Kunnia vastuullisille äideille ja isille!

Lapsen tekeminen on monen mielestä hauskempaa kuin heidän kasvattamisensa vastuullisiksi yhteiskunnan jäseniksi. Lähes kuka tahansa pystyy lapsen tekoon, mutta aika moni ei siinä vaiheessa edes ajattele omaa kasvattajan vastuutaan. Ensimmäisen kerran asia saattaa välähtää mieleen joitakin viikkoja myöhemmin, kun raskaus varmasti todetaan. Osa päätyy raskauden keskeytykseen, mutta kovin helppo päätös ei ole sekään. Yleensä suhteet itseen, ympäristöön ja läheisiin joutuvat siinä tilanteessa liian lujille, maailma täyttyy syytöksistä, itsesyytöksistä ja katumuksesta.

Sovellusten kehittäminen on monien systeemityöläisten mielestä hauskempaa kuin niiden ylläpitäminen. Kaikkein kauheinta on jonkun muun kehittämän sovelluksen ylläpitäminen. Voi kuinka moni sovellus onkaan saanut alkunsa hetken huumassa, pienenä omien rajojen ja teknisten mahdollisuuksien kokeilemisena. Kun sitten on saatu aikaan jotakin käyttäjälle näkyvää ja tätä miellyttävää, ei peliä uskalletakaan panna poikki. Näin syntyvät ei-toivotut sovellukset. Kun kehittämisestä, ikään kuin huomaamatta, tulee sitten ylläpitoa ja virheiden korjailua, katoaa idean alkuperäinen isä yleensä hiljaa takavasemmalle. Siinä on sitten äiti yksin onnettoman sovelluksensa kanssa.

Toukokuun toinen sunnuntai juhlimme äitiä. Tasavallan presidentti palkitsee ansioituneita äitejä, jotka ovat hyvin usein joutuneet yksinään kantamaan monilapsisen perheen hoitovastuun, siinä onnistuen. Ehkä jo lähivuosina myös vastuulliseksi todetut isät huomioidaan samalla tavalla isänpäivänä, mutta kuka muistaa sovelluksen ylläpitäjiä?

Ehkä me sytykeläiset, Suomen ainoan vastuullisen systeemityöyhdistyksen jäsenet voisimme ottaa tämän asian hoitaaksemme. Vuoden ohjelmistoinnovaatioista ja kehittämisprojekteista on jo jaettu palkintoja, mutta nostetaan nyt kunniaan ne jotka sen todella ansaitsevat! Miltä tuntuisi olla ”Vuoden 2001 jatkokehittäjä”, ”Vuosikymmenen jatkuvan palvelun sankari”, ”Kuukauden kunnossapitäjä” tai ”Viikon virheentappaja”? Pokaali hyllyssä tai paras autopaikka käyttöön kuukaudeksi piristäisi kummasti jokaista ennen niin vähän arvostettua ammattilaista, sitä tärkeimmän työn tekijää. Otetaan tämä vakavasti, niin valtakunnassa kuin kukin omissa yrityksissämme ja vaikkapa tiimeissäkin. Jo tällä viikolla voi aloittaa!

Pekka Forselius, Sytyke ry:n puheenjohtaja

Taitossa sattunut virhe...

Edellisessä numerossa (2/01) oli taitossa sattunut paha kömmähdys Eija Hamina-Mäen artikkelissa “Tiimeistä tehoa projektityöhön” sivulla 17. Lipsahdus oli poistanut osan artikkelista kokonaan, jonka johdosta tavoitteena ollut viesti ei välittynyt lukijoille.

Artikkelin kokonaisuudessaan voi lukea Systeemityöyhditys Sytyke ry:n www-sivuilta www.pcuf.fi/sytyke.

Ylläpitomenettely

Minna Oksanen,
TietoEnator Digital Finance Oy

Johdanto

Ylläpito, sovellushallinta tai vastaaava järjestelmien käyttöönoton jälkeen tapahtuva kehitystyö systeemityön osana on jäänyt vähälle huomiolle julkisuudessa, esim. tutkimuksessa ja koulutuksessa - ja myös systeemityömalleissa. Pääosin systeemityömallit on tehty uuden tietojärjestelmän rakentamisen jäsentämistä varten.

Aikaisemmin ylläpitoa on pidetty itsestäänselvyytenä ja luonteeltaan teknisenä. Rakenteellisia syitä ylläpidon suureen määrään ei olla mietitty eikä ratkaisuja ylläpidon tehokkaaksi ohjaamiseksi etsitty. Huomiota on kiinnitetty vain ylläpitotyötä mahdollisesti helpottaviin tekniikoihin ja työkaluihin. Ylläpitotyötä prosessina ja vaiheina ei ole juurikaan mallinnettu, vaikka yleisesti tunnustetaankin, että systemaattisessa dokumentoinnissa sekä virhe- ja puuteilmoitusten kirjaamisessa tai tuotannossa toimivan järjestelmän toimintakunnon seuraamisessa olisi parantamisen varaa.

Ylläpitomalli

Ylläpitotyössä törmätään usein seuraaviin ongelmiin: muutoksia ei tehdä selkeinä projekteina, töitä ei niputeta; kaikkia käyttäjien muutospyyntöjä ei kirjata; muutoksia ei kuvata riittävän systemaattisesti.

Hyvässä ylläpitomallissa on jäsennetty sitä työtä, joka alkaa järjestelmän käyttöönotosta. Lähtökohtana on antaa toimintamalleja sekä kertaluontoisten ylläpitoprojektien hallintaan että jatkuvana toimintana tapahtuvaan ylläpitoon, lähinnä häiriöja muutoshuoltoon. Ylläpitomallin tehvävänä on toimia käytännön työssä oppaana ja siksi siihen kerätään hyväksi havaittuja ohjeita, dokumenttipohjia.

Ylläpitomalli ei ole suoraan mikään tekniikka tai työkalu eikä sen käyttäminen edellytä mitään tiettyjä apuvälineitä vaan se on systeemityömalli ylläpidon tehokkaan ohjauksen ja työnkulkujen aikaansaamisen tueksi. Kyseisessä ylläpitomallissa on pyritty vastaamaan seuraaviin haasteisiin:

1. Järjestelmän jatkokehitys on organisoitava kokonaisuuksiksi ja vietävä läpi projekteina
2. Käyttäjän tarpeet on seulottava haluista ja luotava tehokas käytäntö systemaattiselle muutospyyntöjen sekä virhe- ja puuteilmoitusten kirjaamiselle ja analysoinnille
3. Muutosten testauskäytäntöjen tulee olla riittävän kattavia, koska muutokset tehdään tuotannossa toimiviin järjestelmiin
4. Muutosten toteutus tulee olla hallittua, selkeisiin kokonaisuuksiin jaettua, jolloin muutosten suunnittelu ja seuranta on helpompaa. Myös itse ylläpitotyön tekijät saavat vaiheistuksen myötä yksilölliset työsuunnitelmat
5. Järjestelmästä on ylläpidettävä loogisia ja toiminnallisia (ei vain käyttöympäristöön sidottuja) "business-tason" dokumentteja, joiden avulla voidaan arvioida ympäristömuutosten, esim. liiketoiminnan muutoksen vaikutus järjestelmään
6. Mallia ei tarvitse ottaa käyttöön yhdellä kertaa, vaan käyttöönotto voidaan joustavasti sovittaa tarpeen mukaan, myös vanhoihin järjestelmiin. Esim. voidaan määritellä järjestelmän kriittisyyden ja ohjelmistoiän mukainen tavoitetaso
7. Järjestelmän elinkaaritiedot talletetaan, jotta ylläpito ei jää liian henkilösidonnaiseksi. Myös seuraavien ohjelmoijien ja muutosten suunnittelijoiden tulee saada helposti selville aiemmin tehtyjen muutosten syyt ja vaikutukset järjestelmään
8. Ylläpito organisoidaan ja toteutetaan suunnitellusti. Mallin käyttöönoton yhteydessä kukin sovellusalue tekee ylläpitomenettelyn kuvauksen ja sovelluskohtaisia ylläpitosuunnitelmia. Ylläpitomenettelyn kuvauksessa kuvataan yksityiskohtaisesti miten mallia sovelletaan juuri kyseisessä ympäristössä ja ylläpitosuunnitelmassa kuvataan sovelluskohtaisesti ylläpidolle asetettavat tavoitteet.

Ylläpitosopimus ja -suunnittelu

Ylläpidon vastaanoton yhteydessä tehdään ylläpitosopimus, jossa: sovitaan ylläpidon kohteesta ja ajasta; nimetään sekä asiakkaan sovellusvastuuhenkilöt että toimittajan ylläpitovastuuhenkilöt ja sovitaan heidän vastuut ja toimivaltansa; sovitaan sovelluksen kiireellisyysluokasta ja hinnoitteluperusteista ja määritellään jatkuvan ylläpidon hallinta- ja projektikäytännöt.

Ylläpitosopimuksen kanssa samanaikaisesti tehdään myös ylläpitosuunnitelma, jossa määritellään sovelluskohtaisesti ylläpidolle asetettavat sisällölliset tavoitteet. Jos ylläpitoyhteistyö on laajaa tehdään lisäksi sovellusalueittain ylläpitomenettelyn kuvaus erikseen ao. asiakasympäristöön. Soveltamisohjeissa kerrotaan yksityiskohtaisesti, miten ylläpitomallin pelisääntöjä sovelletaan.

Ylläpitosuunnitelman tavoite on määritellä suuntaviivat sovelluksen jatkuvalle ylläpidolle, ja tätä kautta edesauttaa sovelluksen kunnossapitoa koko sovelluksen elinkaaren ajan. Ylläpitosuunnittelma tehdään ylläpitosopimuksen laatimisen yhteydessä eli yleensä pian sovelluksen käyttöönoton jälkeen.

Ylläpitosuunnitelma tehdään yhdessä asiakkaan kanssa sitä päivitetään tarvittaessa eli yleensä silloin, kun sovelluksen koko tai kriittisyys olennaisesti muuttuvat. Ylläpitosuunnitelmaan tulevista suurehkoista muutoksista, esim. muutokset tavoitteissa ja ylläpidon kohteessa, päättää sovellusseurantaryhmä. Pienemmistä muutoksista tiedotetaan kaikille asianosaisille.

Ylläpitohenkilön vaihdosmenettelyt kuvataan myös suunnitelmaan ja ylläpitohenkilöiden kouluttaminen kuvataan huoltokoulutussuunnitelmassa, joka sisältää koulutuksen tavoitteet, perehtymisen asiakkaan toimintaan, käytännön järjestelyt ja yksityiskohtaisen koulutusohjelman, jonka läpikäytyään henkilö pystyy sovelluksen ylläpitovastuulliseksi.

Ylläpitosuunnittelun tarkoituksena on muotoilla pidemmän aikavälin tavoitteet ja menettelytavat, joita toteutetaan sovelluksia ylläpidettäessä.

Ylläpitosuunnittelun tuloksena syntyy: sovelluskohtainen myös asiakkaalle tarkoitettu ylläpitosuunnitelma (MITÄ ylläpidetään eli asiat) ja ylläpitomenettelyn kuvaus. Mallin soveltamisohje (MITEN ylläpidetään eli prosessit) kuhunkin ympäristöön. Koska ylläpito on luonteeltaan jatkuvaa, antavat ylläpitosuunnittelun lopputuloksena syntyvät asiakirjat puitteet pitkän aikavälin, koko sovelluksen elinajan jatkuvalle ylläpidolle. Myös laatutavoitteet sisältävät pitkän aikavälin tavoitteita ja ovat suhteellisen pysyviä.

Yläpidon lajit

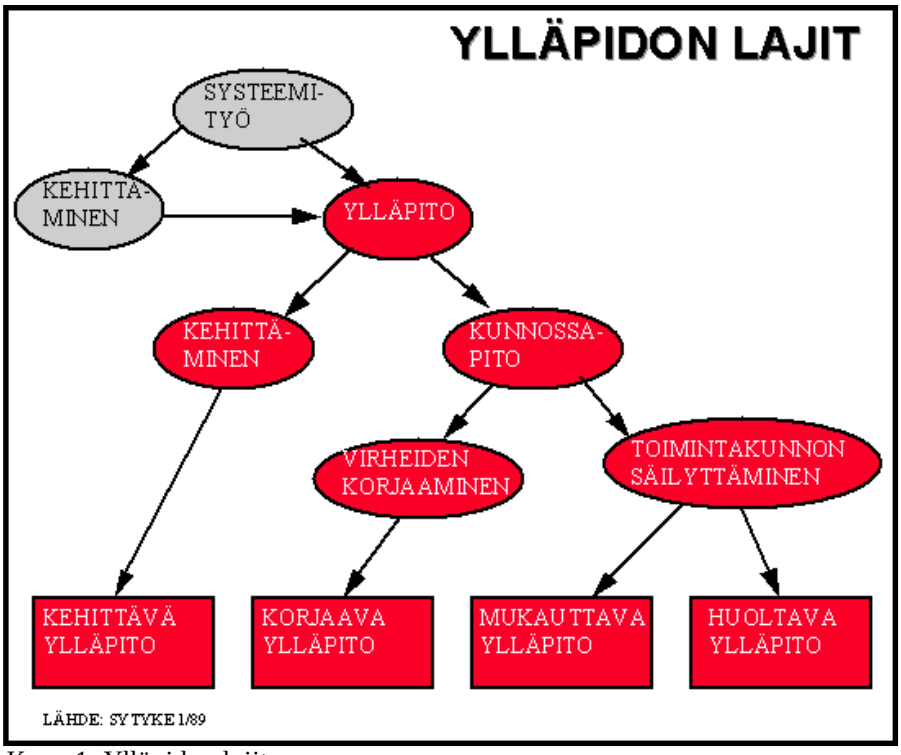
Ylläpito voidaan määrittää systeemityös käsitehierarkian kautta. Systeemityö jakaantuu järjestelmien kehittämiseen ja ylläpitoon. Ylläpito puolestaan jakautuu tuotantokäytössä olevan järjestelmän kehittämiseen ja kunnossapitoon. Kunnossapito koostuu vielä virheiden korjaamisesta ja järjestelmän toimintakunnon säilyttämisestä. Lajit voidaan kuvata oheisen kuvan (alla) mukaan. (Sytyke 1/89).

Järjestelmän jatkuvaan ylläpitoon kuuluvat:

9. häiriöhuolto, jolla tarkoitetaan sovelluksen käyttöä estävien virheiden välitöntä korjaamista
10. pienmuutosten hallinta, jolla tarkoitetaan pääasiassa sopeuttavien sovellusmuutosten tekoa. Ylläpidon syy on usein tietojärjestelmän ulkopuolella. Pienmuutoksia voivat aiheuttaa esim. sovelluksen liittymien tarpeet, viranomaisen vaatimat tiettyyn päivämäärään sidotut pienmuutokset tai pienten käyttäjää häiritsevien puutteiden poistaminen järjestelmästä
11. sovellusseuranta ja laaduntarkkailu, joilla tarkoitetaan tietyin aikavälein tapahtuvaa ja koko sovelluksen elinajan kestävää sovelluksen kunnossapitoa. Sovellusseurantaa ovat mm. sovelluksen liiketoiminta-alueen kehityksen ja huoltotilanteen seuranta
12. ylläpitosuunnittelu, jolla tarkoitetaan sovellusalueilla tapahtuvaa ylläpitotyön kehittämistä (kirjataan ylläpitomenettelyn kuvaukseen) ja sovelluskohtaisten ylläpitotavoitteiden kirjaamista ylläpitosuunnitelmiksi.

Jatkuvan ylläpitovalmiuden panostusmäärään ja sitä kautta kustannuksiin vaikuttavat erilaiset sovelluksen laajuutta ja kriittisyyttä kuvaavat muun muassa seuraavat asiat: laiteympäristöt, laajuus (massat volyymit), työvaiheet, kriittisyys, monimutkaisuus, liittymät, tulosteet, tosiaika ja erätoimintojen suhde.

Jatkuvan ylläpidon aikana syntyneet dokumentit talletetaan ylläpitomenettelyn mukaisesti.



Kuva 1: Ylläpidon lajit

Ylläpito projektina

Kertaluonteista ylläpito on silloin kun se on järjestelmän toimintaa kehittävää ja mukauttavaa ylläpitoa ja se tehdään projektina. Ylläpitoprojektit ovat aikaan sidottuja ja erikseen käynnistettäviä. Useimmat ylläpito hankkeet tulisi niputtaa suuremmiksi kokonaisuuksiksi eikä tehdä yksittäisinä muutoksina.

Ylläpito projekti sisältää samat vaiheet kuten systeemityöprojekti yleensäkin: muutoksen aloitus, muutoksen toteutus ja muutoksen käyttöönotto.

Projektin alussa kerätään ja luokitellaan jatkuvan ylläpidon aikana kerätyt muutospyynnöt. Niiden pohjalta määritetään projektin sisältö. Ensimmäisenä tprojektissa toteutetaan kiireellisimmiksi luokitellut tehtävät, jotka haittaavat sovelluksen toimintaa. Mukaan projektiin voidaan ottaa mys niitä muutospyyntöjä, jotka ovat ns. Nice to have. Jatkuvan ylläpidon aikana kaikki muutospyynnöt on kirjattu sovelluksen huoltokirjaan, josta ne nyt kasataan projektin tehtäviksi. Projektimuotoisessa ylläpidossa käytetään projektin aikana projektikansiota.

Muutostyötä helpottamaan on olemassa sovelluksen sovelluskirjan kuvaukset, joita ylläpidetään muutostyö edistyessä. Muutosten valmistuttua tehdyt muutokset kuvataan huoltokirjaan historiatiedoksi. Muutoksia toteutettaessa ja testattessa on muistettava, ettei sovelluksen muu toiminta häiriinny. Siksi muutosten testauksen yhdeydessä on tehtävä regressiotesti, jolla testataan muutoksen vaikutukset muuhun sovelluslogiikkaan.

Muutosten ja häiriöiden hoitaminen

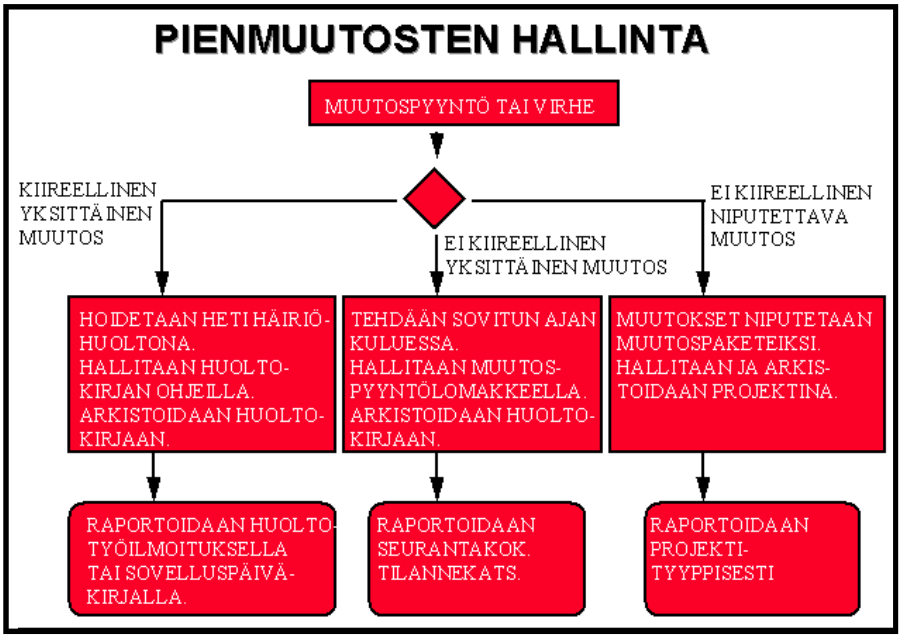
Pienmuutosten hallinnan tavoite on tehostaa ylläpitoa silloinkin, kun ei ole kyse kiireellisestä häiriöhuollosta, ja kun sovelluksen kehittämistä ei ole tarkoituksenmukaista organisoida projektiksi. Yleensä pienmuutokselle on tyypillistä, että sen ajoittuminen on riippumaton itse tietojärjestelmän toiminnasta, eivätkä tietojärjestelmän ylläpitäjät voi suuremmin vaikuttaa pienmuutoksen käyttöönottohetkeen. Useimmiten pienmuutoksen tekemisen syy on tietojärjestelmän ulkopuolella. Tyypillisiä pienmuutoksia ovat:

Ajoitetut, sovitun ajan sisällä tehtävät sopeuttavat pienmuutokset esim. viranomaisen määräyksestä tehtävät pienmuutokset tai teknisen toimintaympäristön muutoksesta (laitteisto, varusohjelmisto) aiheutuvat ylläpitotyöt sekä sovelluksen liittymien tarpeista johtuvat pienmuutokset. Toiseksi pienmuutokset ovat sovelluksen käyttäjiä häiritsevien virheiden/puutteiden hallittu korjaaminen.

Yleensä impulssi pienmuutoksen tekemiseen tulee joko asiakkaan vastuuhenkilöltä tai ylläpitoa koordinoivalta kehittämisryhmältä. Aikataulun määrää usein joku tietojärjestelmän ulkoinen tekijä, esim. viranomainen tai järjestelmän liittymä. Tällaisessa tapauksessa ei yleensä ole aikaa eikä tarkoituksenmukaistakaan organisoida projektia, koska työ täytyy saada käyntiin melko nopeasti ja se on pienuutensa vuoksi helposti hallittavissa ilman projektimenettelyä.

Pienmuutos on yleensä selkeä rajattu kokonaisuus, jonka tekemisestä on vastuussa ylläpitovastuuhenkilö ja tehtävän suorittaminen perustuu hänen ammattitaitoonsa ja kokemukseensa sovelluksesta. Pienmuutosten hallinta perustuu muutospyyntöjen ja virheilmoitusten huolelliseen dokumentointiin. Muutospyyntölomakkeelle ja sen liitteisiin kootaan kaikki muutostyöhön liittyvät asiakirjat: asiakkaan tilaus, arvio työstä asiakkaalle, sisältää yleensä myös työsuunnitelman , käsittelysäännöt asiakkaan vastuuhenkilöiltä tai käyttäjiltä ja testaustulokset.

Pienmuutosten hallintaja asiakkaan vastuuhenkilöiden ja ylläpitovastuuhenkilöiden tiiviiseen yhteistyöhön. Ylläpitovastuuhenkilö sopii työn aloittamisesta asiakkaan vastuuhenkilön kanssa. Työn edetessä ylläpitovastuuhenkilö kirjaa sen elinkaaritiedot joko muutospyyntölomakkeelle tai huoltokirjaan sovitun ylläpitomenettelyn mukaisesti. Työn valmistuttua dokumentit talletetaan huoltokirjaan. Jos tehty pienmuutos sisältää



Kuva 2: Pienmuutosten hallinta

toiminnallisia muutoksia niin myös sovelluskirja tulee päivittää muutoksen käyttöönoton yhteydessä.

Asiakkaan vastuuhenkilölle pienmuutoksen elinkaaresta ilmoitetaan, jos mahdollista sähköpostilla, tai puhelimitse vähintään työtä aloitettaessa ja käyttöönottopäivänä. Kiireellisissä tapauksissa työ voidaan aloittaa asiakkaan vastuuhenkilön suullisella luvalla ja hoitaa tilaus työn tekemisen aikana kuntoon. Muulloin arvio pienmuutoksesta toimitetaan asiakkaan vastuuhenkilölle, joka päättää työn käynnistämisestä. Työn toteutumisesta kustannuksineen raportoidaan kehittämisryhmän tilannekatsaukseen.

Huoltovastuuhenkilön rooli

Ylläpitomenettelyyn kuuluu myös kuvata vastuullisten henkilöiden toimet. Ylläpitosovpimuksessa on nimetty huoltovastuuhenkilöt ja kuvattu heidän toimenkuvansa.

Oheinen kuva (edellisellä sivula) kuvaa pienmuutosten hallinnan suhdetta aiemmin kuvattuihin ylläpidon lajeihin (häiriöhuolto ja ylläpito projektina) huoltovastuu henkilön näkökulmasta.

Huoltovastuuhenkilön tehtäviin kuuluu pyrkimys sovelluksen häiriöttömään toimintaan. Häiriön sattuessa hänellä on vastuu häiriöhuollosta yhdessä muiden huoltovastaavien kanssa.

Huotovastuuhenkilön tulee: seurata vastuullaan olevan sovelluksen tilaa ja ongelmia sekä tehdä ehdotuksia parannuksista; vastata omasta oppimisestaan; olla mukana häiriöiden hoitamisessa, kunnes häiriö on selvitetty ja hoidettu loppuun saakka informoida häiriötilanteessa liittymäsovelluksia; sopia kursseista, lomista ja muista poissaoloista muiden huoltohenkilöiden kanssa niin, että kaikkien sovelluksen huoltohenkilöiden poissaolot eivät ajoitu samaan aikaan.

Huoltovastuuhenkilön toimivaltaan kuuluu vaatia: itselleen huoltotöiden hoitamiseen tarvittavaa tukea kokeneemmilta myös silloin kun häiriöhuolto tehdään ns. normaalin työajan ulkopuolella; liittymien asiantuntijoiden tukea tarvittaessa häiriön hoitamiseen oman järjestelmäalueen ulkopuoleltakin, myös asiakkaalta.

Häiriön sattuessa vastuuhenkilön tehtävänä on ensin yhdessä asiakkaan kanssa analysoida häiriön laatu ja kriittisyys ja hänellä Huoltovastuuhenkilöllä on kokonaisvastuu häiriöstä loppuun asti, ellei toisin sovita.

Jos muutos on tehtävä heti huoltotyönä, pitää käydä läpi seuraavat kohdat: selvittää virheen ja sen korjauksen vaikutukset, muista kokonaisuus; harkita vaihtoehtoisia ratkaisuja kuten virheen aiheuttaneen tapahtuman ohitus ilman ohjelmakorjausta tai ajon siirtäminen myöhemmin ajettavaksi; tehdä muutos tarvittaessa ja testata huolella (testipaketti); päivittää huoltokirja ja sovelluskirja ja informoida liittymäsovelluksia, käyttäjiä, sovellusvastuuhenkilöitä ja Atk-keskusta.

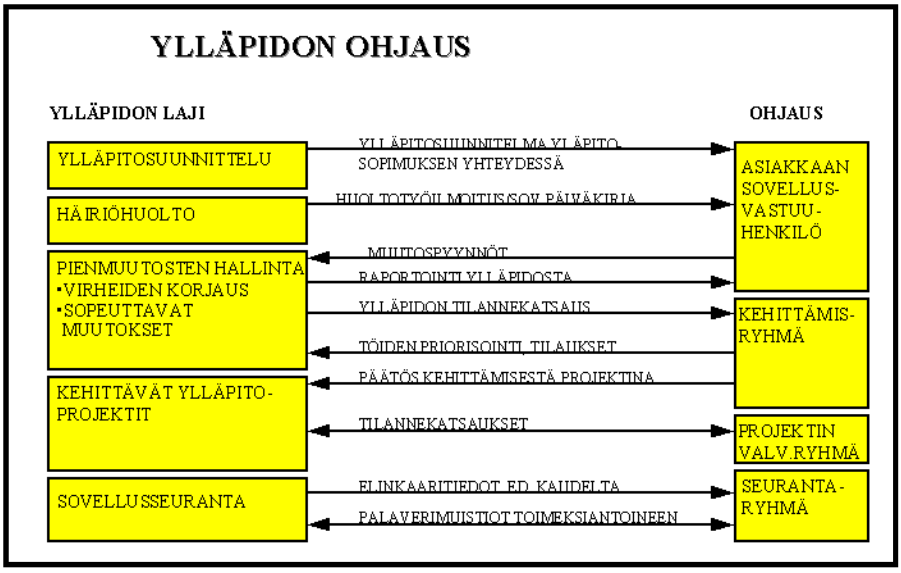
Häiriöhuollon helpottamiseksi selvitetään etukäteen valmiita ratkaisuja häiriöstä toipumiseen. Näitä ovat mm: valmiit ajojonot tietokantapalautuksiin; voidaanko ohjelman suoritus aloittaa alusta; miten ohjelman toimintaa voidaan jatkaa ilman aloitusta alusta; mahdollisuudet ohittaa virheen aiheuttava tapahtuma; pidetään testipaketti aina testausvalmiina ja kirjataan selvitetyt häiriöt huoltokirjaan. Niistä voi olla apua seuraavalla kerralla.

Yhteenveto

Kun yrityksessä otetaan ylläpitomenettely käyttöön on mietittävä ylläpitotoiminto kokonaisuutena. Sidosryhmien on tiedettävä oma roolinsa. Ylläpitomenettelyn olisi kuuluttava henkilöstön koulutusohjelmaan. Sen käyttöönotto vaikuttaa toimintotapojen muutokseen. Oheisen kaavion (alla) avulla näkyy, miten tässä esimerkkinä olevassa ylläpitomenettelyssä ylläpidon vastuujako ja ohjaus on käsitelty.

Kyseisessä yrityksessä ylläpitomenettely on käytössä kaikissa ylläpitosovelluksissa ja sen on todettu helpottavan ylläpitotyötä.

Koonnut: Minna Oksanen
Lähde: TietoEnatorin Finanssiyksikön ylläpitomalli FILMA



Kuva 3: Ylläpidon ohjaus

Asiakasyhteistyö jatkuvassa palvelussa

Minna Oksanen,
TietoEnator Digital Finance Oy

Oletko koskaan huokaissut työpäivän jälkeen, että olisipa kiva joskus tehdä jotain ihan uutta sovellusta? Johtuisiko se siitä, että suurin osa systeemityöstä, jopa 70 – 90 %:a on valmiiden järjestelmien ylläpitämistä -työtä, jota kuitenkaan kukaan ei haluaisi tehdä.

Miltä sitten ylläpito näyttää asiakkaan näkökulmasta? Asiakkaalle ylläpito on tärkeää. Asiakkaan ei kannata hankkia uusia sovelluksia, jos vanha sovellus on pienin muutoksin muutettavissa toimivaksi. Asiakkaat laskevat järjestelmien kustannuksia ja uusi järjestelmä tulee pääsääntöisesti kalliimmaksi kuin vanhan järjestelmän ajantasaistaminen. Hyvä esimerkki on Y2K, jolloin asiakkaiden kaikki järjestelmät käytiin läpi ja tehtiin muutoksia. Vain muutama järjestelmä rakennettiin uudelleen, koska vanhat järjestelmät eivät enää olleet tehokkaita. Paras anekdootti, mitä kuulin, oli se, kun henkilö huomasi muuttavansa omia 70-luvulla tekemiään ohjelmia. Itse asiassa kyseinen muutostyö elätti muutaman vuoden toimittajaorganisaatioita.

Suuri osa ylläpitotyöstä tapahtuu asiakkaan toimitiloissa tai ainakin asiakkaan kanssa ollaan jatkuvassa yhteistyössä. Olen kerännyt tähän artikkeliin muutamia ajatuksia, joita on syntynyt työskennellessäni useita vuosia asiakkaan luona ylläpitotehtävissä.

Ensimmäisenä avainkohtana on yrittäjyys. Pitäisi aina muistaa, kun on töissä asiakkaalla on kaikesta huolimatta oman yrityksen edustaja. Jos kuvittelee olevansa itsenäinen yrittäjä, on helpompi muistaa kahden yrityksen välinen rajapinta. Omakohtaisesti tiedän raja-aitojen kahden yrityksen välillä hälventyvän, kun on pitkään saman asiakkaan kanssa tekemisissä. Välillä tuntuu siltä, ettei kuulu minnekään. Omalla toimistolla ei tunne ketään ja vastaavasti asiakkaan tilaisuuksiin ja sisäisiin palavereihin ei voi osallistua. Jotkut väittävät, ettei asiakkaan kanssa saisi käydä edes syömässä tai kahvilla. Tärkeämpää on kuitenkin toimiva yhteistyö suhde asiakkaan kanssa. On vain muistettava se, että mistä asioista asiakkaan kuullen voi puhua. Ei ole soveliasta kertoa oman yrityksen tai yksikön sisäisistä asioista asiakkaalle.

Asiakkaan silmissä ylläpidosta vastaava henkilö on toimittajan edustaja. Tällöin tulisi muistaa, että pitää käyttäytyä kuin myyntimies. Lähtökohta on niinkin yksinkertainen kuin, että on osattava käytöstavat ja toimia asiakkaan odotusten mukaan. Asiakkaan kanssa toimiessa jopa pukeutumisella on merkitystä. Siisti vaatetus antaa kuvan pätevästä ja tehokkaasta toimittajasta.

Yhteistyön sujumiseksi tulisi muistaa, että pyritään samaan päämäärään. Vaikka tietyissä tilanteissa toimittaja ja asiakas ovat eri puolilla pöytää esim. neuvoteltaessa kustannuksista, niin ei ole tarkoituksenmukaista, että molemmat organisaatiot ajavat vain omia etujaan. Asiakkaan pyrkimys on aina kustannussäästöihin, vaikka se ei pitkällä aikavälillä olisikaan kannattavaa. Pitäisi sen sijaan saada asiakas ymmärtämään, mitä lisäarvoa hän saa pitkäaikaisesta kumppanuudesta.

Asiantuntijaorganisaatiossa asiakas ei aina ole oikeassa vaan toimittajan tehtävänä on tarvittaessa esittää asiakkaan kannalta järkeviä toimintatapavaihtoehtoja.

Mikäli omalla yrityksellä on toimiva ylläpitomenettely, on ylläpidon toteuttaminen ja hallinnointi huomattavasti helpompaa. On olemassa runko, jota käyttäen saa mallia ja tukea omaan toimintaan. Yksi yleisimmästä tilanteista on se, että asiakkaat toivovat muutoksia ohi virallisen muutoskäsittelyn. Jos näihin toiveisiin suostutaan, joudutaan aina siirtämään aikaisemmin sovittuja aikatauluja. Tällaisissa kannattaa pyytää asiakasta priorisoimaan tehtävät. Näin toimittajalle ei jää vastuuta mahdollisista aikataulujen muutoksista.

Toimintamallien lisäksi oma organisaatio voi myös tukea neu-

vomalla ongelmatilanteissa. On kuitenkin tärkeää, että asiakkaalle näkyy palvelu yhtenä kokonaisuutena, eikä niin että on olemassa ne tietyt gurut, joilta asiakkaat voivat suoraan kysyä ratkaisuja ongelmiin.

Asiakkaan kannalta ylläpitotyö kulminoituu niihin tilanteisiin, jolloin ylläpidosta vastuulliset henkilöt vaihtuvat joko työkierron kautta tai sitten henkilön vaihtaessa työnantajaa. Pahimpia tilanteita ovat ne, kun kuukauden irtisanomisajalla pitäisi kouluttaa seuraaja. Työnkierrossa siirto voidaan organisoida joustavasti niin ettei asiakas huomaa palvelun tasossa poikkeamaa. Jotta siirto saadaan onnistumaan, tarvitaan ylläpitosuunnitelmaa. Vastuu vaihdetaan suunnitelmien mukaan asiakas kokoajan huomioiden. Asiakkaan muutosvastarinta saadaan murrettua sillä, että asiakas näkee seuraavan vastuuhenkilön selviytyvän ja oppivan työtä. Uuden vastuuhenkilön tukena on aina edellinen vastuuhenkilö. Tavoitteena on se, että kukaan ei olisi koskaan yksin vaan sovelluksella olisi aina varavastuulliset. Asiakkaan näkökulmasta henkilön vaihdos tuo aina kustannuksia, mutta samalla uusi henkilö voi tuoda mukanaan uusia tehokkaampia työmenetelmiä ja näkökulmia toimintaan.

Yhteenvetona, jotta asiakasyhteistyö saadaan onnistumaan on siis oltava toimiva ylläpitomenettely ja hyvin suunniteltu sovelluskohtainen ylläpitosuunnitelma, jota myös noudatetaan.

Minna Oksanen,
Dw-asiantuntija,
TietoEnator Digital Finance Oy

Jälleen on pidetty

SYTYKKEEN PERINTEINEN LAIVASEMINAARI (5.-7.9.2001)

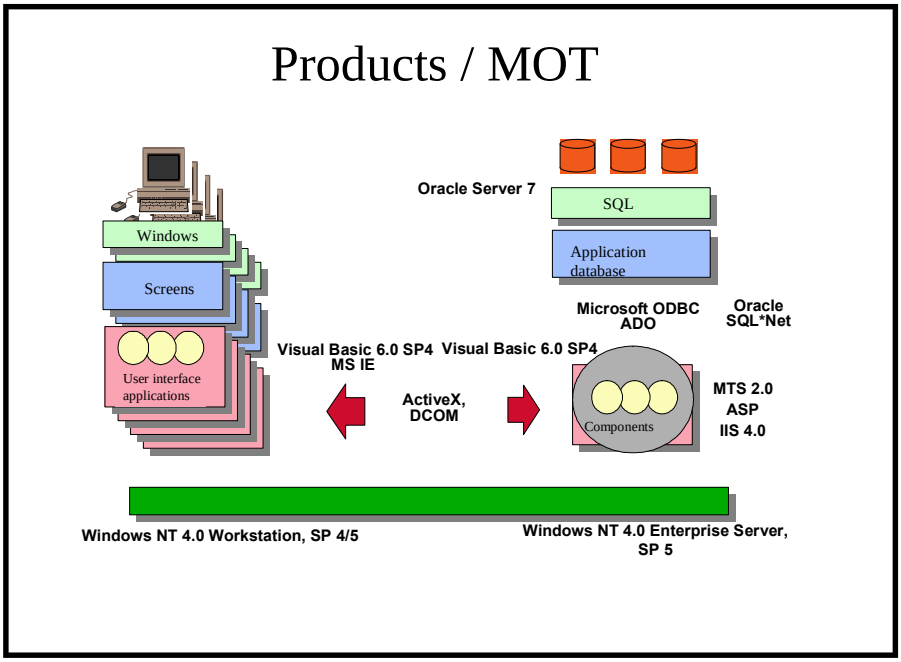
Miljoonat kiitokset kaikille luennoitsijoille ja sponsoroijille!

Enemmän juttua risteilystä seuraavassa Sytykkeen numerossa...

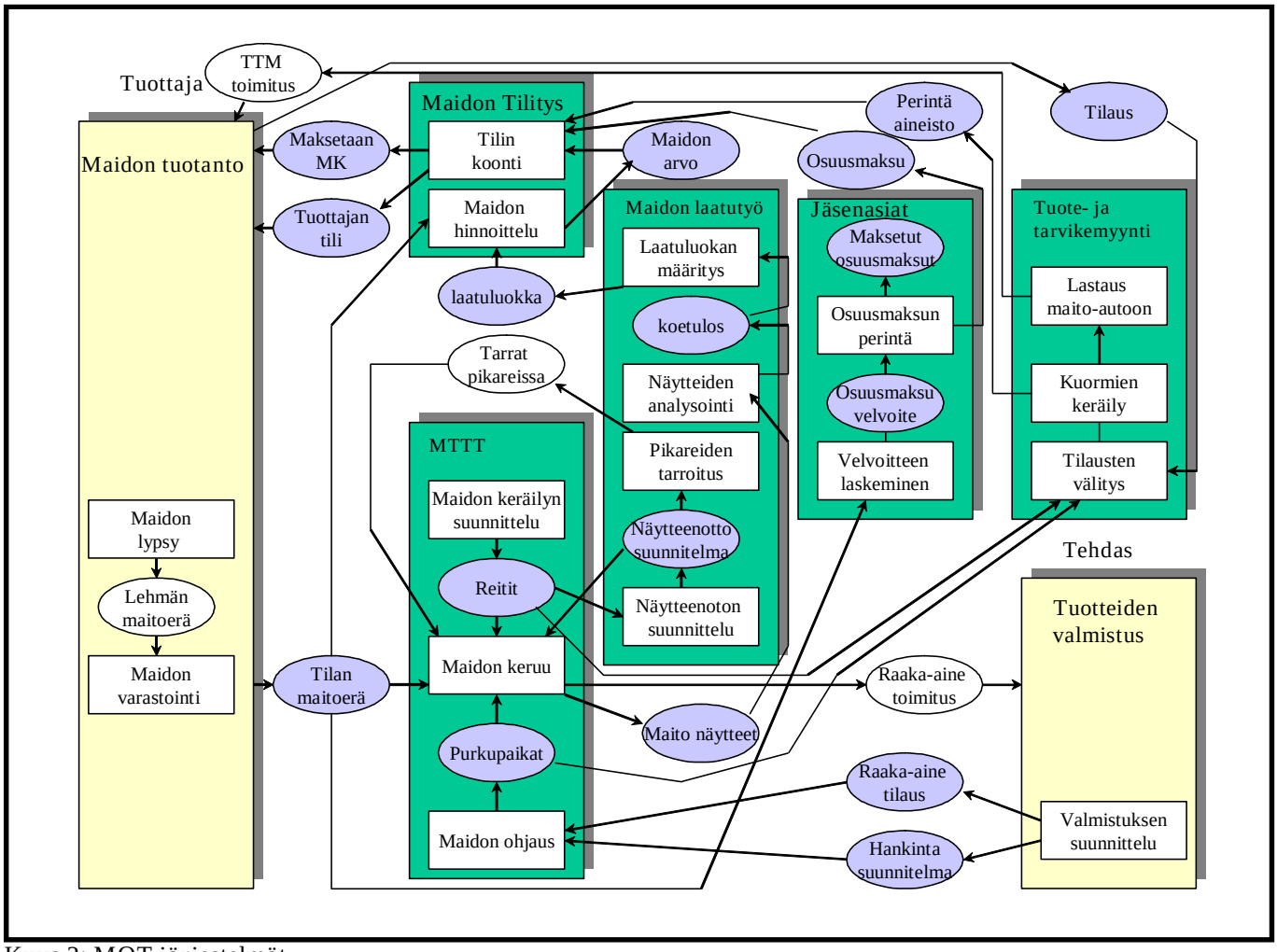
Laajan räätälöidyn järjestelmän ylläpito

Kimmo Vaikkola,
ICL Invia Oyj

Olemme ICL Inviassa toteuttaneet Valion ja valiolaisten osuuskuntien käyttöön laajan MOT-järjestelmäkokonaisuuden (MOT = Maidon Osto Tuottajilta). Järjestelmä on tehty kolmitasoarkkitehtuuriratkaisuna Microsoft-tekniikalla Oracle-tietokantapohjaisesti (kuva 1, MOT teknologia). MOT-järjestelmä (kuva 2, MOT-järjestelmät) on toteutettu yhteistyössä asiakkaiden kanssa lähtien liikkeelle asiakkaan omista prosesseista. Järjestelmä siis tukee prosesseja ja sillä hoidetaan keräilyreittien suunnittelu, maitoraaka-aineen keräily tiloilta, maidon laadun seuranta, raaka-aineen ohjaus



Kuva 1: MOT-teknologia.



Kuva 2: MOT-järjestelmät

tuotantolaitoksiin, osuuskuntien tilitykset tuottajille, osuuskuntien tuottajien jäsenyyteen liittyvät asiat ja lisäksi järjestelmän kautta välitetään tuottajien tarviketilauksia osuuskuntien myyntijärjestelmien välillä. Järjestelmään ja palvelun yhteyteen ollaan juuri lisäämässä maidontuottajille toteuttamamme internet-pohjaisen Valma-järjestelmän ylläpito. Kuvassa 3 on MOT toimintaympäristö.

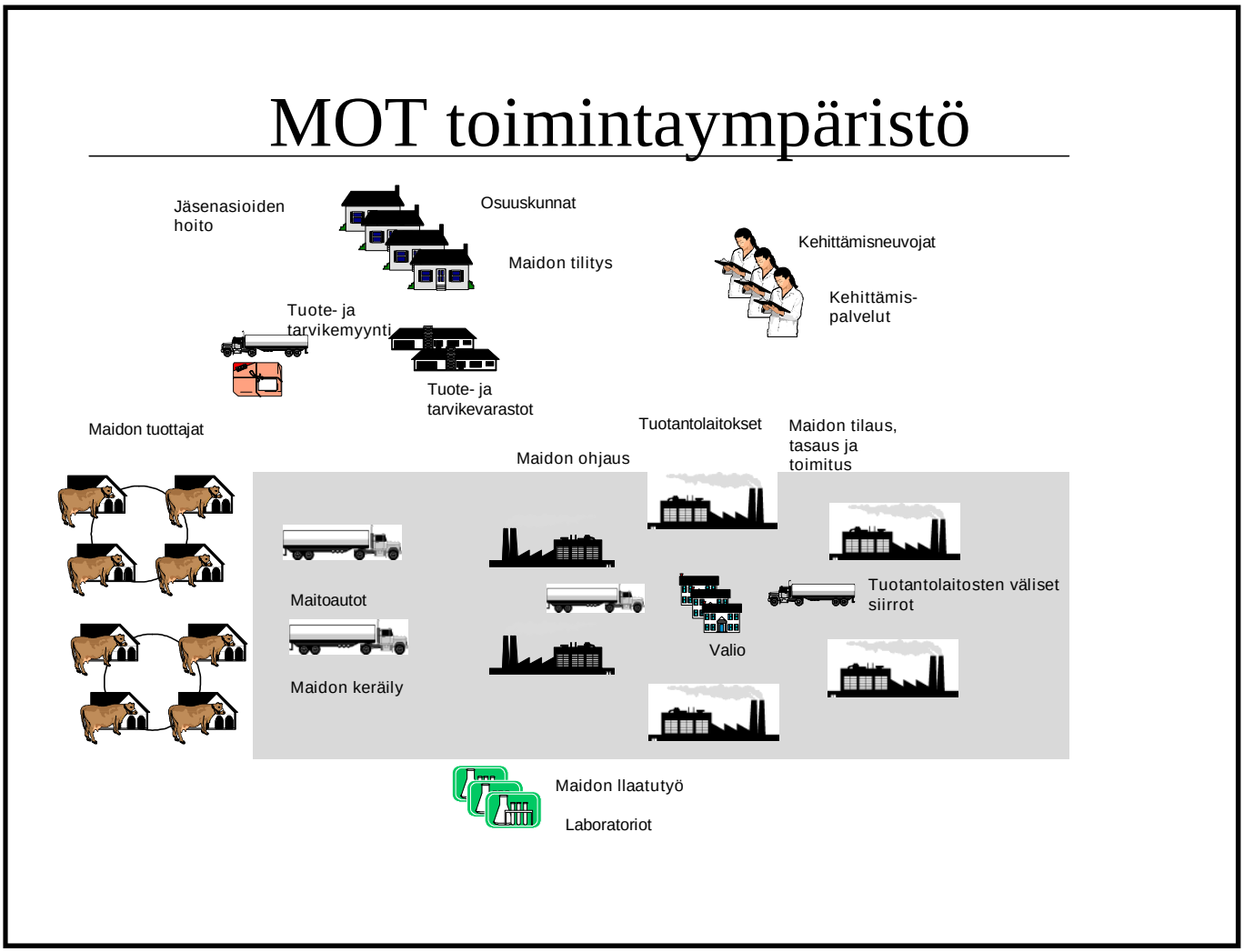
Järjestelmä koostuu tällä hetkellä useista sovellus- ja tietokantapalvelimista erillisissä tuotanto-, hyväksymistestaus- ja kehitysympäristöissä.

Liittymiä järjestelmästä sisään ja ulos on myös useita, mm. maitomäärät autojen automaattisista keräilyjärjestelmistä, maidon analyysitulokset laboratoriosta, veloitukset myyntijärjestelmistä ja tilitystiedot maksatukseen. MOT-järjestelmän käyttäjiä on yli 200 useissa erityyppisissä käyttäjäryhmissä, osa käytöstä tapahtuu selaimella ja osa työasemassa olevalla käyttöliittymäsovelluksella.

Tuotantokäyttö on alkanut vuonna 1998 ja järjestelmä on saavuttanut jo selkeän ylläpitovaiheen.

MOT-järjestelmän ylläpidon organisointi

ICL Inviassa tämäntyyppinen järjestelmä hoidetaan palveluprojektina. Projektiryhmä kootaan tarpeen mukaan ja riippuen järjestelmän kehitysvaiheesta asiantuntijoiden määrä voi hiukan vaihdella eri aikoina. Järjestelmien versiointi tietysti helpottaa palveluprojektien organisointia. Järjestelmän kehitysvastuu on yhteisellä kehitysryhmällä, jossa asiakasta edustavat eri osajärjestelmien prosesseista vastaavat henkilöt, lisäksi mukana on asiakkaan tietohallinnon edustus, toimittajan edustajina ovat asiakasvastuullinen palvelupäällikkö ja allekirjoittanut järjestelmän vastuullisena palvelupäällikkönä. Kehitysryhmä kokoontuu noin kerran kuukaudessa ja käsittelee kaikki eri prosessien hoitoon liittyvät järjestelmän kehitysasiat. Toimittajan puolella on erillinen järjestelmän käyttäjiä palveleva käytön tukiyksikkö joka palvelee puhelinneuvonnalla ja lisäksi sähköpostin välityksellä. Kaikki yhteydenotot kirjataan ICL Invian ServiceCenter-seurantajärjestelmään. Tuen apuna toimivat vielä järjestelmäasiantuntijat jotka ratkaisevat kullekin ServiceCenterissä kohdistettuja ongelmia. ServiceCenter on osoittautunut erittäin hy-

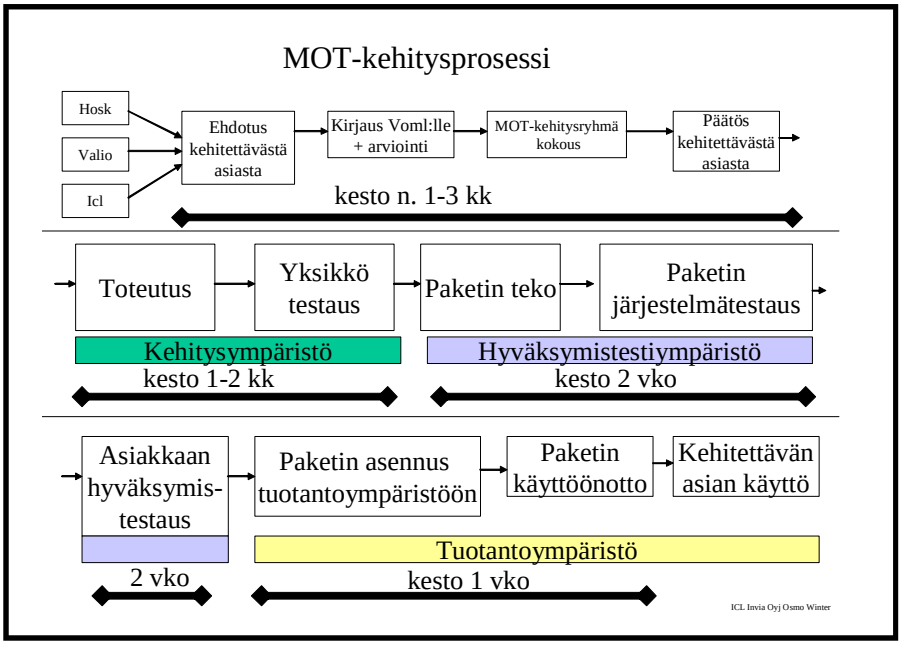


Kuva 3: MOT-toimintaympäristö

väksi yhteydenpitokanavaksi ylläpitotyössä kun asiantuntijat käytännössä ovat eri puolella maata ja osa tekee työtä etänä kotoa käsin. Lisäksi asiakkaan laitteistoja hoitavat vielä laitteistomanagerit. Projektissa on tällä hetkellä mukana 2-3 asiakastuessa työskentelevää, 1-2 järjestelmätestaajaa, 7-8 järjestelmäasiantuntijaa ja muutama laitteistomanageri, joilla toki on useiden muidenkin järjestelmien laitteistot hoidettavanaan. Päivittäistä kanssakäymistä helpottaaksemme kokoonnumme kerran viikossa lyhyehköön aamupalaveriin, joka käytännössä pidetään neuvottelupuhelimen välityksellä, koska emme kaikki työskentele samoissa tiloissa.

Ylläpidossa noudatettava prosessikaavio

Kukin osaprosessi tai kuka tahansa siinä työskentelevä voi tehdä kehitys- ja muutosehdotuksensa kehitysryhmän jäsenille (kuva 4, MOT-kehitysprosessi). Tarpeita kerätään myös puhelintukemme ja asiantuntijoidemme toimesta. Kuvan esittämä ajallinen kaari on viitteellinen, pikemminkin nyt on pyritty pitkäjänteisempään kehitystyöhön ja versiotoimitusten harventamiseen. Tarpeet luokitellaan jaottelulla virhe-, ongelma-, muutos- tai lisätyö. Prosessivastaavat voivat täsmentää ja suodattaa ehdotettuja asioita ennen kuin niitä ryhmässä käsitellään. Käsittelyn jälkeen työlistalle hyväksytyt asiat aika ajoin paketoidaan selkeiksi versioiksi joiden toteutus, yksikkötestaus, järjestelmätestaus ja hyväksymistestaus on selkeästi vaiheistettu. Tällä hetkellä on riittävät laite-, tietokanta- ja sovellusresurssit eri vaiheiden kattavaan läpivientiin. Harvennetuilla versiotoimituksilla saadaan paremmin varmistettua myös asiakkaan mahdollisuudet tehdä kattavampi hyväksymistestaus. Versiot myös dokumentoidaan ja loppukäyttäjille tehdään erilliset versiosaatteet uusista, korjatuista ja muuttuneista piirteistä järjestelmässä.



Kuva 4: kehitysprosessi

Kehitysryhmätyöskentely asiakkaan kanssa

Kehitysryhmätyöskentely asiakkaan kanssa on muotoutunut pitkän yhteistyön ja kokemuksen tuloksena varsin rakentavaksi ja saumattomaksi. Kun suuren järjestelmän alkuaikojen lastentaudeista on päästy, on kehittäminen sujunutkin todellisen kumppanuuden hengessä. Kaikki osapuolet ovat sitoutuneet yhteisiin tavoitteisiin ja tehtyihin yhteisiin ratkaisuihin. Asiakkaan eri prosesseissa toimii omat projektiryhmänsä, tällä hetkellä voisi ehkä sanoa virtuaaliset projektiryhmänsä, koska eri prosesseissa on tällä hetkellä vähän toisistaan poikkeava tilanne. Osa prosesseista on varsin stabiilissa tilassa eikä niitä tukeva järjestelmäosuuskaan vaadi juuri panostuksia. Tarvittaessa asiakkaalta löytyy prosessit tunteva joukko joka saadaan koolle viemään asioiden kehitystä eteenpäin, samoin meillä toimittajana on hallussa tekniikan osaaminen ja asiakkaan toiminnat tuntevia asiantuntijoita.

Toimittajan käyttämät tukiprosessit

ICL Inviassa on osana laatujärjestelmää kuvattu myös palveluprosessi. Prosessin tehtävänä on huolehtia palvelujen tuottamisesta asiakastyytyväisyydestä huolehtien, tehokkasti, tuloksekkaasti, virheettömästi ja palvelusopimusten mukaisesti ja henkilöstön tyytyväisyydestä huolehtien. Yhteistä palveluprosessia noudattamalla saavutetaan em. tavoitteisiin vaikuttavia etuja:

- yhteiset työmenetelmät
- yhteinen toiminnan kehittämistapa
- päällekkäiset ja tarpeettomat työvaiheet vähentyvät
- asiakaspalvelu nopeutuu ja virheiden määrä vähenee
- asiakkaille välittyy ammattimainen ja laadukas vaikutelma palvelusta
- yksiköiden ja ryhmien välinen yhteistoiminta tehostuu
- henkilöiden perehdyttäminen ja tehtäväkierto tehostuu
- vastuut, roolit ja toimintatavat selviä
- loma- ja sijaisjärjestelyt helpottuvat
- palvelun ohjaus, ennakoitavuus ja vertailtavuus paranee
- tavoiteasetanta selkiytyy

Osana palveluprosessiamme on myös järjestelmällinen palveluosaami-

sen kehittäminen. Lähes kaikki palveluprojekteissa mukana olevat ovat käyneet kaksipäiväiset palvelutreenit. Talotasolla on säännöllisesti palveluun liittyvää jatkokoulutusta palvelutyypeittäin, asiakkuuksittain sekä vielä erikseen yleisempiä palveluprosessin koulutustilaisuuksia että aihekohtaisia infotilaisuuksia. MOT-palveluprojektissa asiakkaan kaikki yhteydenotot on ohjattu MOT-tukipuhelimeen tai tuen käyttämään sähköpostiin. Yhteydenotot siis kirjataan talotason SC-järjestelmään, joka toimii sekä seuranta- että palautejärjestelmänä. Lisäksi sitä käytetään palveluista raportointiin ja mittaamiseen. ServiceCenteristä on mahdollista lähettää suoraan sähköpostina ratkaisut asiakkaalle, jonka nimissä asia on järjestelmään viety. Käytännössä toimitamme vastaukset tuen sähköpostia käyttäen, tällä pyritään varmistamaan, että ratkaisu tarvittaessa voidaan ohjata kokonaisille käyttäjäryhmille, joita varten käytössämme on valmiit jakelulistat. Samaa järjestelmää käyttää ICL Invian laajalle asiakaskunnalle tarkoitettu CallCenter. Sovelluspuolen versionhallinnassa on käytössä MS Visual Studion VisualSourceSafe versionhallintatyökalu. Osa teknisestä dokumentaatiostakin pystytään ylläpitämään tämän avulla.

Testauksessa olemme projektin aikana ottaneet käyttöön Mercuryn TestDirectorin. TD:n avulla suunnitellaan ja viedään läpi tällä hetkellä järjestelmätestit ja testauksen aikainen yhteydenpito testaajan ja toteuttajan välillä. TestDirectorin käyttöä on tarkoitus jatkossa edelleen laajentaa sekä testien automatisointiin että asiakkaan käyttöön hyväksymistestauksessa. Hallinnointia ja dokumenttien seurantaa tukee myös yhdessä asiakkaan kanssa käytettävä ICL Invian Kimppa-kumppaniverkko. Kimppa on ICL Invian NetCommunityyn pohjautuva yhteinen verkko jonka kautta mm. kaikki järjestelmien hallinnointiin liittyvät asiakirjat ovat asianosaisten saatavilla selainkäyttöliittymällä.

Kokemukset, asiakastyytyväisyys, henkilöstötyytyväisyys

Järjestelmän valmistuttua oli paljon ongelmia jotka on saatu ratkaistua. Tällä hetkellä asiakas on tyytyväinen ja nimenomaan systemaattinen työ ylläpitomenettelyjen kehittämisessä on koettu yhteiseksi ponnistukseksi. Mitä jämäkämmin on pysytty versioinnissa sen helpompaa on ollut versionhallinta ja kanssakäyminen oman projektihenkilöstön kesken. Asiakaskin on ymmärtänyt versioinnin tärkeyden ja versioiden välisten korjausten ja toimitusten minimoinnin. Asia on juuri nyt ollut keskeisesti esillä kun varaudutaan euron tuloon ja tehdään järjestelmän euroversiota. Asiakastyytyväisyyttä mitataan jatkuvasti satunnaisotannoilla ServiceCenteriin kirjatuista yhteydenotoista. Järjestelmäkohtaisesti valitaan 10-15 asiakasta joka kuukausi puhelinhaastatteluun. Viime aikoina saamamme arvosanat ovat olleet kiitettäviä tai lähellä kiitettävää. Henkilöstötyytyväisyys onkin mutkikkaampi asia. ICL Invia palkitsee toki aika ajoin myös asiakaspalvelussa tai muuten työssään ansioituneet. Palkitsemisesityksiä voi tehdä kuka tahansa eivätkä vain esimiehet. Tällä hetkellä palveluprojektissamme työskentelee useita projektin aikana hyvästä työstä palkittuja. Osaltaan ylläpitotyön mielekkyyttä lisää tässä ympäristössä järjestelmäkokonaisuuden laajuus ja monipuolisuus. Kukin työhön osallistuva voi jossain määrin vaihtaa osaaluetta ajoittain ja samalla kehittää sekä teknistä että toimialaosaamistaan.

Lähteenä on käytetty MOT-järjestelmän dokumentaatiota sekä ICL Invia Oyj:n laatujärjestelmää.

Kirjoittaja Kimmo Vaikkola toimii järjestelmän palvelupäällikkönä ja on ollut mukana järjestelmän määrittely- ja toteutusvaiheesta lähtien ja vastannut jo edeltävän valiolaisten osuuskuntien tilitysjärjestelmän ylläpidosta.

SYTYKE ry:n WWW-sivujen ylläpitosuunnitelma

Helena Venäläinen,
FD Finanssidata Oy

Sivuston hoito alkaa niiden julkistuksesta

Sytyke ry:n www- sivujen uudistuksen yhdeksi tavoitteeksi asetettiin sivuston sisällön tuottamisen hajautus sekä sisällön ajantasaisuus. Yhdistyksellä ei ole palkattuja sivuston hoitajia - ylläpitoon on siis saatava mukaan mahdollisimman moni. Erityisenä haasteena nähtiin sivuston tietojen pysyminen ajantasalla, onhan WEB nykyisin täynnä jo syntyessään vanhentunutta tietoa.

Sivuston rakenne ja tekninen toteutustapa on suunniteltu siten, että myös hajautettu sisällöntuottaminen on mahdollista, vaikkakin materiaalin julkistaminen päätettiin toistaiseksi pitää keskitettynä. Sivuston ylläpidosta vastaavat johtokunnnan voisittain valitsema web-toimikunta, yhdistyksen puheenjohtaja, Systeemityölehden päätoimittaja ja toimitussihteeri sekä kerhojen vetäjät. Jäsenten osallistuminen sivuston kehittämiseen on myös ollut yhtenä tärkeimmistä tavotteista. Linkki- yms. vihjeiden lisäksi toivomme suoraa palautetta sekä kehittämisehdotuksia sivuston osasta vastaaville ja web-toimikunnalle.

Allaoleva suunnitelma kuvaa ylläpitovastuut ja antaa raamit päivitysten hoitamiselle tietojen ajankohtaisuuden varmistamiseksi. Suunnitelma talletetaan tietysti osak-

Päivityskohde	Tarkistys/Päivitystarve	Sisällön vastuuhenkilö
Etusivun kehysosa	Tarkistus vuosittain helmikuussa, muuten palautteen/tarpeen mukaan.	Käytettävyysvastaava
Etusivun ajankohtaista	Vähintään 1 kerta/kk	Käytettävyysvastaava
Etusivu - tapahtumat	Tarpeen mukaan	Webmaster, jolle kerhonvetäjät, lehdentoimittajat, yms. ilmoittavat ilmoitustekstin ja ilmoitusajan.
Sytyke ry: Yhdistyksen tiedot, yhteystiedot	Vuosittainen kunnostus tammikuussa, toimintakalenteri helmikuussa, päivityksiä tarvittaessa	Puheenjohtaja
Sytyke ry: Yhdistyksen historiaosa		Webmaster
Sytyke ry: WEB-ylläpitosuunnitelma	Tammikuussa	Koordinaattori
Lehti – mediakortti toimitusprosessi	Vuosittain helmikuussa /tietojen muuttessa	Toimitussihteeri
Lehti –yleistiedot, ohjeet kirjoittajille, toimintakalenteri, historia, kuvaus lehtien sisällöstä	Tammikuu, maaliskuu, toukokuu, elokuu, marraskuu	Päätoimittaja + ao. lehden toimitusvastaavat
Lehti – artikkelit	Ilmestymisestä vuoden kuluttua	Webmaster
Julkaisut	Uusien ilmestyessä	Webmaster
Kerhojen yleisesittely ja historia	Vuosittain tammi/helmikuussa, uusien kerhojen syntymisen myötä	Webmaster kokoaa tiedot kerhonvetäjiltä
Kerhotoiminta	Kerhon syntyessä esittely (kts. yllä) ja toimintakalenterin ensimmäinen versio, jonka ylläpito tarpeen mukaan syntyneistä tuloksia.	Kerhonvetäjät; Rela-kerho Testauskerho Datanomit Sujuva systeemityö
Tietolinkit	Tammikuu, elokuu + Tarpeen mukaan jäseniltä saatujen vihjeiden perusteella	WEBteknikko + jäsenet
Tietopankki	Tarpeen mukaan jäseniltä saatujen vihjeiden perusteella	WEBteknikko + jäsenet
Menetelmämarkkinat	Tarpeen mukaan jäseniltä saatujen vihjeiden perusteella	WEBteknikko + jäsenet
Viihdy!	Tammikuu, elokuu +	Rakennevastaava

si WWW-sivustoa. Sivujen teknistä tuottamista koskevaa sekä muuta tekniikkaan liittyvää apua saa Webmasterilta sekä Webteknikolta.

Suunnitelman toimivuus on selkeiden vastuiden, säännöllisien tarkastusaikataulujen ja palautteen kohdentamisen varassa. Vapaaehtoistyöhön perustuvassa yhdistyksessämme joudumme kaikki jatkuvasti prioritoimaan ajankäyttöämme ja nähtäväksi jää, miten hyvin allaoleva suunnitelma käytännössä toimii ja miten dynaamiset sivut Sytyke ry saa.

Sivuston osat, päivitystarve ja vastuuhenkilö

Allaolevassa taulukossa on kuvattu sivujen/sivuston osien päivitystarve sekä sisällön ajantasaisuudesta ja luettavuudesta vastaavat. Periaatteena on, että kyseisten sivujen yhteydessä mahdollisesti pyydettävät palautepostit ohjataan sisällöstä vastaavalle, jolloin hän voi tarvittaessa reagoida saamaansa palautteeseen.

Taulukossa esiintyvät WEB-vastuulliset ovat vuonna 2001:

- Koordinaattori
  Helena Venäläinen
- Käytettävyysvastaava
  Heini Holopainen
- Rakennevastaava
  Minna Oksanen
- Webmaster
  Lauri Laitinen
- Webtekniikka
  Erkki Pöyhönen

Helena Venäläinen,
Yksikönpäällikkö,
FD Finanssidata Oy

Muistathan käydä tutustumassa
Systeemityöyhdistys SYTYKE ry:n kotisivuihin:

www.pcuf.fi/sytyke

Sovelluksen kunnossapidon oikea hinta?

Pekka Forselius,
STTF Oy ja
Katrina D. Maxwell,

Ohjelmiston elinkaaren aikana kulutetaan huomattavasti enemmän rahaa ja työtunteja sovellusten ylläpitämiseen kuin ensimmäisen tuotantoon otettavan version kehittämiseen. Tästä ylläpitotyöstä merkittävä osa on yleensä jatkokehittämistä ja uusien piirteiden lisäämistä, mutta yllättävän paljon kuluu myös pelkkään ohjelmiston kunnossapitoon. Suurimmissa suomalaisissa organisaatioissa jopa satoja henkilötyövuosia joka vuosi. Mittavasta työmäärästä huolimatta toistaiseksi on tutkittu hyvin vähän niitä tekijöitä, jotka vaikuttavat kunnossapitotyön tuottavuuteen ja kustannuksiin. Uuskehitysprojektien tehokkuudesta ja siihen vaikuttavista tekijöistä on julkaistu huomattavasti enemmän tutkimustuloksia. Tässä artikkelissa esitellään erään pankin sovellustietokannan tilastollisen analysoinnin yhteydessä löytyneitä tutkimustuloksia ja tuottavuuteen vaikuttavista tekijöistä tehtyjä havaintoja. Artikkeli pohjautuu 95-prosenttisesti samojen kirjoittajien keväällä 2001 julkaistuun kirjoitukseen [9].

1. Taustaa

Jollei kunnossapitotyötä mitata ja valvota millään tavalla, sen määrä saattaa paisua huomaamatta tarpeettoman korkeaksi. Alun perin taitamattomasti ylisuureksi mitoitettu työmäärä tahtoo säilyä vuodesta toiseen samalla tasolla, oli siihen sitten todellista syytä tai ei. On olemassa myös vaara että tarkemmin valvotuissa kehittämisprojekteissa tehtyjä suunnitelmien ylityksiä piilotetaan kontrolloimattoman kunnossapidon piikkiin, jotta projektit saadaan "näyttämään hyvältä". Kunnossapitotyön arvioiminen ja oikean tason päätteleminen ovat useimmille organisaatioille yhä edelleen liian vaikeita tehtäviä – riippumatta siitä ovatko ne kunnossapidon toimittajia vai tilaajia.

Muun muassa edellisessä kappaleessa mainituista syistä johtuen eräs Suomen suurimmista pankeista alkoi kerätä toteutuneita työmäärätietoja paitsi projekteista, myös eri tyyppisistä ylläpitotöistä ja kaikesta muustakin henkilöstön ajankäytöstä. Tämä järjestelmällinen tiedonkeruu käynnistettiin jo vuonna 1985, mutta sitä kehitettiin ja täydennettiin askel askeleelta kyseisen vuosikymmenen loppuun mennessä. Jokainen tietojärjestelmätyön läheisyydessä leipäänsä ansainnut työntekijä, mukaan lukien johtajat ja sihteerit, raportoi viikoittain kaiken tekemänsä työn kohdistettuna projekteille, sovellusten ylläpitoon tai hallinnollisiin linjatehtäviin. Ylläpitotyön osalta pyrittiin erottelemaan kunnossapitotyö, pienet projektoimattomat laajennukset ja mahdollinen tuotannon hoito toisistaan.

Pankin tietämys kehittämisprojekteista oli jo valmiiksi erittäin hyvä, kiitos mittavan järjestelmien kokonaisuudistusohjelman, johon kuuluvat projektit olivat systemaattisen salkunhallinnan piirissä alusta lähtien. Vuosien 1987 ja 1995 välisenä aikana pankissa kehitettiin ja otettiin tuotantoon 250 uutta sovellusta. Samaan aikaan ylläpidettiin edellisen järjestelmäsukupolven tietojärjestelmiä vanhassa keskuskoneympäristössä. Uusi järjestelmä perustui pääosin henkilökohtaisilla työasemilla, paikallispalvelimilla ja keskuskoneella toimiviin 3-kerrosarkkitehtuurin mukaisiin ratkaisuihin. Vanhan ja uuden järjestelmän rinnakkain elo aiheutti omat lisähaasteensa sekä kehittämiselle että kunnossapidolle. Kahden erillisen keskuskoneympäristön ylläpito tulee ymmärrettävästi kalliiksi, joten sitä ei haluttu jatkaa yhtään tarpeettoman pitkään. Ylläpitotyön kokonaismäärä ja osuus kuitenkin kasvoivat vääjäämättä näiden vuosien aikana, joten siihen herättiin kiinnittämään myös entistä enemmän huomiota. Lisämausteensa ja ylimääräistä painetta henkilöstökustannusten suuntaan toivat lisäksi 1990-luvun alkupuoliskon lama ja pankkikriisi. Myös ohjelmistojen kunnossapitokustannukset oli siis syytä saattaa mahdollisimman alhaiselle tasolle, vaarantamatta kuitenkaan millään tavalla itse pankkipalveluiden tuotantoa.

Saadakseen todellista tietoa sekä sovellusten ylläpitäjien että niiden omistajien vuotuista budjetointia ja toimintasuunnittelua varten pankin kehityspäälliköt täydensivät sovellustietokantaa huomattavasti 1990-luvun alussa. Siihen koottiin sovelluksittain suunnitellut ja toteutuneet ylläpitotyömäärät, sovelluksen koko toimintopisteinä, ikä, tuotantoonottoajankohta, tuotannossa havaittujen virheiden lukumäärä, help-desk kyselyjen määrä, runsaasti erilaisia luokittelutekijöitä ja vastuuhenkilöiden arviot kymmenen valitun riskitekijän suhteen. Myös sovellusten volyymi- ja käyttökustannustietoja koottiin samaan

kantaan. Kun tietoja koottiin useamman vuoden ajalta, oli parhaista yksittäisistä sovelluksista talletettuina 90-luvun puolivälissä yli 120 yksittäistä tietoa. Niiden tarkastelu ja käyttäminen johtoryhmätyöskentelyn sekä vuosisuunnittelun tukena oli säännöllistä, mutta syvällistä tilastollista tutkimusta tietokannan tiedoille ei silloin tehty. Uskottiin ja tiedettiin, että tärkeimmät päivittäisjohtamisessa huomioitavat seikat ja sovellusten poikkeavuudet tulivat muutenkin tarpeeksi selvästi näkyviin.

2. Tutkimuskysymykset

Kannattaako omia tietokantoja yleensä tutkiakaan tilastollisesti? Mitä tämä pankki olisi voinut oppia sovellustietokantansa tutkimuksesta? Kun analysoitava tietokanta on tarpeeksi suuri, voidaan tilastotieteen keinoin osoittaa esimerkiksi se, millä tutkituista tekijöistä on todellista vaikutusta kunnossapitotyön kustannuksiin. Toinen vastaava esimerkki on sovelluksen iän vaikutus kunnossapitokustannuksiin. Mielenkiintoinen tieto, joka ei selviä välttämättä pinnallisen tarkastelun avulla lainkaan. Samoin se, onko eri kehittämisvälineillä ja käytetyillä ohjelmointikielillä selvää vaikutusta tuotannossa esiintyviin virhemääriin tai kunnossapidon kustannuksiin. Ovatko vaikkapa Telonilla kehitetyt sovellukset ylläpidon kannalta tehokkaampia kuin perinteisellä Cobolilla väännetyt? Mitä tuotantovirheiden määrä vaikuttaa ylläpidon kustannuksiin? Muun muassa näihin kysymyksiin pyritään vastaamaan tässä artikkelissa. Johtopäätökset voi tiivistää seuraavaan perustavanlaatuiseen kysymykseen: Olisiko pankin kannattanut muuttaa ylläpito- tai kehittämiskäytäntöjään, jos sillä olisi ollut tämä tilastolliseen tutkimukseen perustuva uusi tieto käytössään siihen aikaan? Olisiko joitakin asioita kannattanut tehdä toisella tavalla?

Vaikka tässä tutkimuksessa käsitelläänkin vain yhden yksittäisen pankin sovellusten ylläpitotietoja, pitäisi tulosten herättää laajempaakin kiinnostusta. Moni suomalainen IT-organisaatio on ammattimaisuudessaan ja järjestelmällisyydessään vasta nousemassa sille tasolle, jolla parhaat pankit olivat 10 vuotta sitten. Vaikka Cobol on jo vuosia sitten tuomittu kuolemaan, tosiasia on se että ylivoimainen enemmistö tänä päivänä kunnossapidon piirissä olevista ohjelmistoista on koodattu sillä, tai sen kanssa saman tasoisilla muilla kielillä [1]. Cobol-sovellukset ja keskuskoneympäristöt dominoivat niin vahvasti että jopa 83 % maailman tapahtumankäsittelyvolyymista hoidetaan niiden avulla [2]. Yksittäisten uusien, seksikkäämpien kehittämisvälineiden osuus varsinkin kriittisten ja suurivolyymisten järjestelmien ylläpidossa on edelleen häviävän pieni, mutta kun se jossain vaiheessa lähtee kasvuun, kannattaa vanhemmilla kielillä hankittua

Muuttujan nimi	Arvoväli (tai lkm)	Määritelmä
		Luokittelumuuttujat
BORG	5	Pankkitoimintatyyppi (yksityispankki, yritys-, suuryritys-...)
MORG	17	Muutoskokous (maksuliikenne, ottolainaus, lainat, …)
APPTYPE	4	Sovellustyyppi (info-palvelu, operatiivinen taustatoiminto, operatiivinen asiakaspalvelu, liittymäpalvelu)
DBMS	2	Tietokannanhallintajärjestelmä (DB2, IDSN)
TPMS	5	Tapahtumankäsittelyjärjestelmä (IMS, erä,…)
CRITIC	5	Kriittisyysluokka, siedettävän tuotantokatkoksen lyhyys
T	4	Sovelluksen toipumisvalmius
		Kvantitatiiviset tiedot
OBLEFF	19 - 3031	Vuotuinen kunnossapitotyön määrä tunteina
TOTALFP	18 - 2328	Sovelluksen koko toimintopisteinä
PCOBOL	0.1 – 1.0	Cobol-koodin osuus
PTELON	0 - .87	Telon-koodin osuus
PEASY	0 - .52	Easytrieve+ koodin osuus
PJCL	0 - .96	JCL-koodin osuus
AGEE93	2 - 85	Ylläpitokuukausien yhteismäärä, sovelluksen ikä (kk)
AVETRAN	0.1 - 345	Tuhansia tapahtumia vuorokaudessa, tapahtumamäärä
DISKSP	0.1 - 39012	Levytila (MB)
CPU	0.1 - 2197	Vuorokautinen CPU-käyttö (s)
DDY	0 - 163	Tuotannossa löytyneiden sovellusvirheiden lkm/vuosi
		Riskitekijät
R1	1-5	Käyttäjien määrä, palvelun laajuus
R2	1-5	Tuotantoonoton joustavuus
R3	1-5	Muutoskokousmenettelyn joustavuus
R4	1-5	Ylläpidettävyys
R5	1-5	Dokumenttien kunto
R6	1-5	Henkilöriippuvuus
R7	1-5	Ajoituskriittisyys
R8	1-5	Online-käsittelyn integraatioaste
R9	1-5	Eräkäsittelyn integraatioaste
R10	1-5	Kapasiteetin kasvunvara

kokemusta ja tietämystä edes yrittää hyödyntää.

3. Tutkimuksessa käytetty aineisto ja keskeinen käsitteistö

250 sovellusta yhdessä liiketoimintaympäristössä on paljon. Valitettavasti ne kaikki eivät olleet yhtä kurinalaisen valvonnan ja suunnittelun piirissä. Myös työaikojen raportointikäytännöissä oli osastokohtaisia eroja. Kaikkien sovellusten osalta kunnossapidon osuutta ei pystynyt erottamaan jatkokehittämisestä ja kaikkien sovellusten vastuuhenkilöt eivät vastanneet riskitekijäkyselyihin. Myös monien muiden muuttujien keräystarkkuudessa oli vaihtelua. Nyt tekemässämme tilastollisessa analyysissa päätimme keskittyä vain kaikkein täydellisimmät tiedot omaaviin sovelluksiin.

Kaikkiaan 250 keskuskonesovelluksen joukosta valikoitui 67 "täydellistä" sovellusta, joista oli tallessa sekä eritellyt työmäärätiedot vuodelta 1993, mitattu koko ja virhemäärät vastaavalta ajalta. Riskitekijät löysimme 56:lle näistä sovelluksista. Kaikki tutkimukseen mukaan tulleet sovellukset olivat pankin itse kehittämiä ja lähes kaikki käyttivät DB2-tietokantaa. Viittä lukuun ottamatta kaikissa joukon sovelluksissa oli käytetty ainakin osittain Cobolia, useimmissa jonkin verran JCL:ää. Muut käytössä olleet kehitysvälineet olivat Easytrieve+ sekä Telon, joka on Cobolia tuottava koodigeneraattori. Taulukossa 1 (edellisellä sivulla) ovat lueteltuina ne 28 muuttujaa, joita tutkimme tässä tilastollisessa analyysissa. Jaoimme ne kolmeen ryhmään: luokittelumuuttujiin, kvantitatiivisiin tietoihin ja riskitekijöihin.

Jokaiselle ylläpidon riskitekijälle oli laadittu 5-portainen sanallinen asteikko, jonka avulla sovellusten vastuuhenkilöt arvioivat oman sovelluksensa haavoittuvuutta. Kuvassa 1 (yllä) on esimerkki riskitekijän määrityksestä. Kaikkien riskitekijöiden määritykset ovat kirjoittajalla ja ne on julkaistu englanninkielisinä keväällä 2001 [10]. Myös pankissa tuohon aikaan käytetty sovelluksen kokonaisriskiasteen laskentakaava esitellään samassa artikkelissa.

5.Dokumenttien kunto

Millainen on sovelluksesi dokumenttien kunto asteikolla 1-5, erittäin hyvästä erittäin huonoon?

- 1 kun sovelluskuvaus mukaan lukien tietohakemistokuvaukset ovat ajantasalla ja koodi hyvin kommentoitua
- 2 kun sovelluskuvaus on keskeisiltä osiltaan kunnossa, tietohakemistokuvaukset ajantasalla ja ohjelmakoodi hyvin kommentoitua
- 3 kun sovelluskuvaus sisältää kaikki osat ja on pääosiltaan kunnossa, mutta osittain vanhentunutta
- 4 kun dokumenteista puuttuu merkittäviä osia kokonaan mutta koodi on selkeää
- 5 kun kirjallisia dokumentteja ei ole ja ohjelmakoodinkin ymmärtäminen tuottaa vaikeuksia

Taulukossa 2 (seuraavalla sivulla) vertaillaan tutkimukseen valikoituneen osajoukon ja alkuperäisen tietokannan keskimääräisiä tunnuslukuja. Niistä näkyy, että osajoukko edustaa melko hyvin koko sovelluskantaa. Vain tuotannossa esiin tulleiden sovellusvirheiden tiheydessä on merkittävä ero, mutta se johtunee siitä että monet eniten käytetyistä ja suurista sovelluksista olivat näiden täydellisimmin mitattujen ja valvottujen joukossa (kuten tietysti pitääkin). Luvut kertovat myös pankin sovellussalkun laajuudesta.

Sovellusten koko mitattiin lähdeohjelmakirjastoista laskemalla kunkin sovelluksen lähdeohjelmista suoritettavien lauseiden määrä. Laskentaa varten toteutettiin omat pienet laskentaohjelmat kunkin eri kielen kirjastoympäristöön. Käytössä olleita kieliä olivat Cobol, Telon, Easytrieve+ ja JCL, jotka kukin laskettiin erikseen. Sovelluksen koko toimintopisteinä saatiin käyttämällä takaisinlaskentamenetelmää ja Capers Jonesin julkaisemia muuntotaulukoita [3]. Täten muutetut, eri kielillä tehdyt osuudet sovelluksista voitiin sitten laskea yhteen, mitä raakojen lausemäärien kohdalta ei ole mielekästä tehdä. Mittaustulosten järkevyys varmistettiin muutamalla pistokokeella käsin laskettuja toimintopistemääriä vasten. Niiden perusteella voitiin uskoa, että sovellusten koot olivat riittävän tarkasti laskettuja ja etenkin niiden keskinäiset kokosuhteet oikein. Koon laskenta lähdeohjelmakirjastoista suoritettiin kesällä, kun kehittämistoiminta muutoin oli hieman normaalia vähäisemmällä tasolla.

Kunnossapidolla tarkoitetaan tässä yhteydessä kaikkia sovelluksen toimintakuntoisena säilyttämiseksi tarvittavia pakollisia sovelluksen kehittämistoimenpiteitä. Sellaisia ovat etenkin havaittujen virheiden korjaamiset, mutta myös pienehköt, lähinnä teknisen ympäristön kehittymisestä johtuvat muutokset. Esimerkiksi varusohjelmistoista, lähdeohjelmakirjastoista, tietokannanhallinta-, tapah-

	Valittu 67:n sovelluksen osajoukko	Koko 250 sovelluksen sovellustietokanta
Keskikoko (fp)	471	425
Yhteiskoko (fp)	31531	106,200
Keskimääräinen kunnossapitotyömäärä (h)	521	495
Yhteensä kunnossapitotyömäärä (h)	34907	123750
Kunnossapitotyön keski-tuottoaste (h/fp vuodessa)	1,11	1,16
Keskimääräinen vuotuinen sovellusvirheiden määrä	9.4	6.9
Sovellusvirheiden kokonaismäärä vuodessa	632	1725

tumankäsittely- tai käyttöjärjestelmistä saattaa tulla uusia versioita, joiden mahdollistamia uusia piirteitä kannattaa ottaa käyttöön, mutta jotka edellyttävät sovelluksen muuttamista ja testaamista. Nämä kaikki työt siis sisältyivät pankissa sovellusten kunnossapitoon. Sitä vastoin kaikki puhtaasti liiketoiminnallisista syistä tehdyt muutokset ja sovellusten toiminnalliset laajennukset suunniteltiin ja valvottiin joko erillisissä projekteissa tai säännöllisesti kokoontuvien muutoskokousten alaisuudessa. Myöskään sovellusten tuotannonhoitoa, kuten parametrien ylläpitoa ja käsittelyn ohjausta ei sisältynyt kunnossapitoon. Sovellusvastuuhenkilöt raportoivat kaiken tekemänsä kunnossapitotyön viikoittain työajanseurantajärjestelmään, joka kokosi tiedot vuosittaisen suunnittelun ja seurannan tarpeisiin.

Virhetilastointi tehtiin pankissa hyvin huolellisesti. Pankkien sovellusvirheet saattavat pahimmillaan aiheuttaa todella suuren sotkun monen ihmisen elämään, mutta usein melko mitättömiltäkin tuntuneet tuotantohäiriöt päätyivät sanomalehtien sivuilla pohdittaviksi. Siksi kaikki tuotannossa havaitut virheet rekisteröitiin ja luokiteltiin tätä tarkoitusta varten kehitettyyn järjestelmään. Virheet jakautuivat 9 eri tyyppiin ja viiteen vakavuusluokkaan. Virhetiedon keruu oli mahdollisimman automatisoitua, mutta sovellusvastuuhenkilöiden piti aina "sulkea" virheet, kun niiden syyt oli selvitetty ja asianmukaiset korjaus- tai estotoimenpiteet tehty. Automaattisesti valvomo-ohjelmistosta kirjautuneiden ilmoitusten lisäksi myös loppukäyttäjien edustajat ja help-desk toimihenkilöt tekivät kirjauksia. Tuotannonhoitajat huolehtivat automaattisten kirjausten alustavasta luokittelemisesta. Virhetyyppi eli syykoodi perustui virheen lähteeseen, jotka olivat "sovellussuunnittelu", "parametristo", "varusohjelmisto", "käyttöohjeistus", "operointi", "asiakasliittymä", "sovellusliittymä", "tietoliikenne" ja "laitteisto". Käytetyt viisi vakavuusluokkaa vaihtelivat vähäpätöisestä "kauneusvirheestä" erittäin vakavaan, ei-toivottua julkisuutta ja runsaita korjaustöitä aiheuttavaan. Tässä tutkimuksessa esitetään ainoastaan tyyppiin "sovellussuunnittelu" kuuluvia lukuja, käyttämättä vakavuusluokittelua. Noin 8 % kaikista tuotantohäiriöistä oli kirjattu tähän luokka18] an.

Sen, millaista sovellusten laatua edellisessä kappaleessa kerrotut luvut kuvastavat, saa jokainen lukija päättää itse. Erittäin suuri osa kirjatuista virheistä oli luonnollisestikin vähäpätöisiä, ja käyttäjätyytyväisyysmittauksissa saatujen vastausten perusteella yksittäisten sovellusten laatu oli vähintäänkin tyydyttävä, usein hyvä tai erittäin hyvä. Peräti 21 sovelluksista toimi tarkasteluvuotena täysin virheettömästi, 10 selvisi yhdellä ja 6 kahdella kirjauksella, mitä voidaan pitää päivittäisessä tuotantokäytössä olevalle sovellukselle erittäin hyvänä saavutuksena, muutamaa poikkeusta lukuun ottamatta millä tahansa toimialalla. Joitakin suorastaan ongelmatapauksina käsiteltäviä sovelluksiakin joukkoon mahtui. Seitsemällä sovelluksella oli yli 20 havaittua suunnitteluvirhettä, eli noin joka toinen viikko aiheutui uusi niistä johtunut tuotantohäiriö. Virhetilastointia ja tilaston aika ajoin tutkimista voidaan joka tapauksessa pitää suositeltavana, joskin yksittäisten sovellusten keskinäisen hyvyyden vertailu suhteuttamatta kokoon, ikään yms. pitäisi jättää tekemättä.

4. Tutkimustulokset

Sovelluskanta analysoitiin vaiheittaisella muuttujamenetelmällä, samalla tavalla kuin Kitchenham [4] tutkiessaan kehittämisprojektien tehokkuuteen vaikuttavia tekijöitä. Tämän tutkimuksen tavoitteena oli löytää sellainen muuttujayhdistelmä, joka kaikkein parhaiten selittää vuotuisen kunnossapitotyön määrää. Tätä tarkoitusta varten tutkittiin koko valikoitunutta 67 sovelluksen joukkoa. Tässä yhteydessä tehtyjen alustavien havaintojen perusteella tutkittiin lisäksi hieman syvemmin Telon-kehitysvälineellä tehtyjen 11 sovelluksen joukkoa, tavoitteena selvittää tämän välineen käyttöön siirtymisen merkitystä verrattuna aiemmin käytössä olleeseen "raakaan" Coboliin.

4.1 Vuotuisen kunnossapitotyömäärän ennustaminen

Joka syksy kokoontuivat sovellukset omistavien pankin liiketoimintayksiköiden edustajat ja sovellusten ylläpidosta vastaavien osastojen johto "budjettineuvotteluihin", sopimaan seuraavalle vuodelle varattavista voimavaroista. Silloin käytiin läpi kunkin sovellusalueen tärkeimmät näköpiirissä olevat projektit ja arvioitiin eri tyyppisten ylläpitotöiden määrää. Myös sovellusten käyttökustannukset

ja eri tyyppisiä valmisohjelmisto- ja laitehankintoja oli esillä, mutta henkilötyö oli kuitenkin kaikkein tärkein näissä neuvotteluissa. Ylläpitotyön osuus tästä oli ja on huomionarvoinen, sillä laajalti [5,6,7] uskotaan, että koko ohjelmiston elinkaaren aikaisista kustannuksista ylläpidon osuus on 50-80 %. On luultavaa, että erittäin pitkäikäisiin sovelluksiin turvautuvilla toimialoilla, kuten pankki- ja vakuutustoiminnassa, ylläpitotyön osuus on vielä normaaliakin suurempi. Muutosten ja korjaavan kunnossapidon osuudet kannattaa arvioida ja mitata erikseen, sillä etenkin jälkimmäinen suhteutettuna vaikka virhemääriin ja sovellusten kokoon antaa huomattavasti paremman vertailukelpoisuuden eri sovelluksille kuin epämääräisemmistä tekijöistä johtuva laajentava ylläpito. Pääosin erillisiä, mielenkiintoisia kysymyksiä ovat "*uuden sovelluksen ensimmäisen käyttövuoden kunnossapitotyömäärä*" ja toisaalta "*vanhan sovelluksen seuraavan käyttövuoden kunnossapitotyömäärä*".

4.1.1 Ennustemalli

Tästä tutkitusta aineistosta käytetyllä tutkimusmenetelmällä parhaaksi osoittautunut kunnossapitotyömäärän ennustemalli esitetään taulukossa 3 (alla). Siihen valitut viisi muuttujaa selittävät 61.5 % työmäärän vaihtelusta. Yksittäiset muuttujat ovat taulukossa tärkeysjärjestyksessä. Kaikkein tärkeimmäksi tekijäksi yksinään osoittautui sovelluksen koko, joka selittää 27 % työmäärän vaihtelusta. Kuten odottaa sopii, vuotuisen kunnossapitotyömäärän havaittiin kasvavan sovelluksen koon kasvaessa. Toiseksi tärkeimmäksi tekijäksi nousi eräkäsittelyn integraatioaste. Mitä tiukemmin sovellus oli sidottu eräkäsittelyympäristöönsä, sitä isompi oli vuotuinen kunnossapitotarve. Mielenkiintoista sinänsä, että online-käsittelyn integrointi ei vaikuttanut läheskään yhtä vahvasti.

Sovellusalue, eli muutoskokous (MORG) osoittautui kolmanneksi tärkeimmäksi muuttujaksi. Kukin sovellusalue edustaa pankkitoiminnan sisällä omaa liiketoimintasektoria; joiden erilaisesta perinteestä ja toimintakulttuurista johtuen työskentelyn tehokkuus ja suoraviivaisuus näyttävät vaihtelevan merkittävästi. Tämä noudattelee aikaisemmissa tutkimuksissammekin havaittuja linjoja liiketoiminta-alueen merkityksestä sovelluskehityksen tuottavuuteen [8]. Tehokkaimman ja tehottomimman sovellusalueen keskimääräisessä tuottoasteessa oli peräti nelinkertainen ero. Se kuulostaa todella suurelta, mutta sitä voidaan kyllä perustella monella tavalla. Sovellusrakenteet, käyttöliittymäratkaisut, liiketoiminnan kehittymisvauhti ja niin edelleen selittävät hyvin näitä eroja. (Osittain tämän tutkimuksen ja osittain pitkän kehittämisprojektien tuottavuustekijäseurantansa pohjalta FiSMAn ylläpitotyöryhmä kehitti talven 2000-2001 aikana kunnossapitotyön olosuhteiden arviointiin tarkoitetun, 22-tekijäisen tilanneanalyysimenetelmän.)

Neljänneksi tärkeimmäksi kunnossapitotyön työmäärää selittäväksi tekijäksi kohosi – ehkä hieman yllättäen – muutostenhallinnan joustavuus. Mitä byrokraattisempi ja tiukemmin valvottu muutoskontrolli ilman "pikakäsittelyjä" sovelluksella oli, sitä pienemmällä kunnossapitotyöllä selvittiin. Mitä helpompaa ja joustavampaa muutosten tekeminen oli, sitä enemmän korjaavaa ylläpitoa aiheutui. Tämä kuulostaa tietysti tarkemmin mietittynä ihan järkevältä, mutta oli todella hauskaa että se kohosi näin selvästi esiin myös tilastollisissa tutkimuksissa. Yleisesti ottaen sovelluksen moniomistajuus ja runsaat liittymät muihin sovelluksiin vaativat tavallista tarkempaa kontrollia, kun taas yhden omistajan erilliseen sovellukseen suhtaudutaan helposti paljon leväperäisemmin. Ensin mainittuihin seuloutuvat tehtäviksi vain todella tärkeät muutokset, kun taas jälkimmäisiin voidaan lisätä ominaisuuksia pelkän päähänpiston perusteella.

Viidenneksi, viimeiseksi malliimme mukaan kelpuutetuksi tekijäksi nousi sovelluksen ikä. Se, että ikä vaikutti negatiivisesti kunnossapitotyömäärään, oli iloinen yllätys. Mitä vanhempi sovellus, sitä vähemmän kunnossapitotyötä. Se kertoo sovellusten hyvästä laadusta ja ylläpitotyön hyvästä hallinnasta. Joissakin aiemmin julkaistuissa tutkimuksissa on käsitelty ohjelmistojen "ylläpidettävyyttä" ja väitetty sen heikkenevän ikääntymisen myötä. Sen pitäisi sitten näkyä myös kunnossapitotyön kasvamisena. Näin varmasti käykin useimmille sovelluksille jossakin elinkaaren vaiheessa, mutta ainakin nämä tässä tutkimuksessa mukana olleet sovellukset olivat vielä elinkaarensa alkuvaiheessa ja paranivat vuosi vuodelta. Olivathan vanhimmatkin aineistossa olleista sovelluksista vasta noin 7 vuotta tuotan-

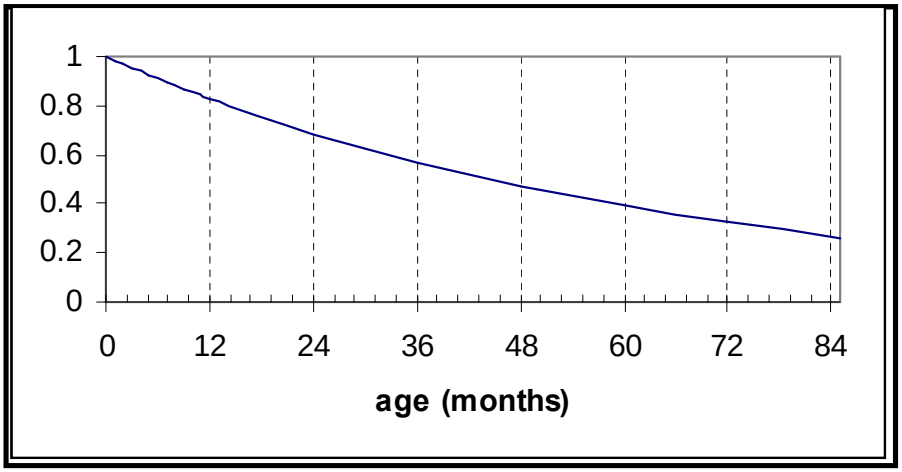
TUTKITTU MUUTTUJA	VAIKUTTAVA MUUTTUJA	VAIKUTUKSEN SUUNTA	VAIHTELUN KOKONAIS-SELITYSASTE
Kunnossapitotyön määrä	Sovelluksen koko toimintopisteinä	positiivinen	61.5%
	Eräkäsittelyn integraatioaste	positiivinen	
	Muutoskokous, sovellusalue	tapauskohtainen	
	Muutoskokousmenettelyn joustavuus	negatiivinen	
	Ylläpitokuukausien määrä	negatiivinen	

nossa olleita. Tarkastelimme siis hieman tarkemmin näitä elinkaarensa alkuvaiheessa olevia sovelluksia, joille löysimme kenties hyödyllisen kaavan. Jos vakioimme kaikki muut muuttujat (niitä edustaa vakio k oheisessa kaavassa), havaintojemme mukaan työmäärä noudattaa seuraavaa iästä riippuvaista exponenttifunktiota:

Siten ohjelmiston "ikäkerroin" on 1 sovelluksen tullessa tuotantoon ja lähtee sitten laskemaan melko nopeasti. Oli mielenkiintoista havaita, että iällä on eksponentiaalinen vaikutus työmäärään, ainakin tässä sovellusjoukossa. Eksponentiaalisia funktioitahan käytetään paljon liikkeenjohdossa, taloustieteessä ja luonnontieteissä kuvaamaan erilaisten ilmiöiden kehittymistä ja rappeutumista. Samalla tavalla kuin luonnontieteessä puhutaan aineiden hajoamis- tai hajaantumisasteista, voisimme me puhua kunnossapitotyön vähenemisasteesta. Aineiston mukainen vähenemisasteen kuvaaja esitetään kuvassa 2 (oikealla).

Aivan oman tarkastelunsa ansaitsee sovelluksen ensimmäinen elinvuosi tuotannossa. Havaitsimme huomattavan nopeata työmäärän vähenemistä ensimmäisten kuukausien aikana, mutta valitettavasti aineisto ei tukenut näin tarkkaa analyysia riittävästi. On kuitenkin selvää, että kuvan 2 mukainen 20 % aleneminen painottuu vuoden alkukuukausille ainakin päivittäin käytettävien sovellusten kohdalla.

Tässä pankissa vuotuisen kunnossapitotyön määrän havaittiin vähenevän sovelluksittain keskimäärin 17.2 % vuodessa. Niinpä 100 tuntia tänä vuonna kunnossapitoa vaatinut sovellus selviää ensi vuonna 82.8 tunnilla ja seuraavana 68.6 tunnilla – ellei sovellukselle tapahdu mitään muuta merkittävää. Käytännössä niitä usein laajennetaan, mutta koon muuttuminenkin on melko helppoa



Kuva 2: Sovelluksen kunnossapitotyön vähenemisaste iän funktiona

mitata tarkasti, ettei tarvitse tyytyä työmäärien arvaamiseen.

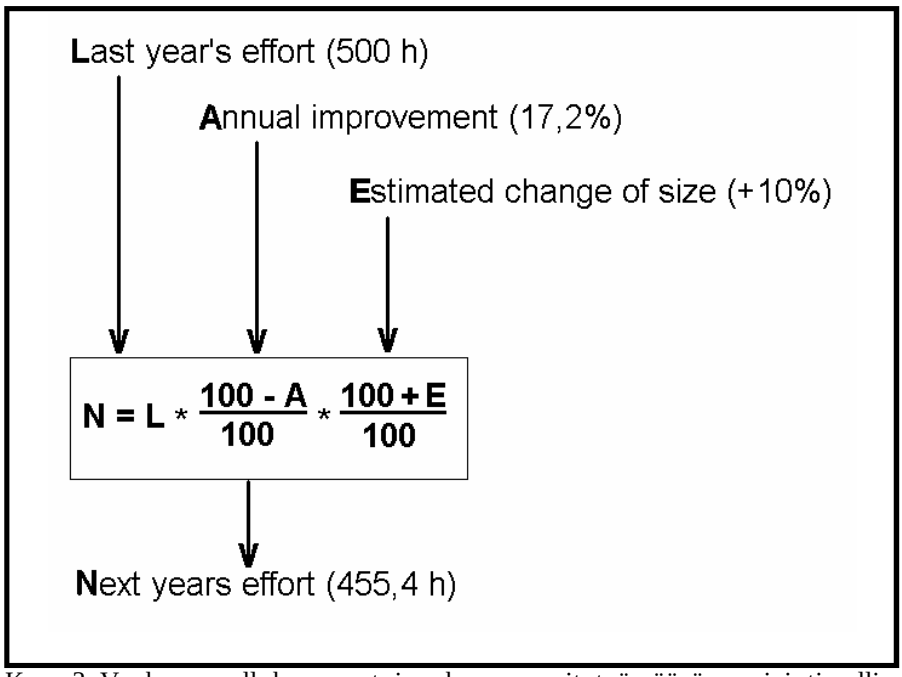
Minkään muun muuttujan lisääminen malliimme ei olisi parantanut sen selittävyyttä, joten tyydyimme yksinkertaisuuden vuoksi edellä kuvattuihin viiteen. Mielenkiintoinen ja eräitä aiemmin "laadun kustannuksista" julkaistuja väittämiä kyseenalaistava havainto oli, ettei tuotannossa havaittujen suunnitteluvirheiden lukumäärä vaikuttanut kovinkaan selvästi kunnossapitotyön määrään. Se selitti yksinään vain 3.5 % työmäärästä. Sen sijaan se korreloi kyllä vahvasti sovelluksen koon kanssa, joka oli 27 % selitysasteellaan kaikkein tärkein muuttuja. Mitä isompi sovellus, sitä enemmän suunnitteluvirheitä. Siksi ainakaan puhdas sovellusvirhemäärä ei olisi tuonut malliimme mitään lisää. Emme tutkineet tarkemmin, olisiko paremmin sovelluksen laatua kuvaava kokoon suhteutettu virhemäärä muuttanut asetelmaa.

4.1.2 Mallin käyttökelpoisuus ennustamiseen

Vaikka yllä kuvattu 5-muuttujainen malli onkin kovin hyödyllinen kasvattaessamme ymmärrystämme kunnossapitotyön kustannustekijöistä, emme voi suositella sen käyttämistä sellaisenaan työmäärien ennustamiseen, ainakaan yhtään laajemmassa määrin. Koska aineistossamme oli kerättyinä toteutuneita työmääriä useilta peräkkäisiltä vuosilta, kokeilimme mallia 26 sovelluksen osalta. Vertailun vuoksi kokeilimme perusyksinkertaista käytännön mallia, eli yritimme "ennustaa" toteutunutta työmäärää suoraan edellisen vuoden toteuma-arvolla. Jälkimmäisellä tavalla tulos oli parempi, joskaan ei läheskään täydellinen silläkään. Tarkat tulokset tästä "menetelmävertailusta" on julkaistu toisaalla [10].

Kaikkein suurin arviointiongelma on epäilemättä uusien sovellusten kanssa, ensimmäisen vuoden työmäärää arvioitaessa. Kokemuksemme perusteella sen kunnossapitotyömäärä on elinkaaren alkupuolella kaikkein suurin ja arviointi vaikeinta, koska edellisen vuoden toteutumatietoja ei ole lainkaan käytettävissä. Tähän tapaukseen kehittämämme malli tuo jonkun verran apua, auttamalla valitsemaan tärkeimpien tekijöiden osalta samanlaisia, jo tuotannossa olleita sovelluksia ja suhteuttamaan työmääräarvion niiden kokemusten, iän yms. perusteella.

Kuvassa 3 (yllä) on yksinkertainen arviointimallivanhojen sovellusten seuraavan vuoden työmäärän arvioimiseksi. Edellisen vuoden toteutunut työmäärä, yrityskohtainen kunnossapitotyön vähenemisaste ja arvio so-



Kuva 3: Vanhan sovelluksen vuotuisen kunnossapitotyömäärän arviointimalli

velluksen koon muutoksista ovat melko hyvin riittävät lähtötiedot.

4.2 SSSSS SSSSSSSSS kehittämisvälineiden valinta

Joskus, tai itse asiassa aika useinkin tutkimustyötä tehtäessä käy niin, että tulokset rupeavat ohjaamaan työtä uusille urille. Sillä tavalla löydetään ihan uusia asioita, jotka saattavat joskus osoittautua jopa alkuperäistä tutkimusaihetta tärkeämmiksi. En nyt ota kantaa siihen, kävikö tässä tutkimuksessa niin. Jonkun mielestä varmaan kyllä, ja useimpien muiden kannalta ei. Me havaitsimme joka tapauksessa työskentelyn alkuvaiheissa, että eri kehittämisvälineillä ja niiden osuudella kehittämistyössä oli vaikutuksia kunnossapitotyön määrään ja työn tuottavuuteen. Tämä vaati tarkempaa tutkimista mielestämme.

"Aikanaan Telonin myyntimies väitti, että hänen edustamansa väline olisi tehokkaampi sekä itse kehittämisvaiheessa että varsinkin aikanaan ylläpidossa, kuin aiemmin käytössä ollut Cobol."

Yllä olevan kaltaisia muistoja voi olla monellakin välinevalintoihin osallistuneella, mutta hyvin usein tuollaiset väitteet ovat jääneet todentamatta. Ne unohtuvat siinä kuin vaalilupauksetkin. Niihin kannattaisi kuitenkin suhtautua vakavasti, varsinkin jos suunnittelee 250 uuden pitkäikäiseksi aiotun sovelluksen kehittämistä seuraavien 10 vuoden aikana. Vähempikin varmasti riittäisi välinevalintojen strategisen tarkastelun tarpeen huomaamiseen. Voimakkaasti integroiduissa ympäristöissä välineiden vaihtaminen on aina ollut todella hankalaa ja kallista, eikä tähän päivään mennessä ole ilmestynyt niin "avoimia" ja "helppokäyttöisiä" välineitä, että paranemista olisi pystytty todella osoittamaan. Siksi strategiset välinevalinnat pitäisi pystyä sekä tekemään että pyrkiä todentamaan oikeiksi mahdollisimman aikaisin. Myös vaikutus ylläpitotyöhön kannattaisi selvittää.

Pankissa tärkeimpänä syynä Telon-ohjelmageneraattorin valinnalle mainittiin odotukset sen tehokkuudesta, vaikutukset laatuun ja sen myötä vaikutus myös ylläpitotyön tehostumiseen. Lisäeduksi mainittiin myös mahdollisuus myöhemmin siirtyä tekemään kehitystyötä hajautetuissa ympäristöissä. Viimeksi mainitusta hyödystä aineistossamme ei ole mitään näyttöä, mutta vaikutuksesta ylläpitoon on. Oliko päätös hankkia Telon järkevä?

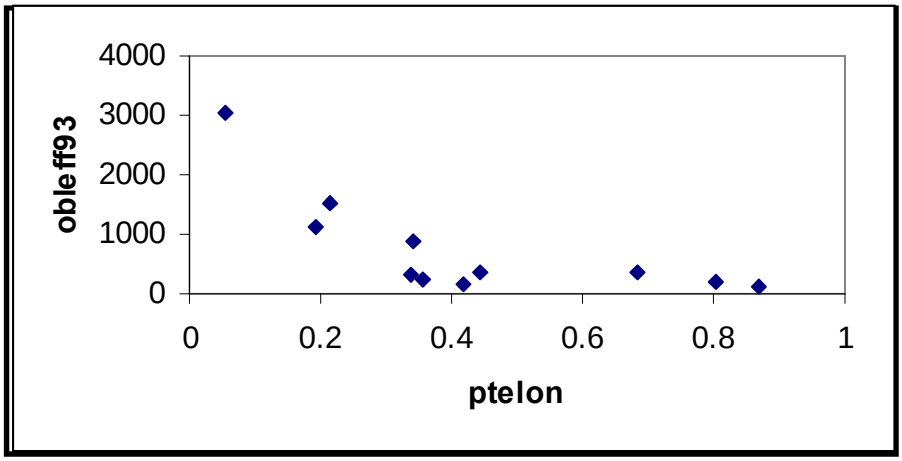
11 sovelluskannan sovelluksista oli kehitetty pääasiallisesti Telonilla. Havaitsimme, että kunnossapidon tuottavuus Telon-sovelluksissa oli keskimäärin yli 2 kertaa parempi kuin Cobol-sovelluksissa. Pienestä havaintojoukosta johtuen tämä Telon-syrjähyppymme ei oikein täytä tilastollisen analyysin vaatimuksia, mutta siitä välittämättä päätimme tonkia asiaa vähän tarkemmin. Tutkimme siis tätä 11 sovelluksen otosta, jolle kehitimme kolme omaa mallia.

Telon-mallimme ovat taulukossa 4 (seuraavalla sivulla). Havaitsimme erittäin vahvan negatiivisen korrelaation Telon-koodin suhteellisen osuuden ja kunnossapitotyömäärän välillä. Mitä enemmän sovelluksesta oli kehitetty Telonilla, sitä vähemmän työtunteja käytettiin. Myös suhteutettaessa kunnossapitomäärä sovelluksen kokoon, ilmiö oli aivan selvä. Telonin prosenttiosuuden kasvaessa 31:stä 87:ään kunnossapidon tehokkuus jopa nelinkertaistui. Kuvassa 4 (yllä) esitetään kunnossapitotyömäärän ja Telonin osuuden suhde.

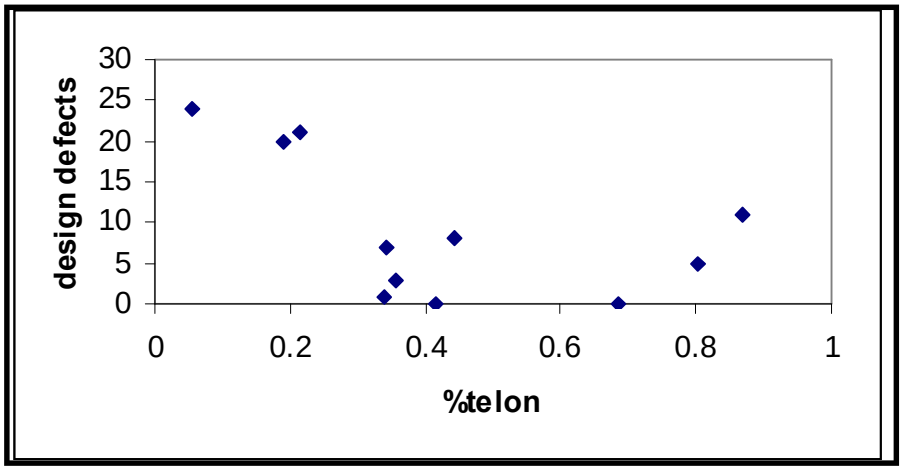
Olimme myös kiinnostuneita Telonin vaikutuksesta sovellusten laatuun. Löysimme edellistä heikomman, mutta kuitenkin merkittävän riippuvuuden Telon-käyttöasteen ja virhemäärän välillä. Lineaarinen malli väittää, että mitä enemmän koodista on tuotettu Telonilla, sitä vähemmän suunnitteluvirheitä tuotannossa esiintyy. Kuitenkin tarkasteltaessa kuvaa 5 (vasemmalla alla) voidaan havaita, että kuvaaja ei ehkä olekaan lineaarinen vaan eräänlainen u-käyrä. Sen mukaan näyttää siltä, että laadun kannalta optimaalinen Telonin osuus on 50 ja 70 % välissä.

TUTKITTU MUUTTUJA	VAIKUTTAVA MUUTTUJA	VAIKUTUKSEN SUUNTA	VAIHTELUN KOKONAIS-SELITYSASTE
Kunnossapito-työmäärä	Telon-koodin osuus prosentteina	negatiivinen	72%
Kunnossapito-työmäärä	Telon-koodin osuus prosentteina	negatiivinen	64%
Suunnittelu-virheiden määrä	Telon-koodin osuus prosentteina *	negatiivinen	22%

Taulukko 4: Telon-sovellusten tutkimustulokset ja mallit



Kuva 4: Kunnossapitotyömäärä suhteessa Telonin käyttöön



Kuva 5: Suunnitteluvirheiden määrä suhteessa Telonin käyttöön

5. Johtopäätöksiä ja oivalluksia

Useat ohjelmistojen jatkokehittämisen ja kunnossapidon kanssa tekemisissä olevat ammattilaisetkin kokevat kunnossapitotyön arvioinnin hankalaksi, eikä heitä helpottamaan ole julkaistu kovin paljon empiirisiä kvantitatiivisia tutkimuksiakaan [9]. Tällä yhden pankin aineiston tilastollisella analyysilla olemme toivottavasti kyenneet osoittamaan, millaisilla tekijöillä siinä ympäristössä oli todellista vaikutusta kunnossapitotyön määrään ja kunnossapidon hintaan. Vaikka onkin kiinnostavaa ymmärtää eri tekijöiden tärkeys, vielä arvokkaampaa on jos kunnossapidosta vastuussa olevat pystyvät tämän tietämyksen perusteella parantamaan käytäntöjään ja siten vaikuttamaan tulevan kunnossapitotyön tehokkuuteen. Seuraavissa kappaleissa tarkastelemme lyhyesti, mitkä näistä tekijöistä ovat – tai ainakin voisivat olla – vastuuhenkilöiden kontrolloitavissa ja ohjattavissa. Mitä asioita pankissakin olisi voinut tehdä toisin, jos tämä tieto olisi ollut käytettävissä?

Tehokkaiksi osoittautuneiden ohjelmankehitysvälineiden, kuten tässä erikoistapauksessa Telonin käyttämistä voidaan organisaatioissa lisätä vähitellen, uuskehityshankkeiden kautta. Koska Telonin käyttö selvästi tehosti kunnossapitotyötä ja vähensi tuotannossa esiin tulevien sovellus virheiden määrää, sen käyttöön olisi pitänyt ohjata kehittäjiä määrätietoisesti. Myös uuskehittämisessä Telon on osoittautunut noin 30 % tehokkaammaksi kuin Cobol, sekä Suomessa koottujen FiSMAn tietojen että Australiassa kerättyjen ISBSG-projektien perusteella [11]. Telon ei kuitenkaan ole ratkaisu kaikkeen, joten sille huonoimmin soveltuvien toiminnallisten piirteiden toteuttamiseen kannattaa huoletta käyttää parempia välineitä. Kohtuus siis kaikessa, mutta Telonin – tai yleisemminkin parhaiden välineiden – suhteellista osuutta vähitellen korkeammaksi.

Kunnossapitotyön määrä kasvoi eräkäsittelyn integraatioasteen kasvaessa. Tätäkin olisi voitu kontrolloida jonkin verran, joskin vähäisessä määrin. Tapahtumien kerääminen välivarastoihin ja niiden purkaminen sieltä tekevät prosessin huomattavasti monimutkaisemmaksi ja haavoittuvammaksi kuin hoidettaessa yhteydet online-sanomien avulla. Merkittävä ero kunnossapitotyön määrään tulee myös

häiriöiden ajoituksesta. Eräyhteyksien hoitaminen kun ajoittuu usein sellaiseen aikaan, etteivät sovellusvastuuhenkilöt ole valmiiksi työpaikalla. Aina todellista valinnan varaa ei tietenkään edes ole, mutta silloin kun online-vaihtoehto on tarjolla, siihen kannattaisi tarttua. Niin ainakin tämä tutkimus osoitti, mutta on muistettava että pankissa oli runsaasti kokemusta molemmilla tavoilla integroimisesta. Jos oma tekninen osaaminen loppuu eräyhteyksien tasolla, lienee paras tyytyä niihin.

Muutostenhallinnan joustavuuden käänteinen vaikutus kunnossapitotyön määrään oli lievä yllätys, mutta osoittautui hieman tarkemman tarkastelun kautta ihan uskottavasti. Syy kunnossapitotyömäärän kasvuun ei ollutkaan joustavassa ja nopeassa muuttamisessa vaan hallitsemattomassa päätösprosessissa. Tekevälle sattuu, niin sanoakseni. Sovellusten riippuvuuteen organisaatioista ja muista sovelluksista ei ole tehtävissä oikeastaan yhtään mitään, mutta pienimpien ja erillisimpien sovellusten omistajien ja ylläpitäjien kannattaisi kiinnittää ylimääräistä huomiota prosessiensa muodollisuuksiin. Vaikkapa lisäämällä jokaiseen muutokseen hyväksyminen allekirjoituksin, prosessi saattaisi pysäyttää siksi tärkeäksi hetkeksi jonka unohtunut kultainen ajatus vaatii. Yksi itse aineistosta johtunut ongelma oli tämän kyseisen riskitekijän moniselitteisyys. Suosittelemmekin sen jakamista kolmeen osaan: muutosten keskinäiset riippuvuudet, sidosryhmien määrä ja päätöksenteon määrämuotoisuus.

Liiketoimintaluokituksiin ja vastaaviin ei tietenkään voi vaikuttaa, mutta on tärkeätä ymmärtää tuottavuuserojen olemassa olo ja yrittää selvittää oman sovellusympäristön perustuottoaste. Tämä tuskin haittaa sovelluksen omistajan ja ylläpitäjän välisissä neuvotteluissa, ellei oma taso ole poikkeuksellisen kurja.

Me havaitsimme myös sellaisen ilmiön että kunnossapitotyön tuottoaste (h/fp) aleni sovelluksen koon kasvaessa. Tämä on erittäin mielenkiintoista, sillä usein ollaan pelätty isojen sovellusten aiheuttavan tehottomuutta kunnossapitoon. Ainakaan tämän pankin sovellusten kokoluokassa näin ei ollut, vaan siellä saavutettiin kiistatta mittakaavaetuja. Niitä olemme löytäneet jo suomalaisesta sovelluskehittämisestä myös uuskehityspuolelta [8], joten liialliseen sovellusten rajaamiseen ei ehkä kannata mennä edes komponenttihuumassa.

Koska sovelluksen ikä osoittautui tärkeäksi tekijäksi ja kunnossapitotyön määrä nopeasti alenevaksi heti hyvin tehdyn sovelluksen elinkaaren alussa, suosittelemme tehostettua työmäärien seurantaa ensimmäisille kuukausille. Seuraavan kuukauden todellisen tarpeen voi aina arvioida edellisten 1-3 kuukauden toteumien perusteella. Täten kunnossapitoon ei varata turhaan resursseja, mutta heitä ei myöskään vapauteta liian aikaisin muihin töihin. Ensimmäiset tuotantokuukaudet ovat myös laadullisesti erittäin tärkeitä. Jos niiden aikana yritetään selvitä alimiehityksellä ja korjailla eteen tulleet häiriöt vain jotenkuten, voidaan sovellukselle aiheuttaa elinikäisiä haittoja jotka näkyvät sitten kokonaiskustannuksissa ja ylläpitohenkilöstön turhautumisena.

Viimeinen mielenkiintoinen havainto oli, ettei dokumenttien kunto vaikuttanut merkittävästi kunnossapidon määrään. Ehkä aineistossamme olleiden sovellusten suhteellinen nuoruus selittää tämän. Niiden kuvaukset eivät olleet vielä ehtineet rapistua. Tai sitten pankissa työkäytännöt olivat tältä osin riittävän hyviä. Osajoukossa ei ollut yhtään täysin dokumentoimatonta sovellusta. Yksi mahdollinen selitys voi olla myös 1990-luvun alkupuoliskon laman ja pankkikriisin vaikutuksesta lähes olemattomaksi mennyt henkilöstön vapaaehtoinen vaihtuvuus. Jokaisen sovelluksen alkuperäiset kehittäjät olivat yhä käytettävissä. Missään tapauksessa kyseisen riskitekijän pysymisestä piilossa ei saa päätellä, ettei dokumentteja kannata kirjoittaa.

Kaiken kaikkiaan meille jäi tästä tutkimuksesta päällimmäiseksi tunnelma, että olemme ainakin itse oppineet paljon uutta ylläpitotyöstä. Kunnossapito, tai mieluummin nykyisin ”jatkuva palvelu” voi olla iloinen, haastava ja erittäin mielenkiintoinen asia, kun sen oikein ottaa.

Kirjoittajista:
FM, MBA Pekka Forselius *toimii johtajana ja projektijohdon kouluttajana sekä konsulttina STTF Oy:ssa. Hän on aktiivisena jäsenenä mm. FiSMAn (Finnish Software Metrics Association) ylläpitotyöryhmässä ja Sytyke ry:n johtokunnassa. Hän jatko-opiskelee Jyväskylässä ja tekee tutkimusyhteistyötä mm. INSEADin kanssa. Hänen yhteystietonsa löytyvät tarvittaessa Sytyke ry:n sivuilta. Sähköposti kulkee osoitteeseen pekkaf@sttf.fi.*

PhD Katrina D. Maxwell on omistamansa Ranskassa, Fontainebleaussa toimivan Datamax yhtiön toimitusjohtaja, kahden viimeisen vuoden kansainvälisen ESCOM-konferenssin ohjelmatoimikunnan puheenjohtaja, jonka erikoisalaa ovat sovelluskehitystyöhön liittyvien aineistojen tilastolliset analyysit. Hän on myös suomalaisen Experience Pro –kokemustietokannan pääanalysoija.

Kirjoittajat ovat tehneet toistensa kanssa säännöllistä tutkimusyhteistyötä vuodesta 1996 lähtien.

6. Lähdeluettelo

[1] Carr, D. and R.J. Kizior, "The Case for Continued Cobol Education", IEEE Software, Volume 17, Number 2, March/April 2000, pp. 33-36.

[2] Arranga, E., "In Cobol's Defense", IEEE Software, Volume 17, Number 2, March/April 2000, pp. 70-75.

[3] Jones, C., "A new look at languages", Computerworld, November 7, 1988, pp. 97-103.

[4] Kitchenham, B., A Procedure for Analyzing Unbalanced Datasets, IEEE Transactions on Software Engineering, vol.24, no.4, April 1998, pp.278-301.

[5] Ramaswamy, R., "How to Staff Business-Critical Maintenance Projects", IEEE Software, May/June 2000, pp.90-94.

[6] Kemerer, C.F. and S. Slaughter, "Methodologies for Performing Empirical Studies: Report from the International Workshop on Empirical Studies of Software Maintenance", Empirical Software Engineering, Vol.2, 1997, pp.109-118.

[7] Kemerer, C.F. and S. Slaughter, "An Empirical Approach to Studying Software Evolution", IEEE Transactions on Software Engineering, Vol. 25, No. 4, July/August 1999, pp. 493-509.

[8] Maxwell, K. and P. Forselius, "Benchmarking Software Development Productivity", IEEE Software, Volume 17, Number 1, January/February 2000, pp. 80-88.

[9] Kemerer, C.F. and S. Slaughter, "Determinants of Software Maintenance Profiles: An Empirical Investigation", Software Maintenance Research and Practice, Vol. 9, 1997, pp.235-251.

[10] Maxwell, K. and Forselius, P. "Cost Drivers of Annual Maintenance: One Commercial Bank's Experience", Proceedings of the 12th European Software Control and Metrics Conference, April 2001, pp. 233-242.

[11] ISBSG, The Benchmark Release 6, International Software Benchmarking Standards Group Limited, Australia, p.55.

■ TIETOTEKNIIKAN LIITTO ry

Yhteystiedot: PL 325, 00181 HELSINKI
puh. (09) 4765 8530, fax (09) 4765 8595
sähköposti: jasiat@ttlry.fi, kotisivu: http://www.ttlry.fi

Henkilöjäsen

☐ **LIITTYMINEN 2001**

☐ **MUUTOS** (laita vanhat tiedot tarvittaessa sulkeisiin)

☐ **EROAMINEN**

Nimi		**Jäsennumero**	
Lähiosoite			
Postinumero ja postitoimipaikka			
Puhelin/työ		**Puhelin/koti**	
Matkapuhelin		**Syntymäaika (pp.kk.vvvv)**	
Sähköposti			
Ammattinimike		Tieto tulee esille vuosikirjaan.	
Työnantajan nimi		Tieto tulee esille vuosikirjaan, mikäli työnantaja on yhteisöjäsenemme.	
	☐ **Työnantaja maksaa jäsenmaksun *)**	*) Jäsenmaksu voidaan veloittaa työnantajalta, jos tämä on liiton yhteisöjäsen. Työnantajan hyväksyntä merkitään tälle lomakkeelle. Kysy tarvittaessa ohjeita puh. (09) 4765 8530.	
	Työnantajan allekirjoitus/hyväksyntä		Yhteisöjäsenen jäsennumero
Opiskelijajäsenyys	☐ **Liityn opiskelijajäseneksi** Toimitan todistuksen tämän lomakkeen liitteenä	Päätoimiset päiväopiskelijat saavat jäsenyytensä edullisemmin. Liitteeksi tulee laittaa todistus, jossa on arvioitu valmistumisaika.	
Perhejäsenyys	☐ **Liityn perhejäseneksi** Pääjäsenen nimi ja jäsennumero	Perhejäseneksi voi liittyä, jos samassa taloudessa asuva henkilö on jo Tietotekniikan liitto ry:n henkilö-/opiskelijajäsen. Perhejäsenyyteen ei sisälly lehti-/vuosikirjaetuja.	

Jäsenetulehtipaketti

Valitse vain yksi lehtipaketti (x)

- ☐ ITviikko
- ☐ ITviikko + Tietokone
- ☐ ITviikko + Prosessori
- ☐ ITviikko + Macmaailma
- ☐ ITviikko + Hifi
- ☐ ITviikko + KotiPC
- ☐ ITviikko + Pelit
- ☐ ITviikko + Tiede

Jäsenetulehdet alkavat ilmestyä noin 2 - 3 viikon kuluessa liittymisestä. Kesken vuotta liittyville toimitamme loppuvuoden lehdet.

Tietojen luovutus

Tietojani

- ☐ saa luovuttaa markkinointiin
- ☐ ei saa luovuttaa markkinointiin
- ☐ ei saa laittaa esille vuosikirjaa

Jäsenyhdistyksen valinta

Jäsenmaksuun sisältyy yhden yhdistyksen jäsenmaksu. Merkitse valintasi sarakkeeseen A.

Maksamalla lisämaksun voit kuulua useampaan yhdistykseen. Merkitse haluamasi lisäyhdistykset sarakkeeseen B.

A	Opiskelijayhdistykset	B
☐	Asteriski ry - Turun yliopisto	☐
☐	Blanko ry - Oulun yliopisto	☐
☐	OtaDATA ry - TKK/Espoo *) *) Läheta OtaDatan jäsenhakemus suoraan heille: OtaData ry, PL 52, 02151 Espoo	☐

A	Teemayhdistykset	B
☐	PC-Käyttäjät ry	☐
☐	Tietojenkäsittelytieteen Seura ry	☐
☐	Tietoturva ry	☐
☒	Systeemityöyhdistys ry – SYTYKE ry **Huom!** Datanomit ja Tradenomit on Sytykkeen kerho vuodesta 2001 lähtien	☐
☐	IT-kouluttajat ry	☐

A	Alueyhdistykset	B
☐	Etelä-Pohjanmaan TKY ry	☐
☐	Etelä-Saimaan TKY ry	☐
☐	Helsingin TKY ry	☐
☐	Imatran TKY ry	☐
☐	Kanta-Hämeen TTY ry	☐
☐	Keski-Pohjanmaan TTY ry	☐
☐	Keski-Suomen TKY ry	☐
☐	Kymen TKY ry	☐
☐	Lahden TKY ry	☐
☐	Lapin TTY ry	☐
☐	Mikkelin TTY ry	☐
☐	Pirkanmaan TKY ry	☐
☐	Pohjois-Karjalan TKY ry	☐
☐	Pohjois-Pohjanmaan TKY ry	☐
☐	Satakunnan TKY ry	☐
☐	Savon TKY ry	☐
☐	Varsinais-Suomen TKY ry	☐

Jäsenmaksut vuonna 2001

Jäsenmaksut vuonna 2001 laskutetaan kuukausiperusteisesti. Kun liityt maaliskuussa, veloitamme jäsenmaksua 10:ltä kuukaudelta.

Lisätietoja:
puh. (09) 4765 8530
sähköposti: jasiat@ttlry.fi

Henkilöjäsen ITviikko + 1 ammatti/harrastelehti	**360 mk**
Vain ITviikko	**240 mk**
Opiskelija ITviikko + 1 ammatti/harrastelehti	**240 mk**
Vain ITviikko	**150 mk**
Perhe-/Lisäjäsenyys	**80 mk**

Jäsenmaksun laskutamme postittamalla tilisiirtolomakkeen.

Päiväys ja jäseneksi liittyvän allekirjoitus