Mitä uutta auringon alla?

Osallistuin viime joulukuussa Java-kurssille. Mukana oli myös joukko, ainakin vielä silloin, arvostetun ja menestyvänkin e-busines -yrityksen työntekijöitä. Kaikki olivat, nuoria heillä oli kovan tason tekninen tietämys, opinnot kesken Otaniemessä ja he olivat töissä enemmän kuin täyspäiväisesti.

Kysyin lounastauolla, millaisilla suunnittelumalleilla ja menettelyllä te näitä ohjelmianne ja järjestelmiänne nykyisin kehittelette? Vastauksena oli ensin ällistynyt hiljaisuus ja niin sanottu hidas katse. Lopulta todettiin, että menetelmillä? malleilla? Ei me mitään menetelmiä tarvita. Markkinoinnin kundit tulevat kertomaan mitä asiakkaat toivoo ja sitten me alamme tehdä.

Ei mitään uutta auringon alla. Samantapainen oli omankin työpaikkani epävirallinen suunnitteluperiaate 70- luvulla kun aloitin työuraani. ”Kuppi kuumaa ja koodaamaan” se kuului.

No, nämä olivat kärjistettyjä esimerkkejä. Maassamme on tehty erittäin korkeatasoista, kurinalaista ja standardoitua systeemityötä vuosikausia. Tulokset myös näkyvät moderneina ja palvelukykyisinä järjestelminä. Emme vain huomaa niitä koska kaikki toimii hyvin.

Tämän lehden teema sai alkunsa jo vuoden 1999 Sytykkeen Tukholman –risteilyllä. Keskustelimme yhdistyksen silloisen puheenjohtajan Silja Räisäsen kanssa systeemityön tilasta ja siitä minkälaista kehitystä on tapahtunut. Tuon keskustelun perusteella syntyi työryhmä, joka teki Sytykkeen vuoden 2000 jäsenkyselyn. Siinä tiedustelimme systeemityön haasteita ja pullonkauloja.

Aihe ja työryhmän saama palaute olivat niin tärkeitä ja työ niin mielenkiintoista, että päätettiin perustaa Sujuvan Systeemityön jatkotyöryhmä. Sen tehtäväksi otettiin selvittää systeemityön ajankohtaisia virtauksia ja niitä asioita joilla työ saadaan sujumaan. Tavoitteeksi asetettiin tämän lehden yhden numeron toimittaminen.

Artikkelien kirjoittajat edustavat monipuolista ja laajaa kokemusta. Useimmilla on takanaan jo pitkä ura. Itse asiassa niin pitkä, että palavereissa välillä muisteltiin miten ohjelmia tehtiin reikäkorteille ja miten optimoitiin sortteja.

Olemme pyrkineet käsittelemään systeemityön elinkaarta ja sen hallinnointia. Artikkeleissa käsitellään myös systeemityötä eri osapuolten näkökulmasta:

- johdon,
- työryhmien ja
- yksilön kannalta

Kirjoittajat edustavat omia henkilökohtaisia näkemyksiään emmekä väitä, että tässä lehdessä olisi esitetty ainutkertainen totuus. Toiveena on herättää Sytykkeessä keskustelua teemasta. Koska useimmat kirjoittajat ovat pitkän linjan kulkijoita kaipaan joukkoon myös nuoremman polven kommentteja ja mielipiteitä.

Jukka T Virtanen,
johdon konsultti,
Deltrain Oy,
jukka@deltrain.fi,
040-525 1840

Strategiasta hankkeiksi ja projekteiksi

Jukka T Virtanen,
Deltrain Oy

Koska yrityksillä yleensä on rahaa ja henkilöstöä rajallisesti, hankkeet ja projektit pitää asettaa tärkeysjärjestykseen. Siksi tarvitaan järjestelmä. Järjestelmän on oltava sellainen että hankkeista saa kokonaiskuvan ja järjestelmän on tuotettava mahdollisimman ajantasaista ja luotettavaa faktatietoa työn edistymisestä ja vaadituista resursseista. Vain näin johto voi tehdä tarvittavat päätökset tärkeysjärjestyksestä ja resurssien käytöstä ja siten varmistaa, että hankkeet toteuttavat asetettuja liiketoiminnan strategioita ja tavoitteita.

Menettelytavat ja tarvittavat tietojärjestelmät ovat olemassa. Suurin haaste on se miten ne saataisiin käyttöön.

1 Näkökulma

Tarkastelen tässä kirjoituksessa yritysstrategian ja hanke/projektisalkun hallinnan yhteyttä pääasiassa johdon ja kokonaissuunnittelun näkökulmasta. Tyypillisesti on kyse isohkosta organisaatiosta tai konsernista jossa on useita kymmeniä hankkeita ja projekteja joissa työskentelee satoja tai jopa tuhansia ihmisiä.

2 Määritelmiä

Kuvaan tässä jaksossa niitä elementtejä jotka liittyvät liiketoiminnan ja hankehallinnan johtamiskokonaisuuteen.

Toiminta-ajatus: Miksi olemme olemassa
Visio: Missä haluamme olla 2-5 vuoden kuluttua
Tavoitteet: Mitä meidän on tehtävä toteuttaaksemme visiomme
Strategia: Miten aiomme saavuttaa tavoitteemme
Hanke: Joukko toimia (Program: Set of activities implementing a strategy), joilla toteutamme tavoitteemme
Projekti: Hankkeen osa (Activity), jolla on omat tavoitteensa ja vastuuhenkilönsä, projektipäällikkö.

2.1 Toiminta-ajatus, visiot , strategiat ja tavoitteet

Tämä esitys lähtee siitä, että yrityksellä on kattava liiketoimintastrategia jota ylläpidetään säännöllisesti (vähintään vuosittain ja aina tarvittaessa) ja jossa on kuvattu yrityksen tavoitteet ja kehittämisalueet.

2.2 Hankkeiden ja projektien sekä operatiivisen toiminnan suhde

Tyypillinen hankkeelle on iso kehittämiskokonaisuus, joka muodostuu sekä liiketoiminnan- että tietotekniikan kehittämisprojektista/projekteista ja mahdollisesti vielä muusta projektoimattomasta työstä.

2.2.1 Hankkeet ja projektit

Hankkeet ja projektit kannattaa luokitella niiden strategisuuden mukaan. Yksittäinen projektikin voi olla yritykselle strateginen.

Hanke tai projektit on strateginen jos se on:

- Erittäin kallis
- Epäonnistuminen vaikuttaa merkittävästi yrityksen tulokseen tai koko yritystoiminnan jatkuvuus on kiinni hankkeen onnistumisesta

Muut projektit

Pienkehittäminen / ylläpito on toimintaa jota yleensä jatkuvasti tehdään ja se on työn ja resurssien hallinnassa huomioitava.

2.2.2 Operatiivinen toiminta

Useimmissa organisaatiossa päivittäinen operatiivinen toiminta tuottaa liiketoiminnan tuloksen ja vie suuren osan kehitysprojekteissa mukana olevien työajasta. Kehityshankkeita suunniteltaessa tämä päivittäinen kuormitus on otettava huomioon.

Eräässä konsultoimassani yrityksessä tietotekniikka tuotantovastaavia haluttiin kiinnittää moniin projekteihin. Tekijöitä oli vähän ja tarvittava asiantuntemus oli vain muutamalla henkilöllä. Syntyi tilanne jossa moni projektipäällikkö kuvitteli voivansa käyttää yhtaikaa samoja henkilöitä lähes täyspäiväisesti. Kun suunniteltu työmäärät selvitettiin, todettiin ettei monelle olisi edes 24 tunnin työpäivä riittänyt puhumattakaan, että he olisivat pystyneet hoitamaan päivittäisen päätyönsä, jonka sujuminen kuitenkin on yrityksen elinehto.

3 Toiminnan kokonaiskuva

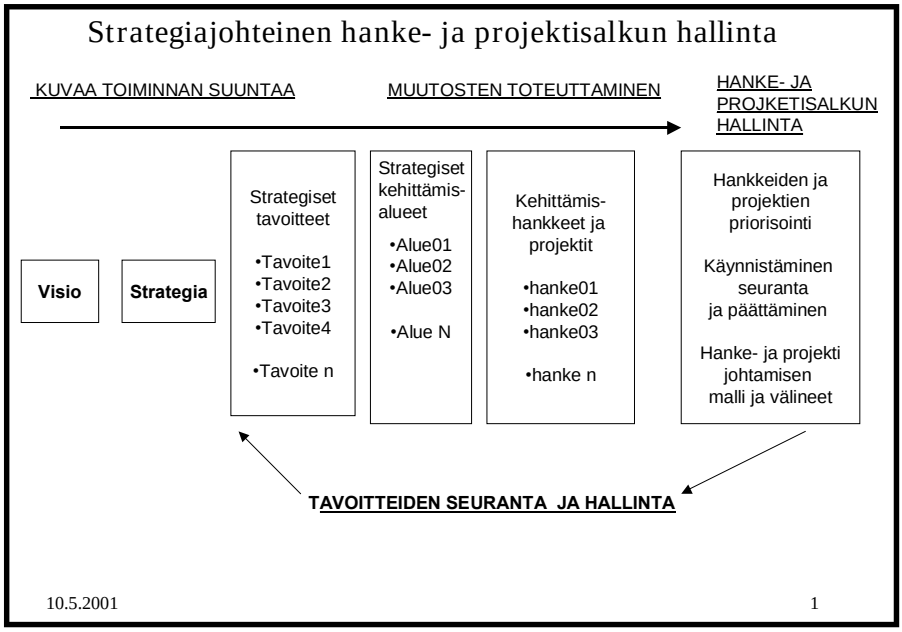
Oheinen kuva esittää hanke- ja projektisalkun hallinnan keskeiset elementit ja niiden suhteet. Kts kuva seuraavalla sivulla.

Liiketoiminnan suunnittelu kertoo mihin ja millä keinoin yritys on menossa. Muutokset kohdistuvat strategisille kehittämisalueille ja niitä toteutetaan kehittämishankkeilla ja projekteilla. Hankkeiden ja projektien hallinta edellyttää yhteisiä pelisääntöjä. Näitä pelisääntöjä kuvaavat sovitut menettelytavat ja mallit, joilla:

- asetetaan tärkeysjärjestykseen
- resurssoidaan
- ohjataan

hanke- ja projektitoimintaa sekä seurataan tavoitteiden toteutumista ja tehdään tarvittaessa korjaavia toimenpiteitä.

Nämä prosessit on tyypillisesti iteratiivisia koska sekä ympäristötekijät että yrityksen sisäiset tekijät voivat muuttua nopeasti. Johtamiseen kuuluu toiminnan ja tavoitteiden seuranta. Entistä useammin johtamiseen myös liittyy ja pi-



täisi liittyä, yrityksen kannustejärjestelmä.

4 Hankehallinnan ja projektijohtamisen malli ja välineet

4.1 Menettelytapa ja mallit

Menettelytapa kuvaa sen dokumentointi- ja päätösprosessin, jonka yksittäinen hanke ja projekti käy läpi koko elinkaarensa aikana. Menettelytapa antaa yhdenmukaiset mallit työn suunnittelu- ja hallintadokumenteille. Lisäksi kuvataan kuka tekee ja vastaa mistäkin asiasta ja päätöksestä. Menettelytavan tulee ohjata ja sitoa niin yritysjohtoa kuin yksittäisestä hankkeesta vastuussa olevaa henkilöäkin. Vain omalla sitoutumisellaan yritysjohto mahdollistaa koko organisaation sitoutumisen yhtenäiseen toimintamalliin.

4.2 Hankehallinnan tietojärjestelmä

Monia hankkeita ja projekteja hallittaessa tarvitaan tietojärjestelmää, joka kokoaa tiedot yhteen ja antaa mahdollisuuden muodostaa kokonaiskuva. Pelkkä Järjestelmä ei kuitenkaan riitä vaan sen käyttöä koordinoimaan ja tukemaan tarvitaan oma vastuuyksikkönsä, hanke/projektitoimisto.

Hanke- ja projektihallinnan ohjelmistoja löytyy valmispaketteina niin kotimaisilta kuin kansainvälisiltäkin toimittajilta. Niillä on tietenkin valmispakettien hyvät ja huonot puolet ja usein ne vaativat muokkaamista ja parametrointia kun niitä sovitetaan yrityksen toimintatapaan. Yritys voi tietenkin myös itse rakentaa järjestelmänsä mutta se on useimmiten liian raskas ratkaisu. Pakettiohjelmistolla voidaan saavuttaa suurin osa hyödyistä.

Ohjelmistoa valittaessa on syytä ottaa huomoon eri käyttäjäryhmien tarpeet. Projektityön ammattilainen, joka suunnittelee yksittäistä projektia vaatii eri asioita kuin ylin johto joka on kiinnostunut kokonaiskuvasta, rahasta, aikataulusta ja tavoitteista.

Järjestelmä tulee tukee eri käyttäjäryhmien tarpeita:

Johto	**Hanke/projektipäälliköt**	**Resurssit**
Helppokäyttöinen	**Vahvat analyysiominaisuudet: aika, kustannukset, resurssit**	**Helppokäyttöinen**
Hyvät yhteenveto- ja esitysominaisuudet	**Raporttien ja näkymien monipuolinen muokkaaminen**	**Toteutumatietojen syöttö helppoa**
Mahdollisuus Top-Down suunnitteluun	**Monipuoliset projektin suunnittelu- ja seuranta-ominaisuudet**	**Graafisuus**

5 Strategian ja hankehallinnan linkki

Kunnollista hanke- ja projektinhallinnan tietojärjestelmän avulla on varsin yksinkertaista luokitella suunnitteilla ja työn alla olevat hankkeet ja projektit. Järjestelmän tulosteiden, yhteenvetojen ja näkyminen avulla on sitten helppo tarkastella ja seurata eri luokkiin kuuluvia hankkeita.

Ainakin seuraavat tulee määritellä niin, että hankkeesta tai projektista tiedetään jo kun sitä ehdotetaan toteutettavaksi.

- onko se strateginen?
- mihin tavoitteeseen se kuuluu?
- mihin kehittämisalueeseen se kuuluu ?
- missä vaiheessa se on? (ehdotettu, toteutuksessa ja niin edelleen)
- kuka omistaa hankkeen?

Kun kaikki hankkeet ja projektit on kuvatulla tavalla luokiteltu, järjestelmän keskitetystä tietovarastosta voidaan tulostaa esimerkiksi:

- raportti kaikista strategisista hankkeista (kustannukset, tavoitteet, aikataulut ja vaaditut resurssit)
- tai
- vastaava raportti tietyn organisaatioyksikön omistamista hankkeista
- tai
- operatiivisen työn vaatimista resursseista (henkilöt ja kustannukset) edellyttäen tietysti, että operatiivinen

työkin on kuvattu ja resursoitu järjestelmään.
- tai
- muilla erilaisilla haku- ja kategoriointitekijöillä muodostettuja raportteja

Samojen yhteenvetoraporttien kokoaminen perinteisellä tavalla eri yksiköiden ja eri henkilöiden mahdollisesti ylläpitämistä suunnittelumapeista ja erilaisista irrallisista projektinhallintaohjelmistoista on niin iso työ, että siihen tuskin missään isommassa organisaatiossa ryhdytään. Kerran se vielä onnistuu mutta viikko- tai kuukausitason seuranta on käytännössä mahdotonta.

Kun tällaisia luokitteluja tehdään, tulisi kaikkien hankkeiden ja projektien löytyä omasta luokastaan. Jos näin ei käy, on syytä kysyä miksi sellaisiin hankkeisiin on ylipäätään ryhdytty.

6 Hyödyt

Mitä hyötyä kuvaamastani hanke/projektisalkun hallinnan järjestelmästä on?

Järjestelmä on:

Strategian ohjausväline

Järjestelmä helpottaa yrityksen johtamista ja antaa johdolle selkeän yleiskuvan, joka perustuu liiketoiminnan vaatimuksiin ja mahdollisimman ajantasaisiin faktoihin.

Antaa mahdollisuuden käyttää resurssit tehokkaasti

Tietotekniikka-alalla on ollut pulaa jo kauan osaavista tekijöistä, mutta viime aikoina on nähty, että rahaakin on vain rajallisesti. Kehittyneillä hanke ja projektihallinnan järjestelmillä havaitaan helposti henkilöresurssien yli- ja alikuormitustilanteet jo suunnitteluvaiheessa. Samoin on helppo tehdä kustannusarvioita niin yksittäisistä hankkeista kuin erilaisista kokonaisuuksistakin.

Ympäristön muutosten vaikutusten arviointi helpottuu

Kun ympäristössä tapahtuu merkittäviä muutoksia on helppo tehdä uusia arvioita kuten esimerkiksi: kuinka monta uutta henkilöä tarvitsemme yllättävän lainmuutoksen vuoksi vai vapautammeko voimia keskeyttämällä jonkin hankkeen.

7 Haaste, yrityksen kulttuurin sopeuttaminen järjestelmään

Pelkkä kehittyneen hanke/projektihallinnan ohjelmiston käyttöönotto ei takaa suuria hyötyjä. Tarvitaan enemmän.

Pitää kuvata aiemmin mainitut yhteiset pelisäännöt ja toiminnan prosessit sekä perustaa tarvittava tukiorganisaatio.

Tämäkään ei vielä riitä vaan hyötyjen saamiseksi koko järjestelmä ja toimintatapa pitää jalkauttaa organisaatioon. Tämä vaihe on yllättävän raskas ja aikaa vievä. Muutos uuteen on perinteisesti tuskallinen.

Kuten aina strategisissa hankkeissa, ylimmän johdon sitoutuminen ja tuki on ratkaisevaa.

Mitä isompi organisaatio niin sitä vaikeampaa on luopua reviiriajattelusta ja tietojen panttaamisesta. Kaikilla organisaatiotasoilla keksitään helposti syitä miksi vanha systeemi oli parempi ja miksi uusii on niin vaikea ja aikaa vievä. Usein koetaan hyvin pelottavana se, että muutkin saavat nähdä miten hankkeet tai projektit edistyvät. Varsinkin, kun ainakin karkea seuranta on mahdollista ilman vastuuhenkilön selityksiä.

8 Lopuksi

Tässä artikkelissa on käsitelty tilannetta lähinnä suurten hanke- ja projektikokonaisuuksien kannalta. Olisi kuitenkin mielenkiintoista keskustella myös siitä kuinka tällaisia ajatuksia voidaan soveltaa pienemmissä organisaatioissa.

Keskitetyt hankesalkun hallinta ohjelmistot ovat kalliita tai ainakin tulevat kalliiksi, joten pienemmille yrityksille voisi olla tarpeellinen kevennetty malli. Millainen se voisi olla.?

Lähteet:

David I Cleland ja William R. King 1998: Project Management Handbook, Van Nostrand Reihold, New York.

Gerry Johnson ja Kevan Scholes 1988: Exploring Corporate Strategy, Prentice Hall.

IT-Johdon ja liikkeenjohdon toimiva vuorovaikutus –tilaisuudessa pidetyt esitykset.

Jukka T Virtanen,
johdon konsultti,
Deltrain Oy
jukka@deltrain Oy
040-525 1814

Liiketoimintahyötyjen toteutuminen IT-projekteissa on onnenkauppa. Onko TietoEEM ratkaisu ongelmaan?

Markku Laulajainen,
TietoEnator Oyj

IT- investoinnit muodostavat merkittävän osan organisaatioiden investoinneista. Ei ole mitätön kysymys, mitä tuottoja näistä sijoituksista saadaan. IT- investointi on temppu vaan ei tulos. IT-projektille ei riitä, että se tuottaa halutun lopputuotteen optimaalisilla resursseilla. Sen on tuotettava myös haluttu vaikutus liiketoimintaan. Pelkkiä IT- investoiteja ei enää ole vaan investoinnit kohdistuvat toiminnan muuttamiseen. Miten ohjaamme investointimme tulosta? TietoEnatorissa on kehitetty yhteistyössä Teknillisen korkeakoulun kanssa liiketoimintahyötyjen suunnittelu- ja ohjaamismenettely, TietoEEM. Se auttaa luomaan organisaation yhteisen vision tavoiteltavista vaikutuksista sekä prosessiin, IT- elementteihin kuin ihmisten osaamiseenkin liittyvistä kehittämistarpeista. Se ei erottele "E-liiketoimintaa" perinteisistä ratkaisuista, vaan kytkee ne yhteen. Onko TietoEEM IT-tuotannon"The Missing Link"?

Yritykset käyttävät huomattavan osan investoinneistaan tietotekniikan ylläpitoon ja hankintaan. Yli 40 % kaikista elinkeinoelämän investoinneista suuntautuu tietokoneisiin, ohjelmistoihin sekä telekommunikaatioon.

Kansainväliset tarkastelut (mm. Robert Solow, Paul Strassman,..) IT-investointivolyymien ja erilaisten taloudellisten tunnuslukujen välillä eivät ole osoittaneet selvää korrelaatiota. Ilmiötä kutsutaan tuottavuusparadoksiksi. Sen selittämiseen on pantu paljon panoksia. Ilmiön esittäminen voi tuntua pessimismiltä, sillä esim. prosessi-, pankki-sekä telekommunikaatioliiketoiminnassa jo pelkkä datan käsittelyn automatisointi on synnyttänyt kustannussäästöjä "investoi tai kuole"-hengessä. Kun tarkastelemme informaation hallinnan tuomia mahdollisuuksia tehdä asiat eri tavalla sekä koko liiketoiminnan muuttamista, IT:n tuottamat hyödyt liiketoiminnalle on vaikeammin tunnistettavissa. IT:n vaikutusmekanismia liiketoimintaan ei tunneta.

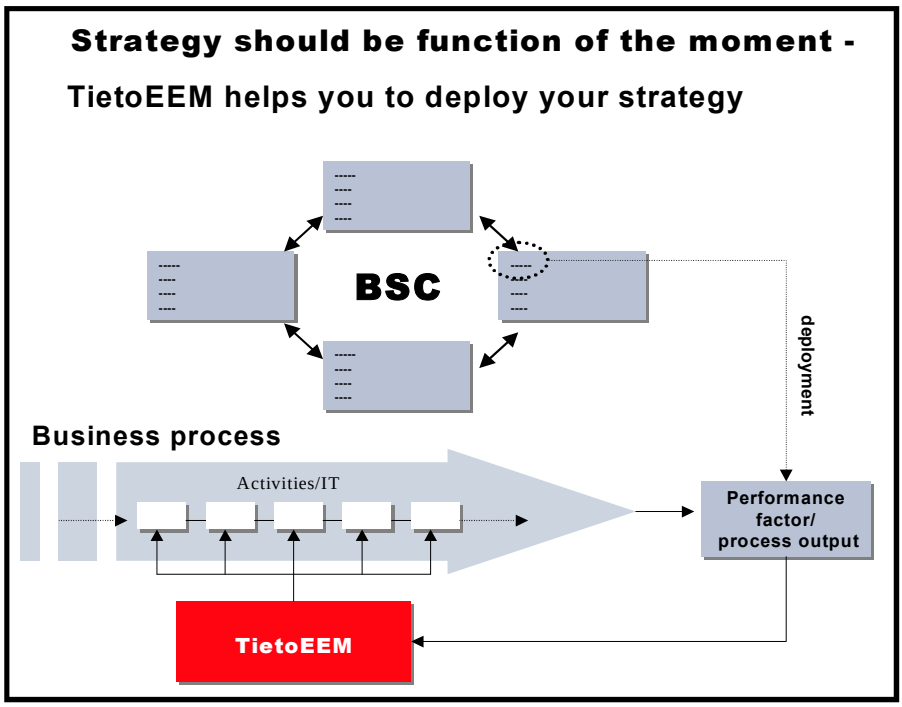
Tuottavuusparadoksille on lukuisia selityksiä. Osa niistä on "akateemisia", tässä "maalaisjärjellä" ymmärrettäviä:

- Taloudelliset tulosmittarit eivät tavoita IT:n aikaansaamia hyötyjä, koska tulokseen vaikuttavat samanaikaisesti lukuisat IT:stä riippumattomat tekijät, ts. koko "markkinamekanismi". Vaikutukset onkin mitattava operatiivisina tunnuslukuina yksikön tai prosessin tasolla.
- IT:tä ei osata käyttää tyydyttämään liiketoiminnan tarpeita. Kyse ei ole niinkään teknisestä kykenemättömyydestä vaan kykenemättömyydestä tunnistaa liiketoiminnan todelliset tarpeet sekä nähdä uudet mahdollisuudet omassa toiminnassa. IT-investoinnilla tavoiteltavia vaikutuksia liiketoimintaan ei ole yleensä edes määritetty. Jos ne on kuvattu, niiden toteutumisen seuranta on sattumanvaraista.
- IT:aa, prosesseja, organisaatiota ja ihmisten osaamista on kehitetty toisistaan riippumattomina ilmiöinä, ei systeeminä. Pelkkiä IT- projekteja ei ole. IT- ratkaisut vaikuttavat myös tapaan toimia aiheuttaen muutoksia prosesseihin sekä osaamiseen. Vastuu hyödyistä kuuluu liiketoiminnan johdolle ja prosessin omistajille ei IT-johdolle eikä IT- asiantuntijoille.

Asiakastarveanalyysi on osoittautunut erääksi ohjelmistotuotantoprosessin heikoimmaksi lenkiksi. Tavoiteltavia liiketoimintavaikutuksia ei toimituksissa yleensä ole kuvattu. Painopiste on tekemisen, ei tuloksen ohjaamisessa. Ilman liiketoimintalähtöistä asiakastarveanalyysiä IT- toimituksen eri vaiheet ei ole verifioitavissa eikä koko toimitus validoitavissa. Tällöin ohjelmistotuotantoprosessista ja toimituksesta puuttuu ydin. Lisäksi tarveanalyysin tulisi olla luova ja innovaativinen osaprosessi, joka poikkeaa muuten kurinalaisesta tuotantoprosessista.

Informaatio on "sähköisen liiketoiminnan " perusta. Kuitenkin varsin vähälle huomiolle informaatiotalouden rakenteissa ja IT- toimituksissa on jäänyt informaation keskeiset laatuominaisuudet:

- luotettavuus
- validisuus
- tehokkuus.



Kuva: TieetoEEM auttaa strategioiden maastouttamisessa.

TietoEEM auttaa strategioiden maastouttamisessa

TietoEEM (Enabler- Effect Map) on uusi ajattelumalli ja menettely ylläkuvatun ilmiön ratkaisemiseksi. Se mahdollistaa organisaation eri osaamisalueiden välisen dialogin. Tämän tuloksena syntyy yhteinen ymmärrys siitä mitä liiketoimintavaikutuksia kehittämisinvestoinnilla haetaan sekä mitä pitäisi tehdä niiden aikaansaamiseksi.

TietoEEM:n avulla pyritään varmistamaan, että liiketoimintaympäristön todelliset tarpeet ohjaavat kehittämisinvestointeja ja että IT- ratkaisun, prosessien, organisaation sekä ihmisten osaamisen kehittämistä ohjataan samanaikaisesti toimituksen eri vaiheissa.

Tavoitteena on strategioden maastouttaminen muuntamalla ne tarvittaviksi prosessien elementeiksi. Näin esim. Balanced Scorecard tunnussuureisiin liittyvät tavoitteet toimeenpannaan prosessin, IT- arkkitehtuurin ja ohjelmistojen ominaisuuksina.

Tavoitellut liiketoiminnan vaikutukset toteutuvat uuden käyttöönotetun prosessiversion kautta. IT ei yksin tee mitään. Bittien lisäksi tarvitaan lähes aina molekyylejä haluttujen hyötyjen aikaansaamiseksi.

TietoEEM luo edellytykset prosessin ja sitä tukevien sovellusten koko elinkaaren aikaisen kehittämisen ohjaamiselle sekä toimittajayhteistyön jatkuvuudelle. Menetelmän käyttö on jatkuvaa toimintaa aivan kuten toiminnan kehittäminenkin.

TietoEEM on kehitetty Tieto-Enatorissa yhteistyössä Teknillisen korkeakoulun, Soneran sekä Suomen Postin kanssa pilottiprojekteja käyttäen. Se perustuu Teknillisessä korkeakoulussa kehitettyyn EEM- teoriaan. Teoria on kuvattu teoksessa Holopainen, Lillrank & Paavola:”Tietotekniikan linkki liiketoimintaan, Otava 1999”.

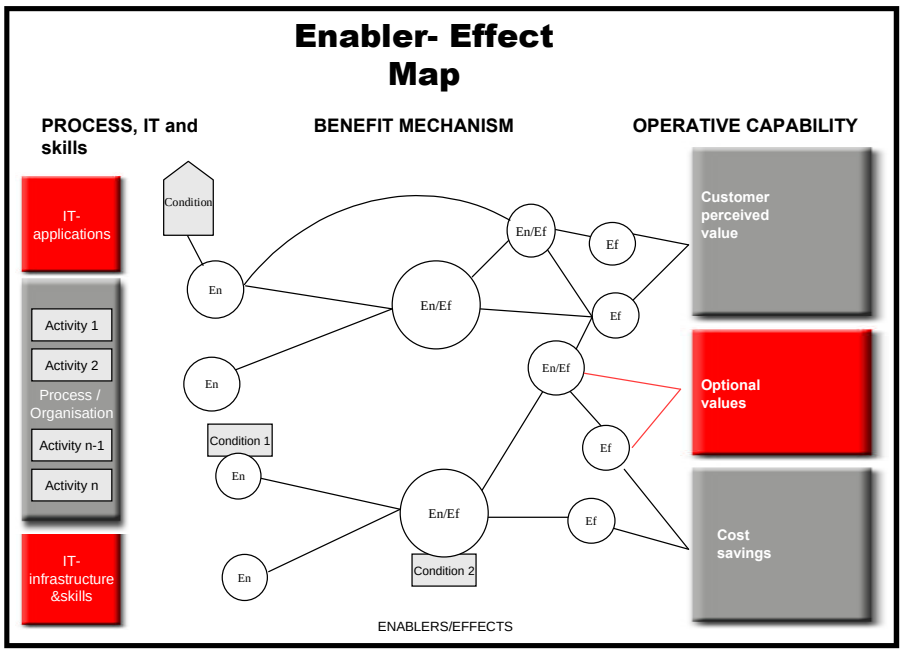
TietoEEM:llä ohjataan asiakasarvoa, kustannustehokkuutta sekä optioarvoja

TietoEEM koostuu lomakepohjista sekä hyötykartoista (Enabler- Effect Map, EEM). Lomakkeille kootaan liiketoiminnan todelliset tarpeet ts. kehittämistä ohjaavat tavoiteltavat vaikutukset. Ne ovat strategiasta, ydinosaamis- ja asiakastarpeesta johdettuja suorituskykytekijöitä, ilmiöitä, joilla voi olla tavoitearvo sekä IT- tuotteeseen liittyviä teknisiä vaatimuksia.

Kartta kuvaa ns. mahdollistaja-vaikutusketjujen avulla IT:aan, prosessiin, organisaatioon sekä henkilöstön osaamiseen liittyvät elementit, jotka yhdessä mahdollistavat tavoitellut vaikutukset. Tapauskohtaiset polut kartalla, ”hyötyjen realisointiprosessit”, kertovat kuinka IT:n käyttö osana muuta toiminnan kehittämistä saa aikaan liiketoiminnallisia hyötyjä. Strategian, tavoiteltavien vaikutusten sekä kartan ylläpitäminen on jatkuvaa toimintaa.

TietoEEM ohjaa liiketoiminta-alueen tai prosessin operatiivista kyvykkyyttä, joka liiketoiminnalliselta merkitykseltään voi olla strategista. Tavoiteltavat vaikutukset jakautuvat kolmeen luokkaan: prosessin asiakkaan kokemaan arvoon, prosessin kustannussäästöihin sekä optioarvoihin. Asiakkaan kokema arvo kuvaa kuinka paljon asiakasarvoa prosessi tuottaa. Se liittyy joko toimitteeseen tai palveluprosessiin. Kustannussäästöt kuvaavat kuinka tehokkaasti prosessi toimii. Kustannussäästöt sisältävät henkilöstö- ja pääomakustannusten lisäksi myös laadunpuute- l. ”hukkakustannukset”. Optioarvot kuvaavat kuinka paljon strategista potentiaalia löytyy. Optioarvot vastaavat ”rahamaailman” optioita. Ne voidaan lunastaa suotuisissa olosuhteissa joko asiakkaan arvona, kustannussäästöinä tai strategian muuttamisena ja aivan uutena liiketoimintana. Tällöin edellytetään strategista oivallusta hyödyntää prosessia ja IT:aa uudella, toimialalle ennen kokemattomalla tavalla.

EEM-kartan päätepiste ei tarkoita yrityksen lopullista tulosta. ”Operatiivisesta tuloksesta” on pitkä matka tuloslaskelman viimeiselle riville. Avoimessa markkinataloudessa tulokseen vaikuttavia muuttujia on lukuisia, joten prosessikehitys- ja IT-investoinnin vaikutusta liiketulokseen on käytännössä mahdotonta määrittää. Hyvin toteutettu prosessinkehitys- ja IT-hanke ei pelasta huonosti hoidettua liiketoimintaa.



Kuva: Hyötykartta koostuu mahdollistajista ja vaikutuksista.

TietoEEM:n käyttö hyödyttää kaikkia osapuolia

TietoEnatorissa TietoEEM:n käytön keskeinen päämäärä on asiakkaan arvontuotannon osaaminen ja yhteisen asiakkuusprosessin jatkuva hallinta niin, että molemmat osapuolet hyötyvät. Menetelmän käyttämisestä hyötyvät niin asiakkaan kuin toimittajankin puolelta mm. liiketoiminnan johtajat, tietohallintajohtajat, account managerit, kehitysjohtajat, markkinointihenkilöstö, konsultit, projektipäälliköt sekä toimitusten vastuuhenkilöt.

Asiakas pyrkii ostopäätöstä tehdessään maksimoimaan hyödyn l. nettoarvonsa. Hyötykartta kuvaa tavoiteltavat vaikutukset sekä niiden mahdollistajat l. tarvittavat panokset. Nämä luovat perustan investointipäätökselle. Hyötykartan kustannussäästöt ja tuottavuuden kasvu vastaavat investoinnin tuotto-odotuksia. Asiakkaan kokema arvo sekä optioarvot sensijaan ovat uusia elementtejä investointipäätöksen perusteiksi.

TietoEEM vahvistaa ohjelmistotuotannossa laadun asiakasnäkökulmaa. Painopiste on ollut tuotanto- sekä tuotekeskeisessä menettelyssä. Asiakas kuitenkin punnitsee jatkuvasti tavoiteltavaa hyötyä l. laatua hintaa ja haittoja vasten. Toimittaja puolestaan seuraa laadun tuottamisen ja kustannusten välistä suhdetta. Laatu ja hinta ovat täten ikuisia asioita ohjelmistotuotannossakin. Laadunvarmistuksen painopiste siirtyy tavoitellun hyödyn varmistamiseen mallitemppuihin vertaamisesta.

TietoEEM on modulaarinen menetelmä. Se on riippumaton toimituksessa sovellettavasta ohjelmisto- tai ylläpitotuotannon rakentamismallista sekä projektinhallintamenettelystä. Projektitoimituksissa hyötydokumentaatio syntyy ja tarkentuu tarjous-, sopimus-, asiakastarve- analyysi- sekä määrittelyvaiheiden yhteydessä muodostaen perustan ylläpitotoimituksille.

Ylläpitotoimituksissa hyötydokumentaatio laaditaan aina ylläpitosuunnitelman laatimisen yhteydessä.

TietoEEM noudattaa yleisesti käytössä olevaa QFD (Quality Function Deployment)- tuotekehitysperiaatetta. TietoEEM:ssä on QFD:n monitasoiset ”what and how”- matriisit pelkistetty yhteen tasoon. TietoEEM:ää voidaan käyttää sekä asikastarpeen hallintaan että tuotteen sisältämien komponenttien innovointiin. Se on käyttökelpoinen myöskin erillisten tuotteiden muodostaman tuotekokoonpanon hallintaan.

TietoEEM on käyttökelpoinen myös organisaation koko projektisalkun hyötylähtöisessä hallinnassa sekä projektien priorisoinnissa.

IT-tuotannon lisäksi menetelmä soveltuu muihinkin toiminnan muutostilanteisiin. Se on käyttökelpoinen johtamisen, ideoinnin sekä dokumentoinnin menettely erilaisissa vaikutusmahdollistaja l.mitä-miten tilanteissa. TietoEEM on hyödyllinen, kun on tarve erottaa tulos tempusta!

Markku Laulajainen,
Johdon konsultti,
TietoEnator Oyj,
Markku.laulajainen@tietoenator.com,
puh. 040 5274230

Uljas, uusi, JOUSTAVIEN PROSESSIEN maailma

Kristian Rautiainen ja Maaret Pyhäjärvi, TKK/Ohjelmistoliiketoiminnan ja –tuotannon laboratorio

Viime vuosina kiinnostus joustavampiin metodologioihin (englanniksi *agile* tai *lightweight methodologies*) on kasvanut nopeasti ohjelmistoprosessien yhteydessä. Kun tänä päivänä käy ohjelmistotuotannon konferensseissa, puhe ei enää pyöri pelkästään CMM:n tai ISO:n eri standardien ympärillä, vaan yllättävän moni kyselee kokemuksista ja näkemyksistä joustavampiin prosesseihin, etenkin näkyvimpään niistä, eXtreme Programming (XP).

Joustavien prosessien ideana on tuoteen mahdollisimman nopea sopeuttaminen muuttuviin olosuhteisiin ja vaatimuksiin tuotteen kehittämisen aikana. Erityisen hyvin ne siis soveltuvat Internet-maailmaan, jossa muutosnopeus, sekä liiketoiminnan että teknologian osalta on suuri ja toisaalta tuotteen tekemiseen käytettävissä oleva aika käy yhä pienemmäksi.

Joustava prosessi on tyypillisesti iteratiivinen, eli se perustuu nopeisiin, usein toistuviin, intensiivisiin ja systemaattisiin iteraatioihin, jotka alkavat hyvinkin aikaisessa vaiheessa prosessia ja osallistavat myös asiakkaan. Tällainen toimintatapa asettaa kovat vaatimukset tuotearkkitehtuurille, tuotetta kehittävälle tiimille, sekä asiakkaalle. Ja vastoin yleistä käsitystä joustava prosessi ei ole lupa hakkeroida (”koodataan ja korjataan”) unohtaen kaiken muun, eikä saman asian uudelleen suunnittelemista, kunnes uskotaan sen toimivan. Itse asiassa joustavat prosessit vaativat hyvin paljon kurinalaisuutta, siinä missä ”perinteisemmät” prosessitkin. Erot nousevat ehkä selvimmin esille dokumentoinnin luonteessa ja määrässä, sekä henkilökohtaisen kommunikoinnin korostamisessa. Joustavia prosesseja onkin luonnehdittu ”ihmisläheisiksi”.

XP (eXtreme Programming) on, jos ei tunnetuin niin ainakin viime aikoina näkyvin joustavista prosesseista. Se on tarkoitettu pienille (2-10 henkilöä) tiimeille, kuten monet muutkin joustavista prosesseista. Näin halutaan varmistaa tiivis, henkilökohtainen kommunikointi. XP:n soveltuvuudesta eri asioihin ollaan montaa mieltä ja keskusteluissa se yleensä herättää vahvojakin tunteita, sekä myönteisiä että kielteisiä. XP:n luoja, Kent Beck, käytti sitä ensimmäisen kerran laajemmassa yhteydessä 1996 Chryslerilla C3 palkanlaskenta-järjestelmäprojektissa.

XP käsittää neljä arvoa: kommunikointi, palaute, yksinkertaisuus ja rohkeus. Näiden arvojen perusteella on luotu 12 toisiaan täydentävää käytäntöä, joita pitäisi noudattaa:

- Suunnittelupeli – Yhdistämällä liiketoiminnallisia prioriteettejä ja teknisiä arvioita määritetään seuraavan julkaisun laajuus. Realismin iskiessä suunnitelmaa päivitetään. Asiakas tekee päätökset ja hänen on oltava jatkuvasti tavattavissa.
- Pienet julkaisut – Ensimmäinen, pienimuotoinen, toimiva järjestelmä julkaistaan mahdollisimman nopeasti, minkä jälkeen julkaistaan uusia versioita hyvin lyhyissä iteraatioissa (viikosta kuukauteen).
- Metafora – Ohjaa kaikkea kehitystä jaetulla tarinalla systeemin toiminnasta.
- Yksinkertainen rakenne – Järjestelmän pitäisi aina olla mahdollisimman yksinkertainen. Ylimääräinen kompleksisuus poistetaan heti havaittaessa. Esimerkiksi jos sama logiikka toteutetaan kahdessa eri paikassa on ne yhdistettävä ainoastaan yhteen paikkaan.
- Testaus – Ohjelmoijat kirjoittavat yksikkötestejä jatkuvasti, Yksikkötestin täytyy mennä virheettömästi läpi kehitystyön jatkamiseksi. Asiakkaat kirjoittavat testit jotka osoittavat ominaisuuksien olevan valmiita (hyväksymistestit).
- Uudelleenfaktorointi – Ohjelmoijat muotoilevat järjestelmää uusiksi muuttamatta sen toimintaa poistaen samaa asiaa toteuttavaa koodia, yksinkertaistaen, lisäten joustavuutta ja parantaen kommunikaatiota.
- Pariohjelmointi – Kaikki tuotantoon menevä koodi kirjoitetaan kahden ohjelmoijan yhteisvoimin yhteisellä koneella. Toinen kirjoittaa koodia ja toinen käytännössä katselmoi koodia koko ajan. Roolia voidaan vaihtaa usein saman päivän aikana.
- Kollektiivinen omistajuus – Kuka tahansa voi muuttaa mitä tahansa osaa koodista koska tahansa. Yksikkötestien ajolla nähdään heti rikkoiko muutos koodin.
- Jatkuva integrointi – Integroi ja rakenna järjestelmä monta kertaa päivässä, aina kun tehtävä on suoritettu.
- 40-tuntinen viikko – Sääntönä töitä tehdään vain 40 tuntia viikossa. Ylitöitä ei saa olla kahtena peräkkäisenä viikkona.
- Asiakkaan läsnäolo – Sisällytä tiimiin aito käyttäjä, joka on täysipäiväisesti saavutettavissa mahdollisten kysymysten ilmetessä.
- Koodauskäytännöt – Ohjelmoijat kirjoittavat kaiken koodin sovittujen käytäntöjen mukaisesti kommunikaatiota painottaen.

XP:tä tarkemmin katsoessa havaitsee, ettei joustava prosessi ehkä olekaan niin kevyt. Prosessiin on sisällytetty peruselementit äärimmilleen vietyinä ja hauskaan muotoon puettuina. Monet käytännöt ovat vanhoja, hyviksi havaittuja käytäntöjä, joista osa ehkä on jo ehtinyt unohtuakin vuosien varrella. Arkkitehtuurisuunnittelun puutetta on kritisoitu, mutta itse asiassa se sisältyy metaforaan ja uudelleenfaktorointiin.

Erityisen äärimmilleen vietyä on testaus. Ajatusmaailman mukaan ominaisuutta ei ole olemassa ennen kuin sille on tehty automatisoidut testit. Tämän avulla pyritään saavuttamaan vakaa poh-

ja jolle rakentaa, koska muutosten jälkeen testit voidaan jälleen ajaa ja todeta järjestelmän toimivuus siltä kannalta. Tämä toimintatapa on melko raskas ja aikaa vievä, sillä testit on pidettävä jatkuvasti ajan tasalla.

Pariohjelmointi on mielenkiintoinen käytäntö, jolla haetaan nopeutta ohjelmointiin laadun kärsimättä. Samalla varmistetaan myös oppiminen ja se, ettei vain yksi henkilö osaa koodia, mikä voi olla suuri riskitekijä. Maailmalla on tehty muutama pienimuotoinen koe, joissa lähinnä pariohjelmointikäytäntöä on verrattu tavalliseen ohjelmointiin. Vaikka tulokset ovat epäluotettavia, näyttää siltä, että pariohjelmoinnilla saa nopeammin tuloksia aikaan, mutta sen tehokkuus on huonompi, koska siinä on kaksi henkilöä kiinni. Toisaalta missään kokeessa koodia ei ole katselmoitu muiden käytäntöjen kohdalla ja XP:ssä tämä tulee ikäänkuin kaupan päälle. Erään workshopin yhteydessä osallistujat kertoivat käyttävänsä pariohjelmointia silloin tällöin lyhyissä jaksoissa, etenkin kun pitää saada jokin asia toimimaan nopeasti.

TKK:n olio-ohjelmointikurssilla opiskelijat tekivät pienimuotoisen harjoitustyön, jossa pyrittiin käyttämään osaa XP:n käytännöistä. Harjoitustyö oli niin pieni, ettei sen perusteella voi vetää mitään vahvoja johtopäätöksiä. Yksi havainto oli kuitenkin, että XP on prosessina hyvinkin haastava ja varmasti vaatii suorittajaltaan melkoisesti kokemusta ja näkemystä sekä ajatuksien syvempää ymmärtämistä tehokkuuden saavuttamiseksi. Suunnitelmissa on lukuvuoden 2001-2002 aikana tehdä laajempi koe, jossa katsotaan tarkemmin joitakin XP:n vahvuuksia ja heikkouksia johonkin muuhun toimintatapaan verrattuna.

XP:n lisäksi on olemassa muitakin joustavia prosesseja, joihin kannattaa tutustua, jos asia vähänkään kiinnostaa. Monet ajatukset vaikuttavat hyviltä ja ainakin ne voivat toimia inspiraationa prosessien kehittämiselle. Ehkä paras paikka lähteä liikkeelle on Martin Fowlerin artikkeli, joka löytyy WWW:stä:

http://www.martinfowler.com/articles/newMethodology.html

Siinä kerrotaan joustavien prosessien ajatusmaailmasta sekä esitellään eri prosesseja lyhyesti. Jokaisen prosessin kohdalla on linkit ja referenssit, joiden avulla löytää syventävää tietoa.

Mukavaa lukemista!

Kristian Rautiainen ja
Maaret Pyhäjärvi
TKK/Ohjelmistoliiketoiminnan ja –tuotannon laboratorio
(http://soberit.hut.fi/),
SEMS-projekti
(http://soberit.hut.fi/sems/)

Jälleen tulossa

SYTYKKEEN PERINTEINEN LAIVASEMINAARI

5.-7.9.2001

Aiheita: Oliot ja komponentit

Tarkempia tietoja ohjelmasta ja järjestelyistä aikatauluineen tiedotetaan jäsenistölle alkusyksystä.

Kipin kapin aika kalenteriin!

Nopeuksien tarkkailua tehostettava!

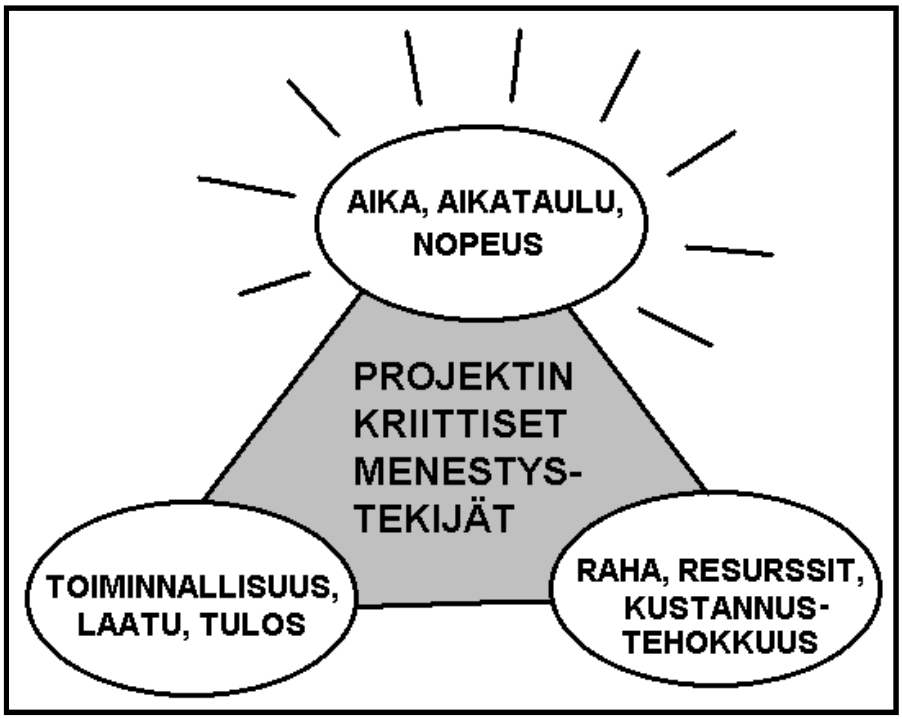
Havaintoja suomalaisen projektityön sujuvuuden tiimoilta

Tom Weckström,
Lifix Systems Oy ja
Pekka Forselius,
STTF Oy

Automatisointi ja sen myötä ohjelmistot ovat ehdottomasti yksi tärkeimpiä menestyksellisen liiketoiminnan mahdollistajia tämän päivän maailmassa. Ohjelmistojen merkitys kasvaa yhä, ja yhä suuremmat osat globaalista markkinataloudesta tulevat entistä riippuvaisemmiksi ohjelmistojen moitteettomasta toiminnasta ja aikataulun mukaisista toimituksista. Julkishallinnossa vannotaan usein kustannustehokkuuden nimiin ja tietyillä turvallisuuskriittisillä toimialoilla laatu on kaikki kaikessa, mutta yleisesti ottaen toimitusnopeudesta on tullut tärkein vaatimus ohjelmistott ttttttttttttttt tt tttttt ttttttttt kaikkein kriittisin menestystekijä niin uuden talouden startup firmoille kuin tiukkaa keskinäistä kilpailua käyville isoille ohjelmistotuottajillekin. Näin yritysten edustajat ainakin väittävät.

Onko aika todella noussut tai nousemassa projektikolmion kahden muun ulottuvuuden, toiminnallisuuden ja resurssien, yläpuolelle? Mitkä ovat tämän kehityskulun syyt ja seuraukset? Miksi ohjelmistoprojektit kestävät niin kauan kuin ne kestävät? Miten niitä voisi nopeuttaa? Miten sinun yrityksesi projektien toimitusnopeutta mitataan ja parannetaan?

Tosielämässä tavoiteaikataulunsa ja kustannusarvionsa jopa sadoilla prosenteilla ylittäneet ohjelmistoprojektit ovat hyvin yleisiä. Tässäpä suutarin kenkien veroinen dilemma: ohjelmistot tehostavat kaikissa suhteissa liiketoimintaa, mutta ohjelmistojen kehitys on riskialtista ja kontrolloimatonta. Yrityksiä on jopa mennyt konkurssiin myöhästyneiden ohjelmistonkehitysprojektien tai epäonnistuneiden ohjelmistojen käyttöönottoprojektien vuoksi.



Järjestelmällinen nopeuksien valvonta ja mittaaminen

Ohjelmistotuotanto on vähitellen muuttumassa taiteesta tieteeksi. Tai teollisuudeksi – ihan kuten haluatte. Viimeisten kahdenkymmennen vuoden aikana ohjelmistotuotannon ja ohjelmistoprojektien hallinnan tieteellistämiseksi on tehty melkoinen määrä tutkimustyötä. Pikkuhiljaa metriikat ja prosessit kehittyvät. Eräs tärkeä askel tässä kehitystyössä on ollut prosessien mitattavuuden kehitys ja mittaustulosten “sisäänsyöttö” takaisin prosessiin sekä mahdollisuus korjaaviin toimenpiteisiin mittaustulosten perusteella. Kehittyneet mittausmenetelmät voivat auttaa myös nopeuttamaan tuotantoa ja vähentämään riskejä. Mittaus toimii prosessin aistina ympäröivästä ja käsillä olevasta tilanteesta ja mahdollistaa siten reaktiivisen toiminnan, aivan kuten hyvä näkö antaa autoilijalle mehdollisuuden painaa jarrua heti, kun hän näkee hirven tiellä. Ohjelmistokehityksen prosessien mittaaminen ja mittaustulosten kerääminen eivät vielä kuitenkaan riitä — on opittava ymmärtämään sekä mittauksen perusteita että mittaustuloksia. Miksi projekti myöhästyi kahdella kuukaudella? Yksi kysymys synnyttää toisen. Hiljalleen poraudutaan ongelman ytimeen.

Mittaustulosten tutkiminen ja analysointi

Ohjelmistoprojektien tutkimus on ollut Suomessa aktiivista jo kymmenen vuoden ajan. Aikanaan Laturi-projekti ja viimeisten viiden vuoden aikana FiSMA (Finnish Software Metrics Association) ovat uurtaneet uraa kokoamalla runsaasti sekä parhaiden ohjelmistoyrityksissä työskentelevien asiantuntijoiden tietämystä että tietoa todellisista ohjelmistoprojekteista. FiSMAn kautta on myös avautunut hyviä yhteistyökanavia kansainvälisiin tutkimuslaitoksiin ja korkeakouluihin. Täällä kerättyjen ohjelmistoprojektien tuottavuuteen, ohjelmistojen koon mittaamiseen ja muihin vastaaviin ilmiöihin liittyviä asioita on tutkittu ja tuloksia julkaistu hyvinkin paljon. Esimerkiksi Barbara Kitchenham, Martin Shepperd, Isabella Wieczorek, Ross Jeffrey, M. Lee, K. Maxwell ja P. Forselius ovat julkaisseet artikkeleita, joissa suomalainen projektitietokanta on ollut tärkeässä asemassa. Kaksi viimeksimainittua ovat tutkineet ohjelmistotuotannon tuottavuutta sekä tuottavuuden ja projektin tilannetekijöiden korrelaatioita eri toimialoilla. Koska tietokantaan talletetut kokonaistyömäärät ja ohjelmistojen koot toimintopisteinä ovat hyvin huolellisesti kerättyjä, ovat myös löydetyt tulokset osoittautuneet mielenkiintoisiksi ja hyödyllisiksi.

Päinvastoin kuin tuottavuutta, projektien ajallista kestoa kalenteriajassa on tutkittu toistaiseksi melko vähän. Ajallisen keston riippuvuudet muihin, kenties tuntemattomiinkin projektiparametreihin, ovat vielä hämärän peitossa. Intuitiivisesti voimme päätellä jotain, mutta mittauksia ja analyysiä varten tarvitaan konkreettiset muuttujat ja mittaustuloksia. Projektin kalenterikeston ja muiden projektin muuttujien väliset riippuvuudet voivat osoittautua hyvinkin monimutkaisiksi ja ominaisuuksiltaan dynaamisiksi. Miten projektikolmion ulottuvuuksien välillä sukkuloidaan menestyksekkäästi? Aikataulupaineiden kasvaessa tarvitsemme työmääräarvioinnin, rakenteellisen suunnittelun ja kompleksisuusanalyysin rinnalle ymmärtämystä myös projektien aikatauluista ja niihin vaikuttavista muuttujista.

Tuntemattomilla vesillä on paras tehdä huolellisia esitutkimuksia ennen sukeltamista. Samalla voi kuitenkin kokeilla jotakin poikkeuksellistakin, koska vanhat konstit eivät ole tuoneet riittävästi tietoa ihan itsestään. Perinteisen korrelaatioanalyysin lisäksi jotkin uudenaikaiset tiedon analysointimetodit voisivat auttaa projektien aikataulutietojen ymmärtämisessä ja nopeassa hahmottamisessa. Tämä oli lähtöolettamuksenamme lähtiessämme tutkimaan ohjelmistoprojektien aikatauluja itseorganisoituvalla kartalla (Self-Organizing Map, SOM). Tutkimus toteutettiin Tom Weckströmin erikoistyönä Teknillisessä Korkeakoulussa. Erikoistyön tavoitteena oli tutkia projektien keston ja muiden projekteista kerättyjen muuttujien riippuvuuksia sekä projektien klusteroitumista erilaisiin ryhmiin. Tutkimuksellisena päämääränä oli lisäksi tarkoitus selvittää, miten projektidataa voi ylipäätään tutkia SOMilla.

Seuraavassa kappaleessa Tomin oma kertomus erikoistyöstä ja sen keskeisistä tuloksista:

Ensisukellus Experience-tietokantaan nopeusnäkökulmasta

”Käytettävissäni oli Experience Pron projektitietokannan otos, joka sisälsi 379 projektia vuosilta 1982 - 1999. Projektidata oli äärimmäisen moniuloitteista, eli kerättyjä muuttujia oli paljon. Muuttujien kokonaismäärä oli 48. Kukin projekti oli siis SOM-ajattelumallilla yksi 48-ulotteisen avaruuden piste. SOM pyrkii kuvaamaan tätä projektipisteiden joukkoa mahdollisimman hyvin ja samalla pelkistämään projektipilven informaatiota kuitenkin säilyttämällä oleellisen tiedon projektien välisistä suhteista, esim. kahden projektin samankaltaisuudesta annettujen muuttujien suhteen. Tuloksena SOM -algoritmin käytöstä on projektipilven esitys itseorganisotuvana karttana. Karttapinta ikäänkuin taipuu projektijoukon ylle kuvaten sen yleisimpiä ominaisuuksia. SOMin vahvuus on sen tietomallissa ja mallin sisältämän tiedon visualisointimahdollisuuksissa.

Projektidataa oli käytettävissä Euroopan suurimman projektitietokannan verran. Ongelmana tutkimuksessa oli projektidatan aukkoisuus. Monesta projektista puuttui oleellisia aikatauluun liittyviä parametreja ja usein myös niiden parametrien arvoja, joita halusin verrata projektin kestoon. Tutkin projektijoukoa kahdessa osassa: ennen vuotta 1995 olleet projektit ja vuoden 1995 jälkeen päättyneet projektit. Keskityin 156 vuoden 1995 jälkeiseen projektiin kahdesta syystä: niiden tiedot olivat täydellisemmät ja tutkimuksen tulokset palvelevat paremmin teollisuutta, kun tulokset on saatu viimeaikaisista projekteista.

Tarkempi tutkimusmenetelmän esittely, projektidatan analyysi ja tulokset ovat luettavissa erikoistyöraportistani (Investigating Software Project Data with Self-Organizing Map, Tom Weckström, 2001).

Huomioita

Tiivistelmänä tutkimukseni löydöksistä todettakoon: Projektien kalenterikeston ja muiden projektimuuttujien, kuten tilanneanalyysin muuttujien välillä ei ole selviä korrelaatioita. Toisaalta projektidata ei kuitenkaan ole klusteroitunutta. Tämä tarkoittaa sitä, että esimerkiksi eri teollisuudenalojen ohjelmistoprojektit tai eri ohjelmointikielillä toteutetut projektit jakautuvat keskenään melko tasaisesti projektiavaruuteen. Eri yritysten projektien sijoittuminen tasaisesti itseorganisoituneelle kartalle osoittaa, että Experience Pro on projektitiedonkeruumallina ja projektinhallinan apuvälineenä toimiva konsepti. Jos projektit klusteroituisivat voimakkaasti, Experience Pron kaltainen malli todennäköisesti epäonnistuisi projektien parametrien estimoinnissa.

Toisaalta Forselius, Lee ja Maxwell ovat kukin tutkimuksissaan huomanneet, että yksi muuttuja selittää yli 40% projektidatan hajonnasta. Tämä muuttuja on yritys. On siis huomattavasti helpompaa tutkia yhden yrityksen projekteja, koska yritystekijän vaikutuksen arvailun sijaan voidaan keskittyä projektin oleellisten parametrien ja projektin keston välisten suhteiden selvittämiseen.”

Tomin heittämiä haasteita ja vinkkejä ohjelmistoyritysten tuotekehitys- ja projektinhallintaorganisaatioille

”Jos todella haluatte tulla nopeammiksi ja tehokkaammiksi, kerätkää projektitietoja yrityksenne sisällä, mitatkaa, analysoikaa ja mahdollistakaa siten jatkuva oppiminen yhä nopeammaksi yhä laadukkaampien ohjelmistojen toimittajaksi. Tähän työhön liittyy myös jatkuva mittareiden ymmärtämyksen kehitystyö. Jokaisessa yrityksessä on varmasti erityisiä mittareita, jotka yleispätevien mittareiden lisäksi tukevat projektidatan analyysiä.

SOMia voi käyttää yrityksen sisäisenkin projektitutkimuksen apuna, kun data on lähes täydellistä, eli kaikki parametrit on tunnollisesti merkitty muistiin projektin aikana ja sen jälkeen. Lisäksi projektidataa on oltava kohtuullisen paljon, esimerkiksi ainakin 150 - 200 projektin verran, SOM -analyysiä varten.

Kun projektidataa kerran on ryhdytty yrityksessä keräämään, sen voi vallan mainiosti luovuttaa myös laajempaa tutkimusta varten Experience Pro -tietokantaan, että esimerkiksi laajemmat SOMilla tehtävät projektidatan analyysit olisivat mahdollisia. Yhdessä on mahdollista oppia huomattavasti nopeammin kuin erikseen, ainakin tärkeimmistä ja yleisimmistä ilmiöistä.

Toiveenani on, että kerätty projektidata palvelisi yrityksen sisällä strategisen ja operatiivisen päätöksenteon tukena, antaisi mahdollisuuksia todistaa numeroilla intuition herättämiä kysymyksiä ja väitteitä ja ohjaisi kohti tehokkaampaa ohjelmistotuotantoa ja yhä nopeampia projekteja sekä yritysten ja yliopistojen välillä universaalien ohjelmistotuotannon lainalaisuuksien etsinnässä ja *muussakin tutkimustyössä.”*

Summa summarum

Kuten yllä olevat ajatukset paljastavat, todellista tietoa ohjelmistotoimitusten nopeuteen vaikuttavista seikoista ei ole vielä läheskään tarpeeksi. Toimiva tiedonkeruukonsepti sitä vastoin on, joskin tähän mennessä kerätty tieto saattaa olla juuri nopeuden tarkkailun kannalta puutteellista. Varmasti siitäkin saadaan kuitenkin nykyistä enemmän irti, kunhan toimeen tartutaan. Tuoreimpien projektien osaltahan tietosisältö oli selvästi aukottomampaa ja parempaa kuin vanhimpien. Sekin osoittaa asian tärkeyden muutosta. Aukkoja on silti vieläkin.

Ilmeisesti toimitusnopeus ei todellisuudessa olekaan vielä huipputärkeä asia, koska sitä ei mitata systemaattisesti. Sitten kun joku ohjelmistotoimittaja mainostaa toimittavansa ”tuhannen function pointin ohjelmiston samalla hinnalla mutta keskimäärin 30 % nopeammin kuin kilpailijat”, voidaan puhua todellisista menestystekijöistä ja varmasta menestyksestä.

Pekka Forselius

Pekka Forselius on Software Technology Transfer Finland Oy:ssä johtajana ja projektijohdon konsulttina. Hän on FiSMAn Experience-jaoston toiminnanjohtaja ja hallituksen sihteeri sekä SYTYKE ry:n johtokunnan puheenjohtaja. Pekka on valmistunut Filosofian maisteriksi Jyväskylän yliopiston Informaatioteknologian tiedekunnasta, suorittanut MBA-tutkinnon ja toimi mm Tom Weckströmin tässä artikkelissa käsitellyn erikoistyön ohjaajana.

Tom Weckström

Tom Weckström on Lifix Systems Oy:n teknologiajohtaja. Aiemmin hän on työskennellyt mm. F-Securella projektinhallinnan asiantuntijana. Tom on valmistunut Diplomi-insinööriksi Teknillisen Korkeakoulun tietotekniikan osastolta.

Varasta vielä vähän lisää

Pirjo Salo,
TietoEnator Oyj

Parinkymmenen vuoden systeemityökokemus on vahvistanut alkuperäisen uskoni siihen että näillä markkinoilla pärjää parhaiten, jos on tarpeeksi yksinkertainen ja osaa varastaa.

Luottamus

Hankalinta on löytää jotain kunnollista käytettävää johtuen parista omasta ennakkoluulosta. Ensimmäinen perinteinen uskomukseni on ollut se, että alan ehdottomasti innovatiivisin, osaavin ja virheettömintä tuotosta tekevä henkilö istuu minun työpisteessäni. Toisen ammattilaisen ajattelemaan, tekemään tai kokoamaan ei voi mitenkään luottaa. Ja joka tapauksessa keksin pyörän nopeammin itse kuin otan selvää miten naapurin keksimä pyörä toimii tai miten sen saisi sopimaan siihen missä tarvitsen nyt juuri jotain sen tapaista. Toinen este on ajatus siitä ettei kenelläkään ole niin monimutkaista ympäristöä, vaikeita sääntöjä ja ongelmallisia ratkaisuita kuin juuri minulla. Eikä moiseen viidakkoon siis voi olla tarjolla mitään valmista, kaikki on tehtävä itse, työkaluista alkaen. Kunhan asenteensa saa muutettua tuloksekas varastaminen on mahdollista ja se muuttuu hieman kunniakkaammaksi menettelyksi eli uusiokäytöksi.

Avoimuus

Sujuvuutta uusiokäyttöön tulee mikäli halutut kohteet ovat avoimesti saatavilla ja ne on kuvattu jollakin ymmärrettävällä tavalla. Tarvitaan siis mahdollisuutta tutkia mustan laatikon sisältöä ja dokumentti laatikon olemuksesta.

90-luvun alussa näytti siltä kuin systeemityöhön tulisi ryhtiä ja kokoonpanon hallinta helpottuisi, kun ryhdytään käyttämään kuvauskantoja ja niitä hyödyntäviä suunnitteluvälineitä. Olin vilpittömän innostunut siitä, että kaikkien työvaiheiden tulokset ovat saatavilla yhdessä yhteisessä paikassa. Muutos vaikutti suorastaan huimalta siihen verrattuna että dokumentit olivat jonkun kesälomalaisen työaseman kovalevyllä tai parhaimmillaan konsultin kannettavassa. Ja mikä mahdollisuus kopioida hyviä ratkaisuja, oppia ja samalla yhteismitallistaa syntyviä tuloksia. Toisin kuitenkin kävi, ratkaisut ja niiden kuvaukset katosivat kuvauskannan uumeniin eikä ole mitään lainattavaa tai kopioitavaa vaan on helpompaa määritellä oma ‘postinnumerot’ yksilö ja suunnitella vastaava koodisto ja toteuttaa siihen tarvittavat ylläpito ja päivitys toiminnot.

Pienen mutta lohdullisen poikkeuksen tässä piilottamisen ja salaamisen maailmassa tekee html internetissä. Kolmisen vuotta sitten otin tehdäkseni vaatimattomat kotisivut, siitäkin huolimatta etten osannut yhtään mitään vaan olin ainoastaan innostunut hölmö. Pelastukseksi tuli varastaminen ja avoin ja antelias ‘view source’. Kulutin useamman tunnin harhailemalla samantapaisia asioita julkaisevien toimijoiden sivuilla ja etsimällä jotain miellyttävän näköistä ja selkeää sivustoa. Ja löydettyäni sopivan, varastin kaiken mahdollisen html:n. Lopulliseen ja toimivaan tulokseen tarvittiin sentään osaajaa, pelkällä varastamisella ei guruksi ylene.

Standardit

Viime vuosina liikennesäännöt ovat muuttuneet vain niiden noudatettaviksi jotka ovat kyllin tyhmiä uskoakseen niihin tai keskimääräistä taitamattomampia kuljettajia. Käytäntö ei kuitenkaan ole muuttanut liikennettä sujuvammaksi.

Sama yleinen suunta huolestuttaa minua levitessään systeemityöhön. Itsenäisyys, luovuus ja kyky kulkea omia polkujaan ovat kunnioitettavia ja hyviä ominaisuuksia mutta niiden käyttäminen täydellä teholla systeemityössä vaikeuttaa yhteiskäyttöisyyttä, ylläpidettävyyttä ja yleistämistä. Esimerkiksi jos perinteisintä standardia tietokannan tai ohjelman nimeämisestä pidetään kuolleena kirjaimena, siirtyminen sovellusalueesta toiseen tai ympäristöstä toiseen vaatii aina uuden oppimista. Standardit ja säännöt ovat toisinaan puutteellisia ja niiden kehittyminen on usein hidasta, mutta niissä on runsaasti sitä yhteistä ajattelua, kokemusta ja osaamista jonka avulla kitka pienenee. Minun on turhaa mietiskellä miten euro, dollari, henkilötunnus, pankkitili, puhelinnumero tai sähköpostiosoite merkitään kun jotkut ovat sen jo pähkäilleet ja vieläpä julkaisseet.

Pari ohjetta otsikon toteuttamiseksi

- Tunnista testatut ratkaisut, kattava dokumentaatio jne. Ja varasta niitä.
- Keräile tyylikkäitä ja taloudellisia komponentteja, vaikket tarvitsisikaan. Niistä oppii.

Älä suojele mustasukkaisesti omia tekemisiäsi, ne ovat kelvollisia ilman anteeksipyyntöjäkin.

Pirjo Salo,
TietoEnator Oyj

Ohjelmistotuotannon roolit ja dokumentit

Irmeli Nyman,
YLE

Miksi roolit eivät näy systeemityössä?

TV- ja radio-ohjelmien tekijät eivät jää piiloon, sillä jokaisen ohjelman tiedoissa aina listataan henkilöt rooleittain. Miksi atk-ohjelmien tekijöitä ei kerrota atk-ohjelmatiedoissa? Kyllä meille atk-ohjelmien tekijöillekin kelpaisi maine ja kunnia puhumattakaan ohjelmien uudelleen ajamisesta tulevista korvauksista! Osaisikohan joku vastata tähän kysymykseen?

Tässä artikkelissa keskitytään atk-ohjelmien rakentamisessa tarvittaviin rooleihin. Kirjoittajan roolissa on Irmeli Nyman ja lisätietämystä ovat antaneet Paula Miinalainen, Pirjo Ahtiainen ja Silja Räisänen.

Termejä

Tietojärjestelmä on Tietotekniikan liiton atk-sanakirjan mukaan ihmisistä, tietojenkäsittelylaitteistoista, tiedonsiirtolaitteista ja ohjelmistoista koostuva järjestelmä, jonka tarkoitus on tietoja käsittelemällä tehostaa, helpottaa jotakin toimintaa tai tehdä toiminta mahdolliseksi. Ohjelmisto on siis osa tietojärjestelmää. Ohjelmistotuotanto tarkoittaa tässä ohjelmiston rakentamista ja pelkkä tuotanto tarkoittaa ohjelmiston käyttöä tosi tarkoituksessa, vastakohtana on testaus. Ohjelmistotuotantoa kutsutaan myös sovelluskehitykseksi, silloin kun tavoitteena on tehdä sovellus. Sovellus on tehtävän suoritus tiettyä menetelmää tai tekniikkaa käyttäen, kuten tietojenkäsittelytehtävän toteutus tietotekniikan avulla.

Roolit ja osaaminen

Eräässä historian vaiheessa oli ammattikunnat, jotka osasivat tehdä tiettyjä töitä, joita muut eivät sitten saaneetkaan tehdä. Tämä ei päde enää, varsinkin ohjelmistoja saa tehdä kuka hyvänsä ja usein jälkikin on sen mukaista. Hyvän ohjelmiston tekeminen vaatii kuitenkin hyvää ammattitaitoa. Tarvittava ammattitaito tuodaan esille roolien avulla ja tavallista on, että jokainen ohjelmistotuotantoon osallistuja toimii työn kestäessä useassa roolissa. Jos henkilö joutuu ottamaan roolin, johon tarvittavaa osaamista hänellä ei ole, tulisi hänen tuoda se ilmi. Johdon tehtävä on ratkaista asia sitten joko nimeämällä osaava henkilö ko. rooliin tai hankkia konsultointia tai koulutusta.

t t t tt tt tt tt tt t tt t t t t tt t t tttttt t t t t t t t tttt tttt t tt t t t

Tämän artikkelin keskeinen osa on esitetty jäljempänä näkyvässä taulukossa.

Taulukossa as-etuliite tarkoittaa asiakasta ja it-etuliite informaatiotekniikkaa.

Taulukossa ohjelmistotuotanto on jaettu seitsemään vaiheeseen alkaen määrittelyllä ja päättyen ylläpitoon. ”Projektin ohjaus” - sarakkeessa on lisäksi kuvattu projektityössä tarvittavat roolit.

Vaikka rooli jo kertoo tekemisen laadun, on taulukossa vielä korostettu tekemistä siten, että taulukkoon

- toiselle riville on koottu päättämisen ja vastaamisen roolit
- kolmannelle riville hyväksymisen, neuvomisen ja ideoinnin roolit
- neljännelle riville on koottu varsinaisen tekemisen eli suorittamisen roolit
- viidennelle riville tiedottamisen
- kuudennella rivillä on mainittu roolit, jotka suorittavat katselmoinnin
- seitsemännelle riville on lista niistä dokumenteista, jotka pitäisi valmistua kussakin vaiheessa
- kahdeksannelle riville on mainittu rungon nimi, joka on tarkoitettu dokumenttien laatijan tarkistuslistaksi. Joitakin runkoja on koottu artikkelin loppuun
- viimeisellä rivillä on sitten säilytyskansion nimi, joka voi olla sekä pahvinen että sähköinen.

Projektin suunnittelu roolien avulla

Jokainen ohjelmistotuotanto on projekti, joka alkaa kun on tehty päätös ja kun on olemassa resursseja päätöksen toteuttamiseen ja päättyy kun tuotanto alkaa. Projekti käynnistyy siten, että nimetään projektipäällikkö. Projektipäällikkö tekee projektisuunnitelman. Projektisuunnitelma voisi aluksi tehdä nimeämällä projektin tarvitsemat roolit, ei henkilöt. Sen jälkeen voisi etsiä henkilöt ja mieluiten niin, että samalle henkilölle tulee useita rooleja. Myös projektipäälliköllä pitäisi projektissa muitakin rooleja, koska sillä tavoin hän pääsee parhaiten sisälle asioihin ja oppii itsekin uutta. Tärkeää on, että myös ohjausryhmän jäsenille kerrotaan, että missä eri roolissa kukin päätöksiä tekee. Esim. ohjausryhmän jäsen voi olla paitsi päättäjänä myös asiantuntijana ja käyttäjänä.

Määrittely ja prosessien kehitys

Määrittelyn lähtökohtana on tavallisesti vaatimukset uudelle järjestelmälle. Määrittelyn alussa pitäisi myös olla kuvaus nykyjärjestelmästä, koska silloin uudetkin projektin jäsenet tietävät mitä jo on olemassa ja mikä ainakin pitää uudessa järjestelmässä toimia ja ettei samoja asioita tarvitse kovin monta kertaa kysyä käyttäjiltä. Lähtötietoina saisi myös olla tavoitejärjestelmän kuvaus, joka saa olla mieluimmin olla laaja kuin liian suppea, eikä siinä tarvitse ajatella teknistä toteutusta., vaan sitä mikä olisi paras lopputulos. Edellä mainitut asiakirjat pitäisi syntyä as-suunnittelijan roolissa.

Määrittely pitää tehdä hyvin ja siinä pitäisi myös olla aikaa ja valtuuksia kehittää itse toimintaprosessia. Määrittelyä helpottaa nykyisin se, että on olemassa

välineitä, joilla voidaan helposti protoilla. Määrittelyn tuloksena syntyy uuden toiminnan prosessikuvaus ja tavoitetilan toiminnallinen määrittely. Nämä asiakirjat tekee it-suunnittelija yhdessä asiakkaan kanssa. *Määrittelyvaiheen päätyttyä nähdään vasta mitä kaikkea pitää tehdä, joten projektisuunnitelmaa on syytä tarkistaa tässä vaiheessa. Myös ohjausryhmälle pitää määrittelyn tulokset esitellä hyvin. Tässä vaiheessa myös nähdään mitä kannattaa ensin toteuttaa ja mikä myöhemmin.*

Suunnittelu

It-arkkitehti, it-tietoliikenne-, it-tietokanta- ja it-modulisuunnittelija suunnittelevat järjestelmän rakenteen ja päättävät millä työvälineellä toteutus tehdään. Jos ohjelmiston rakenteen suunnittelussa ja toiminnan yleisten periaatteiden suunnittelussa onnistutaan, saadaan ohjelmisto, joka palvelee käyttäjiä hyvin ja jota on myös helppo ylläpitää ja kehittää.

Käyttöliittymän suunnittelua pitää tehdä jo määrittelyvaiheessa, mutta tarkka suunnittelu onnistuu vasta arkkitehtuurin suunnittelun jälkeen.

Toteutus

Toteutus on ohjelmien kirjoittamista, testaamista ja esille tulevien ongelmien ratkomista. Joskus joudutaan palaamaan suunnittelu ja jopa määrittelyvaiheeseenkin, jos ongelmat eivät ratkea tai siksi, että vaatimuksia muutetaan.

Jos halutaan hyvää ylläpidettävyyttä, pitää ohjelmoinnissa noudattaa ainakin seuraavia periaatteita:

- sovitaan moduulien ja tietokenttien nimeämissäännöt ja noudatetaan niitä
- kommentoidaan koodi
- parametrejä ei upoteta koodiin, vaan niille tehdään oma ylläpitorutiini

Integrointi ja järjestelmätestaus

Testaustarve riippuu suuresti tietojärjestelmästä ja ohjausryhmän pitäisi ottaa kantaa siihen, miten hyvin testataan. Jos testaus tehdään hyvin, tehdään testaussuunnitelma, jonka mukaan testataan ja joka vielä on sellainen, että se voidaan toistaa helposti myös myöhemmin, kun ohjelmistoa on muutettu. Testaamista varten on nykyisin olemassa hyviä apuvälineitä, joita kannattaa käyttää, jos testaustarve on toistuva.

Tiedotus, koulutus ja neuvonta

Ohjelmistoa tehtäessä on erityistä huomiota kiinnitettävä sen käytettävyyteen ja siihen, että ohjelmistosta löytyy apua käyttäjälle. Jos käyttäjiä on paljon, ei luokkakoulusta voida järjestää, koska siihen ei ole aikaa eikä rahaa. Mutta nykytekniikalla tietoa voidaan jakaa helposti.

As-tiedottajalla onkin tärkeä rooli. Hän yhdessä it-ylläpitäjän kanssa huolehtii, että tieto kulkee käyttäjille koko järjestelmän elinkaaren aikana.

Tuotanto

Projekti päättyy, kun tuotanto alkaa ja projektin tehtävät siirretään tuotantoa hoitaville henkilöille. Isojen järjestelmien tuotantoon otto suorittaan oman projektina. Jokaisella järjestelmällä on ns. pääkäyttäjä, joka voi antaa toisille käyttäjille enemmän oikeuksia ja joka voi tehdä tiettyjä rutiineja, joita muut käyttäjät eivät voi tehdä. Järjestelmävalvoja voi toimia pääkäyttäjän roolissa, jos hän tuntee riittävästi järjestelmää, mutta tavallista on, että pääkäyttäjän tehtäviä hoitaa it-ylläpitäjä, silloin kun asiakkaan pääkäyttäjä ei ole käytettävissä.

Ylläpito

Varsin hyvä olisi, jos ohjelmiston ylläpitokin luovutettaisi eri henkilöille eikä se jäisi tekijöille. Tällä tavalla varmistettaisi dokumentointi ja vähennettäisi töiden henkilöriippuvuutta ja lisättäisi organisaation oppimista tai ainakin olisi varahenkilö, joka jotain osaisi, vaikka ei voisi ottaakaan ylläpitovastuuta. Ylläpitäjän tulee käsitellä ja luokitella muutospyynnöt seuraavasti

- tutkimaton muutospyyntö
- selvitetty muutospyyntö
- hyväksytty muutospyyntö

OHJELMISTOTUOTANNON ROOLIT JA DOKUMENTIT

	PROJEKTIN OHJAUS	MÄÄRITTELY	SUUNNITTELU	TOTEUTUS	JÄRJESTELMÄ-TESTAUS	KOULUTUS JA NEUVONTA	projekti päättyy V TUOTANTO	YLLÄPITO
PÄÄTTÄÄ/ VASTAA	- ohjausryhmä	- ohjausryhmä	- ohjausryhmä	- ohjausryhmä	- ohjausryhmä	- ohjausryhmä	- järjestelmä-päällikkö	- omistaja
HYVÄKSYY/ NEUVOO/ IDEOI	- projektiin osallistujat	- as-asiantuntija - as-käyttäjä	- it-asiantuntija	- it-kolleega	- as-käyttäjä		- as-käyttäjä	- as-käyttäjä
SUORITTAA	- it-projektipäällikkö - as-projektipäällikkö	- as-suunnittelija - it-suunnittelija - as-prosessin suunnittelija	- it-arkkitehti - it-käyttöliittymä-suunnittelija - it-tietoliikenne-suunnittelija - it-tietokanta-suunnittelija - it-moduli-suunnittelija	- it-ohjelmoija	- as-testaaja - it-testaaja	- as-kouluttaja - as-neuvoja	- järjestelmävalvoja - as-pääkäyttäjä	- it-ylläpitäjä
TIEDOTTAA	- as&it-tiedottaja	- as-tiedottaja	- it-tiedottaja		- it-tiedottaja	- as-tiedottaja	- as-tiedottaja.	- as-tiedottaja
KATSELMOI	- muiden projektien päälliköt ja asiantuntijat	- as-käyttäjä - as-asiantuntija - as-projekti-päällikkö - it-projektipäällikkö	- ulkopuolinen asiantuntija - it-projekti-päällikkö	- it-kolleega - it-ohjelmoija - it-ylläpitäjä	- as-käyttäjä			
DOKUMENTIT	- projekti-suunnitelma - sopimus - seurantarap.	- toiminnallinen määrittely tavoitetilasta - nykytilan kuvaus	- tekninen määrittely	- kommentoitu ohjelmakoodi	- testaus-suunnitelma - testitapaukset - testausdokumentit	- käyttäjän ohjeet	- tuotannon ohjeet	- muutospyyntö - dokumenttien ylläpito
RUNGOT	- projekti-suunnitelman runko		- teknisen määrittelyn sisältörunko	- kielikohtainen tarkistuslista	- testaus-suunnitelman runko	-	- tuotantokansio-runko	
SÄILYTYS-KANSIOT	- projektikansio		- suunnittelukansio	- moduulikansiot	-testauskansio	- käyttäjän ohjekansio	- tuotantokansio	- muutospyynnöt

- aikataulutettu muutospyyntö
- toteutettu, tiedotettu ja arkistoitu muutospyyntö
- Muutokset on hyvä niputtaa ja toteuttaa yhdellä kertaa, koska muutokset pitää testata ja tiedottaa.

Irmeli Nyman,
YLE

Projektisuunnitelman runko

1 Johdanto
1.1 Yleiskuvaus
1.2 Tuote
1.3 Suunnitelman ylläpito
1.4 Viitatut dokumentit
1.5 Määritelmät termit ja nimeämissopimukset (1

2 Projektin organisointi
2.1 Projektin vaiheistus
2.2 Organisaation rakenne
2.3 Sidosryhmien kuvaus
2.4 Projektin henkilöt rooleittain ja yhteystiedoittain (1

3 Projektin ohjaaminen
3.1 Tavoitteet ja priorisointi
3.2 Oletukset , riippuvuudet ja reunaehdot
3.3 Riskien hallinta
3.4 Seuranta ja ohjaus
3.5 Henkilöresurssien käytön suunnittelu

4 Tekniikka
4.1 Menetelmät ja työkalut
4.2 Dokumentointi
4.3 Tuotteenhallinta
4.4 Laadun varmistus

5 Vaiheet ja aikatauluja budjetti
5.1 Projektin osittaminen
5.2 Riippuvuudet
5.3 Resurssien käyttö ajanfunktiona ja vaiheittain (1
5.4 Budjetti ja resurssien allokointi
5.5 Aikataulu (1

1) Näitä kohtia ainakin ylläpidetään projektin aikana ja näistä voidaan muodostaa seurantaraportti.

Tavoitejärjestelmän runko

- nykyjärjestelmän kuvaus (lomakkeita, tulosteita, ohjeita)
- tavoitejärjestelmän määrittely
  - tavoitteet
  - tapahtumat ja työnkulku esim.
    - asiakaspalvelija päivittää asiakkaan tiedot
    - asiakas tekee tilauksen
    - kuljettaja hakee kuljetustilauksen
    - johtaja pyytää kululaskelman
  - käyttöliittymän tietosisältö
  - näkymät ja tilastot
  - muista järjestelmistä tarvittavat tiedot

Teknisen määrittelyn runko

- ohjelmiston sanallinen yleiskuvaus
- kaavio rakenteesta ja liittymistä
- yksityiskohtaiset kuvaukset
- vuotapahtumat ja käsittelysäännöt
- ajastetut tapahtumat ja käsittelysäännöt
- ohjaustapahtumat ja käsittelysäännöt
- roolit ja niihin liittyvät käyttöoikeudet
- käyttöliittymän ja lomakkeet ja niihin liittyvät tarkistukset
- näkymät ja tilastot
- liittymät muihin järjestelmiin
- suojaukset ja varajärjestelmät

Moduulikansion runko:

- tietokantakuvaukset
- tietuekuvaukset
- moduulikuvaukset
- koodauslistat
- rakennekuvat

Järjestelmätestauksen runko

- testauksen kohde ja tavoitteet
- testausympäristö
- testauksen organisointi ja raportointi
- testausstrategia ja integrointisuunnitelma
- toimintojen testaustapaukset ja hyväksymiskriteerit
- toiminnallisten ominaisuuksien testaus
- erikoistilanteet
- ominaisuudet joita ei testata

Tuotantokansion runko

- yhteystiedot
- luettelo tekemättömistä töistä
- työnjako, kuka saa tehdä ja mitä
- ajojen ajo-ohjeet
- asennusohjeet
- kantojen hoito-ohjeet
- tehdyt tuotantoon otot
- virhetilanteiden hoito
- parametritietojen ylläpito-ohjeet
- varmistusmenettely
- varajärjestelmä

Tiimeistä tehoa projektityöhön

Eija Hamina-Mäki,
TietoEnator Oyj, Finanssiryhmä

Olemme oppineet ja jo tottuneetkin toteuttamaan niin mittavat kuin pienemmätkin ohjelmistotuotantohankkeemme projekteina. Hallitsemme – ainakin periaatteessa – projektityötaidot hyvin. Kuitenkaan projektit eivät aina onnistu aivan toivotulla tavalla ja projektipäälliköt kokevat, että projektin ohjaamisessa, organisoinnissa ja työtavoissa olisi vielä kehitettävää.

Tiimityö on ollut yksi lähivuosien ”muoti-ilmiöitä”, jonka toteuttamisesta on sekä hyviä että pettyneitä kokemuksia. Entäpä jos projektityötapaa vahvistetaan tiimiajattelun periaatteilla? Watt S. Humphrey on valottanut kirjassaan Introduction to the Team Software Process[SM] tätä lähestymistapaa ja on itse hyvin vakuuttunut sen toimivuudesta. Olisiko tässä ”perinteiseen” projektityöhön kaivattua vahvistusta?

Humphrey on kuvannut Team software process'in (TSP), joka on teollinen ohjelmistoprosessi laajahkoille projekteille. TPSi on TPS:n kevennetty versio, joka perustuu samaan konseptiin ja samoihin menettelyihin, mutta sopii pienemmille projekteille. TPSi:tä voisi luonnehtia, että se on kehys, yhdistelmä prosessia, tuotetta ja tiimityötä.

Seuraavaan artikkeliin on poimittu joitain Humphreyn (TPSi-) ajatuksia, joita kirjoittaja on vapaasti suomentanut ja mukauttanut omien kokemustensa pohjalta.

Team software process

Projektitiimin tyypillisenä tehtävänä on rakentaa ohjelmistotuote ja rakentaminen toteutetaan iteratiivisesti kahdessa tai kolmessa syklissä eli kehittämiskierroksella. Tämä työskentelytapa antaa mahdollisuuden laadun parantamiseen ja korjausten tekemiseen jo tuotantoprosessin aikana. Rakentamisprosessi on aina myös oppimisprosessi kaikille siihen osallistuville. Ensimmäisessä syklissä rakennetaan pieni toimiva tuotteen ydin. Jokaisella onnistuneella kerroksella rakennetaan lisää toiminnallisuutta tuotteeseen. Kullakin kehittämiskierroksella tai uudessa projektissa käytetään aiempaa tietoa ja kokemusta hyväksi.

Tiimeistä on jonkin verran kokemusta – hyvää ja huonoa. Kokemuksesta tiedämme, että joukko lahjakkaita, osaavia henkilöitä työskentelemässä yhdessä ei ole tiimi eikä tiimit synny sattumalta tai itsestään. Tälle joukolle tarvitaan yhteiset tavoitteet, yhteinen toimintasuunnitelma ja nimetty vetäjä. Lisäksi jäsenten tulee ymmärtää toistensa vahvuudet ja heikkoudet, tukea tiimitovereitaan ja kyetä tarvitessa pyytämään apua.

Mikä tiimi on?

Dyerin mukaan tiimi koostuu vähintään kahdesta henkilöstä, jotka työskentelevät kohti yhteistä tavoitetta tai päämäärää, missä jokaisella tiimin jäsenellä on oma erityinen roolinsa tai tehtävänsä ja missä päämäärän saavuttaminen vaatii jollain tavalla kaikkien tiimin jäsenten panoksen.

Tiimin koko voi olla parista henkilöstä aina satoihin henkilöihin. Parhaiten toimivat ja tehokkaimmat tiimit ovat kuitenkin sellaisia, joissa henkilöitä on vain muutama, jolloin jäsenet tuntevat toisensa ja ovat henkilökohtaisesti yhteyksissä toisiinsa. Projektitiimin suositeltava koko on alle parikymmentä henkilöä, vaikka sitten koko hanke muodostuisikin useasta tiimistä.

Kaikki ryhmät eivät ole tiimejä. Seuraavien ehtojen tulee täyttyä, jotta ryhmää voidaan pitää tiiminä:

- tiimin tehtävän tulee olla selkeä, täsmällisesti määritelty, niin että jokainen tietää, mitä tiimin tulee tehdä tai saavuttaa
- tiimi on selkeästi rajattu: on selvää, ketkä siihen kuuluvat ja ketkä eivät. Jokainen tiimin jäsen tuntee toisensa, jokaisen työ on toisille näkyvää, jokainen tietää myös kaikkien toistensa roolit
- tiimi kontrolloi tehtäviään. Henkilöt tietävät mitä, miten ja milloin tehdä sekä milloin tehtävän pitää olla valmis. Jäsenet tietävät, että he ovat vastuussa tehtävästään ja he kontrolloivat käyttämäänsä prosessia. Lisäksi heillä on kyky tehdä annettu tehtävä ja tieto, ettei ketään muuta ole valjastettu ao. tehtävää tekemään.

Tiimityön pulmia

Vaikka usein nähdään tiimityön edut, on myös olemassa yleisiä tiimityölle ominaisia ongelmia, kuten tiimin johtaminen, yhteistyö, osallistuminen, hitaus, laatu. Tehoton johtaminen aiheuttaa suunnitelmista muutakin kuin oikeanlaisia henkilöitä, tehtäviä ja työskentelymahdollisuuksia. Tiimillä pitää olla tärkeä tehtävä ja pitää olla ympäristö, joka tukee tiimityötä. Tiimi joutuu kohtaamaan voimakkaita haasteita ja tiimiä pitää rohkaista suunnittelemaan ja johtamaan omaa työtään. Tiimin tukipilareita ovat:

- koheesio
- tavoitteet
- palaute
- yhteiset puitteet.

Tiimin koheesio eli yhdessäpitävä voima

Tiimin jäsenet muodostavat fyysisesti ja emotionaalisesti yksikön, jossa kommunikoidaan vapaasti ja paljon. Jäsenten ei suinkaan tarvitse olla hyviä ystäviä keskenään, mutta heidän työskentelynsä on läheistä, toinen toistaan tukevaa ja kunnioittavaa. Jos koheesiota on vain vähän, ryhmän jäsenet toimivat enemmänkin yksilöinä kuin joukkueena, heidän saattaa olla vaikeaa tehdä kompromisseja, eikä heillä ole yhteisiä arvoja ja tavoitteita. Jos ryhmällä on vahva koheesio, he viettävät paljon aikaa yhdessä tehden yhteistyötä vahvassa vuorovaikutuksessa toistensa kanssa.

Eija Hamina-Mäki,
Projektipäällikkö,
TietoEnator Oyj, Finanssiryhmä,
eija.hamina-maki@tietoenator.com

Ammattimainen systeemityö perustuu standardeihin

Risto Nevalainen,
STTF Oy

Tarvetta standardeihin olisi, mutta kypsyys ei vielä riitä

Muutama viikko sitten olin mukana Suomen mobiiliklusterin lähivuosien strategian määrittelykokouksessa. Siellä eriteltiin mobiiliklusterimme erilaisia vahvuuksia ja välttämättömyyksiä muutama vuosi tulevaisuuteen. Keskustelussa kaikkein tärkeimmäksi asiaksi nousi alan kansainvälisten standardien huomiointi ja niihin vaikuttaminen! Jäi mielikuva, että asioihin osattiin suhtautua todella ammattimaisesti. Tietysti tein heti mielikuvamatkaa, miten ”systeemityöklusterin” strategiaistunnossa olisi käynyt – tokkopa standardit olisivat tulleet mieleenkään. Niin vähän niihin panostetaan, että niitä ei edes tunneta nimeltä tai aihepiirinä. Kunnolliseen käyttämiseen saatikka niihin vaikuttamiseen ei olla päästy. Toki jotkut asiat ovat tunnettuja enimmäkseen teknisten standardien alueella, esim. nykyinen kova kiinnostus XML-asioihin.

Suuri osa systeemityön perusstandardeista on luotu ISO:ssa. Lisäksi toimittajilla ja välinevalmistajilla on lukuisia de facto- standardeja, esim. UML. Esim. ODB/CORBA on aihepiiri, jota ollaan parhaillaan standardoimassa monilla tahoilla. Tässä artikkelissa pohditaan standardien merkitystä ja niiden merkityksen kasvua tulevassa, komponenttiperustaisessa ohjelmistokehityksessä.

Lähtökohtani on, että laadukas ja teollinen ohjelmistotuotanto tarvitsee standardeja. Tätä en aseta mitenkään kyseenalaiseksi – niinkuin eivät näyttönsä antaneet mobiiliklusterin yrityksetkään. Valitettavasti standardeja ei voi ottaa käyttöön ja rakentaa toimintaansa niiden varaan ilman merkittävää oppimista ja panostamista. On oltava laatukulttuuria ja systematiikkaa, muuten homma uupuu puuhastelun asteelle. Yrityksiä kiertäessäni olen havainnut, että sana on alkanut kyllä mennä perille eli hyviäkin esimerkkejä on jo olemassa.

t ttttttttt tt tt ttttttttttt standardeilla ei kysyntää

Standardeilla on ollut huono maine tietotekniikka-alalla. Monille tulee mieleen jo käytöstä poistuneiden tekniikoiden jälkikäteinen kirjaaminen. Tai sitten jotakin, mikä ei tapahdu kun ne suuret amerikkalaiset firmat eivät standardeista kuitenkaan välitä. Antakoon markkinavoimien ratkaista!

Valtaosaltaan kuva on oikea. Standardointi on ollut tehotonta ja hidasta. Joskus on yritetty standardoida liian suuria keskeneräisiä asioita, esim. ISO/OSI. Kumma kyllä, samat asiat löytävät usein tiensä käytäntöön ns. de facto- standardien ja kaupallisten ratkaisujen kautta, esimerkiksi Internetin standardeina tai vaikkapa tietoturvana.

Tietoliikenneala on ollut uskollisempaa standardeille kuin tietotekniikka tai ohjelmistotuotanto. CCITT:n, nyttemmin ITU:n avulla on saatu aikaan vakaita, standardienmukaisia ratkaisuja. Suuret tietoliikennejätit tekevät kyllä omia tulevaisuuden konseptejaan, mutta määriteltyään pitävät ne avoimesti saatavilla. WAP ja UMTS ovat tästä hyviä esimerkkejä.

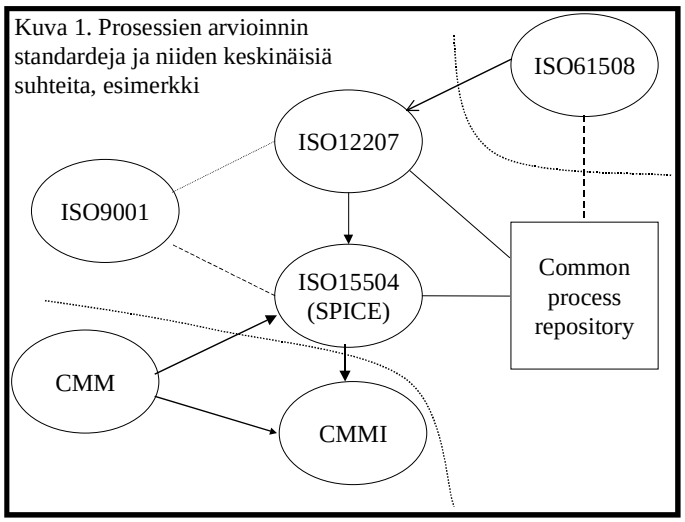
Ohjelmistotuotanto on kulkenut kaupallisten ratkaisujen lieassa. Edes kaikkein pyhimpiä asioita ei toteudu, kuten kunnolliset merkkistandardit. Ne ovat olleet hyvin määriteltyjä jo kymmenen vuotta mutta eipä vieläkään ole ollut mahdollista käyttää skandeja esim. sähköposti- tai webbiosoitteissa. Todella uskomatonta ja suorastaan hävytöntä! Vasta nyt kaupallinen paine on noussut tarpeeksi korkeaksi tällaisten perusasioiden kunnolliselle käyttöönotolle.

Ohjelmistotuotannon standardien tarve

Ohjelmistotuotannon standardeja on karkeasti ottaen kolmenlaisia: tekniikoita (esim. ohjelmointikielet, alustat, ODP), menetelmiä (esim. tuotelaatu, ohjelmiston koon mittaaminen) tai konsepteja (esim. ISO9000-3, SPICE, CMMI). Tämän artikkelin mielipiteet perustuvat lähinnä menetelmien ja konseptien standardointiin. Niiden osalta tilanne on paranemassa kaiken aikaa. Syitä kiinnostuksen kasvuun on mielestäni kolme: 1) ohjelmistotuotannon kansainvälistyminen, 2) ohjelmistoliiketoiminnan muutos, erityisesti tuotebusiness ja 3) saadut kohtalaisen hyvät kokemukset.

Suuret ohjelmistoyritykset ovat globaaleja. Harvalla niistäkään on yksin sananvaltaa standardeihin. Moni aikansa markkinavaltias on yrittänyt sanella standardeja tai on hidastanut muiden tekemien standardien käyttöä. Jos muskeli ei ole muuhun riittänyt, kehitystä on sabotoitu tekemällä omia ”yhteensopivia” ratkaisuja. Nykyään kukaan ei usko enää saavansa yksin aikaan merkittäviä standardeja, vaan niitä tehdään virallisissa elimissä tai teollisissa konsortioissa. Suomalaiset kansainvälistyneet ohjelmistotalot ovat taasen pääosin standardien hyödyntäjiä. Ei ole enää mitään mieltä tehdä omia standardeja, vaan käyttää ja soveltaa mitä kansainvälisesti on tarjolla. Nykyään asiakkaatkin vaativat standardienmukaisia ratkaisuja.

Ohjelmistoliiketoiminta on muuttumassa komponenttipohjaiseksi. Rajapinnat ja yhtenäiset alustat korostuvat, samoin komponenttien hallinta ja jakelu. Kunnon standardit ovat ainut mahdollinen ratkaisu. Virallisten standardointielinten merkitys onkin kasvanut. Esimerkiksi ODPC on siirtynyt nyttemmin standardien osalta ISO:n suojiin (ISO/IEC JTC1 SC7/WG19). Myös ISO:n piirissä on tapahtunut keskittämistä ja toiminnan nopeuttamista, jotta alussa mainitut



Kuva 1. Prosessien arvioinnin standardeja ja niiden keskinäisiä suhteita, esimerkki

Kuva 1

standardoinnin epäkohdat poistuisivat. Esimerkiksi ODP otettiin ISO:n piiriin vasta kun se oli verraten kypsää ja vakiintunutta.

Monet standardit koetaan toimiviksi ja järkeviksi! Tämä antaisi uskoa siihen, että ne eivät ole enää jälkijättöisiä vaan jopa edistyksellisiä. ISO onkin ollut aloitteellinen. Tarvetta uusiin standardeihin seurataan ja vain järkevimmät asiat otetaan työn alle. Yhteistyötä tärkeiden osapuolten kesken on lisätty, että vähät voimavarat riittäisivät. Tätä kautta esim. IEEE:n standardeille on tulossa paljon painoa, kun niitä ryhdytään julkaisemaan lähivuosina massoittain myös ISO-standardeina.

Muutama sana laatustandardeista

Laatustandardit ovat systeemityöläisille kiinnostavin aihepiiri ISO:n työssä. Näitäkin on monenlaisia ja monentasoisia. Jotkut ovat laajoja ns. viitekehysstandardeja (esim. tuleva SQUARE, tuotelaadun viitekehys), jotkut normatiivisia vaatimusjoukkoja vaatien tulkintaa ja sovitusta (esim. tulevat prosessimallit ISO12207 ja ISO15288), jotkut puolestaan täsmällisiä todentamiseen ja yhteensopivuuteen käytettäviä standardeja (esim. ISO15504, SPICE ja ISO14403, Functional Sizing).

Valitettavaksi minkään yksittäisen standardin tuntemus ei riitä ollakseen täysiverinen asiantuntija alalla, vaan on tunnettava niiden muodostama kokonaisuus. Esimerkkinä tällaisesta kokonaisuudesta on ohjel-mistoprosessien arvioinnin standardiperhe, ks. kuva 1. Sekin kuva on varsin rajoittunut, eikä tuo esiin muutamia uudempia herkkupaloja esim. tietoturvan ja järjestelmien luotettavuuden standardiehdotuksista.

Monissa standardeissa on sisäänrakennettuna useita eri käyttäjä- tai soveltamisnäkökulmia. Esim. ISO15504 on määritelty käytettäväksi yhtä hyvin ohjelmistokehityksessä kuin ohjelmistohankinnassakin. Sen avulla voi mitata ohjelmistotuotannon eri prosesseja, esim. tekninen kehitystyö, projektinhallinta, ns. tukiprosessit ja vaikkapa organisaation menettelyt ohjelmistotuotannossa. Kun näkökulmia on monta ja samalla on tehtävä globaali standardi, siitä tulee välttämättä varsin laaja. Esim. CMMI on reilut 600 sivua ja vastaava ohjelmistohankinnan prosessisetti sekin reilut 400 sivua!

Miltä lähitulevaisuus näyttää?

ISO:n standardeista kaikkien aikojen hitti on ollut ISO9000-sarja. Sen perävirrassa on syntynyt monenlaista viralista ja de facto-standardia. Ohjelmistotuotannossa keskeisimpiä ovat ISO12207 (ohjelmistotuotannon prosessimalli), ISO15504 (ohjelmistotuotannon kyvykkyyden arviointimalli SPICE ja sen amerikkalainen sovellus CMMI) ja tuotelaadun standardiperhe (ISO9126, ISO14598). Ohjelmistotuotannon (software engineering) rinnalle on tullut järjestelmätekniikka (systems engineering). Näiden varaan rakennetaan seuraavat kymmenen vuotta. Myös ISO9000-sarja on uudistumassa. Juuri saatiin valmiiksi suomennettu perusversio ISO9001:2000 julkaisu. Siitä ollaan parhaillaan tekemässä ohjelmistoalan sovitusta ISO9000-3:2000.

Standardeja tulee myös ohjelmistotuotannon osaamisen hallintaan. Parhaillaan kommentointikierroksella on ns. Software Engineering Body of Knowledge, versio 0.9. Siinä määritellään systeemityöläisen kymmenen keskeisintä osaamisaluetta hyvinkin tarkasti. Kun tähän sovelletaan osaamisen luokitteluasteikkoa (esim. Bloomin asteikko 1-6) ja määritellään kunkin systeemityöammatin vaatima osaamisen syvyys tällä asteikolla, saadaankin varsin yhtenäinen ammattikuva aikaan. Tätä voidaan ruveta opettamaan ja itse kukin voi hakea vaadittua taitoprofiilia vastaavan ammattisertifikaatin. Tällä tavalla voi ilmaista omat taitonsa ja onhan moinen sertifikaatti varsinainen valttikortti palkkaneuvotteluissakin!

Miten pääsisi mukaan?

Standardeista tietäminen on osin viitseliäisyydestä kiinni. Monet varsinkin de facto-standardit ovat helposti webistä löydettävissä, esim. CMMI ja osa CORBA-tyyppisistä avoimia rajapintoja edistävistä standardeista. Kuuluu ikään kuin pelin henkeen olla avoin ja jakaa tietoa sitä kaipaaville.

Valitettavasti osa varsinkin de jure – eli virallisista standardeista on kauppatavaraa eli ne pitää ostaa vieläpä paperijulkaisuna. Tosin sähköinen jakelukin on käynnistymässä. Esimerkkinä vaikkapa ISO9000-standardisarja, joka on viisainta ostaa SFS:stä.

Kaikkein mielenkiintoisinta ainakin minulle ovat uudet, kehitteillä olevat standardit. Niistä perillä pysyminen vaatii panostamista. Esimerkiksi ISO:n uudet ohjelmistotuotantostandardit (ISO IEC JTC1 SC7) saa vain olemalla aktiivisesti mukana standardointityössä. Suomessa homma on viety hiukan pitemmälle eli on perustettu FiSMA (Finnish Software Metrics Association), jolle ohjelmistotuotannon standardien seuraaminen on delegoitu Tietotekniikan kehittämiskeskus TIEKE ry:stä. Osa ohjelmistotuotannon standardeista on Helsingin yliopiston seurannassa, lähinnä ns. ODP-toiminta. FiSMAan pääsee kuka tahansa yritys jäseneksi ja pääsee siten standardoinnin tietopalvelujen piiriin. Lisätietoja FiSMAsta löytyy yrityksemme kotisivuston www.sttf.fi kautta.

Risto Nevalainen,
toimitusjohtaja,
STTF Oy,
riston@sttf.fi,
GSM 0500-507750

Tuotteenhallinta - Configuration Management

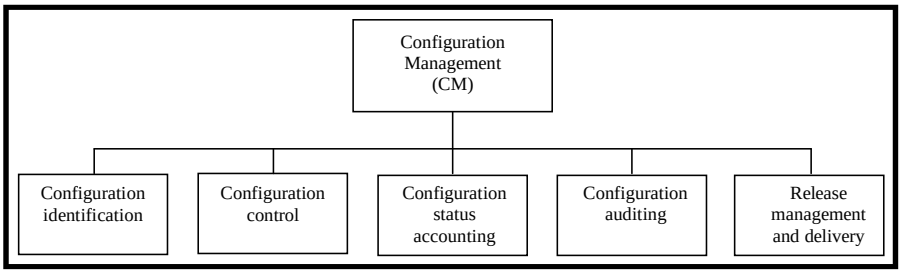
Ulla Vanhanen,
Helsingin liiketalouden ammatti-korkeakoulu

Useimpien it-ammattilaisten kohdalle on joskus sattunut tilanne, jossa

- jo kerran korjattu virhe on putkahtanut uudelleen esiin
- ohjelma, joka toimi vielä eilen, ei jostain selittämättömästä syystä toimi enää tänään
- toimitettaessa asiakkaalle uutta ohjelmistoversiota, ei olla varmoja, mitkä piirteet ja korjaukset siihen on toteutettu.

Jos tuotteenhallinta olisi hoidettu asianmukaisesti, ei olisi tarpeen hukata aikaa tällaisten tilanteiden selvittämiseen ja korjaamiseen.

Tuotteenhallinta on määritelty laatustandardeissa ohjelmistotuotannon tukiprosessiksi. Siitä huolimatta näyttää siltä, että tuotteenhallinta on yksi vähiten tunnetuista systeemityön osa-alueista. Viitteitä tästä antaa myös vuosi sitten suoritettu SYTYKE -jäsenkysely [Hamina-Mäki]. Itse kyselyssä ei ollut yhtään suoraan tuotteenhallintaan liittyvää kysymystä, eikä raportoitujen tulosten perusteella vastauksissa noussut esiin yhtäkään käytössä olevaa tuotteenhallinta-välinettä, vaikka kysymyksen 3 [1)] yhteydessä olisi ollut mahdollista tuoda näitä esiin. Välineitä toki on käytössä, varsinkin lähdekoodin versionhallintaan. Tällöin väline on usein integroitu niin hyvin kehittämisympäristöön, että käyttäjät eivät välttämättä edes huomaa välineen olemassaoloa - ja hyvä näin.



Kuva 1: Tuotteenhallinnan osa-alueet

Esitän aluksi joitakin artikkelin kannalta keskeisiä määritelmiä.

Rakenneosa (Configuration Item = CI): Rakenneosalla tarkoitetaan mitä tahansa ohjelmiston kehittämiseen tai valmiiseen ohjelmistoon liittyvää osaa tai kokonaisuutta (määritysdokumenttia, suunnitteludokumenttia, lähdekoodia, käännöksen tuloksena syntynyttä binäärikoodia, suoritettavaa ohjelmaa tai moduulia, itse ohjelmistoa, käyttöohjetta, testimateriaalia, ...), joka tulee olla yksilöitävissä, talletettavissa, testattavissa, katselmoitavissa, käytettävissä, muutettavissa, toimitettavissa ja/tai ylläpidettävissä. Rakenneosat saattavat olla hyvin erilaisia, toisten ollessa monimutkaisia ja toisten yksinkertaisempia. Rakenneosa saattaa myös koostua toisista rakenneosista (esim. ohjelma joka muodostuu moduuleista). [Kelly]

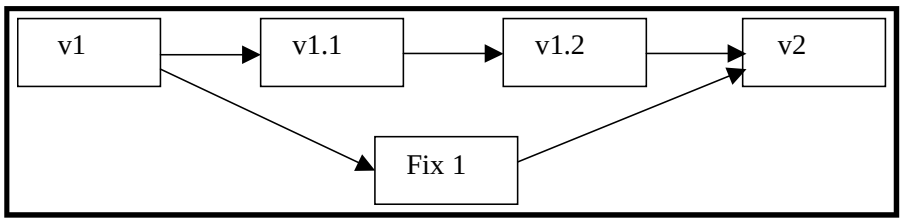
Generointi (build): Generoinnilla tarkoitan suoritettavan ohjelmiston tai sen osan rakentamista lähdekielisistä rakenneosista. Generointi edellyttää vaadittujen toimenpiteiden suorittamista oikeassa järjestyksessä sekä oikeiden työkalujen käyttämistä kussakin vaiheessa. Esimerkiksi C++ -kielellä koodattu moduuli tulee kääntää C++ -kääntäjällä. Cobol-ohjelma puolestaan Cobol-kääntäjällä. Jos suoritettavan ohjelman kokoamiseen tarvitaan link-ohjelmaa, on lähdekieliset moduulit käännettävä ennen linkkausta. Vähänkin laajemman ohjelmiston generoinnissa on lukemattomia vaiheita. Tämän vuoksi generointi yleensä automatisoidaan kuvaamalla se suoritettavana komentojonona. Kutsun tätä komentojonoa jatkossa systeemimalliksi (system model).

Kuvauskanta (repository): Käytän termiä kuvauskanta tarkoittaessani kaikkia tuotteenhallinnan alaisuudessa olevia tietovarastoja. Kuvauskanta sisältää mm. niin toteutettavan ohjelmiston kuvaukset, lähdekoodin, suoritettavien ohjelmien generoinnissa tarvittavat työkalut, käyttöohjeet, muutospyynnöt kuin virheilmoitukset.

Työtila (work space): Työtila on tuotteenhallintavälineiden tarjoama työskentely-ympäristö, joka eristää yksittäisen kehittäjän muusta samanaikaisesta kehittämisestä. Tarvittaessa työtilaan voidaan päivittää toisten tekemiä muutoksia. Esimerkiksi, kun halutaan testata samanaikaisesti eri työtiloissa tehtyjen muutosten yhteensopivuutta.

Versio (version): Rakenneosasta luo-

[1] Onko yrityksessäsi käytössä seuraavia työkaluja?
- Toimistojärjestelmät
- Piirto-ohjelmat
- CASE-välineet
- Testausvälineet
- Muita



Kuva 2: Rakenneosan versiopuu, jossa versiot v1,v1.1, v1.2 ja v2 edustavat päähaaraa, eli peräkkäisiä revisioita. Fix 1 on väliaikainen variantti, johon on tehty ilmoitetun virheen korjaus. Variantti yhdistetään päähaaraan versiossa 2, joka on version 1 jälkeen seuraava asiakkaalle julkistettava versio. Tuotteenhallintavälineet tarjoavat tukea varianttien yhdistämiseen.

daan uusi versio joko sen vuoksi, että

- jotain ominaisuutta halutaan parantaa tai lisätä uutta toiminnallisuutta
- jotain ominaisuutta tulee muuttaa, jotta rakenneosa toimisi uudenlaisessa ympäristössä.

Ensimmäisessä tapauksessa rakenneosasta luodaan uusi revisio, jälkimmäisessä variantti.

Revisio (revision): Rakenneosa on toisen rakenneosan revisio, jos se on luotu muuttamalla tätä aikaisempaa versiota, jonka se korvaa. [Whitgift]

Variantti (variant): Variantit ovat rinnakkaisia versioita. Variantin luominen rakenneosasta ei paranna rakenneosan ominaisuuksia, vaan sovittaa alkion uuteen käyttöympäristöön. [Whitgift]

Variaatioon johtavaa tekijää kutsutaan variaation ulottuvuudeksi (esim. käyttökieli tai laitteisto, jolla ohjelma suoritetaan). Useasta samanaikaisesta tekijästä johtuvaa variaatiota kutsutaan moniulotteiseksi (multidimensional) variaatioksi. [Mohler]

Väliaikainen variantti: Tarve väliaikaisen variantin luomiseen syntyy samanaikaisesta kehittämisestä. Esimerkiksi havaittaessa käytössä olevassa ohjelmistossa pikaista korjaamista vaativa virhe sen jälkeen, kun uuden ohjelmistoversion kehitys on jo aloitettu. Tällöin virheenkorjausta varten luodaan väliaikainen variantti, joka yhdistetään versiopuun päähaaraan ennen uuden version toimittamista asiakkaille. Useimmista tuotteenhallintavälineistä löytyy ominaisuus, joka tukee yhdistämistä.

Muutospyyntö (change request): Muutospyynnöllä yleisimmillään tarkoitan sekä ohjelmiston kehittämiseksi tehtyä muutospyyntöä, että virheen tutkimista ja korjaamista varten tehtyä virheilmoitusta.

Mitä tuotteenhallinta sitten on?

Tuotteenhallinta on peräisin kokoonpanoteollisuudesta (lentokoneteollisuus). Kokoonpanoteollisuudessa laaditaan rakennepiirustuksia, joissa kuvataan tuotteen ja sen komponenttien rakenne. Yhden pienenkin osan muuttaminen saattaa aiheuttaa laajempia muutoksia. Näiden suunnitteluun ja rakenteen kuvaamiseen liittyvien dokumenttien hallinnasta on käytetty nimitystä tuotetiedon hallinta (Product Data Management = PDT). Tietojärjestelmien rakentamisen yhteydessä on puhuttu joko tuotteenhallinnasta (Configuration management = CM) tai ohjelmiston tuotteenhallinnasta (Software Configuration Management = SCM). Vuonna 1998 Brysselissä, tuotteenhallintaa käsittelevässä symposiumissa lanseerattiin lyhenteelle SCM uusi merkitys: System Configuration Management, joka kattaa sekä tuotetiedon hallinnan että ohjelmiston tuotteenhallinnan. Näiden kahden katsotaan olevan varsin lähellä toisiaan, mistä syystä niiden tutkimukseen ja kehittämiseen käytettävät voimavarat halutaan yhdistää. Lisäksi miltei kaikissa laitteissa on nykyisin yhtenä osana sulautettu ohjelmisto. [Estublier]

Artikkelini keskittyy ohjelmiston tuotteenhallintaan (jatkossa CM tai tuotteenhallinta). Tuotteenhallinnan tavoitteena on hallita ja kontrolloida ohjelmiston ja sen rakenneosien luomista ja muuttamista. Käytännössä tuotteenhallinta merkitsee tukea niin versiointiin, muutoksenhallintaan kuin ohjelmiston generointiin ja toimitettavan ohjelmiston kokoamiseen. Tuotteenhallinta palvelee asiakasta, projektipäällikköä ja kehittämistiimin jäseniä tarjoamalla ajantasaista tietoa ohjelmiston ja sen rakenneosien tilasta.

Tuotteenhallinnan osa-alueiksi on eri lähteissä mainittu

- Tunnistaminen (configuration identification)
- Kokoonpanon hallinta (configuration control)
- Tilan seuranta (configuration status accounting)
- Tarkastaminen (configuration auditing).

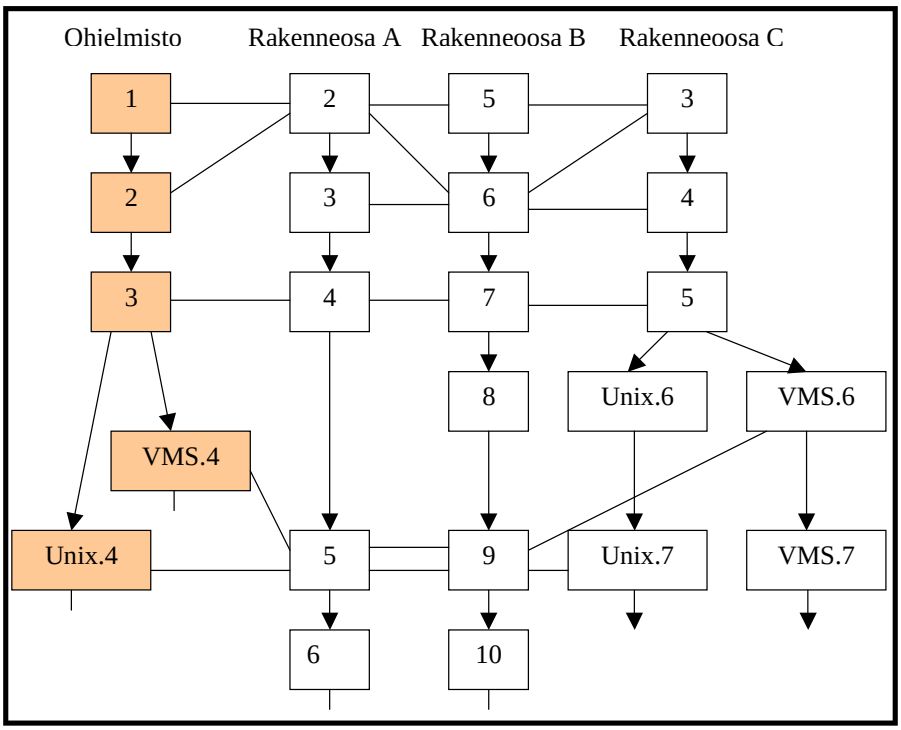
ISO/IEC 12 207:n mukaan tuotteenhallinnan piiriin kuuluu lisäksi

- julkistuksen hallinta ja toimitus (release management and delivery).

Tunnistaminen

Tunnistaminen käsittää ohjelmiston jäsentämisen tuotteenhallinnan tarkoitusperiä palveleviin rakenneosiin, näiden yksiselitteisen nimeämisen, rakenteen kuvaamisen sekä toiminnallisten ja fyysisten piirteiden dokumentoinnin. Tunnistamiseen kuuluu myös kokoonpanon tai yksittäisten rakenneosien versioiden - revisioiden ja varianttien - nimeäminen ja piirteiden dokumentointi. Oleellinen osa tunnistamista on sen määritteleminen, missä vaiheissa ohjelmistosta muodostetaan vaihetaso (baseline), joka toimii perustana seuraavalle kehitysvaiheelle. Ohjelmiston vaihetasot tuotetaan ohjelmistoprosessin tarkistuspisteissä. Niinpä kirjallisuudessa esiintyy yleisesti ainakin seuraavat kehittämisen elinkaaren vaiheisiin liittyvät vaihetasot:

- määritysvaiheen tuloksena syntyvä toiminnallinen vaihetaso (functional baseline), määrityskuvasto
- suunnitteluvaiheen tuloksena syntyvä ohjelmiston rakenteen kuvaava vaihetaso (allocated baseline), suunnittelukuvasto
- tuotevaihetaso (product baseline), joka on hyväksytty toimintavalmis ohjelmisto kaikkine siihen ja sen kokoamiseen liittyvine rakenneosineen.



Kuva 3Ohjelmiston versiot muodostuvat rakenneosiensa versioista [Whitgift]. Ohjelmiston version 1 kokoonpano muodostetaan rakenneosan A versiosta 2, rakenneosan B versiosta 5 ja rakenneosan C versiosta 3. Ohjelmistoversion kokoonpano kuvataan systeemimallilla, joka määrittelee kokoonpanoon tulevat rakenneosat, niiden versiot ja generoinnin. Systeemimalli on rekursiivinen: mikäli rakenneosa on kokoonpano se kuvaa myös, miten rakenneosan versiokohtainen kokoonpano muodostetaan. Kaaviossa on kuvattu, miten ohjelmistosta muodostetaan version 3 jälkeen kaksi erillistä varianttia. Variaation lokalisoinnista (variaation aiheuttavien tekijöiden kokoaminen rakenneosaan C) johtuen pääosa ohjelmistovarianttien rakenneosista säilyy yhteisinä.

Käytännössä nimeäminen tarkoittaa sitä, että varsinaisen nimen lisäksi kuvataan, missä kohteena oleva rakenneosa sijaitsee.

Jotta toteutettavaksi tulevat muutokset saataisiin rajatuksi mahdollisimman harvaan ohjelmiston rakenneosaan, on ohjelmiston rakenteen toimiva jäsentäminen ensiarvoisen tärkeää. Aliohjelmia ja komponentteja suunniteltaessa ovat rajapinnat keskeisessä asemassa. Riippumatta siitä, tehdäänkö systeemityötä perinteisillä vai oliomenetelmillä, tulee suunnittelun perustua kapselointiin ja informaation kätkemiseen.

Tunnistaminen on välttämätön edellytys tuotteenhallinnalle: se, mikä ei ole tunnistettavissa, ei myöskään voi olla hallittavissa.

Kokoonpanon hallinta

Kokoonpanon hallintaan kuuluu kuvauskannan turvaaminen, muutosten hallinta ja ohjelmiston generoinnin hallinta. Tuotteenhallinnan alaiseen kuvauskantaan talletettujen rakenneosien turvaamiseksi on tärkeää suojata kuvauskanta projekti- ja rakenneosakohtaisin käyttöoikeuksin. Tuotteenhallinnan tulee myös varmistaa, ettei samanaikainen kehittäminen aiheuta toteutetun piirteen katoamista, vaan pakottaa väliaikaisen variantin luomiseen.

Tuotteenhallinnalla ei juurikaan olisi merkitystä, jollei muutoksella olisi keskeinen rooli systeemityössä. Hallittu muutos edellyttää

- muutospyynnön tallettamista
- muutoksen arviointia tähän nimetyn henkilön / elimen (configuration Control Board = CCB) toimesta sekä päätöstä muutoksen toteuttamisesta / hylkäämisestä
- toteutettavien muutosten aikataulutamista.

Ohjelmoijan omaehtoisesti toteuttamasta hyvää tarkoittavasta muutoksesta saattaa aiheutua yllättäviä harmeja. Tämän vuoksi ohjelmakirjaston käyttöoikeuksilla tulee varmistaa, että ohjelmistoon ei tehdä muita kuin formaalisti hyväksyttyjä muutoksia.

Vaikkapa vain yhdelle asiakkaalle räätälöidyn ohjelmistonkin muutokset - myös virheen korjaukset, mikäli mahdollista - kannattaa niputtaa tietyin väliajoin tuotettaviin ohjelmistoversioihin. Muutosten seuraamisen kannalta on tärkeää, että sovitusta kehityksestä jää selkeät merkinnät tuotteenhallintaan: kuka muutti, mitä muutti, milloin muutti, mihin muutospyyntöön muutos liittyy.

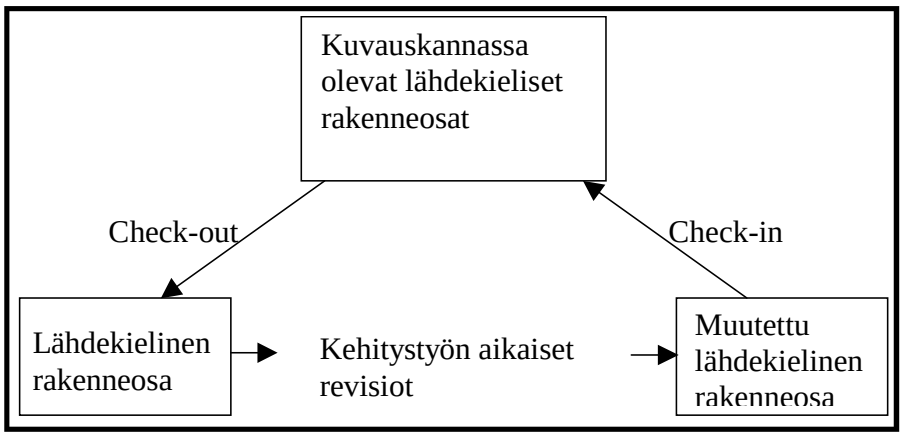
Ohjelmiston generoinnin tulee olla toistettavissa. Ajan ja laitteistoresurssien säästämiseksi, kaikki turhat vaiheet tulee ohittaa hyödyntämällä valmiiksi generoituja ajan tasalla olevia osia. Joskus on tärkeää, että ohjelmiston aiempi versio pystytään tuottamaan lähdekoodista. Tätä varten tulee rakenneosien vanhat versiot ja systeemimalli tallettaa kuvauskantaan. Systeemimallin tulee kattaa myös tarvittavien työkaluohjelmistojen versiot ja niiden toimintaa ohjaavat parametrit.

Kokoonpanon tilan seuranta

Kokoonpanon tilan seuranta kattaa ohjelmiston rakentamiseen ja sen valvontaan osallistuvien kannalta merkityksellisen tiedon tallettamisen ja raportoinnin. Toiminta edellyttää muutospyyntöjen ja erityyppisten rakenneosien (ohjelmien, moduulien, dokumenttien, käyttöohjeiden, ...) elinkaaren määrittelemistä.

Hyvän tilanseurantajärjestelmän tulisi pystyä vastaamaan esimerkiksi seuraavanlaisiin kysymyksiin:

- Mikä on yksittäisen rakenneosan tila?
- Onko tietty muutospyyntö hyväksytty?
- Mihin rakenneosiin kyseisen muutospyynnön toteutus vaikuttaa?



Kuva 4: Lähdekoodin hallinta merkitsee sitä, että kopioitaessa (check-out) ylläpidettävä rakenneosa kuvauskannasta, tuotteenhallinta merkitsee sen kehityksen alla olevaksi. Toinen kehittäjä saa rakenneosan ylläpidettäväkseen vain, jos hän muodostaa siitä variantin. Kun ylläpitäjä päivittää (check-in) muutetun rakenneosan takaisin kirjastoon, saa rakenneosa automaattisesti uuden versionumeron. Vanha versio säilyy edelleen kirjastossa.

- Milloin muutospyyntö hyväksyttiin ja kuka hyväksyi sen?
- Kuka toteutti muutoksen ja milloin muutostyö oli valmis? Kuka katselmoi työn? Kuka hyväksyi sen?
- Mihin rakenneosan versioon kyseinen muutos on toteutettu?
- Mitä eroja uudessa ohjelmistoversiossa on verrattuna edelliseen versioon?
- Kuinka monta muutospyyntöä esitetään kuukausittain ja kuinka suuri osa niistä hyväksytään toteutettavaksi?
- Kuinka monta virheilmoitusta vastaanotetaan kuukausittain?
- Minkälaiset syyt aiheuttavat pääosan virheistä? [Leon]

Ohjelmiston rakenneosia voidaan verrata pankkitileihin. Jokainen rakenneosaan kohdistuva tapahtuma ja suoritettu toimenpide tulee rekisteröidä. Täten rakenneosan yksittäisiä tapahtumia voidaan seurata esiintymisjärjestyksessä. [Leon]

Tarkastus ja katselmointi

Yleensä tuotteenhallinnan yhteydessä tuodaan esiin kolmenlaisia tarkastuksia: toiminnalliset (functional audit), kokoonpanon eheyden (physical audit) sekä tuotteenhallintaprosessin tarkastukset (configuration audit).

Toiminnallisessa tarkastuksessa todennetaan, että ohjelmiston toiminta vastaa sille asetettuja spesifioituja vaatimuksia. Kokoonpanon eheyden tarkastuksessa varmistetaan, että generoitu ohjelmisto tai sen osa sisältää kaikki siihen kuuluvat rakenneosat.

Prosessin tarkastuksella varmistutaan kuvatun tuotteenhallintaprosessin noudattamisesta sekä prosessin toimivuudesta. Tarkastuksessa huomion kohteena ovat mm.

- muutoksenhallintaprosessi
- muutospyyntöjen toteuttaminen
- hyväksyttyjen muutosten jäljitettävyys alkuperäisiin määrityksiin ja vaatimuksiin
- suunnittelukuvastojen saatavuus hyväksyttyjen muutosten toteutuksen tueksi
- suunnittelun yhteydessä tehtyjen ratkaisujen jäljitettävyys alkuperäisiin vaatimuksiin.

Julkistuksen hallinta ja toimitus

Vaikka uusi ohjelmisto tai ohjelmistoversio vastaisi kaikin puolin sille asetettuja vaatimuksia, pudottaisi huonosti hoidettu julkistus ja/tai toimitus pohjan muutoin laadukkaasti läpiviedyltä prosessilta.

Julkistukseen liittyy aina tiedote (release note), jossa on ainakin

- kuvaus ohjelmiston toiminta-ympäristölle asetetuista vaatimuksista
- ohjelmiston asennusohjeet
- ohjeet asennuksen onnistumisen varmistamiseksi suoritettavista testeistä
- ohjeet suoritettavista toimenpiteistä uuden version käyttöön siirryttäessä
- yhteenveto eroista aiempaan versioon verrattuna
- kuvaus julkistetussa versiossa mahdollisesti tiedossa olevista virheistä ja rajoituksista
- ohjeet siitä, miten ottaa yhteyttä toimittajaan virheiden raportoimiseksi, parannusehdotusten esittämiseksi tai ongelmatilanteissa. [Whitgift]

Julkistustiedotteen sisältö vaihtelee tapauskohtaisesti. Se antaa asiakkaalle ensivaikutelman toimituksesta ja on täten varteenotettava mielipiteen muokkaaja.

Asiakkaalle ei yleensä toimiteta kaikkia kuvauskannassa olevia rakenneosia (esim. määrittely- ja suunnitteludokumentteja). Ohjelmistosta saatetaan toimittaa eri asiakkaille erilaiset kokoonpanot. Tämän vuoksi toimituksen kokoamiseen tarvitaan asiakaskohtaiset komentojonot. Nämä, kuten ohjelmiston generoimiseen käytettävät systeemimallitkin ovat versioituvia tuotteenhallinnan alaisia rakenneosia.

Tuotteenhallinnan parhaita käytäntöjä (best practices)

Käytännön työn kannalta on tehokkainta hyödyntää kokemukseen perustuvia, hyväksi havaittuja menettelyjä. Ohessa eräs luettelo tuotteenhallinnan parhaista käytännöistä [White]:

- Talleta kehittämisprosessin tulokset varmaan ja turvallisen kuvauskantaan.
- Tarkista ja valvo rakenneosiin tehtäviä muutoksia.
- Yhdistele rakenneosat versioituviksi komponenteiksi.
- Luo projektin tarkistuspisteissä vaihetasot seuraavan kehitysvaiheen pohjaksi.
- Tallenna muutospyynnöt ja seuraa niiden tilaa.
- Integroi tiettyyn muutokseen liittyvät eri rakenneosien versiot johdonmukaiseksi loogiseksi kokonaisuudeksi (change set) Muutamat

tuotteenhallintavälineet tukevat ns. muutosjoukkoihin perustuvaa mallia. Esimerkiksi koodikatselmusta suoritettaessa, voidaan kirjastosta valita yhdellä komennolla katselmuksen kohteeksi oikeat versiot kaikista niistä rakenneosista, joihin on tehty ylläpitoa tiettyyn muutospyyntöön liittyen.

- Pidä työtila vakaana ja johdonmukaisena: Ohjelmoija kopioi check-out -toiminnolla muutettavat lähdeohjelmat ohjelmakirjastosta yksityiseen työtilaansa, jossa muutokset toteutetaan. Vakaa työtilamalli mahdollistaa työskentelyn, ilman että toiset samanaikaiset muutokset häiritsevät testaamista. Jossain vaiheessa tehdyt muutokset tulee kuitenkin testata muita rinnakkaisia muutoksia vasten. Johdonmukainen malli tarkoittaa, että kehittäjän päivittäessä työtilaansa tällaista testausta varten, työtilaan syntyy päivityksen tuloksena johdonmukainen ja toimiva rakenneosien versiojoukko.
- Tue lähde-elementtien ja komponenttien samanaikaista muuttamista. Aiemmat tuotteenhallintatyökalut edellyttivät peräkkäistä kehittämistä (seuraava ohjelmoija pystyi ottamaan rakenneosan muutokseen vasta, kun edellinen oli päivittänyt sen check-in -toiminnolla takaisin kuvauskantaan).
- Integroi varhaisessa vaiheessa ja usein.Integrointivaiheessa esiintyy usein väärinkäsityksistä johtuvia ongelmia rajapintojen kanssa. Näiden mahdollisimman varhainen havaitseminen on projektin aikataulussa pysymisen kannalta ensiarvoisen tärkeää. Aikainen integrointi on jossain määrin ristiriidassa vakaan työtilan kanssa. Tähän dilemmaan tulisi löytyä tasapainoinen ratkaisu, jolla ei ole negatiivista vaikutusta tuottavuuteen.
- Varmista, että ohjelmiston rakentaminen on toistettavissa.

Miten organisoida tuotteenhallinta

Tuotteenhallinta edellyttää suunnittelua ja laaditun suunnitelman noudattamista. Pienissä projekteissa riittänee viittaus yleisiin toimintaohjeisiin tai laatukäsikirjaan. Näissä tuotteenhallintasuunnitelma voi olla osa projektisuunnitelmaa. Suurissa projekteissa, tai kuvattaessa valmisohjelmiston tuotteenhallintaa, on perusteltua laatia erillinen tuotteenhallintasuunnitelma. Kirjallisuudesta löytyy malliesimerkkejä suunnitelman sisällöstä. Suunnitelma tulee kuitenkin aina sovittaa ja laatia siten, että otetaan huomioon kohteena olevan ohjelmiston erityispiirteet ja kehittämisympäristö.

Pienissä projekteissa tuotteenhallinta voidaan hoitaa hakemistorakenteisiin ja käyttöjärjestelmän toimintoihin tukeutuen. Kuitenkin erään näkemyksen mukaisesti vähintäänkin lähdekoodin hallinta on syytä automatisoida, mikäli samaa ohjelmistoa ylläpitää useampi kuin yksi kehittäjä [Leon]. Automatisointi on ainoa varma keino välttää samanaikaisen kehittämisen yhteydessä helposti tapahtuva tiedon katoaminen.

Isoissa systeemityöprojekteissa nykyaikaista tehokasta tuotteenhallintaa on vaikea ajatella ilman välinetukea. Manuaalisena tuotteenhallinta on aikaa vievää ja virhealtista. Tuotteenhallintaohjelmistoja on tällä hetkellä runsaasti tarjolla. Useat näistä muodostavat tuoteperheitä, joista on valittavissa omia tarpeita vastaavat, tiettyjä tuotteenhallinnan osa-alueita automatisoivat välineet. Näyttää siltä, että kiinnostus näitä työvälineitä kohtaan on viimeaikoina merkittävästi lisääntynyt.

Eräs kasvava automatisointia edellyttävä tuotteenhallinnan kohde on verkkosivujen hallinta. Yksittäisellä konsernilla saattaa olla kymmeniä tuhansia verkkosivuja tai niiden komponentteja hajautettuna ympäri maailmaa. Apuun tarvitaan systeemityön ammattilaisia ja sopivia välineitä. Mainostoimistot ja uusmediayritykset eivät selviä tästä haasteesta ilman ammattilaisten apua. [Dart]

Tuotteenhallinnan käyttöönotto

Tuotteenhallinnan käyttöönotto on muutos. Ihmiselle luonteenomaista on vastustaa kaikkinaista muutosta. It-ammattilaiset verhoavat vastahakoisuuden usein kiireeseen tai vetoavat asiakkaisiin, jotka eivät tule hyväksymään formaalia muutosprosessia ja muutosten niputtamista määräajoin tuotettaviin versioihin. Oman kokemukseni mukaan asiakkaat ovat varsin joustavia ja he ymmärtävät järkevät toiminnan muutokset, kun ne heille perustellaan. Itse asiassa laatutietoiset asiakkaat edellyttävät toimittajiltaan suunnitelmallista toimintaa.

Vaikka tuotteenhallintaa varten otettaisiin käyttöön väline, on kehitystyötä tekevien henkilöiden sitouttaminen tärkeää. Tuotteenhallinta on viimekädessä kurinalaista toimintaa. Sovituista käytännöistä poikkeaminen vesittää välineen mukanaan tuoman hyödyn, mikä näkyy turhaan selvitystyöhön ja korjaamiseen hukattuna työaikana ja asiakastoimituksissa ilmenevinä virheinä. Tästä syystä tuotteenhallinnan käyttöönottoa tulee valmistella huolella ja pyrkiä muokkaamaan mielipiteitä myönteiseksi jo mahdollisimman varhaisessa vaiheessa.

Vaikka välineitä on runsaasti tarjolla ja niissä tuntuu päälisin puolin olevan täsmälleen samat ominaisuudet, on valintaan silti syytä uhrata aikaa. Parhaan hyödyn saavuttamiseksi tuotteenhallinnan tulisi integroitua saumattomasti kehitysympäristöön. Kehitysympäristön työkalujen integrointiin tähtäävien standardien kehittämiseen on panostettu useissa kansainvälisissä projekteissa (esim. PCTE) [Sommerville]. Näillä ei ilmeisesti kuitenkaan ole saavutettu toivottua käytännön hyötyä, sillä merkittävät työkaluvalmistajat ovat kartuttaneet tuotteenhallintaosaamistaan ostamalla tähän erikoistuneita pieniä yrityksiä. Saatuaan näin koko kehittämisen kentän hallintaansa, tarjoavat isot välinevalmistaja omia integroituja kehittämisympäristöjään.

Tuotteenhallinta on laatua

Tuotteenhallinnan, laadunvarmistuksen ja projektinhallinnan välinen raja on paikoin kuin veteen piirretty viiva. Silti tuotteenhallinta on merkittävä erillinen systeemityötä tukeva osa-alue. Laatustandardit edellyttävät sen ohjeistamista.

Kun vuosia sitten osallistuin yrityksessäni järjestettyyn laatukoulutukseen, aloitti vetäjä tilaisuuden lausumalla: ’Laatu vähentää stressiä’. Uskon, että hyvin organisoitu ja toteu-

tettu tuotteenhallinta on merkittävä kiireen ja stressin vähentäjä kaikille kehittämistyöhön osallistuville it-ammattilaisille.

Lähteet

[Dart]
Dart, Susan. Configuration Management. The Missing Link in Web Engineering. Artech House. London. 2000.

[Estublier]
Estublier, J., Favre J., Morat, P. Toward SCM/PDM Integration? System Configuration Management: Proceedings / ECOOP'98 SCM-8 Symposium. Boris Mangnuson (ed.). Springer-Verlag. Berlin 1998.

[Hamina-Mäki]
Hamina-Mäki, Eija. Systeemityölehti. 4 / 2000.

[Kelly]
Kelly, Marion V. Configuration Management: The Changing Image. McGraw-Hill Book Company. Berkshire 1996.

[Leon]
Leon, Alexis. A Guide to Software Configuration Management. Artech House. London. 2000.

[Mohler]
Mohler, A. Variants: Keeping things Together and Telling Them Apart. Configuration Management. Walter F. Tichy (ed.). John Wiley & Sons. Chichester 1995.

[Sommerville]
Sommerville, Ian. Software engineering. Fifth edition. Addison-Wesley. 1995.

[White]
White, Brian A. Software Configuration Management Strategies and Rational Clear Case: a practical introduction. Addison-Wesley. 2000.

[Whitgift]
Whitgift, David. Methods and Tools for Software Configuration Management. John Wiley & Sons. Chichester. 1991.

Ulla Vanhanen,
Helsingin liiketalouden ammattikorkeakoulu,
ulla.vanhanen@helia.fi

Sytykkeen testauskerho

Sytykkeen testauskerho kokoontui 10.4. Nokia Research Centerin tiloissa ja yli 40 testausammattilaista kuunteli arvostettujen testauskonsulttien, Mark Fewsterin ja Lloyd Rodenin esityksiä otsikoilla "Understanding the Value of Testing (or Why You Shouldn't Test!)" ja "It's a Bugs Life". Ajatuksia herättävät esitykset eivät ehtineet kuitenkaan polkaista laajempaa keskustelua työpäivästään väsyneissä testaajissa ennen kuin jo ideoimme kerhon tulevaa toimintaa.

Sovimme seuraavan kokoontumisen Turkuun, Perkin-Elmerin tiloihin maanantaina 11.6. illalla, jonne Päivi Inki meidät auliisti kutsui - Kiitos! Teemana on "Testauksen automatisointi käytännössä" ja puhujina on automatisoinnin parissa pakertaneita, kokeneita testaajia ja muita asiaa tuntevia tahoja. Tilaisuuden aikataulu sovitetaan sopivaksi junalla liikkuville, joten myös hesalaisilla ei ole syytä jäädä pois ;-) Tarkemmin tilaisuudesta tiedotetaan kerhon postilistalla, jolle pääsee ilmoittautumalla Ekille osoitteella

erkki.poyhonen@nokia.com.

Tervetuloa!

Kuva 2: Lloyd Roden vangitsi yleisön mielenkiinnon myöhäisestä ajankohdasta huolimatta.

ZigZag - tulevaisuuden systeemi

Antti-Juhani Kaijanaho,
Jyväskylän yliopiston tietotekniikan laitos, ZigZag-projekti

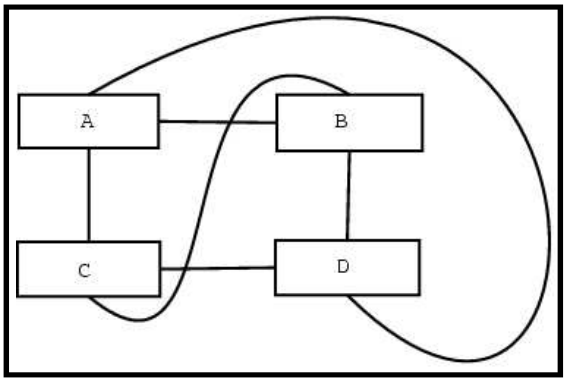
Jyväskylän yliopiston ZigZag-projekti tutkii Tuomas Lukan johdolla uusia tiedonhallintamenetelmiä Ted Nelsonin ZigZag-systeemimallin pohjalta. Työmme näkyvin tulos on GZigZag, tekemämme ZigZag-implementaatio, mutta sivutuloksena on syntynyt muunmuassa yleiskäyttöinen visual object -kirjasto.

Tämä artikkeli kertoo, mikä ZigZag on, esitellen ensin sen tietomallin ja mediamallin, sekä lopuksi sen sovelluksia ja tulevaisuuden näkymiä.

ZigZagin tietomalli

Esittelen ZigZagin avaruusajattelun vaiheittain: ensin yhdellä dimensiolla, sitten kahdella dimensiolla ja lopulta sellaisena kuin se on: ZigZag-avaruutena, jossa dimensioita on rajattomasti.

Soluilla on naapureita. Kullakin solulla on kaksi napaa: positiivinen napa ja negatiivinen napa. Kaksi solua ovat naapureita, jos toisen solun positiivinen napa on kytketty toisen negatiiviseen napaan. Tällä tavalla voidaan muodostaa solujonoja, rangeja (engl. rank): solu kytkeytyy toiseen soluun, joka kytkeytyy kolmanteen soluun ja niin edelleen. Koska kullakin solulla voi olla vain kaksi naapuria (toinen positiiviseen ja toinen negatiiviseen napaan kytkeytyneenä), muodostuu todellakin jonoja eivätkä jonot pääse haarautumaan. Mikään ei toisaalta estä rengasmaisia jonoja eikä myöskään sitä, että solu kytkeytyisi itseensä.



Kuva 1: Yksinkertainen ZigZag-avaruus, jossa on soluja ja kytkentöjä kahdella dimensiolla. Pystysuuntaisella dimensiolla on rengasmainen ranki, jolla solu A on kytketty positiiviseen suuntaan soluun C, solu C soluun B, solu B soluun D ja solu D taas soluun A. Vaakasuuntaisella dimensiolla on kaksi rangia: A on kytketty positiiviseen suuntaan B:hen ja C D:hen.

Lisätäänpä kaikkiin soluihin toinen napapari, joka on erilainen kuin alkuperäinen mutta kuitenkin jokaisessa solussa samanlainen. Soluilla voi olla tämän jälkeen neljä naapuria, ja voidaan muodostaa kahdenlaisia rangeja: sellaisia, joissa kytkennät tapahtuvat alkuperäisiä napapareja käyttäen, ja sellaisia, joissa ne tehdään uusien napaparien avulla.

Nämä napaparityypit ovat nimeltään *dimensioita*, ja ne ovat toisistaan täysin riippumattomia. Kukin solu voi kytkeytyä kullakin dimensiolla korkeintaan kahteen muuteen soluun: kerran positiivisella ja kerran negatiivisella navallaan. Näin avaruuteen muodostuu suuntia: kullakin dimensiolla on *positiivinen* suunta (engl. posward direction) ja *negatiivinen* suunta (engl. negward direction).

ZigZag-avaruudessa navigoinnin kannalta suunnilla on suuri merkitys: käytännössä ainoa tapa liikkua avaruudessa on kulkea solusta toiseen johonkin suuntaan.

Kaikki ZigZag-kytkennät ovat suunnattuja, mutta niitä pitkin pääsee aina kulkemaan molempiin suuntiin. Tämä on tärkeä ero verrattuna tavanomaisten ohjelmointiympäristöjen osoittimiin, jotka ovat myös suunnattuja, mutta niitä pitkin pääsee kulkemaan vain yhteen suuntaan.

Kaksiulotteinen ZigZag-avaruus (tai useampiulotteisempikin, jos vain kaksi dimensiota kiinnostavat) on tapana piirtää niin, että kukin solu on laatikko ja kytkennät ovat viivoja. Toisella dimensiolla positiivinen suunta on aina alas ja toisella aina oikealle. Tässä olennaisinta on se, mistä päin laatikkoa viiva lähtee eikä se, minkä suuntainen viiva pääasiallisesti on; suuntaa käytetään vain dimension yksilöintiin. Esimerkiksi kuvassa 1 on neljä solua, jotka on kytketty kahdella dimensiolla yhteensä kolmeksi rangiksi, joista yksi on rengasmainen.

ZigZag-avaruudessa on rajaton määrä dimensioita. Käytännössä koskaan ei tarkastella kuin muutamaa dimensiota kerrallaan. Ne dimensiot, joita ei tarkastella, voivat silti olla kiinnostavia joissakin muissa tilanteissa.

ZigZag-soluilla on kytkentöjen lisäksi sisältöä. Sisältö voi olla tekstiä, kuvaa, ääntä - periaatteessa mitä vain, mitä referentiaalisena mediana voi käsitellä.

Soluja ja dimensioita käytetään tiedon hallintaan seuraavasti: Solu on tiedon jakamaton yksikkö. Se sisältää yleensä suurin piirtein niin paljon tietoa kuin taulukkolaskennankin solu. Kullakin dimensiolla on jokin merkitys, jokin solujen välinen suhde, jota se ilmaisee kytkemällä soluja toisiinsa. Usein tietty rakenne käyttää useita dimensioita ja mahdollisesti myös sisällöttömiä apusoluja ilmaistaakseen jotain monimutkaisempaa suhdetta. Esimerkiksi tavanomainen yhdestä-moneen-suhde voidaan tilanteesta riippuen tehdä joko yhdellä tai kahdella dimensiolla, ja yleinen monesta-yhteen-osoitin, jossa mikä tahansa solu voi olla sekä osoittajana että osoittamisen kohteena, tarvitsee kolme dimensiota

Kts. kuva 2.

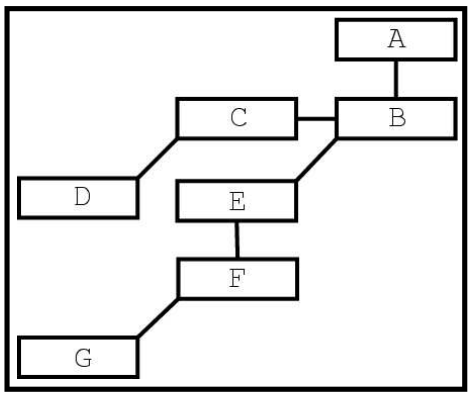
Referentiaalinen media

Referentiaalisen median malli on peräisin Ted Nelsonin legendaarisesta Xanadu-projektista, joka tutkii ja kehittää alkuperäistä ja nykyisiä reilusti vahvempaa hypertekstijärjestelmää. Siinä ideana on käsitellä itse mediapalojen sijasta niiden osoitteita.

Kun mediaa, esimerkiksi tekstiä, syötetään referentiaalista mediaa käyttävään systeemiin, esimerkiksi GZigZagiin, sille annetaan heti pysyvä, maailmanlaajusesti yksikäsitteinen osoite. Tämä osoite ei kerro, missä kyseinen mediapala on, vaan missä se on systeemiin tuotu. Näin esimerkiksi, kun minä kirjoitan tekstiä, sille ä-kirjaimelle, jonka juuri kirjoitin, annetaan osoite. Sille toiselle ä-kirjaimelle, jonka kirjoitin vähän myöhemmin, annetaan toinen osoite. Näin kaikki, mitä systeemiin kirjoitetaan, kirjoitetaan ikäänkuin loppumattomalle paperirullalle (*permascroll*).

Kun juuri kirjoittamaani tekstinpalaa halutaan käyttää esimerkiksi ZigZag-solun sisältönä, käytetään *jaetta* (engl. span), joka on viite sellaisten mediapalojen joukkoon, jotka on tuotu systeemiin perätysten. Esimerkiksi voitaisiin muodostaa jae, joka viittaa juuri kirjoittamaani tekstinpalaan "juuri kirjoittamaani tekstinpalaan".

Jakeiden käyttäminen mahdollistaa monia hyödyllisiä asioita, mm. saman mediapätkän käytön seuraamisen sekä median uudelleenjärjestämisen hamottamisen.



Kuva 2: Yleinen osoitinrakenne: Solu G osoittaa soluun E, solut D ja E osoittavat soluun A. Solut B, C ja F ovat apusoluja. Kuvassa on kolme dimensiota: vaakasuuntainen, pystysuuntainen ja diagonaalinen.

Näkymät

Oleellisen osan ZigZagia avaruudellisen tietomallin lisäksi muodostavat _näkymät_. Datamallin ja näkymän erottaminen on tuttua MVC-mallista, mutta ZigZagissa (erityisesti GZigZagissa) se on viety äärimmilleen.

Näkymän vastuulla on ZigZag-rakenteessa olevan datan näyttäminen. Näkymä voi olla täysin geneerinen, jolloin se näyttää mitä vain dataa (ks. esim. kuva 3), mutta se voi myös olla varsin eriytynyt (ks. esim. kuva 4). GZigZagissa näkymän kirjoittajalla on määräysvalta piirrettävän alueen jokaiseen pikseliin: näkymä saa maalata näytön kuten haluaa. Ainoa rajoite on tarkoituksenmukaisuus.

GZigZagissa näkymän vaihtaminen on myös jouhevaa: näkymien välillä animoidaan samat kohteet samoiksi kohteiksi. Tämän helpoksi tekevä Vob-teknologia on kehitetty projektissamme ja on riippumaton GZigZagista.

Sovelluksia ja tulevaisuuden näkymiä

Tällä hetkellä GZigZag on prototyyppiasteella, ja sovelluksista on olemassa korkeintaan demoja. Yksi tekemämme sovellusdemo esittelee mahdollista keskustelujärjestelmää, johon liittyy kunkin keskustelijan omakuva sekä niiden tuotteiden esittely, joita keskustelu koskee (kuva 4). Olemme myöskin toteuttaneet GZigZagin päälle kuvankäsittely- ja raytacing-toiminnallisuutta. Kehitteillä on myös tekstieditori Nile ("Nile Is a Literate Editor") sekä ohjelmointikieliä ja -ympäristöjä.

Tällä hetkellä kirjoitamme GZigZagia uusiksi Brooksin ohjeen "plan to throw one away" hengessä. Uuteen versioon tulee ohjelmansisäisten parannusten - erityisesti kunnollinen regressiotestijoukko - lisäksi uutta, esimerkiksi yhtenäinen, referentiaaliselle medialle perustuva verkkotuki.

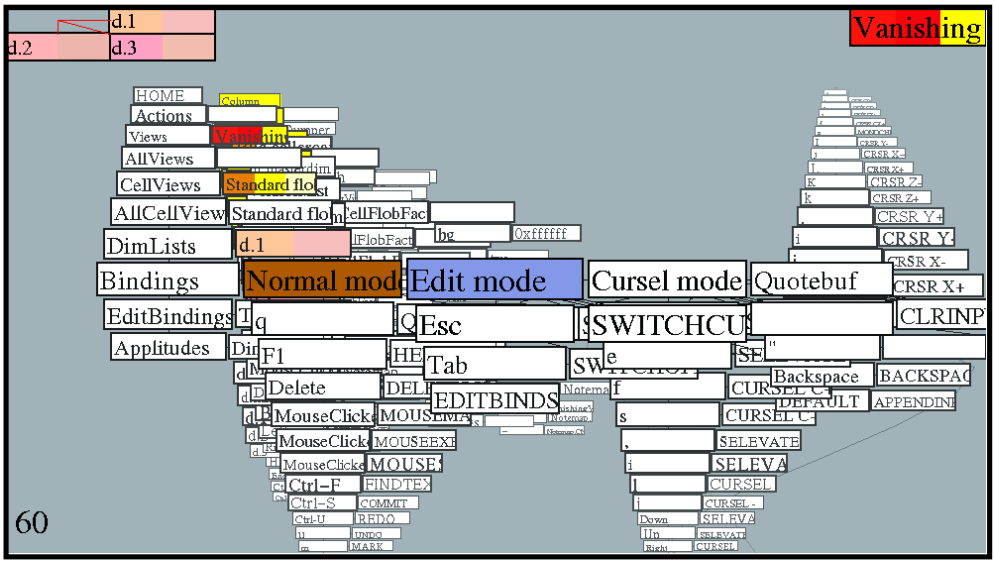
Sujuva systeemityö?

On vaikea kuvitella, miten ZigZag voisi *nyt* auttaa tekemään systeemityötä sujuvammaksi, mutta kunhan se kypsyy, uskomme sen auttavan kaikkia tietotekniikan osa-alueita. Yhtenäinen korkeahkon tason tietorakenne auttaa esimerkiksi pääsemään eroon osoitinvirheistä sekä integroimaan eri sovellukset yhteen.

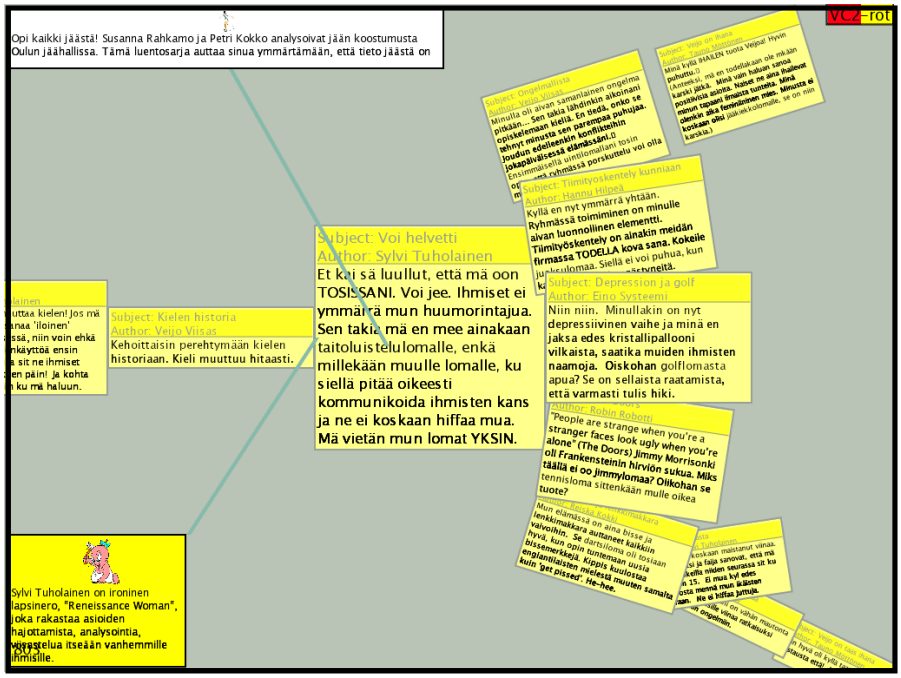
Lopuksi

GZigZag on vapaa ohjelma, ja kaikki projektimme julkinen materiaali (muunmuassa itse Javalla kirjoitettu ohjelma) on saatavissa osoitteesta http://www.gzigzag.org/ . Otamme mielellämme vastaan kommentteja ja ideoita, ja mukaankin saa tulla.

Antti-Juhani Kaijanaho,
tutkimusavustaja,
ZigZag-projekti,
Jyväskylän yliopiston tietotekniikan laitos,
gaia@iki.fi



Kuva 3: Eräs perusnäkymä GZigZagin ohjausrakenteisiin.



Kuva 4: GZigZag-pohjainen keskustelualuedemo

Luovuus ja käytettävyys verkkopalvelun kehittämisessä

Jussi Mertanen,
TietoEnator Digital Finance Oyj

Käytettävyys on sana, joka sykkii jokaisen verkkopalvelun kehittäjän takaraivossa. Myös palvelun visuaalisessa suunnittelussa käytettävyys otetaan aiempaa paremmin huomioon. Kaikki lähtee kuitenkin hyvästä konseptisuunnittelusta. TietoEnator Digital Finance Oy:n luovan suunnittelun yksikkö, Creative Design keskittyy kolmeen osaamisalueeseen: verkkomedian konsepteihin, käytettävyyteen ja visuaaliseen suunnitteluun.

Konseptisuunnittelija on kehittävä keräilijä

Mitä oikeastaan on luova konseptisuunnittelu? Konseptisuunnittelun perusajatus on luoda verkkopalvelun kattoidea. Informaatio on konseptisuunnittelijan ruokaa. Tehtävä vaatii tiedon keräämistä ja yhteistyötä systeemityön, mainonnan ja myös liiketoiminnan strategian määrittelijöiden kanssa. Konseptisuunnittelija on agitaattori, jonka tehtävä on tuoda suunnitteluun loppukäyttäjän elämyksellinen näkökulma. Kun verkkopalvelun tasoa arvioidaan, ratkaisevat käyttäjän saamat kokemukset ja hyödyt. Konseptisuunnittelijalla onkin tärkeä asema esimerkiksi tyyppikäyttäjän mallintamisessa.

Konseptisuunnittelija kerää ja analysoi taustatiedon sekä määrittelee perusidean, kun verkkopalvelua kehitetään. Visuaalisten suunnittelijoiden työ alkaa konseptisuunnittelun loppupuolella. Perinteisesti konseptisuunnittelijoita on kahta lajia, teknisiä ja luovia. Creative Design -ryhmän konseptisuunnittelijat toteavat usein toimivansa näiden välimaastossa.

Creative Design -ryhmä soveltaa myös mainonnan ja viestinnän maailmassa käytettyjä apukeinoja tyyppikäyttäjän mallintamiseen. Mainonnassa kohderyhmistä on jo pitkään hankittu syvällistä tietoa esimerkiksi kvalitatiivisen tutkimuksen avulla. Olemme tuomassa tämän alueen kokemustamme ja osaamistamme myös verkkopalveluiden kehittämiseen. Kvalitatiivisessa tutkimuksessa saadaan syvällisempää tietoa käyttäjien odotuksista, motiiveista ja käyttötilanteista. Laadullista tutkimusta voidaan soveltaa sekä kuluttaja- että b-to-b-palveluja arvioitaessa. Kvalitatiivinen ote hyödyntää erilaisia haastattelutekniikoita tai ryhmäkeskusteluita, ja lähtökohta on humanistinen.

Konseptisuunnittelun tuloksena syntyy aina dokumentti, joita voivat olla palvelun kuvaukset, rakennekaaviot tai analyysit. Kaikilla palvelun kehittämiseen osallistuvilla on näin yhtenäinen käsitys tulevasta palvelusta.

Käytettävyys on kattoajatus

Käytettävyydellä tarkoitetaan yleensä kuinka hyvin jonkin järjestelmän, kuten verkkopalvelun, toimintoja voidaan käyttää haluttuun tarkoitukseen. Käytettävyys kertoo kuinka onnistunutta toimintojen käyttö on. Käytettävyys tulisi olla mukana koko verkkopalvelun kehittämisen kaaressa. Useista yrityksistä löytyy vanhoja esimerkkejä palveluista, jotka kehitettiin protovaiheeseen asti, ja hetkeä ennen julkistamista joku asiantuntija kävi hätäisesti heuristisen listan läpi. Näin voitiin todeta, että käytettävyystekijät on otettu huomioon. Creative Design -ryhmän käytettävyysasiantuntijat ovat kuitenkin mukana projektin alusta asti . Yksi selkeä työskentelymme etu on kiinteä rajapinta tekniseen toteutukseen. Luovassa tiimissä on mukana myös asiantuntijoita, jotka ovat siirtyneet ryhmään tekniseltä puolelta. Samoissa tiloissa toimiminen mahdollistaa jatkuvan vuorovaikutuksen luovien suunnittelijoiden sekä teknisten asiantuntijoiden kanssa. Kaikki tarvittava asiantuntemus löytyy saman katon alta.

Puhuttaessa koko verkkopalvelun kehittämisestä, avainsanoja ovat käytettävyyden suunnittelu ja testaus sekä verkkopalvelun elementtien kehittely. Web-maailmassa työskentelevän graafikon on tunnettava käytettävyyden kultaiset säännöt ja sovellettava niitä. Värien ja fonttien valinnassa otetaan huomioon asiakkaan brandin lisäksi käytettävyys. Prototyyppivaiheessa astuu mukaan käytettävyystestaus, jota tehdään kahdella tasolla, analyysina ja käyttäjätestauksena. Perinteinen heuristinen analyysi tarjoaa hyvän työkalun tähän vaiheeseen. Kun tarvitaan syvällisempää tietoa oikeiden loppukäyttäjien toiminnasta, ryhdytään käyttäjätestaukseen. Näiden lisäksi Creative Design -ryhmä on kehittänyt peruskäyttäjätestausta kevyemmän, yhdistävän testausmenetelmän, josta asiakkaat ovat olleet erittäin kiinnostuneita. Tässä menetelmässä heuristisen analyysin tueksi suoritetaan myös suuntaa-antava peruskäyttäjätestaus. Kun käytettävyys on otettu huomioon palvelun kehittämisen alusta asti, ovat analyysissa tai testissä löytyvien puutteiden korjaukset tai sivujen muutokset yleensä vähäisiä ja helppoja toteuttaa.

Graafikko muuttuu ajan mukana

Perinteinen graafikko ei ole verkkomaailmassa enää pelkkä graafikko. Hän on monitaituri, jonka on pysyttävä ajan hermolla. Perusedellytyksenä ovat tietenkin aina visuaalinen lahjakkuus ja alan koulutus, sillä keskeiset graafisen suunnittelun periaatteet eivät ole ajan saatossa muuttuneet. Verkkopalveluiden kehittämisessä graafikon työssä korostuvat kuitenkin myös tekniikan ja käytettävyyden tuntemus.

Graafiset suunnittelijamme ovat kokeneet visuaalisuuden rinnalla erityisen haasteellisiksi kysymyksiksi muun muassa työmäärän arvioinnin sekä teknisen tietämyksen . Visuaalisen suunnittelun työmäärän arviointia vaikeuttaa usein

luovan ehdotuksen elävä luonne ja ennalta arvaamattomat muutokset. Kun graafikoiden kanssa keskustelee työmäärään arvioinnista, esille nousee usein esimerkkejä, joissa asiakkaan tai visuaalisen suunnittelun tilaajan “ihan pikku juttu“ onkin osoittautunut erityisen aikaa vieväksi ja kimurantiksi tehtäväksi. Graafikko on usein keskeisessä roolissa pohdittaessa palvelun toiminnallisuuden visuaalista esittämistä. Web-graafikon työ eroaa ”perinteisen” graafikon toimenkuvasta siten, että juuri tietämys toiminnallisuudesta ja sen arvioinnista on tärkeä osa työtä Ei liene ihme, että monet loistavat web-graafikot ovatkin itse asiassa peruskoulutukseltaan teollisia muotoilijoita.

Jatkuva kiivas opiskelu ja kouluttautuminen ovat edellytyksiä ammattitaidon ylläpitämiseen ja kehittämiseen. Usein intohimoinen graafikko opiskelee vapaa-ajallaankin visuaalisen suunnitteluun liittyviä uusia asioita. Jo pelkästään ajantasaisten ohjelmaversioiden opiskeluun kuluu aikaa. Creative Design -ryhmässä panostamme myös visualistien jatkuvaan käytettävyyskoulutukseen.

Jussi Mertanen,
taiteellinen johtaja,
TietoEnator Digital Finance Oy

---

*)Hannu Takala, mediakasvatuksen opiskelija, teki ensimmäisen prototyypin suunnittelun t t t t t tttt t tttt tttt t t t t tt tt tt tttt t t tttt t tt tttt tt t t” –versiossa on otettu yhdistyksen mainio sanayhtäläisyys SYTYKE - joka tässä kuvattu perinteisesti tulitikkuna - uuden kehityksen ja älynväläysten pohjaksi, joita symboloi sähkölamppu. Näin vanhasta kehittyy systeemityön kautta uutta.

SYTYTTÄÄKÖ vai peräti kuumentaa?

Nyt SINULLA on mahdollisuus suunnitella yhdistyksellemme oma logo! Osallistu suunnittelukilpailuun joko parantelemalla ylläolevaa ehdotelmaa *) tai kehittelemällä aivan toisenlainen. Virikkeitä voit kukaties saada myös allaolevista

TTL:n logosta sekä muinaisesta Sytyke-ÄSSÄstä.

S

Yhdistyksen johtokunta palkitsee kolme parasta ehdotusta ja pidättää samalla oikeuden jatkokehitellä niitä. Voittajille ilmoitetaan tuloksesta elokuussa 2001 ja palkitut ehdotukset julkaistaan web-sivuillamme sekä Systeemityö-lehdessä.

Nyt vapaasti mielikuvituksen siiville ja lähettämään rohkeasti ehdotuksesi osoitteeseen Helena.Venalainen@osuuspankki.fi. Toimithan ennen juhannusta 2001!

KIITOS jo etukäteen!
Sytyke ry:n web-toimikunta

Organisaatioanalyysi verkkopalvelun kehittämisessä

Eija Kalliala,
Helsingin liiketalouden ammattikorkeakoulu Helia

- ”Projekti ei pysy aikataulussa. Emme ole saaneet mitään aineistoa yritykseltä verkkopalvelun rakentamiseen. Yrityksellä on vain muutama asiakas, eikä yksikään heistä käytä verkkoa.”
- ”Tuntuu turhauttavalta rakentaa sähköistä kauppapaikkaa yhden hengen yritykselle, jossa ei ole edes tietokoneita käytössä.”
- ”Olisihan se tietokanta ehdottamallanne tavalla järkevämpi rakenteeltaan, mutta ei sitä enää voi muuttaa kun koodi on jo valmiiksi kirjoitettu. Tehdään nyt jotain ja toteutetaan muutokset verkkopalvelun käyttönoton jälkeen vaiheessa kaksi.”
- ”Me olemme tehneet vaikka kuinka paljon töitä, mutta ei se näy missään, kun asiakas ei olekaan tarvinnut toteuttamiamme hienoja piirteitä. Olemme joutuneet karsimaan jo toteutettuja asioita verkkopalvelusta pois.”
- ”Asiakas ei tiedä itsekään, mitä haluaa, joten projektimme ei etene.”
- ”Asiakkaan edustajien halut muuttuvat koko ajan ja tuntuu, etteivät he muista, mitä viimeksi olemme sopineet, joten joudumme koko ajan muuttamaan jo tekemäämme.”

Olen neljän vuoden ajan ohjannut ammattikorkeakoulun opiskelijoiden verkkopalveluprojekteja, joita tehdään todellisille yrityksille tai organisaatioille. Valtaosa projekteista saa aikaan kunnollisen organisaatioanalyysin ja sen pohjalta laaditun asiakasta tyydyttävän verkkopalvelun prototyypin tai monesti jopa käyttönotetun toimivan verkkopalvelun. Edellä luettelemani kommentit on poimittu tavalla tai toisella epäonnistuneiden projektien selityksistä ja loppuraporteista. Useimmat ammattikorkeakoulun opiskelijoista ovat työelämässä, monet jopa verkkopalveluja rakentavissa yrityksissä. Tällaisia opiskelijoita on ollut myös noissa epäonnistuneissa projekteissa, joten kommentit heijastavat opiskelijaprojekteja laajemmin verkkopalvelujen hektisen rakentamisen ongelmia.

Vähän historiaa

Noin viisi vuotta sitten multimediaa tehtiin pääasiassa cd-romeille. Suunnittelun periaatteista puhuttiin, sovelluksen idea ja rakenne olisi pitänyt kuvata paperilla ennen toteutusta. Mutta kun työkalut olivat helppoja, monet aloittivat suoraan koodaamisesta ja käyttöliittymästä sovelluksen rakenteen mallintamisen sijaan. Näyttävä sovellus syntyi nopeasti, poltettiin rompulle ja jaettiin käyttäjille. Ylläpitoa ei mietitty. Jotkut asiantuntijat jopa sanoivat, että multimediasovelluksia ei valmistumisen jälkeen ylläpidetä. Myöhemmin jouduttiin pahoittelemaan, että näyttävässä multimediaesittelyssä yrityksen tunnusluvut olivat parin vuoden takaisia, kun uudempia ei ollut voitu esittelyrompulle päivittää.

Multimedia ja yritysesittelyt siirtyivät hiljalleen Internetiin web-sivuille. Ohjeita sivujen lay-outeista, värien käytöistä, marginaaleista, palstoituksesta, kuvista ja niiden koosta, tekstin määrästä sivulla, turhan skrollauksen ja klikkailun välttämisestä jne. on runsaasti. Nämä ovat tärkeitä asioita, mutta yrityksen verkkopalvelun suunnittelua EI aloiteta sivujen lay-outeista. Tämän oivaltaminen tuntuu vaativan ruuvin vääntämistä päässä uuteen asentoon: verkkopalveluprojekti on samanlainen prosessi kuin mikä tahansa systeemityöprojekti, jossa selvitetään, määritetään, suunnitellaan ja toteutetaan organisaatiolle uusi sovellus. Verkkopalveluprojekteissa nimitän selvitys- ja määritysvaiheita organisaatioanalyysiksi. Joskus tuntuu siltä, että uusien sovellusten rakentamisessa olisi palattu vastaavalle tasolle, jolla perinteisessä systeemityössä oltiin 1970-luvulla: irrallisia sovelluksia rakenneltiin ilman suunnittelua tai kokonaisnäkemystä.

Onnistunut verkkopalveluprojekti

Onnistunut verkkopalveluprojekti edellyttää ennen toteutusta ja käyttöönottoa:

- kunnollista projektisuunnitelmaa aikatauluineen, riskianalyyseineen ja kustannusarvioineen
- organisaatioanalyysia
- organisaation nykytilan kartoitusta, ainakin siltä osin, mikä koskee verkkopalvelua
- käyttäjien tarpeiden kartoitusta (ei kannata tehdä sovellusta, jota käyttäjät eivät tarvitse!)

tavoitetilan määritystä

- mittareiden asettamista tavoitetilalle
- verkkopalvelun aiheuttaman toiminnan muutoksen määritystä (jos mikään ei muutu, niin miksi verkkopalvelun rakentamiseen tuhlataan resursseja?)
- verkkopalvelun tietojen kuvaamista (tietojen kuvaus ei ole sama kuin käyttöliittymän kuvaus)
- verkkopalvelun rakenteen kuvaamista (verkkopalvelun rakenne ei yleensä ole sama kuin pääsivu linkkeineen)
- verkkopalvelun käyttöönoton, siihen liittyvän tiedottamisen ja tulevan ylläpidon suunnittelemista (vastuuhenkilöt pitää nimetä ja heille pitää varata riittävästi resursseja, koska toimimaton tai vanhentunut verkkopalvelu on yrityskuvan kannalta huonompi vaihtoehto kuin verkkopalvelun puuttuminen).

Yllättävän usein verkkopalveluprojektissa halutaan vain tehdä jotain näyttävää verkkoon eikä oteta huomioon:

- mihin sitä tarvitaan?
- onko kohderyhmä (asiakkaat) verkossa?
- kuka palvelua ylläpitää?

Artikkelin alussa esittämistäni epäonnistuneiden verkkopalveluprojektien kommenteista ensimmäinen liittyi tilanteeseen, jossa kohderyhmä ei ollut verkossa, joten verkkopalvelun rakentaminen ei ollut yrityksen kannalta mielekästä. Toinen kommentti liittyi tilanteeseen, jossa kohderyhmä ehkä oli verkossa, mutta yrityksellä ei ollut osaamista eikä resursseja verkkopalvelun ylläpitämiseen.

Kolmas ja neljäs kommentti liittyvät tilanteisiin, joissa asiakkaan tarpeita ei kartoitettu riittävästi vaan projekti aloitettiin käyttöliittymästä – ja jouduttiin tekemään paljon moninkertaista ja turhaa työtä, joka olisi kunnon organisaatioanalyysilla vältytty.

Kaksi viimeistä kommenttia lienevät yleisiä missä tahansa systeemityöprojekteissa. Kunnollinen projektinhallinta ja sovittujen asioiden kirjaaminen muistioihin tms. auttaisi, jos asiakas ei muista, mitä viimeksi sovittiin tai muuttaa jatkuvasti mielipiteitään. Tilanne, jossa asiakas ei tiedä, mitä haluaa, vaatii systeemityöläisen ammattitaitoa: kykyä esittää oikeita kysymyksiä, kärsivällisyyttä, kuuntelemisen herkkyyttä, sovellusalueen tuntemusta jne. Tällaisia taitoja systeemityöläisellä on aina pitänyt olla eikä tällaisen osaamisen tarve ole mihinkään kadonnut – eikä ainakaan korvautunut kyvyllä laatia näyttäviä käyttöliittymiä.

Tietojen mallintaminen

Kaikkein vaikeinta verkkopalvelussa tuntuu olevan tietojen mallintaminen. Yllättävän monesti törmään kysymykseen: ”Verkkopalvelun tietojen kuvaus? Eikö se ole sama kuin web-sivujen kuvaus?” Kun selitän, että tietojen kuvauksen pohjalta laaditaan tietokantamalli, ja tietokannassa ovat sovelluksessa käsiteltävät tiedot, jotkut projektiryhmät vakuuttavat, ettei yrityksellä tule koskaan olemaan enempää kuin viisi tuotetta, joten niiden tiedot voidaan kirjoittaa käsin websivulle eikä mitään tietokantoja tai niiden mallintamista tarvita – ja että juuri tässä sovelluksessa tuotetiedot ovat ihan erilaisia kuin yrityksen operationaalisessa tietokannassa.

Sovelluksen rakenteen ja käyttöliittymän eroa olen yrittänyt monen vuoden ajan konkretisoida kysymällä, voisiko olla mahdollista, että verkkopalvelua, jota projektissa nyt rakennetaan, pitäisi tulevaisuudessa tarjota myös kännykän tai kämmenessä kuljetettavan langattoman päätelaitteen kautta. Tällöin tietojen pitäisi olla samat kuin nyt syntyvässä sovelluksessa, mutta käyttöliittymän erilainen. Jotkut vastaavat tähänkin, että sitten, jos mobiilit päätelaitteet joskus toimivat, koko sovellus mietitään alusta asti uudestaan – ja unohtavat, että jos kunnollista kokonaissuunnittelua voidaan uudelleenkäyttää, säästetään aikaa ja kustannuksia, kun samaa työtä ei tehdä monta kertaa.

Toiminnan muutoksen mallintaminen

Toinen ongelmakohta tuntuu olevan verkkopalvelun aiheuttaman toiminnan muutoksen mallintaminen. ”Verkkopalvelun aiheuttama toiminnan muutos? Eihän tämä pieni, viestinnällinen verkkopalvelu muuta yrityksen toimintoja, prosesseja tai tehtäviä millään tavoin.” Vaikka kysymys olisi vain pienestä viestinnällisestä sovelluksesta, jolla organisaatio aloittelee verkkopalveluaan, niin

- jonkun pitää ylläpitää sovelluksen tietoja, jos niitä ei tuoteta suoraan tietokannasta
- jos sovelluksessa on palautelomake tai sähköpostiosoite, jonkun pitää varautua vastaamaan asiakkaiden näiden kautta lähettämiin palautteisiin ja kyselyihin
- asiakkaan toiminta voi muuttua, jos hän saa tarvitsemansa tiedon verkkopalvelusta sen sijaan että soittaisi organisaatioon.

Lopuksi

Olen kuvannut verkkopalveluprojektien ongelmia, mutta nämä ovat kuitenkin pieni vähemmistö monien onnistuneiden verkkopalveluprojektien joukossa. Viimeistään loppuraporteissaan projektiryhmät toteavat, kuinka olennaisesti organisaatioanalyysi vaikutti onnistuneen lopputuloksen saavuttamiseen. Useimmiten kiitos kunnollisesta analyysista saadaan jo ennen sovelluksen valmistumista, kun toimeksiantaja kokee ahaa-elämyksiä ja pystyy korjaamaan toiminnassaan olevia puutteita ja epäkohtia jo ennen tavoitellun verkkopalvelun valmistumista.

Eija Kalliala,
Helsingin liiketalouden ammattikorkeakoulu Helia,
eija.kalliala@helia.fi,
http://myy.helia.fi/~kalei

Apuvälineitä yksilön työprosessin kehittämiseen ohjelmistotuotannossa

Paula Miinalainen,
Arborvitae Oy

Systeemit ja ohjelmistot ovat useimmiten ryhmän työn tulos. Ryhmän työprosessien kehittämiseen on olemassa useita menetelmiä. Ohjelmistojen laadun parantamisen ratkaisuna on pidetty parempaa testausta, parempaa suunnittelua, parempia ohjelmointivälineitä ja muita tekijöitä. Yksilöllisiä apuvälineitä, jotka tukisivat yksilön oman työnkehittämistä ei juuri ole ollut esillä. Sytykkeessä löysimme kaksi tarkastelun kohteeksi.

PSP - The Personal Sotware Process USA:ssa

Sytykkeen ”Systeemityö sujuvaksi”-työryhmässä kiinnostuimme ensimmäiseksi Watts S Humphrayn PSP (The Personal Sotware Process) –menetelmästä. Se on kehitetty USA:ssa The Software Engineering Instituutissa. Menetelmän Humphray on seikkaperäisesti kuvannut kirjassaan A Discipline for Software Engineering (1995). Kirja soveltuu opetuskäyttöön hyvin. Siinä on harjoitukset valmiina ja asiat on esitetty hyvin yksityiskohtaisesti.

PSP:n tarkoituksena on antaa ohjelmoijalle välineet kehittää omaa työprosessiaan. Lähtökohtana on oman työprosessin tunteminen ja tämän tunnistamisen pohjalta oman työn analysointi ja kehittäminen. Menetelmän perustana on joukko lomakkeita, joita tekijä täyttää työn edetessä. Lomakkeita on tarjolla todella paljon. Työprosessin kaikki vaiheet on tarkoitus dokumentoida henkilön omaan käyttöön jatkoanalysointia varten. Tämä vaiheiden ”minuuttikirjaaminen” tuo esiin kaikki piilevätkin työvaiheet ja tekee työn todellisuuden näkyväksi.

Työryhmässä totesimme menetelmän olevan seikkaperäisyydessään hyvin raskas ja vaikeasti soveltuvan suomalaiseen yrityskulttuuriin. Sitä on käytetty erittäin laajojen järjestelmien tekemisessä organisaatioissa, joissa on totuttu tiukkaan kurinalaiseen työhön. Mitähän suomalaiset ”propellihatut” tästä tuumisivat?

Toisaalta Watts S Humphrayn on siirtänyt kirjaan pitkän kokemuksensa ohjelmistotuotannossa. Siinä on paljon hyvää, josta voi ammentaa parhaat palat sovellettavaksi omaan työhön.

Näihin samoihin johtopäätöksiin oli tultu muuallakin. PSP ei ole käynnistynyt pienemmissä organisaatioissa sen byrokraattisuuden ja tiukan kurinalaisuuden vuoksi. On tarvetta löytää menetelmiä, jotka sopivat pieniin ja keskisuuriin yrityksiin.

t tt t ttt tt ttttttttttttt t ttttt t Programming Skills in Industry Euroopassa

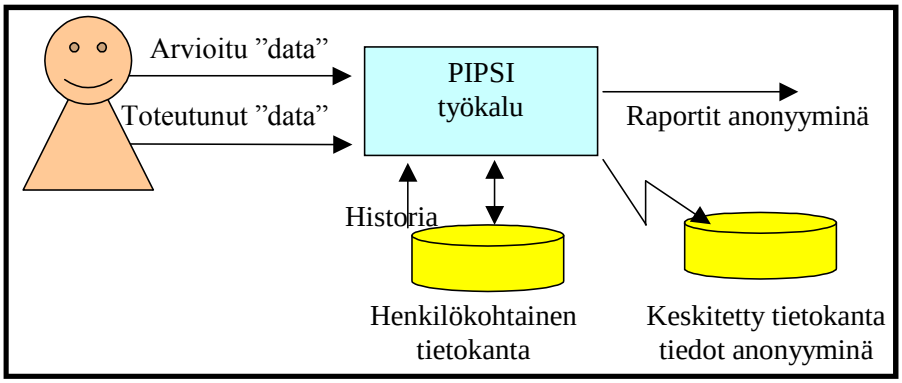
PIPSI on EU-projektissa IPSSI kehitetty yksilön käyttöön tarkoitettu oman työprosessin kehittämismenetelmä. Projektin päämääränä oli löytää keinoja ja apuvälineitä, joilla lisätä systeemityö- ja ohjelmointitaitoja Euroopan keskisuurissa ja pienissä yrityksissä.

PIPSI:n tavoitteena on parantaa ohjelmistokehittäjän henkilökohtaista projektin johtamista ja henkilökohtaista laatujohtamista. Tässä ymmärretään kukin tehtäväkokonaisuus kehittäjän henkilökohtaisena projektina, jonka johtamisesta ja laadusta hän on vastuussa. Tarkoitus on parantaa ohjelmiston kehitystä ”bottom up”.

Henkilökohtainen projektin johtaminen tarkoittaa tehtävän suunnittelua, aika taulun laatimista ja työmäärän arviointia. Mitä paremmin ohjelmoija tuntee oman työprosessinsa sitä paremmin hän oppii myös arvioimaan työtään ennalta ja antamaan realistisia aikataulu- ja työmääräarvioita. Työprosessin yksityiskohtainen rekisteröinti tuo esiin myös mm. sisältömuutosten vaikutuksen työhön ja antaa perusteet järkevälle muutosten hallinnalle. Toteutuskelpoiset aikataulut taas vähentävät tarvetta ylipitkiin työpäiviin ja vähentävät stressiä.

PIPSI-menetelmää tukee PIPSI- ohjelmisto, joka muodostuu kahdesta osasta. Kts. kuva alla.

Ensimmäinen osa on tiedon keruu, jossa yksinkertaisella ja helpolla tavalla rekisteröidään aika, ohjelman koko ja havaitut virheet. Toisessa osassa analysoidaan kerättyä tietoa. Tulokset saadaan



teksti- ja myös grafiikkamuodossa. On tärkeätä huomata, että koko ajan kerätty tieto on vain ohjelmoijan itsensä tiedossa. Hänellä on mahdollisuus lähettää keräämänsä tieto nimettömänä keskitettyyn tietokantaan.

Tiedonkeruu on siis perustana oman työprosessin tuntemiselle ja sitä seuraavalla kehittämiselle. Tekijä huomaa toistavansa tietyn tyyppisiä virheitä. Havainnon jälkeen hän todennäköisesti oivaltaa tavan, jolla välttää toistuvat virheet.

Tiedetään yleisesti, että keskeytykset ovat merkittävä haittatekijä ohjelmointityön tuottavuuden kannalta. Kerätyn tiedon pohjalta voidaan laskea keskeytysten merkitys. Analysoidaan keskeytysten syyt yksilötasolla ja poistetaan keskeytykset. Säästetään aikaa ja rahaa.

Henkilökohtaisen laadun hallinnan kannalta tavoitteena on löytää virheet ja puutteet entistä aiemmin ja näin säästää energiaa, aikaa ja rahaa. Laatutyön avuksi on erilaisia tsekkauslistoja. Näitä listoja voi ”personoida” omaan työtapaan soveltuviksi kerätyn tiedon pohjalta.

Mittareita voi käyttää myös saavutusten mittaamiseen ja asettaa itselleen mitattavia tavoitteita ja välitavoitteita aivan kuin urheilussa. Itse asetetun tavoitteen saavuttaminen lisää omaa työtyytyväisyyttä. Se on henkinen palkinto.

PIPSI on modulaarinen ja sen käyttöönotto voidaan porrastaa. PIPSI-tietokanta on Microsoft Access 2000 kanta. Ohjelmisto on saatavissa yhdellätoista kielellä, myös suomen kielellä. Menetelmä tulee soveltaa kuhunkin yrityskulttuuriin sopivaksi. Ja myös toisinpäin. Kerätty historiatieto toiminasta voi antaa sysäyksen merkittävillekin kulttuuri- ja työtapamuutoksille organisaatiossa.

Kirjallisuutta

Project Control: The Human Factor Proceedings of ESCOM –SCOPE 2000

Watts s. Humphrey, A Discipline for Software Engineering

Watts s. Humphrey, Introduction to the Team Software Process

http://www.compapp.dcu.ie/ipssi/

Paula Miinalainen,
Arborvitae Oy

SYTYKE ry:n WEB-toimikunnan jäsenet 2001

SYTYKE ry:n WEB-toimikunnan jäsenet vuonna 2001 ovat:

- Heini Holopainen, käytettävyysasiantuntija
- Lauri Laitinen, web-master
- Minna Oksanen, rakennevastaava
- Erkki Pöyhönen, tekninen asiantuntija
- Helena Venäläinen, puheenjohtaja

Verkkosivujen uusimisessa mukana olleet Helsingin yliopiston mediakasvatuksen opiskelijat olivat:

- Auli Pasanen, aikuiskasvatus, verkkopedagogiikka
- Katja Pellikka, viestintä
- Hannu Takala, kasvatustiede
- Helena Venäläinen, tietojenkäsittelytiede, mediakasvatus

Dynaaminen asiantuntijuus, ajattelun työkalut ja käsitteiden merkitys systeemityössä

Helena Venäläinen,
Finanssidata Oy

Taustaksi

Sytyke ry:n verkkosivujen uudistuksen yhtenä lähtökohtana oli miettiä, millaista tukea yhdistyksen jäsenet voisivat yhdistyksen kerhojen sivujen kautta saada työssään kehittymiseen.

Systeemityötä tekevät systeemityön ammattilaiset - siis ihmiset. Kun tarkastellaan systeemityön sujumista prosessina ja mietitään keinoja sen kehittämiseen, ei ole syytä unohtaa sitä, miten ihminen toimii ja miten hänen toimintatapoja voitaisiin tukea ja kehittää. Me tiedämme, että ihmisen muistikapasiteetti on rajallinen - toisaalta asiantuntijat pystyvät työstämään saamaansa tietoa erittäin mielekkäästi ja oleelliseen keskittyen.

Yritän selvittää seuraavassa ensin lyhyesti, millaisia ajatuksia on nykyisten asiantuntijuustutkimusten perusteella syntynyt ihmisen kehittymisestä asiantuntijana ja miten asiantuntijuudelle ominaisten metakognitiivisten taitojen kehittymistä voidaan tukea - tietotekniikallakin. Seuraavaksi kuvaan, miten yksilön kuormitusta voidaan vähentää ajattelun apuvälineillä sekä yhteistyöllä. Valotan myös käsitteiden hallitsemisen merkitystä systeemityön sujumisessa. Olen tietenkin samalla pohtinut WEB-uudistuksen innoittamana Sytyke ry:n kerhotoiminnan ja verkkoympäristöjen tarjoamia mahdollisuuksia asiantuntijuuden eri osa-alueiden kehittämisessä.

Asiantuntijamainen toimintatapa

Älykkyys ei ole ominaisuus

Tietotekniikan(kin) parissa työskentelevien ihmisten maailma on muuttunut viimeisten vuosikymmenten aikana entistä pirstaleisemmaksi ja käsitteellisemmäksi eikä muutosvauhti tunnu laantuvan. Emme pysty enää hallitsemaan kaikkea yksin ja älykkäästi toimiaksemme joudumme reagoimaan ulkomaailman vaatimuksiin ja kehittämään taitojamme.

Perinteisen käsityksen mukaan ihmisen älykkyys on synnynnäinen ja kiinteä ominaisuus, johon emme voisi juurikaan vaikuttaa. Kognitiivinen tutkimus asettaa tämän käsityksen voimakkaasti kyseenalaiseksi. Stenbergin mukaan *ihmisen älykkyys on dynaaminen ominaisuus, joka vaihtelee yksilön eri elämänvaiheissa ja jota voidaan kehittää luomalla suotuisat olosuhteet asiantuntemuksen ja korkeamman asteen tiedonkäsittelyntaitojen kehittymiselle.* Älykkyys on hänen mukaansa pikemminkin suhde yksilön henkisten valmiuksien ja yhteiskunnan asettamien vaatimusten välillä kuin yksilön ominaisuus (Hakkarainen).

Dynaaminen asiantuntijuus

Asiantuntijuuden kehityksen avaimena pidetään 90-luvun asiantuntijuustutkimusten perusteella uusien ongelmien ratkaisutaitojen kehittymistä. Pelkkä kokemus ei tee ihmisestä alan huippuosaajaa. Jos osaaminen perustuu kerran hankitun tietämyksen ja taitojen soveltamiseen, joutuvat *rutiiniasiantuntijat* helposti vaikeuksiin kohdatessaan uusia ongelmia tai tilanteita. *Dynaaminen asiantuntija* näkee ongelmassa yleensä enemmän kuin muut, nauttii ongelmasta ja pystyy tuottamaan suuren määrän tietoa sen ratkaisemiseksi. Dynaaminen asiantuntijuus mahdollistaa tarkoituksenmukaisen toiminnan uusissa tilanteissa ja on tärkeä edellytys tietoyhteiskunnassa selviytymiselle.

Viimeisimpien käsitysten mukaan asiantuntijaksi kasvamisessa keskeisintä on ns. progressiivinen eli asteittain syvenevä ongelmaratkaisu. Tällöin osa kokemuksen myötä vapautuvista älyllisistä voimavaroista sijoitetaan uudelleen ongelmanratkaisemiseen vaativammalla tasolla. Dynaamisen asiantuntijuuden ja progressiivisen ongelmanratkaisun tunnusmerkkejä ovat jatkuva oppimiseen investoiminen, vaikeampien ja haasteellisempien ongelmien etsiminen ja pyrkimys oman alansa ongelmien ja tietojen hahmottamiseen syvemmällä ja syvemmällä tasolla. Uskallus ihmetellä, ottaa älyllisiä riskejä ja vaihtaa ajatuksia muiden kanssa ovat keskeisiä tekijöitä kehittymisessä asiantuntijaksi tulevaisuuden tietoyhteiskunnassa (Hakkarainen).

Metakognitiivisten taitojen merkitys

Metakognitio tarkoittaa ajattelu- ja toimintaprosessin suunnittelun, ohjaamisen ja arvioinnin säätelyprosessia. Hyvin kehittyneet metakognitiiviset taidot ovat asiantuntijoiden ylivoimaisen suorituksen taustalla. Tutkimus on osoittanut sen olevan tärkeä tekijä esimerkiksi kirjoittamisprosessissa, uusien käsitteiden oppimisessa ja käsitteellisessä muutoksessa, asiantuntijoiden oppimaan oppimisessa sekä ongelmanratkaisuissa.

Omien tietojen ja taitojen rajojen tunnistamisen on havaittu myös olevan asiantuntijoille ominaista. Nykyisten määritelmien mukaan viisas tuntee omat tietonsa, tietää mitä ei tiedä sekä oivaltaa, mitä periaatteessa voisi tietää (Hakkarainen).

Systeemityö on hyvin pitkälle ongelmanratkaisuprosessi, jolloin sen sujumisen kannalta on ensiarvoisen tärkeää, että siihen osallistuva yksilö pystyy asettamaan tavoitteita ja ohjaamaan toimintaansa niiden suuntaisesti. Oman toiminnan tuloksellisuuden arviointi vaatii siis metakognitiivisia tietoja ja itsearvioinnin taitoja. Järjestelmäkehityksen

avuksi on kehitetty projektityö- ja systeemityömalleja tukemaan ajattelu- ja suunnittelutoimintaa. Niiden metakognitiivisen merkityksen oivaltaminen ja joustava, tilannekohtainen soveltaminen on kuitenkin onnistuneen hyödyntämisen taustalla. Suunnitteluhan ei ole itsetarkoitus, vaan keino saada lopputulos aikaan. Tehokas ongelmaratkaisuprosessi sisältää aina ratkaisuaikaista tuloksellisuuden arviointia ja joustavaa uudelleensuunnittelua sekä prosessimallien jatkuvaa kehittämistä niiden toimivuuden arvioinnin pohjalta.

Teknologian tukemat työskentely-ympäristöt metakognisten taitojen kehittäjinä

Metakognitiivisten tietojen ja taitojen kehittymistä voidaan edistää luomalla sellaisia yhteisöllisen toiminnan muotoja, joiden avulla on mahdollista päästä käsiksi omiin ja toisten metakognitiivisiin prosesseihin. Aktiivinen osallistuminen toiminnan suunnitteluun, ohjaamiseen sekä arviointiin, metakognitiivisten käsitteiden käyttäminen avoimesti ja pohdiskelevasti sekä toiminnan hajautettu säätely ovat edellytyksiä metakognition kehittymiselle.

Kognitiivinen suunnittelututkimus on tuottanut runsaasti tutkimustietoa suunnitteluprosessien luonteesta Luovan ongelmaratkaisuprosessin ja tieteelliseen tietoon ja keksintöihin johtavien suunnittelustrategioiden analysoinnin jälkeen keskityttiin asiantuntijoiden suunnitteluprosessin analysointiin ja suunnittelualojen erityistiedon merkityksen korostamiseen. Viimeksi 90-luvulla on keskeisimmäksi tutkimusalueeksi noussut todellisuuden ulkoisten edustusten (sanalliset ja kuvalliset luonnokset) merkitys suunnitteluprosessin tukena, jaettuun asiantuntemukseen perustuvien (yhteisöllisten) suunnitteluprosessien analysointi ja niitä tukevien verkkoympäristöjen kehittäminen (Hakkarainen).

Suunnitteluprosessi etenee yleensä iteraatiokierrosten mukaan tarkentuvien ideoiden ja toteutusmahdollisuuksien välillä. Tiedonkehittelyä voidaan myös tehdä tieto- ja viestintätekniikkaan perustuvana prosessina, jolloin prosessiin osallistuvat jäsenet yhdessä pyrkivät ratkaisemaan asettamiaan ongelmia yhteisöllisesti verkossa. Toiminta perustuu kuvan tai tekstin talletuksen mahdollistavan tietokannan ympärille rakennetun ohjelmiston hyödyntämiseen sovittuja pelisääntöjä noudattaen. Prosessin edetessä on tehtävää voitava muokata uudelleen tarpeen mukaan.

Ongelmanratkaisuprosesiin osallistujia ohjataan luokittelemaan omia tietokonemuistiinpanoja nk. ajattelutyyppien (esim. ongelma, tarkentuva ongelma, työskentelyteoria, syventävää tietoa, kommentti, metakommentti, yhteenveto) mukaan. Kyseisessä ympäristössä esitetty metakäsittellinen tieto voi sisäistyä myös prosessiin osallistuneiden "tutkijaopiskelijoiden" omaksi tiedoksi (Hakkarainen).

Sytyke ry:n verkkosivut voisivat nähdäkseni joskus tulevaisuudessa sisältää tarkoitetun kyseisen tyyppisen ryhmätyöohjelmiston (esim. Future Learning Envirement, FLE). Kerhot voisivat halutessaan hyödyntää "oppimisympäristöä" jonkin tutkimusongelmansa käsittelyssä, jolloin ongelman ratkaisun myötä siihen liittyvien käsitteiden omaksumisen ohessa tuettaisiin myös osallistujien metakognitiivisien taitojen kehittymistä. Uskoisin näiden taitojen vahvistamiselle olevan sosiaalisen tilauksen "asiantuntijajäsentemme" keskuudessa, sillä koulumaailmamme tapa opettaa ei juurikaan nykyisellään tue niiden kehittymistä.

Asiantuntija osaa hyödyntää ajattelun työkaluja

Tiedonkäsittelyn ulkoistus ja hajautus

Ihminen pystyy käsittelemään kerrallaan vain rajallisen määrän tietoa ja häneltä voidaan odottaa ainoastaan sellaista, mihin hän parhaimmillaan pystyy. Ihminen on kehittänyt näiden rajoitusten kiertämiseksi huomattavan määrän apuvälineitä ja luonut päänsä ulkopuolelle kulttuuritiedon maailman, jonka kehittämiseksi me tietotekniikan ammattilaiset nykyisin työskentelemme.

Kognitiivisen tutkimuksen tulosten mukaan kaiken ihmisen toiminnassaan käyttämän tiedon ei tarvitse olla ihmisen päässä. Hajautetun kognition tutkijat esittävät, että ihmisen muisti- ja päättelyresurssit ovat ilman ulkoista tukea hyvin rajoitetut. (Jokainen voi todentaa tämän suorittamalla vaikkapa kertolaskun 49 * 53 , todennäköisesti teette sen paperilla tai laskukoneella.) Fysikaalisesti hajautettu kognitio tarkoittaa sellaisia älyllisen toiminnan prosesseja, jotka pohjautuvat keinotekoisen ajattelun työvälineiden käyttöön. Sosiaalisesti hajautettu kognitio taas viittaa tilanteeseen, jossa rajoitetuilla ajattelemisen ja päättelemisen resursseilla varustetut ihmiset yhdessä ratkovat monimutkaisempia ongelmia, kuin yksittäiselle ihmiselle olisi mahdollista. Prosessointi siis hajautetaan useisiin päihin ja arkkitehtuurimalliksi valitaan "blackboard", jossa ratkaisu perustuu erilaisia osaamisia hallitsevien prosessien yhteistyössä synnyttämään lopputulokseen. Seinätaulutekniikan suosio ja toimivuus vahvistavat empiirisesti yhteisöllisen tiedonkehittelyparadigman toimivuutta. Asiantuntijamainen tapa rakentaa tietojärjestelmiä on ulkoistaa ja hajauttaa tiedonkäsittely ja tukea näin omien rajallisten resurssien riittävyyttä ja saada tätä kautta sujuvuutta systeemityöhön.

Sytyke ry:n kerhojen ajattelutoiminnan näkyväksi tekeminen jonkin ryhmätyöohjelmiston avulla mahdollistaisi siis hajautetun kognition hyödyntämisen kerhon kulloisenkin tutkimusongelman ratkaisussa ja opettaisi samalla tutkimukseen osallistuville yhteisöllistä tiedonkehittelyä.

Tietotekniikan tukema yhteistyö vaatii pelisääntöjä

Yhteistyötä edistävän tietotekniikan pitäisi mahdollistaa talletetun tiedon ja viestien säilyminen tietokannoissa niin, että 1) yksittäisten tietojen ja viestien väliset suhteet säilyvät, 2) tieto on helposti saatavilla, 3) se on tarkasteltavissa eri näkökulmista sekä 4) tietojen yhteinen tietojen ylläpito on helppoa. Tietotekniikka yksin ei kuitenkaan riitä, sen hyödyntämisen edellyttämät toimintatavat pitää rakentaa aktiivisesti.

Järjestelmän käyttö on käyttäjien aktiivisen muokkaamisprosessin tulosta ja tässä tärkeää osaa näyttelevät käsitysten yhteensopivuuteen hakeutuminen ja sen

seurauksena toimintojen yhteensopivuus. *Yhteistyön tukeminen siis vaatii välineiden muokkaamista työhön sopiviksi ja toisaalta työn muokkaamista sellaiseksi, että siinä voidaan hyödyntää yhteistyötä tukevan tietotekniikan mahdollisuuksia.* Tällainen yhteensovitusprosessin on myös haavoittuva – jos käyttäjät eivät pysty sopimaan yhteisistä käyttötavoista voi mukautumaton osa käyttäjistä joutua ulkopuoliseen asemaan (Karsten).

Riitta Kuusinen on kevällä 2001 julistetussa väitöskirjatutkimuksessaan todennut, että tietoliikenneverkon mahdollistamassa tiedontuottamisyhteistyössä tarvittaisiin ryhmäkeskeistä tiedonkäsittelytaitoa. Koulutuksessa kaikille automatisoitunut itsekeskeinen sosiaalisen tiedonkäsittelyn toimintamalli laukeaa kuitenkin käyttöön. Hänen tulkinnan mukaan tästä syystä tiedontuottamisyhteistyö ei hänen väitöskirjatutkimuksessaan käynnistynyt. Jos halutaan yksilöiden prosessoivan tietämystä yhdessä, heille on luotava siihen kannustavia ja sitä arvostavia sosiaalisia toimintaympäristöjä - pelkkä web-foorum ei riitä. Tietointensiiviset yhteistyöprosessit edustavat tällaisia sosiaalisia toimintaympäristöjä ja siksi niitä käytetään tulevaisuuden työelämässä yhä enemmän erilaisiin tarkoituksiin (Kuusinen).

Jos/kun Sytyke- kerhoissa päätetään hyödyntää jotakin ryhmätyöohjelmistoa ongelmien kehittelyssä, on edessä väistämättä uusien toimintatapojen ja tekniikan yhteensovittaminen sekä sopivien käyttötapojen kokeileminen, niiden sopiviksi kehittäminen ja omaksuminen - ja tämä vie varmasti aikaa.

Symbolien ja käsitteiden merkitys systeemityössä

Symbolit havainnollistavat asiaa ja mahdollistavat tulkinnan

Kirjoitustaidon syntymisen myötä on puhekieli voinut tulla tietoisen tarkastelun kohteeksi ja siihen sisältyvät epäloogisuudet samalla näkyviksi ja paranneltaviksi. Kirjoitustaito on mahdollistanut kulttuuritiedon syntymiseen ja sen välittämisen sukupolvilta toisille. *Kirjoitetut ja varsinkin piirretyt symbolit mahdollistavat meille niiden sisältämien tulkintojen oppimisen jälkeen monimutkaistenkin käsitteellisten asioiden havainnollistamisen* (Hakkarainen).

Tietojärjestelmän suunnittelu ei yleensä tapahdu enää käsityömäisesti yhden ihmisen päässä tai vastuulla vaan tietokoneella erilaisten suunnitteluohjelmien avulla.

Sujuvan systeemityön kannalta on siis erittäin suuri merkitys sillä, millaisia symboleja käytämme tiedonkehittelyssä, miten systemaattisesti teemme järjestelmistä rakennuspiirustuksia ja miten niitä kehittelemme. Vaikka prosessointi tapahtuu pään sisällä, käsiteltävä tieto voi sijaita helpommin hahmoteltavissa vaikkapa UML-kuvana näyttöpäätteellä. Monimutkaisten, abstraktien ilmiöiden käsittely ilman niiden mallintamista havaittaviksi konkreettisiksi kohteiksi ei ole mahdollista.

Systeemityön käsiteviidakko

Tietojärjestelmien rakentaminen edellyttää alan asiantuntijalta melkoisen käsite- ja apuvälineviidakon hallintaa

1) metakognitiiviset käsitteet, taidot ja apuvälineet (esim. projektityö- ja systeemityön vaihejakomallit)
2) erilaiset kuvaustekniikat (esim. UML) notaatioineen ja piirturiohjelmineen.
3) järjestelmäkehitystyön tutkimus/soveltamismenetelmät (esim. olioparadigma, ohjelmistoarkkitehtuurit, ratkaisumallit, kehykset, relaatiomalli)
4) sovelluskehittimet ympäristöineen ja termeineen (esim. JBuilder, MS-Visual Studio, Delphi)
5) ohjelmointikielet (esim. Java, VB, C++) kirjastoineen, tulkkeineen ja kääntäjineen

sekä viimekädessä taitoa käyttää tätä työkalupakkia järkevästi teknisten apuvälineiden avulla liiketoiminnallisten ongelmien ratkaisemisessa.

Tietosysteemin rakentamiseen sisältyy pääasiassa abstrakteja käsitteitä, joiden tehokas hyödyntäminen - sujuva systeemityö - edellyttää niiden syvällistä ymmärtämistä, jota taas ei synny ilman käsitteiden kanssa tapahtuvaa konkreettista työtä. Käyttämiemme symbolien (esim. UML) tai termien jatkuva vaihtaminen (aliohjelma onkin nyt kirjastofunktio) tai vanhan termin käyttäminen uudessa tai monessa eri merkityksessä (esim. komponentti) ei myöskään ole mielekästä. Joutuessamme opettelemaan uuden “kirjoitustavan” tai antamaan käsitteelle uuden sisällön, emme pysty keskittymään varsinaisesti ratkaistavaan asiaan. (Voisimmeko ottaa tässä asiassa mallia matemaatikoilta tai sähköinsinööreitä, heillä on hyvin vakiintuneet symbolit ja kuvauskielet ja niiden oppimisen jälkeen he voivat syventyä ongelmien ratkaisuun.)

Uusien apuvälineiden käyttöönotossa voisimme myös yrittää lieventää käyttäjien “ymmärtämiskynnystä”. Esimerkiksi sovelluskehittimien tehokas hyödyntäminen edellyttäisi niiden taustalla olevien arkkitehtuuri- ja ratkaisumallien sekä näihin liittyvien rajoitusten tuntemista. Koska tämä ei useinkaan ole järjestelmien myyjien intressinä (eikä välttämättä kiinnosta ostopäätöksen tekijääkään), joutuvat järjestelmien käyttäjät opettelemaan kantapään kautta, mikä toimii ja mikä ei. Kun järjestelmäkehityksessä käyttöönotetaan uudenlainen teoreettinen lähestymistapa, ei käsitemuodostuksen tuen panostusta voi vähätellä. Esimerkiksi perinteisen tieto/toimintopohjaisen sovelluskehityksen taitajien vaikeudet ymmärtää ja soveltaa täysin uudenlaista ongelmaratkaisutapaa, olioparadigmaa, on käytännössä huomattu. Uuden työskentelytavan omakohtaisten hyötyjen ymmärtäminen ja sisäinen motivaatio oppia uutta on käsitteidenkin oppimisen ensimmäinen edellytys. Senjälkeen on luotava suotuisat olosuhteet (aika, lähteet) tiedonhankinnalle, sen kehittelylle (yhteisön tuki) ja sitä kautta oppimiselle.

Käsitteiden oppiminen on ymmärtämistä

Konstrutivistisen tietokäsityksen mukaan tieto on pohjimmiltaan yksilön kokemusmaailman uudelleenorganisoitumista eikä se ole sellaisenaan välitettävissä yksilöltä toiselle, sen olemukseen vaikuttavat aina kokemusmaailma, käsitteistö ja näkökulma, joka kulloinkin tietoa synnyttää tai tarkastelee. Käsitteet voidaan ymmärtää sekä

yksilön henkisenä rakenteena että yhteisesti hyväksyttyinä ilmausten merkityksinä. Tieto ei ala käsitteistä vaan käsitteet ovat seurausta siitä, että yksilö tietää jotakin. Käsitestruktuuri tarkoittaa käsitteiden ja yhteyksien muodostamaa tulkinnallista (semanttista) verkkoa. Tällaisen tiedon kartuttaminen tapahtuu pääpiirteissään siten, että aikaisemman tiedon avulla yhdistetään uusia tietoelementtejä. (Kämäräinen).

Käsitteellisellä järjestelmällä tarkoitetaan käsitteiden ja niihin liittyvien ratkaisumallien ja päättelytaitojen nivoutumista toimivaksi kokonaisuudeksi. Uuden käsityksen omaksuminen on vaivalloinen prosessi. Uudet käsitteet tulevat käyttökelpoisiksi vasta kun käsitteiden toiminnalliset eli funktionaaliset roolit ovat muodostuneet. Tämä vaatii käsitteiden pitkäaikaista käyttöä ongelmaratkaisussa.

Tieteellisten käsitteiden merkityksen voi omaksua vain osallistumalla niiden ongelmien ratkaisuun, joita varten käsitteet on alunperin luotu.

Asiantuntijaksi kehittyminen edellyttää ajoittaista pohdiskelevaa ajattelua, jolloin kaikki on hidasta, vaivalloista ja työlästä. Selitysten ja näkemysten loksahtaminen lopulta paikalleen kuitenkin palkitsee huippuelämyksillä. *Tärkeä rooli käsitemerkitysten kehittymisessä on sosiaalisella vuorovaikutuksella, jossa neuvotellaan eri ihmisten samalle käsitteelle antamista merkityksistä ja luodaan näin perusta uusien käsitetulkintojen syntymiselle* (Hakkarainen).

Kun oppilas osallistuu määritelmien muotoilemiseen, hänen abstraktio-, yleistämis-, systematisointi- ja käsitteenmuodostuskykynsä sekä kyky ymmärtää laajoja asiayhteyksiä ja riippuvuuksia kehittyy (Kämäräinen).

Sytyke ry:n kerhotoiminta toimii hyvänä esimerkkinä tämäntapaisesta sosiaalisesta vuorovaikutuksesta. Se tarjoaa puitteet selvittää keskustellen uusia paradigmoja ja käsitteitä sekä pohtia uusien menetelmien ja apuvälineiden hyötyjä ja haittoja. Näin Sytyke ry on tukenut jäsentensä ammatillista kehittymistä jo nykyisten toimintamuotojensa kautta. Käsitemuodostusta tukevaa toimintaa voisi kehittää jatkossa myös verkkoympäristöön, kerhojen "pienessä" piirissä tekemät pohdinnat ja niiden johtopäätökset voisivat olla muidenkin nähtävillä. Myös ongelmien esittäminen ja niihin vastaaminen verkossa (FAQ-tyyppisesti) mahdollistaisi jäsenten käsitteellisen ymmärryksen syventämisen.

Yhteenveto

Dynaaminen asiantuntijuus mahdollistaa tarkoituksenmukaisen toiminnan uusissa tilanteissa ja on tärkeä edellytys tietoyhteiskunnassa selviytymiselle. Se on alati muuttuvassa yhteisössämme oikea lähtökohta osaamisen kehittämiselle - sekä yksilöllisesti, että yhteisöllisesti.

Systeemityön sujumista voidaan merkittävästi edistää kehittämällä asiantuntijamaista toimintatapaa ja siihen liittyviä toimintamalleja, kuten oman ajattelu- ja toimintaprosessin suunnittelun, ohjaamisen ja arvioinnin säätelyprosessia. Ajattelun apuvälineiden (teksti, kuvat) sujuva hyödyntäminen sekä yhteisöllinen tiedonkehittely helpottavat yksilön kuormitusta ja mahdollistavat osaltaan älyllisen toiminnan rajojen ylittämisen.

Kognitiivista rasitusta voidaan vähentää välttämällä turhia kuvaustapojen, terminologioiden ja työvälineiden vaihtoja. Yleisesti käytössä olevia standardien ja termien käyttäminen itsekeksittyjen sijasta edistää käsitteiden selkiytymistä.

Suurempien toimintatapa tai paradigmavaihdosten yhteydessä tuetaan käsitteiden ja kokonaisuuksien ymmärtämistä ja niiden nivoutumista entiseen osaamiseen tehokkaimmin mahdollistamalla niistä keskustelut sekä kytkemällä opiskelu työtehtäviin.

Sytyke-kerhot tarjoavat oivallisen perustan tietoteknisen asiantuntijuuden kehittämiselle:

- Mukana olevilla ihmisillä on aktiivinen ote asiaan sekä halu kehittyä jatkuvasti – joka on kaiken oppimisen ja dynaamisen asiantuntijuuden edellytys.
- Yhdistyksen jäsentilaisuudet ja kerhotoiminta antavat mahdollisuuden opiskella jäseniä kiinnostavia uusia käsitteitä.
- Kerhojen ongelmalähtöinen tiedonkehittelytapa helpottaa uusien käsitteiden liittämistä aikaisempiin tietorakenteisiin.
- Kerhojen ongelmaratkaisutilanteet antavat mahdollisuuden kehittyä myös ongelmanratkaisijoina – ja kehittää näin metataitoja.

Kaksi viimeksi mainittua voivat erityisen hyvin kehittyä yhteisöllisesti, keskustellen, mielipiteitä vaihtaen ja systeemityön kulttuuritietoa näkyväksikin työstäen - tulevaisuudessa SYTYKE-lehden lisäksi toivottavasti myös kehittyvässä verkkoympäristössämme.

Lähteet:

Hakkarainen K., Lonka K., Lipponen L. 1999: Tutkiva oppiminen – älykkään toiminnan rajat ja niiden ylittäminen. WSOY, Porvoo.

Karsten H. 2000. Weaving Tapestry: Colloborative Information Technology and Organisational Change (Kuvakudosta kutomassa: Yhteistyötä tukeva tietotekniikka ja organisatoorinen muutos). Julkaisu sarjassa Jyväskylä Studies in computing numero 3. Jyväskylä.

Kuusinen R. 2001: Väitöskirja: Ongelmana yhteistyökyvyttömyys? Teoreettisen ymmärryksen etsintää web-avusteiselle tiedontuottamisyhteistyölle. Helsingin yliopiston Kasvatuspsykologian tutkimusyksikön julkaisu 2/2001. http://www.conet.org/pdf.html

Kämäräinen J., Haapasalo L. 1998: Hyperteksti. Laatiminen ja käyttö oppimisen, tiedonhankinnan ja kirjallisuuden näkökulmista. Medusa-Software, Joensuu.

Helena Venäläinen, Yksikönpäällikkö,
Finanssidata Oy,
Tietojenkäsittelytieteen ja mediakasvatuksen opiskelija,HY/MEC,
helena.venalainen@osuuspankki.fi

SYTYKE ry:n uusilla verkkosivuilla korostuu asiantuntijuus

Katja Pellikka,
Viestinnän ja mediakasvatuksen opiskelija HY/MEC

Sytyke ry:n verkkosivujen uusiminen toteutettiin yhteistyössä Helsingin Yliopiston mediakasvatuksen opiskelijoiden kanssa. Sytyke ry:n web-toimikunta toimi projektin johtoryhmänä ja teknisenä tukena. Opiskelijat muodostivat sivuston suunnitteluryhmän.

Jäsenten tarpeiden ja muiden sivustoon liittyvien seikkojen monivaiheinen pohtiminen käynnistyi helmikuun alussa ja projektiryhmä luovutti huhtikuun lopussa tutkimusraporttinsa Helsingin yliopistolle sekä Sytyke ry:lle. Web-toimikunta huolehtii sivujen teknisestä toteutuksesta, joka on parhaillaan käynnissä. Sivuston uusi uljas ilme paljastuu toukokuun 2001 kuluessa.

Lähtökohtana jäsenten tarpeet

Verkkosivujen uudistuksella luodaan Sytykkeelle uudenlaista ilmettä ja parannetaan jäsenten palvelua lisäämällä heidän vaikutusmahdollisuuksiaan. Sivujen sisältöä on suunniteltu paremmin tarkoitustaan vastaavaksi ja näitä tavoitteita

Kuva 1: Helena Väänäsen lisäksi Auli Pasanen, Hannu Takala ja Katja Pellikka ttt tt t ttt t t t t t t t tt t t t t t tt t t.

tukevaksi. Yksi parannetuista seikoista on tiedotus sivuston kautta. Sivujen ulkonäön hiominen oli yksi keskeisimmistä muutostarpeista. Sivuston uudistamisessa on otettu huomioon myös jäsenten työkyvyn ja hyvinvoinnin edistäminen ”pehmeämpien” arvojen kautta.

Uusien sivujen toimivuuden ja hyödyllisyyden takaamiseksi on tärkeää, että ne ponnistavat ennen kaikkea Sytykkeen jäsenten tarpeista. Siksi uusitun sivuston kehittämiseksi kerätään jäseniltä jatkuvasti palautetta sivujen toimivuudesta. Toivomme kehittämisehdotuksia ja ideoita, jotta voimme tulevaisuudessakin noudattaa alkuperäistä

Kuva 2: Ideoinnissa ja ideoitten ryvästelyssä seinäpinta on paras mahdollinen

tavoitettamme: tehdä uusi sivusto, joka on lähtöisin jäsenten tarpeista.

Asiantuntijuuden kehittäminen

Sytyke ry:n ytimenä on asiantuntijuuden kehittäminen, jossa kerhotoiminta on avainasemassa. Kerhotoiminnan kehittämisen yhtenä muotona toimii verkkopohjaisen, ongelmalähtöisen tutkimuksen sekä asiantuntijamaisen työskentelytavan edistäminen. Kerhojen yhteisöllisen tiedonkehittelyn edistäminen on uuden sivuston kannalta tärkeää.

Pohjimmiltaan sivut ponnistavat Sytykkeen toiminta-ajatuksesta: Sytyke edistää systeemityöalan yksityishenkilöiden ja yhteisöjen välistä yhteistyötä, sekä toteuttaa alaa kehittävää tutkimustoimintaa.

Katja Pellikka,
Viestinnän ja mediakasvatuksen opiskelija HY/MEC

Kuva 3: WWW-suunnitelman teko on totista puuhaa, ainakin Helena Venäläisen ja Erkki Pöyhösen mielestä.

Systeemityö-yhdistys

Johtokunta 2001

hallitus.sytyke@pcuf.fi
http://www.pcuf.fi/sytyke

Pekka Forselius
(puheenjohtaja)
STTF Oy (Software Technology Transfer Finland Oy)
Porslahdentie 23 G 40, 00980 Helsinki
puh. 050 - 51 60 416
fax. 09-34 42 771
pekka.forselius@sttf.fi

Päivi Hokkanen
(varapuheenjohtaja)
SysOpen Yhtiöt Oy
Pasilankatu 4 B, 00240 Helsinki
puh. 09-613 4711, 040-700 4993
fax. 09-613 4711
paivi.hokkanen@sysopen.fi

Erkki Pöyhönen
Nokia Research Center,
Software Technology Laboratory
PL 407, 00045 NOKIA GROUP
puh. 07180 37595
fax. 07180 36855
erkki.poyhonen@nokia.com

Helena Venäläinen
FD FinanssiData Oy
Teollisuuskatu 1 B,
PL 308, 00101 Helsinki
puh. 09-404 3690, 050-568 6690
helena.venalainen@osuuspankki.fi

Pirkko Kallaperä
Pohjolan Systeemipalvelu Oy
00013 Pohjola
puh. 010 559 2786, 050-567 9352
fax. 010 559 3906
pirkko.kallapera@pohjola.fi

Hannu Kokko
SysOpen Object Team Oy
Pasilankatu 4B, 00240 Helsinki
puh. 040-751 6232
hannu.kokko@sysopen.fi

Heini Holopainen
TietoEnator Oyj
Public Sector, dGov
Tietotie 6, 02100 Espoo
puh. 09-8625 3480, 040- 513 8492
heini.holopainen@tietoenator.com

Varajäsenet

Lauri Laitinen
Kts. Systeemityö-lehden päätoimittaja

Minna Oksanen
TietoEnator Oyj
Kutojantie 10, PL 33, 02631 Espoo
puh. 09-8626 3272, 040-577 6640
fax. 09-8626 2902
minna.m.oksanen@tietoenator.com

Liittokokousedustajat

Lauri Laitinen
lauri.laitinen@nokia.com

Silja Räisänen
silja.raisanen@pohjola.fi

Pekka Forselius
pekka.forselius@sttf.fi

Internet-vastaava

Lauri Laitinen

Toimisto, sihteeri, Systeemityö-lehden tilaukset ja talouden hoito

Anne-Maj Viio
Henrikintie 7 A, 00370 Helsinki
puh. 09-5607 5363
fax. 09-5607 5365
viio@vt.inet.fi

Jäsenyysasiat

Jäsenneksi liittymis- tai jäsentietojen muutoslomakkeiden tilaus:
Tietotekniikan liitto ry, Piia Heiniö
Lastenkodinkuja 1 A (5.krs),
PL 325, 00181 Helsinki
puh. 09-476 5800
http://www.ttlry.fi

Systeemityö-lehti

Julkaisija

Systeemityöyhdistys
SYTYKE ry

ISSN 1237-0525
8. vuosikerta no 2.
Painos 2400 kpl.
Paino: Tuokinprint Oy
Sirrikuja 4 C, 00940 Helsinki
puh. 09-342 4150
tuokinprint@co.inet.fi

Toimitussihteeri

Susanna Koskinen,
Toimistopalvelu Hennax Tmi
Henrikintie 7 A, 00370 Helsinki
puh. 09-5617 830
fax. 09-5607 5365
susanna.koskinen@
hennax.inet.fi

Päätoimittaja

Lauri Laitinen
Nokia Tutkimuskeskus
PL 407, 00045 Nokia Group
Itämerenkatu 11-13, 00211 Hki,
puh. 07180 36551 (NRC)
Tornitaso 3 A 37, 02120 Espoo
puh. 09-465 959
lauri.laitinen@nokia.com

Toimituskunta 2/2001

Silja Räisänen ja Sujuva systeemityö-työryhmä

Seuraava numero:

3/2001 Ylläpito
Toimituskunta: Pekka Forselius

Tilaukset

Systeemityölehti sisältyy yhdistyksen Tietotekniikan liiton suositusten mukaiseen yhdistyksen jäsenmaksuun, joka vuonna 2001 on 360, 240, 240, 150 tai 80 mk/vuosi.
Vuositilaus 100 mk. Irtonumerot 30 mk. Tilaukset yhdistyksen toimistosta.

Internetin WWW-sivut

http://www.pcuf.fi/sytyke

Systeemityö-lehden sisällysluettelo 1988- ja useimmat artikkelit PDF-muodossa.

Ilmoitushinnat

	mv	4 väriä
Takakansi	----------	6.000 mk
Sisäkannet (K1, K2)	----------	5.000 mk
Sisäsivut A4	2.500 mk	4.500 mk
Sisäsivut A5	1.500 mk	3.500 mk
Sisäsivut A6	500 mk	----------