vaatimusmäärittelyn aloittamista liiketoiminnan tavoitteet tulee olla määritelty. Esitutkimuksen lopputuloksena syntyy kuvaus liiketoimintaprosesseista, tietojärjestelmäratkaisujen nykytilasta

Onnistumisen tunnusmerkkejä	Epäonnistumisen tunnusmerkkejä
• Realiteetit huomioiva projektisuunnitelma • Yrityksen tavoitteiden ymmärtäminen • Kaikki projektiin osallistuvat tahot ovat sitoutuneet projektin tavoitteisiin ja aikatauluun • Realiteetit huomioiva projektisuunnitelma • Visio ja suunnitelma identifioitu • Johdon aktiivinen osallistuminen • Kokonaisvaltainen taloudellinen tarkastelu • Ulkopuolisten asiantuntijoiden käyttäminen tarvittaessa • Avoin kommunikointi ja tiedottaminen loppukäyttäjille • Toimittajan toimiala ja prosessiosaaminen • Avainhenkilöille varataan mahdollisuus panostaa riittävästi projektiin koko sen keston ajan • Projektin säännönmukainen mittaaminen etukäteen sovituilla mittareilla	• Puutteet tai epätarkkuudet vaatimusmäärittelyssä • Tilaajalla ja toimittajalla tulkinta erimielisyyksiä vaatimusmäärittelyn sisällöstä • Muutokset määrittelyihin tai tarpeisiin kesken projektin • Testauksen suunnittelun ja toteutuksen laiminlyönti • Lyhyen aikavälin tavoitteisiin keskittyminen • Pätemätön järjestelmätoimittaja • Järjestelmätoimittajan mielipiteiden väheksyminen • Johdon etääntyminen projektin valvonnasta • Hankinnan suunnittelun puutteet • Muutosvastarinta • Puutteet muutosprosessin määrittelyssä ja muutoksen johtamisessa • Epärealistiset odotukset kehitettävää tietojärjestelmää • Ylikilpailuttaminen, halvin tarjous voittaa • Tarvittavien taitojen puuttumisen

Taulukko 1. Tietojärjestelmäprojektien hankintojen onnistumisen ja epäonnistumisen kriteerejä.

sekä toimenpide-ehdotus niiden kehittämiseksi. Kuvaukset kuten kohdealueen, prosessimallinnus, prosessikuvaus, sidosryhmä ja toiminto kuvaukset, mutta myös tekniset, kaupalliset ja organisaation kulttuuria tarkastelevat esitutkimukset ovat usein myös tarpeen. Esitutkimusta on mahdollista lähestyä yrityksen kriittisten menestystekijöiden ja niiden tukemisen kautta.

Lehden päätoimittaja Minna Oksanen painottaa myös organisaation tiedon käsittely rutiinien analysoinnin tärkeyttä. Hyvin aikaisessa vaiheessa tulisi kartoittaa, miten tietoa käsitellään organisaatiossa ja miten liiketoiminta hyödyntää sitä. Näin voidaan myös identifioida, miten tiedon käsittelyllä voidaan tuottaa lisäarvoa yritykselle.

On itsestään selvää, että projektilla tulee olla selkeä sisältö, rajaus, tilaaja, roolit ja tavoitteet. Nämä voidaan varmistaa laadukkaasti toimittavan asiantuntijaorganisaation prosessilla.

Liiketoimintahyödyt on myös hyvä identifioida. Kuka käyttää järjestelmää ja mihin? Mitä hyötyä siitä on? Kuinka paljon apua siitä on, kuinka paljon järjestelmän avulla säästetään aikaa ja kustannuksia.

Mikäli tilaajaorganisaatiolla ei ole riittävää osaamista talon sisällä, kannattaa onnistuneen hankinnan turvaamiseksi turvautua luotettavaan ja ammattitaitoiseen järjestelmätoimittajaan. Toimittajan valinnassa kannattaa myös kiinnittää huomiota toimittavan organisaation toimialaosaamisen lisäksi prosessiosaamiseen, kuten esitutkimuksen, liiketoimintaprosessien kuvaamisen ja uudistamisen osaamiseen.

Prosessien uudelleen määrittäminen

”Huonon prosessin automatisointi tietojärjestelmällä ei ole kannattavaa”

Huonon prosessin automatisointi tietojärjestelmillä ei ole kannattavaa. Ennen varsinaista tietojärjestelmän vaatimusmäärittelyä liiketoimintaprosessien uudelleensuunnittelu on usein tarpeen. Ensiksi prosessit tulee tunnistaa ja määritellä, sen jälkeen uudelleen suunnitella.

Prosessien kuvaus on ensimmäinen askel heikkojen kohtien määrittämiseen ja liiketoiminnan kehittämiseen. On siis tärkeää, että liiketoimintaprosessit tunnetaan ja ne on dokumentoitu.

Tietojärjestelmähankkeen kaikissa vaiheessa on kuitenkin muistettava perusasia: järjestelmän kehittämisen lähtökohtana on yrityksen tarve tehostaa toimintaa ja henkilöstön tehokkuutta. Valmistuessaan järjestelmän tulee olla hyödyksi sen käyttäjille.

Riskianalyysi

”Kulman takana piilevät riskit tulisi kartoittaa mahdollisimman aikaisessa vaiheessa”

Riskien kartoitus on tärkeä osa esitutkimusta, jolloin investoinnin kannattavuuteen ja takaisinmaksuaikaan saadaan kokonaisvaltaisempi näkemys. Perinteisten oppikirjamallien mukaan riskien hallinta lähtee projektisuunnitelmasta. Riskien hallinta tulisi kuitenkin ulottaa osaksi suurten tietojärjestelmähankkeiden hankintaprosessia. Erikseen laadittava riskienhallintasuunnitelma tulisi ulottaa liiketoimintaan liittyviin, sopimusteknisiin asioihin, toimittajaan, osallistujien tietotaitoon ja jopa henkilökemioihin liittyviin riskeihin. Toteutuessaan riskit vaikuttavat projektin onnistumismahdollisuuksiin, laajuuteen ja näin myös takaisinmaksuaikaan. Riskien tunnistaminen ja toimenpidesuunnitelma niiden ehkäisemiseksi pienentää niiden toteutumisen todennäköisyyttä merkittävästi.

Hankinnoissa tulee kuitenkin lähteä liikkeelle peruskysymyksestä: kuinka tietojärjestelmillä voidaan kehittää prosesseja, säästää kustannuksissa tai tuottaa lisäarvoa sen käyttäjille.

Teksti: Pasi Mäkinen, Architect Evangelist

Kirjoittaja toimii asiakkaiden ja kumppanien arkkitehtien yhteyshenkilönä kehittäjä- ja alustayksikössä. Käytännön kokemusta arkkitehtuurityöstä hänellä on 10 vuoden ajalta työskentelystä tietotekniikka-arkkitehtina ja arkkitehtitiimin vetäjänä Capgeminillä.

Pilvet arkkitehtuurin yllä

Pilvipalvelut (cloud computing) on tämän hetken puhutuimpia tietoteknisiä ilmiöitä. Olen keskustellut monien asiakkaiden ja kumppaneiden kanssa siitä, mihin eri toimijoiden tarjoamia pilvipalveluja kannattaa soveltaa. Pilvipalvelut on itsessään vielä vakiintumaton termi. Cloud computing ja pilvipalvelut on käsitteenä peräisin tavasta kuvata tietoverkkokuvauksissa Internet pilvenä. Pilvipalveluille on siis ominaista, että niitä käytetään Internetin kautta. Pilvipalvelujen alle lukeutuu hyvin monenlaisia ratkaisuja valmiista sovelluksista kehittäjille tarkoitettuihin alustoihin. Microsoft on yksi suurista pilvialustoja rakentavista tahoista Amazonin, Googlen ja Salesforce.com:n rinnalla. Käytän tämän artikkelin arkkitehtuuriesimerkeissä parhaiten tuntemaani Windows Azure palvelualustaa, josta tarkempi kuvaus myöhemmin kappaleessa Windows Azure palvelualusta.

Millaisia pilvipalveluja on tarjolla

Sovelluspalvelut, kuten Live Mail, Virtual Earth, Google Docs, Office Online, CRM Online ja Salesforce.com ovat sellaisenaan loppukäyttäjäorganisaation tai kuluttajan hyödynnettävissä. Liitännäispalveluiksi kutsutaan pilvipalveluja jotka täydentävät jo hankittua ratkaisua tai tuotetta, kuten Windows Update, Nokian Ovi tai Exchange postipalvelimen arkistointi- ja roskapostisuodatuspalvelut. Lisäpalveluja käytetään tuotteen ylläpitoon tai tuottamaan varsinaiseen päätuotteeseen lisää ominaisuuksia. (Kuva 1)

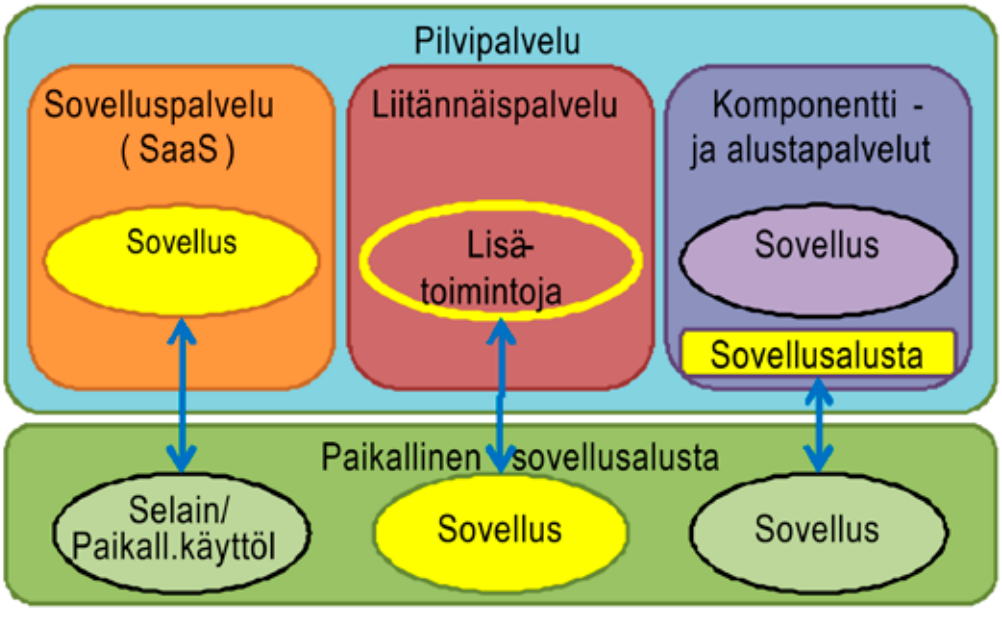
Komponentti- ja alustapalvelut ovat kehittäjien käyttöön tarkoitettuja palveluja, sovellusalustoja ja -kehikoita, joiden toteutus perustuu massiivisesti skaalautuvaan alustaan ja Internetin kautta tarjottuihin rajapintoihin. Komponenttipalveluita voidaan yleensä käyttää niin varsinaisella pilvialustalla toimivissa ratkaisuissa kuin omassa konesalissa tai työasemassa toimivissa sovelluksissa. Komponentti- ja alustapalvelut ovat hyödyllisiä ratkaisuja kehittäville organisaatioille, oli sitten kyseessä loppukäyttäjäorganisaatio, palvelutoimittaja tai ohjelmistotalo.

Mihin pilvialustoja voisi soveltaa?

Aloittavien yritysten kannalta pilvialustat tarjoavat nopean tavan rakentaa uusia palveluja. Palvelun kehittäminen on nopeaa kun aikaa vievä ja kallis tuotantoympäristön hankinta ja pystyttäminen jää pois. Mikäli alusta tarjoaa Azure palvelualustan tapaan valmiin sovellusmallin, hyötyy yritys myös siitä että ratkaisu on automaattisesti skaalautuva ja tuotannon operointi on alustatoimittajan toimesta pidemmälle automatisoitua. Tämä säästää henkilökuluja tuotannon hallinnassa. Myös suurilla yrityksillä on monesti yksittäisiä liiketoimintayksiköitä jotka muistuttavat toimintamalliltaan aloittavaa yritystä. Uusia toimintamalleja ja palveluja on nopeampaa kokeilla ilman perinteisen IT-organisaation hitausmomentteja.

Toinen hyvin pilvialustalle sopiva sovelluskategoria on ns. write once -sovellukset, jotka tehdään yksittäistä tapahtumaa tai kampanjaa varten. Esimerkkinä tapahtumaa varten räätälöitä-

Kuva 1. Pilvipalvelutyypit



vät markkinointi-, ilmoittautumis- ja palautesivut. Pilvipalvelut skaalautuvat helposti isommankin tapahtuman tarpeisiin. Ehkäpä Windows Azurella voisi myös tehdä lippujärjestelmän jota yksi Madonnan keikka ei kaataisi.

Perinteiset yrityssovellukset, kuten tuotannonohjaus, eivät ole vielä asiakkaiden ja kumppaneiden mielestä vielä otollisia siirrettäviksi pilvialustalle. Niiden osalta asiakkaiden on ensin totuttava ajatukseen ja myös pilvialustojen on vielä tarjottava lisää valinnanvaraa palvelutason, maantieteellisen sijainnin ja integraatiomahdollisuuksien suhteen. Mielenkiintoisen avauksen ERP-sovellusten suhteen on tehnyt Yhdysvaltalainen Epicor, joka on lähtenyt toteuttamaan mm. hakupalveluja ERP-järjestelmän liitännäispalveluksi pilvipohjaisena toteutuksena.

Millaisia arkkitehtuureja pilvipalvelualustalle kannattaa suunnitella?

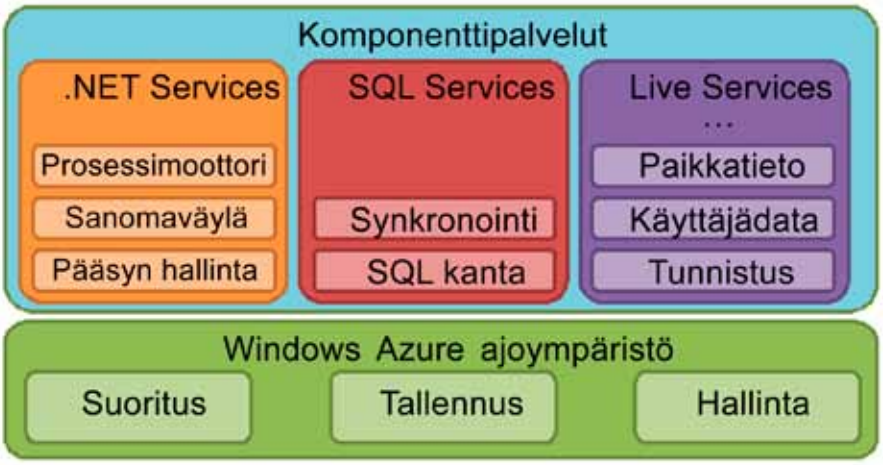
Muutamat suomalaiset kumppanimme ovat edelläkävijöinä jo rakentantamassa sovelluspalveluratkaisuja Azure palvelualustalle, mutta erityisesti tietotekniikan loppukäyttäjinä toimivat yritykset ovat ihmeissään mitä pilvipalvelualustat tarkoittavat. Myös alan toimijoiden kesken on eriäviä näkemyksiä siitä, mihin suuntaan pilvipalvelualustoja pitäisi kehittää. Vaikka palvelualustoja tarjoavia tahoja onkin vähän, on niiden lähestymistavoissa jo eroa.

Toistaiseksi ainoa tuotannossa oleva alusta (beta-status poistettu ja hinnoittelu julkistettu) on Amazonin Elastic Computing Cloud (EC2). EC2:n lähestymistapa on tarjota pienin yhteinen nimittäjä, jonka päälle voi rakentaa hyvin monenlaisia palveluja. EC2:n päällä voi ajaa Windows Server 2003 ja Linux virtuaalikoneita. Tämä jättää asiakkaalle vapauden rakentaa hyvin monenlaisia ratkaisuja. Alustan vapauden vastapainona asiakkaan on myös vastattava itse ratkaisun skaalautuvuuden ja korkean käytettävyyden saavuttamisesta. Myös ratkaisun palvelutason valvonta ja ohjaus on järjestettävä itse. Amazon valvoo vain onko virtuaalikone pystysssä.

Muut suurimmat pilvialustojen rakentajat ovat määritelleet alustojensa rajapinnat ylemmälle sovellustasolle. Google AppEngine ajaa Python sovelluksia, Salesforce.com Force-alusta ajaa Apex-kielellä kirjoitettuja sovelluksia ja Windows Azure-alustalla voi ajaa .NET sovelluksia sekä FastCGI-rajapintaa tukevia sovelluskehikoita, kuten PHP. Sovellusmallin rajoittamisella tavoitellaan parempaa tuotannon palvelutason valvonnan ja hallinnan automatisointia.

Windows Azure palvelualusta

Windows Azure palvelualusta koostuu kahdesta pääkerroksesta: Windows Azure ajoympäristöstä ja kokoelmasta komponenttipalveluja joita voidaan käyttää niin Azure ajoympäristössä kuin omassa konesalissa tai työasemalla suoritettavista sovelluksista. (Kuva 2) .



Kuva 2. Windows Azure -palvelualustan esiversion komponentit.

Windows Azure ajoympäristö tarjoaa palvelut sovellusten suorittamiseen ja sovelluskokonaisuuden hallintaan sekä tallennuspalvelut binääritiedostojen, XML-datan ja sanomajonojen hallintaan. Komponenttipalvelut tarjoavat sovelluskehittäjälle mekanismit relaatiomuotoisen tiedon hallintaan ja synkronointiin, tunnistukseen, pääsyn hallintaan, sanomaintegraatioon, prosessien suorittamiseen, paikkatiedon ja käyttäjädatan hallintaan sekä moniin muihin tehtäviin.

Pilvisovelluksen arkkitehtuurimalli

Azure antaa valmiin sovellusmallin, jossa .NET sovellus voi koostua erilaisista web role ja worker role -komponenteista. Web role -komponentti toteuttaa joko käyttöliittymän ASP.NET -selainsovelluksena tai sanomapohjaisen sovellusrajapinnan Windows Communication Foundation (WCF) -palveluna. WCF-palvelun avulla voi toteuttaa web service, REST ja Atom Publishing -pohjaisia rajapintoja. Worker role -komponentti on tarkoitettu pitkäkestoisempien tehtävien taustalla suorittamiseen. Tyypillisesti worker role saa herätteitä Azure ajoympäristön tarjoaman jonomekanismin kautta web role -komponenteilta. Toki worker role voi myös hoitaa erilaisia ajastettuja tehtäviä. Kukin komponentti-instanssi pyörii Azure alustalla omassa virtuaalikoneessaan.

Monesti Azuren web role - worker role -malli ymmärretään väärin perinteiseksi kerrosarkkitehtuurin käyttöliittymä - liiketoimintalogiikka - tiedonhallinta -rakenteeksi. Web role - worker role -mallin tarkoituksena on kuitenkin erottaa toisistaan käyttäjäinteraktio ja pitkäkestoinen prosessointi, jotta kumpaakin voidaan skaalata toisistaan riippumatta. Esimerkkisovelluksia laajemmissa todellisissa liiketoimintajärjestelmissä kumpikin rooli kannattaa itsessään rakentaa kerrosrakennetta noudattaen. Jos sovellusta tarkastellaan loogisella tasolla, voidaan sovelluksen kerrosrakenne esittää oheisen kuvan (Kuva 3) mukaisesti. Tiedonhallinta ja liiketoimintalogiikka ovat omia loogisia kerroksiaan ja ylin kerros voidaan jakaa

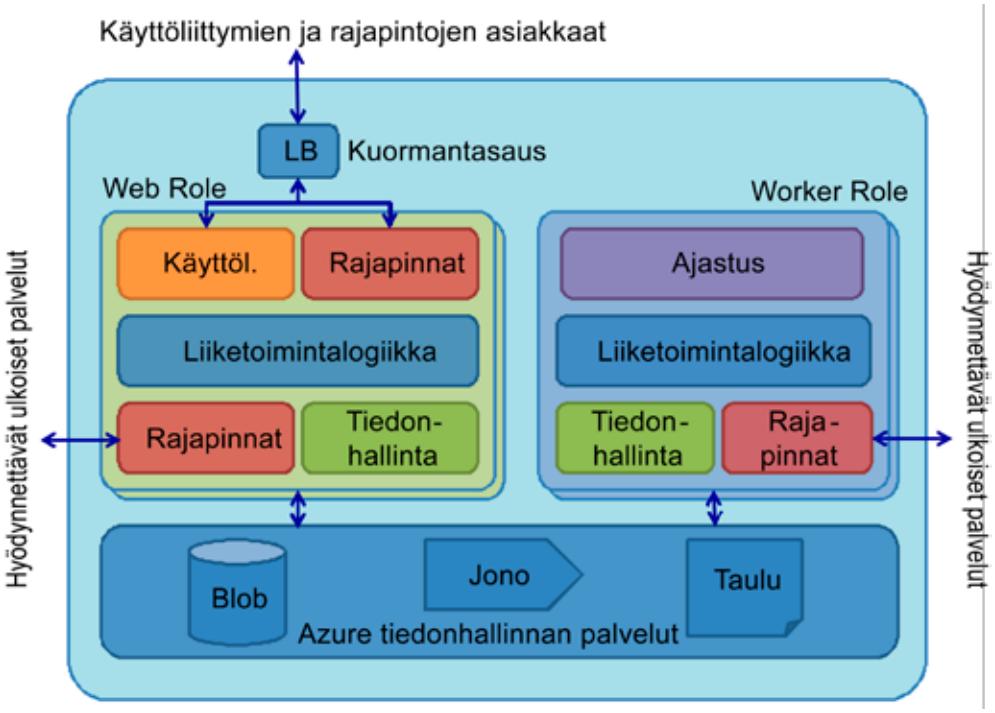
Kuva 3.
Looginen sovellusrakenne.

kolmeen osaan, jotka kaikki käyttävät yhteisiä liiketoimintakerroksen palveluja. Yksi osa koostaa palveluista käyttöliittymiä, toinen muille sovelluksille tarjottavia rajapintoja ja kolmas hoitaa liiketoimintapalvelujen avulla pitkäkestoista taustakäsittelyä.

Jos sama kuva esitetään Azuren web role -worker role -malliin sovitettuna (Kuva 4), nähdään että kummallakin roolilla on sekä liiketoimintalogiikkaa että tiedonhallintaan liittyvää koodia.

Web role -käyttöliittymäosuus tarjoaa selainpohjaisen käyttöliittymän, jota voidaan terästää AJAX ja Silverlight tekniikoilla. (Mikä tahansa webpalvelimelta jaeltava käyttöliittymätekniikka käy, kuten XBAP ja Flash.) Web rolen rajapintaosuus tarjoaa ulkopuolisille sovelluksille sanomapohjaiset sovellusrajapinnat web service, REST tai Atom Pub rajapinnoilla. Toteutustekniikkana on .NET 3.x:n Windows Communication Foundation. Rajapinnan asiakkaana voi olla esimerkiksi jokin liiketoimintasovellus .NET Services -palvelun Service Bus -mekanismin kautta, perinteinen asennettu käyttöliittymä tai vaikkapa Outlook-sovellukseen integroitu käyttöliittymä.

Kuva 4.
Sovellus Azure-palvelualustalla.



Worker role -ajastusosuus koordinoi pitkäkestoisempaa prosessointia joko jonosta tiedonhallintakerroksen välityksellä vastaanotettujen tapahtumien tai erillisen ajatusmekanismin perusteella. Worker role -komponentteja voi myös ketjuttaa jonojen välityksellä käsittelyketjuiksi. Jonosta luetut tapahtumat voivat päätyä eri worker role instansseille, joten jonon kautta välitetyille tapahtumille ei voida taata tiettyä käsittelyjärjestystä.

Web ja worker role voivat jakaa tiedonhallintaan ja rajapintoihin liittyvää koodia. Nykyinen Azure SDK tarjoaa valmiin Storage Client -kirjaston (lähdekoodeineen) joka yksinkertaistaa Azure -alustan tiedonhallinnan palvelujen käyttöä. Vastaavasti tiedonhallintakerrokseen kuuluu esim. Azure-palvelualustan SQL Services (sovelluksen oma relaatiomallinen data) tai Live Services (käyttäjän omistama XML tai blob data) -palvelujen käyttö. Ulkoisten palvelujen käyttö kuuluu myös loogisesti tiedonhallintakerroksen kanssa samalle tasolle. Käytettäviä ulkoisia palveluja voivat olla esimerkiksi .NET Services -palvelun Service Bus -mekanismin kautta erilaiset yrityksen sisäiset liiketoimintajärjestelmät, muut Live Services -palvelut tai kokonaan muilla pilvialustoilla pyörivät palvelut.

Pilvipalvelut ovat seuraava kehitysaskel

Vaikka pilvipalvelut hakevat vielä muotoaan, uskon että kyseessä on seuraava merkittävä kehitysaskel tietojenkäsittelyn kehityskulussa. Kuten meillä on edelleen käytössämme keskuslaitepohjaisia ratkaisuja, eivät pilvipalvelutkaan sovi kaikkeen, eikä vanhaa ja toimivaa kannata niillä väkisin korvata. Monenlaisia uusia mahdollisuuksia ne kuitenkin tarjoavat. Elämme mielenkiintoisia aikoja!

Hyödyllisiä artikkeleita aiheesta löytyy mm. David Chappell & Associates konsulttitoimistolta:

1. *A Short Introduction to Cloud Platforms: http://www.davidchappell.com/CloudPlatforms--Chappell.pdf*
2. *Introducing Windows Azure: http://www.davidchappell.com/writing/white_papers/Introducing_Windows_Azure_v1-Chappell.pdf*
3. *Introducing the Azure Services Platform: An Early Look at Windows Azure, .NET Services, SQL Services, and Live Services: http://www.davidchappell.com/writing/white_papers/Azure_Services_Platform_v1.0--Chappell.pdf*

Teksti: Pauli Kartano
Jukka Uusitalo

Kokonaisarkkitehtuuri valtionhallinnon tietojärjestelmätyössä

Pauli Kartano toimii projektipäällikkönä valtiovarainministeriössä Koulutukseltaan hän on tuotantotalouden DI ja ollut mukana ValtIT-hankkeessa 2006 lähtien. Sitä ennen hän on ollut Pääesikunnassa. Työt ovat liittyneet pääasiassa kokonaisarkkitehtuuri-kehitykseen.

Jukka Uusitalo toimii neuvottelevana virkamiehenä valtiovarainministeriön hallinnon kehittämisosastolla Valtion IT-toiminnan johtamisyksikössä. Hän on yhteentoimivuuden kehittämisohjelman vastuuhenkilö ja hänen vastuualueenaan on yhteentoimivuuden ja valtionhallinnon kokonaisarkkitehtuurin kehittäminen ja kehittämisen ohjaus. Aiemmin Uusitalo on toiminut arkkitehtuurin kehittämistehtävissä puolustusvoimissa.

Valtionhallinnon kokonaisarkkitehtuuri on työkalu koko valtionhallinnon toiminnan ja tietojärjestelmien kehityksen ohjaamiseen ja yhdenmukaistamiseen. Yhdenmukaistamisessa on keskeistä määrittää yhteiset toiminnot ja palvelut sekä varmistaa kokonaisuuden yhteentoimivuus. Valtionhallinnon kokonaisarkkitehtuuri käsittää kaikki arkkitehtuurin keskeiset näkökulmat eli toiminnan, tiedot, tietojärjestelmät sekä teknologiat. Viime vuosina on kokonaisarkkitehtuurissa alettu kuntasektorin kehitystyön myötä alettu tarkastella koko julkishallinnon järjestelmien kokonaisuutta.

Kokonaisarkkitehtuurin tavoitteet ja sisältö

Valtionhallinnon kokonaisarkkitehtuuri sisältää sekä valtiotason yhteiset linjaukset ja arkkitehtuurikuvaukset että eri osapuolien, kuten hallinnonalojen, kohdearkkitehtuurit. Siten kokonaisarkkitehtuurilla on useita eri kehittäjiä ja ylläpitäjiä. Valtiovarainministeriön tehtävänä on erityisesti valtiokonsernin ylätason yhteisten linjausten ja arkkitehtuurin kokonaisrakenteen määrittely, kertoo neuvotteleva virkamies Jukka Uusitalo valtion IT-toiminnan johtamisyksiköstä (ValtITstä). Myös kokonaiskuva ja erilliset palvelukohtaiset arkkitehtuurikuvaukset ValtIT:ssä suunniteltavista ja toteutettavista valtionhallinnon yhteisistä IT-palveluista ovat keskeinen osa kokonaisarkkitehtuuria. Projektipäällikkö Pauli Kartano, ValtITstä kuvailee arkkitehtuurikuvausten ylätasoa kartaksi, josta voidaan tunnistaa yhteiset palvelut, eri osapuolien vastuulle kuuluvat toiminnot sekä vielä kehitystä vaativat valkoiset alueet. Kokonaisarkkitehtuurin osaksi on tarkoitus saada myös hallinnonalojen omia arkkitehtuureja eli kuvauksia esimerkiksi terveydenhoidon, liikenteen tai opetuksen palveluista ja toimintaprosessista, tiedoista sekä tietojärjestelmistä ainakin niiltä osin kuin nämä kiinnostavat muita hallinnon osapuolia. Näiden kohdearkkitehtuurien kehittämisestä ja ylläpidosta vastaavat esimerkiksi ministeriöt. Kohdearkkitehtuurit liitetään kokonaisarkkitehtuuriin niin, että tiedot ovat kaikkien käytettävissä.

Valtionhallintotasoisesta käsittelystä ollaan oikeastaan siirrytty tarkastelemaan koko julkishallintoa. Siten kokonaisarkkitehtuurin piiriin ovat tulossa myös kunnat ja välillisen valtionhallinnon organisaatiot, kuten Kansaneläkelaitos.

Tehty työ ja tulokset

Valtionhallinnon kokonaisarkkitehtuurimenetelmä valmistui vuonna 2007. KuntaIT on tämän jälkeen laatinut menetelmästä jatkokehitetyn version. Vastikään on käynnistetty työryhmä, joka

tulee tuottamaan näiden yhdistelmänä julkishallinnon yhdenmukaisen arkkitehtuurimenetelmän. Menetelmä julkaistaan julkishallinnon suosituksena (JHS) ja sen suunnitellaan valmistuvan vuoden 2009 loppuun mennessä. Jyväskylän yliopiston vetämässä FEAR-projektissa on myös laadittu ohjeita ja malleja menetelmän käyttämisestä hankkeissa.

Kokonaisarkkitehtuurityön osana on tehty laaja valtionhallinnon nykyisten tietojärjestelmien kartoitus ja laadittu arkkitehtuurin toiminta- ja ohjausmalli. Nykyjärjestelmiä saatiin valtavasti listattua, mutta samalla myös huomattiin miten vaikeaa tietojen saaminen valtionhallinnon organisaatiolta oli kun tällaisia arkkitehtuurikuvauksia ei toistaiseksi juurikaan ole, toteaa Pauli Kartano. Toiminta- ja ohjausmalli kuvaa, miten arkkitehtuuria voidaan käyttää koko valtionhallinnon ja eri hallinnonalojen toiminnan kehittämiseen ja ohjaamiseen. Toiminta- ja ohjausmallin käyttöönotto on kunkin hallinnonalan oman aktiivisuuden varassa ja sellaisenaan mallia ei liene otettu käyttöön missään, kertoo Jukka Uusitalo. Hän jatkaa, että toisaalta arkkitehtuuritietoisuus on selvästi kasvanut ja maaperä valtionhallinnossa on tullut viime vuosina otollisemmaksi yhteiskäyttöisille ratkaisuille ja yhteentoimivuuden vaatimuksille.

Konkreettisimmat arkkitehtuurin osat ovat syntyneet ValtIT:n yhteisten IT-palveluiden kehityksen ja valtionhallinnon organisaatioiden kokonaisarkkitehtuurityön tuloksena. ValtIT:n hankkeissa on suunniteltu ja toteutettu keskeisiä kokonaisarkkitehtuurin osia. Näiden hankkeiden suunnitelmat ovat tavoitetilan arkkitehtuurin kuvauksia, kuten asiointitilin tavoitetilan arkkitehtuurikuvaukset. Toteutetut tai pian valmistuvien palveluiden kuvaukset ovat taas osa valtionhallinnon kokonaisarkkitehtuurin nykytilaa, kuten vuoden 2010 alussa käyttöönotettava valtionhallinnon yhteinen integraatioalusta. Jotkut ministeriöt ja virastot ovat myös laatineet omia kokonaisarkkitehtuurejaan tai niiden osia. Pauli Kartano toteaakin, että useita kokonaisarkkitehtuurin osia on jo olemassa, mutta keskeinen ongelma on osat yhdistävän ylätason puuttuminen.

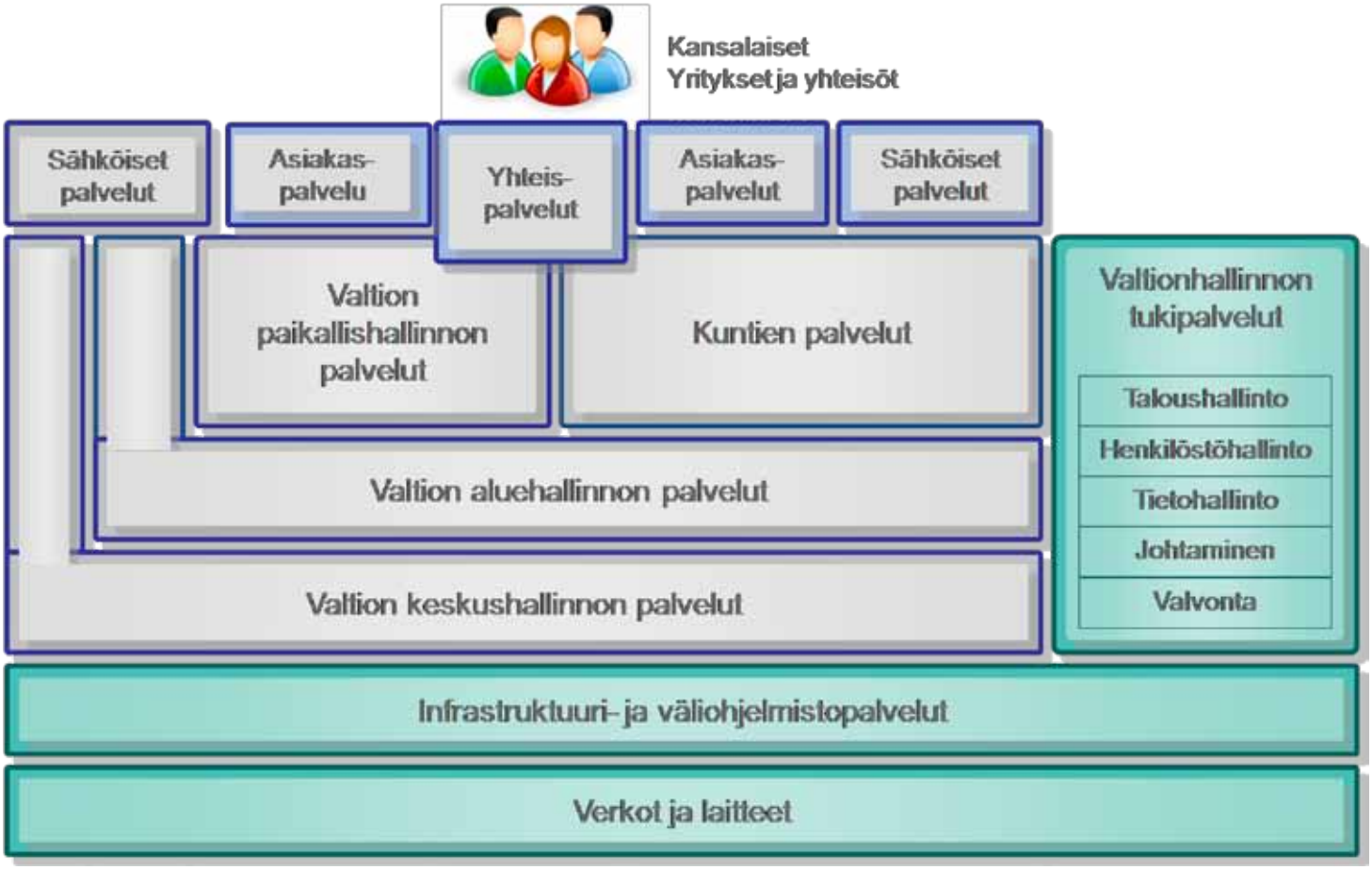
Lähiajan suunnitelmat

Valtiovarainministeriö on käynnistämässä Valtiotason arkkitehtuurit –hanketta, jonka vetäjänä tulee toimimaan Jukka Uusitalo. Hankkeen tavoitteena on luoda ylätason arkkitehtuurilinjauksia ja –kuvauksia ohjaamaan ja tukemaan valtionhallinnon toiminnan ja tietojärjestelmien kehitystä. Hankkeessa tuotetaan valtionhallinnon arkkitehtuuriperiaatteet ja -linjaukset sekä niihin perustuva valtiotason toiminnallinen, tieto ja IT-palveluiden arkkitehtuuri. Arkkitehtuureista pyritään erityisesti määrittelemään niiden rakenne, karkea sisältä sekä eri osien omistajat. Erityisesti tietoarkkitehtuurissa huomioidaan

koko julkishallinto. Samassa hankkeessa määritellään myös valtioneuvoston kokonaisarkkitehtuuri. Hanke toteutetaan vuosien 2009 ja 2010 aikana, kuitenkin siten, että ensimmäiset versiot tuloksista valmistuvat jo vuoden 2009 lopussa.

Toinen keskeinen vuodelle 2009 suunniteltu tehtävä liittyy perustietovarastojen käytön edistämiseen. Uusitalo linjaa teknisten ja hallinnollisten ratkaisujen käsittelyn omiksi projekteiksi ja suunnittelee, että työ voitaisiin aloittaa perustietovarastojen tietojen välittämisen tavoitetilan arkkitehtuurin laatimisella. Työssä määriteltäisiin yhdenmukainen ja keskitetty ratkaisu, jolla tiedot olisivat eri osapuolien saatavissa. Myös perustietovarastojen rajapintojen määrittely ja kuvaaminen on olennaista. Tavoitetilassa otetaan toki myös huomioon nykyiset toimivat tiedonjakeluratkaisut. Erillisenä projektina käsitellään myöhemmin tietojen jakelun hinta, vastuu ja muita kysymyksiä.

Syksyllä 2009 on suunnitteilla avata yhteentoimivuusportaali, joka on keskeinen kokonaisarkkitehtuurin kuvausten ja tiedon jakelukanava, kertoo portaalin kehittämisprojektia vetävä Pauli Kartano. Portaalista käyttäjien on helppo saada esimerkiksi valtiotason arkkitehtuurikuvaukset, hallinnonalojen kohdearkkitehtuurien kuvauksia tai tarkempia kuvauksia yhteisistä IT-palveluista. Jatkossa portaaliin tulee myös keskeisten tietovarastojen rajapintojen kuvauksia. Yhteentoimivuusportaalista löytyy myös arkkitehtuurimenetelmä ja muuta tukimateriaalia.



Teksti: Ari Hovi

Tietoarkkitehtuuri

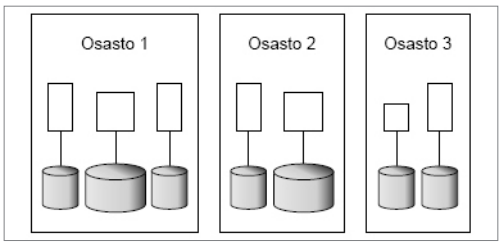
Kirjoittaja toimii tiedonhallinnan konsulttina yrityksessään Ari Hovi Oy, joka myös järjestää laajasti tiedonhallinnan koulutusta.

Lähtötilanne

Yritysten ja julkishallinnon organisaatioiden järjestelmien levyillä pyörii suuri määrä arvokasta, vuosien varrella suurella vaivalla ja kustannuksilla kerättyä tietoa. Tietojen talteen saaminen ei ole yksinkertaista. Nykyisin suurin osa käytössä olevista tietojärjestelmistä ei ole itse tehtyjä, vaan ostettuja sovelluspaketteja. Niiden tietosisällön ja tietorakenteiden kuvauksia ei useinkaan ole saatavilla. Tärkeät omaa toimintaa kuvaavat tiedot ovat siis usein tuntemattomissa tietorakenteissa.

Useimmiten tietojärjestelmiä on useita, on taloushallintaa, henkilöstönhallintaa ja lisäksi omaa toimintaa tukevia muita järjestelmiä. Järjestelmien omistajat ovat yleensä tietyllä osastolla tai jossakin vastaavassa organisaatioyksikössä. Ongelmana on ns. tietojen siiloutuminen eli eriytyminen irrallisiksi saarekkeiksi, ks. kuva 1. Samoja tietoja, kuten asiakkaita, oman organisaation rakenteita ja henkilötietoja on talletettuna monessa eri järjestelmässä.

Kuva 1.
Organisaation tiedot jakautuvat usein erillisiin ”siiloihin”.



Siiloutuminen johtuu osittain organisoinnista. On järkevää, että kukin osasto tai toiminto ajaa omaa asiaansa ja hoitaa omia tietojaankin. Ongelma tulee siitä, että kenelläkään ei ole kokonaiskuvaa organisaation tiedoista. Yleensä ei ole yli organisaatio- tai sovellusrajojen meneviä tietomalleja. Ei ole myöskään organisaatioyksikköä, jolle tällaisten kokonaiskuvausten laatiminen ja ylläpito kuuluisi.

Miksi tietoarkkitehtuuri

Tietoarkkitehtuuria voisi verrata uuden asuinalueen suunnitteluun. Asiat eivät etene niin, että sinne tänne rakennetaan erilaisia taloja ilman kokonaissuunnitelmaa. Kun uutta asuinaluetta aiotaan rakentaa, tehdään ensin asemakaava. Se on helikopterikuva tulevasta asuinalueesta, siis sen verran ylhäältä katsottuna, että kokonaiskuva syntyy, mutta yksityiskohdat eivät näy (eikä niitä vielä ole suunniteltukaan). Me tietoammattilaiset taas määräämme usein tarkkojen asunto- tai korkeintaan talokohtaisten piirustusten eli tässä tapauksessa tietomallien kanssa ja ylemmän tason asemakaavamalli puuttuu. Kun tarkkoja piirustuksia eli esim. 250 käsitettä sisältävää käsitemallia sitten esitellään ohjausryhmässä tai johdon palaverissa, ymmärrystä ei synny. Asemakaavatason kaaviolla taas päästään keskustelemaan ja aletaan puhuakin samoista asioista.

Tietoarkkitehtuurin tavoitteena on kuvata organisaation tiedot eri tasoilla, sekä helikopterikuvana yleiskuvan saamiseksi siiloutuneista tiedoista että tarkalla tasolla.

Kuvausmenetelmänä on tietokantojen suunnittelusta tuttu käsiteanalyysi (ER-modeling), jolla nyt kuvataan sekä ylätason asemakaavamalleja että tarkempia aluekohtaisia malleja. On siis tärkeää, että laaditaan eritasoisia malleja. Ylimmän tason malli mahtuu A4-arkille ja se toimii hyvin ”esittelykalvona” ja esim. tietointegraation suunnittelun pohjana. Siitä saa kokonaiskuvan siitä, mitä tietoja organisaatiossamme on ja oikein kuvattuna vielä missä eri järjestelmissä tiedot ovat. Tämä auttaa mm. tietovarastojen ja tietointegraation suunnittelua suuresti.

Ylätason mallinnuksessa ei välttämättä tarvitse kuluttaa paljon aikaa. Edellisessä tekemässäni koko yrityksen tietoarkkitehtuurimallinnuksessa meni alle kymmenen päivää. Tämä edellyttää hyvää organisointia eli oikeiden ihmisten paikalle saamista ja lisäksi kokemusta mallinnuksen vetäjältä.

Mitä on tietoarkkitehtuuri

Tietoarkkitehtuurin tarkoituksena on kuvata yrityksen tai julkisyhteisön keskeinen tietosisältö yli organisaatio- ja tietojärjestelmärajojen. Kuvauksissa käytetään tietomalleja eli piirustuksia ja kaavioita sekä sanallisia määritelmiä. Tietoarkkitehtuuria voivat hyödyntää eri tasot organisaatiossa, joten tarvitaan sekä yleisempiä kokonaismalleja että tarkempia aluekohtaisia kuvauksia. (Kuva 2)

Tietoarkkitehtuurissa kuvataan perusjärjestelmissä ja tietovarastoissa olevat tiedot; laajimmillaan myös tietoja taulukkolaskimissa, dokumenteissa ja arkistoissa. Kun tiedot yhteisesti kuvataan ja määritellään, aletaan puhua yhteistä kieltä, vältytään usein kalliiksi tulevilta väärinymmärryksiltä. Tietoarkkitehtuuri parantaa näin tietojen semanttista (merkitykseen liittyvää) yhteensopivuutta ja siirrettävyyttä. Tietomallien avulla voi kohdentaa missä eri järjestelmissä tiedot ovat, mikä helpottaa mm. tietointegrointia.

Tietoarkkitehtuurin etuja

Tietoarkkitehtuurin mallien avulla voidaan mm.

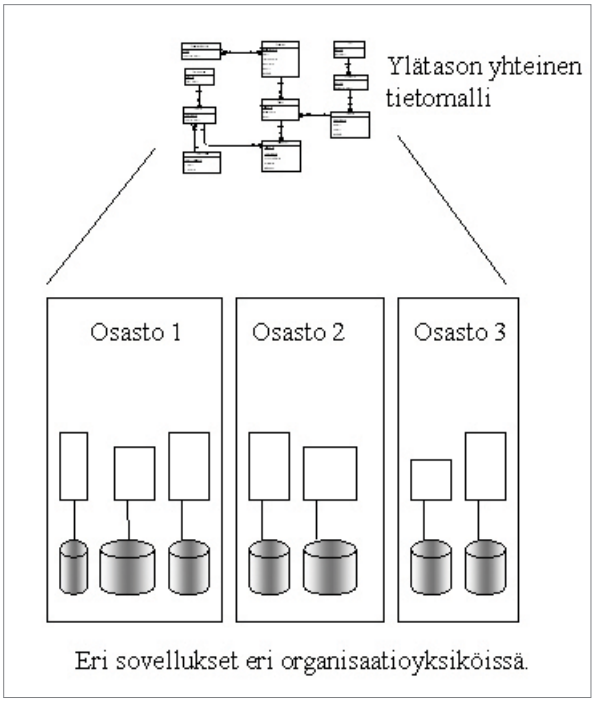
- saada kokonaiskuva organisaation tiedoista
- suunnitella tietovarastojen kehittämistä ja sitä kautta kehittää raportointia ja Business Intelligence -ratkaisuja
- saada toimittajilta parempia ja tarkempia tarjouksia lisäämällä tietomallit tarjouspyyntöihin
- suunnitella järjestelmien kehittämistä ja laajennuksia
- ohjata ja kehittää tietojen integrointia eri sovellusten välillä
- helpottaa SOA-arkkitehtuurin käyttöönottoa

Tyypillistä tietovarasto- / BI –hanketta käynnistäessä ei tietomalleja tai tietoarkkitehtuuria löydy. Niinpä itse DW- / BI–hankkeessa joudutaan yleensä tekemään ”minitietoarkkitehtuuri” eli selvittämään missä tarvittavat tiedot ovat, mitä ne tarkoittavat ja tehtävä sitten mallinnus. Jos tietoarkkitehtuuri olisi jo valmiina, päästäisiin paljon nopeammin liikkeelle ja tulokset olisivat paremmat.

Tietovarastojen yhteydessä näkyy tietoarkkitehtuurin puuttuminen toisellakin tavalla. Ilman sitä on vuosien varrella eri puolilla organisaatiota rakennettu lukuisia erillisiä datamartteja ja raportointikantoja – eli lisää siiloja.

Mallinnuksesta

Kuten edellä todettiin, mallinnuksessa käytetään perinteistä ER-menetelmää, mutta sovellettuna UML luokkakaavio käy oikein sovellettuna myös, kyse on tällöin vain notaatiosta. Mallinnusistunnot ovat keskeisellä sijalla. Tuloksena syntyy tietomalli (käsitemalli), mutta tärkeinä sivutuloksina on vielä kaksi asiaa. Ensinnäkin mukana olevat ihmiset oppivat paljon toisiltaan (”ai teidän osastolla ajatellaan noin, no nyt ymmärrän”) ja sitä kautta puhuvat enemmän yhteistä kieltä. Tämä asia on erittäin keskeistä, jokainen löytää esimerkkejä pitkäaikaisista väärinymmärryksistä eri porukoiden välillä – onko kukaan uskaltanut laskea mitä se maksaa? Toiseksi mallinnuksessa mukana olevat ihmiset sitoutuvat: ”tämä on meidän malli meidän tiedoista” (ei siis jonkun konsultinretaleen taas tekemä kummallinen häkkyrä).



Kuva 2. Tietoarkkitehtuurissa kuvataan tiedot yli sovellusten ja organisaatioyksiköiden.

Tarvitaan lisäksi tarkempia malleja eli yksityiskohtaisempia piirustuksia, jotka puolestaan ovat enemmän IT-ammattilaisia varten. On tärkeää, että tarkemmat aluekohtaiset mallit mappautuvat hyvin ylätason malliin.

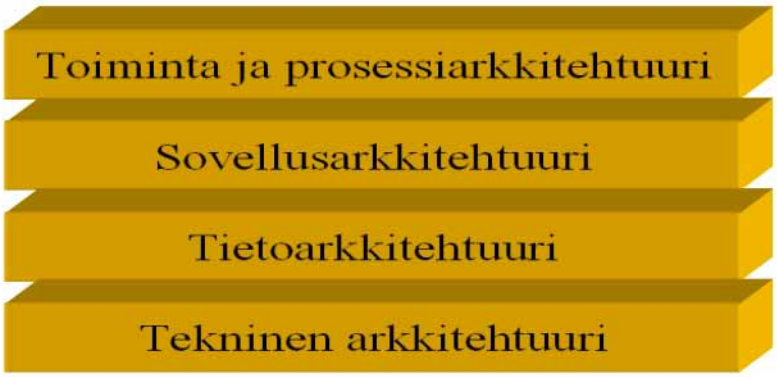
Muita menetelmiä

Mallinnus kuvauksineen on tärkein tietoarkkitehtuurin menetelmä. Lisäksi on hyvä laatia tietovirtakaavio, joka saattaa olla jo tehtykin, mikäli on laadittu tietojärjestelmä- tai sovellusarkkitehtuuri. Tietovirtakaavio tukee erittäin hyvin tietomalleja.

Lisäksi voidaan tehdä tietojen elinkaarikuvauksia sekä käsite/prosessimatriisikaavioita.

Yritysarkkitehtuuri

Tietoarkkitehtuuri liittyy ylempään arkkitehtuurikäsitteeseen, yritysarkkitehtuuriin (Enterprise Architecture). Joskus käytetään myös termiä Informaatioarkkiehtuuri, mutta tällöin rajautdutaan usein web-palvelinten ja selainmaailman tietoihin. Muita yritysarkkitehtuurin osia ovat tietojärjestelmä- eli sovellusarkkitehtuuri, laitearkkitehtuuri ja prosessiarkkitehtuuri, ks kuva 3. Kaikissa arkkitehtuureissa on sama ajatus: tarjotaan valmiita ratkaisuja käytettäväksi, jotta ei

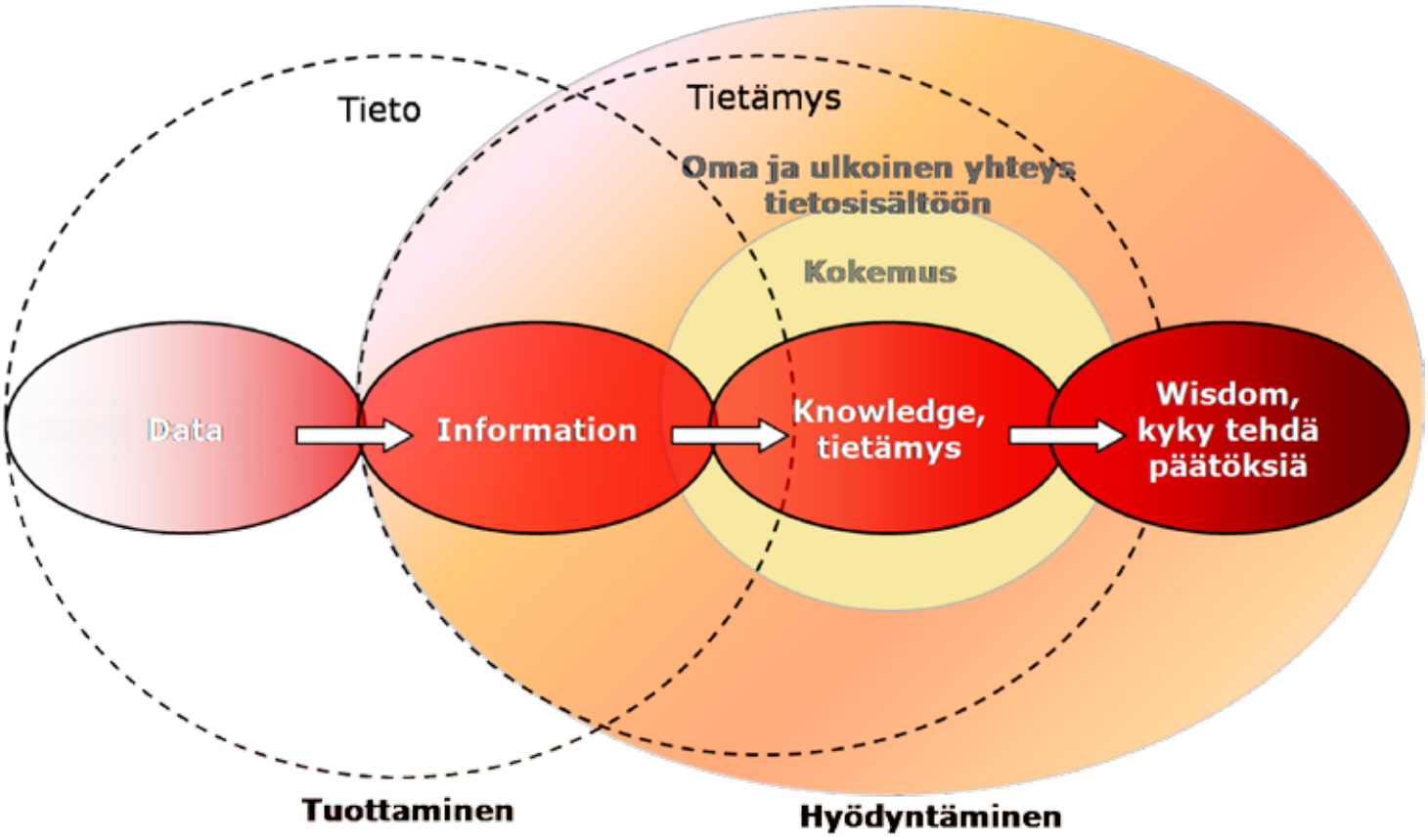


joka osastolla tarvitsisi pohtia samoja ratkaisuja. Tavoitteena on, ettei kaikkien yksiköiden tarvitse ”rakentaa omia voimaloitaan ja viemäröintejään”.

Arkkitehtuurin tehtävänä on auttaa hankkeita, ettei tarvitsisi käyttää aikaa ”turhaan” luovuuteen ja päästään tekemään oikeasti tärkeitä ja hyödyllisiä asioita. Viime kädessä arkkitehtiryhmät yrityksissä ja julkishallinnon organisaatioissa tuottavat kulusäästöjä. Tosin aina on menossa jonkinlainen haitariliike: arkkitehti- ja muita tukiporukoita lisätään kulujen säästämiseksi ja sitten jossakin vaiheessa joku on sitä mieltä, että nuo pätevät tyypit pitää laittaa ns. oikeisiin projekteihin - kunnes taas huomataan että tarvitaan arkkitehti- ja tukihenkilöitä…

Tietoarkkitehtuurin tai yritysarkkitehtuurin onnistuminen riippuu organisaation kypsyydestä ymmärtää tietoresurssin arvo. Jos talletettuja tietoja osataan arvostaa, alkaa tietojen arvostamisen kulttuuri kehittyä ja sen seurauksena tieto- ja yritysarkkitehtuurit saavat arvonsa mukaisia toteutuksia.

Seuraavan sivun artikkeliin liittyvä kuva:
Lähtötiedon jalostaminen viisaudeksi.



Teksti: Minna Oksanen

Tietojärjestelmän laadun ratkaisevat tietosisällön oikeellisuus ja merkitys

Kirjoittaja toimii BI/DW asiantuntijana Digiassa, on DAMA Finlandin perustajajäsen ja Systeemityölehden päätoimittaja.

Tietojärjestelmän laadun ratkaisee tiedon oikeellisuuden lisäksi sen merkitys. Mikä mielenkiintoisinta, tiedon merkitys vaihtelee asiayhteyden mukaan. Tiedon oikeellisuuteen on kiinnitettävä huomiota jo tietojärjestelmän pohjatöitä tehdessä, koska tiedon merkitys näkyy vasta tietoa hyödynnettäessä.

Ovatko 0 ja tyhjä sama asia?

Kysymykseen voi vastata sekä kyllä että ei. Sillä on merkitystä riippuen asiayhteydestä. Esimerkiksi onko kurssilla ollut osallistujia vai onko kurssia pidetty lainkaan. Jos molempiin kysymyksiin vastataan arvolla 0, ei voida tietää varmuudella, kumpi tilanne on ollut. Tietotarpeelle löytyy merkitystä, ainakin silloin, kun on tarve tietää kuuluuko kurssi oppilaitoksen valikoimiin.

Merkitys ei ole ainoa tiedon sisällölliseen laatuun liittyvä kriteeri. Muut tärkeät kriteerit ovat oikeellisuus, yhdenmukaisuus, paikkaansapitävyys ja kattavuus. Nämä kaikki on huomioitava, kun tietoa syötetään. Myöhemmin virheellisten tietojen korjaaminen on sekä hankalaa että kallista.

Tiedolla on myös elinkaari, sillä sen merkitys on sidottu käytettävyyteen paikan ja ajan suhteen. Haastava esimerkki on kuntauudistus. Miten Salon tietoja vuodelta 2008 voidaan verrata vuoden 2009 tietoihin? Toisaalta, onko tarvettakaan verrata. Tähän vastaus löytyy järjestelmän käyttötarkoituksesta.

Tiedon laatu vaatii organisaatiolta sitoutumista

Laadukas tieto ei synny tietojärjestelmien kautta. Niitä käytetään toki prosessin tukena ja esimerkiksi tietojen integraatiossa niille voidaan tehdä automaattisia tarkistuksia. Kaikkein tärkeintä on kuitenkin ihmisten tekemä työ, erityisesti tietojen syöttäminen järjestelmään. Vaatimuksia määriteltäessä tiedon tarkistusprosessin tulee olla määriteltynä. Tietoarkkitehtuuria määritettäessä on huomioitava tiedon laadun hallinta. Jo määrityksissä kerrotaan sallitut arvot ja tarvittavat tarkistusrutiinit.

Aina tiedon laadun merkitys ei näy vielä operatiivisessa toiminnassa voisi olla seuraava kuulemani: Palvelu oli tilattu vuoden 2008 viimeiselle päivälle joulukuun puolivälissä. Kun palveluntilaaja tammikuussa tarkistaa, miksei ole saanut palvelua, niin järjestelmään olikin syötetty vuosiluvuksi 2009. Järjestelmä ei tähän reagoinut, mutta asiakas olisi joutunut odottamaan kokonaisen vuoden haluamaansa palvelua.

Lähtötiedon jalostaminen viisaudeksi

Suomeksi tieto on yleiskäsite, joka tarkoittaa sekä tietoalkioita ja jalostettua tietoa toisin kuin esim. englanniksi, jossa data tarkoittaa lähtötietoa, syöttötietoa ja information jalostettua tietoa.

Tiedon jalostusprosessin havainnollistamiseksi voidaan käyttää esimerkkinä oheista kenkäparia. Kävellessäni kadulla näen näyteikkunassa kaksi kenkää ja vieressä lapun, jossa on teksti 50 EUR. Tämä ei vielä kerro minulle mitään. Tiedot yhdistämällä tiedän, että kengät maksavat 50 euroa. Tässä vaiheessa tieto on jalostunut informaatioksi. Jalostusprosessina käytetään yleensä tietovarastopohjaista raportointia ja tilastollisia analyysejä. Tiedonjalostusprosessi täydentyy, kun yhdistetään saatuihin tuloksiin tulkinta. Aikaisemmat kokemukset ja tietosisällön tunteminen vaikuttavat ymmärrykseen tietosisällöstä.Siksi kenkäesimerkissä tiedän, onko kengät kalliit vai halvat, kun ne maksavat 50 euroa.

Juhla-avokkaista voin maksaa enemmän kuin peruskumisaappaista, jolloin kyseiset kengät olisivat edullinen ostos. Jokaisella meistä on lisäksi eri käsitys tästä. Näin voidaan todeta, että tiedolla on subjektiivinen merkitys sen käyttötarkoituksen mukaan eli on syntynyt tietämystä. Tietämystä tarvitaan, jotta saadaan tehtyä viisaita päätöksiä. Kenkäesimerkissä päätöksen tekoon vaikuttaa ei vain hinta, mutta myös se tarvitsenko kengät. Jos korko olisi katkennut edellisessä kadun kulmassa, tarve olisi akuutti ja kävelisin kauppaan ostamaan kengät.

Toinen havainnollistava esimerkki on, että tiedonjalostusprosessissa hiotaan harmaasta hiilestä timantti. Timantista tulee sitä kirkkaampi, mitä parempi lähtötiedon laatu on.

"Information quality is no longer an optional characteristic of information systems. Information quality is a requirement for effective business performance."

"It has been demonstrated that poor quality data can cause business losses in excess of 20% of revenue and can cause business failure."

(Larry English, Information Impact)

Teksti: Tarja Raussi
Pentti Virtanen

Vaatimusmäärittelyä ketterästi vai perinteisesti?

Tarja Raussi on pitkän linjan systeemityöammattilainen, joka kouluttaa ja konsultoi Tieturi Oy:ssä mm. prosessien mallintamista ja vaatimusten määrittelyä ja hallintaa. Vaikka UML käyttötapauksineen onkin Tarjan toinen kotimainen kieli, hänen mielestään vaatimusmäärittelyn tavan valintaan vaikuttaa kulloinenkin tilanne.

Pentti Virtanen on toiminut ohjelmistokehittäjänä, projektipäällikkönä ja ohjelmistotuotannon prosessien kehittäjänä vuodesta 1981. Hän on tietojenkäsittelytieteen filosofian tohtori ja sertifioitu Scrum Practitioner, joka kouluttaa ja konsultoi Tieturi Oy:ssä mm. ketteriä menetelmiä, ohjelmistoarkkitehtuureja ja tietoturvaa.

Onko ketterän ja perinteisen vaatimusmäärittelyn välillä oikeasti eroja? Eivätkö molemmat tähtää laadukkaaseen lopputulokseen? Vai voiko kummallakin tavalla saada sutta aikaiseksi?

Vaatimusmäärittelyn merkitys projektin onnistumiselle on tunnustettu Standish Groupin tutkimuksissa jo yli 10 vuoden ajalta. Koko ajan tärkeimpänä kriteerinä on ollut käyttäjien osallistuminen - se on tuoreimmassakin tutkimuksessa numero 1. Samoin vaatimusten selkeyttä ja muutosten hallintaa on painotettu. Näitä ei kukaan kiistäne.

Vaan miten vaatimusmäärittelyä pitäisi tehdä? Pitäisikö kaikista toiminnallista vaatimuksista tehdä käyttötapaukset? Riittäisikö Scrumin tyyppinen tuotoslista? Onko siinä riittävästi informaatiota? Keskustelua käyvät perinteistä vaatimusmäärittelyä kannattava Hannele ja ketteryyttä suosiva Pena.

Dokumentoinnista

Hannele: Kunnon kuvaukset olla pitää!

Vasta jos vaatimukset on kunnolla kuvattu proosana tai käyttötapausten avulla, voi toteutuksesta tulla jotakin järkevää. Eihän koodari voi mitenkään tajuta jostain yksittäisistä lauseista tai parin rivin kuvauksista mitä hänen pitäisi tehdä. Ja mahtaisiko silloin projektipäälliköllekään olla kokonaiskuvaa järjestelmän laajuudesta ja monimutkaisuudesta?

Pena: Tuotoslistaan kirjoitetaan tosiaankin vain käyttäjätarina yhdellä virkkeellä mutta se ole kaikki. Ketterä ohjelmistotuotanto perustuu jatkuvaan vuorovaikutukseen asiakkaan kanssa. Määrittelyä tekevä asiakas on mukana työssä koko sen ajan tarkentaen ja ohjaten sitä ohjelmiston kehittyessä. Suullinen, vuorovaikutteinen kommunikaatio vähentää väärinymmärryksiä verrattuna perinteiseen kirjalliseen vaatimusmäärittelyyn.

Määrittelyä tarkennetaan myös luomalla testitapaukset hetikohta työn alkaessa. Asiakkaalla on tässä usein apuna ammattitestaaja, joka osaa täsmentää vaatimukset yksiselitteisillä testitapauksilla.

Ketterässä kehityksessä projekti ei viivästy odottaessaan kokonaisvaltaisen vaatimusmäärittelydokumentin valmistumista, johon voi pahimmassa tapauksessa mennä useita vuosia. Dilbert ei saanut alkua pitemmälle kolmessa vuodessakaan

Hannele: ”Asiakas on mukana ohjaten...” Hmm... Silloin se vaatisi aika monta henkilöä, kun käytännössä isojen järjestelmien parissa yksi käyttäjä tekee vain tiettyjä hommia. Eikä välttämättä kellään ole selkeää kokonaiskuvaa kaikista osa-alueista. Tämä viimeksi mainittu tuntuu tulevan jatkossa yhä enemmän esille, sillä kun jatkuvasti palkataan vain määräaikaisia, niin eihän kenellekään ennätä kehittyä mitään kokonaiskuvaa. Ja varsinkaan silloin, jos projektia tehdään ketterästi eli ei dokumentoida ollenkaan. Dokumentista voisi katsoa, miten asiat liittyvät toisiinsa.

Osaako käyttäjä kertoa toteuttajalle tarvittavat säännöt riittävän yksiselitteisesti? Kehittäjä kun usein tulee jostain it-talosta eikä ole koskaan edes kuullut moisesta sovellusalueesta.

Itse olen törmännyt juuri tässä kohdin ongelmaan: vaikka kuinka yritetään laittaa sääntöjä selkeästi kuvauksiin, niin kuitenkin aukkoja jää aina.

Pena: Ketteryyden rinnastaminen dokumentoimattomuuteen on varsin yleinen väärinkäsitys. Ketterät projektit dokumentoivat, mutta tekevät sen eri tavalla. Lähdekoodi on ohjelmoijalle edelleenkin tärkein tiedon lähde. Sen laatuun kiinnitetään paljon huomiota. Käyttäjille pitää edelleen tehdä käyttöohjeet. Lisäksi ketterässä työtavassa syntyvät testitapaukset, jotka esimerkkien avulla kertovat kuinka ohjelmiston pitäisi toimia. Testitapaukset säilytetään samassa paikassa lähdekoodien versionhallinnassa kanssa, jolloin ne myös löytyvät - mitä ei voi aina sanoa perinteisistä dokumenteista.

Työn pilkkominen pieniin merkityksettömiin sirpaleisiin on eräs perinteisen työtavan ongelmista, joihin ketteryyden serkku, laiha ohjelmistotuotanto (Lean Software Development) antaa lääkkeitä. Suuri joukko ihmisiä sähläämässä kymmenien projektien kanssa yhtä aikaa ei voi olla menestyksellinen tapa tehdä töitä. Ketteryyden käyttöönoton tarkoituksenakin on tuoda organisaation ongelmat näkyville. Kertomasi on yksi korjausta kaipaavista ongelmista.

Mutta palatakseni vaatimusmäärittelyyn, mikä onkaan ison vaatimusspeksin arvo dokumenttina olettaen ensin, että löydät oikean version siitä. Todennäköisesti se on vanhentunut ja ohjelmiston toteutus on aivan toisenlainen.

Hannele: Niin, viittasin kyllä omassa kommentissani siihen oikeaan liiketoimintatyöhön, en projektiin. Harvempi ihminen tekee työssään hienosti prosessia päästä päähän. Pahimmillaan se oma työ voi olla yksittäisiä työtehtäviä, jotka sijoittuvat ihan eri prosesseihin. Ja silloin on kyllä vaikea nähdä omaa osuuttaan prosessissa.

Joo, tuo mitä sanoit dokumenttien versioista, pitää kyllä paikkansa - valitettavasti. Liian usein käy niin, että vaatimusmäärittelyn dokumentti tehdään ja sitten se unohdetaan pölyttymään hyllyyn. Vaatimusmäärittelykuvaustenhan pitäisi olla aina ajan tasalla, sillä ne kertovat, mitä liiketoiminta haluaa järjestelmän tekevän.

Silloinhan on helppoa ottaa aina viimeisin versio. Sillä ihan samalla tavalla kuin kerroit testitapausten menevän ketterässä koodin mukana versionhallintaan, niin vaatimusdokumentaatiokin voidaan laittaa versionhallintaan. Eli samassa paketissa on määrittelyt ja niitä vastaava koodi + muut tarvittava aineisto.

Vaatimusmäärittelykuvauksia voidaan käyttää myös uusien ihmisten perehdyttämiseen ko. järjestelmään. Näkyyhän esim. käyttötapauskaaviosta suoraan, kuka saa käyttää mitäkin toiminnallisuutta.

Pena: Vaatimusmäärittelydokumenttien käyttö perehdyttämisessä järjestelmään perustuu olettamukseen, että vaatimusdokumentit ja lopullinen järjestelmä ovat yhdenmukaiset. Tämä on harvoin totta, sillä kehitettäessä uutta tuotetta tai ylipäätään mitä tahansa ohjelmistoa, on luonnollista ja hyödyllistä, että hyvät ideat otetaan käyttöön ja ristiriitaisuudet korjataan. Tällöin vaatimusdokumenttien ylläpidosta tulisi kokopäivätyö. Reaalimaailmassa tämä jätetään tekemättä.

Ohjelmistoon perehdyttämisessä kannattaa käyttää käyttäjien osalta käyttö-ohjeita, jotka tietysti tehdään myös ketterissä projekteissa ja kehittäjien osalta lähdekoodia ja niitä tukevia malleja. Etenkin Extreme Programming kiinnittää valtavasti huomiota koodin luettavuuteen ja ymmärrettävyyteen. Mallitkaan eivät ole pannassa ketterässä ohjelmistotuotannossa. Ne eivät kuitenkaan ole itsetarkoitus vaan kiinteä osa kehitystyötä.

Liiketoimintaprosessien kuvauksia käytetään ihmisten työn ohjaamiseen myös ketterässä ohjelmistotuotannossa. Erona on se, että ne syntyvät projektin aikana ei kauan sitä ennen. Koska myös liiketoimintaprosessit muuttuvat sitä mukaan kun opitaan uutta, kannattaa niiden sisältö jäädyttää niin myöhään kuin mahdollista.

Hannele: Käyttäjien perehdyttämiseen tietysti käytetään käyttöohjetta. Ja koulutusmateriaaleja.

Mutta ajattelinkin lähinnä sovellusohjelman ylläpitäjiä. Vie tolkuttomasti aikaa, jos koodaaja lähtee etsimään koodista, mihin kohtaan muutokset tulevat. Ja siks toiseks, kun muutostyötä tilataan ylläpidossa, vaatimukset pitää kuitenkin kuvata. Onhan silloin helpompi, jos voi viitata olemassa olevaan kuvaukseen ja kertoa, mikä toiminnallisuudessa muuttuu. Mutta kuten sanoit, valitettavasti reaalimaailmassa tuo dokumenttien päivitys kyllä tahtoo unohtua, jos sitä ei ole otettu normaaliksi työtavaksi.

Eivät kai liiketoimintaprosessit sen takia muutu, että projektissa opitaan uutta???!!! Prosessi on prosessi ja se muuttuu vasta, kun prosessia kehitetään tai liiketoiminta muuttuu.

Artikkelin alussa viitataan Standish Groupin CHAOS 2004 –tutkimukseen, josta osia on julkaistu netissä Standish Groupin Jim Johnsonin haastattelussa 25.8.2006 http://www.infoq.com/articles/Interview-Johnson-Standish-CHAOS

Ketterästi vai perinteisesti?

Tarkan tason työnkulku voi kyllä hioutua projektin aikana - varsinkin jos on kyse valmisohjelmiston käyttöönotosta. Liian usein on käynyt niin, että kun kuvittelimme saavamme säästöjä ja hyötyä valmisohjelmistosta, niin sen toimintatapa onkin ihan erilainen ja jopa hankala. Ja sitten käyttäjät manailevat, kun ohjelman käyttö on hankalaa.

Luettavuus ja ymmärrettävyys

Pena: Koska ohjelmakoodi on todellakin tarkoitettu ohjelmoijien luettavaksi, kiinnitetään sen luettavuuteen ja laatuun erityistä huomiota. Olen olemassa koodausstandardit, jota tiimin jäsenet valvovat tekemällä töitä pareina ja antamalla kaikkien tiimin jäsenten tehdä työtä koko koodin kanssa. Pariohjelmointi on erinomainen tapa tehdä koodikatselmointia.

Isossa ohjelmistossa tarvitaan tietenkin myös malleja ja pakkausrakenne, joka ohjaa seuraavaa tehtävää tekevän ohjelmoijan suoraan muutosta tarvitsevaan osaan ohjelmistossa. Malleja ja ohjelmiston rakennetta ei kuitenkaan jäädytetä liian aikaisin vaan sitä mukaa kun ohjelmisto kehittyy. Mallien tekemisen pitää palvella ohjelmiston rakentamista, eikä olla oma tarpeettomia paperidokumentteja tuottava norsunluutorni.

Hannele: Joo, olen minäkin urani aikana nähnyt sellaisia tarpeettomia dokumentteja, mutta myös tarpeellisia.

Olen kokenut hyväksi toiminnallisuuden kuvauksessa käyttää käyttötapauksia - ne todella toimivat. Saadaan selkeästi kuvattua, mitä käyttäjä tekee ja mitä järjestelmä tekee. Ja lisäksi tulee kuvattua, mitä järjestelmän sisällä pitää tapahtua, eli käsittelysäännöt ja virhetarkistukset.

Joskus olen kyllä huomannut, että käyttäjät kuvaavat tuonne helposti myös manuaalisia juttuja eli kuvaavat myös sellaista mitä ei ole tarkoitus toteuttaa järjestelmään.

Onkos tähän ketterällä puolella mitään ratkaisua? Vai mennäänkö vain suusanallisen pohjalta tekemään mikä-se-olikaan tuotoslistaa?

Pena: Tarkoitat varmaan käyttäjätarinoita (user story)? Ne ovat yksinkertaisia virkkeitä, joiden avulla kerrotaan, mitä ohjelmiston pitää sisältää. Ne kirjoitetaan tuotoslistalle muistilapuksi ja niitä täydennetään suusanallisesti, mutta muistiinpanojen tekemistä ei tietenkään ole missään kielletty. Mm. digikuvat valkotauluille piirretyistä käyttöliittymäluonnoksista ovat suosittuja.

Scrum on hyvin yksinkertainen kehys, joka ei sisällä ohjeita siitä, miten insinöörikäytännöt pitää hoitaa. Käyttäjien manuaalisen toiminnan kuvaaminen on asia, jonka tekotavan projekti päättää. Joissain organisaatioissa käytetään kirjallisia toimenkuvia, joissain taas ei.

Ei-toiminnalliset vaatimukset ja arkkitehtuuri

Hannele: Hankalampi juttu on nuo suorituskyky- yms. teknisemmät vaatimukset. Niitä on kyllä aika vaikea kuvata. Ja olen huomannut, etteivät toimittajat tahdo sitoutua niihin mitenkään.

Pena: Arkkitehtuuri on tärkeässä osassa ketterässä ohjelmistotuotannossa. Se ei kuitenkaan ole norsunluutorneissa tehty PowerPoint-arkkitehtuuri vaan konkreettisesti yhteydessä ohjelmiston tekemiseen. Ennen kuin tuotoslistaa (product backlog) pääsee kunnolla tekemään, tarvitaan arkkitehtuurivisio: näkemys siitä mitä kaiken kaikkiaan ollaan tekemässä. Se voi olla yksi A3, joka kertoo mistä kokonaisuudesta on kysymys ja miten se liittyy ympäröiviin järjestelmiin.

PowerPoint-arkkitehtuurin vastakohta on suoritettava arkkitehtuuri, joka yksinkertaisimmillaan sisältää ohjelmointikielellä tehdyt rajapintamääritykset, integraatiotestit ja stubit, jotka toteuttavat integraatiotestit. Jatkuvasti ajettavat integraatiotestit varmistavat kehitystyön yhden- ja arkkitehtuurin mukaisuuden työn edetessä.

Ei-toiminnallisten vaatimusten testaaminen tapahtuu luonnollisena osana ensimmäisiä työjaksoja (Sprint). Scrumissa pääosa ensimmäisten työjaksojen työstä menee arkkitehtuuriin ja infrastruktuuriin. Koska mukana on kuitenkin aina myös liiketoimintaa hyödyntävää toiminnallisuutta, tulee arkkitehtuuri testattua jo ensimmäisten työjaksojen aikana.

Hannele: Arkkitehtuuri on sitten arkkitehtien asia.

Vaatimusmäärittelyssä me keskitytään siihen, millaista toiminnallisuutta ja tietosisältöä haluamme järjestelmään. Tosin tuleehan siellä esille myös niitä nopeita vasteaikoja ja ”pitää noudattaa java-arkkitehtuuria” yms. asioita. Minusta siinä onkin ristiriita, että yleensä opetetaan, että arkkitehtuurin voi luoda vasta kun vaatimukset on kuvattu, mutta eihän uusi järjestelmä tule tyhjiöön. Siellähän on jo muut järjestelmät ja ehkä jopa sovittu välineistö mitä pitää käyttää. Lisäksi useimmat projektit ovat ylläpitoprojekteja - olemassa olevaa sovellusta kehitetään, jolloin arkkitehtuuri on jo olemassa. MItenkäs tuo mainitsemasi Scrum tähän suhtautuu?

PowerPoint on ihan hyvä väline, sillä se löytyy useimmilta! Jos arkkitehtuuria ei ole kuvattu mitenkään, niin pitääkös taas etsiskellä koodia, ennen kuin tietää mitä pitää korjata. Meillä ainakin arkkitehdit antavat myös toteutusohjeita mm. miten koodi jaetaan moduleihin jne. Usein arkkitehti kulkeekin toteutuksen projektipäällikkö -nimikkeellä. Koodarit tarvitsevat ohjausta, muutenhan sovellus on yksi tilkkutäkki, jos jokainen tekee tyylillään.

Pena: Vaatimusten kirjoittaminen kokonaan irrallaan arkkitehtuurista on haasteellinen tehtävä, sillä nämä riippuvat voimakkaasti toisistaan. Esimerkiksi asiakas saattaa vaatia miljoonaa yhtäaikaista käyttäjää alle sekunnin vasteajalla kunnes kuulee kuinka kalliiksi tällaisen toteuttaminen tulee. Helposti tulee myös vaadittua asioita, joiden toteuttaminen on kohtuuttoman hankalaa. Pienellä muutoksella vaatimuksiin voidaan saada suuria parannuksia arkkitehtuuriin ja siten myös toteutukseen.

Scrum soveltuu luonnostaan ylläpitoprojekteihin. Muutostarpeet kirjataan tuotoslistaan, priorisoidaan ja toimitetaan kuukausittaisissa toimituserissä. Olemassa olevaa arkkitehtuuria voidaan tietysti käyttää myös Scrumia käyttävässä ylläpitoprojektissa.

PowerPoint on tarkoitettu diaesitysten tekoon. Ohjelmistoarkkitehtuurien kuvaamiseen käytetään yleensä UML-mallinnusnotaatiota, jolle on olemassa hyvä, muihin ohjelmistotuotannon työkaluihin integroitu työvälinetuki. Loogisen arkkitehtuurin määritteleminen kokonaisvaltaisesti etukäteen on käytännössä mahdotonta aivan samalla tavalla kuin toimistokäyttäjälläkin kansiorakenteen ennakointi. Niinpä liikkeelle lähdetään arkkitehtuurin visiolla, jota sitten täydennetään työn edetessä. Kuten huomasitkin, arkkitehtina toimiva tekninen projektipäällikkö on mukana koko projektin ajan ja pystyy ohjaamaan työtä sekä suullisesti, että ohjelmiston rakenteita kehittäen.

Sopiiko ketteryys kaikkeen?

Hannele: Eihän tuo Scrum ihan mahdottomalta kuulostakaan, jos sitä kerran tuolla tavoin hallitusti tehdään...

Itse kun olen törmännyt vain sellaisiin ketteryyden nimiin vannoviin, jotka sanovat ettei mitään tarvitse dokumentoida tai arkkitehtuuriakaan luoda, vaan tehdään vain. Tuli eittämättä mieleen nuo 80-luvun RAD-jutut, jotka olivat enemmän ”hutaisten tehty” -juttuja.

Mutta taitaa olla niin, ettei tuo ketteryyskään joka paikkaan sovi?

Pena: Tilanneriippuvaistahan tämä on. Esimerkiksi pienissä ylläpitotöissä työn ennustettavuus on aika hyvä, sillä tekninen ympäristö, tekijät ja tilaajat ovat kaikki samoja kuin ennenkin. Silloin on mahdollista kirjoittaa vaatimukset dokumenttina ja saada aivan kelvollinen muutos ohjelmistoon. Ketteryys taas on parhaimmillaan silloin, kun ei tiedetä mitä tarvitsee tehdä puhumattakaan siitä miten se pitäisi tehdä.

Jälkikirjoitus:

Tämä artikkelin pohja luotiin ketterästi Tieturin web-sivuston keskustelupalstalla. Keskustelua isännöi Tarja omalla nimellään. Mutta sitten hän toi keskusteluun perinteisen näkökulman Hannelen roolissa, kun taas Pentti otti ketteryyden puolestapuhujan roolin nimettömänä keskustelussa. Roolinimiä käytettiin siksi, jotta keskustelu ei leimautuisi sisäpiirijutuksi. Paras työskentelytapa vaatimusmäärittelyyn on riippuvainen projektista ja olosuhteista. Ei siis ole olemassa vain yhtä oikeaa tapaa tehdä vaatimusmäärittelyä.

Lue koko keskustelu http://www.tieturi.fi/web/guest/keskustelut ja sieltä keskustelualue Teknologiat ja menetelmät. Kommentit, vastaväitteet ja täydennykset ovat luonnollisesti tervetulleita.

Molempi parempi! Eli valitaan tapa tarpeen ja tilanteen mukaan.

Teksti: Matti Vuori

Vaatimusmäärittelyn huonoimmat käytännöt

Kirjoittaja toimii konsultointipalvelujen tuotepäällikkönä Cybercom Finlandissa ja työssään konsultoi ja kouluttaa organisaatioita monissa mm. laadunvarmistukseen, testaukseen, käytettävyyteen ja riskienhallintaan liittyvissä asioissa. Vaatimusmäärittelyhistoriaa hänellä on monissa merkeissä, tulevaisuuden tuotteiden kehittämismenetelmien kehittelystä vaatimusmäärittelykoulutukseen ja määrittelyjen analysointiin palveluna.

Monien asioiden yhteydessä puhutaan parhaista käytännöistä ja niitä pyritään toteuttamaan käytännön toiminnassa. Tämä pätee myös vaatimusmäärittelyyn. Monien organisaatioiden tilanne on kuitenkin se, että ensin pitäisi pystyä välttämään huonoimpia käytäntöjä, perinteisiä peruspatologioita, ja vasta sen jälkeen miettiä toiminnan kehittämistä eteenpäin ja vaiheittaista optimointia. Tämä artikkeli esittää lukijoiden pohdittavaksi joitakin vaatimusmäärittelyn huonoimpia käytäntöjä, joihin edelleen törmätään liian usein.

Unohtuu hyvä prosessi, tarkastellaan vain dokumenttia

Vaatimusmäärittelyllä on monta olemusta: Vaatimukset kuvaava dokumentti, vaatimusten syntymiseen johtava prosessi, yhteinen ymmärrys kehittämisen kohteesta sekä yhteinen sopimus vaatimuksista. Usein arvioidaan vain dokumenttia, mutta tärkeämpää voi olla prosessi, joka tuottaa yhteistä ymmärrystä, oppimista ja sitoutumista.

Vaatimusmäärittelyjä ei katselmoida eri osapuolten kanssa

Olipa projektin asetelma millainen tahansa, vaatimusmäärittelyn kunnollinen katselmointi on elintärkeää. Jokaisessa määrittelyssä on suuri joukko virheitä, epäselvyyksiä, monikäsitteisyyttä jne. Niiden yli päästään ymmärrykseen vain istumalla alas ja käymällä määrittelyt kunnolla läpi – ja sitten parantamalla niitä.

Puuttuu ajatus konseptista

Jos järjestelmän ideaa, ydintä, muista erottavia piirteitä, tarkoitusta, käyttäjä-ajatusta ja käyttötilanteita ei ole jäsennetty, kehitetään asiaa, jota kukaan ei ymmärrä tai tulkitsee omista lähtökohdistaan. Nykyään tämä tilanne on varsin yleinen: aina ei tiedetä, luodaanko toiminnanohjausjärjestelmää, projektinhallintaohjelmistoa, hajautettua laskutusapua tai jotain aivan muuta. Ja jokainen projektin osapuoli näkee asian eri tavalla kuin toiset.

Liian suuri skouppi

Perinteinen synti on se, että määritetään liian suuria järjestelmiä, joita ei enää ymmärretä ja joiden kehittelylle tulee koon vuoksi vähintään psykologisia esteitä. Skoupin rajoittaminen ja hallinta ja vaiheittainen kokonaisuuksien kehittäminen ovat keskeisiä menestystekijöitä.

Asiakkaan vaatimuksista ei tuoteta oikeaa vaatimusmäärittelyä

Asiakkaan kertomat vaatimukset ovat harvoin riittävällä tasolla järjestelmän kehittämiseksi. Niitä pitää arvioida, täydentää, laajentaa, ottaa mukaan uusia näkökulmia. Vasta sitten voidaan puhua perustellusta vaatimusmäärittelystä. Tämä on tehtävä jollakin tasolla, vaikka asiakas ei haluaisikaan maksaa vaatimusmäärittelyn tekemisestä.

Käyttäjien tarpeita ei selvitetä

Edelleen voidaan jättää käyttäjätutkimukset tekemättä. Tällöin ei ole kunnollista kuvaa siitä, millaisille ihmisille, mihin tarpeisiin, millaiseen arkeen järjestelmää ollaan kehittämässä. Tulok-

sena on järjestelmä, joka ei riittävästi tue käyttäjien tarpeita eikä sovellu saumattomasti heidän arkeensa.

Ei-toiminnallisia vaatimuksia ei määritetä

Vaikka nämä on nähty jo pitkään oleellisina, on usein edelleen se tilanne, että vaatimusmäärittelystä ei löydy tietoa käytettävyyden, suorituskyvyn, tietoturvallisuuden yms. vaatimuksille. Tällöin niitä ei oteta systemaattisesti huomioon, eikä niiden objektiivinen verifiointi testaamalla ole mahdollista.

Vaatimusten jäsentämisessä käytetään vain yhtä näkökulmaa

Vaatimusmäärittelyä ja arkkitehtuurisuunnittelua yhdistää tämä: kummassakin tarvitaan useita näkökulmia. Joskus jumiudutaan vain yhteen. Kaikki vaatimukset esimerkiksi nähdään käyttötapausten kautta. Näin ei kuitenkaan nähdä kaikkia vaatimuksia. Hyvä määrittelytyö katselee asioita eri vinkkeleistä paljastaen siten kohteen eri piirteitä, jättämättä mitään varjoon, silmien katvealueille.

Ei tehdä prototyyppejä

Ihminen ei ymmärrä ennen kuin näkee. Jos mihin, tämä pätee monimutkaisiin tietoteknisiin systeemeihin. Näkemisen lisäksi on hyvä päästä kokeilemaan asioita. Prototyypitys auttaa tässä. Niitä ei aina tehdä, koska kuvitellaan asiat selkeiksi ja koska prototyypitys tietysti aiheuttaa työtä ja kustannuksia. Mutta ilman prototyyppejä yhteinen ymmärrys kohteesta jää epämääräiseksi.

Vaatimustenhallintatyökalun vuoksi ei nähdä metsää puilta

Vaatimustenhallintatyökalut mahdollistavat valtavien hierarkioiden tuottamisen, mutta samalla tuottavat ongelmia kokonaisuuden näkemiselle, piilottavat konseptin ja vision näkymättömiin. Vaatimustenhallintatyökalut eivät yleensä ole parhaimmillaan vaatimusten luomisvaiheessa, mutta ovat arvokkaita vaatimusten hallinnassa ja muutoksenhallinnan apuna.

Vaatimusmäärittely ei kuvaa vaatimuksia, vaan edellytettäviä toteutusratkaisuja

Hyvä vaatimusmäärittely antaa suunnittelulle tilaa, ja luo lukkoon vain sellaisia reunaehtoja, jotka on pakko lukita, koska ne ovat olemassa ja niitä ei voida muuttaa (kuten konekanta).

Asioiden priorisointi jää puolitiehen

Priorisointi on keskeinen keino kehitettäessä elegantteja järjestelmiä. On tehtävä päätöksiä siitä, mitkä kohderyhmät ovat tärkeämpiä kuin jotkut muut ja mitkä käyttötarkoitukset ovat tärkeämpiä kuin jotkut muut. Epäkypsässä toiminnassa kaikki asiat koetaan yhtä tärkeiksi.

Vaatimusmäärittelyn tekevät amatöörit

Vaatimusmäärittely koetaan helposti liian helpoksi ja ajatellaan, että kuka tahansa voi kerätä vaatimukset eri ammattiryhmiltä ja niputtaa ne vaatimusmäärittelyksi. Oikeasti kyseessä on projektin kriittisin vaihe ja siihen tarvitaan osaavia ammattilaisia, jotka tietävät todellisuuden näennäisen helppouden takana, osaavat välttää sudenkuopat ja yhdistää eri osapuolten tarpeet harmoniseksi kokonaisuudeksi.

Jokin kohderyhmä varastaa määrittelyn

Esimerkiksi hallinto kehittää järjestelmää itselleen, vaikka ajatuksena olisi tukea liiketoiminnan arkea työtä. Tämä on normaalia organisaatiopsykologiaa, mitä pitää hallita osaavalla vaatimusmäärittelyn ja konseptoinnin prosessilla.

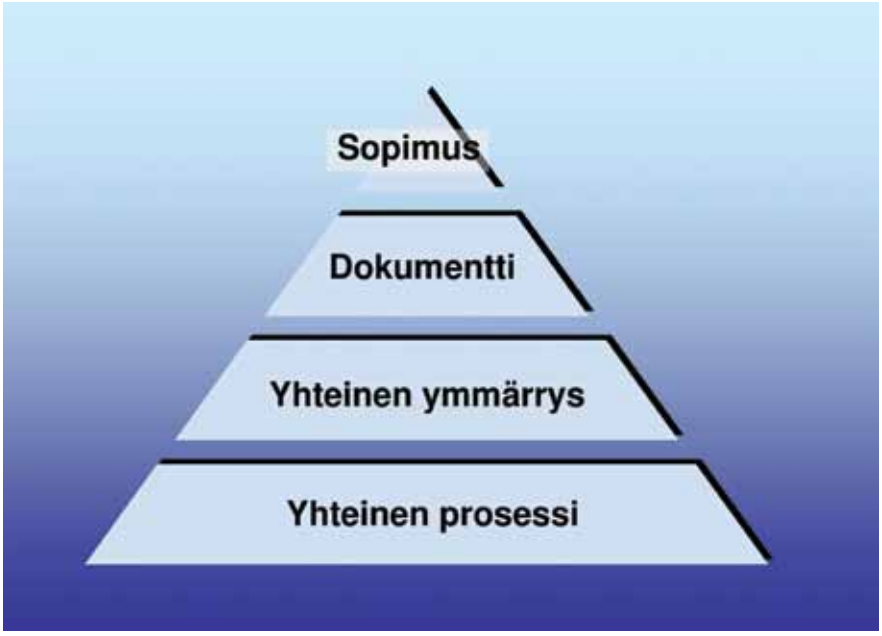
Taustalla ei riskianalyysejä

Vaatimusmäärittelyn pitää perustua sen ymmärtämiseen, mikä on tärkeää liiketoiminnan, prosessien ja käyttäjien näkökulmasta. Riskianalyysin pitäisi olla jokaisen projektin alkumetrien tehtäviä. Se auttaa tunnistamaan järjestelmän kriittisiä piirteitä, joiden vaatimuksiin ja suunnitteluun on panostettava erityisesti. Riskianalyysit ovat aina merkittäviä oppimiskokemuksia.

Lopuksi

On selvää, että huonoimmat käytännöt ja toiminnan kypsyys vaihtelee aloittain. Samoin edellä kuvattujen asioiden esiintyminen riippuu vaatimusmäärittelyn tekijästä – tilaaja, toimittaja vai kolmas osapuoli. Näitä asioita kannattaa kuitenkin aina pyrkiä tunnistamaan, sillä huonoimmille käytännöille on tyypillistä se, että yksikin niistä voi vaarantaa koko järjestelmähankkeen.

Vaatimusmäärittelyllä on useita oleellisia olemuksia.



Teksti: Kalle Huhtala

eLearning-sisältöjen määrittely ja konseptointi

Kirjoittaja on toiminut suunnitellut multimediakoulutussisältöjä ja -välineitä Metso Oy:ssa, ja toiminut eLearning-sisältötuotannon suunnittelu- ja johtotehtävissä vuodesta 2000 alkaen Talent Code Oy:ssä, Tieturi Onlinessa, Tieturi Visionissa sekä Tieturissa.

Kallen suurimpana mielenkiinnon kohteena on tiedon visualisointi, tarinankerronta ja vuorovaikutteisten, mediarikkaiden oppimissisältöjen suunnittelu aihealueesta riippumatta. Koulutukseltaan Kalle on diplomi-insinööri.

eLearning-sisältötuotanto on yksi digitaalisen viestinnän osa-alue. eLearning-sisällön lähtökohdat ja tuotantomenetelmät muistuttavat mainontaa, viestintää, tuotedokumentaatiota ja koulutusmateriaalia. Tässä artikkelissa esitellään visuaalisesti näyttävän, houkuttelevan ja vuorovaikutteisen eLearning-sisällön toteuttamiseksi tarvittavan määrittelydokumentin eli konseptin piirteitä.

eLearning-sisällöistä

Verkko-oppiminen eli eLearning on hitaasti vakiintunut yhdeksi oppimisen ja kouluttamisen välineeksi koulu- ja yritysmaailmassa. eBusineksen tavoin eLearning kärsi 2000-luvun alun e-hypestä liiallisine odotuksineen. Internetin kehitys ja yleistyminen koko kansan työ- ja huvivälineeksi on poistanut esteitä tietokoneavusteisen oppimisen tieltä. Hakukoneet, verkkovideot, pelit, chatit, verkkotietosanakirjat ja -yhteisöt ovat tuoneet ”vahingossa oppimisen” luontevaksi osaksi jokapäiväistä elämää.

Kouluissa ja yrityksissä eLearning tähtää jonkin tietyn aiheen tulokselliseen oppimiseen perinteisen luokkahuoneen ulkopuolella ja lisänä. Koulumaailman kollaboratiivisessa ja konstruktivistisessa oppimisessa keskeinen oppisisältö voi olla ohjattu keskustelu ja ryhmätyöt verkossa. Tuote- ja ohjelmistokoulutuksen välineinä tarvitaan kuitenkin yritys- ja tapauskohtaista eLearning-sisältöä. Tällaisen oppimistarkoitukseen tehdyn multimediamateriaalin määrittely ja tuotanto vaatii rahallista ja ajallista panostusta. Kannattava sisältötuotanto edellyttää tämän määrittelytyön kokemusta ja osaamista.

Lopputuloksena on 15-60 minuutin (itse)opiskelupaketti, joka jakautuu johdantoon, 3-5 ”lukuun”

Puolustusvoimille tehty ilmatorjunta-aselajia esittelevä laaja eLearning-kokonaisuus.

ja lopputestiin. Sisällössä pyritään välttämään tekstiä ja korvaamaan se kuvilla, animaatioilla, videolla ja mahdollisuuksien mukaan peleillä, testeillä ja tarinoilla. Käytettävyydeltään ja muodoltaan sisällön on pystyttävä kilpailemaan yleisen verkkosisällön, pelien jne kanssa. Toistuvasti haaveillaan oppisisällöstä, jota yritykset voisivat itse tuottaa. Haaveeksi hyvä sisältö kuitenkin jää, sillä sen tekemisessä vaaditaan asiantuntemuksen lisäksi tarinankertojan, kouluttajan, toimittajan, taiteilijan ja ohjelmoijan kykyjä - ja aikaa. Tämä osaaminen kannattaa yleensä ostaa ulkoa.

eLearning-konseptoinnin erityispiirteitä

Ohjelmistosuunnittelussa käytettävät dokumentit ovat hyvin yksityiskohtaisia ja kuvaavat tarkastikin lopputuloksen. Mainostoimiston briiffi antaa laajahkot reunaehdot ja taustatiedot kampanjan toteutukselle. eLearning-konsepti on jotain näiden väliltä. Se on työkalu, jonka ei ole edes tarkoitus kuvata valmista tuotetta yksityiskohtaisesti. Se kuitenkin listaa tarkkaan projektin lähtökohdat ja lopullisen oppisisällön osat ja toiminnallisuudet. eLearning-budjetit ovat tarvittavaan luovaan työhön, median käsittelyyn ja ohjelmointiin nähden rajallisia, joten hyvä konsepti on rahan arvoinen.

Konseptidokumentti elää tuotantoprosessin alkuvaiheessa oman aikansa ajatusten työkaluna ja peilinä. Projektin lopputuloksena syntyvä eLearning-sisältö tarkentuu vasta tuotantovaiheessa. Konseptisuunnitelmassa pyritään siis löytämään sisällön tärkeimmät asiat, valitsemaan niille sopivimmat esitysmuodot, miettimään oppimisen mittarit ja kartoittamaan tarvittava lähdemateriaali ja osaaminen. Toteutustiimiä varten konseptissa luodaan sisällön ”henki” ja tyyli kuvaamalla tyypillisiä loppukäyttäjiä ja käyttöympäristöä sekä esittämällä visuaalisia layout-esimerkkejä.

Konseptin pohjalta voidaan projektisuunnitelmassa jakaa käytettävissä olevat resurssit (aika, raha ja osaaminen) niin että sisältötuotannon toteutusprojekti on mahdollista tehdä kannattavasti.

eLearning-sisältö on usein yksisuuntaista viestintää, ei webbipalvelu, jonka sisältöön oppijat aktiivisesti vaikuttaisivat. Tästä syystä erityisiä use case -tyyppisiä kuvauksia konsepti ei sisällä paitsi mahdollisten simulaatioiden ja pelien kuvauksissa.

Kuka konseptin suunnittelee?

Konseptisuunnittelijan työpaikkailmoitus voisi kuulua vaikka näin: ”Hei, sinä korkeasti oppinut ja yleissivistynyt, kielitaitoinen tarinankertoja, joka hallitset toimisto-ohjelmat vasemmalla kädelläsi ja osaat poimia rusinat pullasta. Näet näkyjä ja saat muutkin tekemään niin. Tiedät, ettet luule tietäväsi kaikesta kaikkea. Tule meille konseptisuunnittelijaksi!”

Työ- ja elämänkokemus, hyvä koulutus, sosiaaliset ja viestintätaidot, tietoteknisten välineiden hallinta, mielenkiinto uusiin asioihin ja rohkeus heittäytyä tyhmäksi ovat siis hyvän konseptisuunnittelijan edellytyksiä. Konseptisuunnittelijan työpanos keskittyy projektin määrittelyvaiheeseen. Joskus konseptisuunnittelija jatkaa projektipäällikkönä ja/tai käsikirjoittajana. Saman tyyppisiä ominaisuuksia edellytetään mainostoimiston Art Directorilta tai copywriterilta.

Periaatteessa eLearning-konseptisuunnittelijalle kaikki aihealueet ovat yhtä uusia ja tuntemattomia. Konseptisuunnittelijan ammattitaitoa on auttaa asiantuntijoita löytämään olennaiset ja ulkopuolisen oppijan näkökulmasta tarpeelliset asiat yksityiskohtien suosta, sekä tarjota näihin oppimisen kannalta hyväksi havaittuja sisältömalleja (tarinat, pelit, sisältörakenteet jne).

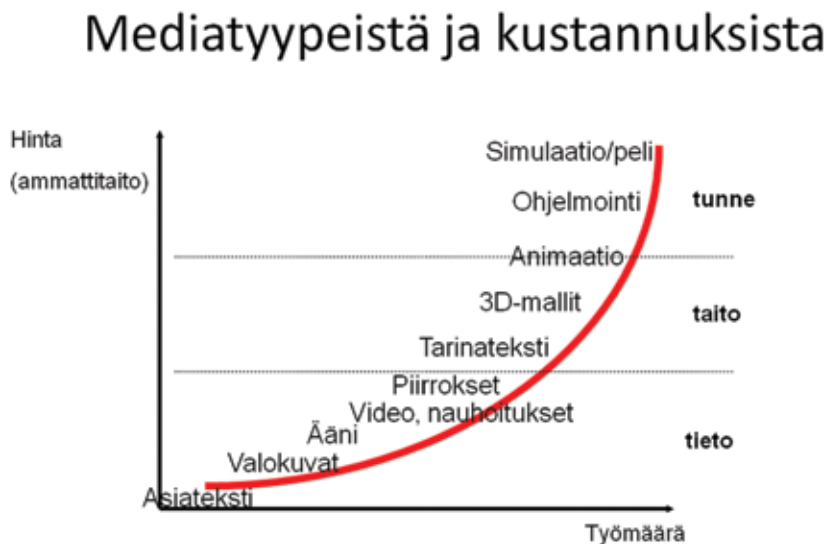
Määrittelypalaverit

Jos mahdollista, eLearning-sisältöprojekti kannattaa jakaa määrittely- ja tuotantovaiheeseen. Konseptisuunnitelma tehdään siis määrittelyvaiheessa, jonka työmääräarvio on 4-6 työpäivää. Perusteet konseptisuunnitelmalle saadaan kerättyä esim. kahdessa määrittelypalaverissa. Oleellista palavereissa on sitouttaa asiakas projektin sisältöön ja tuotantoon. Määrittelyvaiheen tuloksena on konseptisuunnitelma, layout-esimerkkejä ja projektisuunnitelma tuotantovaihetta varten.

Ensimmäisessä palaverissa tarkennetaan eLearning-projektin oppimis- ja muut tavoitteet, riskit ja haasteet sekä kerätään tiedot tyypillisistä käyttäjistä ja käyttöympäristöstä. Lisäksi hahmotellaan sisältörakenne (infoarkkitehtuuri) karkealla tasolla.

Toisessa määrittelypalaverissa 1-2 viikon kuluttua tarkennetaan sisältörakennetta, valitaan käytettävät mediat ja aihealueet, joihin panostetaan eniten suunnittelu- ja toteutustyötä, sekä jaetaan sisällöntuotantovastuita ja pohjustetaan projektisuunnitelmaa.

Viesti menee tehokkaimmin perille, kun oppijaan vaikutetaan tunnetasolla. Konseptisuunnittelija valitsee missä kohdin sisältöä tähän on varaa.



Konseptisuunnitelma

Seuraavassa esitellään konseptisuunnitelman tyypillinen sisältörakenne. Dokumentin tärkeimmät osat ovat pedagoginen konsepti, sisältökonsepti, infoarkkitehtuuri ja layout-ehdotus. Käytännössä konsepti on n. 10 sivun dokumentti, jota täydentävät esim. PPT-muotoiset sivukartat sekä 2-6 layout-esimerkkiä.

Tiivistelmä ja projektin tausta

Tiivistelmä kiteyttää konseptin ja suunnitelman. Sitä voi käyttää sisäisen markkinoinnin ja viestinnän apuna. Projektin tausta kertoo, miksi projektia ollaan tekemässä: kenen toimesta hanketta tehdään, mihin siinä pyritään, mitkä olivat lähtökohdat. eLearning-sisältöprojekti voi kestää pitkäänkin (tyypillisesti 3-8 kk). On hyvä kirjata lähtökohdat muistiin!

Projektin tavoitteet

Hyvällä eLearning-sisältöprojektilla on muutama tarkasti kuvattu oppimistavoite. Tavoitteet voivat olla mitattavia: esim. tietty määrä oppijoita suorittanut kurssin tietyillä pisteillä määrätyssä ajassa.

Kohderyhmä

Oletetun kohderyhmän erittely ja esittely ovat tärkeitä sekä oppisisällön rakenteen että sisällön ja käyttöliittymän suunnittelun kannalta. Käyttäjät voidaan esitellä stereotyyppisinä henkilöinä, joista on kirjoitettu kuvaus (2-6 kpl). Näiden hahmojen ajattelu auttaa sisällön suunnittelijoita oikeille raiteille.

Käyttöympäristö

Tässä kuvataan kaksi asiaa: Millaisessa ympäristössä sivustoa käytetään (kotona, töissä, ulkona, sisällä, mobiili/pöytäkone...) ja käyttäjäkoneiden yleiset tekniset ominaisuudet (joko tiedetyt tai toivotut). Nämä tiedot ovat käytettävyyden ja sisältösuunnittelun kannalta tärkeitä!

Tekninen ratkaisu

Tämä kuvaus palvelin-, ylläpito- ja toteutusvälineistä listaa palvelinympäristön, julkaisujärjestelmät, Flash-versiot, mediatyypit, tarvittavat ohjelmat ja lisenssit yksityiskohtaisesti. Tieto on tarpeellista toteuttajille ja suunnittelijoille, ja varmistaa myös, että asiakas tietää tekniset vaatimukset.

Pedagoginen konsepti

Pedagoginen lähestyminen ja ratkaisut kuvataan tässä osassa. Itseopiskelupaketeissa pedagoginen lähestyminen painottuu oppimistavoitteiden mukaisen sisällön selkeään esitykseen kohderyhmä huomioiden, ja aktiivisen oppimisen tukemiseen sopivilla harjoituksilla ja animaatioilla.

Infoarkkitehtuuri

Infoarkkitehtuuri esittelee sivuston navigaatiorakenteen, sivukartan. Tämä voi olla erillinenkin dokumentti, joka toteutetaan esim. Wordin numeroituna sisällysluettelona tai PPT:n organisaatiotyökalulla. Infoarkkitehtuuri antaa kuvan sisällön laajuudesta ja navigoinnista.

Sisältökonsepti

Tässä on koko konseptin pihvi. Sisältökonsepti kuvaa kurssin toiminnan ja siinä käytetyt sisältöratkaisut (animaatiot, pelit, videot, keskusteluryhmät, linkit…). Tämä on oleellista tietoa layoutin, kuvituksen ja sisältötuotannon tarpeisiin! Myös sisällön kieliversiointitarpeet on huomioitava ajoissa.

Layout-ehdotus

Ehdotus tai lopullinen suunnitelma siitä, miltä eLearning-ratkaisu näyttää. Leiskat voivat olla ”rautalankamalleja” tai viimeisteltyjä ehdotuksia. Layout-ehdotukset ovat ainoa tapa saada kommentteja projektin omistajilta tai kohderyhmältä! Layout-ehdotuksia voi myös käyttää käytettävyystestaukseen.

Ratkaisun ylläpito

Alustava suunnitelma ratkaisun ylläpidosta: sen aikataulutus, vastuut ja vaadittavat resurssit (omat ja vieraat). Esim. perehdytyssisältöjä on päivitettävä vähintään kerran vuodessa ja päivityksen budjetti on tyypillisesti 5-10% tuotantohinnasta.

Yhteenveto

Koska konseptidokumentti on suurelta osin tekstimuotoinen, se jättää vielä varaa projektin osapuolten omille tulkinnoille ja väärinkäsityksille. Parhaassa tapauksessa määrittelyvaiheen lopputulos onkin jopa ”klikattava demo” jossa ainakin sisällön rakenne, laajuus ja käyttöliittymä on konkretisoitu.

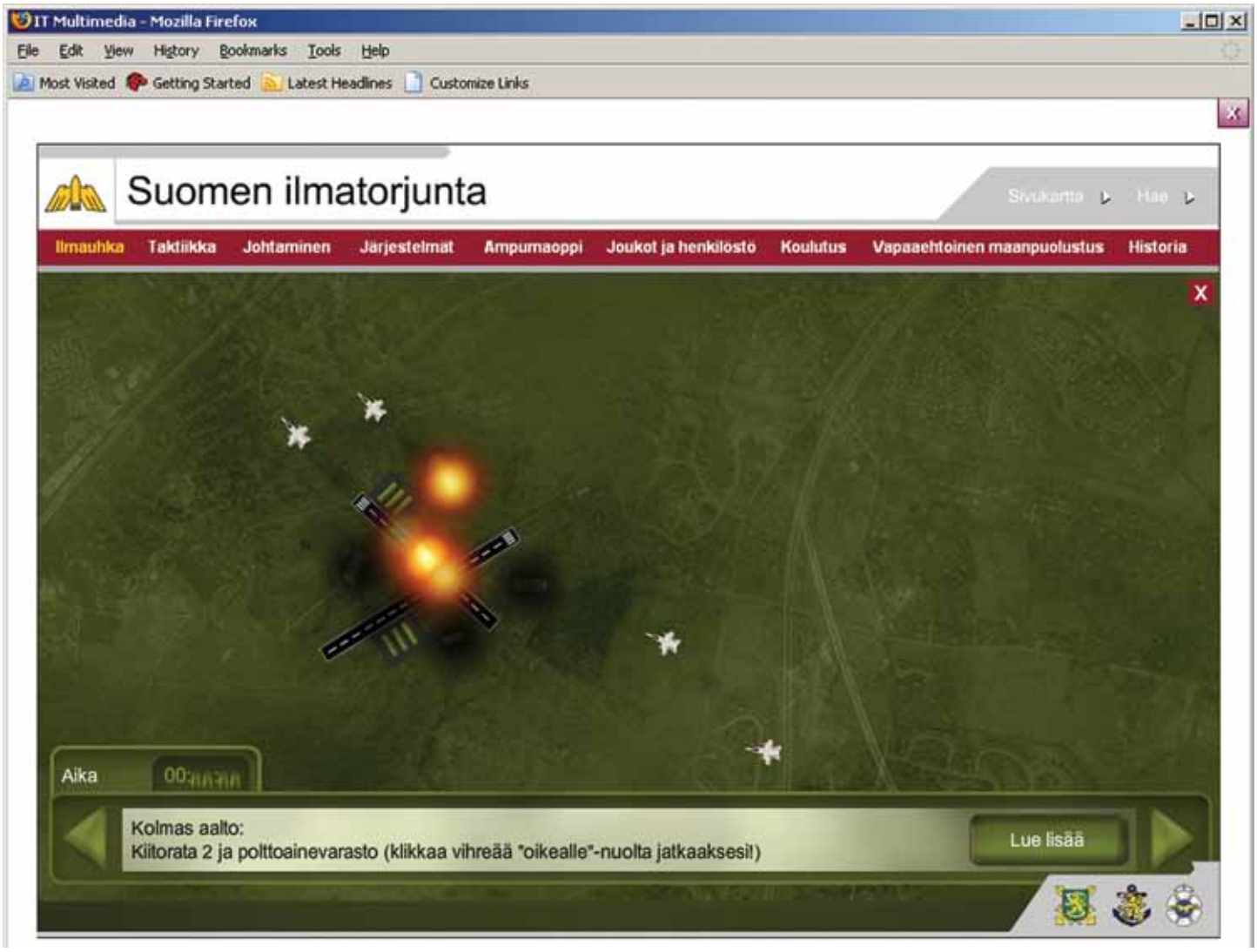
Ylläkuvattu konseptimalli on ”keskiraskas” mutta sitä voi keventäen soveltaa pientenkin oppisisältöjen toteuttamiseen. Nämä asiat joudutaan joka tapauksessa aina selvittämään sisältösuunnittelussa.

eLearning-sisältöjen tulevaisuudesta

Erityisen haasteen eLearning-sisältöjen suunnittelulle asettaa nuorten työntekijöiden mukanaan tuoma todellinen digitaalinen kuilu: parikymppiset käyttävät tietotekniikkaa täysin eri tavalla kuin jo varhaiskeski-ikäisetkin, ja he ovat käyttäneet sitä oppimiseen jo opiskeluaikanaan. Heidän medialukutaitonsa ja odotuksensa ovat eri tasolla kuin lyijykynäkauden kasvattien. Oppisisältöjenkin olisi oltava yhä lyhyempiä, mediapitoisempia ja keskityttävä mielellään vain yhteen asiaan kerrallaan - voitaisiin puhua ”oppimisen musiikkivideoiden” tai oppimispelien käsikirjoittamisesta.

Heterogeeniselle kohderyhmälle sopivan yhden sisällön tuottaminen vaikeutuu entisestään. Hyvänä puolena on mielekkään (lue: ei-teksti-

painotteisen) verkkosisällön tuotantovälineiden kehittyminen ja kustannusten lasku. Hyvä eLearning-konsepti ja toteutus on usein melko kulttuurisidonnainen kieleltään ja ilmiasultaan, joten tuotannon ulkoistaminen muihin maihin ei aina ole mahdollista.



Ilmauhka-käsitteen havainnollistamista animaation keinoin.

Konseptisuunnittelusta, case Suomen ilmatorjunta

Kesällä 2009 julkistettava Suomen ilmatorjunta -multimediaesitys ja verkkopalvelu oli konseptisuunnittelun näkökulmasta mielenkiintoinen hanke. Projektin taustalla oli tarve tuottaa aselajista multimediaesitys sekä DVD:lle että puolustusvoimien koulutusportaalin verkkosotakouluun. Vastaavat aiemmat esitykset toimivat konseptin malleina, mutta sähköisen ilmaisun, kohderyhmän, käyttöympäristön ja internetin muutokset aiheuttivat konseptiin päivitystarpeita.

Projektin kohderyhmä on laaja: varusmiehet, reserviläiset ja puolustusvoimien palkattu henkilöstö. Ikähaitari siis vaihtelee noin 15-70 vuoden välillä.

Puolustusvoimien projektit ovat muotonsa ja sisältönsä puolesta monia muita hankkeita yksityiskohtaisempia ja tarkempia, ja samalla erikoisia. Kuten minkä tahansa aiheen kanssa, tässäkin konseptisuunnittelijaa auttaa mahdollinen aihealueen ymmärtäminen ja esitieto. Näin suuressa projektissa kokemuksella saadaan aikaan jo huomattavia kustannus- ja aikasäästöjä.

Peruskonseptin päivitys tapahtui muutamissa määrittelypalavereissa käytyjen keskustelujen pohjalta parin kuukauden aikana. Pienempiä sisältökokonaisuuksia konseptoitiin projektin aikana, esim. ”Lentorykmentin ilmaisku lentotukikohtaan”-animaatio. Projektin aikana todettiin, lähdemateriaalia löytyvän internetistä niin hyvin, että konseptiin sisällytettiin linkkilistat. Perinteisesti linkkejä internetiin on näissä kursseissa vältetty jotta oppija keskittyisi juuri käsillä olevaan asiaan. Lisäksi puolustusvoimien verkosta ei ole ollut pääsyä internetiin tietoturvasyistä.

Konseptisuunnittelun tärkein anti ilmatorjunnan esittelyyn oli ”Ilmauhka”-käsitteen tuominen kokonaisuuteen mukaan. Ilmauhka on se viholliskuva ja vastavoima, jonka takia aselaji on olemassa. Ilmauhka muodostaa taustan ja motivaation aselajin esittelylle. Tarinankerronnan ja mediasuunnittelun puolesta ilmauhka tarjoaa myös dramatisoinnin mahdollisuuksia. Kohderyhmänä oleville reserviläisille ja varusmiehille ilmauhka-käsite on vieras tai vanhentunut. Konseptisuunnittelun ansiosta nähtiin metsä puilta.

Teksti: Riikka Lehtikuja

Kirjoittaja on valtionhallinnon it-konsultti Logicassa. Hän tekee vaatimusmäärittelyä, määrittelyä ja katselmointia sekä jatkuvissa palveluissa että kehittämisprojekteissa.

Vaatimusmäärittelyn taso ja työn jatkaminen

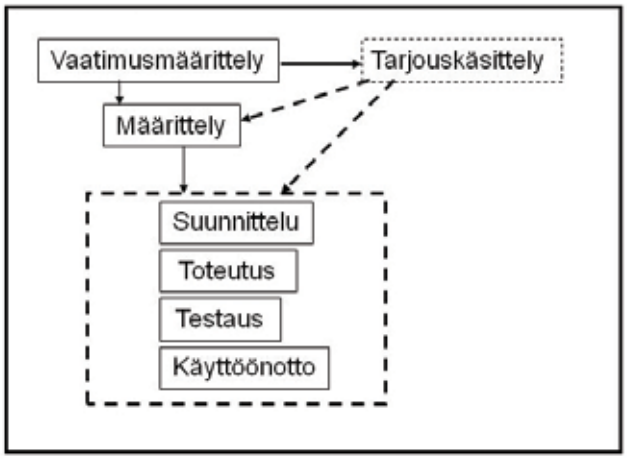
Vaatimusmäärittely on projektin systeemityövaiheista ensimmäinen ja jatkettaessa eteenpäin projektin tekijäkaarti voi vaihtua. Vaatimusmäärittelyn pohjalta järjestetään usein tarjouskilpailu ja sen voittaja voi olla eri kuin vaatimusmäärittelyn tekijä. Vaatimusmäärittely voidaan tehdä myös tilaajan omin voimin.

Toimittajan vaihtumisesta on omat hyötynsä: esimerkiksi systeemityön taso ja dokumentaation kattavuus tulevat herkemmin esille ja puutteet huomataan ajoissa.

Tämän artikkelin pohjana ovat kirjoittajan omat kokemukset valtionhallinnon järjestelmien määrittelyissä, joissa on käytetty oliomäärittelyä (kuvauksena luokkamallit ja käyttötapaukset).

Vaatimusmäärittely tarjouspyynnön osana

Systeemityön eri vaiheet (vaatimusmäärittely-määrittely-suunnittelu-toteutus-testaus-käyttöönotto) ovat vakiintuneet, mutta vaatimusmäärittelyn ja määrittelyn rajanveto ei ole kuitenkaan selkeä. Vaatimusmäärittely tehdään usein sekä järjestelmän määrittelemiseksi että tarjouskilpailun aineistoksi. Tarjouspyynnön liitteenä oleva järjestelmän kuvaus voi olla kuitenkin nimeltään vaatimusmäärittely tai määrittely ja sen sisällön taso voi olla hyvin vaihteleva. Myös käsitys siitä, kuinka paljon alkavassa projektissa tarvitaan lisämäärittelyä, jotta suunnitteluun ja toteutukseen voidaan työn pohjalta edetä, vaihtelee. Tarjouspyynnön tekijällä ja vaatimusmäärittelyn pohjalta tarjousta tekevillä voi olla eri käsitys dokumentaation tasosta: toisen mielestä on tehty jopa määrittely, kun toisen mielestä dokumentaatio on vasta vaatimusmäärittelyn tasolla. Tarjouspyynnön, tarjouksen ja niitä seuranneen sopimuksen puitteissa työtä joka tapauksessa jatketaan.



Kuva 1. Vaatimusmäärittely, määrittely ja tarjouskilpailu toteutusprojektista.

Tarjouskäsittelystä syntyviä hyötyjä

Yhteinen tapa tehdä määrittelyä tai vaatimusmäärittelyä oliomäärittelynä UML-kaavioineen helpottaa tarjouspyynnön ymmärtämistä ja tarjouksen tekemistä sen pohjalta.

Kun vaatimusmäärittelyn dokumentaatio on osa tarjousta, sitä luetaan monessa organisaatiossa. Parhaimmassa tapauksessa sekä työtä tilaava organisaatio että tarjouksen tekijät (erityisesti tietysti tarjouskilvan voittaja) saavat siten lisää tietoa siitä, millainen on hyvä vaatimusmäärittely ja minkälaisesta määrittelystä on tehokasta jatkaa eteenpäin. Samalla kertyy kokemusta ja tietoa mahdollisista tulevista riskeistä projektin aikana.

Työn jatkaminen toisten tekemästä vaatimusmäärittelystä

Alalla vallalla olevat kuvaustavat helpottavat tarjouskilpailun voittajan jatkamista seuraaviin työvaiheisiin.

Toiminnallisuuden osalta suunnittelu ja toteutus vaativat oliomäärittelyltä käsitemallin, tilakaaviot, käyttötapauskaaviot ja käyttötapaukset toimivana loogisena kokonaisuutena. Jos tarjouspyynnön mukana olleet kuvaukset eivät muodosta ehjää kokonaisuutta, on niitä täydennettävä.

Varsinkin jos toimittaja on vaihtunut, on vaatimusmäärittelyn läpikäynti joka tapauksessa tarpeen, mikä vahvistaa lopputuloksena syntyvän järjestelmän laatua. Tässä vaiheessa aiempien tekijöiden oliomäärittelytaidot näkyvät ja ne vaikuttavat projektin etenemiseen.

Vaatimusmäärittelyn kuvausten ongelmia

Kuvaan seuraavassa tyypillisimpiä ongelmatilanteita, joihin olen törmännyt ja joista aiheutuu erityistä vaivaa.

1. Käyttötapausten puolinaiset rakenteet

Käyttötapauskaavioiden ja käyttötapausten rakenteiden puutteista ovat seurauksena toiminnat ja säännöt, jotka on kopioitu moneen käyttötapaukseen, tai joista viitataan toisten käyttötapausten yksittäisiin kohtiin.

Korjaamattomasta tilanteesta seuraisi huonoa toteutusta tai ainakin työlästä suunnittelua, ja erityisesti hankaluutta järjestelmämuutosten yhteydessä.

2. Käsitemallin puutteet

Liiketoimintakäsitteiden keskinäisiä suhteita ei ole mietitty vaadittavalle tasolle: esimerkiksi

- miltei kaikki käsitteet on liitetty toisiinsa
- liitokset on kuvattu varmuuden vuoksi ”kattavasti” (0..n)
- periytymistä ei käytetä oikein: ennemmin jätettäisiin periytyminen kuvaamatta, kuin kuvataan väärin

Tällaiset ongelmat nähdään helposti luokkakaaviosta, vaikkei projektin aiheesta ei vielä tiedetä mitään.

3. Aiempien järjestelmien kuvaus- ja käsittelypuutteet

Vanhan järjestelmän puutteellinen kuvaus on suuri ongelma: aiemman järjestelmän toiminnallisuutta, jota ollaan siirtämässä uuteen järjestelmään, ei ole kuvattu.

Syynä voi olla, että vanhan järjestelmän kuvauksia ei ole ylläpidetty dokumenteissa tai että niitä ei ole kuvattu käyttötapauksina, joten vanhan ja vaatimusmäärittelyn tekijöille tutun kuvaamiseen ei ole panostettu, vaan ennemmin on keskitytty uuteen toimintaan. Seuraus: voidaan saada ihmeteltäviksi vanhan järjestelmän oikotienä tehdyt ratkaisut, jotka on kuvattu käsitteellisesti oudosti.

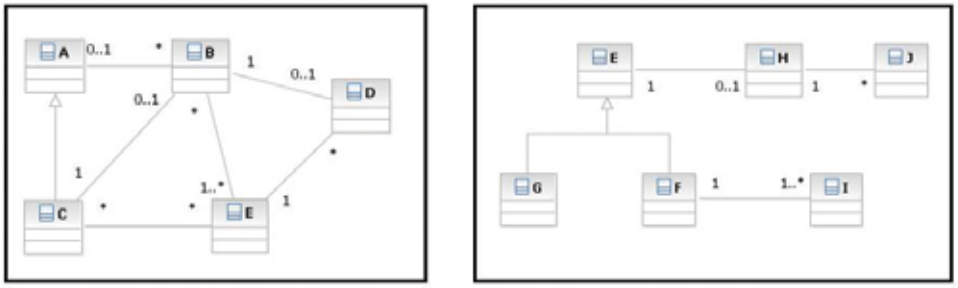
Molemmissa tapauksissa voidaan menettää monien vuosien kehittämistyön aikana tuotettuja käyttäjäkunnalle arvokkaita palveluita, joita halutaan siirtää uuteenkin järjestelmään.

Jos vanha järjestelmä aiotaan jättää toimimaan entisellään ja siihen rakennetaan liittymät uudesta järjestelmästä, on liittymät mietittävä jo uuden järjestelmän vaatimusmäärittelyvaiheessa. Muuten seurauksena voi olla, että järjestelmien välisiä rajapintaratkaisuja voidaan joutua tekemään keskelle hyvin vilkasta tietojen hakua. Se on työmäärältään, tarkoituksenmukaisuudeltaan ja tehokkuudeltaan huono ratkaisu.

Edellä mainitut ongelmat aiheuttavat tietysti missä tahansa systeemityön vaiheessa vaikeuksia, mutta erityisesti tekijöiden vaihtuessa. Sen sijaan, että työ voi jatkua suoraan kohti toteutusta, joudutaan käymään käsitemalli perusteellisesti alusta alkaen läpi ja suunnittelemaan käyttötapausrakenteet uudelleen, tai tekemään laajoja puuttuvien osien kuvauksia.

Oppimisen paikka

Lääkkeeksi erilaisiin vaikeuksiin käy perinteinen tapa eli oppiminen. Tähän kuuluu kouluttautuminen sekä systeemityön tehtävissä että menetelmissä ja aktiivinen ote työyhteisön systeemityöprosessien kehittämiseen: oppiminen vaatimusmäärittelyn ja tarjouskilpailun ongelmista ja hyvistä kokemuksista.



Kuva 2. Käsitemallista epäilyttävä ja paremman näköinen rakenne.

Teksti: Ilkka Pirttimaa

Kirjoittaja työskentelee pääarkkitehtina konsernin tietohallinnossa Stockmannilla. Työssään hän pyrkii löytämään ja käyttöönottamaan menetelmiä, joilla saadaan kehitettyä liiketoiminnan tarpeisiin sovelluksia jotka ovat helppokäyttöisiä ja turvallisia.

Tietoturvan määritteleminen väärinkäyttötapauksia mallintamalla

Perinteisesti järjestelmän määrittelytyössä tunnistetaan käyttäjät ja heidän käyttötapauksensa. Järjestelmän turvallisuus määritellään kuitenkin usein ei-toiminnallisena vaatimuksena, jolloin usein lopputuloksena on myös ei-toiminnallinen turvallisuus. Tässä artikkelissa esitellään menetelmä, kuinka väärinkäyttäjiä ja väärinkäyttötapauksia mallintamalla myös järjestelmän turvaominaisuudet saavat toteutuksessa paremmin tavoiteltavan muodon. Lopuksi esitetään, kuinka väärinkäyttötapausmallinnus voidaan liittää osaksi yritysarkkitehtuuritasoa ja mikä tärkeintä – Kuinka piirrosvälineet opetetaan uusille tavoille.

Järjestelmän toiminnallisuuden kuvaus käyttötapauksina

Lähes kahden vuosikymmenen ajan järjestelmän toiminnallisuuden määrittämiseksi on käytetty käyttötapauskuvauksia. Kaikki tuntevat UML:n visuaalisen tavan piirrellä järjestelmän käyttäjiä tai suunniteltavan järjestelmän ulkopuolisia järjestelmiä tikku-ukkoina. Nämä käyttäjät, aktorit, käyttävät käyttötapauksia, jotka mallinnetaan valkoisina soikioina. Käyttötapaukset voivat puolestaan kutsua toisia käyttötapauksia. Varsinaiset käyttötapaukset voidaan kuvata hyvinkin tarkkaan tekstinä, jossa kuvataan, miten käyttäjän ja järjestelmän vuorovaikutus kulkee, esiehtoineen ja poikkeuskäsittelyineen.

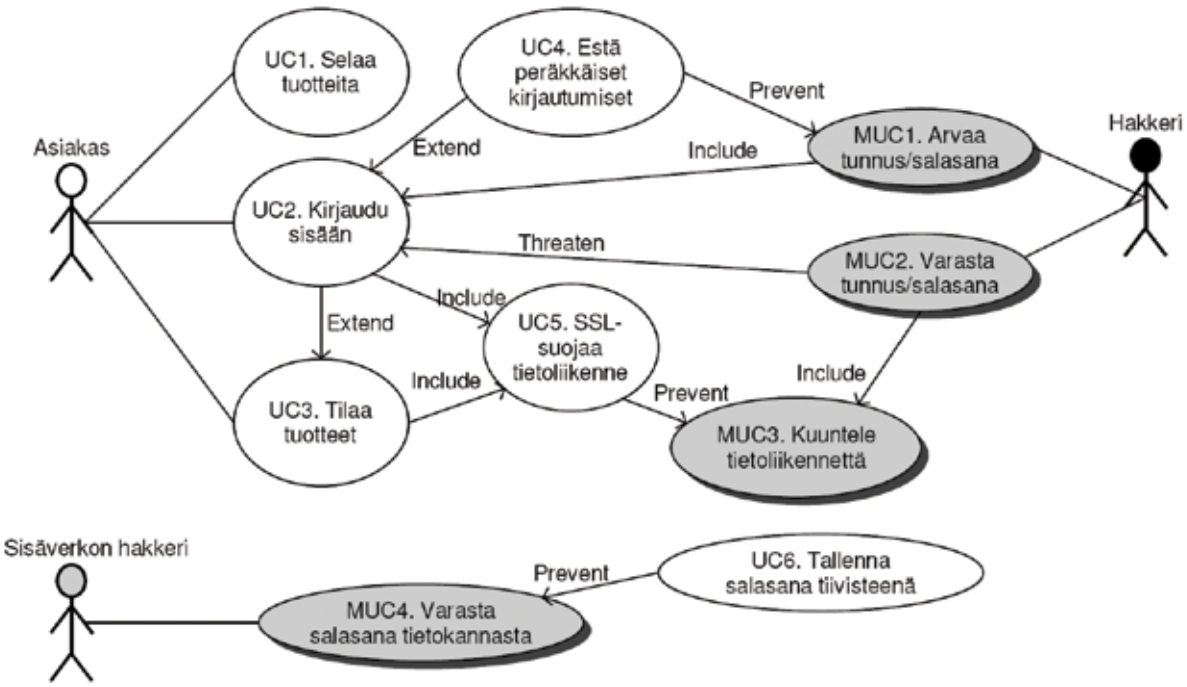
Tässä artikkelissa on esimerkkikaavio (Kuva 1.) kuvitteellisesta nettitilausjärjestelmästä, jonka liiketoimintavaatimuksina on tuoteselailu (UC1), sisäänkirjautuminen (UC2) ja tuotteiden tilaus (UC3).

Järjestelmien tietoturvaongelmat

Kun järjestelmän käyttötapauksille määritellään testitapauksia, hyvä määrittelijä lisää mukaan testejä, joilla voidaan havaita myös järjestelmään mahdollisesti syntymiä tietoturvaongelmia. Tämä menettely ei kuitenkaan aina takaa, että järjestelmän turvallisuutta olisi kattavasti mietitty tai suunniteltu.

Usein tietoturvavaatimukset kuvataan yleisinä ei-toiminnallisina vaatimuksina, tyyliin ”Järjestelmän on oltava tietoturvallinen”. Tämä helposti kuitataan ajatuksella, että kunhan järjestelmä asennetaan palomuurin taakse ja ylläpitäjät tekevät alustaan tietoturvapäivitykset, niin järjestelmä on turvallinen. Käytännössä tietoturvaongelmat ovat nykyään ohjelmiston toiminnallisuuteen liittyviä ja näin ollen palomuurilla ne eivät häviä.

Tällaisia ongelmia ovat mm. mahdollisuus varastaa tunnuksia/salasanoja tai muuttaa järjestelmän tietokantaa SQL Injection-hyökkäyksillä puuttellisen syötteentarkistuksen takia.



Kuva 2. Liiketoiminnan käyttötapaukset sovellukselle ovat UC1-3. Hyökkäjän käytttötapaukset ovat MUC1-4. Järjestelmään on määritelty UC4-6 estämään hyökkääjien aikeet.

Myös järjestelmän toteutusta tarjoavalle toimittajalle ei-toiminnalliset vaatimusmääritykset ovat haastavia: Kyseessä on toiminnallisuus, joka ei ole mitään konkreettista. Mikäli järjestelmän tarjouksessa varaudutaan tarkemmin määrittelemättömien turvallisuusaspektien toteutukseen, on tarjous kalliimpi kuin kilpailijalla.

Väärinkäyttäjät ja väärinkäyttötapaukset

2000-luvulla Norjalaiset Guttorm Sindre ja Andreas L. Opdahl esittivät laajennoksen perinteiseen käyttötapausmallinnukseen. Ajatuksena on tunnistaa järjestelmän käyttäjien lisäksi potentiaaliset väärinkäyttäjät. Väärinkäyttäjät kuvataan mustina tikku-ukkoina. Tämän jälkeen mietitään, mitkä ovat väärinkäyttäjien käyttötapaukset. Esimerkissäni hakkeri yrittää arvata brute-forcella käyttäjätunnuksia ja salasanoja (MUC1) sekä varastaa tunnuksia/salasanoja (MUC2). MUC1 sisältää käyttötapauksen UC1 (Kirjaudu sisään) – Näin on siis määritelty, kuinka hakkeri pyrkii väärinkäyttämään järjestelmää.

Kun väärinkäyttötapaukset on tunnistettu voidaan suunnitella uusia järjestelmään toteutettavia ominaisuuksia, joilla estetään väärinkäyttäjän käyttötapaus. Annetussa esimerkissä näin saatiin järjestelmään kaksi uutta toiminnallista ominaisuutta: UC4 estää peräkkäiset kirjautumisyritykset ja UC5 lisää vaatimuksen kirjautumistoiminnon ja tuotetilauksen tietoliikennesalaukselle. Samalla saadaan lisää systematiikkaa testitapausten määrittelylle: On helppo lisätä testit, joissa tarkistetaan UC4:n ja UC5:n toimivuus.

Sisäverkon väärinkäyttötapaukset

Sindren ja Opdahlin menetelmästä löytyy myös oma harmaan värinen tikku-ukko sisäverkon väärinkäyttötapausten mallintamiseen. Mallissani olen piirtänyt tämän vasemmalle kuvaamaan paremmin järjestelmien käyttäjien keskuudesta tulevaa uhkaa. Esimerkissäni kuvataan uhka, jossa sisäverkon hakkerin väärinkäyttötapaus on järjestelmän käyttäjien tunnusten ja salasanojen varastaminen suoraan tietokannasta. Tämä väärinkäyttötapaus estetään yksinkertaisesti lisäämällä vaatimus, että salasanat tallennetaan tietokantaan tiivisteinä. Näin uhka on tunnistettu ja siihen on reagoitu suunnittelemalla järjestelmä paremmin.

Muita mahdollisia väärinkäyttäjiä

Väärinkäyttäjien ja väärinkäyttötapausten mallintaminen mahdollistaa kommunikoinnin liiketoiminnan kanssa pohdittaessa riittävää tietoturvan tasoa : Mikä on oikea riskien vs. toteutuskustannusten hinta. On hyvä tunnistaa iso joukko potentiaalisia väärinkäyttäjiä ja kirjata myös päätökset, joissa riski on koettu niin pieneksi, ettei järjestelmään kannata rakentaa suojaus- tai valvontamekanismeja näitä vastaan. Hyviä ehdokkaita väärinkäyttäjiksi ovat esim:

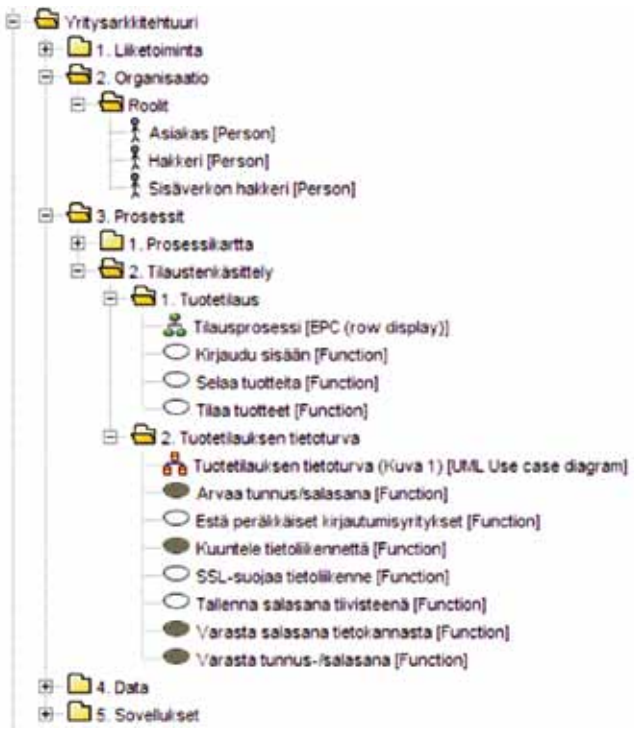
- Sosiaalinen hakkeri puhelimessa
- Sosiaalinen hakkeri työpaikalla
- Siivoojaksi rekrytoitunut hakkeri
- Irtisanottu työntekijä
- Lahjottu työntekijä
- Nettihakkeri
- Kilpailijan palkkaama hakkeri

Väärinkäyttötapaukset osana yritysarkkitehtuureita

Väärinkäyttömallinnuksella on tärkeä osa tietoturvakysymyksissä yritysarkkitehtuuritasolla. On tärkeää, että kuvaukset tehdään johdonmukaisesti keskenään linkittäen ja jo olemassa olevia kuvauksia uudelleenkäyttäen.

Kuvassa 2 esitetään, kuinka myös väärinkäyttäjäroolit on määritelty yleiskäyttöisesti organisaatiohaaraan. Varsinainen liiketoimintaprosessi on kuvattu prosessikaaviona kansiossa ”1. Tuotetilaus” – Siinä muodossa, jossa liiketoiminta haluaa prosessin nähdä.

Kansioon ”2. Tuotetilausten tietoturva” on kerätty vain ne käyttötapaukset, jotka nähdään tietoturvan kannalta oleellisiksi – Osa on väärinkäyttäjän käyttötapauksia, osa on toiminnallisuutta jota tarvitaan niiden estämiseksi. Väärinkäyttötapauskaaviossa (Kuva 1.) varsinaisessa liiketoimintaprosessissa esiintyvät toiminnot ovat oleellisessa osassa – Ne voidaan raahata UML-kaavioon prosessikuvasta, jolloin niiden ulkomuoto muuttuu käyttötapauspallukaksi, objektin periessä kaikki muut tietonsa prosessikaaviosta.



Kuva 2. Väärinkäyttötapaukset osana yritysarkkitehtuurikokonaisuutta.

Mallinnustyökalut

Artikkelia kirjoittaessani havaitsin, että mallinnintyökaluilla voi olla vaikea piirtää mustia/harmaita tikku-ukkoja ja käyttötapauksia. Käyttötapauskaavio ei välttämättä myöskään mahdollista uusien assosiaatioiden piirtämistä (uhkaa, estää). Useilla työkaluilla ”käyttötapausmalliin kuulumattomia” nuolityypityksiä pitää lisätä käsin erillisinä teksti- tai kommenttilaatikoina.

Artikkelin kuvat on piirretty ARIS-mallintimella. Senkään UML-kaaviotyyppi ei tue suoraan tarvittavia assosiaatioita, mutta sen ylläpitotyökaluilla ohjelmalle voidaan opettaa tarvittavat asiat. Objekteille opetettiin 2 uutta attribuuttia, joista ensimmäinen sisältää tarvittavat assosiaatioiden nimet (include, extend, threaten, prevent, …) ja jälkimmäinen (Kuva 3.) sisältää alasvetolistana tiedon, onko kyseessä käyttötapaus, väärinkäyttötapaus vai sisäinen väärinkäyttö. Valittu assosiaatio saadaan näkymään nuolissa ja tikku-ukon/käyttötapauksen väri saadaan vaihtumaan valitun tyypin mukaan. Näin saadaan vakioitua menettely, jolla syntyy Kuva 1:n mukaisia kaavioita.

MS Visiolla tarvittavat assosiaatiot tuodaan käyttöön seuraavasti :

Luodaan uudet stereotyypit:

1. Avaa UML/Stereotypes
2. Lisää 4 stereotyyppiä (include, extend, prevent, threaten) ja valitse niille Base Class=Dependency

Kuva 3. ARIS-mallintimeen voidaan lisätä omia attribuutteja, joilla saadaan lisättyä tarvittava semantiikka

Kaavion piirtäminen

1. Piirrä aktorit ja käyttötapaukset käyttäen ”UML Use Case”-piirrosobjekteja
2. Värjää väärinkäyttäjät ja väärinkäyttötapaukset siten, että fontti on valkoinen ja tausta on musta.
3. Piirrä käyttötapausten väliset suhteen käyttäen ”UML Static Structure”-piirrosobjektivalikoiman Dependency-nuolta
4. Tuplaklikkaa nuolta ja valitse alasvetolistalta haluttu stereotyyppi, jonka aiemmin lisäsit

Yhteenveto

Tietoturvan huomioiminen ohjelmistokehityshankkeissa on usein abstrakti asia, johon ei välttämättä osata oikealla tavalla tarttua. Se, kenen vastuulla tietoturvan rakentaminen on, saattaa olla hämärän peitossa. Arkkitehdit luottavat koodareihin, koodarit luottavat palvelimen asentajiin, palvelimen asentajat luottavat tietoliikennekavereihin. Ilman oikeanlaista suunnittelua ja tietoturvariskikartoitusta järjestelmästä ei tule turvallinen ja sen korjaaminen jälkikäteen voi olla hyvin kallista. Juuri tällaiseen kokonaisvaltaiseen tietoturvasuunnitteluun kuvattu menetelmä on hyvä. Se tarjoaa tarttumapintaa sekä toteuttajille että järjestelmän testaajille – mitä turvaominaisuuksia järjestelmään pitää toteuttaa ja millä niiden toteutukset todennetaan ja testataan.

Olipa suunnittelutyökaluina sitten mikä tahansa ohjelma tai vaikkapa vain kynä ja paperi, suosittelen tämän suunnittelumenetelmän käyttöönottoa. Vain ottamalla menetelmän rohkeasti käyttöön voi työkaluohjelmiltakin odottaa kehitystä tukemaan tätä mainiota menetelmää. Myös UML kehittyy!

Artikkelissa käytetyt suomennokset

Käyttötapauskuvaus		**Väärinkäyttökuvaus**	
Actor	Käyttäjä	Misuser	Väärinkäyttäjä
Use case	Käyttötapaus	Misuse case	Väärinkäyttötapaus
Include	Sisältää	Threaten	Uhkaa
Extend	Laajentaa	Prevent, Mitigate	Estää

Lähteet:

Guttorm Sindre, Andreas L. Opdahl:

Capturing Security Requirements through Misuse Cases

How do you become smart?

by Ivar Jacobson

"You need to be smart" was my previous column. But how do you become smart? I really think being smart is at the core of being successful in software. The answer depends on who you are. Let's assume you are a leader of some sort, for instance a project manager.

Smart is not the same thing as being intelligent. You can be intelligent without being smart. I know lots of people who are intelligent but not smart. You can be very smart without being very intelligent.

Smart is not the same as having common sense. You can have common sense without being smart, but if you are smart you must however have common sense.

Smart is to do exactly right, not to find a broad solution that is just about right. If you are smart you are very sensitive to finding what is right.

To become smart you need first and foremost to have knowledge and secondly to have experience applying that knowledge. You need knowledge to understand what software development is all about. You need to know how to get good software, quickly and at low cost. It is as simple as that <grin>.

Good software is useful (obvious but not trivial), reliable (works always) and extensible (can grow as long as you want to use it). You need to know how you create good software which means you need to learn some technical practices such as architecture, use cases and iterations/sprints.

Nothing makes you as quick (and agile) as having motivated people. You get this by adopting proven agile practices such as how you work as a team, how you organize your work, etc. As I have said many times, we all need to be agile.

The most powerful way to get software at low cost is by developing as little software as possible. Instead you acquire the many pieces of software you need from other sources. This is called software reuse. You can reuse your company's old software, buy software from a vendor or download open source software. You need to know how to put it all together with some glue and how to only develop what is really needed. Thus you need some other more advanced practices such as service-oriented architecture or product-line engineering.

Thus you need knowledge on practices.

You also need experience in using these practices. This is not trivial. Experience comes from having seen a lot of what works and what does not, either directly or through listening to others. This is a subtle (I hope) plug for what we do - we work with many different projects and we gain insight into what works and doesn't. But it's also a plug for communities, for being open to new ideas and experiences, and to strive to be continuously learning. There is a saying that good judgment comes from experience, most of which comes from bad judgment. Except that you don't need to repeat the mistakes of others to learn.

Experience makes you smart through "focus" - knowing what to focus on and what can be ignored. It's not the amount of effort applied, but knowing where and when to apply that effort.

Unfortunately, nor knowledge about practices neither experience is something you learn at school. Professors can rarely help you. Where do you learn it then? Most people learn it at work, by doing. For now, let us just re-state the obvious: you need it one way or another.

I will come back to this another time.

You may now ask, given that you have knowledge of important practices and you have experience: "Will I now become smart?"

Of course eventually, it comes back to you. We cannot all become equally smart but we can all become smarter.

Tohtori Ivar Jacobson on ohjelmistomenetelmien vaikutusvaltaisimpia hahmoja, joka on ollut mukana kehittämässä monia ohjelmistoalaa muuttaneita menetelmiä. Näitä ovat: komponentit ja komponenttiarkkitehtuurit, käyttötapaukset, aspektisuuntautunut ohjelmistokehitys, Unified Modelling Language (UML) ja Rational Unified Process (RUP). Viime aikoina hän on keskittynyt käytäntöihin perustuvaan ohjelmistotuotantoon, jonka suosio kasvaa nyt ympäri maailmaa.

Ivar on ketterien (Agile) menetelmien vahva puolestapuhuja ja kehittää niitä fiksummiksi. Hänen mottonsa on "Kaiken tulisi tulla ketterämmäksi, mutta ketterä ei ole kaikki". Ivar on kirjoittanut kuusi suosittua kirjaa ohjelmistotuotannosta. Ivarin yrityksellä, "Ivar jacobson International":lla on tomintaa Yhdysvalloissa, Iso-Britanniassa, Kiinassa, Singaporessa, Australiassa, Saksassa ja Ruotsissa.

Tästä Systeemityö-lehden numerosta alkaen Ivar kirjoittaa lehteemme kolumnia ohjelmistotuotannon muutoksen aallonharjalta, tervettä järkeä ja pitkää perspektiiviä unohtamatta.

Teksti: Pentti Salmela

”Ilman liikkeestä syntyi ajatus.
Sanat saivat sen kiinni ja antoivat sille muodon.
Kirjoitettuna se sai juurensa maahan, pysyi.
Ja se lensi kasvamaan uusia ajatuksia maailman muissa tuulissa.”

Kirjoittaja on työskennellyt suurissa suomalaisissa materiaalivaltaisissa (Kemira Oy, Oy Lohja Ab ja Alko-Yhtiöt Oy) ja tietovaltaisissa (Työterveyslaitos) organisaatiossa it-alan johto- ja asiantuntijatehtävissä. Erilaisissa toimintaympäristöissä ja monissa it–projekteissa mukana olleena hänellä on ollut mahdollisuus seurata systeemityön kentässä käytännössä tapahtunutta kehitystä.

Systeemityön uusi ulottuvuus

Organisaation hiljainen tieto tuo systeemityölle uuden vaatimuksen. Tulee hallita ja ymmärtää myös hiljaisen tiedon prosesseja, tiedon muunnoksia sekä ihmisen osaamisen ja it:n vuorovaikutusta. Haasteena on kuvata nämä prosessit uuden ympäristön edellyttämällä tavalla. On kehitettävä uudentyyppisiä it-projekti- ja systeemityömalleja, jotka huomioivat tietämyksen hallinnan koko kentän.

Systeemityön alueen laajentuminen

Viimeisen viidenkymmenen vuoden aikana systeemityön[a)] kenttä on laajentunut konttorityön automatisoinnista kattaen nyt lähes koko toiminnan ja viestinnän alueen. Lisäksi systeemityön piiriin on tullut kuvassa 1 esitetty hiljaisen tiedon ja ihmisten osaamisen alue. Tämän alueen olemassaolo on ollut selvillä jo 1950-luvulta saakka (Polanyi, 1966). Kuitenkin vasta 1990-luvun lopussa varsinkin tietopainotteisessa organisaatiossa nousi esille tietämyksen hallinta (Knowledge Management) -termi. It:n käytössäkin on etsitty kiinnekohtaa tästä uudesta käsitteestä. Tieto on saanut organisaation hiljaisesta tiedosta ja ihmisen osaamisesta uuden ulottuvuuden. Kuitenkin it:n käytön ja hiljaisen tiedon välinen suhde on koko ajan ollut enemmän tai vähemmän epäselvä. Vielä tänä päivänä hiljaisen tiedon alue on systeemityön uusi, hallitsematon ulottuvuus.

Kuvan 1 aikajanalla on kuvattu tietotekniikan käyttöalueen ja samalla systeemityön alueen laajenemista sen lyhyen historian, 50 vuoden aikana. Käyttöalue on jaettu kolmeen alueeseen; tietotekniikan, toiminnan ja viestityn tiedon sekä hiljaisen tiedon ja osaamisen alueisiin. Kuten pystyakselilla esitetään, tiedon rakenteellisuusaste[b)] vähenee siirryttäessä tekniseltä alueelta, toiminnallisen alueen kautta hiljaisen tiedon alueelle. Systeemityön alue on laajentunut konttorityön automatisoinnista ja hyvin teknispainotteisesta toiminnasta kattamaan suuren osan koko inhimillisen elämän toiminta-alueesta. Tätä systeemityön alueen laajenemista on aina edeltänyt jokin uusi teknisen tason keksintö, väline tai menetelmä, joka on mahdollistanut it:n uuden käyttötavan.

Internet -tekniikkaan siirryttiin 1990-luvun alkuvuosina. Verkko- ja tietoliikennetekniikka tarjosi organisaatioille mahdollisuuden kehittää asiakasprosessejaan niin, että osa prosessin hoitoa siirrettiin asiakkaan itsensä tehtäväksi. Uuden tekniikan avulla pystytään nyt kehittämään verkossa toimivia palveluja ja verkon kautta jaeltavia tuotteita. Dokumenttienhallintajärjestelmät paransivat henkilöstön mahdollisuuksia tuottaa dokumentteja ryhmätyönä. Näin it:llä tuetut prosessit ovat laajentuneet sisältämään myös organisaation tuotekehitys- ja innovaatioprosessit.

Tiedon liikkeen perinteisissä kuvauksissa kuvataan lomakkeiden, dokumenttien ja datan liikkumista erilaisina työnkulku- ja prosessikaavioina. On kuvattu hyvin rakenteellisessa muodossa olevan tiedon liikkumista paikasta A paikkaan B ”horisontaalisesti vaakatasossa”. Nykyisetkin projekti- ja systeemityömallit ottavat hyvin satunnaisesti huomioon tietämyksen hallinnan mukanaan tuoman tiedon uuden ulottuvuuden.

Tiedon eri olomuodot ja käsittelymekanismit

Ikujiro Nonaka ja Hirotaka Takeuchi (1995, s. 73) ovat tarkastelleet uuden tiedon syntymistä organisaatiossa ja tuovat esille implicit (hiljaisen tiedon) - explicit (rakenteellisen) tiedon akselin. He esittävät, että organisaatiossa tietoa syntyy tiedon siirtymisistä tällä akselilla, hiljaisen tiedon

ja rakenteellisen tiedon vuorovaikutuksesta.

Stevensin (1986) mukaan ihmiskunnan informaatiohistoriassa on tunnistettavissa suuria käänteitä, jotka vaikutukseltaan ovat olleet niin merkittäviä, että ne ovat vaikuttaneet ajattelutapoihin. Näitä suuria käänteitä, jotka ovat leimanneet pitkiä aikakausia, ovat ainakin inhimillisen kielen synty, kirjoitustaidon ja kirjapainotaidon keksiminen sekä elektronisen tiedonvälityksen ja -käsittelyn käyttöönotto.

Kaikkien näiden suurien käänteiden taustalla on tunnistettavissa jokin käyttöönotettu uusi tekniikka tai mekanismi, joka on mahdollistanut täysin uudentyyppisen tiedon siirron tai tallentamisen. Samalla se on edellyttänyt tiedon ja sen käsittelysääntöjen saattamista entistä rakenteellisempaan muotoon. Kun suuri käänne on tapahtunut, tiedon rakenteellisuusakselilla on liikuttu aina askel eteenpäin.

Kuvassa 2 Nonakan ja Takeuchin hiljaisen ja rakenteellisen tiedon akseli on jaettu Stevensin esittämiin informaatiohistorian suurien käänteiden mukaisiin luokkiin (Salmela, 2008). Kuvaan 1 verrattuna toiminnan ja viestityn tiedon alue on jaettu puhutun ja dokumentoidussa muodossa olevan tiedon alueisiin. Tieto on näin jaettu neljään sen olomuotoa kuvaavaan luokkaan seuraavasti:

1. Hiljainen eli tacit -tieto on tiedon vähiten rakenteellinen tiedon luokka. Siihen kuuluu kaikki se informaatio, joka aivoissamme on, ja josta vain hyvin pienestä osasta olemme tietoisia (Polanyi 1966). Se on kokonaisvaltaista, kokemukseen perustuvaa tietoa. Siihen sisältyy geneettinen, ruumiillinen, intuitiivinen ja kokemusperäinen tieto.

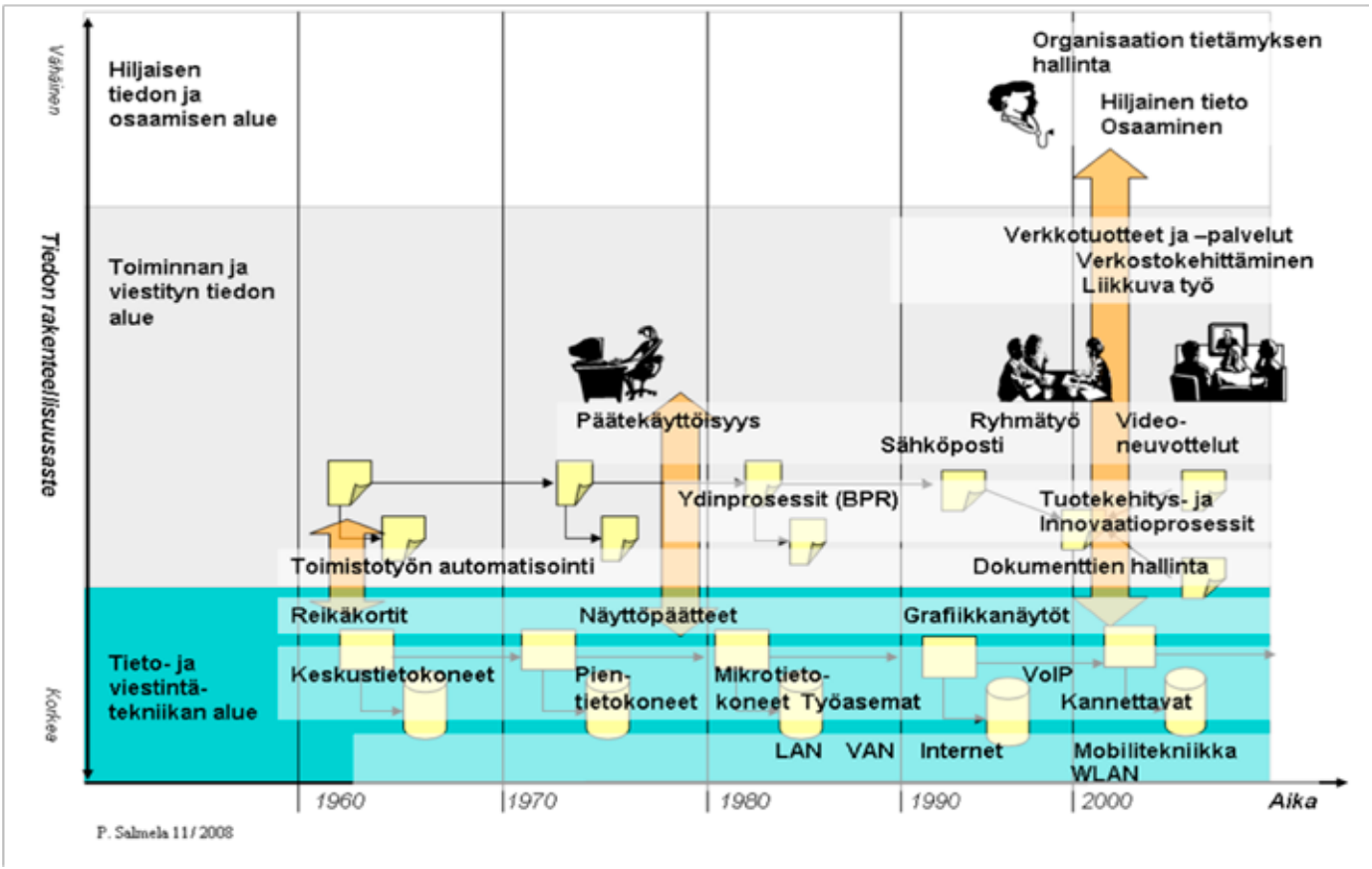
Tähän luokkaan kuuluvan tiedon hallintamekanismeja ovat osaamisen, päättelyn ja innovaation mekanismit, abstrahoiminen, mentaaliset mallit[d)] sekä mallioppiminen. Myös kuvaamataiteet, runous ja musiikki ovat keinoja, joilla hahmotetaan hiljaisen tiedon aluetta.

2. Puheen, puhutun sanan muodossa oleva tieto on seuraava tiedon rakenteellisuutta kuvaava luokka. Puheen tuottaminen on strukturointiprosessi, jolloin puhuja kieliopin syntaksin avulla yhdistelee semanttiset käsitteet ymmärrettävään, kuulijalle välittyvään muotoon.

Puheen muodossa olevan tiedon hallintamekanismeja ovat puhutun kielen syntaksi, kielitaito, retorinen hahmottaminen, kielellistäminen, opettaminen, keskusteleminen ja kuunteleminen.

3. Dokumentoidussa muodossa oleva tieto on astetta rakenteellisempaa, täsmällisempää kuin puhuttu tieto. Merkittävin ero puhuttuun tietoon nähden on kuitenkin tiedon säilyvyydessä, mahdollisuudessa varastoida dokumentoitua tietoa.

Dokumentoidun tiedon mekanismeja ovat kirjoitus- ja kirjapainotaito, kirjoittaminen ja luke-



Kuva 1. Systeemityön alueen laajentuminen.

minen. Ja tiedon varastointivälineitä ovat kirjat, julkaisut, lehdet, kirjastot ja dokumenttitietokannat.

4. It-järjestelmiin ja tietokantoihin tallennettu rakenteellinen tieto on määritelty yksikäsitteisellä tarkkuudella eikä tietoon tai sen käsittelysääntöihin sisälly monitulkintaisuutta. Tiedon käsittelysäännöt ovat ohjelmoitu valmiiksi toiminnoiksi tietokoneohjelmiin niin, että tietojen haku, päivitys, käsittely ja tulostus tapahtuvat automaattisesti, annettujen tai saatujen käskyjen mukaan.

Tietokantoihin tallennetun rakenteellisen tiedon hallintatekniikoita ovat systemointi- ja systeemityömenetelmät, ohjelmointikielet, tietokannat, tietoverkot, käsiteanalyysit, tieto- ja viestintätekniikka.

Tietojärjestelmän kehittäminen ja käyttöönotto tiedon muunnosprosessina

Perinteisesti tietojärjestelmän[e)] kehittäminen on perustunut vaiheittaiseen etenemismalliin (Sell, 1976, Tietojenkäsittelyliitto, 2002). Kuvassa 2 it-projektin etenemistä on tarkasteltu sekä aika- että tiedon rakenteellisuutta kuvaaville akseleille sijoitettuna. Kaaviossa on esitetty tiedon liikkumisen päävirta kohde- (tilaavan tai käyttäjä-) ja toimittavan organisaation sisällä ja välillä. Toimittavalla organisaatiolla ymmärretään projektiorganisaatiota, jossa yleensä on mukana tilaavan käyttäjäyksikön, it-yksikön ja järjestelmän toimittavan ohjelmistotalon edustajia.

Sen lisäksi että it-projekti etenee vaiheittain aika-akselilla eteenpäin, se liikkuu vertikaalisessa suunnassa alas ja takaisin ylös tiedon rakenteellisuusakselilla. Tieto muuttaa olomuotoaan tietojärjestelmää rakennettaessa mielikuvatason, ei-rakenteellisesta tiedosta rakenteelliseksi tiedoksi. Ja tietojärjestelmää käyttöönotettaessa it-järjestelmäksi[f)] tiivistetty tietämys puretaan koko organisaation uudeksi toimintatavaksi ja osaamiseksi. Puhutussa ja dokumentoidussa muodossa oleva tieto toimii siirtokerroksena hiljaisen tiedon ja tekniikan välissä.

Seuraavassa tarkastellaan tietojärjestelmän rakentamista ja käyttöönottoa vaihe vaiheelta:

1. Projektin perustaminen

Tietojärjestelmän kehittämisenkin lähtökohtana ovat mentaaliset mallit, mielikuvat. Tilaavan organisaation jollakin henkilöllä tai henkilöillä on syntynyt mielikuva, miten it:n avulla voitaisiin kehittää toimintaa. Mielikuva on syntynyt ehkä innovaatioprosessissa, valmisohjelmistoihin tutustumalla, benchmark -lähtöisesti ja uuden tekniikan mahdollisuuksia arvioimalla.

Kehittämistyössä organisoidutaan projektiksi. Tämä varmistaa eri osapuolten vuorovaikutuksen, osaamisen siirtymisen ja suunnitelmallisen etenemistavan. Projektiryhmässä on edustettuna eri osaamistaustaisia, toisaalta toimintaa tuntevia, it-järjestelmän tulevia käyttäjiä ja it-asiantuntijoita.

2. Esitutkimus

Tietojärjestelmän rakentamista edeltävänä toimenpiteenä nimetty projekti- tai selvitysryhmä yleensä selvittää kannattaako kehittämishankkeeseen ryhtyä. Rakennettavasta järjestelmästä tehdään esiselvitys tai -tutkimus.

Esitutkimusvaiheessa vielä operoidaan tyypillisillä hiljaisen tiedon välineillä; mentaalisilla malleilla, hahmotelmilla ja mielikuvilla. Kuitenkin tavoitteena on siirtyä mielikuvatason malleista astetta rakenteellisemman, dokumentoidun tiedon tasolle. Mielikuva on saatava jaettua yhteiseksi mielikuvaksi oman organisaation keskuudessa, jotta voidaan päättää, kannattaako hankkeessa edetä. Yhteistä käsitystä etsitään keskustelemalla ja kuvaamalla rakennettavan järjestelmän toimintaa ja tavoitteita, rajauksia ja riskejä.

2.1 Päätös tietojärjestelmän rakentamisesta

Esitutkimusvaiheen lopuksi rakennettava it-järjestelmä kuvataan esitutkimusraportissa sillä tarkkuudella, että syntyy riittävä kuva ja yksimielisyys sen toiminnasta, tavoitteista, saavutettavista hyödyistä ja kustannuksista sekä riskeistä. Näin voidaan tehdä päätös it-järjestelmän rakentamisesta tai rakentamatta jättämisestä.

Esitutkimusraportin perusteella tehtävä päätös it-järjestelmän pääpiirteistä, sen tavoitteista ja kustannuksista sekä projektin käynnistämisestä vastaa Nonakan mallin hyväksymisvaihetta:

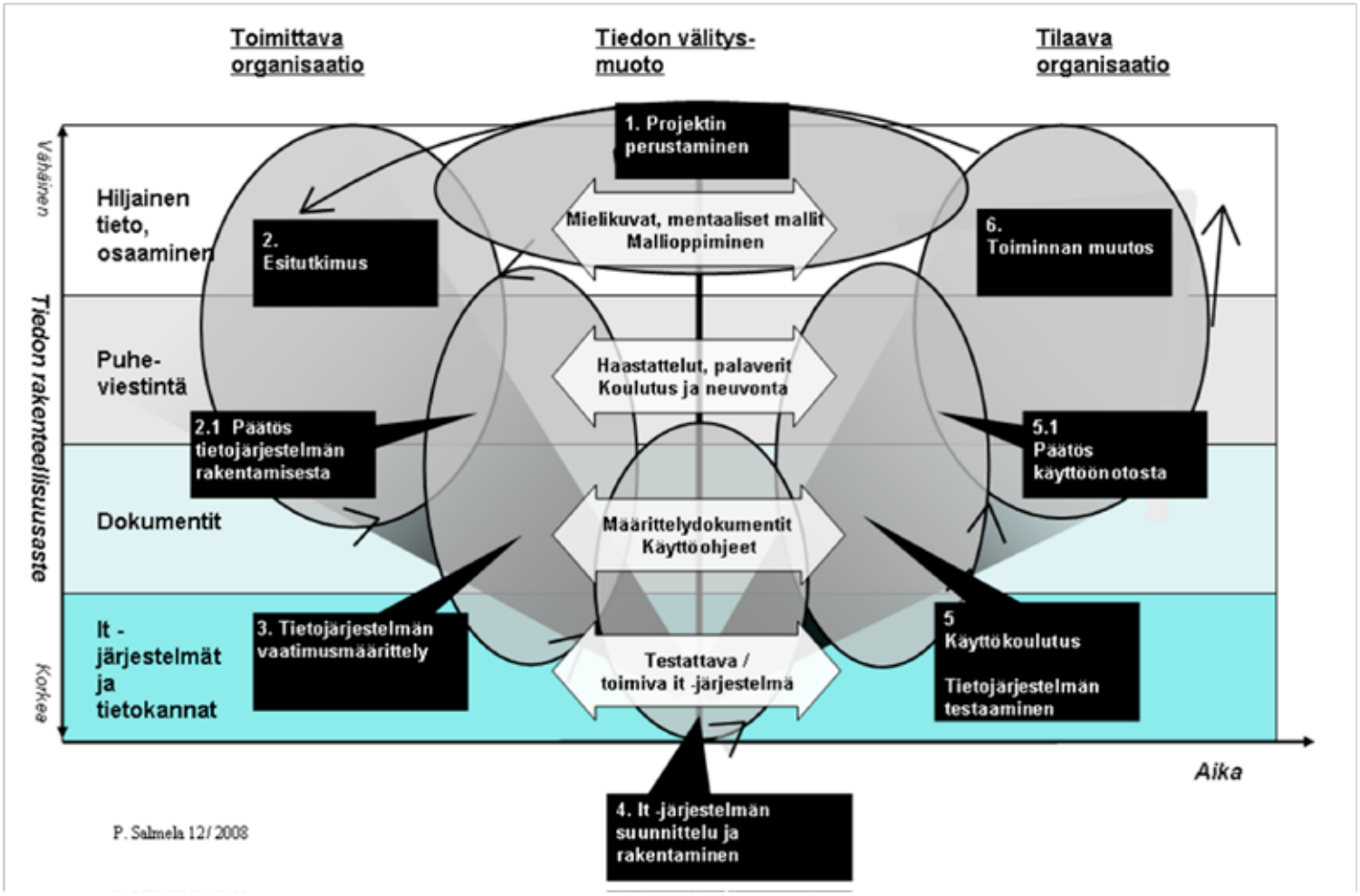
Käsitteelliselle tasolle hahmotettua ideaa tms. arvioidaan organisaation arvoja, tavoitteita ja päämääriä vasten, esim. onko idea taloudellisesti kannattava ja edelleen kehittämisen arvoinen tai sopiiko tuote yrityksen imagoon.

3. Tietojärjestelmän vaatimusmäärittely

It-järjestelmän rakentamisprojektin vaatimusmäärittelyvaiheessa haastatellaan tulevan it-järjestelmän käyttäjiä. Järjestelmän vaatimuksista sovitaan projektipalavereissa. Vaiheen tavoitteena on rakenteellistaa vielä osittain mielikuva -tasolla oleva it-järjestelmä järjestelmän toiminnallisiksi vaatimuksiksi. Vaatimusmäärittelydokumentissa kerrotaan yksikäsitteisesti uuden järjestelmän tavoitteet, miten sen tulee toimia. Mielikuvien ja puheiden tasolta siirrytään rakenteellistetun dokumentoidun tiedon tasolle.

4. It-järjestelmän suunnittelu ja rakentaminen

Tietojärjestelmän rakentamisvaiheessa tekemisen painopiste on it-ammattilaisilla. Heidän tehtävänsä on muuntaa puheen ja dokumentoidun tiedon tasolla oleva tieto tietokoneen kielelle, toimivaksi ohjelmaksi. Yleensä tämäkin tapahtuu



Kuva 2. Tietojärjestelmän kehittäminen ja käyttöönotto tiedon muunnosprosessina.

kaksivaiheisena prosessina:

A. Rakennettavasta it-järjestelmästä tehdään kirjallinen it-tekninen määrittely eli it-suunnitelma, jossa on kuvattu rakennettava järjestelmä; tietokannat, tietorakenteet, käsittelysäännöt ja tulosraportit it-käsitteitä ja -terminologiaa käyttäen.

B. Tämän jälkeen kuvattu järjestelmä koodataan tietokoneen ohjelmointikielellä toimivaksi ohjelmaksi.

Vaatimusmäärittelydokumentissa it-järjestelmälle asetetut toiminnalliset vaatimukset ovat järjestelmätoimituksen perusta. Toimivan ohjelmiston rakentaminen vaatii kuitenkin lisäksi täydentäviä keskusteluja it-järjestelmän rakentajien ja käyttäjien kesken.

Kun valmisohjelmisto on kehittämisen lähtökohtana, se antaa valmiin, yhteisen mallin, jonka pohjalta rakentaminen voi jatkua. Valmisohjelmisto on sen toimittajan osaamisen valmis, tuotteeksi rakenteellistettu toiminnan yleinen malli. Mitä paremmin valmisohjelmiston toimintamalli vastaa tilaajan toiminnan mallia, sitä helpompaa uuden, tilaajan tarpeisiin sovitetun järjestelmän rakentaminen on.

5. Käyttökoulutus ja testaaminen

Tietojärjestelmän testaus- ja käyttökoulutusvaiheessa varmistutaan käyttöönottavan järjestelmän oikeasta toiminnasta ja järjestelmän tulevat käyttäjät koulutetaan. Testausvaiheessa tiedonvaihto tapahtuu dokumentoidun tiedon muodossa. Kaikki it-järjestelmän virheet ja puutteet dokumentoidaan, samoin sovitut ohjelmamuutokset.

Käyttäjien koulutus ja käytön neuvonta on pääasiassa puhutun tiedon tasolla tapahtuvaa viestintää. It-järjestelmään tiivistetty osaamista siirretään käyttäjäorganisaation osaamiseksi.

5.1 Päätös käyttöönotosta

Kun it-järjestelmä toimii yhteisesti sovitulla tavalla, käyttöohjeet laadittu ja käyttäjät on koulutettu, voidaan tehdä päätös uuden järjestelmän käyttöönottoajankohdasta.

6. Toiminnan muutos

Toimintatavan muutosvaiheessa it-järjestelmäksi tiivistetty ja rakenteellistettu organisaation tietämys palautetaan laajemman ihmisjoukon voimavaraksi. It-järjestelmän käyttäjäorganisaatio ryhtyy työskentelemään uudella, uuden it-järjestelmän tukemalla tavalla. Työskentely tapahtuu pääasiassa ihmisten hiljaisen tiedon, osaamisen alueella.

It-projektit – rakenteellistamista ja osaamisen yhdistämistä

Ståhle (1999) on tarkastellut tiedon syntymistä oppivassa organisaatiossa. Hänen mukaansa on erotettavissa kolme tietämyksenhallinnan toimintaympäristöä; kaoottinen, kompleksinen ja mekanistinen. Akselit, joiden puitteissa kaoottista toimintaympäristöstä siirrytään mekaaniseen, ovat tiedon järjestäytyneisyys (vastaa tämän artikkelin 'tiedon rakenteellisuusaste' -termiä) ja osaamisen ennakoitavuus. Edellisessä luvussa esitettyä it-projektin kehittämismallia voidaan tarkentaa huomioimalla Ståhlen esittämä tietämyksenhallinnan kenttää määrittävät akselit.

It-järjestelmää rakennettaessa tiedon kaoottisesta toimintaympäristöstä siirrytään äärimmäisen pitkälle järjestäytyneen rakenteellisen tiedon ja ennakoidun osaamisen toimintaympäristöön, jossa toimintasäännöt on automatisoitu. Siirtymä tapahtuu kompleksisen toimintaympäristön, projektissa mukanaolevien osaamista yhdistämällä, dialogin, tulkintojen ja määrittelydokumenteista sopimisen kautta. Kaoottisen maailman ristiriitaiset mielikuvamallit tarkentuvat vaiheittain ensin määrittelydokumenteiksi ja sitten mekaanisesti toimivaksi it-järjestelmäksi.

It-järjestelmän rakentaminen hiljaista tietoa rakenteellistamalla ja osaamista yhdistämällä, tietojärjestelmän käyttöönotto osaamista jakamalla ja sisäistämällä toiminnan muutos.

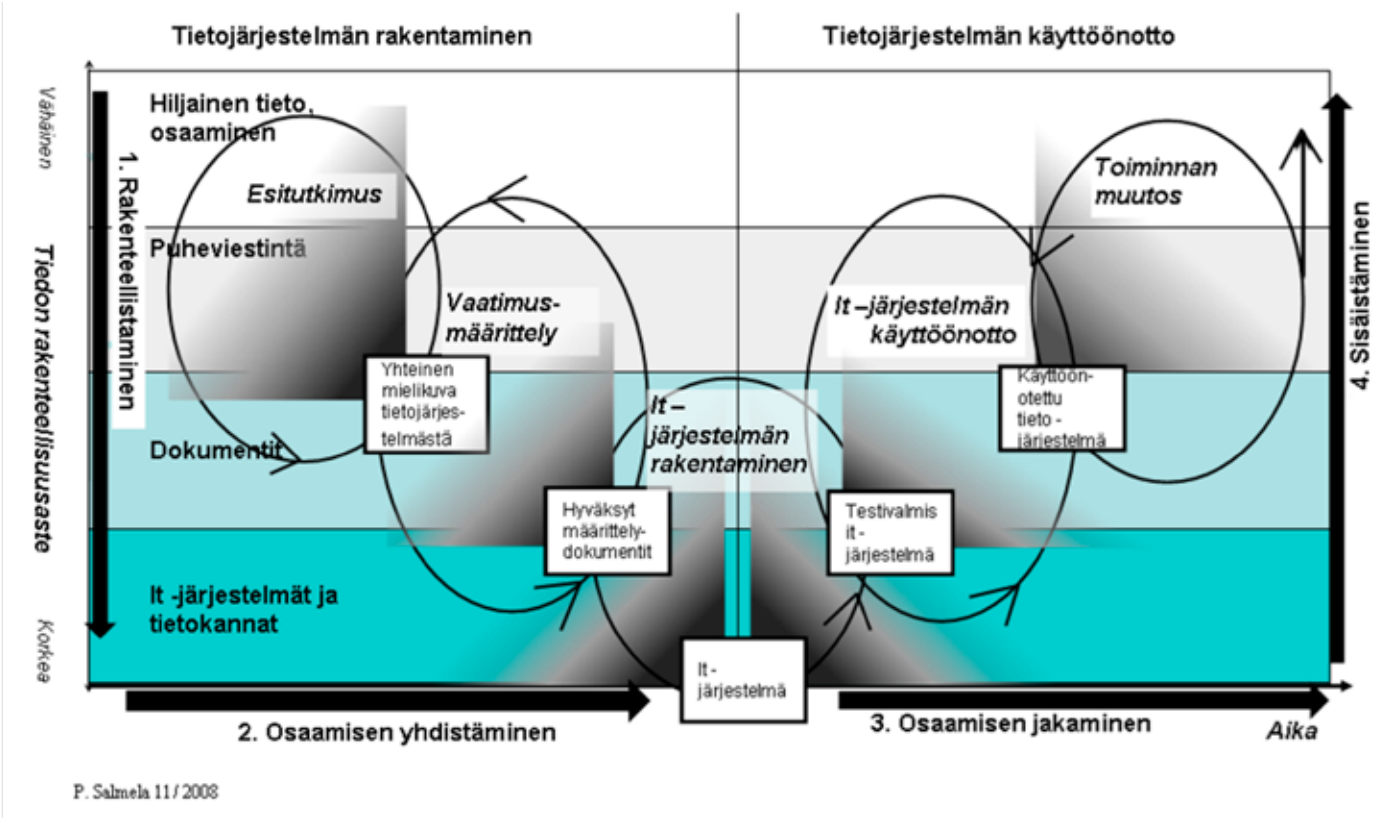
Kuvassa 3 on estetty tämä osaamisen yhdistämis- ja rakenteellistamisprosessi kaaviona. Kun it-järjestelmää rakennetaan, jokaisessa vaiheessa järjestelmä tarkentuu kahdesta suunnasta:

Rakenteellistamisen (tiedon suuremman järjestäytyneisyyden) kautta: Esitutkimusvaiheessa eritasoiset ja erilaiset mielikuvatasolla olevat mallit tarkennetaan keskeisiltä osin yhteiseksi malliksi, käsitykseksi rakennettavasta it-järjestelmästä. Vaatimusmäärittelyssä tätä mallia tarkennetaan dokumentoiduksi vaatimusluetteloksi ja it-järjestelmän rakentamisvaiheessa määrittelydokumenteissa kuvattu järjestelmä ohjelmoidaan tietokoneella toimivaksi ohjelmistoksi.

Osaamisen yhdistämisen kautta: Yhteistä mielikuvaa etsitään eri käyttäjätahojen, it-asiantuntijoiden ja johdon osaamista yhdistelemällä ja yhdentämällä. Tämä edellyttää yhteisiä keskusteluja, järjestelmän tavoitteiden ja riskien kartoitusta, kustannusarvioita ja mahdollisesti benchmark-vierailuja. Tilaaja- ja toimittajaorganisaatiot pyrkivät luomaan yhteisen käsityksen tilattavasta, rakennettavasta ja toimitettavasta it-järjestelmästä.

Käyttöönottovaiheessa it-järjestelmään tiivistetty osaaminen puretaan käyttäjäorganisaation käyttöön muuttuneeksi toimintatavaksi. Tämäkin tapahtuu kahdesta suunnasta:

Kuva 3.
It-järjestelmän rakentaminen hiljaista tietoa rakenteellistamalla ja osaamista yhdistämällä, tietojärjestelmän käyttöönotto osaamista jakamalla ja sisäistämällä toiminnan muutos.



Osaamisen jakamisen kautta: It-järjestelmään tiivistetty osaaminen jaetaan organisaation käyttöön käyttöohjeiden, käyttökoulutuksen, uuden järjestelmän käyttöönoton ja mallioppimisen kautta.

Sisäistämisen kautta: It-järjestelmään rakenteellistettu tieto puretaan ja sisäistetään järjestelmän käyttäjien hiljaiseksi tiedoksi. Tietojärjestelmällä tavoiteltu muutos sisäistetään organisaation muuttuneeksi toimintatavaksi.

Yhteenveto

Tietämyksen hallinta tuo it-projektin hallintaan uuden ulottuvuuden. Tulee hallita myös organisaation hiljaista tietoa, tiedon muunnoksia sekä ihmisen osaaminen ja it:n vuorovaikutusta. Esitetyssä projektimallissa it-järjestelmän vaiheita tarkastellaan kahdessa tasossa: Projekti etenee aika-akselilla ja samalla tapahtuu tiedon olomuodossa muunnoksia tiedon rakenteellisuusakselilla

Tietojärjestelmiä rakentamalla organisaatiossa olevaa hiljaista tietoa tiivistetään tieto- ja viestintätekniikalla käsiteltävään muotoon. Tietojärjestelmän kehittäminen on vaiheittain etenevä prosessi, jossa järjestelmä muuntuu mielikuvatason mallista toimivaksi ja käyttöönotetuksi it-järjestelmäksi. Rakentamisvaiheessa järjestelmän toiminta täsmentyy sekä osaamisen yhdistämisen että rakenteellistamisen suunnaista. Kun tietojärjestelmää otetaan käyttöön, it-järjestelmään rakenteellistettu toimintamalli puretaan organisaation uudeksi muuttuneeksi toimintatavaksi. Purkaminen tapahtuu it-järjestelmään tiivistettyä osaamista jakamalla ja sisäistämällä toiminnan muutos.

Vaikka tässä esityksessä kohteena on it-järjestelmän kehittäminen, mallissa esitetyt periaatteet ovat pitkälle yleispäteviä. Mitä tahansa kehittämistapahtumaa, joka perustuu ihmisten osaamisen ja tekniikan vuorovaikutukseen, voidaan tarkastella esitetyn mallin avulla. Etenemismalli soveltuu yhtä hyvin verkkotuotteiden kehittämis- tai innovaatioprojektien läpivientiin kuin tietojärjestelmien kehittämiseenkin.

Artikkelissa on käytetty seuraavia käsitteitä:

a) Systeemityö on tietosysteemin rakentamista, kehittämistä ja kunnossapitämistä.

b) Rakenteellinen tieto -termiä käytetään synonyyminä käsitteellisen eli eksplisiittisen (Explicit Knowledge) tiedon kanssa. Tämä tieto on muodollista, systemaattista ja tarkkaan määriteltyä. Sitä voidaan prosessoida ja tallentaa suhteellisen helposti samoin kuin viestiä ja jakaa. Se esitetään yleensä tieteellisten kaavojen, käyttöohjeiden, lomakkeiden ja tietokoneohjelmien muodossa.

c) Tieto-termiä käytetään hyvin yleisenä käsitteenä tarkoittamaan organisaatiossa olevan tiedon kaikkia muoto- ja - tietämystä, informaatiota, tietoa ja dataa. Termiin ei myöskään liity ”tosiuskomus ”tai ”tulkittu informaatio” -vaatimusta.

d) Keneth Craik (1943) esitti ajatuksen mentaalisista malleista. Hänen mukaansa ihmismieli rakentaa pienimuotoisia malleja, joilla se hahmottaa tapahtumia. Mentaaliset mallit rakennetaan havainnoista mielikuvituksen keinoin tai keskustelujen pohjalta. Mallit ovat pääasiassa visuaalisia mielikuvia tai abstrakteja.

e) Tietojärjestelmä koostuu tointaan hoitavista ihmisistä, toiminnan ohjeista, tietojen käsittely- ja siirtolaitteista sekä tiedoista.

f) It-järjestelmä on tietojärjestelmän it-tekniikkaan perustuva osakokonaisuus, joka tukee toimintaa ja sen tietojenkäsittelyä.

Lähteet

Craik, K. (1943). The nature of explanation, Cambridge: Cambridge University Press

Crawford J: Kent (2006). The Project Management Maturity Model, Information Systems Management, vol 23, Number 4.

Nonaka, I. & Takeuchi, H. (1995). The knowledge creating company: how Japanese companies create the dynamics of innovation. New York: Oxford University Press.

Polanyi, M. (1966). The tacit dimension. Garden City, N.Y.: Doubleday.

Stevens, N. D. (1986). The history of informations. Julkaisussa: Advances in Librarianship, vol 14, s. 1–48.

Ståhle, P. (1999). Kaaoksen hallinta vie menestykseen. Haastattelu Sonera Soundi - konsernilehdessä 1/ 99.

Ståhle, P. & Laento, K. (2000). Strateginen kumppanuus – avain uudistumiskykyyn ja ylivoimaan. Helsinki: WSOY.

Salmela, P. (2008). Hiljainen ja rakenteellistettu tieto asiantuntijaorganisaation toiminnan kehittämisessä, Informaatiotutkimus 27 (2008),

Salmela, P. (2008). It –projektit –uusi toimintatapa käyttöön hiljaista tietoa rakenteellistamalla, Projektitoiminta 2/2008, Projektiyhdistys ry:n jäsenlehti, Espoo

Sell, M. (1975). Projektin ohjaus, Tietojenkäsittelyliiton julkaisu no 31, Salo

Tietojenkäsittelyliitto (2002). Tietojärjestelmän hankinta, Talentum Media, Vantaa

Wikipedia, Tietotekniikan historia

KUUTAMOLLA

Pohjatöitä

Ohjelmiston kehitys alkaa aina jostain ideasta. Ja se päättyy siihen, kun ohjelmisto vaihdetaan uuteen tai otetaan muuten vain pois tuotannosta. Siinä välissä tapahtuu kaikenmaailman mielenkiintoisia asioita, usein ihan henkilötyövoimavuosikaupalla. Tämän elinkaaren alkuvaiheessa tehdään paljon päätöksiä. Nämä päätökset aika pitkälti vaikuttavat siihen, miten kauan se koko elinkaari tulee kestämään ja ennen kaikkea, paljonko se tulee maksamaan. Monastihan kalleus johtuu siitä, että alun virheitä joudutaan paikkaamaan sitten, kun ohjelmisto on jo käytössä. Eikä niitä tekovaiheessakaan ole halpaa korjata. Kyynisimmät ja omanarvontuntoisimmat ohjelmistokehitysprosessin alkuvaiheiden asiantuntijat ovatkin sitä mieltä, että heidän työnsä on tärkeintä koko prosessissa. He asettavat askelmerkit ja tekevät nuotit koko sille lopulle orkesterille, joka lähtee kiihdyttämään kohti ponnistuslautaa. He tekevät suurimman työn ja heidän käsistänsä päästämät tuotokset lähtevät liukuhihnalla ohjelmistotehtaaseen, jossa robotit vain tekevät sen työn, joka niille on ohjeistettu. Kaikki luovuus, suuret päätökset, valtava vastuu ja laatu on näiden aivojen aikaansaannosta. Useimmiten nämä samat tyypit ovat myös sitä mieltä, että heidän tästä työstänsä saama palkinto on aivan naurettavan liian pieni. Aivan kuin allekirjoittaneenkin, vaikkakin eri syistä, koska palkkanihan on lahjoihini verrattuna oikeastikin naurettavan pieni.

Määrittelytyö – kuten mikä tahansa ohjelmistokehitysprosessin aikainen työ – olisi hyvin paljon mukavampaa ja helpompaa, mikäli olemassaolevaa maailmaa ei olisi. Helppohan se oli Luojan tyhjästä luoda maailma, kun ei tarvinnut miettiä mitään ympäröivien parametrien mukanaan tuomia ongelmia. Mutta koittaisipa nyt luoda vaikkapa tähän kuun ja maan väliin uuden planeetan aatameineen ja eevoineen. Aurinkokunnan herkkä status quo vadis menisi sekaisin ja lopputuloksena voisi olla vaikka mitä hankalan lopullista. Ainakin kuutamosta sekaisin menevien ihmisten sekaisinmenot menisivät sekaisin. Ei se määrittelijälläkään kovin helppoa ole, kun pitää ensin ymmärtää mitä on sorkkimassa ja sitten vasta voi miettiä, mitä sinne sekasotkuun haluaisi lisätä. Ja sitten kun on omasta mielestään saanut asiat selkeiksi ja dokumentoitua seuraavaa vaihetta varten, niin eikös niitä aletakin heti retostella. ”Kyllä tämäkin olisi ollut järkevämpi tehdä toisella tavalla”, ”Onpas epäselkeästi tehtyä, tästä tulee kyllä kamala lopputulos, jos koskaan niin pitkälle päästään”, ”Ai tämmöstä, kuka pässi tällaista on käskenyt tehdä, no, täytyy yrittää, vaikkei tästä oikein mitään voi tehdä” ja muuta musertavan hapanta.

No, jokaisellahan meistä on oma risti kannettavana. Se, onko jonkun toisen työ tärkeämpää kuin toisen, on ikuisuuskysymys, johon ei kyllä kannata tämän palstan puitteissa lähteäkään. Varsinkin, kun artikkelitoimittajien työ on kuitenkin aina tärkeämpää. Ja kyllä kolumnistitkin joutuvat tekemään paljon pohjatyötä artikkeleidensa eteen. Minäkin joudun kärsimään asiattomista mistääntietämättömien besserwissereiden huomautuksista niin oman työni alkuvaiheissa kuin pitkin toteutusta ja usein vielä toteutuksen jälkeenkin ja eritoten silloin, kun tuotos on jo tuotannossa eli painettuna ja lukijoilla. Pohjatyö on kuitenkin se, mikä loppujen lopuksi pitkälti on ainakin suuressa roolissa sen suhteen, miten tehtävä kokonaisuus tulee paikkansa täyttämään. Jos siellä tehdään virheitä, niin kyllähän koko projekti lähtee etenemään väärään suuntaan. Käy niinkuin Mars Climate Orbiterille, avaruusluotaimelle, jonka oli tarkoitus tutkailla naapuriplaneettamme ilmakehää ja lomasäitä. Onnistuneen avaruuslennon jälkeen perillä Marsin kieppeillä luotaimen piti asettua kaikessa rauhassa puolentoistasadan kilometrin korkeudelle Marsin pinnasta tiirailemaan kelejä. Kuinka ollakaan, NASA:n yksi alihankkija olikin käyttänyt systeemissään vanhoja mittayksiköitä, paunoja, tuumia ja jalkoja, kun taas NASA erehdyksissään käyttikin metrijärjestelmän yksiköitä. Lopputuloksena luotain asettuikin vain 57 kilometrin korkeuteen, jossa kitka ja muut ilmakehälliset hankaluudet aiheuttivat loppujen lopuksi luotaimen putoamisen ja tuhoutumisen. Eikä siitä ole vielä edes kymmentä vuotta. Niin pienestä se joskus on kiinni. Ehdotankin, että tästä lähtien kaikki muistavat katselmointitilaisuuksissa kysäistä, mitkä yksiköt ovat käytössä. Jos puhutaan vuosista, niin muistakaa, että Pluton vuosi kestää melkein kaksi ja puoli sataa maan vuotta. Joskus projektit näyttävät kyllä olevan suunniteltuja Plutovuosien mukaan, ainakin jälkeenpäin.

Yrittäkää nyt kuitenkin tästäkin artikkelista huolimatta jaksaa pitää kiinni tinkimättömästä laadusta ja työmoraalista ja muistakaa ennen kaikkea olla mukana vuosituhannen merkittävimmässä tilaisuudessa, Sytyke ry:n 30-vuotisjuhlaseminaarissa Finlandia-talolla, nähdään siellä!

SYTYKE ry on vuodesta 1979 toiminut valtakunnallinen systeemityöntekijöiden ammatillinen yhdistys, joka kehittää alan ammattilaisten välistä yhteistyötä ja tutkimustoimintaa.

Teemayhdistyksen jäseneksi voivat liittyä kaikki systeemityöstä kiinnostuneet yksityiset henkilöt, yhdistykset ja yritykset. SYTYKE ry:n toiminta-alueena on koko Suomi. SYTYKE on Tietotekniikan liitto Ry:n jäsenyhdistys.

Lisätietoja SYTYKE ry:stä: www.sytyke.org

TOIMISTO

Susanna Koskinen
Systeemityöyhdistys Sytyke ry
Talvikkitie 40 A 33
01300 Vantaa
p. 09 56075363
f. 09 56075365
sytyke@hennax.fi

Hallitus 2009

Puheenjohtaja
Mitro Kivinen
p. +358 40 589 2724
mitro.kivinen@kolumbus.fi

Seppo Takanen
p. +358 50 581 0140
seppo.takanen@codebakers.fi

Marianne Malila
malila68@gmail.com

Juha Jääskinen
juha.jaaskinen@digia.com

Timo Sundman
timo.sundman@tietoenator.com

Tiina Kiuru
tiina.kiuru@ri.fi

Jyri Partanen
jyri.partanen@sulake.com

Varajäsenet

Ilkka Pirttimaa
p. +358 50 389 0022
ilkka.pirttimaa@stockmann.fi

Altti Lagstedt
altti.lagstedt@haaga-helia.fi

Liittokokous-edustajat

Minna Oksanen
minna.oksanen@gmail.com

Seppo Takanen
seppo.takanen@codebakers.fi

Lauri Laitinen
lauri.laitinen@nokia.com

SYTYKE ry:n hallituksen sähköpostilista:

hallitus@sytyke.ttlry.fi

Sytyttääkö? - Liity jäseneksi

Systeemityöyhdistyksen jäseneksi liitytään Tietotekniikan liiton kautta (http://www.ttlry.fi/, 020 741 9898, jasenasiat@ttlry.fi) valitsemalla jäsenyhdistykseksi Systeemityöyhdistys ry. Nykyinen Tietotekniikan liiton jäsen voi liittyä joko vaihtamalla jäsenyhdistystä tai liittymällä lisäjäseneksi.

Tietotekniikan liiton henkilöjäsenmaksu vuonna 2009 on alkaen 49€ ilman lehtiä, 1 lehti 63€. Erityisryhmien hinnoittelusta lisätietoja Tietotekniikan liitosta. Lisäjäsenyys maksaa 12€/yhdistys.

Osaamisyhteisöt 2009

Systeemityöyhdistyksessä toimitaan niin yhdistystasolla kuin aihepiireittäin erikoistuneissa osaamisyhteisöissä. Monipuolisessa tarjonnassamme löytyy jokaiselle jotakin. Vaihtoehtona on myös perustaa omalle kiinnostukselleen uusi osaamisyhteisö - SYTYKE-hallitus toivottaa toimintaehdotukset tervetulleeksi. Osaamisyhteisön toimintaan pääset mukaan laittamalla postia vetäjälle.

JavaSIG on Javan käyttäjien ja harrastajien intressiryhmä, vetäjänä Simo Vuorinen.
simo.vuorinen
@oracle.com

ProjektiOSY - PrOSY pyrkii yhdistämään Systeemityön projektitoiminnasta ja sen kehittämisestä kiinnostuneet, vetäjänä Petteri Puurunen.
petteri.puurunen@tieto.com

TestausOSY - FAST on testauksen keskustelu- ja yhteistyöverkosto, vetäjänä Maaret Pyhäjärvi.
maaret.pyhajarvi@iki.fi

DAMA Finland keskittyy tiedon, informaation ja tietämyksen hallintaan, yhteyshenkilönä Minna Oksanen.
minna.oksanen@gmail.com

ViestintäOSY järjestää yhteistoimintaa viestintäsovellusten alueella, vetäjänä Tapani Ranta.
tapani.ranta@generum.fi

RELA keskittyy relaatiotietokantoihin vetäjänään Lauri Pietarinen.
lauri.pietarinen
@relational-consulting.com

MallinnusOSY jakaa tietoa tietojärjestelmien mallintamisesta, vetäjänä Juha Jääskinen.
juha.jaaskinen@digia.com

KäytettävyysOSY vaihtaa kokemuksia käytettävyyden kehittämiskeinoista sekä kehittää omaa käytettävyysosaamistaan. Vetäjänä ja yhteyshenkilönä Helena Venäläinen.
helena.venalainen@op.fi

SOA SIG on keskittynyt palveluarkkitehtuuriin (Service-Oriented Architecture, SOA). Vetäjänä Janne J. Korhonen.
jjk@jannekorhonen.fi

KAOS on kokonaisarkkitehtuurin osaamisyhteisö. Toimintaa suunnittelee perustamiskokouksessa valittu isännistö, yhteyshenkilönä Pasi Mäkinen.
pasim@microsoft.com

Perusteellista pohjatyötä

Korkeinkin torni alkaa maasta. Torni pysyy pystyssä, kun perustukset on rakennettu lujaksi. Sama pätee tietojärjestelmissä. Pohjatyö lähtee prosesseista ja liiketoiminnan asettamista vaatimuksista. Hyvin suunniteltu arkkitehtuuri pitää tietojärjestelmän kunnossa tuulisissakin olosuhteissa. Muutokset viedään projekteissa menestyksellisesti läpi hyvällä suunnittelulla, toteutuksella ja testauksella. Tee kunnollinen pohjatyö ja tule kurssille.

Onnistu ostossa, varmistu vaatimuksista

IT-sopimukset	3.–4.6.
Tietojärjestelmän hallittu hankintaprosessi	5.6.
Prosessien mallintaminen	18.–20.5. • 19.–21.8.
ARIS prosessien mallintamisessa	15.–16.6.
Vaatimusten määrittely ja hallinta	23.–25.6. • 31.8.–2.9.
Testaus järjestelmän tilaajalle	26.5. • 13.8.
Tietojärjestelmän tehokas käyttöönotto	12.6. • 14.9.

Ansioidu arkkitehtina, suunnittele sulavasti

Arkkitehtuurin suunnittelu (UML)	2.–3.6. • 17.–18.8.
Yritysarkkitehtuurin suunnittelu	25.–26.5. • 3.–4.9.
TOGAF Certification Course	22.–25.6.
Järjestelmäintegroinnin valmennusohjelma	8.–9.6. alkaen
BPM pähkinänkuoressa	4.6.
SOA-palveluiden määrittely ja suunnittelu	4.–5.6. • 20.–21.8.
SOA ja Web Services yleiskatsaus	12.6.
Määrittely ja suunnittelu (UML)	15.–17.6. • 2.–4.9.
.NET-arkkitehtuuri	11.6.

Pätevöidy projektipäällikkönä

Tietojärjestelmäprojektin suunnittelu ja läpivienti	10.–12.6.
Ketterän ohjelmistoprojektin läpivienti	8.–10.6.
.NET-järjestelmän projektipäällikkövalmennus	12.–14.5.
Java-järjestelmän projektipäällikkövalmennus	18.–20.5. • 24.–26.8.
Java-projektipäällikkövalmennus 2: ketteryys, tuottavuus ja tuloksellisuus	15.–16.6.
Certified ScrumMaster	11.–12.6.
Project Leadership	17.–18.6.

Testaa taitavasti

Testauksen valmennusohjelma	1.–2.6. • 17.–18.8.
ISEB ISTQB Foundation Certificate	15.–17.6. • 7.–9.9.
ISEB Intermediate Certificate in Software Testing	8.–10.6. • 31.8.–2.9.
Test Driven Development Workshop (TDD, .NET, Java)	22.–23.6.
Järjestelmän suorituskyvyn testaus	22.–23.6.

Ajankohtaisseminaarit

ExcelLent Controller	9.–10.6.
Talousviestintä	11.6.
eMarketing – sähköisen markkinoinnin valmennusohjelma	22.–24.9.

PS: tarkista myös, onko yrityksellä jo koulutussopimus Tieturin kanssa!

Tieturi

KAIKKI KURSSIT: WWW.TIETURI.FI

ilmoittautumiset@tieturi.fi | puh. 09 4315 5333 | tekstiviesti 040 353 1000