SYSTEEMITYÖ

Systeemityöyhdistys SYTYKE ry:n jäsenlehti N:o 3/2006

Projektien tuottavuus ja laatu

Syyskokous ja Jäsenilta 29.11.06

Varaa jo kalenteriisi 29.11. klo 15:00
Sytykkeen jäsentilaisuudelle ja syyskokoukselle.

Tilaisuus pidetään Trainers' House:lla vuoden teemaan
'Tehokkuus ja tuottavuus' liittyvästä aiheesta.

Tervetuloa!

Hallitus

Tarkemmin tietoa sähköisessä jäsenkirjeessä ja web-sivulla www.sytyke.org

Kulkeeko posti?

Onko työpaikkasi tai osoitteesi vaihtunut? Päivitäthän yhteystietosi liiton jäsenpalvelujen kautta.
Saatko sähköisen jäsenkirjeen? Jos et, niin todennäköisin syy on, että osoitetiedoissasi ei ole sähköpostiosoitetta, sähköpostiosoitteen tallennusvaiheessa on tapahtunut 'näppihäiriö' tai sinulla on uusi sähköpostiosoite.
Saatko jäsenlehden? Sytyke-lehti ilmestyy neljä kertaa vuodessa. Sisältö on arvokas ja säästämisen arvoinen. Jos nyt luet kaverin lehteä, niin liity Systeemityöyhdistyksen jäseneksi tai tarkista osoitetietosi. Systeemityöyhdistys SYTYKE ry on Tietotekniikan liiton jäsenyhdistys.

Yhteystietojen päivitykset
Yhteystietojen muutokset saa nopeimmin voimaan Tietotekniikan liiton sivuilla www.ttlry.fi oman sähköisen tunnuksen avulla. Etusivulla kuvan alla on Jäsenpalvelut – painike.
Käyttäjätunnus on jäsennumero ja salasana on sama kuin aikaisemmin tt.tori -sivustolla käytetty. Jäsennumerosi voit tarkistaa jäsenmaksulaskusta tai jäsenetulehtien osoitelipukkeesta. Ellet ole aikaisemmin käyttänyt palvelua tai olet unohtanut salasanasi, voit pyytää sen sähköpostilla osoitteesta jasenasiat@ttlry.fi.

Yhteystiedot kannattaa pitää ajan tasalla, jotta tärkeä jäsenposti saavuttaa Sinut.

Syysterveisin,

Lea Virtanen
Sytyke ry:n tiedottaja

Hyvä tietotekniikan liiton jäsen!

TTL Tietotekniikan liitto ry

Näin päivität tietosi

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

Toivomme sinun erityisesti varmistavan, että sähköpostiosoitteesi jäsentiedoissa on oikea.

Henkilökohtaisempaa palvelua - Sinun eduksesi

Tietotekniikan liitto jäsenyhdistyksineen, osaamisyhteisöineen ja kerhoineen haluaa palvella jäseniään henkilökohtaisemmin ja paremmin, tarjota tietoa juuri Sinua kiinnostavista aiheista. Palvelun parantamiseksi uusimme verkkopalvelumme huhtikuussa 2006. Uuden palvelun avauduttua voit auttaa meitä räätälöimään palvelumme juuri Sinulle.

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

Tietotekniikan liitto ry
Lars Sonckin kaari 12
02600 Espoo
www.ttlry.fi
etunimi.sukunimi@ttlry.fi
jasenasiat@ttlry.fi
p. 020 741 9898
f. 020 741 9889

Julkaisija
Systeemityöyhdistys Sytyke ry
Puhelinvastaus- ja sihteeripalvelu VT Oy
Susanna Koskinen
Henrikintie 7 A, 00370 Helsinki
p. 09 5607 5363
f. 09 5607 5365
sytyke@hennax.fi

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

Toimitussihteeri
Susanna Koskinen
sytyke@hennax.fi

Taitto
Speaking Bark Tmi
Katja Tamminen
sb@cabaroo.com

Toimituskunta 3/2006
Markku Niemi ja Seppo Takanen, PrOSY

Lisätietoja lehdestä
www.sytyke.org/st

Tilaukset
Systeemityölehti sisältyy yhdistyksen Tietotekniikan liiton suositusten mukaiseen yhdistyksen jäsenmaksuun.
Vuositilaus 20 €
Irtonumerot 5 €
Hyvissä ajoin ennen painatusta tehty vähintään 50 kpl lisätilaus 2 €/kpl.
Tilaukset yhdistyksen toimistosta.

Kansikuva
Minna Oksanen

Seuraava numero
4/2006 Testauksen arki Suomessa
Toimituskunta: Maaret Pyhäjärvi
Ilmestyy: to 14.12.2006

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

Painos: 2500 kpl
ISSN 1237-0525
13 vuosikerta, nro. 3

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

Arvonlisävero 0%
Vakiopaikan vähintään vuodeksi varanneille 20% alennus.

Teksti: Markku Niemi ja Seppo Takanen (kuvassa vas), PrOSYn aktiiveja

Pääkirjoitus: Projektien tuottavuus ja laatu - vähemmällä enemmän ja parempaa?

Tällä lehdellä PrOSY tuo oman näkemyksensä Systeemityöyhdistyksen vuositeemaan. Voivatko tuottavuus ja laatu käydä käsi kädessä - vai ovatko ne toisilleen vastakkaisia? Viimeistään tätä lehteä työstäessämme vakuutuimme, että molemmat ovat mahdollisia yhtä aikaa. Oikealla tiedolla, toimintatavoilla, menetelmillä ja välineillä voi edesauttaa kumpaakin.

Artikkelien järjestykseksi valitsimme etenemisen pehmeistä, joskus laadullisiksi luonnehdituista näkökulmista tuottavuuden mittaamisen ja seurannan koviin menetelmiin ja välineisiin. Ihmisen näkökulmaa ei ole unohdettu, sillä yksilöiden väliset tuottavuuserot ovat alalla tunnetusti dramaattisia. Entä menetelmät – onko kansikuvammekin symboloima vesiputousmalli jo aikansa elänyt, eikö vanhassa enää vara olekaan parempi?

Tuottavuuden parantumisen argumentit jonkin uuden menetelmän tai välineen puolesta herättävät kysymyksen, millaista tilastollista tai tieteellistä näyttöä on. Ketterät ohjelmistokehitysmenetelmät ovat vakuuttaneet jo monia, mutta missä on tilastollinen näyttö? Sellaisesta yritimme kovasti saada kirjoituksia tähän lehteen, mutta emme onnistuneet. Eikö ketteryyden vaikutuksia ole mitattu, vaikka tämäkin lehti kertoo ohjelmistotuotannon tuottavuuden seurannan menetelmistä ja mittareista? Jäämme mielenkiinnolla odottamaan selvityksiä aiheesta.

Lehtemme loppukevennyksenä saat nauttia Kuutamolla-kolumnistimme kesäisistä projektinäkemyksistä.

Sisällys

Teksti: Janne Korhonen

Kirjoittaja Janne Korhonen toimii Business Development Managerina Fujitsu Services Oy:ssä. Hän työskentelee tällä hetkellä terveydenhuollon sovellusliiketoiminnan parissa.

janne.korhonen @fi.fujitsu.com

Tuottavuuden unohdettu ulottuvuus

Luovaa työtä

Muistan urani alkuajoilta Juhan. Juha ei varsinaisesti ollut Java-ohjelmoija, kuten meidän nelihenkinen tiimimme toimiston samassa siivessä. Jo pitkän uran c-ohjelmoijana tehnyt Juha oli kuitenkin tutustunut Java-kieleen töidensä ohessa. Juhan taidot paljastuivat, kun tiimillämme oli vaikeuksia saada erästä projektia aikataulussa valmiiksi. Juha auttoi tekemällä omien töidensä ohessa kirjaston, jonka avulla saimme projektin valmiiksi muutamassa päivässä. Juhan tuottavuus jätti meidät sanattomaksi. Käytännössä Juha olisi toteuttanut koko projektin samassa ajassa kuin meidän neljän hengen tiimimme. Juhan tuottavuus oli siis ainakin viisinkertainen meihin verrattuna.

Tällaiset yksilöiden väliset tuottavuuserot ovat tuttuja kaikille ohjelmistoja työkseen tehneille. Useiden tutkimusten mukaan saman kokemuksen omaavien ammattilaisten väliset tuottavuuserot ovat tyypillisesti vähintään kymmenkertaisia. Gerald Weinberg nosti tämän ohjelmistotuotannon inhimillisen ulottuvuuden esille jo vuonna 1971 ilmestyneessä kirjassaan The Psychology of Computer Programming. Viime vuosikymmeninä tämä havainto on kuitenkin jäänyt taustalle, ja ohjelmistokehityksen tuottavuutta on pyritty parantamaan lähinnä prosesseja kehittämällä.

Perinteisen prosessiajattelun näkökulmasta ohjelmiston kehittäminen jakaantuu yleensä suunnittelu- ja toteutusvaiheisiin. Tämä kahtiajako juontaa juurensa perinteisestä insinöörityöstä. Perinteisessä teollisuudessa insinöörit tekevät suunnitelmat, joiden mukaan tuotanto toistaa itseään. Ohjelmistotuotanto sen sijaan on alusta loppuun luovaa suunnittelutyötä. Sen päämääränä on monimutkaisten abstraktien käsitemallien ja prosessien kuvaaminen tietokoneelle täsmällisessä muodossa. Täsmällisyys erottaa ohjelmistosuunnittelun monesta muusta luovasta työstä. Tietokone ei ymmärrä kuin täsmällisiä ohjeita. Usein kuulee puhuttavan, ettei ohjelmoijia enää tarvita, kun tietokoneet oppivat ymmärtämään luonnollista kieltä. Tämä päättely on virheellinen. Vaikka tietokone ymmärtäisi luonnollista kieltä, olisi jonkun kuvattava sille täsmällisesti, kuinka toimia missäkin poikkeustilanteessa. Ohjelmistotuotanto ei siis sisällä itseään toistavaa vaihetta, jota voisi optimoida taylorismin oppien mukaan.

Ketterät menetelmät ovat uudelleen nostaneet esiin ohjelmistokehityksen luovana työnä. Ne korostavat yksilöiden merkitystä ohjelmistotyön tuottavuuden osana. Harvoin ohjelmistosuunnittelua kuitenkaan tarkastellaan yksilön tuottavuuden näkökulmasta. Tunnettu menestysresepti kuuluukin: ”Some people deliver software some don’t. Hire people who do.”

Hullu töitä tekee

Veljeni aloittaessa ensimmäisessä kesätyössään tehtaan linjalla hän oli innoissaan tarjotusta urakkapalkasta. Mikäli hän taittelisi tietyn määrän aihioita, palkan päälle tulisi reilu bonus. Ensimmäisen päivän jälkeen veljeni olikin ylittänyt tavoitteet reilusti ja odotti vähintäänkin suurta kiitosta. Kiitosten sijaan hän joutui työkavereidensa puhutteluun. Uuden kesätyöntekijän ei ollut soveliasta olla liian tuottelias. Samalla tahdilla jatkaessaan hän nostaisi kaikilta vaadittavan suorituksen tasoa.

Luovaa työtä tehdessä aika kuluu ongelmien ratkomiseen. Näitä ongelmia on mahdoton ennakoida, ja ulkopuolisen on vaikeaa seurata työn etenemistä. Tekijällä on tällöin paljon valtaa. Tätä valtaa on helppo käyttää. Kun motivaatio alkaa laskea, on helppoa laskea tuottavuutta. Usein tämä tapahtuu tiedostamatta. Useat tutkimukset ovat osoittaneet, että yksilöiden motivaatio on tärkein tuottavuuden mittari luovaa työtä tehdessä. Lyhyellä tähtäimellä motivoituneet ihmiset tekevät työnsä tehokkaammin ja pitkällä tähtäimellä he kehittyvät ammatillisesti nopeammin. Tämän seurauksena yksilöiden väliset tuottavuuserot kasvavat pitkällä tähtäimellä jopa kymmenkertaisiksi.

Steve McConnell esittelee ohjelmistokehittäjien motivaatiosta tehtyä tutkimusta kirjassaan Rapid Development. Tutkimuksen mukaan ohjelmistosuunnittelijoiden kolme tärkeintä motivaation lähdettä ovat asetettujen tavoitteiden saavuttaminen, mahdollisuus inhimilliseen kasvuun ja työ itsessään.

Ohjelmistosuunnittelijat ovat erittäin motivoituneita saavuttamaan asetetut tavoitteet. Parhaiten motivoivat itse asetetut tavoitteet. Tämän vuoksi kannattaa esimerkiksi ottaa suunnittelijat mukaan aikatauluja tekemään. Hyvä tavoite on realistinen ja selkeä, eikä se muutu kesken tavoitejakson. Tavoitteiden tulee olla myös linjassa keskenään.

Tehokkainta on asettaa vain yksi tavoite. Tavoitteet ja tulokset on myös käytävä läpi ennen ja jälkeen tavoitejakson.

Ohjelmistoammattilaisen osaaminen vanhenee nopeasti. Ohjelmistosuunnittelu houkutteleekin ihmisiä, jotka haluavat oppia jatkuvasti uutta. Organisaatio voi hyödyntää tätä motivaation lähdettä yhdistämällä yksilöiden ja yritysten tavoitteet. Motivoivimpia ovat tehtävät, jotka pakottavat työskentelemään osaamisen äärirajoilla. Haasteellisia tehtäviä tarjoava organisaatio myös houkuttelee motivoituneimpia ja parhaita ammattilaisia.

Ohjelmistosuunnittelu itsessään on hyvin palkitsevaa työtä. Esimerkiksi Linus Torvalds on todennut tehneensä Linuxin huvin vuoksi. Ohjelmistosuunnittelu tarjoaa mahdollisuuden toteuttaa vapaasti omaa luovuuttaan. Monimutkaisten ongelmien ratkominen ja mahdollisuus jatkuvaan uuden oppimiseen tekee ohjelmistosuunnittelusta poikkeuksellisen mielenkiintoisen ammatin. Organisaatio voi hyödyntää tämän esimerkiksi vähentämällä pakollisia hallinnollisia tehtäviä, jotta suunnittelija voi keskittyä varsinaiseen työhön.

Richard Hagmanin ja Greg Oldhamin mukaan ihmiset kokevat työnsä motivoivaksi, kun se tuntuu merkitykselliseltä, he tuntevat kantavansa vastuun työnsä tuloksista ja he saavat nähdä työnsä konkreettiset tulokset. Tämän vuoksi on tärkeää, että suunnittelijat toimivat läheisessä yhteistyössä asiakkaiden kanssa. Samalla hiljainen tieto siirtyy asiakkailta suunnittelijoille ja tarvittavan dokumentaation määrä laskee. Kun suunnittelijat ja asiakkaat mieltävät ohjelmistoprojektin yhteiseksi oppimisprosessiksi päästään parhaaseen tulokseen.

Hiekkaa rattaisiin

Suunnittelijan motivaatio on myös helppo tuhota. Fred Herzberg kutsuu motivaatiota laskevia tekijöitä kahden tekijän motivaatioteoriassaan hygieniatekijöiksi. Tällaisia tekijöitä ovat esimerkiksi tehtäviin soveltumaton työympäristö. Mikäli hygieniatekijät ovat kunnossa, ei niitä parantamalla voi enää nostaa motivaatiota, mutta niiden ollessa puutteellisia motivaatio laskee. Ohjelmistosuunnittelijan motivaatio on helppo viedä esimerkiksi riittämättömällä laitteistolla tai minimalistisella perehdytyksellä.

Epäpätevä johtaminen vie ohjelmistosuunnittelijoiden motivaation. Motivaatio katoaa varsinkin, jos suunnittelijoita pyritään ohjaamaan heidän omalla osaamisalueellaan tai heidän työtään ei arvosteta. Suunnittelijat täytyy ottaa mukaan heitä koskevien päätösten tekoon. Luovan työn tekijälle ei voi kertoa, mitä pitäisi tehdä. Heille täytyy kertoa, mikä on tavoite ja tarjota riittävät resurssit. Jos ymmärtää ohjelmistotuotannon luovana oppimisprosessina, sen johtaminen on helpompaa.

Ohjelmistosuunnittelijat liittävät ammatillisen itsetuntonsa työnsä laatuun, ei sen määrään. Usein ohjelmistoprojektien ongelmat syntyvät, kun vaikea asia sivuutetaan toiminnallisessa määrittelyssä ylimalkaisella kuvauksella ja sen monimutkaisuus tulee esille vasta, kun se pitäisi kuvata täsmällisesti ohjelmointikielellä. Yleensä tässä vaiheessa ei voida enää palata määrittelyn alkuun ja korjata ongelmia käsitemallien monimutkaisissa riippuvuussuhteissa. Kun tyydytään ratkaisemaan ongelma vain poikkeustapauksen ratkaisevalla oikopolulla, menetetään samalla suunnittelijoiden sitoutuminen ja motivaatio. Tätä ongelma voidaan pienentää ketterien menetelmien keinoin iteroimalla ja jatkuvalla refaktoroinnilla. Suunnittelijoiden vastuualueet on hyvä jakaa selkeästi niin, että he voivat huolehtia oman kokonaisuutensa laadusta, vaikka jokin muu osuus laajassa ohjelmistossa ei olisikaan täydellinen. Kokonaisuuksien toimivuutta voidaan valvoa esimerkiksi automatisoidulla testaamisella.

Ohjelmistosuunnittelu on luovaa työtä, jonka tuottavuus perustuu suurelta osin suunnittelijoiden motivaatioon. Tämä inhimillinen ulottuvuus on hyvä pitää mielessään ohjelmistotuotantoprosessien tuottavuutta parannettaessa. Uskonkin, että lähivuosina tullaan kaukoulkoistusten vastapainona näkemään motivoituneiden huippusuunnittelijoiden varaan rakennettujen pienten ohjelmistotalojen lopullinen läpilyönti.

Luettavaa kiinnostuneille:

Alistair Cockburn, Agile Software Development, Addison-Wesley

Frederick P. Brooks, The Mythical Man Month

Gerad M. Weinberg, The Psychology of Computer Programming, Van Nostrand Reinhold Company

Herzberg, F. 1987, 'One More Time: How Do You Motivate Employees?' Harvard Business Review September-October 1987

Jack W. Reeves, Code as Design: Three essays

http://www.developerdotstar.com/mag/articles/reeves_design_main.html

Martin Fowler, The New Methodology

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

Pekka Himanen, Hakkerietiikka ja informaatioajan henki, WSOY

Tom DeMarco & Timothy Lister, Peopleware Productive projects and teams, Dorset House Publishing & Co

Teksti: Sebastian Nykopp ja Lasse Koskela
Kuva: Lasse Koskela

Sebastian Nykopp toimii konsulttina Reaktor Innovationsilla. Menetelmä- ja arkkitehtuurikonsultoinnin lisäksi hän osallistuu aktiivisesti käytännön ohjelmistokehitykseen Reaktorin asiakasprojekteissa.

Lasse Koskela toimii Reaktorilla menetelmäkonsulttina. Hän on ajanut ketterien menetelmien käyttöönottoa Suomessa ja ulkomailla usean vuoden ajan, organisoi mm. perustamansa Agile Finland -yhteisön tapahtumia, ja puhuu ahkerasti muissa alan tapahtumissa. Oikean työnsä ohessa Lasse työstää kirjaa testivetoisesta ohjelmistokehityksestä kansainvälisille markkinoille.

Projektien laadun parantaminen ketterillä menetelmillä

Järjestelmäkehitysmenetelmissä perinteinen viisaus sisältää merkittäviä puutteita. Tämä on yksi syy siihen, että liian moni IT-projekti koetaan jossakin määrin epäonnistuneeksi. Kuitenkin perinteinen ”vesiputousmalli” on pitänyt pintansa ja on edelleen eniten käytetty järjestelmäkehitysmenetelmä.
Artikkelissa käydään läpi perinteisessä mallissa havaittuja puutteita ja esitellään vaihtoehtoisena lähestymistapana ns. ketterät menetelmät, joiden kautta projektien tuottavuutta ja laatua voidaan merkittävästi parantaa.

Missä mennään?

IT-alan historia sisältää enemmän kuin riittävästi tarinoita epäonnistuneista projekteista, saavuttamatta jääneistä tavoitteista ja laadultaan heikoista järjestelmätoteutuksista. Mutkikkaat vaatimukset, kiire ja jatkuvasti kypsymätön teknologia tekevät järjestelmien kehittämisestä erittäin vaativaa. Tämä kuuluu asiaan. On kuitenkin käynyt yhä selkeämmin ilmi, että projektityössä sovelletut menetelmät ovat tärkeä osatekijä projektien haasteissa ja laatuongelmissa.

IT-alan alkuajoilta asti käytössä olleet perinteiset projektimallit eivät ole pystyneet tarjoamaan tyydyttävää pohjaa tietojärjestelmien laadukkaalle ja pitkäjänteiselle kehittämiselle kohtuullisilla elinkaarikustannuksilla.

Hyvän kuvan tilanteesta antaa Standish Groupin 13 000 IT-projektia kattava tutkimus vuodelta 2003:

- 34 prosentissa projekteista lopputuote toimitettiin alkuperäisessä aikataulussa, budjetissa ja määritellyssä laajuudessa
- 15 prosenttia projekteista keskeytettiin ennen toimitusta
- 51 prosentissa projekteista toimitettiin keskimäärin vain 52% toiminnallisuudesta, 82% aikataulun ylityksellä

Viimeisen vuosikymmenen aikana on saatu positiivisia kokemuksia ketteristä menetelmistä, joiden avulla pyritään tuottamaan järjestelmiä nopeammin, laadukkaammin ja pienemmillä riskeillä siten, että lopputulos vastaa todellisia ja todennäköisesti jo projektin aikana muuttuvia tarpeita.

Ongelmallinen vesiputousmalli

Maailmalla yleisesti käytössä oleva IT-projektimalli tunnetaan lempinimellä ”vesiputous”. Malli perustuu projektin jakamiseen toisiaan järjestelmällisesti seuraaviin vaiheisiin kattavaan määrittelyyn, suunnitteluun, toteutukseen, testaukseen ja tuotantoon siirtoon. Tieto siirtyy vaiheiden välillä suurina kokonaisuuksina, useimmiten dokumenttien muodossa. Vesiputousmallin perusongelmiin kuuluu se, että kussakin vaiheessa rakennetaan olettamuksia olettamusten päälle siten, että merkittävä osa virheistä ja väärinkäsityksistä huomataan vasta testausvaiheessa. Tällöin projekti on jo viimeistelyvaiheessa ja virheiden korjaaminen on vaikeaa eikä tehtävään ole riittävästi aikaa.

Oleellinen tekijä vesiputousmallin ongelmien taustalla on tarpeiden ja vaatimusten luontainen epävarmuus. Tarpeista ei ole projektin alussa riittävän tarkkaa tietoa. Monia yksityiskohtia selviää vasta kehittämisen aikana ja tahtotila saattaa muuttua, kun järjestelmästä tai sovelluksesta saadaan ensimmäisiä käyttökokemuksia. Myös dokumenttivetoinen kommunikointi on usein tehotonta ja keskeinen väärinkäsitysten lähde. Vaikka aikataulu ja budjetti pitäisivätkin, projektin laatu kärsii. Usein toteutetaan järjestelmä, joka vastaa vanhaa tavoitetilaa sen sijaan, että tukisi liiketoiminnan nykyisiä tarpeita.

Kun projektin aluksi tehdään kattava määrittely, on lopputuloksena usein liian laajaksi määritelty järjestelmä, johon on varmuuden vuoksi pyritty sisältämään kaikki mitä mahdollisesti voidaan tarvita. Tätä kuvastaa Standish Groupin joidenkin vuosien takainen tutkimustulos, jonka mukaan arviolta 45 prosenttia vesiputousmallilla toteute-

tuista järjestelmän ominaisuuksista ei käytetä ja 19 prosenttia ominaisuuksista käytetään harvoin.

Pitkälti yksinkertaisuutensa ja näennäisen loogisuutensa ansiosta vesiputousmalli on kuitenkin onnistunut säilyttämään paikkansa IT-projektimallien de facto -standardina.

Ketterät menetelmät

Ketterät menetelmät (eng. agile methods, agile development, agile project management) on nimike joukolle ohjelmistokehitysmenetelmiä ja -prosesseja, jotka pohjautuvat neljään periaatteeseen. Näiden periaatteiden mukaan ketteriin menetelmiin lasketaan sellainen joka suosii…

…ihmisiä ja näiden vuorovaikutusta ennemmin kuin työkaluja ja prosesseja,

…toimivaa järjestelmää ennemmin kuin yksityiskohtaisia suunnitelmia,

…avointa yhteistyötä ennemmin kuin sopimusneuvotteluita, ja

…muutokseen reagoimista ennemmin kuin aiemmin tehtyjen suunnitelmien orjallista seuraamista.

Tämä ei siis tarkoita etteivät työkalut, prosessit, suunnitelmat ja sopimukset olisi tärkeitä. Ketterien menetelmien periaatteet vain painottavat, että paras lopputulos syntyy, kun keskitytään yhteiseen päämäärään hyödyntäen henkilöstön täyttä potentiaalia ja seurataan edistystä jatkuvasti konkreettisen tuotteen kautta ja muuttamalla suuntaa, kun tienviitta kertoo, että ollaan ajamassa väärään suuntaan.

Puhuttaessa ketterien menetelmien eroista vesiputousmaisiin, suunnitelmavetoisiin malleihin, ei voida sivuuttaa näiden prosessimallien päätavoitteita. Suunnitelmavetoiset mallit pyrkivät toimittamaan suunnitellun toiminnallisuuden ajallaan ja budjetissa olettaen, että muutoksia suunnitelmiin tulee vain vähän. Kuten Standish Groupin tilastot todistavat, tässä kuitenkin useimmiten epäonnistutaan. Ketterät menetelmät pyrkivät toimittamaan liiketoiminnan todellisia tarpeita vastaavan ratkaisun tavalla, jolla pysytään budjetissa ja aikataulussa.

Ketterissä menetelmissä tähän tavoitteeseen pyritään käytännössä kolmen pääasiallisen ajurin voimin:

1. Pienet tuotantoerät ja nopea palautesykli
2. Korkea laatu ja alhaiset elinkaarikustannukset
3. Korkea kannattavuus ja alhainen riskitaso

Pienet tuotantoerät ja nopea palautesykli

Ohjelmistoteollisuudessa tuotantoeräksi voidaan käsittää tietty määrä toiminnallisuutta. Ketterissä menetelmissä pienet tuotantoerät tarkoittaa siis ”vähän mutta usein”. Rakennetaan vähän toiminnallisuutta kerrallaan ja viedään valmistuneet erät tuotantoon mahdollisimman nopeasti (esimerkiksi kuukauden välein). Jokaisen tuotantoerän lopputuloksena on nimittäin tuotantokelpoinen versio järjestelmästä. Toisin sanoen, järjestelmä on jatkuvasti siinä tilassa, että julkaisupäätös on puhtaasti liiketoiminnan päätös.

Reaktor Innovationin Sebastian Nykopp pitämässä esitelmää Agile seminaarissa Helsingissä 17.3.06.

Liiketoiminnan kannalta tämä tarkoittaa myös sitä, ettei projektin omistajien tarvitse odottaa hamaan tulevaisuuteen nähdäkseen, mitä toimittaja on saanut heidän rahoillaan aikaiseksi. Ja kun asiakas pääsee antamaan palautetta, muuttamaan priorisointia, korjaamaan väärinkäsityksiä ja toisaalta itse ymmärtämään tarpeitaan entistä aikaisemmin, on tällä kumuloituva positiivinen vaikutus projektin edetessä. Ero todella on valtava siihen nähden, että usein todellisuus paljastuu vasta kun on jo liian myöhäistä.

Korkea laatu ja alhaiset elinkaarikustannukset

Järjestelmän tuotantojulkaisut kuukauden välein tai jopa useammin saattavat ensi silmäyksellä vaikuttaa epärealistiselta, ottaen huomioon ylläpitovaiheen muutoksiin perinteisesti liitetyt suuret kustannukset ja pitkät testausjaksot.

Syklin nopeuttamisen ja ylläpitokustannusten pienentämisen mahdollistavat laaja automaatio, periksiantamaton korkeimman mahdollisen laadun tavoittelu sekä järjestelmän elinkaarikustannusten sisäistäminen.

Ketterät menetelmät tukeutuvat laajaan automaatioon testauksen osalta, millä pyritään tekemään laadunvarmistuksesta nopeaa ja kustannustehokasta. Tämä mahdollistaa muutosten tekemisen helposti, turvallisesti ja nopeasti.

Ketteriä menetelmiä hyödyntävä projekti voi parhaimmillaan alkaa tuottaa lisäarvoa heti projektin alkuvaiheessa. Aikaisin tuotantoon viety järjestelmän osakokonaisuus voi esimerkiksi parantaa johdon näkyvyyttä operatiiviseen toimintaan ja tuoda säästöjä asiakaspalvelun tehostuneen toiminnan myötä. Lisäarvoa tuottavien

osakokonaisuuksien ansiosta ei tarvitse odottaa vuosikausia koko järjestelmän valmistumista ja rahavirran kääntymistä.

Kattavan testauksen tekeminen mahdolliseksi ja kannattavaksi nopeiden iteraatioiden sisällä edellyttää korkean laadun tuottamista iteraatiosta toiseen. Korkea laatu vaikuttaa pitkällä tähtäimellä merkittävästi projektin elinkaaren aikana kertyviin kokonaiskustannuksiin.

Sen sijaan, että uuden toiminnallisuuden lisääminen tulisi päivä toisensa jälkeen vaikeammaksi, hitaammaksi ja kalliimmaksi niin kuin perinteisten menetelmien kanssa usein on käynyt, ketterät menetelmät ajavat IT-organisaatiota kohti tilannetta, jossa muutosten aiheuttama kustannus nousee erittäin hitaasti.

Korkea kannattavuus ja alhainen riskitaso

Ketterien menetelmien periaatteisiin kuuluu, että projektin omistaja voi uudelleen priorisoida järjestelmän toteutettavat toiminnallisuudet ennen jokaista tuotantoerää. Usein tuotantoerälle valikoituu näin sisältö siten, että tuotto on mahdollisimman suuri panostukseen nähden.

Olettaen, että projektin henkilöstökustannukset ovat kiinteät ja että liiketoiminta priorisoi tehtävää työtä sen tuomien hyötyjen tai säästöjen avulla, ketterillä menetelmillä toteutettava projekti pystyy parhaimmillaan tuottamaan positiivisen kassavirran jo ensimmäisestä iteraatiosta alkaen.

Rahoitusmielessä tämä tarkoittaa sitä, että ketterillä menetelmillä on mahdollisuus tehdä IT-projekteista erittäin nopeasti itseänsä rahoittavia tai vastaavasti huomata nopeasti, että projekti ei todennäköisesti tule ikinä olemaan kannattava. Perinteisellä vesiputousmenetelmällä tehtäessä tilanne on varsin erilainen. Käytännössä vesiputousprojekti tuottaa pelkkiä kustannuksia kuukausitolkulla koko budjetin verran ja alkaa onnistuessaankin tuottaa lisäarvoa huomattavasti myöhemmin kuin ketterillä menetelmillä johdetulla projektilla olisi ollut mahdollista.

Ketteriä menetelmiä käyttävä organisaatio sitoo IT-investointeihinsa huomattavasti pienemmän osan varallisuudestaan, minkä voi kääntää suoraan alhaisemmaksi riskitasoksi. Ketteriä menetelmiä käyttävä organisaatio voi halutessaan lopettaa järjestelmän rakentaminen siinä vaiheessa, kun lisätoiminnallisuuden tuoma arvo ei enää kata tekemisen kustannuksia. Näin tuodaan johdolle vaihtoehto, jossa projekti voidaan lopettaa ennenaikaisesti ja silti saada liiketoiminnallisia hyötyjä projektissa siihen asti toteutetun toiminnallisuuden avulla.

Menetelmä on vain työväline

Menetelmäkeskustelussa on tärkeä pitää mielessä ettei mikään menetelmä pysty tarjoamaan helppoa tai automaattista tapaa toteuttaa tietojärjestelmiä laadukkaasti. Tietojärjestelmien menestyksekäs rakentaminen vaatii asiantuntemusta ja motivaatiota, projektimallista riippumatta.

Ketterillä menetelmillä onnistuminen edellyttää yritykseltä ja projektitiimiltä vahvan asiantuntemuksen lisäksi jatkuvaa kurinalaisuutta ja avointa kommunikointia. Epäkohdat, laatuongelmat ja tehottomuudet löytyvät nopeasti ja itse kukin joutuu jatkuvasti vastaamaan tiimille omasta työstään. Perinteiseen malliin verrattuna tämä voi tuntua joidenkin mielestä raskaalta. Pitkällä aikavälillä avoimuudesta ja ketteryydestä hyötyy kuitenkin niin yritys, projekti kuin yksilökin.

Reaktor Innovations Oy on IT-asiantuntijapalveluihin ja ohjelmistoratkaisuihin keskittynyt suomalainen yritys. Reaktorin tarjontaan kuuluvat mm. integraatioratkaisut, asiakaskohtaiset järjestelmätoimitukset, sovelluskehityspalvelut ja koulutuspalvelut. Lisätietoja: www.ri.fi

Helena Venäläinen,
Systeemityöyhdistys SYTYKE ry,
puheenjohtaja 2006

Käytettävyyttä kehittämään!

Hyvä käytettävyys (usability), helppokäyttöisyys tai käyttäjäystävällisyys mainitaan useimpien tietojärjestelmien vaatimuksissa. Kukapa meistä tahtoisi käyttää laitetta tai ohjelmistoa, joka on epäsopiva tai käytettävyydeltään huono. Mutta kuinka moni meistä kuitenkin joutuu jatkuvasti tekemisiin tällaisten tuotteiden kanssa. Miksi tilanne on tällainen? Eikö asialle voisi tehdä jotakin? Mitä SYTYKE voisi tehdä?

Olemme jo viime keväänä käynnistäneet asian pohdinnan johtokunnassa. Ajattelimme tarjota eri yrityksissä ehkä yksinkin asiaa edistäville henkilöille yhteisön, jossa he voivat verkostoitua, vaihtaa kokemuksia käytännössä toimiviksi osoittautuneista käytettävyyden kehittämiskeinoista sekä kehittää omaa käytettävyysosaamistaan.

Yhteisön teemat voisivat liittyä esimerkiksi käytettävyysnäkökulman kokonaisvaltaiseen huomioimiseen järjestelmäkehityksessä, käyttäjäkeskeiseen suunnitteluun tai käytettävyyden testaamiseen ja arviointiin.

Verkkopalveluiden käytettävyys tai informaatiomassojen hallinta ovat nekin laajoja tutkimusalueita. Kiinnostavaa olisi teknisten standardien sijaan hakea toimivia ratkaisuja siihen, miten tieto parhaiten toimisi työn tukena. Miten vaikkapa ohjeita kannattaa kirjoittaa, mikä toimii, mikä ei.

Perustetaan uusi osaamisyhteisö (OSY) ja nostetaan käytettävyys sille kuuluvaan kunniaan arkkitehtuurien ja prosessien rinnalle!

Organisoidutaan, perustetaan postituslista, sovitaan toimintamuodot alkutaipaleelle, valitaan sopiva nimi ja tullaan viimeistään vuoden 2007 alussa esiin SYTYKEeen ja Tietotekniikan liiton verkkosivuilla ja julkaisuissa.

Olen luvannut hoitaa OSYn käynnistysvaiheen organisoinnin, yhteydenpidon ja sähköpostilistan perustamisen. Käytettävyyden kehittämisestä kiinnostuneet henkilöt ympäri Suomea voivat ilmoittaa sähköpostiosoitteensa ja puhelinnumeronsa sekä halutessaan myös kiinnostusalueensa tai odotuksensa osoitteeseen helena.venalainen@op.fi.

Tulkaa joukolla mukaan!

Teksti: Tarmo Toikkanen

Tarmo Toikkanen (PsM) toimii tutkijana ja ohjelmistokehittäjänä Taideteollisen korkeakoulun Medialaboratoriossa, Oppimisympäristöjen tutkimusryhmässä
Lisätietoja:
http://tarmo.fi
http://lemill.org
http://legroup.uiah.fi

XP2006: Ketterät menetelmät hajautetuissa kehitysympäristöissä

XP – eXtreme Programming. Vuosituhannen alussa se oli käytännössä ainoa laajalti tunnettu radikaali ketterä menetelmä ohjelmistojen tekoon. Nykyisin menetelmiä on tusinoittain, mutta XP:n mukaan nimetty vuosittainen konferenssi pitää vanhasta nimestään kiinni. Tänä vuonna XP2006 toi alan gurut Ouluun.

Hajautetut projektit

Merkittävä teema XP2006:ssa olivat jaetut ja hajautetut projektit ja niiden hallinta. Itsekin pidin tästä teemasta esityksen, jonka tässä referoin.

Kun projektiin kuuluvat tiimit eivät sijaitse samassa paikassa, puhutaan jaetusta (distributed) projektista. Tyypillisessä skenaariossa yksi tai useampia tiimejä on sijoitettu halpatuotantomaihin pääarkkitehtien pysyessä yhtiön kotimaassa.

Pääongelmia ovat:

projektin tilan näkyvyys heikkenee,
projektin aika- ja kustannusarviot huononevat,
integrointi ja kokonaisuuden hallinta vaikeutuvat, ja viestintäkanavat heikkenevät.

Kustannusten alentaminen ajaa yhä useamman yrityksen käyttämään ulkomaisia tai alihankintatiimejä, joten jaetun projektin haasteet ovat varsin yleisiä.

Etenkin pienemmissä projekteissa ja PK-yrityksissä yleisiä ovat myös hajautetut (dispersed) projektit, joissa tiimin jäsenet voivat sijaita eri puolilla maailmaa. Usein yksi tiimin jäsen työskentelee Suomessa toisessa kaupungissa muiden ollessa PK-seudulla, tai tiimiin on hankittu ulkomainen erikoisasiantuntija joka pysyy omassa maassaan. Hajautetun projektin haasteet ovat samat kuin jaetunkin, lisättynä yhdellä vakavalla ongelmalla; tiimidynamiikka kärsii.

Ketterien menetelmien pääpiirteet, verrattuna perinteisiin "raskaisiin" menetelmiin, auttavat yllä mainituissa ongelmissa oheisen kuvan mukaan.

Ketterien menetelmien pääpiirteet ja ongelmien ratkaisut

Tiheät toimitukset:
Kun uusi versio julkaistaan 2-4 viikon välein, projektin näkyvyys ei voi heiketä merkittävästi.

Todellisuus dokumenttien sijaan:
Koska työssä käsitellään itse toimivaa ohjelmaa eikä vain speksejä, sekä projektin näkyvyys että arvioiden pitävyys paranevat.

TDD (test-driven development):
Automaattiset testit auttavat laadun valvonnassa, asiakkaan tilaamien ominaisuuksien toiminnan varmentamisessa sekä tuotteen integroinnissa.

Toimintatapojen arviointi ja parannus:
Joka julkaisun yhteydessä työn tekijöiden itse tekemä toimintatapojen arviointi ja parannus nostavat arvioiden laatua ja parantavat näkyvyyttä, sekä tietysti nostavat tuottavuutta.

Tiheät arviot:
Joka julkaisun yhteydessä tehty arvio asiakkaan kanssa parantaa projektin näkyvyyttä ja vähentää kokonaisriskiä.

Jatkuva integrointi:
Poistaa integrointiongelmat käytännössä täysin.

Toisaalta ketterät menetelmät itsekin kärsivät hajautuksesta, pääosin näin; eri paikoissa sijaitsevat tiimit sekä asiakkaan edustaja vaikeuttavat pariohjelmointia, asteittaista määrityksen tarkentamista ja asiakasedustajan täyttä hyödyntämistä. Eri paikoissa sijaitsevat tiimien jäsenet vaikeuttavat pariohjelmointia, heikentävät luottamusta, vaikeuttavat flow'n muodostumista ja hidastavat tiimin itseorganisoitumista.

Yhteenvetona voisi todeta, että ketterien menetelmien peruspiirteet itse asiassa korjaavat monia jaetun ja hajautetun kehityksen vakavimpia ongelmia, eli ne tältä osin soveltuvat raskaita BDUF-menetelmiä paremmin hajautettuihin skenaarioihin. Toisaalta taas ketterät menetelmät luottavat raskaita enemmän ihmisten väliseen kommunikaatioon, jota taas hajautus vaikeuttaa huomattavasti. Monia viestintäongelmia voidaan kuitenkin korjata nykyisillä internetiä hyödyntävillä viestintävälineillä.

Viestinnän tehostaminen hajautetussa kehityksessä

Kuinka sitten parantaa viestintää, kun tiimin jäsenet sijaitsevat pitkin maailmaa? Tässä oma esimerkkini: toimin projektin "scrum masterina", eli eräänlaisena fasilitaattorina. Tehtäväni on siis pitää projekti rullaamassa ja poistaa ongelmat kehittäjien tieltä.

Projekti on 3-vuotinen EU-rahoitteinen R&D-projekti, jossa kehitetään EU-maiden opettajille oppimateriaalin jakelu-, tuotanto- ja hakupalvelua. Kehittämämme LeMill-palvelinohjelmisto on vapaata ja avointa koodia. Kehittäjät sijaitsevat neljässä maassa eivätkä ole ennen tavanneet toisiaan.

Ensimmäinen ja tärkein vaihe on saada kaikki kehittäjät yhteen paikkaan: kasvotusten tapaaminen ja ajan viettäminen on olennaisen tärkeää, eikä sitä mikään videokonferenssi voi korvata. Optimaalinen alussa yhdessä vietettävä aika olisi 2-3 kuukautta, mutta jo viikko on parempi kuin ei mitään. Yhdessäolon aikana tutustutaan, opetellaan toimimaan yhdessä ja koetaan jotain mielenkiintoista (kuten saunominen Suomessa). Tavoitteena on saada luotua jotain yhteistä tiimin jäsenten välille.

Etävaiheen alettua keskeisessä asemassa on "information radiator" eli tiedonvälittäjä. Ilman hajautusta tämä tyypillisesti olisi yhteisen työtilan seinät, mutta hajautetussa tilanteessa tiedonvälittäjä on netissä. Me päädyimme Trac-nimiseen avoimeen ohjelmistoon, joka yhdistää wikin, versionhallinnan (Subversion) ja tikettijärjestelmän yhteen kokonaisuuteen. Tähän järjestelmään tallennamme kaikki suunnitteludokumentit, user storyt, tehtävät, virheraportit, kokousten pöytäkirjat, aivan kaiken. Ja kaikki on kaikkien luettavissa, ei mitään hierarkkisia käyttörajoituksia. Vaarallisinta hajautuksessa on, että joku putoaa kärryiltä, joten kaiken tiedon on oltava netissä saatavilla – käytäväkeskusteluihin ei voi enää luottaa.

Eräs Tracin vahvuus on sen yksinkertaisuus ja joustavuus – wikiä voi käyttää monella eri tavalla. Olemme joutuneet itse opettelemaan toimivat käytännöt ja ne on tallennettu wikiin ohjeiksi. Tämä on koettu selvästi paremmaksi vaihtoehdoksi kuin jäykkä ja kontrolloitu projektinhallintajärjestelmä, sillä tämäkin projekti on vajaan vuoden aikana muuttunut, kasvanut ja kehittynyt niin paljon, että tiukempi järjestelmä olisi moneen kertaan rajoittanut, miten asioita voidaan tehdä.

Pelkkä pull-tekniikka ei kuitenkaan riitä yhteydenpitoon – kaikkien kehittäjien postilaatikkoon kolisee jatkuvasti sähköposteja Trac-järjestelmästä: aina kun joku lähettää lähdekoodia versionhallintaan tai mihin tahansa tikettiin tehdään muutoksia, kerrotaan tästä kehittäjien postituslistalla. Toista postituslistaa käytetään kaikkien asianomaisten väliseen kommunikointiin.

Lisäksi kehittäjät aina töissä ollessaan liittyvät yhteiselle IRC-kanavalle, jossa he voivat selvitellä kulloinkin ajankohtaisia ongelmia. Kanavalla on myös botti, joka kertoo Tracissa tapahtuneista muutoksista. Muutokset ovat saatavilla myös RSS-feedinä.

Työn organisoinnin suhteen suuria ongelmia oli alussa siinä, että yksikään kehittäjä ei tee pelkästään tätä projektia. Hyväksi ratkaisuksi havaitsimme viikon mittaiset keskitetyt kehityssprintit, jolloin kaikki kehittäjät tekevät vain tätä projektia. Toinen toimiva ratkaisu on wikiin tehty viikkokalenteri, johon kehittäjät merkitsevät, milloin ovat töissä projektin parissa.

Työn suunnittelua tehdään postituslistan lisäksi myös audio- ja videokonferensseissa. Skype ja FlashMeeting ovat projektissa yleisimmin käytettyjä välineitä. Ja tietysti niitä pakollisia kasvotusten tapaamisiakin on, mutta vain muutamia vuodessa.

Kaikki käyttämämme ohjelmistot ovat vapaita tai ainakin ilmaisia ohjelmistoja. Niiden avulla monet hajautetun työskentelyn ongelmat on ainakin lievennetty niin, että toistaiseksi (10 kuukautta projektin alkamisesta) olemme aikataulussa. Tuottavuus on kuitenkin ehdottomasti alle sen, mitä yhtenäisellä tiimillä voitaisiin saada, eli hajautetusta tiimistä ei ainakaan näillä lääkkeillä saa samaa tehoa irti kuin motivoituneesta, samassa tilassa toisiaan tukevasta tiimistä.

Seuraavalla sivulla XP lyhenteitä ja terminologiaa.

XP lyhenteitä ja terminologiaa (T. Toikkanen)

BDUF: Big Documentation Up-Front, eli dokumentit edellä etenevä projekti. Projektille tyypillistä on kattava dokumentointi ja koodin ilmaantuminen vasta verrattain myöhään projektissa. Vertaa TDD.

FlashMeeting: Tutkimustyöhön tarkoitettu videokonferenssipalvelu, joka vaatii ainoastaan Flashia tukevan selaimen, kaiuttimet, mikrofonin ja valinnaisesti web-kameran. www.flashmeeting.com. Monia vastaavia palveluita on olemassa vaihtelevin hinnoitteluin.

IRC: Suomessa kehitetty jutustelualusta Internet Relay Chat, jota etenkin ohjelmistokehityspiireissä suositaan. IRCin kanaville voidaan luoda botteja, jotka valvovat kanavaa, poistavat spämmääjät, antavat lisätietoa kanavaan liittyvistä aiheista, jne.

RSS: Uutisvirtojen välittämiseen käytetty xml-formaatti. Modernit selaimet osaavat käsitellä rss-feedejä suoraan (esim. Firefoxin ”Live Bookmarks”).

Scrum master: SCRUM-projektinhallintamenetelmään kuuluva rooli, joka on sekä hallinnon että kehittäjien ulkopuolella. Scrum Masterin tehtäviin kuuluu organisatoristen esteiden poistaminen kehitystyön tieltä sekä kehittäjien tukeminen SCRUMin sääntöjen noudattamisessa. http://scrumalliance.org

Skype: Suosittu ilmainen internet-puhelujärjestelmä, jolla voi soittaa myös tavallisiin puhelimiin. http://www.skype.com

TDD: Test-Driven Development, eli testit edellä etenevä projekti. Tyypillisesti TDD-projektissa vaatimusmäärittely kuvataan automaattisesti suoritettavina testeinä, joita lähdetään toteuttamaan. Minkä tahansa ominaisuuden toteutus aloitetaan tekemällä yksikkötestit, ja kun testit menevät läpi, on ominaisuus valmis. Dokumentaatiota syntyy verrattain vähän, ja tyypillisesti vasta projektin lopuksi – aiemmin tuotettu dokumentaatio kun vanhenee välittömästi.

Tikettijärjestelmä: Järjestelmä, johon kaikki tehtävät kirjataan, arvioidaan, tarkennetaan, annetaan tehtäväksi ja kuitataan tehdyiksi. Muita termejä: issue management, trouble ticket system.

User story: Joissain ketterissä menetelmissä käytetty vaatimusten dokumentointitapa. User storyssa kuvaillaan asiakkaan näkökulmasta, asiakkaan ymmärtämällä kielellä, miten tietty toiminto tapahtuu. Asiakkaan on täten mahdollista ymmärtää ja varmistaa, että ohjelma todella toimii niin.

Wiki: Yhteisöllinen sivusto, jossa kaikki jäsenet voivat tasavertaisesti ja helposti muokata kaikkia sivuja. Toisin kuin useimmat muut internetin viestintävälineet, Wiki kannustaa yhteisymmärryksen muodostamiseen.

Teksti: Pekka Abrahamsson
Outi Salo
Kuva: XP2006-tapahtuman kuva-arkisto

Ketteriin menetelmiin keskittyvä XP2006-konferenssi oli menestys!

Pekka Abrahamsson
tutkimusprofessori,
XP2006 -ohjelmatoimikunnan päällikkö
VTT

Outi Salo
tutkija,
XP2006 local chair
VTT

Ketterät (engl. agile) menetelmät ovat olleet ohjelmistoteollisuuden saatavilla 1990-luvun lopulta lähtien. Kirjallisuutta aihepiiristä on ilmestynyt ripeämmällä tahdilla kuin mistään muusta ohjelmistoalan innovaatiosta pitkään aikaan. Suomessakin on ketteryyteen liittyvistä tutkimustuloksista ja yritysten saavutuksista uutisoitu viime aikoina.
Ketteriin menetelmiin on ladattu paljon odotusarvoa, kuten ohjelmistotuotannon kustannusten aleneminen, ohjelmistovirheiden määrän väheneminen ja tuotantosyklin nopeutuminen sekä nopea reagointi asiakkaalta tuleviin muutostarpeisiin. Perustana on ajattelutavan muutos, jossa painopiste on eri sidosryhmien kommunikaatiossa ja yhteistyössä sekä muutosherkkyydessä.

Oulussa järjestettiin kesäkuussa 2006 ketteriin menetelmiin ja niiden soveltamiseen keskittyvä konferenssi XP2006, jossa suomalainen ohjelmistoteollisuus oli kattavasti läsnä. Noin 300 konferenssiin osallistujasta joka toinen oli suomalainen. Osallistujia saapui 25 maasta ja 85 % heistä oli teollisuuden edustajia. Yrityksiä paikalla oli yli 100. Tämä sinänsä on jo poikkeuksellista, koska konferenssi keskittyy menetelmiin ja prosesseihin, eikä niinkään työkaluihin tai kehitysteknologiaan.

Eurooppalaiset ovat ottaneet vuosittain järjestettävän XP-konferenssisarjan omakseen sillä, osallistujista peräti 90 % oli Euroopan eri maista. Suomen lisäksi paljon osallistujia oli Englannista ja Ruotsista.

Teollisuuden vahva läsnäolo näkyi myös konferenssin ohjelmasta. Perinteisen tutkimuksen ja kokemuksiin keskittyvien sessioiden lisäksi konferenssissa oli 18 puolen päivän tutoriaalia, 13 interaktiivista työpajaa, kolme paneelikeskustelua ja yli 20 kellon ympäri aikataulutettua open space tapahtumaa, jossa läsnäolijat tunnistivat uusia sessioiden aiheita ja itseorganisoituvasti järjestivät ne. Pääpuhujina tapahtumassa oli ohjelmistoteollisuuden legendat Barry Boehm (Software engineering economics kirjan kirjoittaja ja Spiraalimallin kehittäjä) ja Kent Beck (Extreme Programming menetelmän kehittäjä). Lisäksi filosofi Pekka Himanen, Jack Järkvik Ericssonilta ja Sean Hanly Exoftwarelta Irlannista käyttivät avauspuheenvuoronsa. Ensi vuoden XP2007 -konferenssi järjestetään kesäkuussa Comossa, Italiassa.

Voidaan väittää, että ketterät menetelmät sekä niiden uudet variaatiot ovat tulleet jäädäkseen hyväksyttyinä ohjelmistojen kehittämisen teknologioina. Tätä kuvaa se, että IEEE:n standardikomitea on ottanut tavoitteisiinsa kehittää ohjeistuksia ketterien menetelmien soveltamiseen. Ensi vaiheessa standardi keskittyy asiakkaan roolin jäsentämiseen sekä siihen, millaisia tuotoksia asiakas voi olettaa ketteriä menetelmiä käyttävän organisaation tuottavan. Standardia työstävässä ryhmässä on Suomesta mukana jäseniä VTT:ltä, F-Securelta ja Nokialta. Nykyinen versio on vedos kolme ja sitä pyritään testaamaan tulevan syksyn ja talven aikana empiirisesti. Työryhmän vetäjänä toimii yhdysvaltalainen Scott Duncan, joka esiintyi myös XP2006 -konferenssissa.

Ketterien menetelmien ongelmat ja haasteet liittyvät niiden soveltamiseen moniteknologisissa ympäristöissä ja toimintaan asiakkaiden kanssa, jotka haluavat pelkästään kiinteähintaisia projekteja. Tilannehan on sinänsä paradoksaalinen, koska kiinteähintaisuus perustuu tietyn toiminnallisuuden, resurssien ja aikataulun kiinnittämiseen, joka puolestaan sallii joustamisen lähinnä laadun kustannuksella. Ketterät toimintamallit korostavat suunnittelun syklisyyttä, hyvää laatua ja toiminnallisten piirteiden hallittua priorisointia sekä teknisessä että liiketoiminnallisessa mielessä.

Ei voida väittää, että ketterät menetelmät ratkaisisivat kaikki ohjelmistotuotannon ongelmat. Kuitenkin ne ovat hyvä pelinavaus uudentyyppiselle keskustelulle ja käytännölle alalla. Suuret ohjelmistostandardit kuten CMM(I) jyräsivät 1990-luvulla vahvasti ja ketteryys voidaan nähdä jonkinlaisena vastavetona tälle liikkeelle. Nyt havaitaan, että ohjelmistojen kehittämisessä voidaan käyttää useita vaihtoehtoisia polkuja. Ammattimaisuus, hyvät suunnittelu- ja toteutuskäytänteet sekä selkeät toimintamallit ovat aina tarpeen, kun hyviä ohjelmistoja kehitetään.

kuva: Spiraalimallin kehittäjä Barry Boehm (kuvassa) oli yksi XP2006 –tapahtuman pääpuhujista

Ketteriin menetelmiin liittyvää koulusta, tutkimusta ja tiedonvaihtoa tapahtuu aktiivisesti myös Suomessa. Agile Alliance -sivuilta (www.agilealliance.org) löydät myös Suomen tapahtumia käsittelevää tietoja. Tiedossa on mm. koodaustapahtumia pitkin syksyä. Tapahtumat ovat yritysvetoisia ja esimerkiksi suomalainen Reaktor Innovations toimii aktiivisena verkostossa. Lisäksi Jyväskylässä pidetään marraskuussa vuotuinen agile -seminaari, jossa tutkimus- ja yritysmaailma esittelee viimeisiä tuloksiaan. Tietoa tapahtumasta löytyy VTT:n agile -sivustoilta (agile.vtt.fi) syyskuun aikana.

Webbilinkkejä:
http://www.xp2006.org
http://www.agilealliance.org
http://agile.vtt.fi

Lisätietoja ensi kesäkuussa pidettävästä XP2007 konferenssista (Como-järvi, Italia):
http://www.xp2007.org

Teksti: Harri Töhönen

Harri Töhönen,
CTO / Qentinel Oy
harri.tohonen@qentinel.com
040 5527035

Kirjoittaja on yksi ohjelmistojen laadunvarmistukseen ja testaukseen keskittyneen Qentinel Oy:n perustajia. Hänellä on yli 14 vuoden kokemus ohjelmistokehityksestä- ja liiketoiminnasta. Tällä hetkellä hän vastaa Qentinelin palvelutuotteiden kehityksestä ja ylläpidosta. Viimeisten vuosien aikana hän on toiminut myös testauskonsulttina, testauspäällikkönä ja testausprosessien kehittäjänä.

Laatua projektiin; mekaaninen ja orgaaninen taso

Projektityössä kustannustehokkaan laadun tuottamiseen on panostettava eri tasoilla. Liiketoimintatasolla laatua tavoitellaan sopimuksellisilla ja liiketoiminnallisilla tavoitteilla. Nämä kiteytetään projektin laatutavoitteiksi sekä edelleen työsuunnitelmiksi ja tehtäviksi. Systeemi- ja projektityön ohjeet voidaan nähdä mekaanisina laatuun vaikuttavina toimina.

Orgaaninen taso näkyy yksittäisen projektijäsenen henkilökohtaisena sitoutumisena, laadun tiedostamisena ja toimintatapoina. Orgaanisella tasolla toimet ovat usein moninaisia ja suhteessa hyvinkin pieniä, mutta niillä voi olla ratkaiseva merkitys mekaanisten toimien onnistumiselle ja kustannustehokkuudelle.

Lähtöpiste - määritä oma laadun tarpeesi

Projektien onnistumista mitataan usein aikaan sidotulla kustannustehokkuudella. Tässä optimointitehtävässä laatu on yksi suurimmista pelinappuloista. Sen oikealla määrittelyllä ja rajaamisella vaikutetaan niin projektin aikaisiin kuin sen tulosten ylläpidollisiinkin kustannuksiin.

Laadun määrittelyssä on oleellista aloittaa tarkastelu projektin aiotuista lopputuloksista. Laadun määritys ei löydy valmiina organisaation laatusuunnitelmista, mallipohjista tai edellisten projektien määrityksistä. Asiaa voidaan lähestyä kysymällä: Mikä tekee projektin tuotoksesta onnistuneen liiketoiminnalle? Onko se voluumi, minimoitu käyttäjätuen tarve vai helppo päivitettävyys? Minkä takia käyttäjät tulevat arvostamaan lopputulosta? Onko se monipuolinen toiminnallisuus vai yksinkertainen helppokäyttöisyys? Mitä lainsäädäntö ja alan määräykset tarkoittavat lopputuloksen kannalta? Vastaamalla näihin kysymyksiin muodostetaan projektin laatutavoitteet ja luodaan pohjaa tarkemmalle vaatimusten määrittelylle ja priorisoinnille.

Mekaaninen ulottuvuus - prosessi

Prosessi on keskimääräinen tapa tuottaa keskimääräistä laatua. Prosessi ei kerro projektille mitä laatu on mutta se antaa raamit ja ohjeet laadun rakentamiselle ja varmistamiselle. Laadun rakentamista edistävät oikein mitoitetut ja lomitetut suunnittelun, tekemisen ja laadunvarmistamisen iteraatioita. Työnkulut, roolitukset, tarkistuskäytännöt, mallipohjat ja tarkistuslistat ovat prosessia tukevia työkaluja, joiden avulla pyritään välttämään kaaosta ja pyörän uudelleenkeksimistä. Lisäksi prosesseissa on usein myös itse prosessin kehittämiseen ja suorituskykyyn liittyviä mittareita ja käytäntöjä. Hyvä prosessi jättää projektille liikkumavaraa soveltamiseen ja kustannustehokkaimpien menetelmien ja käytäntöjen valintaan.

Projektikohtainen laatustrategia sisältääkin juuri laatutavoitteita vasten optimoidut valinnat ja prosessin soveltamissuunnitelman. Laatustrategia tiivistyy testausstrategiaan, jossa laatutavoitteiden ja tunnistettujen riskien ohjaamana valitaan testauksen painopistealueet, testausprosessit ja tärkeimmät testausmenetelmät. Projektin edetessä laatutavoitteisiin ja riskien kattamiseen palataan aina testauksen tuottaman tilannetiedon avulla.

Kuva 1. Projektin tavoitteet, laatutavoitteet ja riskit laatustrategian ohjaajina. Laatustrategia sisältää projektikohtaiset valinnat ja mitoitukset käytettäville laadunvarmistuksen prosesseille ja mekanismeille.

Orgaaninen ulottuvuus - soveltaminen

Tavoitteiden ja prosessien tarkastelun jälkeen päästään projektin onnistumisen tärkeimpään tekijään eli ihmiseen. Laatutavoitteet ja niiden varmistamiseen liittyvät toimet sekä valinnat on motivoitava ja siirrettävä laatustrategian tasolta jokaisen projektin jäsenen omakohtaiseksi missioksi. Jokaisen on nähtävä kuinka oma rooli ja tekeminen vaikuttavat osakokonaisuuksien kautta lopullisiin laatutavoitteisiin.

Eräänä sitouttamiskeinona voidaan käyttää henkilö- tai ryhmäkohtaista 'eläytymistä' laatuun. Eläytymisessä esimerkiksi tietyn osajärjestelmän tai ominaisuuden suunnittelusta vastaavalle ryhmälle annetaan tarkasteltavaksi muutama laatutavoite, joita vasten ryhmä itse määrittelee, mitä laatu tarkoittaa heidän työssään. Laatu voi edellyttää erityishuomiota tiettyihin teknisiin riskialueisiin, se voi olla tehostettua vuorovaikutusta toisen kehitysryhmän kanssa tai se tarkoittaa nopeutettua palaute - korjausketjua.

Päävastuu työntekijöiden 'laadukkaan tekemisen' eli laatutaitojen kouluttamisesta ja kehittämisestä on organisaatiolla, mutta projektitasollakin voidaan arvioida millaisia yhteisiä tekemisen malleja ja periaatteita kannattaa noudattaa. Esimerkkejä usein yksinkertaisista mutta hyödyllisistä käytännöistä ovat ajanhallinnan, priorisoinnin ja kommunikoinnin niksit:

- tee ensin lista päivän tehtävistäsi ennenkuin syöksyt tekemään niitä
- tee asia ja ajatus kerrallaan, kasaa keskeytykset listaan
- älä piiloudu roolin tai prosessin taakse, muista yhteinen tavoite
- ajattele kahdesti ennenkuin hylkäät 'yksinkertainen on kaunista'
- tunnista itsessäsi NIH (Not Invented Here) -ilmiö

Laatutaitojen oppiminen on tehokasta juuri projektityön aikana, jolloin projektilaisia lähellä olevien käytännön syy-seuraus -esimerkkien ja kokemusten kautta teoriat voidaan omaksua ja jäsentää toistettaviksi taidoiksi. Tämä edellyttää onnistuneiden tapahtumien tunnistamista ja pyrkimystä löytää yksilökohtaisista suorituksista ominaisuuksia ja tekijöitä, jotka voidaan kommunikoida ja opettaa eteenpäin. Myös epäonnistumisista voidaan oppia vastaavalla tavalla. Jokainen projektin jäsen voi harjaantua ympäristönsä 'laatutietoiseen' tarkkailuun, mutta laatuvastaavien, projektipäälliköiden ja muiden vetovastuussa olevien pitäisi tietoisesti kehittää ja hyödyntää tätä taitoa.

Projektipäälliköille tärkeä ominaisuus on myös pystyä tunnistamaan ja hyödyntämään työntekijöiden vahvuuksia. Usein kannattaa harkita milloin osaprosessia tai tehtävää säätämällä saadaan tekijöiden parhaat ominaisuudet käyttöön. Esimerkiksi toinen henkilö nauttii tehdessään asioita proseduraalisesti - anna hänelle urakoita. Toinen taas on erinomainen moniajossa - häneltä sujuvat siis koordinointi tai ympäristön tukitehtävät. Toinen pystyy tekemään ja dokumentoimaan formaaleja testitapauksia, kun taas toinen on mestari löytämään vikoja intuition ja tutkivan testauksen avulla.

Laatu - kokonaisuus

Laadun tuottaminen projektissa voidaan jakaa mekaanisempaan makrotasoon ja ihmisläheisempään mikrotasoon. Tasojen merkitys on hyvä tunnistaa ja ymmärtää; kumpikaan ei yksinään riitä vaan niiden kokonaisuus ratkaisee.

Hyvä laatujohtaminen on tasapainon löytämistä asiajohtamisen ja ihmisjohtamisen välillä, prosessin ja soveltamisen harmoniaa.

Ei voi välttyä ajatukselta, että tässä tasapainon löytämisessä on myös selkeä analogia suunnitelmaohjatun (plan-driven development) ja ketterän (agile development) kehityksen mallien väliseen tasapainoiluun, mutta tämä tarkastelu onkin jo oman jatkopohdiskelunsa arvoinen asia.

Teksti: Pertti Virtanen

Pentti Virtanen, Systeemityön asiantuntija, Tieturi Oy FT tietojenkäsittelytiede, Väitöskirja: "Measuring and Improving Component-Based Software Development", vuonna 2003.

Malliohjattu ohjelmistoarkkitehtuuri ja tuottavuus

Ohjelmistotuotannon tehtävänä on luoda yhteinen ajatusmalli jonkun yleensä liiketoimintaan liittyvän ongelman ratkaisemiseksi. Sitä voi verrata paremmin uuden Ferrari-mallin suunnittelemiseen kuin T-Fordin tuotantolinjaan. Lopputulos ei ole ennalta tiedossa vaikka jonkinlainen visio ohjaakin työn etenemistä. Samalla tavoin kuin autoteollisuudessakin tuote kehittyy testattavien prototyyppien avulla. Ohjelmistojen ajatusmalliluonne näkyy siinä, että väli- ja lopputuotokset ovat abstrakteja.

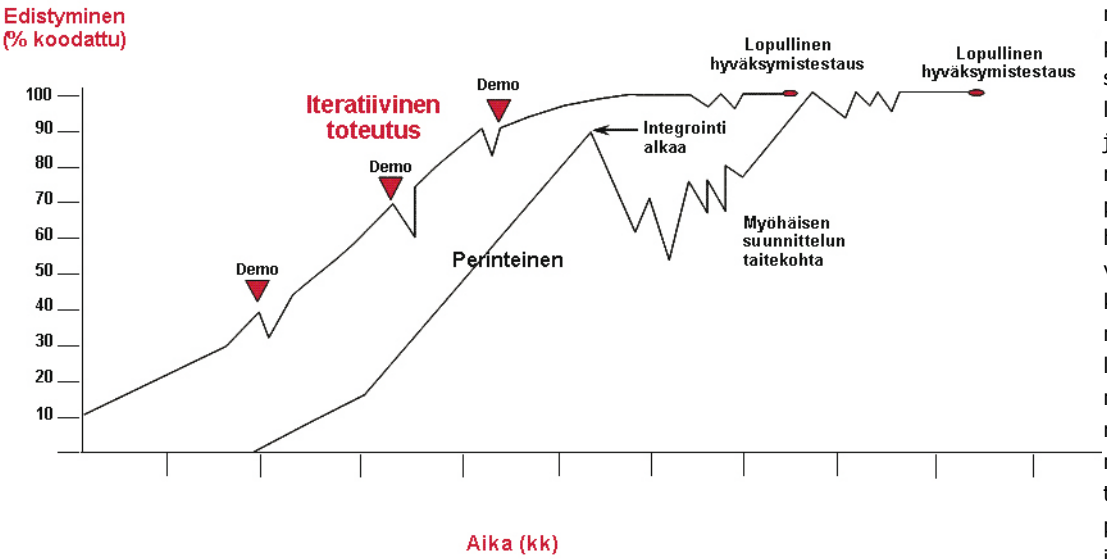
Arkkitehtuurin merkitys

Arkkitehtuuri on osa suunnittelua. Se kuvaa tuotteen ja sen keskeiset osat korkealla abstraktiotasolla. Se määrittää, millainen auto uusi Ferrari on. Olennaista on myös päättää, mikä osa tuotteesta voi muuttua, mikä ei. Fordin T-malli oli kuuluisa siitä, että sen sai vain mustana. Tuotearkkitehtuurin määrittelyn jälkeen muotoilijat, moottorisuunnittelijat ja muut suunnittelutyöhön osallistujat voivat itsenäisesti jatkaa suunnittelutyötä. Tuotearkkitehtuuri siis mahdollistaa suuren ihmisjoukon rinnakkaisen ja tuottavan yhteistoiminnan.

Ohjelmistoarkkitehtuurin tärkein tehtävä on jakaa rakennettava ohjelmisto itsenäisiin osiin ja määrittää osien väliset rajapinnat. Ohjelmistoarkkitehtuurin kerrosrakenne, ohjelmointikieli ja tekninen alusta ovat ohjelmistojen vaikeasti muutettavia osia. Kun arkkitehtuuri on kiinnitetty, erillään työskentelevät ohjelmistokehittäjät ja tiimit voivat tuottaa, testata ja integroida tarvittavat komponentit.

Ohjelmistotuotanto on pahamaineisen muutosherkkää. Muutoksia tulee paljon ja ne ovat laajoja eikä niitä voida jättää toteuttamatta heikentämättä syntyvän ohjelmiston käyttökelpoisuutta.

Perinteisessä toteutuskeskeisessä ohjelmistotuotannossa arkkitehtuuri ja muu suunnittelu laaditaan projektin alussa ja niiden dokumentaatio jäädytetään. Jo pelkästään dokumenteissa olevat puutteet ja ristiriitaisuudet aiheuttavat toteutustyön aikana suuren joukon korjauksia. On myös tavanomaista, että toteutus ja dokumentoitu arkkitehtuuri ovat ristiriidassa keskenään. Arkkitehtuurin ja koko ohjelmiston toimivuus selviää lopullisesti vasta tuotannon alkaessa. Projektin loppuvaiheeseen kuulu laaja integraatiotestausvaihe, jolloin tyypillisesti havaitaan suuri joukko ongelmia, joiden korjaamiseen ei ole varattu riittävästi aikaa ja resursseja. Kiireessä tehty työ on laadultaan ja tuottavuudeltaan heikkoa. (Kuva 1)



Kuva 1. Jatkuva integraatio [Jacobson et al: The Unified Software Development Process, Addison-Wesley, 1999]

Ketterät ohjelmistokehitysmenetelmät korostavat jatkuvaa synkronointia niin tiimien kuin kehittäjienkin välillä. Ymmärrettävästikin liiallinen muuttuminen heikentää tuottavuutta. Toisaalta monet ohjelmistotuotannon pahimmista epäonnistumisista ovat johtuneet virheellisistä arkkitehtuureista, jotka on jäädytetty ennen riittävää testausta. Kehitysprojektit jaetaan iteraatioihin, joiden kesto vaihtelee viikosta muutamaan kuukauteen riippuen mm. menetelmistä, projektin laajuudesta ja muutosten määristä. Myös arkkitehtuureja muutetaan vähitellen, mutta arkkitehtuuriprojektien iteraatiot ovat pidempiä kuin kehitysprojektien iteraatiot etenkin, jos arkki-

tehtuuri koskee kokonaista tuoteperhettä tai suurempaa projektijoukkoa.

Tuottavuuteen vaikuttavat tekijät

Ohjelmistotuotannon tuottavuuden keskeisimmät tekijät ovat kehittäjien motivaation ja osaaminen. Tiimityön tehokkuus, riskit ja käytetyt prosessit ja työkalut ovat myös merkittäviä. Koska ohjelmiston elinkaaren aikaisista kustannuksista suuri osa syntyy ylläpitovaiheessa, on lopputuloksen ylläpidettävyys hyvin keskeinen kustannustekijä.

Arkkitehtuurin vaikutus tuottavuuteen kehitystyön aikana syntyy arkkitehtuurin tuotantotiimien integraatiotyötä helpottavasta vaikutuksesta. Toisaalta arkkitehtuuridokumentaation tekemiseen ja ylläpitämiseen kuluva työmäärä heikentää tuottavuutta. Parannuksena tähän on kaksinkertaisen työn välttäminen. Ohjelmakoodi generoidaan suunnittelumallista ja suoraan ohjelmakoodiin tehdyt muutokset palautetaan malliin. Parhaimmillaan malli ja koodi muodostavat yhden yhtenäisen kokonaisuuden. Hyvä arkkitehtuurityö vähentää myös projektin toteutuneista riskeistä johtuvaa työmäärää.

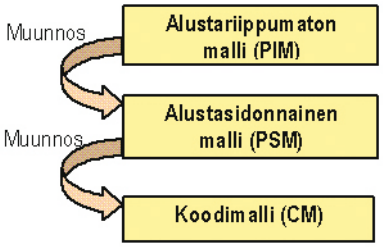
Uudelleenkäytön suunnittelu on luonnollinen osa arkkitehtuurityötä osana tiimien ja kehittäjien välisen työn integrointia. Periaatteessa myös uudelleenkäytöllä on mahdollista saavuttaa merkittäviä tuottavuushyötyjä. Käytännössä organisaatioiden jäykkyydet tekevät uudelleenkäytöstä uudelleenkirjoittamista kalliimpaa.

Hyvä arkkitehtuuri vähentää ylläpitotyötä. Muutosten kohteena olevat komponentit löytyvät helposti ja nopeasti. Komponenttien toiminta on helposti ymmärrettävää ja itse muutostyö voidaan tehdä luotettavasti ja vähällä vaivalla, kun arkkitehtuurissa määritellyt rajapinnat rajaavat muutosten vaikutusalueet.

Malliohjattu arkkitehtuuri

Object Management Groupin malliohjattu arkkitehtuuri Model Driven Architecture (MDA) yhdistää ohjelmistojen mallinnuksen ja koodauksen yhdeksi kokonaisuudeksi. Riippuvuus ohjelmisto- ja laitealustasta on koettu keskeiseksi ongelmaksi, jonka ratkaisemiseksi ensin ohjelmisto mallinnetaan alustariippumattomaksi. Kun alusta on valittu, muunnetaan alustariippumaton malli CASE-työkalua käyttäen alustasidonnaiseksi malliksi. Alustasidonnaisesta mallista voidaan tämän jälkeen generoida ohjelmakoodi. Useat nykyiset CASE-työkalut tukevat koodirunkojen generointia ja mallien luontia ja päivittämistä ohjelmakoodista lähtien (kuva 2). Olennaista on se, että mallit ja koodi pysyvät yhdenmukaisina koko ohjelmiston elinkaaren ajan.

On myös tärkeää, että suunnittelu ja etenkin arkkitehtuurimallit ovat suoritettavia. Tämä mahdollistaa niiden simuloinnin. Kehittyvä koodi lisätään ja yhdistetään malliin, jolloin yhdenmukaisuus voidaan tarkistaa kääntäjien, simulointien ja testauksen avulla. Työstä suurempi osa tehdään mallinnuksena, mutta mallinnuksen laatu on parempi ja mallista tulee osa lopullista tuotetta. On myös periaatteellisesti oikeampaa ajatella ohjelmistoa kehittyvänä prototyyppinä erillisten spesifikaatiodokumenttien ja koodin sijaan.



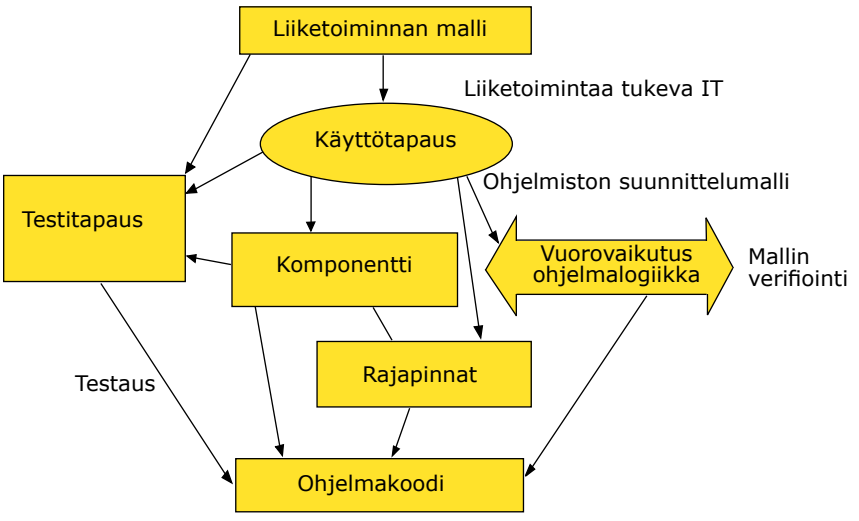
Kuva 2. Model Driven Architecture

Liiketoimintaa tukeva ohjelmisto rakentuu tuettavan liiketoiminnan mallin pohjalle. Liiketoiminnan prosessimalli kuvaa kuinka ihmiset ja organisaatiot työskentelevät keskenään. Käyttötapauksissa kuvataan kuinka ihmiset vuorovaikuttavat tietojärjestelmien kanssa liiketoiminnan aktiviteetteja suorittaessaan. Ohjelmisto toteuttaa käyttötapaukset vuorovaikuttavina komponentteina ja rajapintoina. Komponenteista generoidaan edelleen ohjelmakoodia, jota mahdollisesti vielä tarkennetaan erikseen. Syntyvää kokonaisuutta verifioidaan mallin yhdenmukaisuustarkistuksilla ja testitapauksilla. (kuva 3).

Johtopäätös

Ohjelmistoarkkitehtuuri jakaa rakennettavan ohjelmiston itsenäisiin osiin ja määrittää osien väliset rajapinnat. Erillään työskentelevät ohjelmistokehittäjät ja tiimit voivat tämän seurauksena tuottaa tarvittavat komponentit tehokkaasti ja integraatiotyötä säästäen. Perinteisessä toteutuskeskeisessä ohjelmistotuotannossa integraatiotyö perustuu havaittujen ongelmien korjaamiseen ja tapahtuu kiireessä ja kalliisti projektin loppuvaiheessa. Malliohjattu ohjelmistoarkkitehtuuri (MDA) on tuottavampi kuin dokumenttikeskeinen arkkitehtuuri, koska arkkitehtuurimallin ristiriidat ja ongelmat havaitaan ajoissa. Lisäksi suoritettava arkkitehtuurimalli ohjaa kehittäjien työtä ja varmistaa arkkitehtuurin asettamien vaatimusten noudattamisen myös toteutuksessa.

Kuva 3. Mallien väliset suhteet.



Teksti: Tommi Kilpeläinen, Kronodoc Oy

Dokumenttien hallinnalla laatua ja tuottavuutta

Kirjoittaja toimii Kronodoc Oy:ssä myynnin ja konsultoinnin tehtävissä. Aikaisemmin kirjoittaja on toiminut projektipäällikkönä erilaisissa tietotekniikan tuotekehitys- ja asiakastoimitusprojekteissa.

Projektiemme luonne on muuttunut 2000-luvulla merkittävästi. Yritysten ja yhteisöjen projektikulttuureja, moniprojektiympäristöjen hallintaa, projektiosaamista, resurssien ja aikataulujen hallintaa on kehitetty kiivaasti. Samalla, kun projektien läpimenoaikoja on haluttu nopeuttaa, niin projektien dokumentointi ja projektin aikana syntyvien dokumenttien hallinnan merkitys on kasvanut. Vastaavasti projektien toteutuksen luonne on muuttunut. Projektimme ovat globaaleja ja monitoimittajaprojekteja, joissa tiedon jakaminen on aikaisempaa tärkeämpää, jotta halutut lopputulokset voidaan saavuttaa annetussa aikajanassa.

Lähes kaikki toimenpiteet projekteissa sisältävät kommunikointia ja dokumentointia; projektin avauslomake, muistio, pöytäkirjat, projektisuunnitelma, väliraportti, loppuraportti, manuaalit, etc. Samalla nämä dokumentit ovat myös tehtäviä, lähtötietoja jatkotoimenpiteille tai projektin tärkeitä välietappeja eri osapuolille. Eri yritysten verkkolevyt ja henkilökohtaiset sähköpostit sisältävät turha usein edellä mainittua tietoa ja toimivat viestintäkanavina. Nämä lainalaisuudet

toistuvat sekä pienissä että suurissa projekteissa toimialasta riippumatta.

Edellä mainittua muutosta tukemaan on kehitetty ryhmätyöskentelyä tukevia dokumentinhallintaohjelmistoja, joiden avulla voidaan hallita strukturoitua projektimallia, tehtävien edistymistä sekä jakaa projektien etenemisen aikana syntyvää tietoa ja dokumentaatiota. Ani harvoissa ratkaisuissa on kuitenkaan otettu huomioon, että dokumentaation etenemisen kautta voidaan myös mitata projektin valmiusastetta, laatua ja tuottavuutta.

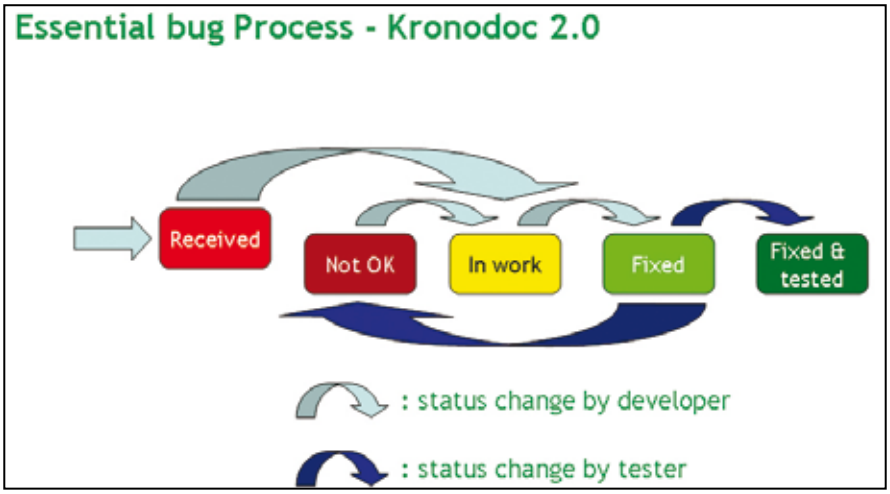
Tätä mittaamista varten dokumentaation lisäksi tarvitaan metatietoa; versionhallintaa, käyttöoikeuksia, elinkaaritietoa, päivämäärätietoa, erilaisia luokittelutietoja, etc. Eräissä ohjelmistoprojekteissa on hyväksi käytetty dokumentinhallintaohjelmistoa tukemaan ja mittaamaan muutoksia, jonka lopputuloksena on saatu mitattua tehokkuuden ja laadun parantumista sekä pystytty vertailemaan projektien tehokkuutta keskenään. Varsinkin ohjelmistoteollisuudessa ja etenkin valmisohjelmistojen osalta tällainen on mahdollista, koska uuden ja vanhan version tuotekehitysprosessit ovat helpostikin vertailukelpoisia.

Ohjelmistoprojektin luonteenpiirteet

Ohjelmistoprojekti käsittää toimenpiteitä, jotka varmistavat, että ohjelmisto toimitetaan ajallaan täyttäen kaikki asiakasorganisaation ohjelmistolle ja sen tuottamiselle asettamat vaatimukset. Valmisohjelmiston tuotannossa pitää vielä laajemmin ottaa huomioon eri asiakasryhmien vaateet ja ohjelmiston mahdolliset virheet sekä toiminnallisuuksien että aikataulujen suhteen ja yhteen sovittaa muutokset aikaisempien tuoteversioiden kanssa.

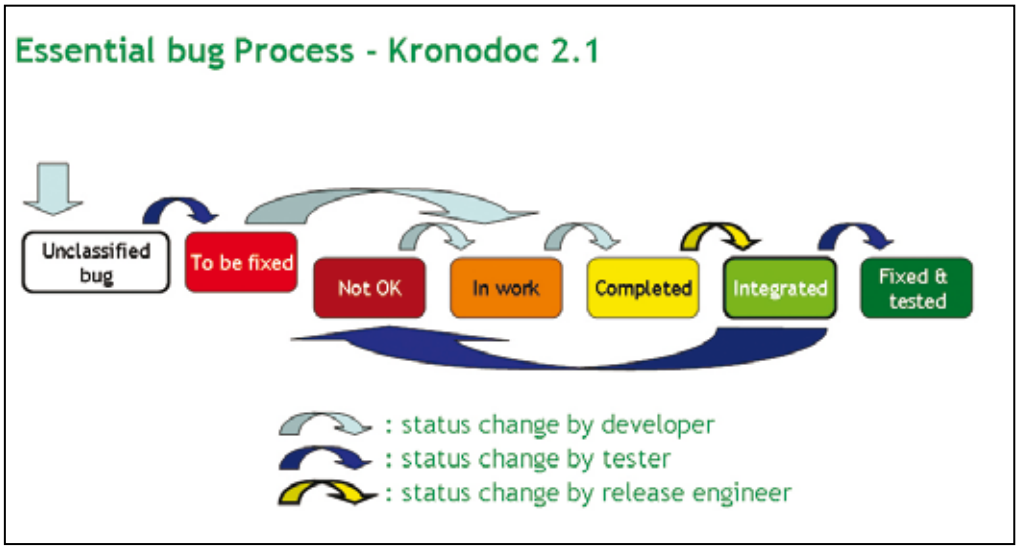
Ohjelmistoprojektin johtoa ja hallintaa tarvitaan, koska ammattimainen ohjelmiston tuotanto tapahtuu aina tietyn aikataulun ja budjetin mukaan, tuotanto-organisaation asettamien rajoitusten määrittelemänä. Ohjelmistoprojektia ei johdeta samoin kuin muiden tuotannollisten alojen projekteja.

Syitä tähän ovat mm. se, että tuote ei ole konkreettinen, edistystä vaikea nähdä, tuote on loppuun saakka muunneltavissa, monet ohjelmistoprojektit ovat kertaluonteisia ja/tai tuotantoprosessi ei ole standardoitunut.



Kuva 1: virheiden alkuperäinen korjausprosessi

Kuva 2: uusi virheen korjausprosessi



Case-esimerkki: Kronodoc Oy:n testausprosessin parantaminen dokumentinhallinnan keinoin

Kronodoc Oy:ssä tehtiin vuonna 2002 mittaus, jolla pyrittiin selvittämään vanhan ja uuden tuoteversion testausprosessin tehokkuus. Lähtötilanteena oli, että oli olemassa kokonaisprojekti, jonka osana oli testauksessa havaittujen virheiden korjausprosessi. Tavoitteena oli olemassa olevan prosessin tehokkuuden ja tuotteen laadun parantaminen sekä todellisten kustannusten selvittäminen ja kustannussäästöjen aikaan saaminen.

Ohjelmiston tuotekehitysprosessi, -vaiheistus sekä vaiheistuksiin kuuluvat tehtävät oli viety dokumentinhallintajärjestelmään ja tehtäville oli annettu tietyt metatiedot, joiden kautta mittaaminen oli mahdollista. Tuotekehityksen projektinhallintatyökaluna käytettiin Kronodoc-dokumentinhallintaratkaisua, johon on sisään rakennettu mm. dokumentaation etenemisen mittaamiseen käytettäviä työkaluja ja raportointia helpottavia ominaisuuksia. Kuva 1 kuvaa alkuperäistä virheiden korjausprosessi.

Lähtötilanne oli, että yhden virheenkorjauksen läpimenoajaksi oli mitattu 34 päivää ja kustannukseksi 415 EUR.

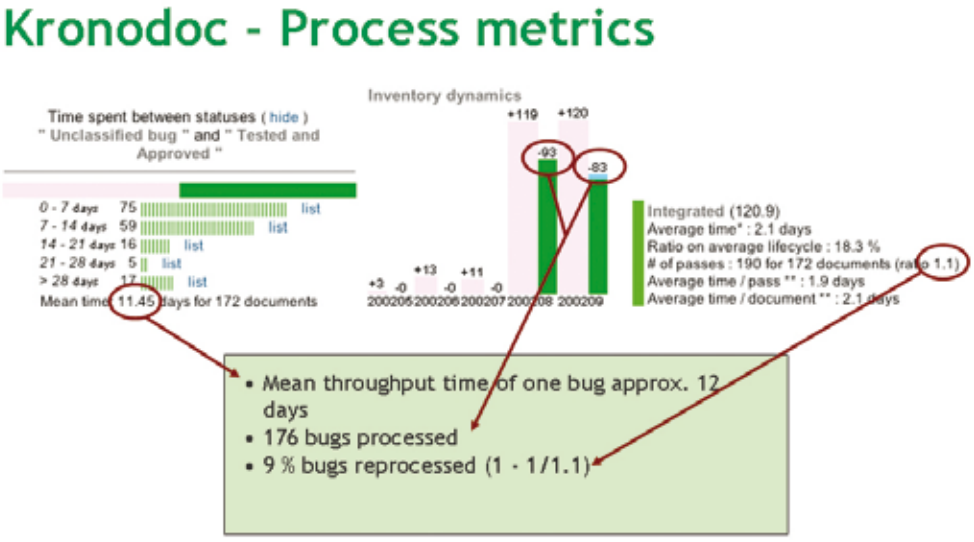
Miettimällä virheen korjausprosessi (kuva 2), virheiden kuvaus ja luokittelu uudelleen sekä kohdistamalla tietyt tehtävät nimetyille henkilöille saatiin aikaiseksi positiivisia tuloksia. Tuloksilla oli nopeasti vaikutusta yksittäisten työntekijöiden työtyytyväisyyteen ja virheen korjauksen laatuun sekä virheiden korjauksen läpimenoaikojen ja kustannusten pienenemiseen.

Vaikka virheen korjausprosessi itse prosessina pidentyi ja korjausprosessiin osallistuvien henkilöiden määrä pysyi samana, niin läpimenoajat pienentyivät oleellisesti. Laadun ja tuottavuuden parantuminen näkyi nopeasti siinä, että yksittäisen virheen korjaamiseen meni nyt keskimäärin 12 päivää. Virheet korjattiin siis noin 3 kertaa nopeammin kuin aikaisemmin ja vain noin 9 % virheistä toistui. Yksittäisen virheen kustannus saatiin pudotettua aikaisemmasta 415 Eurosta 167 Euroon.

Itse tuotekehitysprojektin budjetissa tämä laadun ja tuottavuuden nosto näkyi selvästi. Yksittäisen työntekijän tuottavuuden kasvu oli 150 % ja virheiden korjausaika putosi 30 %:iin aikaisemmasta. Kokonaishyöty joka tällä muutoksella saavutettiin, oli tässä yksittäisessä tuotekehitysprojektissa noin 67 000 Euroa.

Se, miten tähän päästiin, vaati prosessin kehittämistä. Kehitystyön kautta saatiin kokonaisuudet pilkottua oikean kokoisiksi paloiksi. Tehokkaalla työkalulla, strukturoidulla projektimallilla ja oikean tiedon keräämisellä tätä muutosta voitiin tukea. Samalla tiedon jakoa ja kohdentamista parannettiin. Dokumentinhallintajärjestelmässä dokumenttien tilasiirtymien kautta voitiin viestiä

Kuva 3: esimerkki raportista, jolla seurataan virhemäärien ja tilasiirtymien kehittymistä.



eri osapuolille tehtävien valmiusastetta ja jakaa tehtäviä, havainnoida pullonkauloja sekä mitata läpimenoaikoja ja kustannuksia.

Tuotekehitysprojektin vetäjän osalta projektin virheen korjausprosessi oli koko aika seurattavissa ja siten aikataulu- ja resurssivaateen ennustettavissa. Dokumentinhallintajärjestelmän kautta tuotekehityspäällikkö pystyi reaaliaikaisesti ajamaan raportteja ja seuraamaan virhemäärien ja dokumenttien tilasiirtymien kehittymistä. Samalla vastaavat raportit mahdollistivat tehokkaan kommunikoinnin sekä projektin sponsorin että virheen korjausta tekevien kehittäjien suuntaan. Kuvassa 3 on esimerkki tällaisesta näkymästä.

Oikea tietoa oikeille ihmisille oikeaan aikaan

Edellä kuvatun kaltaista parannusta mietitään useissa organisaatioissa ja yksittäisissä projekteissakin kaiken aikaa. Projektien tehokkuuden ja laadun kehittäminen ei ole suoraviivainen prosessi, vaan vaatii projektiorganisaatiolta usein vuosien työn ja tiettyjen asioiden läpikäynnin. Hajautetuissa ja globaaleissa projektiorganisaatiossa tämä haaste moninkertaistuu.

Projektitiedon ja siihen liittyvän dokumentaation hallintaan liittyvät asiat antavat kuitenkin perustan tehokkaalle projektien ja projektiosapuolten hallinnalla. Tiedon pitää olla yhdessä paikassa, se pitää saada oikeille ihmisille ja vieläpä oikeaan aikaa. Kuulostaa helpolta, mutta monissa projekteissa tämä on edelleen suurin haaste. Vasta kun itse projektin hallinta on kunnossa, voidaan aloittaa projektien laadun ja tehokkuuden parantaminen. Viimeisenä askeleena näen projektisalkun ja riskien hallinnan, joka on monessa organisaatiossa vielä vain haaveuni.

Kuva 4 kuvaa edellä mainitsemaani projektitoiminnan kehityskaarta ja niitä askeleita, joita organisaation pitää ottaa ennen kuin mittaaminen ja tehokas projektitoiminnan parantaminen onnistuu.

Kronodoc Oy on suomalainen informaatiologistiikan ratkaisutoimittaja. Informaatiologistiikalla tarkoitetaan tiedon hallintaa, yhteisöllisyyttä ja prosessien hallintaa yritysten sisäisessä ja välisessä arvoverkossa. Informaatiologistiikka tuo ratkaisun vähentämällä projektien laatu- ja tietotekniikkakustannuksia, tehostamalla yksilön työskentelyä ja informaation jakelua sekä parantamalla toimitusten tarkkuutta.

Kuva 4: projektitoiminnan kehityskaari ja askeleet mittaamisen ja parantamisen onnistumiseksi.

Teksti: Timo Koivisto

Timo Koivisto, tj., Stratura Group Oy, on työskennellyt tietojärjestelmien, asiakasprojektien ja tuotekehityksen parissa yli 30-vuotta. Tällä hetkellä hän toimii konsulttina ICT:n, liiketoiminnan kehittämisen, tuotteistamisen ja strategian toimeenpanon parissa. Yrityksellä on menossa strategian toimeenpanoon ja mittaukseen liittyvä tuotekehityshanke.

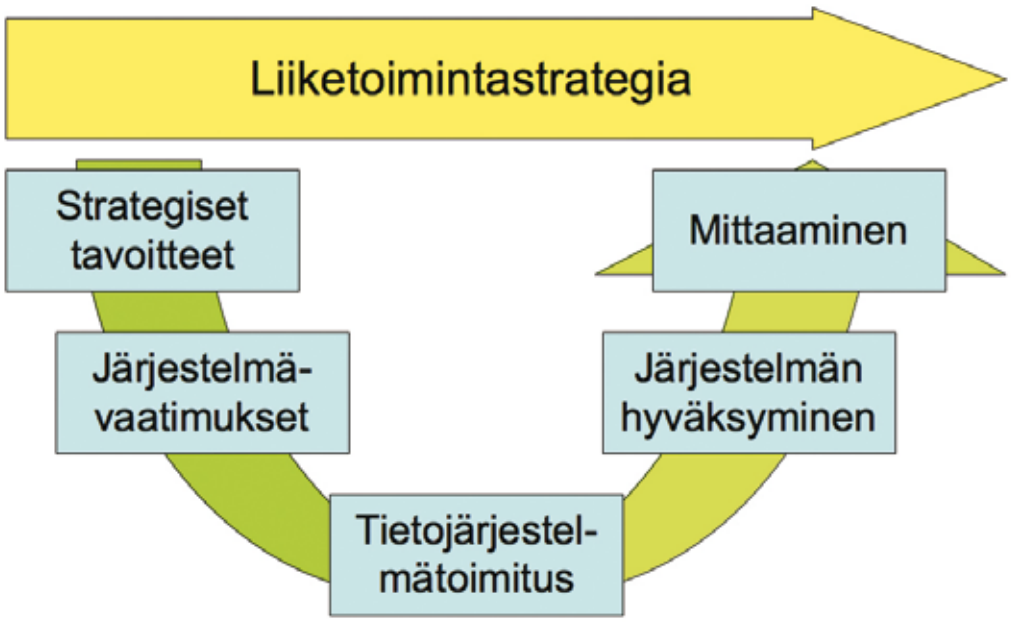
Tietojärjestelmätoimituksen laadun jälkiseuranta

Tietojärjestelmätoimitus, tapahtuu se yrityksen omin tai ulkoisen toimittajan resurssein, on aina seurausta harkinnasta, joka on johtanut hankkeen investointipäätökseen. Tietojärjestelmätoimitus liittyy yrityksen liiketoiminnan kehittämiseen ja toteuttaa yrityksen selkeästi määriteltyjä strategisia tavoitteita ja palvelee yrityksen vision toteuttamista. Tietotekniikkahankkeen on liityttävä yrityksen elinvoiman ja kilpailukyvyn parantamiseen.

Kaukana takana ovat toivottavasti ne ajat, jolloin tietohallinto omassa keskuudessaan käynnisteli tietojärjestelmähankkeita. Jotta tietohallinto ja ulkopuoliset toimittajat voivat toimittaa laadukkaita järjestelmiä, tulee heidän nähdä se kokonaisuus, mihin kukin vähintäänkin merkittävä tietojärjestelmätoimitus liittyy ja mitä tavoitteita sille asetetaan. Liittyykö hanke sisäiseen kehittämiseen, kilpailukyvyn parantamiseen vai markkinaosuuden kasvattamiseen?

Strategian toteutuksen V-malli

Useimmille tietojärjestelmien rakentamisen kanssa tekemisissä oleville henkilöille tietojärjestelmätoimituksen V-malli on tuttu. Siinä määritellään toteutuksen eri tasot, kuten määrittely, suunnittelu ja toteutus sekä näihin liittyvät testaus- ja hyväksymiskriteerit. Oheisessa kuvassa tätä mallia on jatkettu ylöspäin ottamatta kantaa siihen, miten itse järjestelmätoimitus toteutetaan.



Kuva: Tietojärjestelmä strategian toteutuksessa

Vaatimukset tietojärjestelmälle

Tietojärjestelmätoimitus perustuu strategisista tavoitteista johdettuihin järjestelmävaatimuksiin ja projektin määrittelyyn. Näiden ymmärtäminen oikein on onnistuneen järjestelmätoimituksen elinehto. Oikeat vaatimukset taas saadaan riittävälle tasolle konkretisoiduista strategisista tavoitteista.

Tietojärjestelmää määritellessä eivät esimerkiksi ympäripyöreät kasvuvaatimukset riitä, vaan tavoitteiden tulee olla sillä tavoin konkreettisia, että niiden toteutumista on mahdollista mitata. Unohtaa ei toki pidä yrityksen tietohallintostrategiaa ja yrityksen sovellusarkkitehtuuria, jotka ohjaavat toteutusta merkittävässä määrin.

Strategian viestintä

Projektin määrittelyn ja sille tulevien vaatimusten tulee olla oikein ymmärrettyjä. Varsinkin ulkopuolinen toimittaja joutuu varsin usein olemaan vain teknisenä toteuttajana. Tällöin osa toimittajan asiantuntemuksesta jää helposti käyttämättä. Strategian toimeenpanon onnistumisessa viestintä, delegointi ja palautekäsittely ovat keskeisessä asemassa. Näin ollen strategian sisältöön ja tavoitteisiin liittyvät asiat tulee viestiä myös ulkopuolisille toimittajille samassa yhteydessä kun muillekin tietojärjestelmätoimituksen projektiorganisaatiossa toimiville henkilöille – salassapitosopimuksia unohtamatta.

Projektin toimittajan vastuu

Projektin toimittaja voi vastata toimituksesta vain siinä laajuudessa, kun tietoa on saatavilla. Huonosti määritelty projekti luo itseään toteuttavan ennusteen eikä projektilla ole muuta mahdollisuutta kuin epäonnistua.

Usein puhutaan virheen iästä. Mitä aikaisemmin virhe on tapahtunut, vaikkapa suunnitteluvaiheessa, sitä kalliimmaksi sen korjaaminen toteutusvaiheessa tai sen jälkeen tulee. Tätä ajatusta voi viedä vielä pidemmälle taaksepäin aina projektimäärittelyyn saakka. Kalleimpia virheinvestointeja ovat ne projektit, jotka eivät vastaa yrityksen strategisia tavoitteita.

Toimittajan vastuun rajaaminen

Mikä on, tai oikeastaan voi olla, projektin sisäisen tai ulkoisen toimittajan vastuu siinä vaiheessa, kun arvioidaan toteutetun tietojärjestelmän strategian mukaista toteutumista. Jos tilaaja ei ole voinut tai kyennyt määrittelemään projektia siten, että tavoitteet voisivat toteutua, kenen pää laitetaan giljotiiniin. Viisas tietojärjestelmätoimittaja huolehtii ainakin siitä, että sopimuksissa vastuu rajataan projektin määrittelyyn ja asiakasvaatimuksiin.

Konsultoinnin hyödyntäminen

Konsultointi voisi olla merkittävä lisä matkalla parempiin, yrityksen tavoitteita varmemmin tukeviin järjestelmätoteutuksiin. Varsinkin ulkopuolisella järjestelmätoimittajalla voi olla varsin kattava kokemus tyypillisten liiketoiminnan kehittämishankkeiden toteuttamisesta. On siis hyväksi kuunnella kokenutta projektin vetäjää jo projektimäärittelyn alkuvaiheessa.

Ilman erillistä sopimusta ja korvausta ei konsultointia kuitenkaan pidä tilaajana odottaa. Tähän vaiheeseen sijoitetut muutamat tuhatlappuset saattavat tuottaa myöhemmin merkittävän hyödyn.

Monitoimittajaprojektit

Monitoimittajaprojektissa tilanne on tuskin helpompi. Projektin toimituksesta päävastuussa oleva projektipäällikkö muuttuu tiedon vastaanottajasta tiedon tarjoajaksi muille toimittajaosapuolille.

Kaikkein teknisimpiä suorituksia lukuun ottamatta koko järjestelmän toimituskonsortion on hyvä tuntea ne tavoitteet, joihin kehitettävän tietojärjestelmän avulla pyritään. Vähintäänkin oman nahkansa pelastamiseksi päävastuullisen järjestelmätoimittajan on vaikka väkisin kaivettava esiin riittävät tiedot järjestelmän takana olevista todellisista tavoitteista.

Panostaminen vaatimusmäärittelyyn

Toimittajan asiantuntemus mitataan siinä, miten hän onnistuu kaivamaan esiin projektin oleellisen sisällön ja rakentamaan niistä oikeat vaatimukset. Vaatimusten tulee olla sellaisia, että niiden toteuttaminen voidaan testausvaiheessa todentaa ja mitata. Vaikka jotkut vaatimukset ovat luonteeltaan hyvinkin teknisiä, niiden oikealla toteutuksella voidaan aidosti vaikuttaa strategisten tavoitteiden toteutumiseen ja liiketoiminnan kehittämiseen.

Vaatimusten sparraus tulee jo alkuvaiheessa ulottaa tietohallinnon ohi liiketoiminnan tasolle niille henkilöille, jotka viime kädessä ovat vastuussa tavoitteiden toteutumisesta.

Onnistunut projekti

Projekti onnistui täydellisesti. Se pysyi aikataulussa, se pysyi budjetissa, resurssien käytön kanssa ei ollut ongelmia, järjestelmätestaus ei paljastanut puutteita toimituksessa, tilaaja maksoi laskun – ja kaikki osapuolet ovat tyytyväisiä. Vain muutamassa prosentissa tietojärjestelmäprojekteja päästään tähän. Vaikka päästäisiinkin, riittääkö se järjestelmän tilaajalle.

Käyttöönotto

Ensimmäinen merkittävä tositesti on kehitetyn järjestelmän käyttöönoton onnistuminen. Ottamatta kantaa siihen pitääkö käyttöönotto suorittaa erillisessä toteutusprojektia seuraavassa projektissa, on sen onnistuminen tyypillisesti osa toimittajan vastuuta. Projekti kuitenkin päättyy jossain

vaiheessa. Miten käytännössä voidaan olla selvillä siitä onko käyttöönotto todellisuudessa onnistunut vuoden päästä.

Järjestelmän käyttö

Vielä vähemmän toimittaja voi olla selvillä siitä, miten järjestelmän todellinen käyttö on lähtenyt liikkeelle kolmen vuoden päästä. Saavuttaako uusi järjestelmä sen roolin, mikä sille oli ajateltu. Tuliko siitä se pelastava järjestelmä, jonka avulla markkinaosuus Saksassa kaksinkertaistui, jos tämä oli se tavoite.

Järjestelmän toteutusprojekti on mennyt hukkaan, mikäli järjestelmän käyttöönottoa ei toteuteta huolella. Järjestelmän käyttöönottoprojektin voidaan katsoa epäonnistuneen, mikäli toteutettavaa järjestelmää ei tulla käyttämään suunnitellulla tavalla.

Kaiken kaikkiaan koko hankkeen onnistuminen riippuu siitä, miten hyvin se toteuttaa yrityksen strategisia tavoitteita, miten hyvin se viime kädessä auttaa yrityksen kilpailuaseman vahvistamisessa ja vision toteuttamisessa.

Onnistumisen mittaaminen

Järjestelmätoimituksen onnistumisen mittaukseen liiketoiminnan kannalta ei riitä pelkkä järjestelmätestaus. Mittausta tulee laajentaa siten, että projektin jälkeen tehdään koko projektiryhmää koskeva kartoitus, jonka avulla toimituksen eri osapuolilta, sekä tilaajalta että toimittajilta, kerätään palautetta ja näkemyksiä järjestelmätoimitukseen liittyen.

Kartoituksessa selvitettäviä asioita

- miten järjestelmätoimitus vastasi kunkin vastaajan näkemyksen mukaan annettuja tavoitteita
- oliko kukin tietoinen siitä kokonaisuudesta, mihin toimitus liittyi
- kykenikö projektipäällikkö vastaamaan projektin vaatimuksiin
- toimiko toimituskonsortio projektin tavoitteiden mukaisesti
- onnistuivatko osatoimitukset ja kykenivätkö alihankkijat toimimaan odotetulla tavalla.

Selvitys voidaan toteuttaa useilla eri menetelmillä. Esimerkiksi Tekesin tuella toteutetussa Laatutoimitusprojektissa päädyttiin 360-tyyppiseen mittausmenetelmään, jossa eri osapuolet arvioivat toistensa onnistumista projektissa näkemyskartoituksen avulla.

Kattava kartoitus

Kartoituksen avulla saadaan esisijaisesti tietoa toimituskonsortion toiminnasta projektin aikana. Selvitettäviä asioita ovat oheisessa infolaatikossa kuvatut tekijät.

Näkemyskartoituksella voidaan koota laajasti tietoa järjestelmätoimituksen onnistumisesta, sillä projektin vaikuttavuuden arviointi pelkästään lukujen valossa on varsin vaikeata.

Pitkän aikavälin vaikutukset

Todelliset vaikutukset nähdään mahdollisesti vasta vuosien päästä. Myöskin järjestelmän odotettavissa olevan elinkaaren aikaisia vaikutuksia on äärettömän vaikea arvioida projektin valmistumishetkellä. Tyypillistä on esimerkiksi, että toiminnanohjausjärjestelmän käyttöönotto ei alkuvaiheessa suinkaan paranna yrityksen toimintakykyä, mutta edut konkretisoituvat muutaman vuoden käytön jälkeen.

Avainhenkilöiden näkemykset hyötykäyttöön

Yrityksen avainhenkilöt ovat parhaassa asemassa arvioimaan nähtävissä olevia tuloksia oman ammattitaitonsa puitteissa. Tilaajan puolelta kohderyhmän tulee kattaa myös projektiryhmän ja tietohallinnon ulkopuolisia avainhenkilöitä, eli niitä, joiden vastuulla kyseessä olevien strategisten tavoitteiden toteutuminen on. Samassa asemassa voi olla myös toimittajan edustaja, mikäli hänellä on ollut projektissa riittävän keskeinen mandaatti ja hän on voinut nähdä järjestelmätoimituksen vaikutukset liiketoiminnan kannalta.

Tietojärjestelmän toimituksen voidaan katsoa onnistuneen, kun mittauksella ja kartoituksilla voidaan todeta, että tietojärjestelmä osaltaan toteuttaa olemassa olevia strategisia tavoitteita.

Teksti: Perttu Karvinen

Projekteille tuottavuutta ja laatua reaaliaikaisella projektimonitorilla

Perttu Karvinen, projektipäällikkö ja ICT-arkkitehti, toimii CodeBakers Oy:ssä konsulttina.
perttu.karvinen@codebakers.fi

CodeBakers Oy on yksityisesti omistettu ICT-palveluiden tuottamiseen erikoistunut yritys. CodeBakers toimii pääosin kolmella osa-alueella: asiakaskohtainen järjestelmäkehitys ja konsultointi, menetelmäkonsultointi ja asiakaskohtainen (ulkoistettu) ohjelmistotuotekehitys.

Usein esitetään kysymykset, että tehdäänkö yksittäisissä projekteissa oikeita asioita ja tehdäänkö organisaatiossa juuri niitä projekteja, jotka hyödyttävät liiketoimintaa parhaiten ja ovat strategian mukaisia. Lisäksi voidaan kysyä, että perustuuko vastaus faktatietoon. Jos vastaus edellisiin on kyllä, niin tilannne vaikuttaa oikein hyvältä. Usein kuitenkin johonkin edellisistä kysymyksistä joudutaan vastaamaan kielteisesti ja toteamaan että parannettavaa kyllä löytyy.

Tässä artikkelissa pohditaan miten tilannetta voidaan tehostaa ottamalla käyttöön monipuolinen ja reaaliaikainen ohjelmisto, jolla voidaan hallita ja analysoida projektisalkkuja, suunnitella yksittäisiä projekteja sekä seurata projektien etenemistä.

Projektit

Useilla organisaatioilla on käytössään hyvin kuvatut prosessit ja menetelmät yksittäisten projektien läpiviennille, samoin kuin varsinaiselle sovelluskehitykselle. Tyypillisesti projektit toteutetaan käyttäen iteratiivisia ja inkrementaalisia menetelmiä, yhä useammin myös ketterämpiä menetelmiä. Ohjeistot ovat kattavia, eri vaiheisiin löytyy valmiita pohjia, tarkistuslistoja yms. Dokumentit ovat usein word –ja excel –dokumentteja jotka on talletettu levylle tai intranettiin. Kaikista näistä on paljon hyötyä ja ne ohjaavat eri projekteja tekemään asiat samalla tavalla.

Ongelmana usein kuitenkin on, että käytössä on paljon erilaisia ohjelmistoja, joilla dokumentteja käsitellään ja joista tietoa haetaan projektin käyttöön. Projektin tehtävät, kestot, resurssit, riippuvuudet ym. suunnitellaan yleensä jollain ohjelmistolla, josta saadaan tulostettua Gantt-kaavio. Toteumatiedot ovat usein jossain erillisessä tietojärjestelmässä, samoin henkilö- ja osaamistiedot. Tietoja joudutaan kopioimaan usein paikasta toiseen ja virheiden mahdollisuus kasvaa, puhumattakaan siitä paljonko aikaa kaikkeen edellämainittuun kuluu. Projektin tilanteen selvittäminen voi olla vaikeaa, koska ei ole yhtä paikkaa mistä sen voisi päätellä. Projektipäällikkö joutuukin joskus antamaan mutu-arvion projektin tilanteesta esim. projektin johtoryhmälle.

Edellä mainitut asiat eivät välttämättä ole iso ongelma, jos organisaatiossa ollaan kiinnostuneita vain yksittäisistä projekteista ja niiden läpiviennistä. Tilanne muuttuu ratkaisevasti jos organisaatio haluaa käsitellä, seurata ja analysoida useiden projektien muodostamia kokonaisuuksia eli projektisalkkuja.

Projektisalkut

Projektisalkun hallinnassa on kyse usean projektin samanaikaisesta tarkastelusta valituilla kriteereillä. Tällöin ollaan usein kiinnostuneita näkemään miten projektisalkun projektit pärjäävät verrattuna projektisalkun kriteereihin jotka liittyvät suoraan yrityksen liiketoiminta- tai kehitysstrategiaan. Voidaan tarkastella esim. projektien toteutuneita kustannuksia verrattuna suunniteltuihin kustannuksiin, projektien riskien kehittymistä, aikatauluja, resurssitarpeita ym. trendejä. Projektisalkun hallintaan liittyy myös menettelyt projektiehdotusten käsittelyyn ja arviointiin.

Tarkastelemalla yksittäisten projektien sijasta projektisalkkuja organisaatio saa paremman kuvan siitä miten käynnissä olevat projektit toteuttavat organisaation strategiaa ja että tarvitseeko tiettyjen projektien tilaa tai prioriteettia muuttaa jotta tavoitteisiin päästäisiin. Samoin voidaan tarkastella tulevia resurssi- ym. tarpeita. What-If tarkasteluilla voidaan etukäteen arvioida miten tietyt muutokset vaikuttaisivat kokonaisuuteen.

Tarkasteluissa saatetaan havaita, että tietyt projektit käyttäytyvät tavalla joka saattaa johtaa projektien epäonnistumiseen. Esimerkiksi jokin projekti voi varata organisaation kriittiset resurssit liian pitkäksi aikaa, eikä muita tärkeitä projekteja voida tämän vuoksi käynnistää. Voidaan myös havaita, että jossakin projektissa on hyvin paljon muutoksia ja että projekti ei edisty suunnitelmien mukaisesti.

Jotta tällaista tarkastelua voitaisiin tehdä, tulee organisaation kerätä projekteista yhteismitallista tietoa ja tiedon tulisi olla tosiaikaista. Tällaisen tiedon kerääminen ja käsittely ilman keskitettyä tietovarastoa ja ohjelmistoa voi olla hyvinkin työlästä ja aikaa vievää. Varsinkin jos organisaatiossa on käynnissä kymmeniä tai jopa satoja projekteja samaan aikaan.

Prosessien kehittäminen

Edellä mainittuihin haasteisiin voidaan löytää helpotusta monilla tavoilla, mm. ohjelmistoapuneuvoilla. Ennen kuin edes aletaan miettiä ohjelmistojen hankkimista, tulee organisaatiolla olla visio siitä mihin suuntaan projektinhallinnan prosesseja tulisi kehittää ja mitä kehittämisellä haetaan. Avainasemassa ovat sellaisten mittareiden löytäminen, joilla projektien luokittelua ja arviointia voidaan tehdä strategian mukaisesti.

Ensimmäinen vaihe prosessien kehittämisessä on nykyisten prosessien ja käytäntöjen arviointi. Seuraavaksi kartoitetaan tavoitetilan vaatimukset ja prosessit. Kehittämishanke etenee kuten muissakin prosessien kehittämishankkeissa. Sitten kun on selvillä se miten prosesseja ja käytäntöjä tulisi kehittää, voidaan miettiä olisiko ohjelmistoista apua.

Edellämainittu prosessien kehittämishanke on yleensä vaativa ja aikaa vievä eikä siinä kannata hätiköidä. Mahdollinen ohjelmiston käyttöönotto kannattaa tehdä osissa ja keskittyä aluksi sellaisiin osiin, joista on helpointa saada välittömiä hyötyjä. Keskeisiä huomioon otettavia asioita on johdon ja projektitoimintaan osallistuvien ihmisten sitouttaminen ja kehittämiseen mukaan ottaminen alusta alkaen.

Ratkaisuvaihtoehtoja

Markkinoilta löytyy useita ohjelmistoja, joissa on sekä projektinhallinnan että projektisalkunhallinnan ominaisuuksia, mm. Artemis, CA Clarity, IBM Rational Portfolio Manager, Mercury, Microsoft Project, Planview, Primavera. Osa edellämainituista koostuu useasta erillisestä ohjelmistosta, jotka on integroitu keskenään, osa sisältää kaiken toiminnallisuuden yhdessä ohjelmistossa. Lähes kaikissa on tavalla tai toisella mm. seuraavia toimintoja:

projektien suunnittelu ja seuranta,
projektin budjetin hallinta,
resurssien hallinta,
projektisalkkujen hallinta ja analysointi,
raportit, investointikartat, OLAP pivotit,
tuloskortit, what-if tarkastelut,
prosessien ja työnkulkujen hallintamenetelmät,
kommunikointi, hälytykset,
dokumenttien ja versioiden hallinta,
konfiguroitavuus omiin tarpeisiin, ja
integroitavuus muihin ohjelmistoihin.

Osa ohjelmistoista on vahvoilla projektien suunnittelussa, osa projektien analysoinnissa.

Esimerkkinä Rational Portfolio Manager

Rational Portfolio Managerin (RPM) avulla voidaan toteuttaa tosiaikainen ja avoin projektiympäristö, jolla voidaan suunnitella, hallinnoida ja seurata yksittäisiä projekteja, ohjelmia ja projektisalkkuja sekä resursseja ja kustannuksia. Eri osapuolet saavat monipuolisilla analysointivälineillä sekä lukuisilla valmiilla raporteilla totuudenmukaisen ja ajantasaisen kuvan käynnissä olevista projekteista ja ongelmakohtiin voidaan puuttua välittömästi.

Kuva: RPM:n toiminnallisuuksia

RPM:ään voidaan konfiguroida organisaation prosessien mukaisia työnkulkuja, jotka ohjaavat eri osapuolten toimintaa prosessin eri vaiheissa. Työnkuluilla voidaan toteuttaa mm. eri vaiheisiin liittyvät hyväksymismenettelyt ja katselmoinnit. Eri tyyppisille projekteille voidaan tehdä projektipohjia, joissa on valmiiksi rakennetut wbs-rakenteet, resurssiprofiilit, kustannukset, dokumenttipohjat yms. RPM:n käyttöliittymä voidaan konfiguroida hyvin monipuolisesti.

RPM:n resurssienhallintaominaisuudet ovat monipuoliset sisältäen mm. osaamis- ja kyvykkyystietojen hyödyntämisen projektin suunnittelussa ja resurssien kiinnittämisessä projektin tehtäviin. Resurssien analysointiominaisuuksilla saadaan selville nykyiset kuormitustilanteet sekä tulevat tarpeet ja näiden perusteella voidaan tehdä päätöksiä lisäresurssien hankinnasta.

Loppusanat

Johdannossa esitettyihin haasteisiin ei voida vastata pelkästään ottamalla käyttöön ohjelmistoapuvälineitä. Ohjelmiston käyttöönottoa edeltää aina nykytilanteen kartoitusprojekti jossa nähdään prosesseihin tarvittavat muutokset. Laatu- ja tuottavuushyötyjä voidaan saavuttaa vasta sitten kun organisaation investointi, projektinhallinta ja sovelluskehitysprosessit on kuvattu ja käytössä. Tämän jälkeen ohjelmistoilla voidaan tehostaa projektitoimintaa.

Teksti: Pekka Forselius

Toimintopistelaskenta tuottavuuden todentajana

- tiedon ja menetelmien avulla ulos ”projektihelvetistä”

Pekka Forselius
Software Technology Transfer Finland Oy

Kirjoittaja on STTF Oy:n toimitusjohtaja, toimii Senior Advisor-nimikkeellä FiSMA ry:n projektijohtamisen jaostossa ja edustaa Suomea mm. ISO-standardoinnissa sekä ISBSG:ssä, jonka Vice President hän on nykyisin. Hän on toiminut 2000-luvun alkuvuosina Sytyke ry:n hallituksessa, sen puheenjohtajana ja myös TTL:n hallituksen jäsenenä.

Järkyttävää? Ohjelmistoprojekti myöhästyi kolme kuukautta ja muutenkin kalliilta tuntunut työ maksoi 155 000 euroa enemmän kuin alunperin oli arvioitu. Ja ohjelmistosta löytyi ensimmäisen puolen vuoden aikana 42 virhettä.

Joko ohjelmiston toimittaja tai ostaja – mutta yleensä molemmat – tuntevat itsensä petetyiksi, kun projektille arvioitu työmäärä ylittyy ja aikataulu venyy. Ja näin käy aika usein, ainakin jos ostajilta kysytään. Toimittaja syyttää tilaajaa uusien vaatimuksien esittämisestä ja vanhojen muuttamisesta, ja tilaaja toimittajaa ammattitaidottomuudesta. Projektissa mukana olevat kokevat huonoa omaatuntoa, ja syyttävät projektipäällikköä tai toisiaan, eivätkä asiat aina ole hyvin kotirintamallakaan. Pitäähän sen – perheenjäsenten mielestä ”heitä tärkeämmän” – projektin takia olla iltoja ja viikonloppuja poissa jatkuvasti.

Jotta edellä kuvattu, kaikkien kannalta ikävä tilanne ei toistuisi uudelleen ja uudelleen, meidän pitää oppia suhteuttamaan asiat. Vaikka projektin myöhästyminen, hinta ja virheiden määrä voivat kuulostaa liian suurilta ensi kuulemalta, kannattaa sekä projektin tuottavuutta että laatua tarkastella kylmän analyyttisesti. Onko kolme kuukautta pitkä aika? Onko 155 000 euroa paljon rahaa? Onko 42 paljastunutta virhettä puolessa vuodessa liikaa? No, se riippuu kaikkein eniten siitä kuinka suuri projektissa toimitettu ohjelmisto on.

Ohjelmiston toiminnallisen laajuuden voi mitata monellakin eri tavalla – mutta käytetyin ja ainoa standardoitu menetelmä on toimintopistelaskenta. Täsmällisesti puhuen kyseessä on ISO/IEC 14143-1 Functional Size Measurement standardin mukainen laajuuden mittaaminen, jonka voi tehdä millä tahansa viidestä tarjolla olevasta menetelmästä. Yleisesti näitä menetelmiä kuitenkin kutsutaan toimintopistelaskennoiksi (FPA), ja laajuuden mittayksikköä toimintopisteeksi (fp).

> Toimintopiste on ohjelmiston toiminnallisen laajuuden mittayksikkö. Mitä enemmän erilaisia toiminnallisia palveluja ohjelmisto tarjoaa, sitä laajempi se on, ja sitä enemmän sen toteuttaminen yleensä vaatii työtä.

Toimintopiste on siis ohjelmiston koon mittayksikkö, joka on täysin riippumaton käytetystä teknologiasta, kehittämisvälineistä ja ohjelmiston tyypistä. Ei ole olemassa sellaista ”softapalikkaa” (”piece of software”), jonka laajuutta ei voi mitata toimintopisteinä.

Samaa ei voi sanoa mistään muista kehitetyistä mittausmenetelmistä, kuten Use Case Point tai Object Point, puhumattakaan lähdeohjelmien rivimääristä (SLOC). Ne kaikki tosin soveltuvat jotenkin ohjelmistojen koon määrittämiseen, mutta ovat hyvin menetelmä- tai välineriippuvaisia. Pitkään ohjelmistotoimialaa seuranneet ja siinä mukana olleet tietävät että välineet ja menetelmät kehittyvät vuosien kuluessa, joten niihin sidotut mittarit eivät ole erityisen suositeltavia, etenkään pitkäjänteisen toiminnan kehittämisen perustaksi.

Toimintopisteet taas sopivat siihenkin erittäin hyvin. Jos vaikka vuonna 1960 kehitetty 500 toimintopisteen sovellus mitataan tänä päivänä, se on edelleen täsmälleen yhtä suuri, ellei sen toiminnallisuutta ole muutettu. Tämä siitäkin huolimatta että mitään samoja kehittämisvälineitä ei enää ole käytössä.

Toimintopisteetkö ohjelmistoprojektien ”Silver Bullet”?

Monesti olen tavannut ”uskovaisia”, jotka kuvittelevat jonkin uuden menetelmän tai teknologian ratkaisevan kaikkien ohjelmistoprojektien ongelmat. On vannottu strukturoidun ohjelmoinnin, APL:n, kehittimien, case-välineiden ja olioiden nimiin, samoin kuin toimintopistelaskennankin, mutta eiväthän ongelmat minnekään ole hävinneet. Eivätkä häviäkään, mutta voivat vähentyä huomattavasti kunhan arvioinnin teoriaa opitaan ymmärtämään paremmin. Toimintopistemäärä EI ole projektin koko. Se osoittaa vain projektissa kehitettävän ohjelmiston laajuuden. Neliömetrit EIVÄT yksinään määrää rakennuksen hintaa, mutta kertovat sen lattiapinta-alan riittävän tarkasti.

Toimintopisteitä EI voi laskea ennen ohjelmiston toiminnallisten vaatimusten määrittelemistä, mutta kylläkin huomattavasti ennen ohjelmiston toteuttamista. Samalla tavalla kuin rakennuksen

pinta-alankin voi mitata piirustuksista. Laskenta on myös suhteellisen helppoa tehdä paljonkin käyttöönoton jälkeen. Useimmat muut laajuuden mittaamiseen kaavaillut menetelmät soveltuvat käyttöön vain projektin aikana, eivätkä sielläkään kovin aikaisin. Mutta ettäkö ”hopealuoti”? Eihän toki.

Toimintopisteinä mitattu ohjelmiston laajuus on erittäin hyödyllinen lähtötieto sekä arvioitaessa projektin onnistumista jälkeenpäin että estimoitaessa työmäärää etukäteen, mutta kummassakaan tapauksessa se ei yksinään riitä. Oheinen kuva näyttää projektin työmäärän ja lopputuloksen suuruuden välisen riippuvuuden. Jälkimmäinen on vain yksi arviointiprosessin lähtötiedoista.

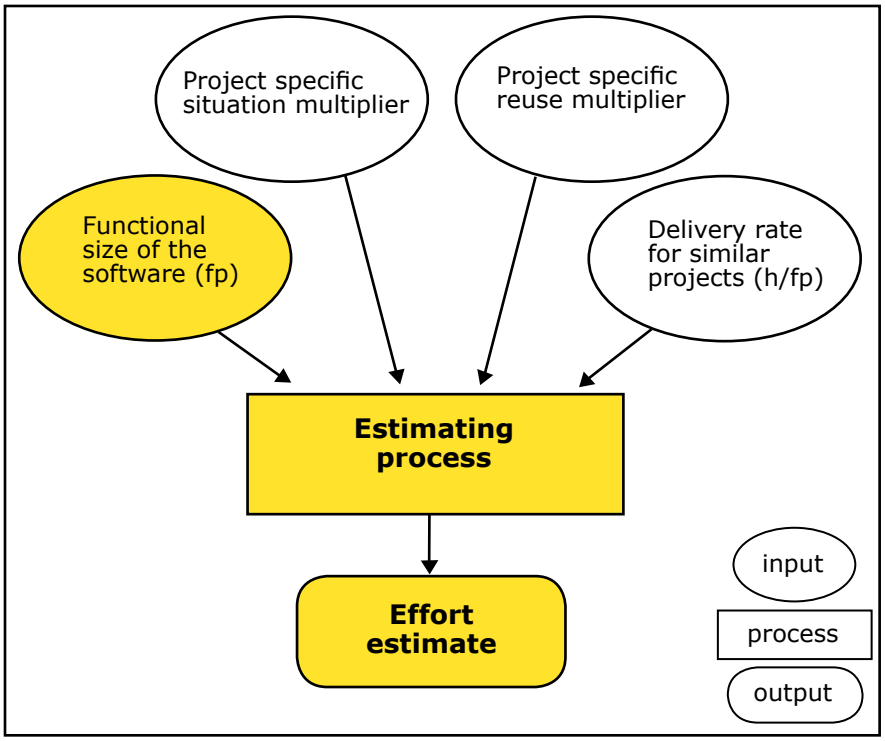
Yksinkertaisimman mallin mukaisesti työmäärän arviointi voidaan tehdä ohjelmiston laajuuden (fp) ja keskimääräisen tuottoasteen (h/fp) perusteella. Jos esimerkiksi 500 fp laajuinen ohjelmisto kehitetään tuottoasteella 5 h/fp, on kokonaistyömäärä 500*5 = 2500 h. Tätä karkeaa arvoa voidaan sitten täsmentää ottamalla huomioon kehittämisolosuhteet ja uudelleenkäytön vaikutus.

Toimintopistelaskenta ja siihen perustuva työmäärien arviointi on tunnettu jo lähes 30 vuotta. IBM:llä työskennellyt Allan J. Albrecht julkaisi ensimmäisen version FPA-menetelmästään vuonna 1979. Sen jälkeen menetelmä on kehittynyt ja siitä on tullut useita parannettuja variaatioita. Uusimpia menetelmiä kutsutaan jo toisen sukupolven menetelmiksi. Toiminnalliseen laajuuteen perustuva arviointi on tunnustettu yhdeksi ohjelmistoprojektien hallinnan huippukäytännöistä (best practice), mutta syystä tai toisesta sen hyödyntäminen on yhä aika vähäistä. Ilmeisesti työn tuottavuus ja arvioiden tarkkuus eivät olekaan kaikille tärkeitä?

Hyviä uutisia tiedon ja ymmärryksen lisääntymisestä!

Parhaat ohjelmistotoimittajat ovat toki hyödyntäneet toimintopisteitä jo vuosia. Ne ovat onnistuneet jopa keräämään itselleen riittävän kokemuskannan arviointiensa pohjaksi. Siitä ne löytävät kulloinkin sopivimman tuottoasteen (h/fp) laskelmiinsa. Myös yleisiä kokemuskantoja on saatu aikaan, sekä Suomessa että maailmalla. Finnish Software Measurement Association FiSMA ry:n jäsenistö aloitti kokemusten keräämisen n.s. Experience-kokemuskantaan jo 1990-luvun alkupuolella.

Samoihin aikoihin perustettiin myös International Software Benchmarking Standards Group eli ISBSG, jonka jäsenenä myös FiSMA nykyisin on. Näiden kahden organisaation kokoamissa kannoissa on yhteensä jo lähes 5000 projektia, joista jokaisesta on mitattu ohjelmiston laajuus toimintopisteinä sekä kehittämiseen käytetty työmäärä



Kuva: Lopputuloksen suuruus ja muut projektin työmäärään vaikuttavat tekijät

tunteina. Perusta toimintopisteiden laajamittaiselle hyödyntämiselle alkaa siis olla kunnossa.

Erityisen mielenkiintoinen ilmiö, joka on rantautumassa vihdoin myös meidän systeemityöläisten maailmaan, on yksikköpohjainen hinnoittelu. Kaikille on itsestään selvää että rakennus- tai kiinteistöjen huoltotyöt hinnoitellaan neliömetreittäin, samoin kuin moottoritiet tai muut väylät kilometreittäin. Miksei siis vastaava käytäntö voisi toimia meidänkin toimialallamme?

Kokemusten perusteella kiinteähintaisuus on osoittautunut huonosti toimivaksi malliksi ohjelmistotoimituksissa, mutta perinteinen tuntipohjainen laskutuskaan ei enää tyydytä - etenkään ohjelmistotyön ostajia. Toimintopisteitä on nyt opittu käyttämään hinnoittelun apuna eri puolilla maailmaa ja kokemukset €/fp-pohjaisesta sopimisesta ovat lähes yksinomaan hyviä. Australiassa ja Italiassa julkishallinnot ovat ottaneet käytännön omakseen, ja viimeksi Brasilia sekä Etelä-Korea ovat seuranneet esimerkkiä. Ensimmäiset askelet on nyt otettu myös Suomessa, eikä tältä tieltä taida olla paluuta. Tulokset sen kuitenkin viimekädessä ratkaisevat.

Lisää tietoa aiheesta osoitteista:
www.sttf.fi
www.fisma.fi
www.isbsg.org

(Kirjoitus on julkaistu aiemmin hieman lyhennettynä Mermit Oy:n asiakaslehdessä 2/2006)

KUUTAMOLLA

Tämä juttu on kirjoitettu kesälomalla. Se lieneekin sitten riittävän peitellysti verhottu anteeksipyyntö kaikista epäoleellisuuksista ja kummallisuuksista, joita juttu saattaa sisältää. Tilanne sinänsähän ei ole sen kummempi loman ulkopuolellakaan, silloin syy vain on joku muu. Joka tapauksessa, kesälomahan on monelle meistä yksi suurimmista henkilökohtaisista projekteista kalenterivuoden aikana. Mutta piipahdetaanpa välillä määrämuotoisen informaation puolelle. Koska siis tätä kirjoittaessani on kesäloma, olen harrastanut hieman mummogoogletusta (vai pitäisikö sanoa pappagoogletusta...mitenkään sukupuolista syrjintää miettimättä, onomatopoeettisesti olen enemmän mieltynyt mummogoogletus-sanaan), eli selailin ei-sähköisiä, kirjahyllystä löytyviä informaationjakeluvälineitä. Löysin Heikki Stenlundin kirjoittaman "Projektin ohjaus" -nimisen teoksen vuodelta 1986. Siinä tiivistetään projektin määritelmä seuraavasti: Projekti on mikä tahansa kokonaisuutena ohjattu, kertaluonteinen ja selvärakenteinen työsuoritus.

Jos määritelmän sanoja tulkitaan vapaasti, voidaan projektiksi kutsua lähes mitä tahansa toimintaa. Ja niin myös tehdään, joskus jopa kyllästymiseen asti, projektista on tullut trendisana. Teknisesti ottaen rajoittavin kriteeri on ehkäpä tuo kertaluonteisuus. Esimerkiksi joka juhannuksena toistuva savusauna-ohjelmanumero ei voine olla projekti, vaikka sillä on selvä tavoite, osallistujat, aikataulu sekä budjetti (kulutetut polttopuut ja lämmitysoluet, saunomisen aikana kulutettu vesi ja olut, saunomisen jälkeen kulutettu makkara, snapsit ja olut, seuraavan päivän aikana kulutettu burana ja vissy). Toisaalta taas kun ajattelee firman minkä tahansa järjestelmän tietyin väliajoin uudistavaa projektia, niin onko nekään nyt sitten niin kertaluonteisia? Jos asiakasjärjestelmä uusitaan, kyllähän siitä projekti tehdään. Ja jos se uusitaan, niin sehän tarkoittaa, että on jo joskus aikaisemmin tehty, ainakin kerran (oletettavasti useammin). Niin ja muuten, mistä sitä voi koskaan tietää, että jokin projekti on kertaluonteinen? Jos vaikka tehdään joku tyystin uusi järjestelmä, jota ei ole aikaisemmin tehty, niin mistä sitä tehtäessä voidaan olla varma, että se jää kertaluonteiseksi? Sehän tarkoittaisi, että lopputulos olisi laadultaan niin karmea, kallis ja hyödytön, että vastaavaan ei koskaan enää ryhdyttäisi. Se tuskin lienee projektin tarkoitus. Sellaisia projekteja tosin aika paljon on olemassa. "Ei koskaan enää juhannuksena moista savusaunomista", moisia projektin päätösmietelmiä saattaa useinkin kuulla. Varsinkin jos arvioitu burana-resurssien määrä tai seuraavan päivän tehokas toipumisaika on ylittynyt ja projektilaisten osallistuminen seuraaviin välittömässä lähitulevaisuudessa oleviin projekteihin on estynyt. Projektiriskit on saatettu arvioida liian optimistisesti ja ehkä kaikkia riskejä ei ole edes osattu miettiä. Ruumiinvammat, parantumattomat ongelmat parisuhteissa, savusaunan ja lähihehtaarien muuttuminen tuhkaksi ja projektilaisten häviäminen lähivesistöihin saattavat olla niitä katastrofeja, joita analysoitaessa saatettaisiin tulla siihen tulokseen, että projektin uusiminen (joka sinänsäkin on siis ristiriitainen termi, koska kertaluonteista asiaa ei siis voida uusia) päätetään jättää tekemättä. Tai oikeastaan pitäisi sanoa, että projekti päätetään jättää uusimatta. Uusimisen tekeminen on sanapari, joka varmasti saa kaikki fennolingvistit kananlihalle, joten sanotaan, että päätetään jättää uusimatta.

Kuitenkin, vaikka kuinka vakaasti päätetään olla uusimatta projektia (tuota muuten kutsutaan sanalla oxymoron, ristiriitaisten sanojen tai lauseiden yhdistämistä, esimerkiksi negatiivinen kasvu, rehellinen poliitikko, hauska pakinoitsija...), saattaa maailma muuttua siten, että projektin ROI ja riski/tuotto -suhde muuttuukin houkuttelevaksi. Useimmiten kyllä houkuttelevaisuus johtuu unohtamisesta ja muistojen kultautumisesta. "Oli se vaan hienoa viime juhannuksena istuksia höyryävänä savusaunan kuistilla virkistävä juoma kädessä ja nauttia kaikessa rauhassa kesäyön kuulaudesta ja käen kukunnasta." Itse asiassa vesisateessa kossupullo kourassa pihapuihin nokisena törmäily ja mökkinaapurin puolisoa kohtaan osoitettu leikkimielisaggressiivinen lähestyminen sekavia örveltäen toisten hoilatessa karaoke-laitteiden parissa oli jotain aivan muuta, mutta tikulla silmään sitä, joka vanhoja muistelee. Jos nyt kukaan yleensä mitään muistaakaan.

Mutta tuossakin, niin kuin kaikissa projekteissa, on tietenkin paljon kiinni projektipäälliköstä. Kunnon projektipäällikkö pitää huolen siitä, että mitään odottamatonta ei pääse tapahtumaan, kukaan ei käytä liikaa resursseja, tekee omat tehtävänsä aikataulussaan eikä sotkeudu toisten "työmaihin".

Nyt syksyllä on taas alkamassa tai jo alkanutkin eräs kulta-aika projektin aloittamisten suhteen. Niin työ- kuin yksityiselämässäkin. Kannattaa välillä muistella menneitäkin. Muistelu on hyvä tapa välttää pieniä ja vähän suurempiakin katastrofeja. Kannattaa myös harrastaa tiukkaa projektipäälliköintiä, arvioida riskit ja sitten vain antaa mennä. Onhan jokainen meistä oman elämänsä projektipäällikkö. Näin kornin lauseen jälkeen on kyllä pakko kokemuksen perusteella todeta, että elämä yllättää aina.

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

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

Lisätietoja SYTYKE ry:stä: www.sytyke.org

TOIMISTO

Puhelinvastaus- ja sihteeripalvelu VT Oy/ Susanna Koskinen
Systeemityöyhdistys Sytyke ry
Henrikintie 7 A
00370 Helsinki
p. 09 56075363
f. 09 56075365
sytyke@hennax.fi

Johtokunta 2006

Puheenjohtaja
Helena Venäläinen
p. +358 10252 3690
+358 50 568 6690
helena.venalainen@op.fi

Paula Miinalainen
p. +358 50 500 2363
paula.
miinalainen@arborvitae.fi

Markku Niemi
p. +358 50 512 4687
make.niemi@kolumbus.fi

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

Tarja Raussi
p. +358 50 548 1823
tarja.raussi@tieturi.fi

Jori Räty
p. +358 50 551 5152
jori.raty@sysopendigia.com

Kari Uusi-Äijö
p. +358 40 835 6541
kari.uusi-aijo@pohjola.fi

Varajäsenet

Lauri Laitinen
lauri.laitinen@nokia.com

Lea Virtanen
lea.virtanen@jobit.fi

Liittokokous-edustajat

Silja Räisänen
silja.raisanen@pohjola.fi

Jori Räty
jori.raty@sysopendigia.com

Helena Venäläinen
helena.venalainen@op.fi

Kati Viik
kati.viik@helsinkilainen.com

SYTYKE ry:n johtokunnan sähköpostilista:

hallitus@sytyke.ttlry.fi

Sytyttääkö? - Liity jäseneksi

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

Tietotekniikan liiton henkilöjäsenmaksu vuonna 2006 on alkaen 52€, erityisryhmien hinnoittelusta lisätietoja Tietotekniikan liitosta. Lisäjäsenyys maksaa 11€/yhdistys.

Osaamisyhteisöt

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

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

ProjektiOSY - PrOSY pyrkii yhdistämään Systeemityön projektitoiminnasta ja sen kehittämisestä kiinnostuneet, vetäjänä Markku Niemi.
make.niemi@kolumbus.fi

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

DAMA Finland keskittyy tiedon, informaation ja tietämyksen hallintaan. Suomen osaston johtoryhmän (boardin) vetäjänä Pekka Valta, yhteyshenkilönä Minna Oksanen.
minna.oksanen@gmail.com

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

OlioOSY kehittää, tukee ja voimistaa oliomallilähtöistä sovelluskehittämistä, vetäjänä Jukka Tamminen.
jukka.tamminen@pp.inet.fi

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

www.tieturi.fi/ohjelmistotuotanto

PROJEKTIN TUOTTAVUUS JA LAATU

Tuottavuudella viitataan suorituskykyyn ja taloudellisiin mittareihin. 1980-luvulta alkaen kilpailu on kasvattanut myös ei-taloudellisten mittareiden merkitystä. On alettu puhua laadusta. Tuottavuus ja laatu ovat tietojärjestelmäprojektien hyveitä, joita edistämme tuottavilla prosesseilla, laadukkailla parhailla käytännöillä ja tehokkailla työvälineillä. Tule ottamaan selvää miten tuottavuutta ja laatua tuetaan ohjelmistoprojekteissa nyt ja tulevaisuudessa!

Laatukäytännöt ohjelmistotuotannossa	25.-26.10.
ITIL Foundations	9.-11.10. ja 1.-3.11.
Projektityön perusteet	23.10. ja 6.11.
Projektin suunnittelu ja läpivienti	25.-27.10.[Hki] • 29.11.-1.12.[Tre]
Tietojärjestelmäprojektin suunnittelu ja läpivienti	25.-27.10.[Hki] • 29.11.-1.12.[Tre]
Taitavan projektipäällikön leadership-taidot	30.11-1.12.
Ketterä ohjelmistokehitys	12.-13.10.[Hki ja Tre]
Nykyaikainen systeemityö	12.10.
Tietojärjestelmän hallittu hankintaprosessi	23.10.[Hki ja Tku]
ISPL Foundations	11.-13.10.
Prosessien mallintaminen	6.11.
Järjestelmävaatimusten määrittely ja hallinta	16.-18.10[Hki ja Tku]
Oliot ja UML määrittelijälle	13.-14.11.
Tietojärjestelmän käyttöönotto	2.11.
Moderni tietojärjestelmäkehitys	22.11.
SOA ja Web Services -yleiskatsaus	9.10.
Yritys- ja integraatioarkkitehtuurin suunnittelu	16.-17.11.
.NET-arkkitehtuuri	10.11.
.NET-järjestelmän projektipäällikkövalmennus	16.-18.10.
Symbian OS for Project Managers	17.-18.10.
Java-järjestelmän projektipäällikkövalmennus	23.-25.10.

Ilmoittautumiset ja lisätiedot: puh. (09) 4315 5333
kurssit@tieturi.fi | www.tieturi.fi/syty

Liiketoiminta kehittyy – kehity sinäkin.