Tavoitekeskeinen ylläpidon mittaaminen asiakastyytyväisyyden parantamisessa

Henna Vallin

19.6.2007

Tietojenkäsittelytieteen laitos

Joensuun yliopisto

Pro gradu -tutkielma

Sisältö

TIIVISTELMÄ

Tässä tutkielmassa käsitellään tavoitekeskeistä mittaamista korjaavan ylläpidon näkökulmasta. Tutkielman pääpaino keskittyy Goal/Question/Metric-menetelmään, jota voidaan hyödyntää systemaattisten parantamistavoitteiden etsimisessä. GQM-menetelmää käsitellään sekä teoreettisesti, että käytännönläheisemmin kirjallisuudessa esitettyjen esimerkkien avulla. Lisäksi puhutaan ohjelmistoprosessin parantamisesta ja esitellään lyhyesti CMMI- ja SPICE-mallit, joita voidaan käyttää ohjelmistoprosessin tukena.

Tutkielmassa analysoidaan suomalaisen mekaanisia ja sähkömekaanisia tuotteita valmistavan yrityksen mittausaineistoa, joka koostuu pääasiallisesti asiakkaiden ilmoittamista virheistä. Mittausaineiston pohjalta johdettiin GQM-menetelmää apuna käyttäen kuusi mittaria sekä tarkennettiin yrityksen käytössä olevaa mittausluokittelua. Mittareiden avulla on tarkoitus saada enemmän tietoa yrityksen ylläpitoprosessista ja asiakastyytyväisyydestä. Mittareiden luotettavuudesta ei vielä voida vakuuttua, sillä yritys ei ole testannut mittareiden toimivuutta käytännössä.

ACM-luokat (ACM Computing Classification System, 1998 version): D.2.8, K.6.3

Avainsanat: GQM, tavoitekeskeinen mittaaminen, ohjelmistoprosessin parantaminen, ylläpidon mittaaminen

1 JOHDANTO

Ohjelmistotuotanto kasvaa nykyisin vauhdilla ja kilpailu markkinaosuuksista tiukentuu koko ajan. Markkinaosuuksien saaminen vaatii ohjelmistosuunnittelijoita erityisesti asiakkaiden huomioimista. Vaikka asiakastyytyväisyyden ylläpitämiseen ja kehittämiseen tarvitaan paljon resursseja, tulee vanhan asiakkaan tyytyväisenä pitäminen viisi kertaa halvemmaksi kuin uuden asiakkaan hankkiminen. Lisäksi tyytymättömät asiakkaat kertovat keskimäärin 7-20 henkilölle ja tyytyväiset asiakkaat vain 3-5 henkilölle kokemuksistaan (Kan, 2003). Tästä syystä asiakastyytyväisyyteen panostaminen on ensiarvoisen tärkeää, jotta ohjelmistosuunnittelijat pystyisivät kilpailemaan markkinoilla ja vastaamaan muiden kilpailijoiden haasteisiin.

Suurin osa ohjelmistoprosessin ongelmista kohdataan ylläpitovaiheessa. Polon (2003) mukaan ylläpitokustannukset ovat lähes poikkeuksetta vähintään 30 % ohjelmistoprosessin kokonaiskustannuksista ja yleensä ne nousevat keskimäärin 60-80 %:iin. Jotta kustannuksia saataisiin supistettua, tulisi ohjelmistosuunnittelijoiden valvoa tarkemmin ylläpidon edistymistä. Lisäksi ylläpidon toimivuudella on suora vaikutus asiakastyytyväisyyteen. Mikäli ylläpito on tehotonta tai hidasta, asiakkaat ovat aiheesta tyytymättömiä.

Ylläpidon mittaaminen on ensiarvoisen tärkeää, jotta sen tehokkuudesta tai vaikutuksesta asiakastyytyväisyyteen, voidaan tehdä johtopäätöksiä, sillä ilman mittaamista johtopäätökset perustuvat oletuksiin ja ne saattavat olla harhaanjohtavia. Mittaamisen tarkoituksena onkin antaa todellista tietoa ohjelmistokehittäjille ylläpidon laadusta. Mittaamisen avulla pyritään havaitsemaan epätoivottuja asioita, kuten esimerkiksi ylläpidon aikana virheellisesti korjattuja muutoksia. Epätoivotut asiat toimivat ylläpidon parantamissignaaleina, joiden avulla ohjelmistokehittäjät voivat luoda strategian ongelman korjaamiseksi.

Tämän tutkielman tarkoituksena on antaa lukijalle kuva, kuinka asiakastyytyväisyyden parantaminen tapahtuu ylläpitoon keskittyvän tavoitekeskeisen mittaamisen avulla. Ensiksi luvussa 2 käsitellään ohjelmistoprosessin parantamista yleisellä tasolla. Ohjelmistoprosessin parantami-

seen voidaan käyttää monia eri keinoja, kuten arvioimalla ohjelmistoprosessia erilaisten mallien avulla. Näitä keinoja voidaan käyttää myös ohjelmiston ylläpidon parantamisessa. Luvussa 3 puhutaan mittaamisesta, sillä se tarjoaa oikein suoritettuna puolueetonta tietoa prosessin sen hetkisestä tilasta. Vaikka mittaaminen on tehokas apuväline prosessin parantamisessa, nousee mittaamisen ongelmaksi usein se, kuinka mittarit tulisi valita oikean tiedon saamiseksi. Luvussa 4 esitetään ratkaisu tähän ongelmaan tavoitekeskeisen mittaamisen avulla. Menetelmän avulla on tarkoitus kertoa, kuinka mittaamisen tavoitteet johdetaan vaiheittain tarvittaviksi mittareiksi.

Tutkielmassa käsitellään lisäksi suomalaisen mekaanisia ja sähkömekaanisia tuotteita valmistavan yrityksen mittausaineistoa. Mittausaineisto koostuu ylläpidon aikana havaituista virheistä ja asiakastyytyväisyyskyselyn tuloksista. Mittausaineiston avulla pyritään esimerkiksi selvittämään, ovatko yrityksen valitut mittarit olleet järkeviä ja kuinka mittaamista voitaisiin kehittää eteenpäin. Jotta yritys voisi itse jatkossa määritellä lisää omia mittareita, käsitellään luvussa 5 tavoitekeskeinen mittaamisen vaiheet yksityiskohtaisemmin lävitse ylläpitoon kohdistuvan esimerkin avulla. Tämän jälkeen luvussa 6 sovelletaan tavoitekeskeistä menetelmää yrityksen mittausaineistolle, jotta yritykselle tulisi käytännönläheisempi kuva menetelmän käytöstä. Lopuksi kerrotaan havaitut parannusehdotukset sekä asiakastyytyväisyyteen vaikuttavat mittarit.

Tutkielman termien käännöksissä on käytetty Tietotekniikan liitto ry:n sanastotoimikunnan sanastoa (Tietotekniikan liitto ry:n sanastotoimikunta, 2003).

2 OHJELMISTOPROSESSI JA OHJELMISTOPROSESSIN PARANTAMINEN

Ohjelmistot voivat olla nykyisin hyvin monimutkaisia ja laajoja. Tästä syystä ohjelmistoprosessin täytyy olla vakaa, jotta ohjelmiston laadulliset vaatimukset voidaan täyttää. Hyvä ohjelmistoprosessin hallinta auttaa tavoitteiden saavuttamisessa, mutta se ei aina takaa haluttua lopputulosta, sillä useasti esimerkiksi ohjelmistoprosessin aikataulu voi pettää. Tämän takia ohjelmistoprosessia tulisi parantaa, jotta halutut lopputulokset saavutettaisiin. Ohjelmistoprosessin parantaminen ei kuitenkaan ole ongelmatonta, koska parannusyritykset epäonnistuvat monesti. Vaikeudet etukäteen tiedostamalla voidaan pyrkiä varautumaan niihin. Tällöin voidaan etsiä ratkaisuja ennen kuin parantamishanke on edes kunnolla alkanut. Tämän luvun tarkoituksena on kertoa, mitä ohjelmistoprosessin parantamisella tarkoitetaan ja mitä menetelmiä parantamishankkeiden toteuttamiseen voidaan käyttää.

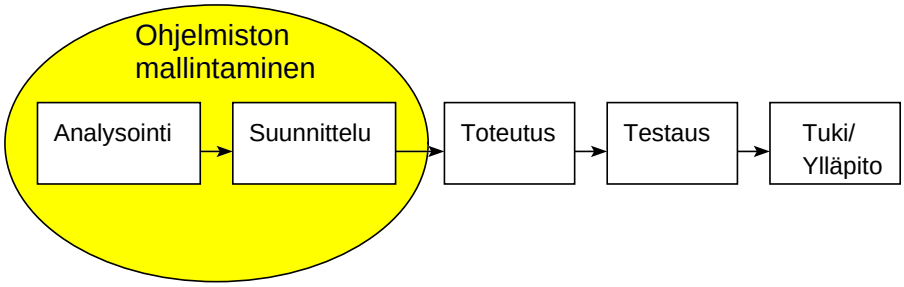
Tämän luvun tarkoituksena on määritellä prosessi, ohjelmistoprosessi ja puhua ohjelmiston laadusta sekä erilaisista ohjelmistoprosessin parantamiseen käytettävistä menetelmistä. Ohjelmistoprosessin yhteydessä puhutaan elinkaarimalleihin kuuluvasta vesiputousmallista, jonka tarkoituksena on toimia apuna ohjelmiston elinkaaren eri vaiheissa. Tämän jälkeen käsitellään ohjelmiston laatua, sillä laadukkaiden ohjelmistoprosessissa syntyvien tuotteiden avulla saavutetaan kilpailuetua muihin ohjelmistojen kehittäjiin nähden. Lopuksi puhutaan ohjelmistoprosessin hallinnasta ja parantamisesta, joiden avulla on helpompi pyrkiä hyvään laatuun. Ohjelmistoprosessin parantamisessa keskitytään siihen, mitä voi parantaa, miten parantamiskohteet tunnistetaan ja miten parantaminen tapahtuu. Lopuksi esitellään lyhyesti ohjelmistoprosessin arviointiin ja parantamiseen käytettävät TQM- ja CMMI-mallit sekä SPICE-standardi.

2.1 Ohjelmistoprosessi

Prosessi (process) on looginen ihmisistä, materiaaleista, energiasta, välineistä ja työtavoista muodostuva kokonaisuus, jonka tarkoituksena on saavuttaa ennalta määritelty lopputulos. Lopputulos voi muodostua joko palveluista, tuotteista tai molemmista (Pall, 1987). Prosessia kutsutaan *ohjelmistoprosessiksi* (software process), mikäli prosessin vaiheet tuottavat lopputuloksena ohjelmistoprosessille ominaisen tuotteen tai palvelun, kuten suoritettavan ohjelmiston tai ohjelmiston käyttöohjeen (Florac & Charleton, 1999; Humphrey, 1989). Yleisiä vaiheita kaikille ohjelmistoprosessille ovat: määrittely (specification) jossa määritellään, mitä ohjelmiston tulisi tehdä; kehittäminen (development), jossa tuotetaan ohjelmisto; validointi (validation), jossa tarkistetaan, vastaako ohjelmisto sitä, mitä asiakas haluaa ja jatkuva kehitys (evolution), jossa muutetaan ohjelmistoa muuttuvien vaatimusten mukaisiksi (Sommerville, 2000).

Ohjelmistoprosessit toteutetaan *elinkaarimallien* (life cycle model), jotka ovat yksinkertaistettuja kuvauksia ohjelmistoprosessista. Mallien avulla pyritään helpottamaan prosessin aikana tehtävää työtä ja se toimiikin muistilistana, josta voidaan tarkistaa, mitä prosessissa seuraavaksi tulee tehdä. Tunnetuin elinkaarimalli on lineaarinen peräkkäisen kehittämisen malli, joka paremmin tunnetaan vesiputousmallina. Vesiputousmalli esittää systemaattisen lähestymistavan ohjelmistoprosessin kehittämiseksi (Pressman, 2000). Ensimmäisen kerran vesiputousmallin tiettävästi esitteli Royce vuonna 1970, mutta jo vuonna 1956 Benington esitti vaiheistamisen idean (Haikala & Märijärvi, 2000). Vesiputousmallista on olemassa useita eri variaatioita. Yleensä *vesiputousmallista* (waterfall model) voidaan erottaa ainakin määrittely-, suunnitteluja toteutusvaiheet (Haikala & Märijärvi, 2000). Kuvassa 1 on esitetty vesiputousmalli Pressmanin (2000) mukaan.

Ohjelmiston mallintaminen (system engineering and modelling). Ohjelmistoprosessin kehitys alkaa ohjelmiston mallintamisella. Mallintamisella saadaan käsitys siitä, mitä asioita asiakkaalle toimitettavassa ohjelmistossa tulisi olla. Tässä kohdin asetetaan yleiset ohjelmatason vaatimukset, jotka määrittelevät asiakkaan tarpeet (Haikala & Märijärvi, 2000). Monet, kuten Hai-



Kuva 1: Vesiputousmalli (Pressman, 2000).

kala ja Märijärvi, pitävät tätä vaihetta koko elinkaaren tärkeimpänä vaiheena, koska tässä vaiheessa tehdään päätös siitä, onko asiakkaan vaatimukset ymmärretty riittävän hyvin ja onko kannattavaa rakentaa asiakkaan tarpeet täyttävä ohjelmisto.

Analysointi (analysis). Analysointivaiheessa määritellään ohjelmistovaatimukset asiakkaan vaatimuksia analysoimalla. Ohjelmistovaatimukset määrittelevät toteutettavan ohjelmiston.

Suunnittelu (design). Suunnitteluvaiheessa suunnitellaan ohjelmistovaatimuksille toteutus.

Toteutus (coding). Toteutusvaiheessa suunnitelma muutetaan lähdekieliseksi ohjelmaksi.

Testaus (testing). Testausvaiheen tarkoituksena on löytää virheitä lähdekielisestä ohjelmasta, jotta nämä virheet eivät päätyisi asiakkaalle toimitettavaan ohjelmistoon.

Tuki/Ylläpito (support). Tuki- ja ylläpitovaihe alkaa, kun ohjelmisto on toimitettu asiakkaalle. Tämän vaiheen aikana opastetaan asiakasta, korjataan tuotteesta löydettyä virheitä, parannetaan tuotteen ominaisuuksia ja sovitetaan tuote muuttuneeseen ympäristöön.

Vesiputousmalli ei sovellu laajojen ohjelmistojen kehittämiseen (Royce, 1970), koska etenkin laajoissa projekteissa joudutaan tekemään paljon muutoksia, sillä *projektit* (project) ovat ainutkertaisia prosesseja, joiden ympäristötekijät muuttavat prosessin kulkua (Choudhury, 1988). Vesiputousmallin yksi heikkous onkin, että se ei ota hyvin huomioon muutoksia, mikäli joudutaan palaamaan edelliseen vaiheeseen. Se voi aiheuttaa muun muassa sekaannusta ja ylimääräisiä kustannuksia.

Ohjelmistoprosessin kehittämiseen on tarjolla useanlaisia malleja, kuten prototyyppien käyttömalli ja vähittäisen kehittämisen malli (Pressman, 2000). Vähittäisen kehittämisen mallin tarkoituksena on kehittää lopputuote pieninä toistuvina kehitysaskelina eli inkrementteinä yhden projektin sisällä. Mallin tarkoituksena on tuottaa jokaisen inkrementin jälkeen toimiva tuote. Mallin vaiheet pohjautuvat vesiputousmalliin. Prototyyppien käyttömalli tarjoaa menetelmän asiakkaan vaatimusten selvittämiseksi. Tarkoituksena on rakentaa tuotteen hahmotelma eli prototyyppi ennen varsinaista rakentamista. Prototyypin avulla asiakkaan on helpompi kertoa tarkemmin, minkälaisen tuotteen hän haluaa (Haikala & Märijärvi, 2000; Pressman, 2000).

Ohjelmiston kehitys, prosessia apuna käyttäen, on merkittävästi helpottanut tunnistamaan ohjelmiston kehityksen ulottuvuudet ja ongelmat. Ohjelmiston kehittämisellä ei tarkoiteta vain hienoja työkaluja tai ympäristöjä, vaan se koostuu merkittävästi myös organisaation, ihmisten, teknologian sekä taloudellisten tekijöiden huomioimisesta. Tärkeimpänä prosessinäkökulman ajatuksena voidaan pitää sitä, että mitä tehokkaammin ohjelmistoprosessiin panostetaan, sitä parempaan ohjelmistotuotteiden laatuun on mahdollisuus (Fuggetta, 2000).

2.2 Ohjelmiston ja ohjelmistoprosessin laatu

Laatu on subjektiivinen käsite ja se merkitsee eri ihmisille eri asiaa. Täten laatua ei voi määritellä yksiselitteisesti sen moniulotteisuutensa takia. Kanin (2003) mukaan laatu, yleisessä mielessä, on abstrakti käsite, joka ei ole mitattavissa tai määriteltävissä oleva asia. Ammattimaisessa mielessä laatu on määriteltävä, jotta voimme tyydyttää asiakkaan toiveet ja odotukset.

Ohjelmiston ja ohelmistoprosessin laadun määrittelyyn voidaan soveltaa Crosbyn (1979) ajatusta laadusta. Crosbyn (1979) mukaan tuote on laadukas, jos sille asetetut vaatimukset toteutuvat. Toisin sanoen *ohjelmisto* on *laadukas* (software quality), mikäli se vastaa kaikkia sille asetettuja vaatimuksia. Laadukas ohjelmisto voidaan saavuttaa *laadukkaalla ohjelmistoprosessilla* (software process quality), sillä sen tarkoituksena on tuottaa prosessin vaiheiden avulla

asiakkaan tarpeet täyttävä ohjelmisto. Ohjelmiston kehityksen aikana tulisi suorittaa säännöllisiä mittauksia, jotta ohjelmistoprosessin ja lopputuotteen laadusta voitaisiin varmistua.

Tuotteen vaatimukset saatettiin päättää vielä 1980-luvulla ilman asiakkaan apua (Kan, 2003), jolloin lopputuotteen laatu ei ollut sitä, mitä asiakas halusi. Laadukasta ohjelmistoa tavoiteltaessa tuleekin antaa asiakkaan mielipiteille painoarvoa, sillä asiakas lopulta päättää, haluaako hän ostaa tuotteen vai ei (Guaspari, 1985). Kanin (2003) mukaan asiakkaan ja tuottajan näkökulmat laadusta poikkeavat. Tuottajan mielestä tuote on laadukas, kun se täyttää asiakkaan asettamat vaatimukset. Asiakkaalle tämä ei riitä, sillä hän haluaa toteutuneiden vaatimusten lisäksi, että tuote sopii juuri hänen mieltymystensä mukaisiin käyttötarpeisiin (fitness for use). Jatkossa tulisikin kuluttaa enemmän aikaa asiakkaan toiveiden ja halujen ymmärtämiseksi. Silloin voidaan toteuttaa asiakkaan tarpeet paremmin ja kasvattaa asiakastyytyväisyyttä. Tätä kautta on mahdollista saada uskollisia asiakkaita sekä parantaa kilpailuetua.

2.3 Ohjelmistoprosessin parantaminen

Ohjelmistoprosessin parantaminen kuuluu osana ohjelmistoprosessin hallintaan. *Ohjelmistoprosessin hallinnan* (software process management) avulla on tarkoitus hallita onnistuneesti ohjelmiston kehitys-, ylläpito- ja tukitoimintoja (Florac & Charleton, 1999), jolloin prosessin tuotteet ja palvelut tuotetaan asiakkaan vaatimusten sekä liiketoiminnan tavoitteiden mukaisesti. Hallittu ohjelmistoprosessi tuo lisää vakautta, joka mahdollistaa muun muassa sen, että asiakkaan ilmoittamat muutospyynnöt saadaan korjattua sovitun aikataulun mukaisesti.

Floracin ja Charletonin (1999) mukaan ohjelmistoprosessin hallinnalla on neljä eri päävastuualuetta: prosessin määrittely, -kontrollointi, -mittaaminen sekä -parantaminen. Nämä vastuualueet ovat samankaltaisia Shewhartin (1939) jatkuvan parantamisen kierron kanssa. Deming (1986) toi saman ajatuksen laajemmalti ihmisten tietoisuuteen plan/do/check/act -menetelmänä. Prosessin määrittelyn tarkoituksena on luoda kurinalainen ympäristö prosessin kontrolloimi-

seen ja parantamiseen. Kontrolloinnin avulla prosessi pyritään pitämään sille asetettujen rajojen sisäpuolella.

Mittaamisen avulla voidaan valvoa kontrolloinnin tehokkuutta ja löytää poikkeamat eli arvot, jotka ylittävät prosessin normaalit rajat. Poikkeamat toimivat prosessin parantamisen signaaleina, jolloin prosessin toimintaa pyritään muuttamaan siten, että prosessi saadaan takaisin hallintaan.

Ohjelmistoprosessin parantamisen (software process improvement, SPI) avulla on tarkoitus tuottaa sellainen perusrakenne ja kulttuuri, joka tukee tehokkaita menetelmiä ja toimintamalleja ohjelmistoprosessin parantamiseksi (Conradi & Fuggetta, 2002). Tämän perusrakenteen tulee täydentää organisaation liiketoiminnan mallia. Ohjelmistoprosessin parantamisella on useanlaisia eri parannuskohteita, kuten kilpailuetu, strategia, motivaatio tai ihmisten roolit. Ohjelmistoprosessin parantamisella ei siis vain tarkoiteta ainoastaan ohjelmiston teknisiä ratkaisuja, vaan parantamishankkeiden tulisi johdonmukaisesti keskittyä organisaation kaikkiin osaalueisiin (Cattaneo & al., 2001).

Van Solingenin ja Berghoutin (1999) mukaan prosessin parantamistavoitteiden tunnistamiseksi voidaan määritellä neljä eri tapaa. Ensimmäinen tapa on johtaa parantamistavoitteet organisaation liiketoiminnan tavoitteista. Toinen tapa on prosessin arviointi, jolloin parantamisen tavoitteet määritellään arviointitulosten perusteella. Kolmantena keinona on prosessin mittaaminen, jonka avulla voidaan tunnistaa prosessissa esiintyviä ongelmia. Viimeisenä vaihtoehtona on ottaa huomioon yksittäisen prosessin tarpeet ja määritellä niistä parantamistavoitteet. Van Solingen ja Berghout (1999) painottavat, että ohjelmistoprosessin kehitystä varten tulee olla selkeästi määritellyt parantamistavoitteet. Muutoin ohjelmistoprosessin parantamisesta voi tulla kaoottista.

Ohjelmistoprosessin parantamista varten on kehitetty kypsyysmalleja (maturity models) ja parantamismenetelmiä (improvement methods). Kypsyysmallien, kuten *CMMI-kypsyystasomallin* (CapabilityMaturity Integration Model), tarkoituksena on määritellä ihanteellinen profiili pro-

sessille. Malli asettaa minimivaatimukset, jotka prosessin tulee täyttää, jotta prosessi voisi saavuttaa tarkoin määrätyn aseman. Parantamismenetelmät, kuten *SPICE-standardi* (Software Process Improvement and Capability dEtermination) ja *TQM* (Total Quality Managenent), puolestaan määrittelevät strategian prosessin parantamiseksi. Parantamismenetelmä kertoo, kuinka prosessia arvioidaan, miten muutoksiin vaadittavat toimenpiteet tehdään, ja kuinka uusien parannushankkeiden syntymistä edistetään jatkossa (Cattaneo 2001; Fuggetta, 2000).

2.3.1 TQM

Toisen maailmansodan jälkeen japanilaisten tuotteet tulivat niin kilpailukykyisiksi, että muuallakin maailmassa alettiin kokeilla ja kehittää Japanissa käytettyjä laadunparantamismenetelmiä (Deming 1986; Åhlberg, 1997). Kokonaisvaltaisen laadunhallinnan TQM:n juuret pohjautuvat myös japanilaiseen laadun parantamiseen ja sitä pidetään lähes kaikkien ohjelmistoprosessien parantamismenetelmien lähtökohtana (Conradi & Fuggetta, 2002). Termiä TQM alettiin käyttää vuonna 1985.

TQM:n perusperiaate on rakentaa kulttuuri, jossa kaikki organisaation työntekijät osallistuvat prosessin, tuotteiden ja palveluiden parantamiseen. TQM:n avainelementit ovat asiakkaaseen keskittyminen, prosessi, työntekijöiden kehittäminen (human side of quality), mittaaminen ja analysointi. Ensimmäisenä avainelementtinä on asiakastyytyväisyyteen keskittyminen. Siihen kuuluu asiakkaan toiveiden ja halujen kartoittaminen, asiakkaan vaatimusten kerääminen sekä asiakastyytyväisyyden mittaaminen.

Toisena avainelementtinä on prosessi. Tavoitteena on vähentää prosessin variaatiota sekä keskittyä jatkuvaan prosessin parantamiseen. Kolmantena avainelementtinä on työntekijöiden kehittäminen, johon kuuluu koko yhtiön laajuinen laatukulttuurin luominen. Kulttuurin luomisessa keskitytään muun muassa johtajuuteen, sitoutumiseen, osallistumiseen, työntekijän valtuuksiin sekä muihin sosiaalisiin sekä psykologisiin tekijöihin.

Viimeisenä avainelementtinä on mittaaminen ja analysointi. Tavoitteena on ylläpitää tavoitekeskeistä (goal-oriented) mittausjärjestelmää, joka keskittyy jatkuvaan laadun parantamiseen (Kan, 2003). Parhaiten kokonaisvaltainen laadunhallinta onnistuu, jos korkein johto sitoutuu laadunkehittämiseen ja sen johtamiseen. TQM:n epäonnistumisen yleisin syy onkin, että johto ei ole riittävän hyvin sitoutunut jatkuvaan laadunkehittämiseen (Åhlberg, 1997). TQM on toiminut esimerkkinä lukuisille malleille, kuten SEI:n CMM -mallille (Kan, 2003), joka on vuorostaan toiminut pohjana CMMI:lle (Chrissis & al., 2003) sekä SPICE:lle (Garcia, 1997; SEI, 2005).

2.3.2 CMMI

Software Engineering Institute (SEI) aloitti vuonna 1986 kehittää ohjelmiston kyvykkyys-kypsyysmallia (Capability Maturity Model for Software, SW-CMM). Alun perin idean kypsyystasoista toi julkisuuteen Philip Crosby (1979) kirjassaan "Quality Is Free" (Paulk & al., 1993). SW-CMM:stä tuli erittäin suosittu ja sen suosion ansiosta alettiin kehittää lisää CMM-malleja. Lopulta 1990-luvun lopussa tuli tarve yhdistää näitä malleja, sillä mallit olivat keskenään epäyhtenäisiä ja niitä oli hankala integroida toisiinsa. Niinpä SEI:n CMM-integrointiope- raatio CMMI alkoi vuonna 1998 yhteistyössä Yhdysvaltojen viranomaisten ja teollisuuden kanssa. CMMI:n missiona oli yhdistää SW-CMM, SECM (The Systems Engineering Capability Model) ja IPD-CMM (The Integrated Product Development Capability Maturity Model). Ensimmäinen julkinen versio 0.2 julkaistiin vuonna 1999 ja uusin versio 1.2 julkaistiin vuonna 2006 (Chrissis & al., 2003; SEI, SEI, 2006).

CMMI-mallin avulla kukin organisaatio voi selvittää välttämättömät tarpeet, jotka vaaditaan tehokkaiden prosessien saavuttamiseksi. CMMI asettaa yksittäisten prosessien tai jopa koko organisaation kattavat parantamistavoitteet ja prioriteetit, opastaa laatuprosesseissa sekä yhdistää erilliset organisaatiot (SEI, 2006). CMMI kattaa systeemitekniikan- (system engineering), ohjelmistotuotannon-, tuote- sekä prosessikehityksen integroinnin ja hankkijan (supplier

Taulukko 1: CMMI:n kyvykkyys- ja kypsyystasojen vertailu (Chrissis & al., 2003).

Taso	Kyvykkyystaso	Kypsyystaso
0	epätäydellinen	———
1	suoritettu	kaoottinen
2	toistuva	toistuva
3	määritelty	määritelty
4	hallittu	hallittu
5	optimoiva	optimoiva

sourcing) mallit (Chrissis & al., 2003). CMMI:ssä on kaksi tapaa käyttää malleja: jatkuva (continuous) ja vaiheittainen (staged). Mallien sisältö on hyvin samantyylinen, mutta niiden lähestymistavat poikkeavat toisistaan. Jatkuva malli keskittyy organisaation yksittäisten osa-alueiden parantamiseen ja vaiheittainen malli soveltuu koko organisaation kehittämiseen.

Taulukossa 1 on lueteltu jatkuvan mallin kyvykkyystasot sekä vaiheittaisen mallin kypsyystasot. Tasojen samankaltaisuuden vuoksi keskitytään kuvailemaan ainoastaan kyvykkyystasoa hieman tarkemmin (Chrissis & al., 2003; SEI, 2002). Tasojen väliset eroavaisuudet koostuvat siitä, että tasoa nolla ei esiinny kypsyystasossa ja nimet poikkeavat toisistaan tasolla yksi. Seuraavalle tasolle pääsemiseksi pitää täyttää kaikki edellisen tason vaatimukset.

0. *Epätäydellinen* (incomplete): Yleisiä tavoitteita ei ole. Prosessi on kokonaan tai osittain suunnittelematon. Ohjelmistoprosessin avulla ei välttämättä saavuteta yksittäisiä prosessialueen tavoitteita.

1. *Suoritettu* (performed)/*Kaoottinen* (initial): Tavoitteena on täyttää yksittäiset prosessialueen tavoitteet. Prosessin välttämättömät tavoitteet saadaan suoritettua. Prosessi voi olla epävakaa ja epätäydellinen määrittelyn, suunnittelun, kontrolloinnin ja toteutuksen osalta.

2. *Toistuva* (managed): Tavoitteena on vakiinnuttaa toistuva prosessi. Prosessin suunnittelevat ja toteuttavat pätevät henkilöt. Henkilöt noudattavat ja kontrolloivat prosessia.

3. *Määritelty* (defined): Tavoitteena on vakiinnuttaa määritelty prosessi. Prosessi on räätälöity vastaamaan organisaation ohjeita ja standardeja. Toistuvan ja määritellyn tason erona on se, että toistuvalla tasolla prosessit voivat olla hyvinkin vaihtelevia samankaltaisissa projekteissa organisaation sisällä. Määritetyllä tasolla tätä variaatiota ei esiinny. Toinen merkittävä ero on, että määritellyn tason prosessit ovat perusteellisemmin määriteltyjä kuin toistuvan tason prosessit. Tällöin parantamisaskeleita on helpompi ymmärtää, analysoida ja toteuttaa.

4. *Hallittu* (quantitatively managed): Tavoitteena on vakiinnuttaa hallittu prosessi. Prosessia mitataan ja tietoa kerätään analysointia varten, jotta prosessia saataisiin parannettua. Määritellyn ja hallitun tason erona on se, että hallitulla tasolla prosessin tulevaisuuden käyttäytymistä voidaan ennustaa. Määritetyllä tasolla vain prosessin laatua voidaan etukäteen ennustaa.

5. *Optimoiva* (optimizing): Tavoitteena on vakiinnuttaa optimoiva prosessi. Prosessin parantamiseen keskitytään jatkuvasti sekä tunnistetaan prosessin puutteet ja ongelmat. Hallitun ja optimoivan tason välisenä erona on, että hallitulla tasolla parantamistavoitteet eivät ole niin kokonaisvaltaisia, toisin sanoen keskitytään vain tiettyjen puutteiden parantamiseen. Hallitulla tasolla prosessin vaihtelevuus on suurempi. Optimoivalla tasolla prosessi on vakaampi.

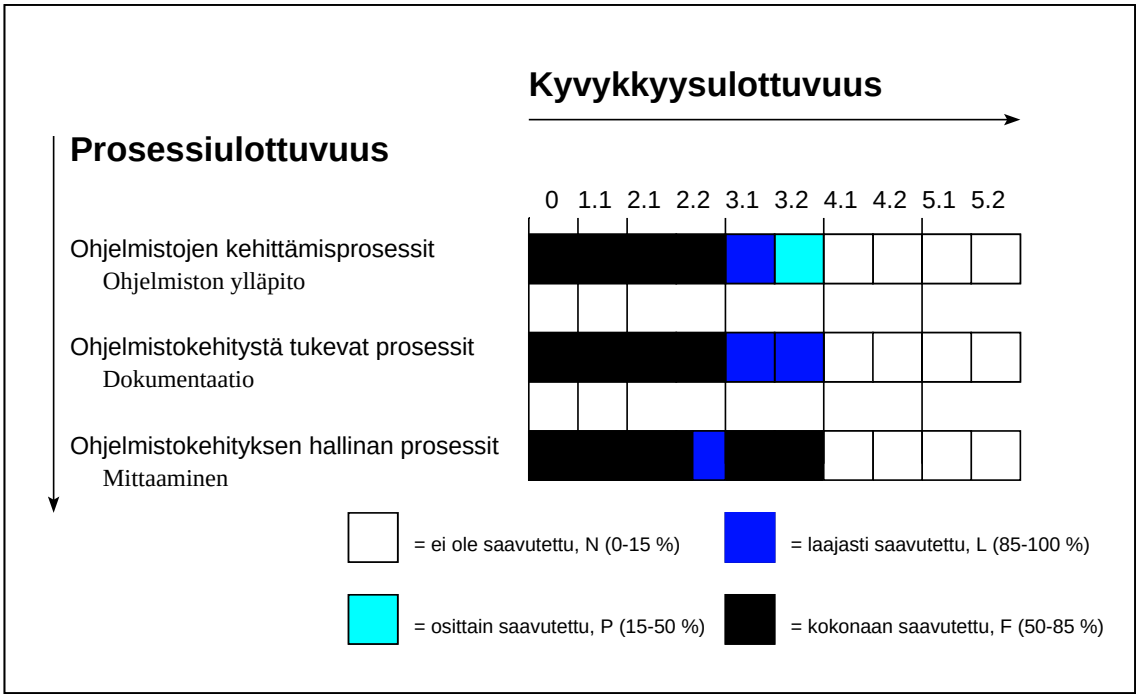
SEI:n (2005a) mukaan moni yritys on ottanut CMMI:n osaksi liiketoimintaansa. Näihin yrityksiin kuuluvat muun muassa BMW, Bosch, Ericsson, Fujitsu, Intel, NASA, Nokia sekä Motorola.

Yrityksen, joka haluaa parantaa liiketoimintaansa, on muistettava, että korkea kypsyystaso ei välttämättä takaa korkeaa tuotteiden laatua. Huonommallakin tasolla voi hyvin pystyä tuottamaan kilpailukelpoisia tuotteita (Conradi & Fuggetta, 2002).

2.3.3 SPICE

Vuonna 1993 kansainvälinen ohjelmistotuotannon standardikomitea (ISO/IEC JTC1/SC7) aloitti SPICE-projektin. Projektin tarkoituksena oli luoda kansainvälinen standardi, ISO/IEC 15504

ohjelmointiprosessin arviointia varten (SPICE, 1998). Standardin tekninen raportti, ISO/IEC TR 15504 julkaistiin vuonna 1998. Teknisen raportin uudelleenjulkaiseminen, ISO/IEC 15504 alkoi vuonna 2003. Uudelleenjulkaisu kattaa samat osa-alueet kuin aikaisempi raportti, mutta asiat on tiivistetty seitsemään eri osaan yhdeksän sijasta. Uudet osa-alueet ovat: osa 1, käsitteet ja sanasto (concepts and vocabulary); osa 2, arvioinnin suorittaminen (performing an assessment); osa 3, ohjeistus arvioinnin suorittamiseen (guidance on performing an assesment); osa 4, ohjeistus prosessin parantamisen ja kypsyyden määrittelyyn (guidance on use for process improvement and process capability determination); osat 5 ja 6 esimerkit prosessin arviointimallista (an exemplar process assessment model) sekä osa 7 organisaation kypsyyden arviointi (assessment of organizational maturity). Osat 1-5 ovat jo ilmestyneet, mutta standardin osa 6 julkaistaan tämän vuoden aikana ja osa 7 on tarkoitus julkaista vuonna 2008 (SPICE Network, 2007).



Kuva 2: Esimerkki SPICE-standardin perusrakenteesta.

Kuvassa 2 on hahmotettu SPICE-standaridin perusidea, joka koostuu, prosessi- ja kyvykkyysu-

lottuvuuksista (ISO/IEC, 2005). Nämä kaksi ulottuvuutta muodostavat SPICE:n perusarkkitehtuurin. Prosessiulottuvuus (process dimension) määrittelee arvioitavat prosessit ja se jakaantuu kolmeen elinkaariprosessiin, jotka puolestaan jakautuvat yhdeksään prosessiryhmään seuraavan jaottelun mukaisesti:

1. Varsinaiset elinkaariprosessit
    - Ohjelmistojen hankintaprosessit (acquisition process group, ACQ)
    - Ohjelmistojen toimitusprosessit (supply process group, SPL)
    - Ohjelmistojen kehittämisprosessit (Engineering process group, ENG)
    - Ohjelmistotoiminnan prosessit (operation process group, OPE)
2. Tukevat elinkaariprosessit
    - Ohjelmistokehitystä tukevat prosessit (support process group, SUP)
3. Organisationaaliset elinkaariprosessit
    - Ohjelmistokehityksen hallinnan prosessit (management process group, MAN)
    - Ohjelmistokehityksen parantamisprosessit (process improvement process group, PIM)
    - Ohjelmistojen resurssi- ja infrastruktuuriprosessit (resource and infrastructure process, RIN)
    - Ohjelmistojen uudelleenkäytön prosessit (reuse process group, REU)

Kuvassa 2 kyvykkyysulottuvuus (capability dimension) kuvaa asteikon prosessin kyvykkyyden mittaamiseksi. Kyvykkyystasoja on kuusi ja ne muistuttavat ominaisuuksiltaan paljolti CMMI:n kyvykkyystasoja. Taulukossa 2 on lueteltu kyvykkyystasot sekä *ominaisuudet* (attributes), joista kyvykkyystasot muodostuvat.

Taulukko 2: SPICE:n kyvykkyystasot ja prosessin ominaisuudet (ISO/IEC, 2005).

Ominaisuus	Kyvykkyystasot ja prosessin ominaisuudet
	Taso 0: epätaydellinen prosessi
	Taso 1: suoritettu prosessi
1.1	prosessin suorittaminen
	Taso 2: toistuva prosessi
2.1	prosessin suorittamisen hallinta
2.2	prosessin työtulosten hallinta
	Taso 3: määritelty prosessi
3.1	prosessin määrittely
3.2	prosessin käyttäminen
	Taso 4: hallittu prosessi
4.1	prosessin mittaaminen
4.2	prosessin ohjaus
	Taso 5: optimoiva prosessi
5.1	prosessin uudistaminen
5.2	prosessin jatkuva parantaminen

SPICE-arvioinnissa prosessin ominaisuudet arvioidaan joko neliportaisella (rating scale) tai prosenttiasteikolla. Neliportainen arvosteluasteikko muodostuu seuraavista kohdista: ominaisuutta ei ole saavutettu (N, 0-15 %), ominaisuus saavutettu osittain (P, 15-50 %), ominaisuus saavutettu laajasti (L, 50-85 %) sekä ominaisuus saavutettu kokonaan (F, 85-100 %). Asteikko kuvaa prosessien toteutumisen kyvykkyyttä ja tietoa voidaan hyödyntää joko prosessien arvioinnissa tai parantamisessa (SPICE Network, 2007). Tietyn kypsyystason saavuttamiseksi täytyy prosessin ominaisuuksien olla kokonaan saavutettu kaikilla edellisillä tasoilla, ja seuraavalle tasolle pääsemiseksi pitää prosessin ominaisuuksien olla vähintään laajasti saavutettu (El Emam & al., 1998). Esimerkiksi kuvassa 2 ohjelmiston kehittämisprosesseihin kuuluva ohjelmiston ylläpito on saavuttanut kypsyystason 2, koska ominaisuudet 1.1-2.2 ovat kokonaan saavutettu. Ohjelmiston ylläpito ei kuitenkaan ole vielä päässyt kypsyystasolle 3, koska omi-

naisuus 3.2 on vain osittain saavutettu. Sen sijaan ohjelmistokehitystä tukeva dokumentaatioprosessi on päässyt kypsyystasolle 3, koska sen molemmat ominaisuudet ovat laajasti saavutettu. Ohjelmistokehityksen hallintaan kuuluva mittausprosessi on saavuttanut kypsyystason 2, koska ominaisuudet 1.1 ja 2.2 ovat kokonaan saavutettu ja ominaisuus 2.2 on laajasti saavutettu. Vaikka ominaisuudet 3.1 ja 3.2 ovat kokonaan saavutettu, ne eivät pysty nostamaan mittaamisprosessia kypsyystasolle 3 ominaisuuden 2.2 takia.

SPICE-standardia ei ole suunniteltu itsenäisesti käytettäväksi. Se sopii yhteenkäytettäväksi muun muassa seuraavien mallien kanssa (SEI, 2005): CMMI; ISO 9001, joka on yleinen standardi yrityksen tuotekehitystä ja tuotantoa varten; Trillium, joka on telealalle tarkoitettu CMM:iin perustuva arviointimenetelmä ja Bootstrap, jonka arviointimenetelmä perustuu ISO 9001 -standardiin ja CMM-arvioitimenetelmään (Haikala & Märijärvi, 2000). SPICE-projekti virallisesti lakkautettiin vuonna 2003, mutta SPICE Network sai valtuudet koordinoida SPICE-yhteisön toimintaa.

Vaikka malleja tarvitaan ohjelmistoprosessien parantamiseen ja työnteon tukemiseksi, malleja ei pitäisi käyttää tavoitteiden saavuttamisen pääkeinona. Mikäli prosessi nähdään kaavoina, prosessin käyttäjän tietämys jää taka-alalle (Aaen, 2003). Ohjelmistotyöhön kuuluu yhtenä osana ihmisten luova työpanos sekä ihmisten välinen kanssakäyminen. Standardoiduilla prosessimalleilla näitä ominaisuuksia ei täysin voi ennalta suunnitella. Prosessin parantamisen tulisi johtaa oppivaan ja käyttäjäkeskeiseen organisaatioon (Conradi & Fuggetta, 2002).

2.4 Asiakastyytyväisyys

Asiakastyytyväisyys on tärkeä tekijä myös ohjelmistotuotannossa. Ilman asiakastyytyväisyyttä asiakasuskollisuuden saavuttaminen on hankalaa, jolloin yrityksen voi olla vaikeampi kilpailla markkinaosuuksista. Ohjelmistoprosessin aikana tulisikin kiinnittää enemmän huomiota asiakkaaseen, jotta asiakkaan tarpeet voitaisiin täyttää paremmin.

Kan (2003) määrittelee *asiakastyytyväisyyden* (customer satisfaction) koostuvan hyvästä lopputuotteen, toimituksen ja palvelun laadusta. Tämän lisäksi Zeithamlin ja Bitnerin (1996) nostavat esiin tuotteen tai palvelun hinnan sekä tilanne- ja yksilötekijät. Tilannetekijät ohjaavat henkilöiden käyttäytymistä ja yksilötekijät profiloivat henkilöä muun muassa iän, koulutuksen ja persoonallisuuden mukaan (Ylikoski, 2001). Asiakastyytyväisyys on siis moniulotteisempi käsite, eikä muodostu pelkästään laadukkaasta lopputuotteesta.

Kuten jo ohjelmistoprosessin laadun yhteydessä todettiin, asiakkaalle ei riitä, että ohjelmisto täyttää asiakkaan vaatimukset. Asiakkailla on paljon piileviä useasti tiedostamattomia toiveita, jotka vaikuttavat asiakastyytyväisyyden muodostumiseen. Jotta asiakkaan toiveet saataisiin täytettyä, tulisi yrityksen ymmärtää asiakkaan arvontuotanto syvällisemmin. Storbackan & al. (1999) mukaan yleiseksi ongelmaksi saattaa muodostua se, että tuotteen toimittaja olettaa asiakkaan pystyvän analysoimaan ja kertomaan, mitä hän haluaa. Lisäksi toimittaja saattaa olettaa, että asiakas pystyy itse tunnistamaan parhaiten tarpeidensa muutoksen. Todellisuudessa asiakas ei välttämättä tiedä, mikä on mahdollista, jolloin yrityksen tulisi pystyä kartoittamaan asiakkaan tulevia tarpeita nykyisten tarpeiden pohjalta. Tällöin asiakkaalle voidaan tarjota uusia mahdollisuuksia, jotka pidemmällä aikavälillä vaikuttavat yrityksen asiakastyytyväisyyteen ja kilpailukykyyn.

Asiakastyytyväisyyttä voidaan selvittää esimerkiksi asiakastyytyväisyyskyselyjen avulla. Kyselyitä tehtäessä tulisi muistaa että, organisaation kiinnostus asiakkaiden mielipiteitä kohtaan lisää asiakkaiden odotuksia (Ylikoski, 2001). Asiakkaat odottavat, että kyselyiden pohjalta tehdään parannustoimenpiteitä, joiden vaikutukset näkyvät esimerkiksi asiakastuen kehittyneempänä palveluna. Jotta parannustoimenpiteiden onnistumisesta voitaisiin vakuuttua, tarvitaan mittareita, joka kertovat konkreettisesti, kuinka parantamisessa onnistuttiin.

3 OHJELMISTOPROSESSIN MITTAAMINEN

Mittaamista tarvitaan jokapäiväisessä elämässä ja sen avulla esimerkiksi sairaanhoitaja voi selvittää lapsen pituuden ja painon tai yleisurheilija voi seurata kilpailutulostensa kehittymistä kauden aikana. Ilman mittaamista teknologia ei voi toimia ja kehittyä, sillä mittaaminen helpottaa ymmärtämään maailmaamme ja parantamaan elämänlaatuamme (Fenton & Pfleeger, 1997).

Tämän luvun tarkoituksena on kertoa, mitkä ovat ohjelmistoprosessin mittaamisen tavoitteet ja mihin mittaaminen kohdistuu. Sen jälkeen perehdytään mittaamisen peruskäsitteisiin, kuten siihen, miten mittarit voidaan luokitella. Mittareiden luokittelun jälkeen tutustutaan hyvän mittarin ominaisuuksiin: luotettavuuteen ja oikeellisuuteen. Ilman näitä ominaisuuksia mittarin tuottama tieto on epäoleellista ja pahimmassa tapauksessa harhaanjohtavaa. Tämän jälkeen esitellään mittaustiedon havainnollistamismenetelmiä, jotka muun muassa Arthurin (1993) ja Kanin (2003) mukaan toimivat hyvin laadun parantamisen apuna. Lopuksi kerrotaan muutamia mittaukseen liittyviä asioita, joiden avulla voidaan välttyä yleisimmiltä ongelmilta.

3.1 Mittaamisen tarkoitus ja -kohteet

Ohjelmistoprosessin mittaaminen (software measurement) on jatkuva prosessi, jossa määritellään, kerätään ja analysoidaan tietoa (van Solingen & Berghout, 1999). Tietojen avulla on tarkoitus oppia, ymmärtää, arvioida, ennustaa, ohjata ja parantaa ohjelmistoprosesseja, -tuotteita sekä -resursseja. (Fenton & Pfleeger, 1997; Park & al, 1996; Pressman, 2000; van Solingen & Berghout, 1999; van Solingen & Berghout, 2001).

Van Solingenin & Berghoutin (2001) mukaan oppiminen on mittaamisen tärkein tavoite, koska ilman sitä ei ole edellytyksiä esimerkiksi ymmärtää prosessin luonnetta ja saada aikaan parannustoimenpiteitä.

Mittaaminen auttaa ymmärtämään, mitä ohjelmiston kehityksen ja ylläpidon aikana tapahtuu.

Tällöin voidaan arvioida prosessien, tuotteiden tai resurssien nykyistä tilannetta ja asettaa tavoitteet tulevalle toiminnalle (Fenton & Pfleeger, 1997). Mittaamisen avulla voidaan esimerkiksi huomata, kuinka paljon virheitä löytyy asiakkaalle toimitetusta tuotteesta. Tämän ymmärtäminen motivoi jatkossa työntekijöitä parempaan tuotteen testaamiseen, sillä mitä virheettömämpi tuote on, sitä vähemmän asiakkaalta tulee valituksia.

Mittaamisen avulla voidaan lisäksi ennustaa, kuinka prosessit, tuotteet tai resurssit voisivat käyttäytyä jatkossa. Tällöin prosessien, tuotteiden ja resurssien toimintoja ohjataan siten, että prosessit saavuttavat tavoitteensa (Fenton & Pfleeger, 1997; Park & al., 1996; Pressman, 2000). Esimerkiksi mittaamisen avulla projektipäällikkö voi ennustaa, kuinka paljon työntekijöiltä kuluu aikaa kunkin työtehtävän suorittamiseen. Tällöin hän voi tarvittaessa muuttaa työtehtävien resursseja esimerkiksi varaamalla enemmän aikaa vaativampiin työtehtäviin.

Mittaamisella pyritään havaitsemaan ongelmat, jotka vaativat parannustoimenpiteitä ja parannustoimenpiteiden avulla prosessit yritetään saada takaisin hallintaan. Parannustoimenpiteiden onnistumisesta voi varmistua vain uudelleenmittauksella. Esimerkiksi jos asiakastyytyväisyyden mittauksessa havaitaan, että asiakkaat ovat tyytymättömiä, parannustoimenpiteet ovat tarpeellisia (Fenton & Pfleeger, 1997; van Solingen & Berghout, 1999).

Mittaaminen vaatii aikaa ja investointeja, joten ennen mittaamisen aloittamista tulee huolella selvittää mittaamisen tarkoitus ja kohde. Mittaamisella on kolme kohdetta: prosessit, tuotteet ja resurssit (Fenton & Pfleeger, 1997). Prosessimittarit mittaavat tuotteen kehitysprosessia, kuten ajankäyttöä. Tuotemittarit mittaavat ohjelmistoprosessissa kehitettäviä välituotteita, kuten vaatimusmäärittelyä, tai lopullista asiakkaalle toimitettavaa tuotetta. Resurssimittarit mittaavat resurssien, kuten henkilöstön, työvälineiden tai menetelmien, merkitystä prosessissa. Niiden tehtävänä on auttaa prosessin ymmärtämisessä ja kontrolloinnissa (Fenton & Pfleeger, 1997; van Solingen & Berghout, 1999).

Mittaamisen kohteet voidaan jakaa vielä sisäisiin ja ulkoisiin ominaisuuksiin. Sisäisellä ominaisuudella tarkoitetaan sitä, että prosessin, tuotteen tai resurssin ominaisuudet voidaan mitata

ainoastaan liittyen kyseiseen prosessiin, tuotteeseen tai resurssiin. Esimerkiksi vaatimusmäärittelyssä olevien kirjoitusvirheiden määrä voidaan mitata suoraan vaatimusmäärittelystä, jolloin mittaamisessa ei tarvitse ottaa huomioon muita ominaisuuksia. Ulkoisten ominaisuuksien mittaamisessa täytyy ottaa huomioon prosessin, tuotteen tai resurssin vaikutus ympäristöön. Esimerkiksi vaatimusmäärittelyn ylläpidon mittaamisessa täytyy huomioida ulkoiset tekijät, kuten henkilöiden taidot ja työvälineet, jotka vaikuttavat vaatimusmäärittelyn ylläpidon laatuun (Fenton & Pfleeger, 1997).

Taulukossa 3 on esitelty mittaamisen kohteet sekä esimerkkejä niiden mitattavista sisäisistä ja ulkoisista ominaisuuksista.

Sisäisten ja ulkoisten ominaisuuksien erottaminen toisistaan on tärkeää, sillä sisäiset ominaisuudet kertovat paremmin, kuinka ohjelmiston kehitystä voitaisiin parantaa tulevissa projekteissa. Ulkoiset ominaisuudet, kuten tuotteen käytettävyys, mitataan yleensä projektin loppuvaiheessa, jolloin voidaan vain todeta, kuinka ominaisuuden kehittämisessä onnistuttiin. Sisäisien ominaisuuksien mittaamisen avulla ongelmat saadaan aikaisemmassa vaiheessa selville, jolloin parantamistoimenpiteisiin ryhtyminen ei ole vielä liian myöhäistä (Fenton & Pfleeger, 1997).

Taulukko 3: Esimerkkejä kohteiden sisäisistä- ja ulkoisista ominaisuuksista (Fenton & Pfleeger, 1997).

	Ominaisuus	
Kohde	**Sisäinen**	**Ulkoinen**
Prosessi		
Vaatimusten laatiminen	aika, työpanos, vaatimuksiin tehtyjen muutosten lukumäärä	laatu, kustannukset, stabiilisuus
Yksityiskohtainen suunnittelu	aika, työpanos, löydettyjen virheiden lukumäärä	kustannukset, kustannus tehokkuus
Tuote		
Määrittelyt	koko, uudelleenkäytetettävyys, syntaktinen oikeellisuus	ymmärrettävyys, ylläpidettävyys
Suunnitelmat	koko, modulaarisuus, toiminnallisuus	laatu, monimutkaisuus, ylläpidettävyys
Testiaineisto	koko, kattavuuden taso	laatu
Resurssi		
Henkilöstö	ikä, palkka	tuottavuus, kokemus
Laitteisto	hinta, nopeus, muistin koko	luotettavuus
Työskentelytilat	koko, lämpötila, valaistus	mukavuus, laatu

3.2 Mittaamisen peruskäsitteitä

Mittausprosessi alkaa, kun syntyy tarve mittaamiselle. Ensimmäiseksi tulee pohtia, mitä mittaamiselta halutaan, mikä on mittaamisen tarkoitus ja mihin mittaaminen halutaan kohdistaa:

prosesseihin, tuotteisiin vai resursseihin. Ennen mittaamista tulee huomioida, ovatko mittarit sopivia kohteen mittaamiseen ja saadaanko niiden avulla luotettavaa tietoa mittauksen kohteesta. Tämän jälkeen valitaan mittarit, suoritetaan mittaus ja lopuksi tulokset julkaistaan erilaisten kaavioiden avulla. Tässä luvussa keskitytään mittaamisen peruskäsitteiden esittelyyn, kuten mittareiden luokitteluun, hyvän mittarin ominaisuuksiin sekä mittaustulosten käsittelyyn kaavioiden avulla. Mittaamisen vaiheista puhutaan luvussa 4 GQM-menetelmän yhteydessä.

3.2.1 Mittareiden luokittelu

Kuten jo taulukosta 3 nähtiin, mittaaminen voi kohdistua lukuisiin prosessien, tuotteiden sekä resurssien ominaisuuksiin. Seuraavaksi tutustutaankin siihen, minkälaisia luokitteluperusteita mittareilla on ja annetaan esimerkkejä, mitä niillä voidaan mitata. Tässä luvussa esitetty luokittelu on koottu Humphreyn (1989), van Solingenin ja Berghoutin (1999) mukaan.

Objektiiviset ja subjektiiviset mittarit. Objektiivisen mittauksen arvoon ei vaikuta, kuka tiedon kerää. Toisin sanoen objektiivisella mittarilla kukin mittaaja saa samoja arvoja. Esimerkiksi lähdekoodirivien lukumäärän mittaus voidaan suorittaa objektiivisesti. Subjektiiviset mittaustulokset vaihtelevat, sillä ihmisten henkilökohtaiset päätökset ja mieltymykset vaikuttavat mittaustulokseen. Esimerkiksi käyttäjän tekemien virheiden määrää suorituksessa voidaan mitata subjektiivisesti (Humphrey, 1989).

Suorat ja johdetut mittarit. Suoran mittarin tulos ei riipu muiden mittareiden tuloksista. Esimerkiksi tuotteessa olevien virheiden lukumäärän tai ohjelmiston lähdekoodirivien lukumäärän saa mitattua suoraan. Johdetun mittarin tuloksen tulkitsemiseksi tarvitaan vähintäänkin kahden muun mittarin tulokset, jotta johdetun mittarin tulosta pystyttäisiin tulkitsemaan. Tuotteen puutetiheyden mittaaminen tapahtuu johdetusti, sillä sen mittausta varten tarvitaan edellä mainittuja suoria mittareita (Springer, 2001).

Absoluuttiset ja suhteelliset mittarit. Absoluuttinen mittaus on tyypillisesti muuttumaton, vaikka uutta mittaustietoa saadaan kerättyä aikaisempien mittaustietojen lisäksi. Esimerkiksi yhden ohjelman koko on absoluuttinen ja riippumaton muista ohjelmien ko'oista. Suhteellisen mittauksen tulokseen vaikuttaa uusi mittaustieto. Esimerkiksi keskiarvon laskeminen on suhteellista. Objektiiviset mittarit ovat usein absoluuttisia ja subjektiiviset ovat suhteellisia (Humphrey, 1989).

Dynaamiset ja staattiset mittarit. Dynaamiseen mittaukseen liittyy aikatekijä: esimerkiksi, kuinka paljon virheitä ohjelmiston kehityksen aikana on löydetty kutakin kuukautta kohden. Dynaamisen mittauksen arvoilla on taipumus vaihdella. Esimerkiksi virheiden lukumäärä voi vaihdella kuukausien mukaan riippuen, mikä vaihe ohjelmistoprosessin kehityksessä on käynnissä. Esimerkiksi testausvaiheessa on normaalia löytää tavallista enemmän virheitä. Staattinen mittaus on pysyvämpää. Sen avulla voidaan esimerkiksi mitata koko ohjelmistoprosessin aikana löydetyt virheet. (Humphrey, 1989)

Ennustavat ja selittävät mittarit. Ennustavat mittarit voidaan muodostaa etukäteen ja niiden avulla pyritään ennakoimaan asioiden käyttäytymistä. Esimerkiksi puutetiheyttä mittaamalla voidaan ennustaa prosessin eri vaiheiden laatutasoa (Fenton & Pfleeger, 1997). Selittävien mittareiden avulla voidaan havaita vasta jälkikäteen, kuinka asiassa onnistuttiin. Esimerkiksi käytettyjen työtuntien avulla voidaan mitata, pysyikö prosessi oikeassa aikataulussa (Humphrey, 1989).

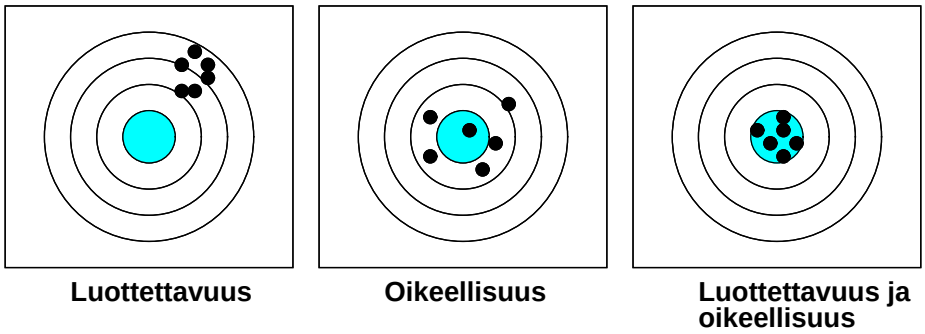
3.2.2 Hyvän mittarin ominaisuudet

Mittaamisen avulla pyritään saamaan kelvollista mittaustietoa, jota voidaan hyödyntää ohjelmistoprosessin kehityksessä. Mittarit tulee suunnitella ja määritellä tarkkaan, jotta mittauksen odotukset täyttyisivät ja mittauksen tulokset eivät olisi harhaanjohtavia.

Kanin (2003) mukaan hyvän mittarin tärkeimmät ominaisuudet ovat luotettavuus (reliability) ja

oikeellisuus (validity). Luotettavuudella tarkoitetaan sitä, että mittaus pyritään tekemään aina samoin menetelmin samoissa olosuhteissa. Täten varmistutaan siitä, että mittauksessa ei esiinny turhaa variaatiota. Luotettavuuden avulla pyritään pääsemään eroon satunnaisista virheistä. Mittauksen automatisoinnilla voidaan esimerkiksi pienentää virheistä aiheutuvaa vaihtelua. Oikeellisuudella tarkoitetaan sitä, että pyritään mittaamaan sitä, mitä on tarkoitus mitata. Oikeellisuuden avulla pyritään pääsemään eroon systemaattisista virheistä. Esimerkiksi ohjelman lähdekoodirivien laskeminen on oikea, täsmällinen mittari ohjelman koon laskemiseksi, mutta se ei sovellu esimerkiksi ohjelman monimutkaisuuden mittaamiseen, sillä ohjelman monimutkaisuus ei ole riippuvainen vain lähdekoodirivien määrästä (Fenton & Pfleeger, 1997).

Kuvassa 3 on hahmoteltu maalitaulussa olevien reikien avulla luotettavuuden ja oikeellisuuden välistä eroa. Sitä parempi luotettavuus on, mitä lähempänä reiät ovat toisiinsa nähden ja sitä oikeellisempi eli tarkempi tulos on, mitä lähempänä reiät ovat maalitaulun keskustaan nähden. Kuvasta voidaan havaita, että luotettavuus ja oikeellisuus eivät ole toisistaan riippuvia ominaisuuksia.



Kuva 3: Luotettavuuden ja oikeellisuuden hahmottaminen maalitaulun avulla (Kan, 2003).

Hyvällä mittarilla tulee olla myös muita toivottavia ominaisuuksia (Humphrey, 1989; Kearney & al. 1986):

- Mittarin pitäisi olla vakaa. Vakaudella tarkoitetaan sitä, että mittaus on toistettavissa oleva, tarkka ja piittaamaton työkalujen, menetelmien tai tuotteen muutoksista.

- Mittarin pitäisi ehdottaa normi. Toisin sanoen mittarin pitää kertoa, millainen arvo on hyvä ja millainen huono. Esimerkiksi nolla on paras arvo, jos mitataan tuotteessa olevien virheiden lukumäärää.
- Mittarin pitäisi ehdottaa parantamisstrategia.
- Mittarin pitäisi olla prosessin luonnollinen oheistuote, sillä prosessin kehitystyön aikana prosessista on helppoa kerätä kohtuullisesti lisätietoa mittausta varten.
- Mittarin pitäisi olla yksinkertainen, sillä monimutkaisia mittareita on vaikeaa käyttää ja tulkita.
- Mittarin pitäisi olla etukäteen ennustettavissa ja jälkikäteen seurattavissa oleva, jolloin mittaajan on helpompi nähdä, kuinka toiminnan muuttaminen parantaa tuloksia.

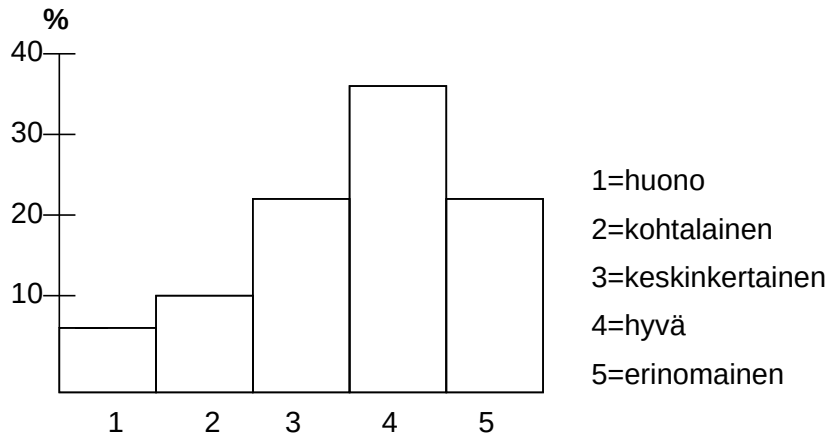
3.2.3 Mittaustulosten havainnollistaminen

Mittaamisen jälkeen alkaa mittaustulosten käsittely, jossa tulokset voidaan havainnollistaa erilaisten kaavioiden avulla.

Histogrammi on yksi yleisimmin käytetty havainnollistamismenetelmä tulosten raportoimiseksi pylväs- (bar) sekä viivakaavion (run chart) ohella. Laadunparantamisen havainnollistamismenetelminä käytetään syy-seurauskaaviota ja tarkastuslistaa, sillä niiden sekä pareto-kaavion avulla ratkaistaan 85 % kaikista laatuongelmista (Arthur, 1993). Tämän kappaleen havainnollistamismenetelmien esitys perustuu Kanin (2003) sekä Arthurin (1993) kirjoihin.

Histogrammi (histogram). Yleisesti käytetty graafinen esitysmuoto, jonka tarkoituksena on antaa yleiskuva mitattavan piirteiden välisistä jakaumista. Kuvassa 4 on histogrammi, joka kertoo tuotteen asiakastyytyväisyyden jakautumisen. Mitta-asteikon tulokset: huono (1), kohtalainen (2), keskinkertainen (3), hyvä (4) ja erinomainen (5) ovat ilmaistu x-akselilla ja y-akseli kertoo kunkin ominaisuuden prosenttiosuuden. Histogrammia käytetään yleisesti järjestyksen, välin

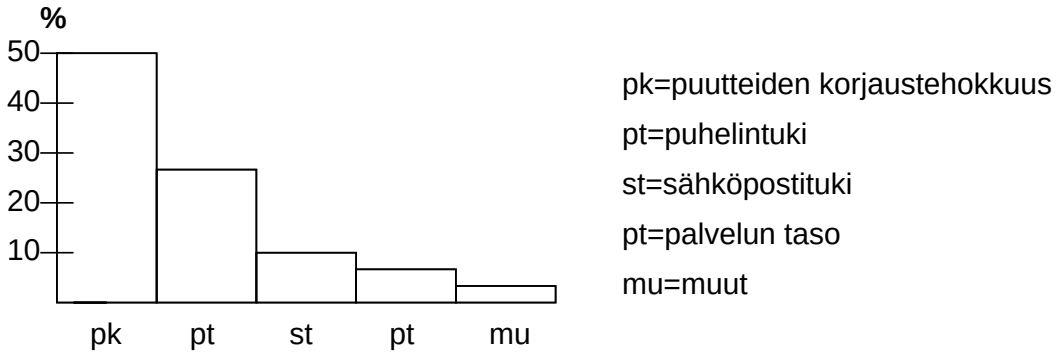
sekä suhteen ilmaisevien mittaustulosten esittämiseen. Nominaalisten mittaustulosten esittämiseen käytetään yleensä pylväskaaviota, jonka avulla voidaan korostaa mittaustulosten luokittelua mutta ei paremmuusjärjestystä.



Kuva 4: Asiakastyyvyväisyysjakauman kuvaus histogrammin avulla (Kan, 2003).

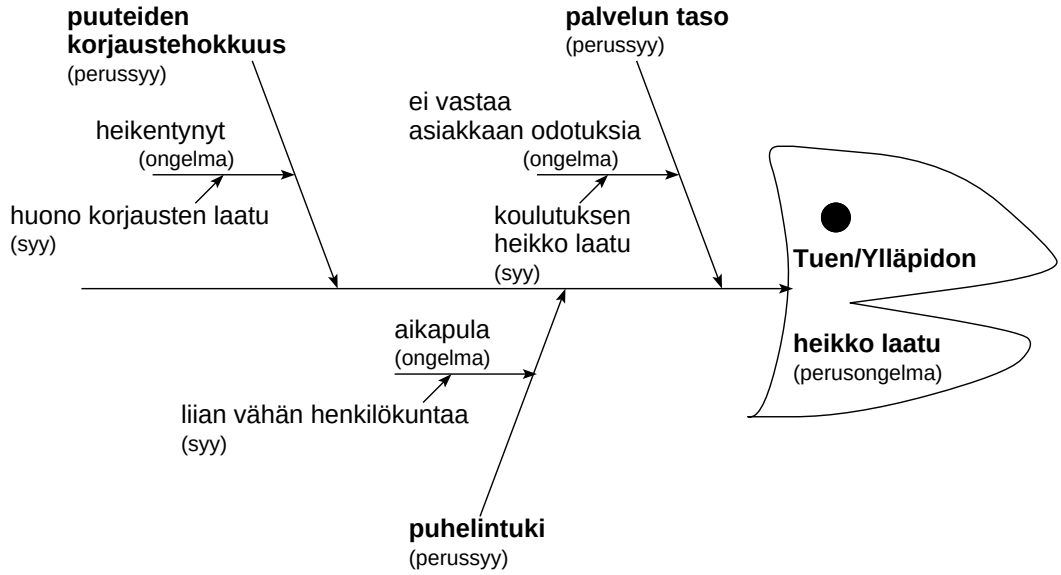
Pareto-kaavio (pareto diagram). Pylväskaaviota muistuttava luokitteluun soveltuva kaavio, jonka tarkoituksena on auttaa ongelma-alueiden tunnistamisessa. Pareto-kaaviota käytetään muun muassa ohjelmistoprosessin eri vaiheiden puutteiden paikantamisessa. Pareto-kaavion perusajatuksena on se, että suurin osa puutteista ryhmittyy pienelle alueelle. Puutteet luokitellaan pareto-kaavioon suuruusjärjestyksessä suurimmasta pienimpään. Täten pareto-kaavion avulla voidaan helposti havaita, mitkä ovat pahimmat puutteet ja mihin kohdin ne ovat keskittyneet, jolloin niiden paikantaminen sekä korjaaminen helpottuvat. Kuvassa 5 on pareto-kaavio, joka kertoo asiakkaan näkökulmasta havaitut puutteet tukiprosessin ja ylläpidon aikana.

Syy-seurauskaavio (cause-effect-diagram). Erittäin yleisesti käytetty kaavio ohjelmistoprosessin kehittämisessä, joka tunnetaan myös kalanruotokaaviona. Kaavion avulla on tarkoitus etsiä syiden ja seurauksien välisiä suhteita. Esimerkiksi pareto-kaavion avulla voidaan havaita, että tukiprosessissa ja ylläpidossa on paljon puutteita, jolloin syy-seurauskaavion avulla etsitään syitä kyseisiin ongelmiin. Syiden etsimisessä voidaan käyttää apuna aivoriihtä, jossa tarkoin valittu työryhmä pyrkii selvittämään, mistä syistä puutteet johtuvat. Kuvaan 6 on hahmoteltu



Kuva 5: Asiakkaan ilmoittamat puutteet pareto-kaaviossa tukiprosessin ja ylläpidon aikana (Arthur, 1993).

syy-seurauskaavion idea. "Kalan päänä" on perusongelma, tukiprosessin ja ylläpidon heikko laatu, johon pyritään etsimään syitä "ruotojen" avulla.



Kuva 6: Tukiprosessin ja ylläpidon puutteiden selvittäminen syy-seurauskaavion avulla (Arthur, 1993).

Tarkastuslista (checklist). Havaintojen keräämis- ja luokittelumenetelmä, jolla on merkittävä rooli ohjelmistoprosessin kehittämisessä ja laadunparantamisessa. Sen avulla voidaan kerätä

kehityksen aikana havaitut puutteet tai tarkistaa, onko kaikki suunnitellut asiat muistettu toteuttaa prosessin aikana. Tarkastuslistaa tulee käyttää ja seurata päivittäin, jolloin siitä tulee tärkeä osa prosessia. Mikäli tarkastuslista on koottu kehitystiimin kokemusten perusteella, se toimii apuna prosessin vakauden ja hallinnan kehittämisessä. Kuvassa 7 on esimerkki tarkastuslistasta, jota voidaan täydentää esimerkiksi "tukkimiehen kirjanpidolla".

	tammi	helmi	maalis	huhti	touko	kesä	heinä	elo	syys	loka	marras	joulu	yht
Tuki/Ylläpito	||	||	|						||				**7**
Ohjelmisto	|||	|		|				||			||	|	**10**
Dokumentaatio	𝍸				|||						|		**9**
Koulutus	||					|||				|		**6**
Muut			|			|	|	|	||			|	**7**
Yhteensä	**12**	**3**	**2**	**1**	**3**	**4**	**1**	**3**	**4**	**1**	**3**	**2**	**39**

Kuva 7: Asiakkaan ilmoittamat ohjelmistokehitysprosessin puutteet ilmaistuna tarkastuslistan avulla (Arthur, 1993).

Laadunhallintaan ja -parantamiseen voidaan käyttää myös muita havainnollistamismenetelmiä, kuten viiva-, piste- (scatter diagram) ja kontrollikaavioita (control charts). Viivakaavion avulla nähdään muuttujan kehityssuunta, jossa x-akseli kuvaa aikaa ja y-akseli haluttua muuttujaa, kuten viikoittaisten virheraporttien määrää. Pistekaavion avulla voidaan selvittää kahden muuttujan välinen riippuvuus. Kontrollikaavio on viivakaavion kehittyneempi versio ja se sisältää vaihteluvälin, jonka sisälle muuttujan arvojen tulee sijoittautua. Mikäli jokin muuttujan arvo menee arvoalueen ulkopuolelle, parannustoimenpiteet ovat tarpeellisia, sillä kontrollikaavion seurannan avulla prosessi pyritään pitämään hallinnassa. Kontrollikaavio on vaativa työkalu, jonka käyttö vaatii tilastotieteen tuntemista. Kontrollikaavioita käytetään enemmän tuotannon puolella, jossa prosessit ovat vakaampia ja helpommin ennustettavissa olevia kuin ohjelmistotuotannossa.

Pienen tai uuden kehitystiimin on hyvä aloittaa laadunkehittäminen kolmella yleisimmin käytetyllä havainnollistamismenetelmällä (pareto-kaavio, syy-seurauskaavio ja tarkastuslista), sillä ne soveltuvat kaikille ja ne ovat helppokäyttöisiä.

3.3 Mittaamisen ongelmat

Ennen mittaamisen vaiheiden läpikäyntiä on hyvä tietää hiukan mittaamisen ongelmista, jotta ymmärrettäisiin mittaamisen olevan vaativa prosessi, jossa pitää huomioida niin tekniset kuin inhimilliset asiat. Mittaaminen jo itsessään on haasteellinen prosessi, sillä mittauksen onnistuminen vaatii oikeiden mittareiden löytämistä ja niiden tulkintaa. Mittaamisessa täytyy huomioida myös henkilöt, jotka suorittavat mittauksen. He saattavat olla haluttomia mittaustoimenpiteille, koska he eivät välttämättä tiedä, miksi se on tärkeää (Wiegers, 1997).

Jotta mittaaminen voisi onnistua, tulee ylemmän johdon olla sitoutunut siihen. Ellei johto ole huolella sitoutunut mittaamiseen, se ei vaadi työntekijöiltäkään riittävää panostusta. Jotta ylempi johto kiinnostuisi mittaamista, heille tulee kertoa, mitä apua mittaamisesta on. Tällöin mittaamisen vieminen työntekijöiden keskuuteen on helpompaa. Kun ihmiset saadaan innostumaan mittaamisesta, vaarana voi olla, että asioita mitataan liikaa ja liian nopeasti. Tällöin mittaustietoa kertyy enemmän kuin sitä ehditään analysoida, jolloin osa tiedosta menee hukkaan. Mittaamisesta pitää tehdä oma kulttuurinsa vähitellen ja aluksi tulisikin valita vain muutamia työskentelyä täydentäviä, kuten aikataulu- tai laatumittareita. Kun ihmiset oppivat, miten mittaustietoa kerätään ja käsitellään, voidaan mittareiden määrää lisätä. Myös liian vähäisessä mittauksessa on omat ongelmansa. Esimerkiksi jos mitataan pelkkää tuottavuutta, ohjelmoijat voivat alkaa ohjelmoimaan mahdollisimman nopeasti, jotta tuottavuus olisi suurempi. Tuottavuuden ohella onkin hyvä mitata samalla laatua, jotta ohjelmoijat eivät pyrkisi kaunistelemaan mittaustuloksia huonolaatuisella lähdekoodilla.

Ennen mittaamisen aloittamista tulee määritellä yhteiset toimintatavat sekä määritelmät, jotta kukin mittaaja kerää mittausaineiston samalla tavalla. Mittausta varten tulee suunnitella lomakkeet, joiden avulla mittausaineisto kerätään. Ennen varsinaisen mittaamisen aloittamista sekä mittarit että mittausmenetelmät tulee testata pilottitestien avulla (Kan, 2003, Wiegers, 1997).

Yksi mittaamisen yleisimmistä ongelmista on, että työtekijät ovat haluttomia keräämään mittausaineistoa, sillä he eivät tiedä, kuinka mittaaminen vaikuttaa heihin (Haikala & Märijärvi,

2000). Mittaamisen merkityksen ymmärtäminen on tärkeää, koska vain silloin voidaan saada kerättyä luotettavaa mittaustietoa. Työntekijät tulee kouluttaa oikeaoppiseen mittausaineiston keräämiseen. Heille tulee myös kertoa, että kerättyä aineistoa ei käytetä työntekijän palkitsemiseen tai rankaisemiseen. Muutoin työntekijä saattaa lopettaa tiedon keruun tai pahimmassa tapauksessa vääristelee tietoa, jotta hänen henkilökohtaiset ansionsa näyttäisivät paremmilta. Mikäli mittaus liittyy henkilökohtaisiin asioihin, kuten lähdekoodin tuottavuuteen, tulisi mittausaineisto säilyttää salassa. Tällöin vain asianomaiset voivat käyttää mittausaineistoa analysoinnin teossa. Työntekijän esimiehen ei tarvitse tietää työntekijöiden henkilökohtaisia paremmuusarvoja.

Työntekijöille on perusteltava, miksi kukin mittari on tarpeellinen ja miten sitä käytetään. Mikäli mittareiden laatijan on hankala vastata työntekijöiden esittämiin kysymyksiin, kannattaa hänen pohtia, onko kyseinen mittari todella tarpeellinen. Lisäksi työntekijöille on kerrottava mittauksen tulokset ja annettava mahdollisuus tutkia kerättyä mittaustietoa jo mittauksen aikana, paitsi henkilökohtaisia mittaustietoja. Tällöin heille ei jää epäselväksi, mitä merkitystä heidän keräämällä mittaustiedolla on, vaan he ymmärtävät, miten mittaamisen avulla on tarkoitus parantaa mitattavaa prosessia (Haikala & Märijärvi, 2000).

Mittaustulosten analysointivaiheessa tulee olla maltillinen, jotta säästyttäisiin harhaanjohtavilta toimilta. Mittaustulosten tulkinnassa pitäisi kiinnittää huomiota enemmän tulosten kehityssuuntaan kuin yksittäisiin, poikkeaviin arvoihin. Mikäli keskitytään vain yksittäisen puuteiden korjaamiseen hahmottamatta kokonaiskuvaa, korjaustoimenpiteet saattavat huonontaa mitattavan kohteen laatua. Mittauksessa esiintyy aina variaatiota, joten yksittäiset poikkeamat ovat luonnollisia mittaustuloksia tarkasteltaessa. Kehityssuunta kertoo paremmin, miten mitattava prosessi etenee. Ennen kuin muutoksia tehdään mitattavaan kohteeseen, tulee varmistua, että kehityssuunnan merkitys on ymmärretty oikein.

Yksi mittaamisen suurimmista ongelmista on se, kuinka mittarit tulisi valita, jotta niiden avulla saataisiin mitattua juuri sitä, mitä halutaan. Tähän ongelmaan on havaittu hyvä, toimiva

menetelmä: *GQM*, tavoite/kysymys/mittari -menetelmä (Goal/Question/Metric -method) (Wiegers, 1997). GQM-menetelmä on tavoitekeskeinen lähestymistapa mittareiden etsimiseen, jossa määritellään ensimmäiseksi mittaamisen tavoitteet, kuten asiakastyytyväisyyden parantaminen. Mittaajan, joka käyttää apunaan GQM-menetelmää, ei tule kysyä "mitä mittareita minun pitäisi käyttää?" vaan "mitä haluan tietää ja oppia?" (Rombach & Ulery, 1989).

Ottamalla huomioon mittaukseen liittyvät ongelmat mittaamisen riskit pienevät ja prosessin parantaminen helpottuu. Pelkkä mittaaminen ei itsessään paranna ohjelmistoprosessia, mutta se antaa lähtökohdat paremman ohjelmistoprosessin kehittämiseksi. Oikein käytettynä mittaaminen on voimakas menetelmä ohjelmistoprosessin kehittämisessä.

4 TAVOITEKESKEINEN MITTAAMINEN

Mittaaminen on olennainen osa tehokasta ohjelmistoprosessin parantamista, sillä mittaamalla oikeita asioita voidaan saada selville, kuinka ohjelmistoprosessia tulisi jatkossa kehittää parempaan suuntaan. Oikeiden asioiden mittaaminen ei ole aina helppoa, sillä esimerkiksi mittaamistavoitteiden asettaminen voi olla hankalaa, jollei tavoitteiden asettaminen perustu johonkin systemaattiseen menetelmään. Tätä varten on kehitetty tavoitekeskeinen GQM-menetelmä, jotta ohjelmistoprosessia voitaisiin mitata mielekkäällä ja tarkoituksenmukaisella tavalla.

Tässä luvussa keskitytään GQM-menetelmään, jota voidaan hyödyntää prosessien, tuotteiden sekä resurssien systemaattisten parantamistavoitteiden etsimisessä. GQM-menetelmän perusideana on johtaa liiketoiminnan tavoitteista mittaustavoitteet eli asiat, joita halutaan tutkia. Tavoitteiden pohjalta laaditaan kysymyksiä, joiden avulla pyritään selventämään sitä, mitä tavoitteilta halutaan. Lopuksi etsitään tai kehitetään sopivat mittarit, joista saatavan tiedon avulla voidaan vastata kysymyksiin ja sitä kautta saada selville, onko asetetut tavoitteet saavutettu.

Tässä luvussa tutustutaan ensiksi GQM-menetelmään yleisellä tasolla, jonka jälkeen läpikäydään GQM-menetelmän eri vaiheet: suunnittelu, määrittely, tiedonkeruu ja tulkinta. Luvussa 5 keskitytään konkreettisemmin siihen, kuinka asiakastyytyväisyyden parantamistavoitteet voidaan jalostaa GQM-menetelmän avulla mittareiksi. Lopuksi tässä luvussa kerrotaan GQM-menetelmän ongelmista sekä GQM-menetelmää täydentävistä ratkaisuista.

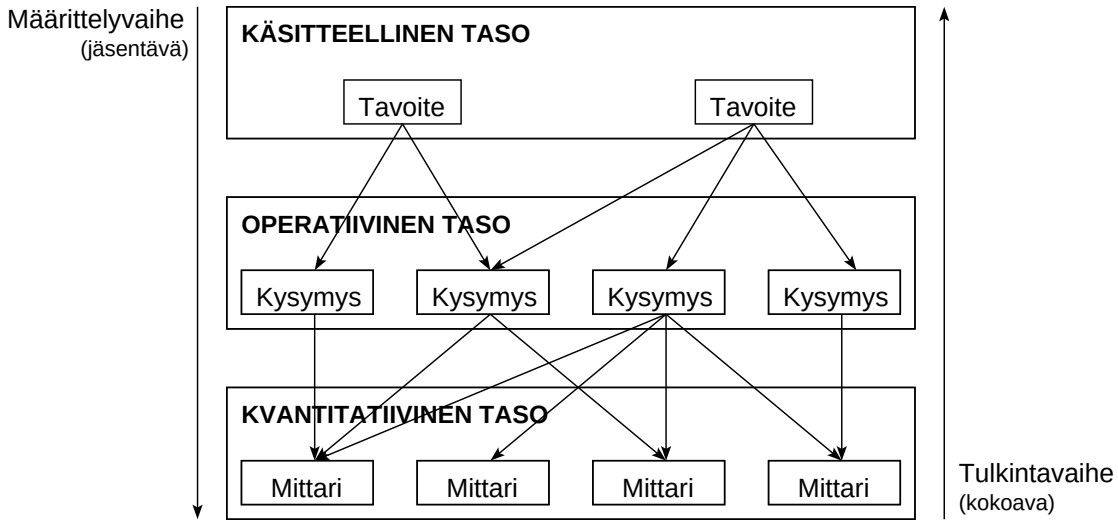
4.1 GQM-menetelmä

Basili ja Weiss suunnittelivat vuonna 1984 GQM-menetelmän NASA:n käyttöön (Basili & Weiss, 1984; Ebert, 2005; Zuse,1998). Menetelmän avulla oli tarkoitus arvioida, kuinka paljon tietyt NASA:n projektit sisälsivät puutteita. Menetelmän aikana tehtiin paljon kokeita ja menetelmää laajennettiin, jotta se soveltuisi paremmin eri lähtökohtiin. Sitä onkin myöhemmin käytetty esimerkiksi *laadun parantamisen paradigman* (QIP, Quality Improvement Paradigm)

tukena, jonka tarkoituksena on parantaa laatua jatkuvasti projektin ja organisaation kokemusten perusteella. GQM-menetelmää muovattiin 1990-luvulla, jotta se soveltuisi paremmin ohjelmistomittaajille, jotka pyrkivät löytämään sopivat mittarit liiketoiminnan tavoitteiden mittaamiseen (Basili & al., 1994b; Goldpractices, 2005).

GQM-menetelmän avulla on tarkoitus saada parempi ymmärrys nykyisten prosessien, tuotteiden sekä resurssien tilasta. Lisäksi sen avulla voidaan valvoa ja ohjata ohjelmistoprosesseja, arvioida uusien teknologioiden sopivuutta sekä löytää parantamiskohteita. GQM-menetelmää voidaan hyödyntää sekä johdon että työntekijöiden tasolla (Goldpractices, 2005).

GQM-menetelmä perustuu sekä jäsentävään (top-down) että kokoavaan (bottom-up) lähestymistapaan. Jäsentävän lähestymistavan avulla tavoitteista jalostetaan vähitellen sopivat mittarit liiketoiminnan tavoitteiden mittaamiseen. Kokoavan lähestymistavan tarkoituksena on koota mittareista saatu tieto ja selvittää, onko tavoitteisiin päästy. Kuvassa 8 on hahmoteltu GQM-mallin perusidea, josta näkyy sekä jäsentävä että kokoava lähestymistapa.



Kuva 8: GQM-mallin perusidea (Basili & al., 1994b; van Solingen & Berghout, 1999).

Kuvan 8 GQM-malli koostuu kolmesta hierarkkisesta tasosta: käsitteellistä, operatiivisesta ja

kvantitatiivisesta (Basili & al., 1994b). Hierarkkisten tasojen avulla pyritään varmistamaan, että mittausprosessin aikana keskitytään oikeisiin mittareihin, jolloin ylimääräiseltä mittaustyöltä vältyttäisiin (Goldpractices, 2005).

Käsitteellisellä tasolla organisaatio määrittää mittaamisen tavoitteet, joita se haluaa mitata. Tavoitteesta tulee ilmetä mittauksen kohde, eli kohdistuuko mittaus prosesseihin, tuotteisiin vai resursseihin. Lisäksi tulee tietää mittaamisen tarkoitus, onko se oppiminen, ymmärtäminen, arvioiminen, ennustaminen, ohjaaminen vai parantaminen (Fenton & Pfleeger, 1997; Park & al, 1996; Pressman, 2000; van Solingen & Berghout, 1999; van Solingen & Berghout, 2001). Sen jälkeen täytyy selvittää, mihin ongelma kohdistuu, ja tarkastellaanko asiaa esimerkiksi johdon, projektitiimin vai asiakkaan näkökulmasta. Lopuksi pitää selvittää ympäristö, kuten projekti, jossa mittaus tapahtuu (Goldpractices 2005).

Operatiivisella tasolla tavoitteista johdetaan kysymykset, jotka pyrkivät hienojakoistamaan tavoitteet selkeiksi, pienemmiksi, mitattaviksi komponenteiksi. Kysymykset suunnitellaan siten, että ne kuvaavat mittauskohdetta valitun ongelman näkökulmasta.

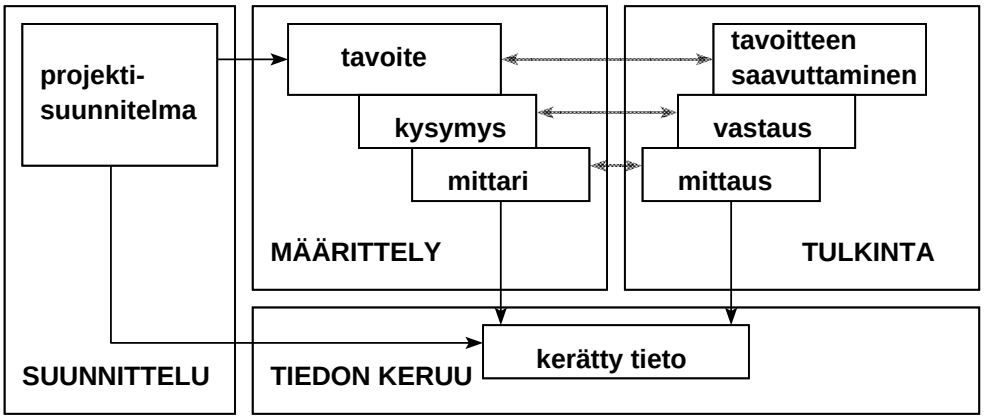
Kvantitatiivisella tasolla kysymyksien pohjalta määritellään mittarit, jotka pyrkivät vastaamaan kysymyksiin. Kaikki kysymykset jalostetaan yhdeksi tai useammaksi mittariksi, jotka voivat olla joko objektiivisia tai subjektiivisia. Mittarin ei tarvitse olla yhteen kysymykseen sidottu, vaan sitä voidaan käyttää vastaamaan useampaan eri kysymykseen.

GQM-menetelmä esittää systemaattisen lähestymistavan liiketoiminnan tavoitteiden mittaamiseksi. Sitä onkin onnistuneesti käyttänyt monet eri organisaatiot, kuten NASA, Motorola, HP ja AT&T (Baldassarre & al., 2003).

4.2 GQM-menetelmän vaiheet

Tavoitekeskeisen mittaamisen, kuten GQM:n, tärkeimpänä tavoitteena voidaan pitää selkeiden mittaamistavoitteiden asettamista. Mittaamalla tarkoin valittuja tavoitteita prosessin ongelma-

kohdat voidaan tunnistaa ja samalla asettaa parantamistavoitteet, jolloin todellinen prosessin parantaminen voi alkaa.

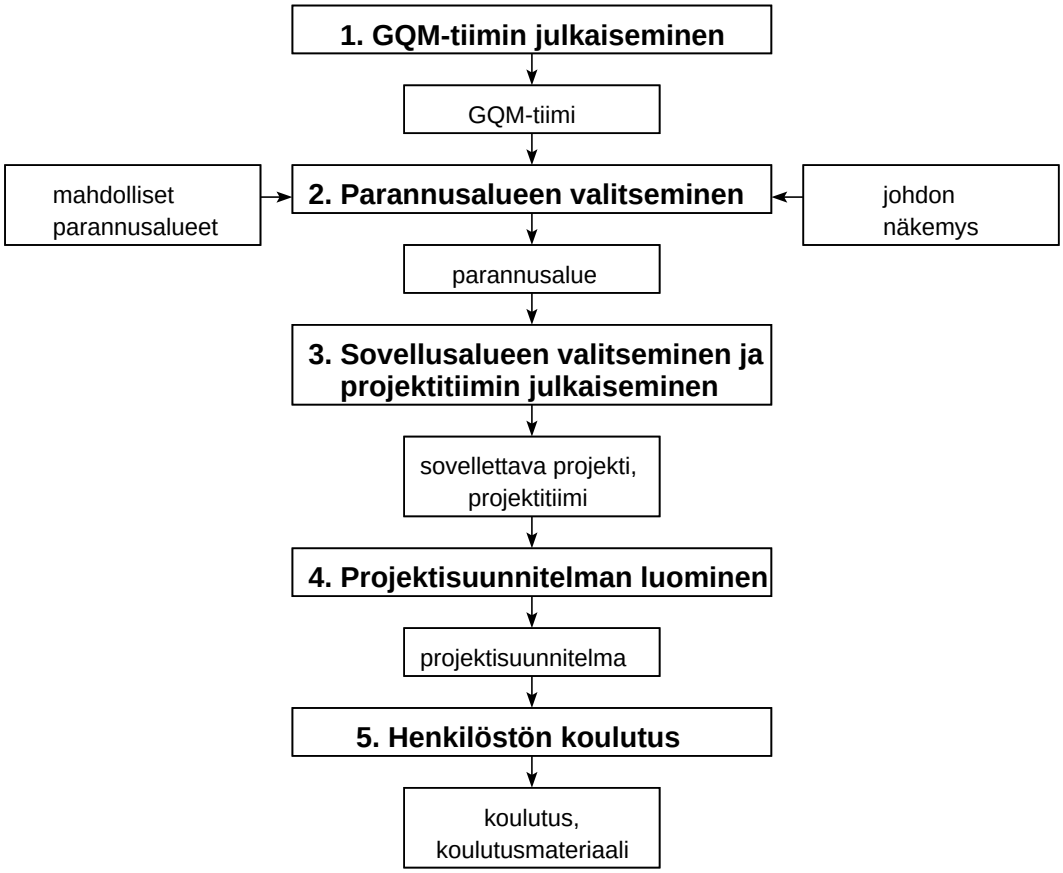


Kuva 9: GQM-menetelmän eri vaiheet (Ebert & al., 2005; van Solingen & Berghout, 1999).

Kuvassa 9 on esitelty GQM-menetelmän vaiheet, jotka koostuvat suunnittelu-, määrittely-, tiedonkeruu- ja tulkintavaiheista. Suunnitteluvaiheen aikana valitaan mittausprosessiin tarvittavat henkilöt ja kerätään mittaukseen vaadittavia esitietoja, jotka esitetään projektisuunnitelmassa. Määrittelyvaiheen tarkoituksena on johtaa tavoitteista kysymykset ja kysymyksien pohjalta mittarit, joiden avulla pyritään keräämään tarvittavat mittaustiedot tiedonkeruuvaiheessa. Kerättyjä mittaustietoja analysoidaan tulkintavaiheessa, jonka tarkoituksena on selvittää, voidaanko mittareiden avulla vastata kysymyksiin ja sitä kautta saada selville, onko halutut tavoitteet saavutettu.

4.2.1 Suunnitteluvaihe

Suunnitteluvaiheen (planning phase) päätarkoituksena on kerätä vaadittavat esitiedot mittauksen läpiviemiseksi sekä motivoida henkilökuntaa tulevaa mittausta varten. Suunnitteluvaiheen aikana tuotetaan projektisuunnitelma, jossa kuvataan alustavat kustannukset, jotka uskotaan menevän mittausprosessin toteuttamiseen, sekä realistinen suunnitelma mittausprosessin tulevista vaiheista. Kuvassa 10 on esitelty suunnitteluvaiheen eteneminen.



Kuva 10: Suunnitteluvaiheen eteneminen (van Solingen & Berghout, 1999).

Suunnitteluvaiheen ensimmäisenä tehtävänä on julkaista GQM-tiimi, joka huolehtii mittausprosessin etenemisestä (Basili & al., 1994b). GQM-tiimillä on ratkaiseva rooli mittauksen onnistumisen kannalta ja siksi GQM-tiimin jäsenten tulee olla oman alansa asiantuntijoita (Goldpractices, 2005).

GQM-tiimin päätehtäviin kuuluu mittausprosessin suunnittelu, määrittely, projektitiimin jäsenten haastattelu, tiedonkeruumenetelmien tarkastaminen, palautetilaisuuden järjestäminen, jossa mitattu tieto analysoidaan sekä edistymisen raportointi ja tulosten levittäminen projektitiimille sekä johdolle (van Latum & al., 1998; van Solingen & Berghout, 1999).

Basili & al. (1994a) esittivät "The Experience Factory" -idean, jonka ajatuksena on luoda loo-

ginen sekä fyysinen organisaatio tukemaan projektin kehitystä. Organisaatio koostuu projektitiimistä, GQM-määrittelytiimistä sekä GQM-analysointitiimistä, jossa GQM-määrittelytiimin tehtävänä on muuttaa liiketoiminnan tavoitteet mittareiksi. GQM-analysointitiimi tehtävänä on käsitellä mittaustieto ja valmistella palautetilaisuudet, joissa mittaustietoa tulkitaan. Van Solingenin ja Berghoutin mukaan GQM-tiimien välistä erottelua ei tarvitse tehdä, sillä heidän mukaansa GQM-tiimit voi korvata yhdellä yhteisellä GQM-tiimillä, koska se on käytännöllisempää, jolloin muun muassa GQM-tiimin roolin ymmärtäminen helpottuu. Siksi tässä tutkielmassa keskitytäänkin jatkossa vain yhden GQM-tiimiin tehtäviin.

Suunnitteluvaiheen toisena tehtävänä on parannusalueen valitseminen. Parannusalueet yleisesti kohdistuvat joko kustannusten, ajankäytön, riskien vähentämiseen tai laadun parantamiseen. Parannusalueen tulisi olla johdettu liiketoiminnan tavoitteista. Tällöin johto hyväksyy parantamishankkeen helpommin (Park & al, 1996).

Ebertin & al. (2005) mukaan ohjelmistoprosessin parantamistavoitteiden asettaminen voi olla hyvinkin vaikeaa ja ohjelmistoprosessia onkin mahdollista parantaa ilman parantamistavoitteiden asettamista tai liiketoiminnan tavoitteita tuntematta. Parantamistavoitteet voidaan tunnistaa esimerkiksi arvioimalla ohjelmistoprosessin kypsyyttä SPICE:n tai CMMI:n avulla. Useasti ei kannata käyttää vain yhtä menetelmää ohjelmistoprosessin parantamiseen, vaan yhdistää eri menetelmiä keskenään. Esimerkiksi GQM:n ja CMMI:n yhdistämisellä on helpompi saada kokonaisvaltaisempi kuva siitä, mitä jatkossa kannattaa mitata, sillä GQM vastaa siihen, miksi ominaisuutta mitataan ja CMMI kertoo, onko yrityksillä edellytyksiä mitata sitä mielekkäällä tavalla.

Arviointien lisäksi GQM-tiimi voi keskittyä yksittäisissä prosesseissa havaittuihin ongelmiin haastattelemalla projektitiimin jäseniä. Yleensä henkilöt, jotka joutuvat tekemisiin ongelmien kanssa, ovat parhaita haastatteluehdokkaita, (Park & al, 1996) sillä he ovat motivoituneita ongelmienratkaisijoita, koska ongelmat vaikuttavat heidän toimintaansa. Mikäli haastattelujen avulla ei pystytä valitsemaan sopivia parannusalueita, voidaan järjestää aivoriihi, jossa kartoi-

tetaan parannusalueita.

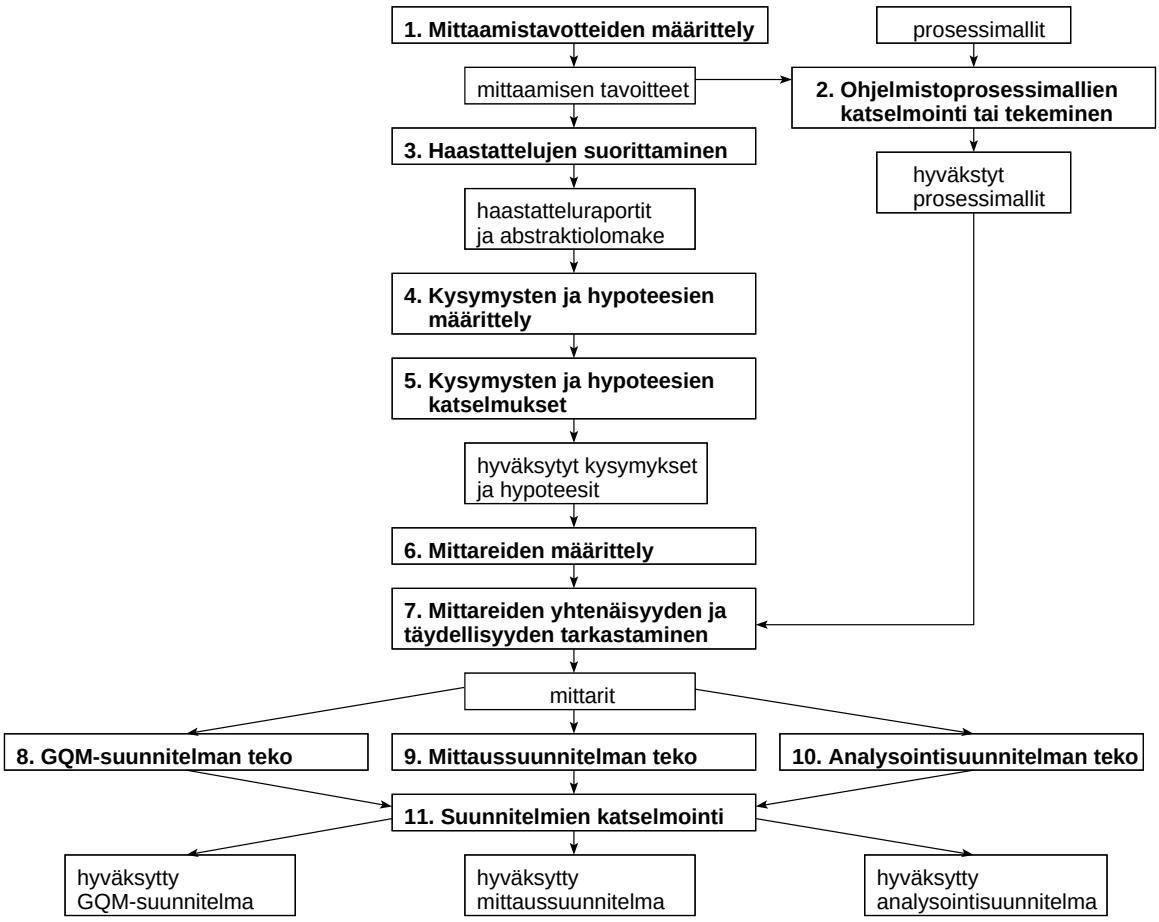
Kun parannusalue on valittu, GQM-tiimi valitsee sovellettavan projektin ja projektitiimin, joka kerää mitattavan tiedon. Tämän jälkeen GQM-tiimi luo projektisuunnitelman, projektitiimin tietojen perusteella. Projektisuunnitelman tärkeimpänä tehtävänä on esittää näkemys mittausprosessista ja saada johdon hyväksyntä mittaamiseen. Projektisuunnitelman kaksi tärkeintä osa-aluetta ovat kustannusmallin teko, joka on esitetty tarkemmin liitteessä 1, ja mittausprosessin realistinen suunnitelma, josta pitää käydä muun muassa ilmi mittausprosessin yleiskuvaus, aikataulu, henkilöt sekä heidän vastuualueensa mittausprosessissa, prosessin hallintaperiaatteet ja kuinka henkilöt koulutetaan mittausprosessia varten.

Lopuksi GQM-tiimi suorittaa projektitiimin koulutuksen. Tässä vaiheessa GQM-tiimin merkitys korostuu, jotta se saisi projektitiimin jäsenet ymmärtämään mittaamisen tärkeyden ja sen, että projektitiimi on avainasemassa mittaamisen onnistumisen kannalta. Projektitiimi on kiinnostunut kuulemaan koulutustilaisuuksissa käytännön asioita: millaisia mittaustehtäviä he joutuvat suorittamaan, miksi ne pitää suorittaa, milloin ne suoritetaan, kuinka paljon ne vaativat aikaa, miten ne vaikuttavat muihin tehtäviin, ja mitä mittaamisesta voi hyötyä ja oppia. Tämän lisäksi projektitiimin tulee ymmärtää GQM-menetelmän toiminta, jotta he ymmärtävät, mitä heiltä edellytetään menetelmän eri vaiheissa (van Solingen & Berghout, 1999). Kun suunnitteluvaiheen tehtävät on suoritettu, siirrytään määrittelyvaiheeseen.

4.2.2 Määrittelyvaihe

Määrittelyvaiheen (definition phase) päätarkoituksena on määritellä tavoitteet, joista johdetaan kysymykset ja kysymysten pohjalta mittarit. Määritelmät dokumentoidaan GQM-, mittaus- ja analysointisuunnitelmiin. Kuvassa 11 on esitelty määrittelyvaiheen eteneminen.

Määrittelyvaiheen ensimmäisenä tavoitteena on määritellä mittauksen tavoitteet (goal, G), jotka GQM- sekä projektitiimi johtavat yhdessä liiketoiminnan tavoitteista. Mikäli projektitiimillä

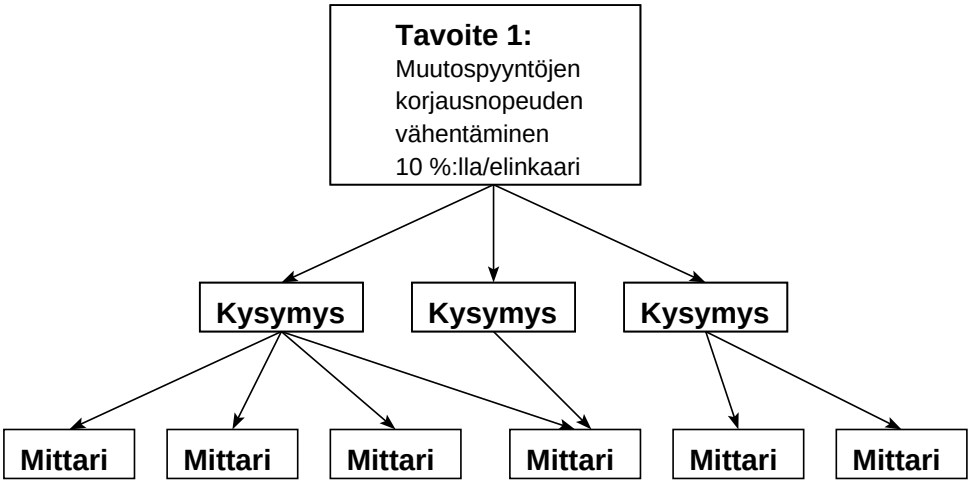


Kuva 11: Määrittelyvaiheen eteneminen (van Solingen & Berghout, 1999).

on vähän kokemusta tavoitteiden asettamisesta, voi GQM-tiimi tehdä sen. Tällöin projektitiimi osallistuu tavoitteiden tarkastustilaisuuteen, jossa sen jäsenet voivat keskustella tavoitteista (van Solingen & Berghout, 1999).

Kuvassa 12 on esitetty esimerkki siitä, millainen tavoite mittaamisella voisi olla.

Mittaamisen tavoitteet täytyy määritellä selkeästi ja kustakin tavoitteesta pitää tulla ilmi mitattava kohde (esimerkiksi muutospyyntöjen käsittely), mikä mittauksen tarkoitus on (esimerkiksi parantaminen), mihin ongelma kohdistuu (esimerkiksi muutospyyntöjen vaikutus projektin elinkaaren kestoon), kenen näkökulmasta asiaa tarkastellaan (esimerkiksi projektipäällikön) ja ympäristö, jossa mittaus tapahtuu (esimerkiksi projektin nimi) (Briand & al., 2002; Goldprac-



Kuva 12: Esimerkki tavoitteista (Goldpractices, 2005).

tices, 2005; Lavazza & Barresi, 2005).

Mittaamisen tavoitteiden määrittäminen on yleisesti vaikeaa. Seuraavien kysymysten avulla on pyritty helpottamaan tavoitteiden tunnistamista (van Latum & al., 1998; van Solingen & Berghout, 1999): Mitkä ovat organisaation strategiset tavoitteet? Mitkä asiat ovat vaikuttaneet strategisiin tavoitteisiin? Miten suorituskykyä voi parantaa? Mitkä ovat päähuolenaiheet? Mitkä ovat parantamistavoitteet? Miten parantamistavoitteet voi saavuttaa? Mitkä ovat mahdolliset mittaustavoitteet ja mitkä ovat niiden prioriteetit?

Kun mittaamisen tavoitteet ovat selvitetty, voidaan ohjelmistoprosessimalleja hyödyntää mittareiden tunnistamisessa. Jos organisaatiolla ei ole sopivia malleja, jotka määrittelevät mittaamisen tavoitteita, täytyy GQM-tiimin kehittää ne. On tärkeää muistaa, että mallien tulee kuvata, kuinka työt täytyy todellisuudessa tehdä, ei sitä, kuinka ne ihanteellisissa olosuhteissa tulisi tehdä (van Solingen & Berghout, 1999). Tämän jälkeen GQM-tiimin tulisi suorittaa haastattelut muuttamalla projektitiimin jäsenten mittaustavoitteisiin liittyvä implisiittinen, hiljainen tieto eksplisiittiseksi, ymmärrettäväksi tiedoksi (van Latum & al., 1998).

Tietojen saamiseksi GQM-tiimi haastattelee yksitellen kutakin projektitiimin jäsentä. Haastatteluissa voidaan käyttää apuna *abstraktiolomaketta* (abstraction sheet). Sitä voidaan käyttää

tiedonkeruuvälineenä, jonka avulla pyritään saamaan selville kuva haastateltavien henkilöiden mentaalisista malleista. Mentaaliset mallit ovat henkilön mielikuvia ja oletuksia eri asioita. Niiden avulla luodaan havainnoitavasta tiedoista ja kokemuksista oletuksia, joihin vaikuttavat henkilökohtaiset mielipiteet sekä kulttuuri. Oletukset muovautuvat johtopäätöksiksi, joiden avulla lopulta tehdään toimenpiteitä (Senge & al., 1994). Parkin & al. (1996) mukaan mentaaliset mallit pitäisi tehdä näkyviksi, sillä ne auttavat ongelmien hahmottamista. Lisäksi niiden avulla voidaan jakaa tietoa muiden kanssa. Mentaaliset mallit ovat voimakkaita työkaluja, sillä niiden avulla voidaan saada selville mahdollisia mittariehdokkaita.

Kuvassa 13 on esitetty abstraktiolomake, jonka esimerkit ovat poimittu kuvan 12 mittaustavoitteesta. Abstraktiolomakkeen avulla pyritään selvittämään ylimmällä rivillä oleva mittaamisen kohde, tarkoitus, ongelma, näkökulma ja ympäristö. Abstraktiolomake voidaan jakaa neljään eri osaan, joista ensimmäisen osan tarkoituksena on selvittää haastateltavan henkilön "vaistot" ja muuttaa ne määritelmiksi. Tiedon avulla voidaan rakentaa malleja ja tunnistaa mahdollisia tavoitteita, kysymyksiä, hypoteeseja ja mittareita GQM-suunnitelmaa varten. Seuraavassa osassa pyritään saamaan selville, mitä mieltä haastateltava on prosessin tilasta. Mikäli hänellä ei ole konkreettista tietoa, perustuu tieto hypoteeseihin. Hypoteesien avulla voidaan havaita mittaamisen tärkeys vertailemalla todellisia mittaustuloksia oletettuihin tuloksiin. Tiedon avulla pyritään tarkastamaan mallien paikkansa pitävyyttä sekä asettamaan tavoitearvoja. Kolmannen osan tarkoituksena on selvittää vaihtelutekijät, jolla on vaikutusta ensimmäisen osan tekijöihin. Neljännessä osassa pyritään selvittämään, vaikuttavat kolmannen osan vaihtelutekijät todellisuudessa ensimmäisen osan tekijöihin. Jos haastateltava ei kykene osoittamaan, että tekijöillä on vaikutusta keskenään, tulee haastattelijan hylätä mahdollinen tekijöiden keskinäinen vaikutus (Briand & al., 1996; Differding & al., 1996).

Van Latum & al. (1998) ovat havainneet, että abstraktiolomaketta voidaan käyttää usealla eri tavalla. Ensimmäisenä tapana on, että lomakkeen täyttää GQM-tiimi tai tiimin jäsen yhdessä projektitiimin jäsenen kanssa. Toisena tapana voidaan pitää sitä, että projektitiimin jäsenet opetetaan itsenäisesti täyttämään lomake. Van Latum & al. (1998) pitävät tätä tapaa hieman

Kohde	Tarkoitus	Ongelma	Näkökulma	Ympäristö
Osa 1: Laatuun keskittyminen Mitä mittareita voidaan haastateltavan jäsenen mukaan käyttää tavoitteen mittaamiseen? Esimerkiksi, mikä on jäsenen arvio muutospyyntöjen korjaukseen kuluvasta ajasta.			**Osa 3: Vaihtelutekijät** Mitkä tekijät vaikuttavat haastateltavan mielestä mittareihin? Esimerkiksi, vaikuttaako muutosten laajuus mittareiden arvoihin?	
Osa 2: Perushypoteesit Mikä on haastateltavan tietämys näistä mittareista? Esimerkiksi, mikä on jäsenen arvio muutospyynnön korjausvariaatiosta?			**Osa 4: Vaihtelutekijöiden vaikutus hypoteeseihin** Kuinka haastateltavan mielestä vaihtelutekijät vaikuttavat mittauksiin? Mitä riippuvuussuhteita niiden välillä on? Esimerkiksi, kestääkö laajemman muutospyynnön korjaaminen kauemmin kuin tavanomaisen muutospyynnön korjaaminen.	

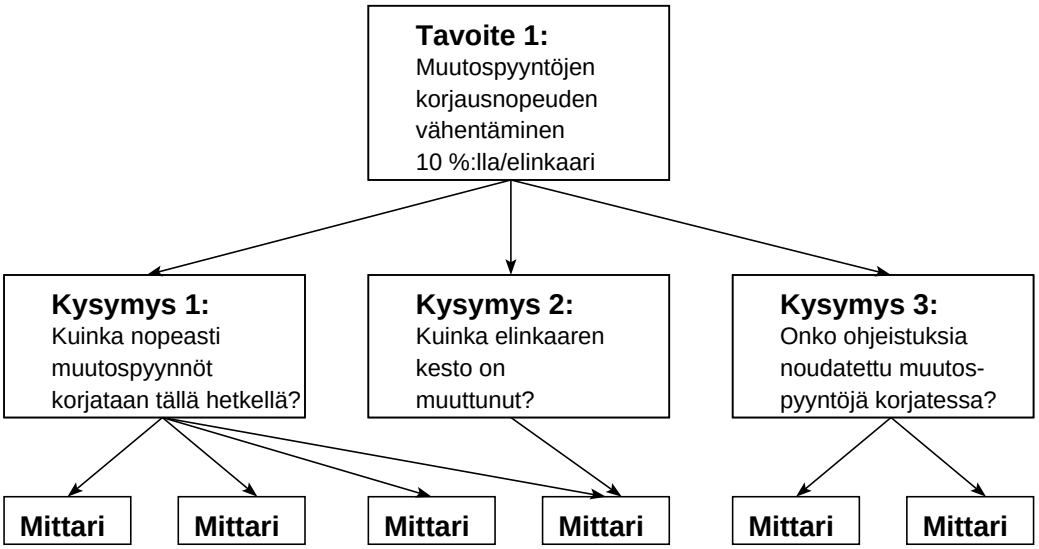
Kuva 13: Abstraktiolomakkeen osa-alueet ja esimerkit niihin liittyvistä kysymyksistä (van Latum & al., 1998; Lavazza & Barresi, 2005).

hankalana, koska lomakkeen osa-alueiden erottaminen toisistaan tuotti projektitiimin jäsenille vaikeuksia, lisäksi opetteluun kului investointeja. Kolmantena tapana on, että GQM-tiimi täyttää lomakkeen etukäteen ennen haastattelua, jolloin lomake toimii haastattelun luonnoksena. Tällöin tulee varoa ettei haastattelija johdattele haastateltavaa omien mieltymystensä mukaisesti. Neljäntenä vaihtoehtona on käyttää abstraktiolomaketta muistilistan tavoin. Van Latum & al. (1998) käyttivät lomaketta apuna mittaustiedon tulkinnassa ja palautetilaisuuden järjestämisessä.

Seuraavaksi GQM-tiimin tehtävänä on johtaa mittaustavoitteista kysymykset (question, Q) ja kysymysten pohjalta hypoteesit. Kysymyksiä luomalla GQM-tiimi pyrkii helpottamaan mittaustiedon tulkintaa. Kysymysten avulla voidaan täsmentää abstrakteja tavoitteita käytännöllisemmälle tasolle. GQM-tiimin tulee varmistua, että kysymysten avulla voidaan tehdä johtopäätöksiä tavoitteiden saavuttamisesta. Projektitiimin haastattelussa esittämiä kysymyksiä voidaan

hyödyntää lopullisten kysymysten suunnittelussa. Useasti kysymykset ovat liian abstrakteja tai yksityiskohtaisia, jolloin niitä tulee jalostaa, jotta niitä voitaisiin hyödyntää tiedon tulkinnassa (van Solingen & Berghout, 1999), sillä liian abstraktien kysymysten avulla mittareiden luominen voi hankaloitua ja liian yksityiskohtaisen kysymysten avulla tavoitteiden tulkitseminen vaikeutuu (Goldpractices, 2005).

Kuvassa 14 on esitetty esimerkki kysymyksistä, jotka ovat johdettu kuvan 12 tavoitteista.



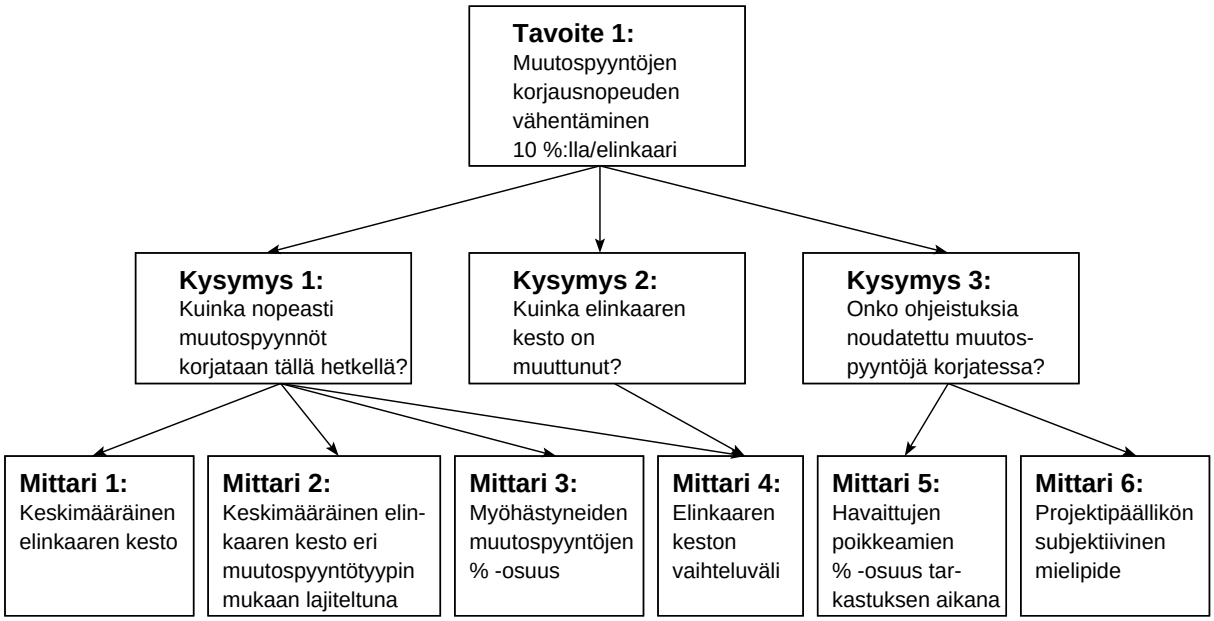
Kuva 14: Esimerkki kysymyksistä (Goldpractices, 2005).

Kysymysten lisäksi tiimien tulee pohtia hypoteesit eli vastaukset tavoitteista johdetuille kysymyksille. Tiedon tulkintavaiheessa mittaustietoa verrataan hypoteeseihin, arvioihin. Tarkoituksena on selvittää, miksi mittaustulokset ja hypoteesit poikkesivat toisistaan tai kenties, miksi ne vastasivat toisiaan. Hypoteeseja muodostamalla projektitiimi liittää uuden tiedon aiempaan tietoon ja tämä saa projektiryhmän paremmin ymmärtämään prosessin, tuotteen tai resurssin luonteen. Lopuksi GQM-tiimin tulisi suorittaa kysymyksien ja hypoteesien katselmointi projektitiimin kanssa.

Kysymysten jälkeen GQM-tiimi johtaa mittarit (metric, M). Mittareiden avulla saadaan kerättyä tieto, jonka avulla voidaan vastata kysymyksiin. Mittareita ei tule kerätä kohtuuttomasti,

sillä mittaustiedon kerääminen ja tulkinta vievät paljon resursseja. Goldpractices (2005) ehdottaakin valitsemaan vain sellaisia mittareita, joita todella tarvitaan. Lisäksi mittareiden tuloksiin voi vaikuttaa erilaiset vaihtelutekijät. Nämä tekijät tulee tunnistaa, muutoin mittaustulokset voivat olla virheellisiä. Yleensä vaihtelutekijät määritellään mittareiksi, jotta niiden vaikutusta voidaan tarkkailla (van Solingen & Berghout, 1999). Lopuksi tulisi tarkistaa mittareiden yhtenäisyys sekä täydellisyys. Tarkoituksena on löytää puuttuvat, epätäydelliset ja epäjohdonmukaiset mittarit.

Kuvassa 15 on esitetty esimerkki mittareista, jotka ovat johdettu kuvan 14 kysymyksistä.



Kuva 15: Esimerkki mittareista (Goldpractices, 2005).

Tämän jälkeen laaditaan Lavazzan ja Barresin (2005) mukaan tärkein suunnitelma: *GQM-suunnitelma* (GQM plan). Se on tärkeä, koska siinä kuvataan tavoitteiden jalostaminen kysymyksiksi ja kysymysten tarkentaminen mittareiksi. Suunnitelma toimii mittausprosessin muodollisena dokumenttina ja siitä tulee käydä ilmi muun muassa kunkin johdetun mittarin suorat mittarit sekä ohjeet mittaustiedon tulkinnalle. Lisäksi sen tulee olla pohjana mittaus-, analysointisuunnitelmalle ja mittauksen tukijärjestelmälle, joka tukee GQM-ryhmää mittaustiedon

käsittelyssä (van Solingen & Berghout, 1999).

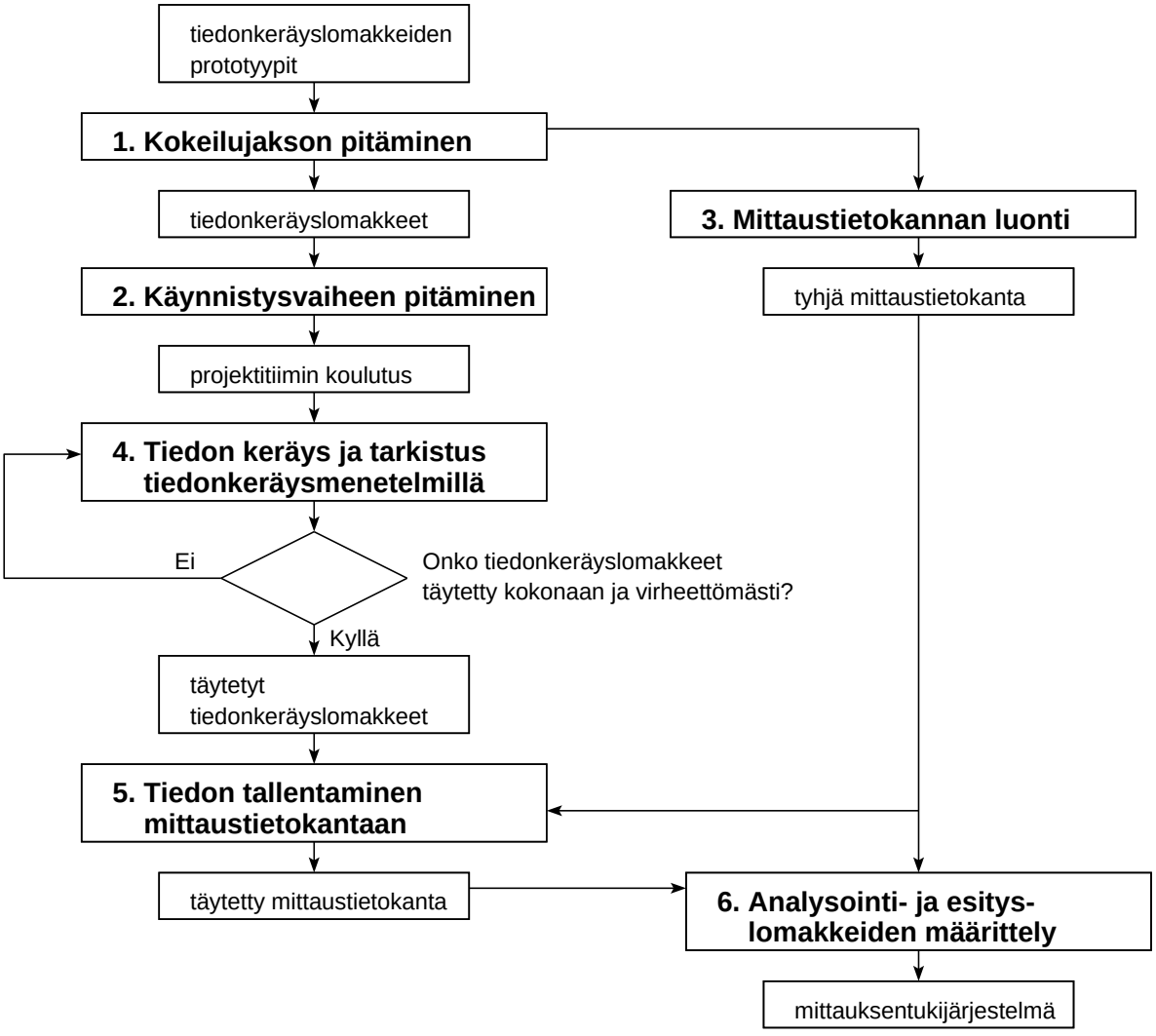
Mittaussuunnitelmassa (measurement plan) tulee mainita kunkin mittarin osalta mittarin nimi, luokittelu ja tekstimuotoinen kuvaus. Lisäksi on mainittava mahdolliset mittarin tuottamat mittaustulokset, yleiskuvaus, mitä mittaustietoa tarvitaan, kuka sitä tarvitsee, mihin tarkoitukseen sekä kerättävän tiedon tarkka kuvaus (kerralla kerättävät tiedot ja jatkuvasti kerättävät tiedot). Mittauksen yhteydessä tulee mainita myös mittauksen suorittava henkilö, mitä koulutusta hän saa ja missä lisäkoulutusmateriaali sijaitsee. Tämän lisäksi tulee mainita aika, jolloin mittaus suoritettiin, tapa, jolla mittaustulos kerätään, kuinka mittaustietojen virheettömyys tarkistetaan, kuinka mittaustulokset julkaistaan, miten niitä hyödynnetään jatkossa, miten niitä säilytetään ja viittauksen GQM-suunnitelmassa olevaan mittariin (Springer, 2001; van Latum & al., 1998; van Solingen & Berghout, 1999).

Analysointisuunnitelman (analysis plan) päätarkoituksena on havainnollistaa, kuinka merkityksellinen mittaustieto esitetään tulkintavaiheessa erilaisten taulukoiden, kaavioiden tai dokumenttien avulla. Tämä helpottaa projektitiimiä tulkinnan muodostamisessa. Lisäksi analysointisuunnitelma antaa projektitiimille kuvan siitä, minkälaisia tuloksia he voivat odottaa saavansa. Suunnitelmassa kuvataan myös hypoteesien merkitys erilaisten kaavioiden sekä taulukoiden avulla. Tämä helpottaa mittaustiedon ja hypoteesien välistä tulkintaa (van Latum & al., 1998). Tämän lisäksi suunnitelmassa tulee kuvata vaihtelutekijät, ja kuinka niitä käytetään mittaustuloksissa. Vaihtelutekijöiden vaikutuksia voidaan arvioida mittaustuloksien tulkinnassa.

Lopuksi GQM-tiimi pitää GQM-, mittaus- ja analysointisuunnitelmille tarkastustilaisuuden, jossa keskitytään seuraaviin asioihin: hyväksyykö projektitiimi tavoitteet, kysymykset ja mittarit; löytävätkö he puuttuvia tai turhia määrityksiä ja hyväksyvätkö he ehdotetun palautemateriaalin (van Solingen & Berghout, 1999). Kun määrittelyvaiheen tehtävät on suoritettu, siirrytään tiedonkeruuvaiheeseen.

4.2.3 Tiedonkeruuvaihe

Tiedonkeruuvaiheen (data collection phase) päätarkoituksena on kerätä yhdenmukaisesti mittaustietoa ja tallentaa se mittauksentukijärjestelmään. Jotta mittaustiedot saadaan kerättyä, tulee keräystä varten laatia tiedonkeruulomakkeet sekä -välineet. Keruumenetelmien toimivuus tulee testata ennen varsinaisen tiedonkeruun aloittamista. Kuvassa 16 on esitelty tiedonkeruuvaiheen eteneminen.



Kuva 16: Tiedonkeruuvaiheen eteneminen (van Solingen & Berghout, 1999).

Ennen tiedonkeruun aloittamista tulee luoda lomakkeet, joilla mittaustieto kerätään. Mittaustiedon voi kerätä usealla eri tavalla (van Solingen & Berghout, 1999): manuaalisilla, elektronisilla tai automaattisilla tiedonkeruumenetelmillä.

Manuaaliset tiedonkeruumenetelmät ovat helppoja ja joustavia, joita usein käytetään GQM-mittaustiedon keruussa. Lomakkeet etenevät GQM- ja mittaussuunnitelmien mukaisesti ja lomakkeesta tulee käydä ilmi mittauksen alullepanijan yhteystiedot; kaikki eri vaihtoehdot, jotka mittauksen aikana voidaan saavuttaa; kohta, jossa selitetään eri vaihtoehtojen merkitykset ja kohta, johon täyttäjä voi kirjoittaa lisähuomautuksia. Projektitiimin jäsenet luovuttavat täytetyt lomakkeet GQM-tiimille, joka kirjaa tiedot tietokantaan. Mikäli lomakkeisiin halutaan tehdä muutoksia, tulee muutokset hyväksyttää projektitiimillä. Muutokset pitää myös korjata GQM- ja mittaussuunnitelmiin. Tällöin jäljitettävyydestä huolehtiminen on ensiarvoisen tärkeää (Baldassarre & al., 2003; Park & al., 1996).

Elektronisten tiedonkeruumenetelmien avulla mittaustiedon keruu helpottuu, sillä tieto kerätään elektronisessa muodossa ja sen siirtäminen tietokantaan nopeutuu. Lisäksi tietojen ylläpito on keskitetty ja kaikki projektitiimin jäsenet pääsevät tietoihin käsiksi. Esimerkkejä elektronisista tiedonkeruumenetelmistä ovat muun muassa sähköposti, tietokantasovellukset ja Internet.

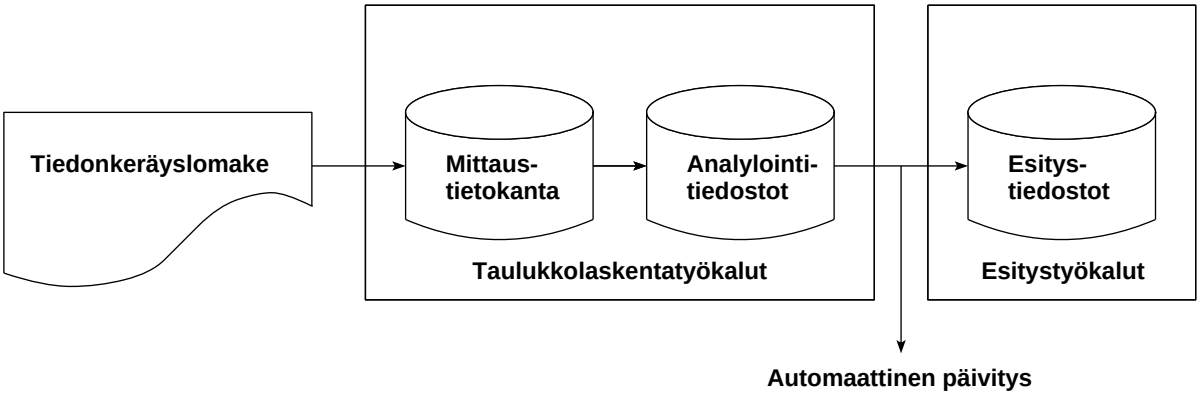
Automaattiset tiedonkeruumenetelmät keräävät mittaustiedot automaattisesti ennalta määrättyjen algoritmien mukaan. Automaattinen tiedonkeruumenetelmä on hyödyllinen, koska se ei vaadi aikaa ja virheellisen tiedon määrä pienenee, sillä ihminen ei suorita tiedonkeruuta (Lavazza, 2000). Vaikka automaattiset välineet voivat olla tehokkaita, tulee tiedon kerääjien olla varovaisia jotteivät välineet rajoittaisi tiedonkeruuta, sillä kaikkea tietoa ei voi kerätä automaattisesti. Van Solingen ja Berghout (1999) korostavatkin, että tärkein tieto tulee ihmisiltä, ei välineiltä.

Tiedonkeruuvaiheen ensimmäisenä tarkoituksena, on testata lomakkeiden, työkalujen ja menettelyjen pätevyys. GQM-tiimi pitää muutamalle projektitiimin jäsenelle parin päivän mittaisen kokeilujakson, jossa testaus tapahtuu. Tämän jälkeen pidetään käynnistysvaihe (kick-off ses-

sion), jossa mittausprosessi esitellään projektitiimille. Käynnistysvaiheen aikana GQM-tiimi pyrkii saamaan projektitiimin hyväksynnän tiedonkeruun aloittamiseksi.

Käynnistysvaiheen jälkeen GQM-tiimi luo tietokannan, johon mitatut tiedot tallennetaan. Tietokanta on lähtökohta mittauksentukijärjestelmän (measurement support system, MSS) luomiselle. Lopuksi luodaan mittauksentukijärjestelmä, jonka tarkoituksena on tukea mittausprosessia. GQM-tiimi luo mittauksentukijärjestelmän yleensä rinnakkain mittausprosessin kanssa.

Mittauksentukijärjestelmä tukee kaikkia mittauksen toimintoja, kuten tiedonkeruuta, varastointia, ylläpitoa, tulkintaa, esittämistä ja paketointia. Se koostuu kolmesta perusosasta: tietokannasta, analysointiosasta ja tiedon esitysosasta. Tietokantaan tallennetaan kerätty tieto, joka voidaan halutessa ottaa analysoitavaksi. Analysointiosassa tietoa lajitellaan, valitaan ja lasketaan. Tämän jälkeen tiedosta muodostetaan kuvaajia ja taulukoita. Lopuksi tieto voidaan siirtää tiedon esitysosaan, joka tukee automaattista esityskalvojen luomista. Kuvassa 17 on esitetty mittauksentukijärjestelmän perusosat.



Kuva 17: Mittauksentukijärjestelmän perusosat (van Solingen & Berghout, 1999).

Jokaista mittaustavoitetta kohdin pitäisi tehdä oma tiedosto, joka sisältää seuraavat taulukot: tavoitetaulukko, jossa kuvataan tavoite ja siihen liittyvät kysymykset GQM-suunnnitelman mukaisesti; tietotaulukko, joka sisältää tavoitteeseen liittyvien kysymysten vastaamiseen tarvittavat tiedot sekä hypoteesit tietokannasta; useita kysymystaulukoita, joissa kussakin on yksi ta-

voitteeseen liittyvä kysymys. Kysymystaulukossa tietotaulukon tieto käsitellään ja muutetaan kuvaajiksi sekä taulukoiksi. Kun tiedot ovat saatu kerättyä, siirrytään tiedon tulkintavaiheeseen.

4.2.4 Tulkintavaihe

Tulkintavaiheen (interpretation phase) päätarkoituksena on yrittää löytää palautetilaisuuksissa vastaukset aiemmin asetettuihin kysymyksiin ja niiden avulla selvittää, onko mittaustavoitteisiin päästy. Tulkintavaiheen tulosten perusteella voidaan tehdä jatkotoimenpiteitä muun muassa keräämällä lisää mittaustietoa ja muuttamalla prosessien toimintaa siten, että mittaustulokset olisivat jatkossa halutunlaisia. Kuvassa 18 on esitelty tulkintavaiheen eteneminen.

Kuva 18: Tulkintavaiheen eteneminen (van Solingen & Berghout, 1999).

Tulkintavaiheen ensimmäisenä tarkoituksena on palautetilaisuuksien (feedback session) valmistelu, jotka GQM-tiimi valmistelee mittaustiedon pohjalta. Valmistelutilaisuudessa luodaan esitysmateriaali, josta pitäisi, GQM-suunnitelmaa apuna käyttäen, pystyä tulkitsemaan, kuinka hyvin asetetut mittaustavoitteet ovat saavutettu.

Tämän jälkeen GQM-tiimi pitää palautetilaisuuden projektitiimille, mikäli projektitiimillä ei ole aikaisempaa kokemusta palautetilaisuuden pitämisestä. Palautetilaisuuksia tulisi järjestää

van Latum & al. (1998) mukaan 6-8 viikon välein, jotta uutta ja mielenkiintoista mittaustietoa saataisiin kerättyä riittävästi. Lisäksi aikaväli on riittävän lyhyt, jotta projektitiimin jäsenten mielenkiinto säilyy asiassa. Palautetilaisuudessa GQM-tiimin rooli on esittää vaihtoehtoisia tulkintoja mittaustuloksista. GQM-tiimin jäsenten ei tulisi tehdä lopullisia tulkintoja, mikäli he eivät ole alan asiantuntijoita. Projektitiimi puolestaan pyrkii muodostamaan johtopäätöksiä ja toimenpiteitä uusista mittaustuloksista kaavioiden ja taulukoiden avulla. Samalla projektitiimin tulisi arvioida uudelleen edellisten palautetilaisuuksien toimenpiteiden hyödyllisyyttä. (van Solingen & Berghout, 1999; van Solingen & Berghout, 2001).

Esimerkki van Latumin & al. (1998) tutkimasta projektista, jossa mittaustieto tulkittiin ja muutettiin johtopäätösten pohjalta toimenpiteiksi:

Projektitiimin jäsenet keräsivät mittaustietoa seitsemän kuukauden ajan siitä, kuinka paljon ohjelmistossa esiintyi erityyppisiä (pieniä, suuria, tuhoisia) puutteita. He tekivät hypoteesin puutteiden laadusta ja esiintymistiheydestä. He arvioivat, että pieniä puutteita esiintyy 60 %, suuria 30 % ja tuhoisia 10 %. Mittaustieto osoitti palautetilaisuudessa, että pieniä puutteita esiintyi seitsemän kuukauden aikana 39 %, suuria 36 % ja tuhoisia 26 %. Projektitiimi tulkitsi saatuja mittaustuloksia ja etsi mahdollisia syitä, miksi odotetut tulokset eivät vastanneet saatuja tuloksia. Tämän pohjalta he tekivät johtopäätöksiä ja suunnittelivat toimenpiteitä asian parantamiseksi. Projektitiimi päätti muun muassa pyrkiä laskemaan puutetiheyden 1,9 %:iin tuhatta ohjelmariviä kohden. Ensimmäiseksi he keräsivät kahdeksan moduulia, jotka sisälsivät eniten puutteita. Sen jälkeen he tekivät moduuleille katselmoinnin ja ohjelmoivat yhden moduulin kokonaan uudestaan. Kaikkien toimenpiteiden jälkeen he jatkoivat mittaamisen suorittamista.

Esimerkissä projektitiimi huomasi, kuinka paljon heidän moduuleissaan oli puutteita. He päättivät asettaa tavoitteet itselleen ja alkoivat tekemään parannustoimenpiteitä asian eteen. Van Latumin & al. (1998) mukaan palautetilaisuuksien pitäminen on erittäin oleellinen osa GQM-menetelmän onnistumisen kannalta, sillä juuri se antaa projektitiimille mahdollisuuden huomata kriittiset ongelmat ja keskittyä niiden poistamiseen. Tällöin todellinen prosessin parantami-

nen voi alkaa.

Van Latum & al. (1998) huomasivat, että palautetilaisuus ei antanut vain parempaa ymmärrystä, vaan useasti he määrittelivät tilaisuudessa lisää kysymyksiä ja joskus jopa uusia tavoitteita.

Jotta palautetilaisuus olisi mahdollisimman tehokas, tulisi van Solingenin ja Berghoutin (2001) mukaan panostaa eri oppimistekijöihin, sillä oppiminen johtaa asioiden ymmärtämiseen ja sitä kautta prosessin parantamiseen. Van Solingenin ja Berghoutin (2001) mukaan projektitiimin jäsenten hyvä motivaatio ja avoin ilmapiiri ovat tärkeimpiä tekijöitä oppimisen onnistumisen kannalta. Lisäksi asiayhteyden ymmärtäminen sekä ohjelmiston nykyisen tilanteen tuntemus auttavat oppimisessa ja lisäävät tietoa, jolloin sopivimpien parannustoimenpiteiden valitseminen on helpompaa. Tiimioppiminen on myös hyvä tapa oppimisen kannalta, koska siinä pyritään, yhdessä työskentelyn lisäksi, ymmärtämään erilaisten prosessien käyttäytymistä. Kun eri käyttäytyminen tunnetaan, voidaan parannustoimenpiteet valita sen mukaan.

Palautetilaisuuden jälkeen GQM-tiimin tulee kirjoittaa tilaisuudesta raportti, josta tulee käydä ilmi kaikki tilaisuudessa esille tulleet havainnot, tulkinnat, johtopäätökset ja toimenpiteet. Raportti jaetaan kaikille projektiryhmän jäsenille ja tärkeimmät tulokset levitetään koko organisaation tietoisuuteen esimerkiksi sisäiseen verkkoon. Mittaustulosten yksityiskohdat on hyvä säilyttää vain GQM- ja projektitiimin tiedossa, sillä ylempi johto ei välttämättä ymmärrä tulosten syvällistä merkitystä (van Solingen & Berghout, 1999; van Solingen & Berghout, 2001).

Kun mittausprosessi on saatu päätökseen, tulee GQM-tiimin tehdä loppuraportti, josta käy ilmi, kuinka tavoitteissa onnistuttiin, ja mitkä olivat prosessin kustannukset hyötyyn nähden. Objektiivisen kustannus-hyöty -analyysin (cost-benefit analysis) tekeminen on vaikeaa, sillä tarkkoja kustannusarvoja ei pystytä laskemaan sen vaikutustekijöiden takia. Mittausprosessista voidaan kuitenkin arvioida, oliko sen toteuttaminen kannattavaa. Kustannus-hyöty -analyysin tekeminen on tärkeää myös siksi, etteivät mittausprosessin ulkopuoliset henkilöt alkaisi spekuloida mittausprosessin kustannuksilla (van Solingen & Berghout, 1999).

Liitteessä 1 käsiteltiin alustavaa kustannusmallia, jonka tiedot tulisi sisällyttää lopulliseen kus-

Taulukko 4: Mittausprosessin kustannukset ja hyödyt (van Solingen & Berghout, 1999).

Kustannukset	Hyödyt
- Mittausprosessiin kulunut aika GQM-tiimin osalta - GQM- ja projektitiimien tapaamisiin kulunut aika - Tiedonkeräyslomakkeiden täyttöön kulunut aika - Mittauksentukijärjestelmän kehittämiseen kulunut aika - Mittausprosessia tukevien laitteiden ja ohjelmistojen ostaminen - Tiedontulkintaan ja palautetilaisuuksien valmisteluun kulunut aika	- Laadun paranemisesta johtuva myynnin kasvu - Parantuneiden kehitysprosessien tuntemuksesta johtuva ohjelmistojen kehitykseen käytettävä ajan säästäminen - Parantuneesta resurssienhallinnasta johtuva kustannussäästöt ja uusien kustannusten välttäminen

tannus-hyöty-analyysiin (cost-benefit-analysis). Lopullisessa kustannus-hyöty-analyysissa tulisi käsitellä taulukon 4 sisältämiä kustannuksia ja hyötyjä.

4.3 GQM-menetelmää täydentävät ratkaisut

Vaikka GQM-menetelmä on laajalti hyväksytty, on sitä kohdin esitetty monenlaista kritiikkiä (Lavazza & Barresi, 2005). GQM-menetelmää on arvosteltu muun muassa siitä, että menetelmä ei ole toistettavissa oleva, jolloin samasta tavoitteesta voidaan päätyä erilaisiin mittareihin, joka johtaa siihen, että tieto ei ole uudelleenkäytettävää. Lisäksi on hankalaa tietää, milloin kysymysten esittäminen tulee lopettaa, ja missä vaiheessa kysymyksistä tulee muodostaa mittarit. Myöskään kaikkiin kysymyksiin ei voida vastata, koska sopivia mittareita ei ole saatavilla (Fuggetta & al., 1998; Lavazza & Barresi, 2005). Lavazzan ja Barresin (2005) mielestä nämä ongelmat johtuvat epätäydellisten ohjelmistoprosessimallien puutteista, sillä GQM-suunnitelmat muodostetaan näiden mallien pohjalta. Mikäli mallit ovat puutteellisia, ovat GQM-suunnitelmat useasti epäselkeitä tai ne eivät ole tarpeeksi yksityiskohtaisia, jotta niitä voitaisiin hyödyntää

jatkossa. Siksi mallien kehitykseen tulisi kiinnittää enemmän huomiota.

Kritiikin takia GQM-menetelmälle on esitetty koko sen kehityshistorian ajan vaihtoehtoisia ratkaisuja, jotka pyrkivät täydentämään sitä. Viimeaikoina sille onkin ehdotettu muun muassa seuraavanlaisia ratkaisuja: AF (Mendonça & Basili, 2000), V-GQM (Olson & Runeson, 2001) sekä GQ(I)M (Goethert & Siviy, 2004; Park & al., 1996).

Mendonça ja Basili (2000) ovat kiinnittäneet huomiota siihen, kuinka meneillään olevaa mittaamista voitaisiin ymmärtää sekä mittaustuloksia hyödyntää jatkossa paremmin, sillä organisaatioiden mittausprosessit eivät aina ole loppuun asti harkittuja. Useasti organisaatiot keräävät hyödytöntä mittaustietoa, tai varteenotettavaa mittaustietoa ei tulla koskaan käyttämään hyväksi. Mendonça ja Basili (2000) ovatkin kehittäneet menetelmän mittaamisen tehokkuuden parantamiseksi. Menetelmässä sovelletaan GQM- sekä attribuutinkohdennus (attribute focusing, AF) -menetelmää. GQM-menetelmän avulla pyritään selvittämään organisaatiolle hyödylliset mittarit ja attribuutinkohdennus-menetelmä perustuu tiedonrikastus- (data minding) tekniikkaan, jonka avulla pyritään etsimään mittaustiedosta mielenkiintoisia tosiasioita, joihin ei ole vielä kiinnitetty huomiota. Mielenkiintoiset tosiasiat esitetään esimerkiksi erilaisten diagrammien avulla, jotta asiantuntijoiden on helppo hahmottaa niissä esiintyvät poikkeamat uuden tiedon löytämiseksi. Mendonçan ja Basilin (2000) mukaan GQM- ja AF-menetelmä täydensivät toisiaan asiakastyytyväisyyden mittaamisessa, mutta lisätutkimusta menetelmien laajemmasta yhteensopivuudesta tarvitaan.

Olsson ja Runeson (2001) ovat esittäneet GQM-menetelmästä laajennoksen V-GQM:n (Validating Goal/Quesstion/Metric), jonka tarkoituksena on analysoida GQM-menetelmän hyvät ja huonot puolet, jotta GQM-menetelmän tarkoituksia, kysymyksiä ja mittareita voitaisiin kehittää jatkossa paremmiksi. Ensimmäiseksi menetelmän avulla tarkistetaan, käytettiinkö kaikkia mittareita, saatiinko niiden avulla enemmän tietoa kuin oli tarkoitus tai pystyttiinkö niiden avulla vastaamaan useampiin kysymyksiin kuin oli suunniteltu. Seuraavaksi analysoidaan kysymykset, joiden avulla mittarit johdettiin. Tarkoituksena on parantaa kysymyksien laatua poistamalla

epäoleelliset kysymykset ja lisäämällä uusia kysymyksiä, jotta niiden avulla olisi helpompi selvittää tavoitteiden toteutuminen. Lopuksi tavoitteita mukautetaan siten, että ne vastaavat kysymyksiä ja mittareita. Olssonin ja Runesonin (2001) mielestä V-GQM kaipaa tarkennusta, sillä esitetty malli ei vielä ole täysin objektiivinen.

SEI on soveltanut GQM-menetelmää ja kehittänyt siitä GQ(I)M (Goal/Question/Indicator/Metric)-menetelmän, johon on lisätty indikaattorit. Indikaattoreiden avulla voidaan parantaa mittausprosessia, sillä indikaattoreiden tehtävänä on muodostaa mittaustuloksista kuvaajia, joiden avulla voidaan etukäteen selventää, kuinka mittaustieto tulee todellisuudessa käyttäytymään. Täten on helpompi tietää, mitä todellisuudessa kannattaa mitata ja valita mittariehdokkaat sen mukaisesti (Park & al., 1996). SEI on käyttänyt indikaattoreita onnistuneesti prosesseissaan ja huomannut, että ilman indikaattoreiden visualisointia hyvän mittausprosessin aloittaminen voi muodostua hankalaksi (Goethert & Siviy, 2004).

GQM-menetelmän ongelmista ja muunnelmista huolimatta GQM-menetelmää on oikein käytettynä erinomainen menetelmä ohjelmistoprosessin parantamiseen ja sitä onkin käytetty onnistuneesti useissa teollisuusprosesseissa (Baldassarre &al., 2003; van Latum, 1998). Seuraavassa luvussa perehdytään tarkemmin, kuinka GQM-menetelmän avulla voidaan keskittyä ylläpidon ja sitä kautta asiakastyytyväisyysprosessin parantamiseen.

5 YLLÄPIDON TAVOITEKESKEINEN MITTAAMINEN

Suurin osa ohjelmistoprosessin ongelmista kohdataan ylläpitovaiheessa (Rombach & Ulery, 1989). Siksi ylläpidon mittaaminen on ensiarvoisen tärkeää, jotta ylläpitoa voitaisiin seurata ja kehittää haluttuun suuntaan.

Tässä luvussa keskitytään ohjelmiston ylläpitoon ja sen mittaamiseen. Ensiksi kerrotaan yleisesti, mitä ohjelmiston ylläpidolla tarkoitetaan, jonka jälkeen perehdytään tavoitepohjaiseen ylläpidon mittaamiseen. Tarkoituksena on kertoa esimerkin avulla, kuinka tavoitepohjaista GQM-menetelmää käyttämällä voidaan johtaa liiketoiminnan tavoitteista ylläpidossa tarvittavia mittareita.

5.1 Ohjelmiston ylläpito

Ohjelmiston ylläpito (software maintenance) on jatkuva vaihe, joka alkaa, kun tuote on toimitettu asiakkaalle. Ylläpidon tarkoituksena on korjata tuotteesta löydettyjä virheitä, parantaa tuotteen ominaisuuksia tai sovittaa tuote muuttuneeseen ympäristöön (IEEE, 1998).

Ohjelmiston ylläpito jaetaan yleensä neljään eri kategoriaan: korjaavaan (corrective), mukautuvaan (adaptative), ennaltaehkäisevään (preventive) ja parantavaan (perfektive). Korjaavan ylläpidon tarkoituksena on korjata tuotteessa havaitut virheet. Mukautuvan ylläpidon avulla pyritään säilyttämään tuotteen toiminnallisuus, kun tuote muuttuu esimerkiksi päivityksen yhteydessä tai laitteiston muuttuessa. Ennaltaehkäisevällä ylläpidolla tarkoitetaan sitä, että ylläpitäjät havaitsevat ja korjaavat virheet ennen kuin ne aiheuttavat häiriötä tuotteen käyttäjälle. Lopuksi on vielä parantava ylläpito, jonka tarkoituksena on korjata ohjelmistosta epäjohdonmukaiset kohdat, jotta käyttäjän olisi helpompi käyttää ohjelmistoa (Fenton & Pfleeger, 1997; IEEE, 1998; Polo, 2003; Pressman, 2000).

Korjaava ylläpito on yleisin ylläpidon muoto. Se alkaa, kun käyttäjä havaitsee tuotteessa ongelman. Muutospyyntö kirjataan ylös, mikäli vastaavanlaista ongelmaa ei ole aikaisemmin havaittu. Lisäksi selvitetään, johtuuko ongelma käyttäjän väärinkäsityksestä ja voidaanko se ratkaista neuvomalla käyttäjää. Mikäli ongelmaa ei voida ratkaista, tulee muutospyyntö luokitella sen aiheuttaman ongelman ja prioriteetin mukaan.

Tämän jälkeen muutos toteutetaan samoja elinkaaren vaiheita noudattaen kuin normaali ohjelmistoprosessin kehittäminen vaatii. Ongelman tunnistamisen jälkeen analysoidaan, miten tuleva muutos vaikuttaa muihin komponentteihin. Sen jälkeen suunnitellaan muutos lähdekoodia, tietokantaa, ohjelmiston ja projektin dokumentaatiota apuna käyttäen. Tämän jälkeen toteutetaan muutos, jonka jälkeen suoritetaan järjestelmätestaus. Sen tarkoituksena on varmistaa, että ohjelmisto kattaa muutoksen lisäksi kaikki aiemmin asetetut vaatimukset.

Lopuksi suoritetaan hyväksymistestaus, jossa tarkistetaan, että ohjelmisto toteuttaa kaikki sille asetetut vaatimukset. Kun muutos on toteutettu ja sen toimivuus on testattu, toimitetaan muutos käyttäjälle. Toimittajan tulee huolehtia muutoksen asentamisesta sekä käyttäjien koulutuksesta (IEEE 1998; Pigoski, 1997). Tarkempi kuvaus ylläpidon elinkaarimallista on esitelty liitteessä 2.

Ylläpidon mittaamiseen on olemassa monenlaisia standardeja ja malleja, jotka jaottelevat mittaamisen eri ominaisuuksien perusteella. ISO/IEC 9126-1 tarjoaa seuraavanlaisen jaottelun (ISO/IEC, 2001): analysoitavuusmittarit (analysability metrics), joiden avulla pyritään muun muassa selvittämään ylläpitäjän tai käyttäjän kuluttamia resursseja, jotka aiheutuvat vian syyn määrittämisestä; muutettavuusmittarit (changeability metrics), joiden avulla pyritään muun muassa selvittämään ylläpitäjän tai käyttäjän toimintatapoja vian määrittämiseksi, vakausmittarit (stability metrics), joiden avulla pyritään muun muassa selvittämään, kuinka vakaa ohjelmisto on muutoksen jälkeen; testattavuusmittarit (testability metrics), joiden avulla pyritään muun muassa selvittämään ylläpitäjän tai käyttäjän toimintatapoja testauksen aikana ja ylläpidon noudattamismittarit (maintenance compliance metrics), joiden avulla pyritään muun muassa selvit-

tämään, kuinka hyvin standardeja, sääntöjä ja käytäntöjä noudatetaan ylläpidon aikana.

ISO/IEC 9126-1:n mukaista ylläpitomittareiden luokittelua voidaan pitää ohjeellisena, sillä ylläpitoa voidaan myös mitata ohjelmistoprosessimittareiden avulla (Pressman, 2000). Täten ylläpitomittareiden tarkka luokittelu on vaikeaa. Toisaalta ylläpidon kehittämisen kannalta ei ole oleellista, mihin luokkaan jokin mittari kuuluu vaan se, että mittarin avulla saadaan kerättyä tarpeellista tietoa. Täten jatkossa ei enää kiinnitetä huomiota tähän luokitteluun.

Seuraavaksi keskitytään siihen, kuinka tavoitteista voidaan jalostaa mittareita ja miten ylläpitoa voidaan mitata asiakastyytyväisyyden kehittämisen näkökulmasta.

5.2 Ylläpitomittareiden muodostaminen GQM-menetelmän avulla

Tässä kohdasa sovelletaan tavoitepohjaista GQM-menetelmää ylläpidon mittaamiseen. Tarkoituksena on havainnollistaa esimerkin avulla, kuinka liiketoiminnan tavoitteesta voidaan johtaa mittareita. Liiketoiminnan parantamistavoitteeksi on valittu asiakastyytyväisyys ja sen näkökulmaksi ylläpidon kehittäminen, jonka kohteena on asiakkaalta tulevat muutospyynnöt. Tarkoituksena on esittää mittareita, joiden avulla seurataan kuvan 12 mittaustavoitteen toteutumista. Tavoitteena on vähentää muutospyyntöprosessin elinkaaren pituutta 10 %:lla.

Jotta GQM-menetelmää voitaisiin käyttää tehokkaasti, tulee kiinnittää huomiota siihen, että mittaustavoitteet ovat tarkoin valittuja. Mittaustulokset ovat useasti merkityksettömiä ilman varteenotettavia mittaustavoitteita. Siksi tässä kohdin keskitytään erityisesti siihen, kuinka mittaustavoitteet tulisi johtaa liiketoiminnan tavoitteista.

5.2.1 Liiketoiminnan tavoitteet

Mittareiden jalostaminen alkaa liiketoiminnan tavoitteiden tunnistamisella. Nämä tavoitteet tulisi kerätä henkilöiltä, joiden tuki on kriittisin mittausprosessin onnistumisen kannalta. Henkilöiden ei välttämättä tarvitse kuulua ylimpään johtoon, sillä mittausprosessin ei tarvitse olla

koko organisaation laajuinen. Esimerkiksi projektipäälliköt voivat toimia hyvinä tietolähteinä. Liiketoiminnan tavoitteet voidaan saavuttaa esimerkiksi haastattelujen tai aivoriihen avulla (Park & al., 1996).

Haastattelu on tehokas keino, jonka avulla voidaan selvittää haastateltavan näkemys esimerkiksi liiketoiminnan tavoitteista. Haastattelun suorittaminen voi olla vaikeaa, sillä ihmisten on hankalaa hahmottaa toisten ihmisten näkemyksiä, mikäli heidän näkemyksensä poikkeavat toisistaan. Tämän takia haastateltavan olisi hyvä keskittyä kontekstittomien kysymyksien esittämiseen, jolloin haastattelija ei johdattele haastateltavaa omien näkemystensä mukaisesti.

Hyvä haastattelija valmistautuu haastatteluun selvittämällä yrityksen taustan ja haastattelun edetessä hän tarkistaa, että hän on ymmärtänyt haastateltavan ajatukset oikein. Tärkeimmät asiat alkavat yleensä hahmottua muutaman haastattelun jälkeen. Tämän jälkeen haastattelija voi keskittyä yleisempien asioiden kartoittamiseen (Leffingwell & Widrig, 2003).

Haastattelun sijasta voidaan myös tehdä kysely, joka on useasti halvempi vaihtoehto haastatteluun verrattuna. Kysely voi olla hyödyllinen, jos tavoite tukee tilastollista analyysiä, joka koostuu rajoitetuista vaihtoehdoista. Leffingwelin & Widrigin (2003) mukaan haastattelua ei pitäisi koskaan korvata kyselyllä, koska kyselyn avulla ei välttämättä synny yhteisymmärrystä asiasta ilman vapaata vuorovaikutusta. Leffingwell & Widrig (2003) painottavat, että jo muutaman haastattelun jälkeen haastattelijan näkemys muuttuu, jonka ansiosta voidaan luoda visio ongelman ratkaisemiseksi.

Aikaisemmin luvussa 4 käsiteltiin abstraktiolomaketta, josta todettiin, että sitä voidaan käyttää tietämyksen muodostustekniikkana haastattelun yhteydessä. Abstraktiolomakkeen avulla haastattelun rakenne voidaan jakaa pienempiin osiin. Tämä mahdollistaa sen, että haastateltava ei siirry äkillisesti ongelmasta toiseen, vaan tietämys ongelmien luonteesta saadaan paremmin selville.

Kuvassa 19 on abstraktiolomake, jossa on esitetty muutospyyntöprosessin korjausnopeuteen vaikuttavia asioita, jotka voivat ilmetä haastattelun aikana. Ensimmäisessä osassa haastattelija

pyrkii saamaan selville, mihin asioihin tulisi kiinnittää huomiota, jotta muutospyyntöprosessia saataisiin kehitettyä nopeammaksi. Toisessa osassa tulisi saada selville haastateltavan arviot muutospyyntöprosessin korjaukseen kuluvasta ajasta. Sen jälkeen haastateltavan pitäisi pystyä tunnistamaan vaihtelutekijöitä, jotka vaikuttavat muutospyyntöjen korjausnopeuteen. Esimerkiksi muutosten laajuus voi kasvattaa muutospyynnön korjaukseen kuluvaa aikaa. Tämän jälkeen neljännessä osassa haastateltavan tulisi pystyä osoittamaan väittämänsä todeksi käytännön kokemuksen perusteella, kuinka muutosten laajuus on hidastanut muutospyyntöjen korjaamista.

Kohde	**Tarkoitus**	**Ongelma**	**Näkökulma**	**Ympäristö**
Muutos-pyynnöt	Parantaminen	Aikarajat	Projekti-päällikkö	Projekti AB

Osa 1: Laatuun keskittyminen	**Osa 3: Vaihtelutekijät**
- Muutospyynnön korjaukseen kulunut aika - Aikaerot muutospyyntöjen korjauksessa - Muutospyynnön kriittisyys	- Muutosten laajuus - Muutospyyntöjen hallinnassa käytettävät apuvälineet - Muustospyyntojen korjauksessa käytettävät apuvälineet
Osa 2: Perushypoteesit	**Osa 4: Vaihtelutekijöiden vaikutus hypoteeseihin**
- Kuinka monta muutospyyntöä korjataan kahden viikon aikana? - Kuinka iso muutospyyntöjen korjausvariaatio on? (esimerkiksi 2 päivää - 3 viikkoa)	- Huonot muutospyyntöjen hallinnassa käytettävät apuvälineet eivät esimerkiksi tue kriittisyyden priorisointia. (vaikuttavat korjausaikaan) - Huonot apuvälineet tekevät jäljitettävyydestä ja valvonnasta vaikeaa. (vaikuttavat korjausaikaan)

Kuva 19: Muutospyyntöprosessin nopeuteen vaikuttavien tekijöiden selvittäminen abstraktiolomakkeen avulla (Goldpractices, 2005).

Haastattelun ja abstraktiolomakkeen lisäksi liiketoiminnan tavoitteita voidaan saada selville aivoriihen avulla. Sen tarkoituksena on generoida uusia ideoita tai toimivia ratkaisuja. Se on

käytännöllinen ja helppo keino, joka koostuu ideoiden tuottamisesta ja niiden karsimisesta. Aluksi kukin ryhmän jäsen keksii niin monta ideaa kuin mahdollista. Aivoriihen tarkoituksena on toimia luovana prosessina, jolloin kukin ryhmän jäsen voi unohtaa normaalit rajoitteet ja käyttää mielikuvitustaan ideoiden generoimiseen.

Yksi aivoriihen perussäännöistä on, että kenenkään ideoita ei arvostella, jotta kukin jäsen uskaltaa tuoda omat ideansa esille. Kun ideat on saatu kerättyä, alkaa ideoiden karsiminen. Karsimisvaiheen tarkoituksena on poistaa epäolennaiset ideat ja jatkaa varteenotettavien ideoiden kehittämistä. Karsimisvaiheen aikana olisi hyvä, että idean keksijä kertoisi tarkemmin ideastaan, jotta muut ryhmän jäsenet saisivat siitä tarkemman kuvan. Ideoista keskusteleminen ja niiden muovaaminen on erittäin tärkeää, sillä Leffingwelin & Widrigin (2003) mukaan kaikki vallankumoukselliset ideat ovat syntyneet ryhmän jäsenten yhteistyönä.

Lopuksi haastattelujen ja aivoriihen avulla löydetyt liiketoiminnan tavoitteet tulisi priorisoida tärkeysjärjestykseen, jotta mittausprosessista olisi enemmän hyötyä liiketoiminnalle (Briand & al., 2002).

5.2.2 Mittaamisen tavoitteet

Liiketoiminnan tavoitteiden tunnistamisen jälkeen tulisi selvittää, mitä mittausprosessin avulla halutaan saavuttaa. Apuna voidaan käyttää olio-kysymyslistaa (entity-question list), jonka avulla pyritään hahmottamaan alustavia kysymyksiä. Kysymykset auttavat mittaamisen alitavoitteiden tunnistamista. Olio-kysymyslistaa hyödynnettäessä ensimmäisenä tehtävänä on valita liiketoiminnan tavoite, jonka prioriteetti on korkein. Tämän jälkeen tavoitteelle tulisi tunnistaa henkilöt mittausprosessin näkökulman ja roolien selvittämiseksi (Park & al., 1996). Henkilöiden tunnistaminen on tärkeää, sillä jokaisella henkilöryhmällä on erilainen lähestymistapa asiaan. Esimerkiksi yrityksen johtajat ovat kiinnostuneita siitä, että päätökset ovat tehty yrityksen toimintaperiaatteiden mukaan ja niiden avulla voidaan saavuttaa voittoja. Projektipäälliköt puolestaan ovat kiinnostuneita siitä, kuinka ongelmat vaikuttavat heidän projekteihinsa (The

Ami Consortium,1995).

Kolmanneksi tulisi luoda karkea luonnos prosessista, jota suunniteltaessa tulisi pohtia, mitä mittausprosessin avulla halutaan saavuttaa ja mitä ongelmia sen aikana kohdataan. Sen jälkeen tulisi selvittää oliot, joihin henkilöt vaikuttavat, kuten syötteet, resurssit, sisäiset artefaktit, toiminnot, tuotteet ja sivutuotteet. Jokaiselle oliolle tulisi listata kysymyksiä, sillä ne auttavat tavoitteiden suunnittelussa. Lopuksi tulisi katsoa prosessia kokonaisuutena, jotta siitä ei olisi unohdettu mitään. Tämän jälkeen kukin vaihe tulisi käydä läpi uudelleen muiden liiketoiminnan tavoitteiden osalta. Taulukossa 5 on esitetty muutamia esimerkkejä olioista ja niiden pohjalta luoduista kysymyksistä.

Liiketoiminnan tavoitteet voidaan muuttaa alitavoitteiksi käyttämällä apuna olio-kysymyslistaa. Kysymyksiä ryhmittelemällä voidaan hahmottaa alitavoitteita, joiden avulla saadaan lopulta muodostettua mittaustavoitteet. Taulukon 6 alitavoite, muutospyyntöprosessin parantaminen, on muodostettu taulukon 5 kysymysten pohjalta.

Alitavoitteiden tunnistamisen jälkeen jokaiselle olio-kysymyslistan kysymyksestä tulisi muodostaa oliot ja attribuutit. *Oliot* (entity) ovat oikean maailman kohteita tai tapahtumia (Fenton & Pfleeger, 1997). Ne voivat olla ohjelmistotuotannon näkökulmasta katsottuna tuotteita, prosesseja, resursseja, artefakteja, toimintaa, organisaatio, ympäristö tai rajoitteita (Park & al., 1996). Olion ominaisuuksia kutsutaan *attribuuteiksi* (attribute) (Fenton & Pfleeger, 1997). Attribuuttien tehtävänä on antaa tietoa oliosta. Esimerkiksi ihmisen attribuutteina voidaan pitää ikää, pituutta ja painoa, joiden tehtävänä on antaa tietoa ihmisestä eli oliosta. Koska attribuuttien avulla saadaan kerättyä tietoa, niiden avulla voidaan hahmottaa mahdollisia mittareita (Park & al., 1996). Taulukossa 7 on esitetty taulukon 6 kysymysten pohjalta muodostettuja olioita sekä attribuutteja.

Lopulliset mittaustavoitteet voidaan asettaa liiketoiminnan tavoitteista johdettujen kysymysten, alitavoitteiden, olioiden sekä attribuuttien avulla. Jokaisesta mittaustavoitteesta tulisi selvitä mittaamisen kohde (esimerkiksi tuote, prosessi tai resurssi), tarkoitus (esimerkiksi oppiminen,

Taulukko 5: Esimerkki olio-kysymyslistasta (Park & al., 1996).

Oliot	Kysymykset
Syöteet ja resurssit	
ihmiset	Ovatko henkilömme päteviä tuottamaan sellasia tuloksia, joita asiakkaat haluavat?
muutospyynnöt	Sisältävätkö asiakkaan muutospyynnöt riittävästi tietoa, jotta muutos saadaan korjattua tehokkaasti?
Sisäiset artefaktit	
muutospyynnöt	Kuinka paljon korjaamattomia muutospyyntöjä on?
Toiminnot	
kehitys	Onko kehitysprosessi selkeä asiakkaalle?
testaus	Hyväksyykö asiakas testauksen tulokset?
korjaus	Onko vikojen korjaamiseen käytetty aika asiakkaan mielestä kohtuullinen? Onko tilanteen edistyminen selkeä asiakkaalle?
Tuotteet ja sivutuotteet	
dokumentit	Ovatko dokumentit luettavia?
lähdekielinen ohjelma	Onko terminologia virheetön? Onko lähdekielinen ohjelma yhtenäinen dokumenttien kanssa? Onko käyttöliittymä yhtenäinen ja selkeä?

Taulukko 6: Alitavoitteen muodostaminen kysymyksiä ryhmittelemällä (Park & al., 1996).

Oliot	Kysymykset
Syöteet ja resurssit	
ihmiset	Ovatko henkilömme päteviä tuottamaan sellasia tuloksia, joita asiakkaat haluavat?
muutospyynnöt	Sisältävätkö asiakkaan muutospyynnöt riittävästi tietoa, jotta muutos saadaan korjattua tehokkaasti?
Sisäiset artefaktit	
muutospyynnöt	Kuinka paljon korjaamattomia muutospyyntöjä on?
Toiminnot	
kehitys	Onko kehitysprosessi selkeä asiakkaalle?
testaus	Hyväksyykö asiakas testauksen tulokset?
korjaus	Onko vikojen korjaamiseen käytetty aika asiakkaan mielestä kohtuullinen?
	Onko tilanteen edistyminen selkeä asiakkaalle?
Tuotteet ja sivutuotteet	
dokumentit	Ovatko dokumentit luettavia?
lähdekielinen ohjelma	Onko terminologia virheetön?
	Onko lähdekielinen ohjelma yhtenäinen dokumenttien kanssa?
	Onko käyttöliittymä yhtenäinen ja selkeä?

Alitavoite:
muutospyyntö-prosessin parantaminen

Taulukko 7: Olioiden ja attribuuttien muodostaminen kysymysten avulla (Park & al., 1996).

Alitavoite: muutospyyntöprosessin parantaminen		
Kysymys	Olio	Attribuutti
Sisältävätkö asiakkaan muutospyynnöt riittävästi tietoa, jotta muutospyyntö saadaan korjattua tehokkaasti?	muutospyynnöt, jotka saapuvat asiakkaalta	määrä (esimerkiksi saapuneiden muutospyyntöjen määrä) riittävyys (esimerkiksi oikein täytettyjen muutospyyntöjen prosenttiosuus) puutteiden jakautuminen (esimerkiksi laiminlyödyt tai väärin täytetyt muutospyynnöt)
Kuinka paljon korjaamattomia muutospyyntöjä on?	korjaamattomat muutospyynnöt	korjaamattominen muutospyyntöjen lukumäärä
Onko vikojen korjaamiseen käytetty aika asiakkaan mielestä kohtuullinen?	muutosten hallintaprosessi	asiakkaan oletus elinkaaren kestosta ajankäytön jakautuminen muutospyynnön toteutuksen/asentamisen aikana

ymmärtäminen, arvioiminen, ennustaminen, ohjaaminen ja parantaminen), ongelman kohdistuminen (esimerkiksi luotettavuus, laatu, käytettävyys ja toiminnallisuus), näkökulma (esimerkiksi käyttäjä, asiakas, ohjelmistosuunnittelija tai projektipäällikkö) ja ympäristö (esimerkiksi projekti, tuote tai tiimi) (Briand & al., 2002).

Basilin & al. (1994b) mukaan mittaamisen kohde voidaan saada selville prosessien ja tuotteiden malleja tutkimalla. Mittaamisen kohdetta voidaan pitää oliona (Park & al., 1996), jonka avulla voidaan tunnistaa toisia olioita sekä varteenotettavia hypoteeseja (Briand & al., 2002).

Organisaation menettelytapoja ja suunnitelmia tutkimalla sekä haastattelujen avulla voidaan selvittää mittaamisen tarkoitus ja ongelma, johon mittaamisella pyritään saamaan vastaus (Basili & al., 1994b). Mittaamisen tarkoituksen asettamisella voidaan paremmin ymmärtää, minkälaista tietoa kerätään (esimerkiksi ennustaminen vaatii tarkkaa tietoa, jotta tilastollisista mallista saataisiin luotettavia) ja miten paljon tietoa tarvitaan (esimerkiksi mitä enemmän tietoa saadaan, sitä tarkempia ennustuksia voidaan laskea). Ongelman kohdistaminen auttaa selventä-

Taulukko 8: Muutospyyntöprosessin parantamis- alitavoitteen pohjalta luotu mittaustavoite.

Kohde	muutospyynnöt
Tarkoitus	parantaminen
Ongelma	aikarajat
Näkökulma	projektipäällikkö
Ympäristö	projekti AB

mään attribuutteja, joita tarvitaan hypoteesien muodostamisessa. Esimerkiksi kuinka hinta on verrannollinen ohjelmiston kokoon nähden (Briand & al., 2002).

Basili & al. (1994b) ehdottavat tavoitteen näkökulman selvittämiseksi organisaatiomallien tutkimista. Tavoitteen näkökulma auttaa ymmärtämään, minkä tiedon hankkiminen voi olla kallista tai vaikeaa. Lisäksi näkökulman avulla voidaan hahmottaa paremmin, mitkä attribuutit kiinnostavat kohteena olevaa henkilöryhmää. Esimerkiksi käyttäjä voi olla kiinnostunut siitä, kuinka usein ohjelmisto menee toimintakyvyttömäksi ja ohjelmiston testaaja on kiinnostunut vikojen määrästä tietyllä ajanjaksolla (Briand & al., 2002).

Parkin & al. (1996) mukaan mittaamisen ympäristön määritteleminen on tärkeää, jotta mittaamisen asiayhteys ja laajuus saataisiin selville. Ilman riittävää tietoa siitä, ketkä käyttävät ja keräävät mittaustietoa, voidaan helposti tehdä virheellisiä johtopäätöksiä, sillä ympäristö vaikuttaa oleellisesti mittaamisen kohteeseen.

Taulukon 8 mittaustavoite on muodostettu taulukoiden 5, 6 ja 7 pohjalta ottamalla huomioon Briandin & al. (2002) esittämät asiat. Taulukon 8 kaltaisen mittaustavoitteen idean syntymiseen voi erityisesti vaikuttaa taulukon 7 kaltainen attribuutti, jonka huolenaiheena on asiakkaan oletus elinkaaren kestosta. Mittaustavoitteen tarkoituksena on pystyä vähentämään jokaisen muutospyynnön elinkaaren pituutta 10 %:lla, kuten kuvassa 12, jotta muutospyyntöjen korjausnopeudella pystyttäisiin vastaamaan asiakaan odotuksiin.

5.2.3 Tavoitteista johdetut kysymykset ja mittarit

Seuraavaksi mittaustavoitteiden pohjalta tulisi muodostaa kysymyksiä, joiden avulla saadaan muodostettua mittarit. Edellä olevat kysymykset olivat tarkoitettu johtamaan liiketoiminnan tavoitteet mittaustavoitteeksi, mutta niitä voidaan myös tarkentaa ja käyttää lähtökohtana mittaustavoitteista muodostettujen kysymyksien suunnittelussa.

Mittaustavoitteiden kysymyksien tunnistamisessa voidaan käyttää samoja menetelmiä kuin liiketoiminnan tavoitteiden tunnistamisessa, kuten esimerkiksi aivoriiheä (van Solingen, 1995).

Basilin (1992) mukaan kysymyksien pitäisi kattaa seuraavat osa-alueet: mittaamisen kohteen määrittely, kohteen laatutekijöiden määrittely ja palaute. Mikäli kohteena on prosessi, kuten tämän luvun esimerkissä, pitäisi prosessin määrittelyyn liittyvien kysymyksien koostua prosessin ja sen vaikutusalan (domain) yhdenmukaisuudesta. Prosessin yhdenmukaisuuteen liittyvät kysymykset voivat esimerkiksi koskea sitä, kuinka hyvin prosessin ja sen arviointimallien vaiheet ovat suoritettu. Vaikutusalan yhdenmukaisuuteen liittyvät kysymykset voivat esimerkiksi koostua prosessin suorittajan huolenaiheista. Esimerkiksi siitä, kuinka ymmärrettäviä muutospyyntöjen kuvaukset ovat. Toisena osa-alueena on prosessin laatutekijöiden, kuten luotettavuuden, määrittelyyn liittyvät kysymykset. Ne voivat koskea muun muassa kerätyn tiedon oikeellisuutta, laatumallien tehokkuutta, sopivuutta tai tulosten järkevyyttä. Esimerkiksi sitä, onko muutospyynnön kaikki tiedot kerätty asianmukaisesti ylös. Kolmas osa-alue koostuu palautteeseen liittyvistä kysymyksistä. Kysymyksien avulla pyritään selvittämään prosessin laatuun, ongelmiin sekä parannusehdotuksiin liittyviä vastauksia, kuten esimerkiksi sitä, tarvitsevatko muutospyyntöjen kerääjät lisäkoulutusta (Basili, 1992; Basili & Rombach, 1988).

Kuvassa 14 on yksinkertaistettu esitys siitä, millaisia kysymyksiä voidaan johtaa taulukossa 8 sekä kuvassa 12 esitetystä mittaustavoitteesta. Seuraavassa listassa on esitetty kuvan 14 kysymykset, ja mitä kullakin kysymyksellä esimerkiksi voitaisiin saada selville. Lisäksi jokaista kysymyksestä on laadittu oliot sekä attribuutit, jotka toimivat apuna mittareiden valinnassa.

- Kysymys 1: Kuinka nopeasti muutospyynnöt korjataan tällä hetkellä?
  - Merkitys: Tarkoituksena on saada selville tämän hetkinen muutospyyntöjen korjausnopeus.
  - Olio: korjaamaton muutospyyntö
  - Attribuutit: korjaukseen kuluva aika, korjauksen laatu
- Kysymys 2: Kuinka elinkaaren kesto on muuttunut?
  - Merkitys: Tarkoituksena on saada selville, onko muutospyyntöjen korjausnopeus parantunut, huonontunut vai pysynyt samana kuin aikaisemmin.
  - Olio: korjaamaton muutospyyntö
  - Attribuutit: korjaukseen kuluva aika, ajan muutos (vaihteluväli, joka kuluu muutospyynnön korjaamiseen)
- Kysymys 3: Onko ohjeistuksia noudatettu muutospyyntöjä korjattaessa?
  - Merkitys: Tarkoituksena on saada selville, kuinka hyvin ohjeistuksia on noudatettu muutospyynnön elinkaaren aikana.
  - Olio: muutospyynnön korjaaminen
  - Attribuutit: ohjeiden noudattaminen

Floracin & Charletonin (1999) mukaan mittareiden valitseminen voi olla vaikeaa ja viedä paljon aikaa, mikäli mittareiden valitsijoilla ei ole selkeää käsitystä, mitkä tekijät voivat vaikuttaa mittareiden arvoihin. Mittareiden valinnan lähtökohtana tulisi selvittää ongelmakohdat listaamalla kysytyt kysymykset ja kysymykset, joihin aiotaan vastata mittaustulosten avulla. Tämän jälkeen tulisi valita oliot ja attribuutit kysymysten pohjalta. Olioiden ja attribuuttien valinnassa

tulisi kiinnittää huomioita vallitseviin ja merkittäviin oliohin sekä attribuutteihin, jotta mittareiden avulla saataisiin kerättyä tarpeellista tietoa. Lopuksi tulisi testata valittujen mittareiden potentiaalinen hyöty esimerkiksi luonnostelemalla kaavioita ja taulukoita, jotka näyttävät, kuinka mittaustulokset käyttäytyvät.

Basilin & al. (1994b) mukaan mittareiden valinnassa tulisi kiinnittää huomiota saatavissa olevan tiedon määrään ja laatuun. Mikäli tieto on luotettavaa ja sen saaminen on helppoa, tulisi sen määrä maksimoida. Tällöin tiedon avulla voidaan tehdä luotettavia päätöksiä ja tiedon kerääminen ei tule kohtuuttoman kalliiksi. Lisäksi kypsempien kohteiden, kuten myöhästyneiden muutospyyntöprosessien, mittaamiseen tulisi käyttää objektiivisia mittareita. Usein niiden tulokset ovat vakaampia kuin subjektiivisten mittareiden, kuten projektipäällikön mielipiteiden, antamat tulokset.

Kuvassa 15 on yksinkertaistettu esitys siitä, millaisia mittareita voidaan johtaa kuvan 14 kysymyksistä. Mittareita suunniteltaessa voidaan käyttää apuna edellisen listan attribuutteja, jotka on muodostettu listassa olevien kysymyksien pohjalta. Seuraavassa listassa on esitetty kuvassa 15 esitetyt mittarit. Jokaisen mittarin kohdalla on kerrottu, mitä tietoa mittarin avulla voitaisiin saada ja mitä hyötyä kerätystä tiedosta voisi olla.

- Mittari 1: Keskimääräinen elinkaaren kesto
  - Merkitys: Tarkoituksena on saada selville, kuinka kauan muutospyynnön korjaamiseen kuluu aikaa. Tämä mittarin avulla kerätään pohjatietoa, jota käytetään mittarin 4 lopputuloksen laskemiseen.
- Mittari 2: Keskimääräinen elinkaaren kesto muutospyyntötyypin mukaan lajiteltuna
  - Merkitys: Tarkoituksena on saada selville, vaikuttaako muutospyyntötyypin vaikeusaste muutospyynnön korjausnopeuteen. Mikäli vaikutus on nähtävissä, osaa projektipäällikkö varata vaikeammille muutospyyntötyypin korjauspyynnöille enemmän resursseja.

- Mittari 3: Myöhästyneiden muutospyyntöjen % - osuus
  - Merkitys: Tarkoituksena on saada selville, kuinka usein muutospyyntöä ei ehditä korjata sille varatussa ajassa. Mikäli myöhästyneiden muutospyyntöjen osuus on merkittävä, tulisi projektipäällikön ensisijaisesti keskittää resurssit niiden korjaamiseen edellyttäen, että muutospyyntötyypit ovat kiireellisiä. Tämän jälkeen projektipäällikkö voi keskittyä varsinaiseen mittaustavoitteeseen.
- Mittari 4: Elinkaaren keston vaihteluväli
  - Merkitys: Tarkoituksena on saada selville, onko näkyvää kehitystä tapahtunut muutospyyntöjen korjausnopeudessa. Mittarin avulla voidaan myös nähdä, onko korjausnopeus taantunut tai pysynyt samana. Mittarin arvo saadaan laskemalla nykyinen keskimääräinen elinkaaren kesto jaettuna edellisen aikavälin keskimääräisellä elinkaaren kestolla.
- Mittari 5: Havaittujen virheiden % -osuus tarkastuksissa
  - Merkitys: Tarkoituksena on saada selville virheiden lukumäärä tarkastuksien avulla. Jos virheiden % -osuus on suuri, voi syynä olla huono ohjeistuksien noudattaminen. Tämä johtaa hidastuneeseen muutospyynnön elinkaaren kestoon.
- Mittari 6: Projektipäällikön subjektiivinen mielipide
  - Merkitys: Tarkoituksena on saada selville, onko projektipäällikkö havainnut ettei ohjeistuksia noudateta riittävän hyvin tai onko ohjeistuksissa jotain puutteita. Ohjeistuksien noudattamisella pyritään lyhentämään muutospyynnön elinkaaren kestoa.

Ylläolevien kysymys ja mittarilistojen avulla on tarkoitus antaa näkemys siitä, millaisia kysymyksiä kuvan 12 tavoitteesta voidaan johtaa. Tavoitteesta voidaan toki johtaa myös muita kysy-

myksiä ja mittareita. Esimerkiksi muutospyynnön kokoa voitaisiin mitata havaittujen virheiden % -osuuden ohella ja selvittää korreloivatko ne keskenään.

Lopuksi mittarit tulisi integroida haluttuun ympäristöön. Integrointi koostuu analysoinnista, diagnosoinnista ja käyttöönotosta (Florac & Charleton,1999; Park & al., 1996). Analysointivaiheen aikana tulisi selvittää, mitä mittareita organisaatio käyttää tällä hetkellä ja kuinka niiden avulla kerätään tietoa. Tällöin ymmärretään, kuinka uudet mittarit tulisi liittää aiempien mittareiden joukkoon. Diagnosoinnin avulla saadaan selville, kuinka hyvin organisaation käyttämät mittarit tukevat uusia mittareita. Tämän avulla voidaan esimerkiksi havaita, mitä mittareita voidaan hyödyntää uusien mittareiden kanssa, mitkä mittarit tarvitsevat korjauksia tai puuttuuko jokin mittari kokonaan. Viimeisenä vaiheena on käyttöönotto, jonka tarkoituksena on ratkaista analysoinnin ja diagnosoinnin aikana huomatut asiat. Tämän jälkeen muutokset otetaan käyttöön ja mittaaminen voi alkaa.

6 ESIMERKKITAPAUS YRITYKSEN YLLÄPIDON MITTAAMISESTA

Tässä kohdassa tutkitaan erään suomalaisen yrityksen mittausaineistoa ylläpidon näkökulmasta. Ensiksi kerrotaan, mitä yritys on mitannut ja sen jälkeen tutkitaan, kuinka yrityksen mittaaminen ja mittaustiedon luokittelu ovat onnistuneet. Lopuksi esitetään muutoksia nykyiseen luokittelukäytäntöön ja etsitään mittareita GQM-menetelmän avulla.

6.1 Analysointi

Yrityksen toimialaan kuuluu mekaanisten ja sähkömekaanisten tuotteiden valmistus, jotka toteutetaan sulautettuja järjestelmiä apunakäyttäen.

Yritys on kerännyt mittausaineistoa vuodesta 2002 lähtien. Mittareiden avulla on mitattu kehityksen aikaisia ja asiakkaan ilmoittamia virheiden lukumääriä, lähetettyjen ja korjattujen virheiden osuutta versioittain, suurimpia yhteydenottajia ja asiakastyytyväisyyttä. Asiakastyytyväisyyskyselyjen avulla on selvitetty tuotteiden teknistä laatua, ohjelmistojen toimitusaikaa, virheettömyyttä ja helppokäyttöisyyttä sekä ohjelmistojen tukipalvelujen laatua.

Virheitä on kerätty yhteensä 34 eri tuotteen osalta. Neljän vuoden aikana kehityksen aikaisia virheitä on raportoitu yhteensä noin 1500 kappaletta ja asiakkaan ilmoittamia virheitä noin 4500 kappaletta. Tiedot ovat pääasiallisesti kerätty asiakkaille tarkoitetun käyttötuen avulla.

Vuonna 2005 yritys on laatinut raportin, jonka tarkoituksena on tarkentaa mittausprosessia. Raportissa on esitetty erityyppiset ohjelmistovirheet, suunnitelma virheiden poistamiseksi, sekä kerrotaan, kuinka toteutusta ja korjausten vaikutusta seurataan. Raportissa esitettävät ohjelmistovirheet ovat luokiteltu virhelähteiden ja asiakasvirheiden mukaan. Virhelähteillä tarkoitetaan erilaisia kehityksen aikaisia virheitä, kuten asennusvirheitä, virheitä, jossa ohjelmointia ei ole

voitu suorittaa sekä virheitä, jotka ovat ilmenneet koulutuksessa tai käyttöohjeessa. Asiakasvirheillä tarkoitetaan asiakkaan kontaktien jakautumista. Nämä virheet ovat jaoteltu käyttötuen puutteeseen, ohjelmavirheisiin, puutteelliseen käyttöohjeeseen sekä asennuksen aikaisiin virheisiin.

Raportissa on esitetty myös suunnitelma virheiden poistamiseksi. Tarkoituksena on luoda jokaisen virheen osalta syy-seurauskaavio, jonka avulla pyritään selvittämään mahdollinen virheen syy. Raportin mukaan virheiden korjaamista seurataan taulukon avulla, johon kirjataan ongelman syy, versio, jossa ongelma esiintyi, korjaussuunnitelma, toimenpiteet korjaamista varten, korjauksen toteuttaja, korjaukseen kuluva aika ja arviointiperusta korjauksen valmistumiselle. Korjausten vaikutuksia on pystyttävä seuraamaan graafisesti. Vaatimuksena on esittää kunkin tuotteen kohdalta kumulatiivinen myynti 12 kuukauden osalta, tuotteen myynti ja virheiden lukumäärä kuukausittain. Lisäksi virhelähteet ja asiakasvirheet tulisi esittää omina esityksinään sekä niissä tulisi mainita kumulatiivinen virheprosentti.

Ennen raportin laatimista kehityksen aikaisia virheitä on kerätty kahden tuotteen osalta. Taulukossa 9 on esitetty näiden tuotteiden luokittelu. Tuotteen X luokittelu koostuu virheen havaitsijasta, tilasta eli virheen korjausprosessin vaiheesta, prioriteetista eli virheen tärkeydestä (prioriteetin yhteydessä yritys on ilmaissut virheen vakavuusasteen), vastuuhenkilöstä, kuvauksesta, havaitsemispäivästä, vireillä olosta eli odotukseen kuluvasta ajasta, meneillä olosta eli aikavälistä joka käytetään virheen korjaamiseen, korjauspäivästä, hyväksymispäivästä ja korjaukseen kuluneesta ajasta, joka ilmoitetaan tunteina.

Tuotteen Y luokittelu on pääpiirteissään sama kuin tuotteen X luokittelu. Merkittävänä erona on, että tuotteen Y kohdalla on luokiteltu projektit, joissa virhe esiintyy. Lisäksi jokaiselle virheelle on muodostettu oma yksilöllinen numero ja jokaisen virheen kohdalla on kerrottu järjestelmä ja versio, missä virhe esiintyy. Tuotteen X hyväksymispäivä on ilmoitettu tuotteen Y luokittelussa viimeisimpänä päivityksenä ja vireillä sekä meneillä olo on jätetty kokonaan pois luokittelusta.

Taulukko 9: Kehityksen aikaisten virheiden nykyinen luokittelu tuotteiden X ja Y osalta.

	Tuote X	**Tuote Y**
1.	virheen havaitsija	projekti
2.	tila	tunnistenumero
3.	prioriteetti	järjestelmä ja versio
4.	vastuuhenkilö	kuvaus
5.	kuvaus	prioriteetti
6.	havaitsemispäivä	tila
7.	vireillä olo/odotus aika	vastuuhenkilö
8.	meneillä olo	viimeisin päivitys
9.	korjauspäivä	virheen havaitsija
10.	hyväksymispäivä	havaitsemispäivä
11.	korjaukseen kulunut aika	korjaukseen kulunut aika
12.	----------	korjauspäivä

Asiakasvirheitä on kerätty kaikkien tuotteiden osalta, joiden luokittelu on esitetty taulukossa 10. Jokaisen asiakasvirheen osalta on kirjattu ylös kuukausi, jolloin virhe on tapahtunut; tuote, jossa virhe esiintyi; tuotteen versio; virheen syy, joka noudattelee vuonna 2005 laaditussa raportissa esiintyvää virhelähteiden luokittelua ja kuukauden aikana esiintyneiden virheiden määrä kaikkien tuotteiden osalta.

Taulukko 10: Asiakasvirheiden nykyinen luokittelu.

	Asiakkaan ilmoitukset
1.	kuukausi
2.	tuote
3.	versio
4.	virheen syy
5.	virheiden määrä/kuukausi

Mittausaineisto kattoi myös muuta informaatiota, mutta sitä ei käsitellä tässä pro gradu tutkiel-

massa, koska aineisto ei liity tutkielman aiheeseen.

Mittausaineiston analysointi painottuu tuotteiden X ja Y analysointiin, koska muiden tuotteiden osalta ei ole kerätty kehityksen aikaisia virheitä. Asiakasvirheitä on raportoitu 16 eri tuotteen osalta ja asiakastyytyväisyyttä on tarkasteltu yhtenäisesti kaikkien tuotteiden osalta.

Mittausaineiston virhejakaumia tutkimalla voidaan havaita, että tuotteen X virhejakauma puolittui vuonna 2003 vuoden 2002 virhejakaumasta. Virhejakauma saatiin puolittumaan myös seuraavana vuonna ja vuoden 2005 heinäkuun loppuun mennessä tuotteen X kohdalla ei ollut havaittu enää yhtään virhettä. (Vuoden 2005 mittausaineistoa oli kerätty heinäkuuhun saakka.) Liitteestä 3 on nähtävissä tarkemmat tuotekohtaiset luvut.

Tuotteen Y kohdalla voidaan havaita, että virheet olivat kasvaneet yli puolella vuodesta 2002 vuoteen 2003 mennessä. Virheet saatiin puolittumaan vuonna 2004, mutta vuonna 2005 virheitä esiintyi heinäkuun lopussa saman verran kuin edellisenä vuotena. Siksi tuotteen Y kohdalla mittaamista pitäisi tarkentaa, jotta saataisiin selville, mistä virheet mahdollisesti johtuvat. Tämä ilmiö on ollut myös havaittavissa asiakasvirheiden kohdalla. Yritys onkin päättänyt kehittää mittaamisen luokittelua ottamalla huomioon ohjelmistovirheiden tyypit. Tämän ansiosta yritys voi mahdollisesti nähdä, mille osa-alueille virheet keskittyvät. Tiedon avulla yritys voi keskittää resurssinsa paremmin ongelman ydinkohtaan.

Ensimmäiseksi kiinnitin tuotteiden kohdalla huomiota niiden luokitteluun, joka on esitetty taulukossa 9. Käytännössä tuotteiden X ja Y luokittelu on lähes samanlainen, mutta jotkin tiedot poikkesivat toisistansa. Mittaamisen yhtenäisyyden kannalta olisi hyvä, jos luokittelu olisi samanlainen kaikkien tuotteiden osalta. Mikäli yritys ei koe sitä tarpeelliseksi, olisi kuitenkin hyvä, jos tuotteen X luokittelussa käytettäisiin järjestelmän versiota, sillä raportin mukaan eri versioiden virheet pitäisi pystyä laskemaan omana kokonaisuutena.

Mielestäni varsinaiseksi ongelmaksi luokittelussa voi muodostua se, että virheiden kuvaukset eivät ole kattavat, jolloin virheen uudelleenpaikannus hankaloituu. Mikäli virheiden kuvauksia ei esitetä missään muualla tarkemmin, tulisi tässä kohdin kiinnittää siihen huomiota. Esimer-

kiksi tuotteen X kohdalla vuoden 2005 mittausaineistossa on raportoitu virheen kuvausta rivillä 245 seuraavasti: "X kaatuu". Vastaavanlaisia puutteellisia nimeämistapoja on useita. Virheen kuvauksesta ei selviä, mikä aiheutti tuotteen kaatumisen, missä kohdin se tapahtuu ja kuinka se saadaan toistumaan uudelleen.

Mittausaineistoa tutkimalla kiinnitin huomiota siihen, että raportissa mainittu jako ei välttämättä ole kattava virhelähteiden jaotteluun. Siksi olisikin hyvä lisätä jaotteluun kohta, jota voidaan käyttää, jos virheelle ei löydy valmista virhetyyppiä. Asiakasvirheiden kohdalla toistui sama ongelma kuin virhelähteiden kohdalla. Asiakasvirheiden luokitteluun tulisi myös lisätä kohta uudelle virhetyypille. Tällöin luokittelusta tulee luotettavampi, koska jaottelu pysyy yhtenäisenä.

Raportissa esitettiin taulukko jossa kerrottiin, kuinka virheiden korjausta tulisi luokitella. Taulukossa esitettiin kohta, joka kertoi ongelman syyn. Raportista ei käynyt ilmi, tarkoitettiinko ongelman syyllä esimerkiksi virhetyypin lähdettä, kuten ohjelmointia ei voitu suorittaa, vai kuvausta siitä, miksi virhetyypin lähde tuotti ongelmia, kuten syytä ohjelmoinnin keskeytymiseen. Mikäli ongelman syyllä tarkoitetaan virhetyyppiä, olisi hyvä, jos toteutuksen seurannasta kävisi ilmi, mistä syystä ongelma johtui. Lisäksi toteutuksen seurantaa voisi tarkkailla paremmin, jos jokaisen ongelman voisi yksilöidä, kuten taulukossa 9 on tehty tuotteen Y kohdalla tunnistenumero-tietueen avulla.

Yritys on toivonut saavansa käyttöön trendimittareita, joiden pitäisi näyttää, onko jokin asia menossa parempaan vai huonompaan suuntaan. Trendimittareiden käyttäminen ei ole aina kovinkaan selkeää, sillä useasti mittareiden tulkitsijat joutuvat tekemään paljon töitä, että he voisivat varmuudella sanoa, mihin suuntaan jokin asia on menossa. Lisäksi trendimittarit tarvitsevat paljon historiatietoa, jotta niistä saataisiin luotettavia. Siksi ne useasti ovatkin jäljessä nykytilanteesta, jolloin ne eivät välttämättä ole kovin luotettavia nykytilanteen arviomisessa (Tenhunen, 2006). Trendimittareita suunniteltaessa tulisi pohtia, mitkä asiat ovat kriittisiä ja kuinka niihin tulisi reagoida. Trendimittareiden ongelmaksi voikin koostua se, että mittaajat olettavat

asioiden olevan kriittisempiä kuin ne todellisuudessa ovat. Seuraavan esimerkin tarkoituksena on osoittaa, kuinka trendimittareiden tunnistaminen ei ole aina mahdollista olemassa olevan tiedon perusteella.

Mittausaineistoa tutkiessani yritin etsiä mittauksen kannalta oleellisia asioita ja osoittaa niiden kriittisyyden. Kiinnitin erityisesti huomiota siihen, että vuonna 2005 tuotteella X ei esiintynyt ollenkaan kehityksen aikaisia ohjelmavirheitä, mutta asiakkaat olivat ottaneet yritykseen 19 kertaa yhteyttä ohjelmavirheen takia. Mittausaineistosta ei ilmennyt, oliko vuonna 2005 ilmestynyt tuotteen X lopullinen versio, koska kehityksen aikaisia virheitä ei ollut raportoitu. Lisäksi asiakastyytyväisyyskyselystä selvisi, että asiakkaat olivat olleet virheettömyyteen tyytyväisempiä vuonna 2004, vaikka asiakkaiden ilmoittamia ohjelmavirheitä esiintyi vuonna 2004 neljä kertaa enemmän kuin vuonna 2005. Toisaalta asiakastyytyväisyyden luotettavuudesta ei voida sanoa mitään, koska tuotteelle X ei ollut tehty omaa asiakastyytyväisyyskyselyä, vaan kyselyssä oli otettu myös muut tuotteet huomioon. Kuten esimerkistä voidaan huomata, trendimittarin tunnistaminen jäisi lopulta arvailujen varaan. Tietenkin mahdollisen lisätiedon, kuten tuotekohtaisten asiakaskyselyjen avuilla, trendimittareiden muodostaminen voisi onnistua, mutta en havainnut, että niiden luominen olisi tässä tapauksessa järkevää.

Seuraavaksi kiinnitin huomiota tuotteiden X ja Y kehityksen aikaisten virheiden määrään. Vertasin jokaisen vuoden aikana havaittujen ja korjattujen puuteiden lukumääriä keskenään. Samalla laskin, kuinka paljon sen vuoden aikana asiakkailta oli tullut valituksia ohjelmavirheiden takia. Lopuksi vertasin saamiani tuloksia myös tuotteiden välillä. Mielenkiintoisinta oli, että tuotteen Y osalta vuonna 2003 oli korjattu 95 % virheistä heinäkuun loppuun mennessä ja asiakkaat eivät olleet valittaneet sinä aikana ollenkaan ohjelmavirheistä. Asiakkaat olivat kuitenkin valittaneet muista virheistä 45 kertaa. Vuonna 2005 lähetettyjä virheitä oli puolet vähemmän kuin vuonna 2003 ja virheitä oli korjattu 96,2 %. Siitä huolimatta asiakkaat olivat tehneet 308 valitusta, joista 32 kappaletta oli luokiteltu ohjelmavirheisiin.

Tulos voi johtua monesta eri syystä, kuten esimerkiksi siitä, että asiakkaat ovat käyttäneet tuo-

tetta huomattavasti enemmän vuonna 2005. Lisäksi havaitsin mittausaineistoa tutkiessani, että kaikkiin asiakkaan ilmoituksiin ei oltu merkitty, mistä syystä virhe johtui. Täten ohjelmavirheitä on voinut olla enemmän etenkin vuonna 2003, sillä tuolloin virheen syy jätettiin merkitsemättä joka kolmannesta virhetapauksesta. Vuonna 2004 vastaava luku oli 17 % ja vuonna 2005 heinäkuuhun mennessä 12 %. Jatkossa olisikin hyvä, että kaikki luokittelussa esitetyt kohdat täytettäisiin, jotta tuloksista saataisiin luotettavia.

Toisaalta tuloksen syy saattaa johtua siitä, että tuotteen Y korjaukset ovat olleet puutteellisia vuonna 2005. Puutteellinen korjaus ei korjaa ongelmaa tai se aiheuttaa korjauksen aikana muita ongelmia (Kan, 2001). Koska tuotteen Y virheistä on vuonna 2005 korjattu 96,2 %, olisi hyvä mitata tarkemmin tuotteen korjauksen luotettavuutta, jotta mittausaineiston avulla voitaisiin saada selville, johtuuko virheiden lukumäärä huolimattomasti korjatuista virheistä.

Lopuksi kiinnitin tarkemmin huomiota korjaamattomien muutospyyntöjen määrään. Vuonna 2003 tuotteen X muutospyynnöistä jätettiin korjaamatta 15,6 % ja tuotteen Y muutospyynnöistä 14 %. Vuotta myöhemmin tuotteen X kohdalla muutospyynnöistä jätettiin korjaamatta 73,7 %, vaikka asiakkaan ilmoittamia ohjelmavirheitä esiintyi samana vuonna 115 kappaletta, joka 62,7 % keskivertoa enemmän. Samana vuonna tuotteen Y muutospyynnöistä jätettiin korjaamatta 44,1 %. Vuonna 2005 heinäkuuhun mennessä tuotteen X kohdalta oli korjattu kaikki sinä vuonna ilmoitetut muutospyynnöt ja tuotteen Y osalta korjaamattomia muutospyyntöjä esiintyi vain 3,8 %.

Kiinnitin huomiota kahteen asiaan, joista ensimmäinen oli korjaamattomien muutospyyntöjen lukumäärä ja toinen korjaamattomien muutospyyntöjen suhteellinen määrä verrattuna asiakkaiden ilmoittamien ohjelmavirheiden määrään.

Mielestäni korjaamattomien muutospyyntöjen määrään tulisi kiinnittää huomiota, sillä jos muutospyynnöistä jätetään huomioimatta yli 70 %, tulisi miettiä, ovatko muutospyynnöt aiheellisia. Toisaalta mittausaineisto antoi viitteitä siihen, että muutospyynnöt ovat aiheellisia, sillä esimerkiksi vuonna 2003 tuotteen X kohdalla asiakkaat olivat havainneet 115 ohjelmavirhettä, joka on

62,7 % keskivertoa enemmän. Yrityksen tulisikin selvittää, johtuvatko asiakkaan ilmoittamat ohjelmavirheet korjaamattomista muutospyynnöistä vai väärin korjatuista muutospyynnöistä. Ohjelmavirheiden selvittäminen voi koitua erittäin työlääksi, mutta tiedon avulla voitaisiin saada selville, ovatko korjatut muutospyynnöt olleet virheellisiä vai korjaamattomat muutospyynnöt epäselkeitä, jolloin niiden avulla ei ole saatu riittävästi tietoa virheen laadusta.

Seuraavassa kohdassa kerrotaan tarkemmin, minkälainen luokittelu ja mitkä mittarit voisivat soveltua tämän kohdan ongelmien selvittämiseen.

6.2 Ratkaisuehdotuksia

Yrityksen mittausaineiston analysointi johti luokittelun uudistamiseen sekä uusien mittareiden määrittämiseen.

6.2.1 Luokittelu

Yrityksen luokittelu ei mielestäni ole tällä hetkellä täysin kattava, koska tuotteiden X ja Y kehityksen aikaisten virheiden luokittelu ei ollut yhtenäistä. Lisäksi luokittelua pitäisi tarkentaa, jotta muun muassa testauksen aikana virheet voitaisiin paikantaa paremmin. Havaitsin myös tarpeen uusien mittareiden käyttöönotolle, sillä nykyisten mittareiden avulla ei saada tarpeeksi tietoa asiakastyytyväisyydestä ja ylläpidon laadusta.

Ensimmäiseksi havaitsin tarpeen virhetyyppien uudelleen luokittelulle. Mittausaineistosta ilmeni, että virhetyyppejä ei pitäisi rajata niin tarkasti kuin ne ovat tällä hetkellä rajattu. Virhetyyppien luokittelussa esiintyi kaksi eri ongelmaa. Ensimmäisenä ongelmana oli se, että virhettä ei oltu luokiteltu mihinkään valmiiseen kategoriaan. Toisena ongelmana oli puolestaan se, että virheelle oli keksitty oma luokittelu. Siksi ehdotankin, että virhetyyppejä tulisi laajentaa vaihtoehtoisella tyypillä, jota voitaisiin käyttää, mikäli virhetyypille ei ole muuta vaihtoehtoa

tarjolla. Tällöin virhetyyppejä ei jätetä kokonaan sijoittamatta ja jaottelu pysyy yhtenäisempänä.

Raportissa oli esitelty, kuinka virhekorjauksien toteutusta tulisi seurata. Toteutuksen seurannassa olisi hyvä kertoa virhetyypin ohella ongelman aiheuttaja. Täten ratkaisusuunnitelman muodostaminen helpottuisi. Lisäksi jokaisella virheellä tulisi olla yksilöivä tunnus, jolloin kaikki virheestä löytyvät tiedot voitaisiin tarvittaessa jäljittää helpommin. Tällöin yksilöivä tunnus tulisi lisätä jokaisen tuotteen mittareihin.

Mikäli virheiden kuvauksia ei esitellä tarkemmin missään muualla, tulisi niitä tarkentaa, jotta virheet saataisiin paikannettua ja niiden korjaukset testattua paremmin. Kiinnitin huomioni siihen, että joidenkin virheiden kohdalla ei ollut mainittu, mikä virheen aiheutti (esimerkiksi vääränlainen syöte tai virheellinen painike), kuinka se saadaan toistumaan (toimintaohjeet virheen aikaansaamiseksi) ja missä kohdin se tapahtui (esimerkiksi käyttöliittymän lomake).

Taulukkoon 11 on koottu uudistettu ja yhtenäisempi virheluokittelu, jota voidaan soveltaa sekä kehityksen aikaiselle, että asiakasvirheille. Luokittelu on muodostettu taulukoiden 9 ja 10 sekä mittausaineistossa esiintyvien puutteiden perusteella.

Taulukon 11 luokittelu noudattelee pääpiirteissään tuotteen Y luokittelua. Tuotteen X luokittelusta on huomioitu vireilläolo eli aika, joka kuluu odotukseen ennen virheen korjausta. Meneilläolo aika on jätetty luokittelusta pois, sillä se voidaan johtaa vähentämällä korjauspäivästä havaitsemispäivä ja vireilläolo aika. Kohdat 14-17 ovat uusia luokittelun ominaisuuksia, joiden tarpeellisuus perusteltiin yllä olevassa tekstissä.

6.2.2 Mittarit

Luokittelun lisäksi havaitsin tarpeen uusille mittareille. Pyrin selvittämään mittareita GQM-menetelmän avulla. Yrityksen tavoitteena oli saada selville, onko sen nykyinen mittaaminen asianmukaista ja pitäisikö sitä muuttaa jollakin tavoin. Mittausaineisto kattoi suurelta osin yllä-

Taulukko 11: Uudistettu, yhtenäisempi virheluokittelu.

	Uusi luokittelu
1.	projekti
2.	tunnistenumero
3.	järjestelmä ja versio
4.	kuvaus
5.	prioriteetti
6.	tila
7.	vastuuhenkilö
8.	viimeisin päivitys
9.	virheen havaitsija
10.	havaitsemispäivä
11.	korjaukseen kulunut aika
12.	korjauspäivä
13.	vireilläolo/odotus aika
14.	virhelähteet
15.	virheen aiheuttaja
16.	toimintaohjeet
17.	sijainti

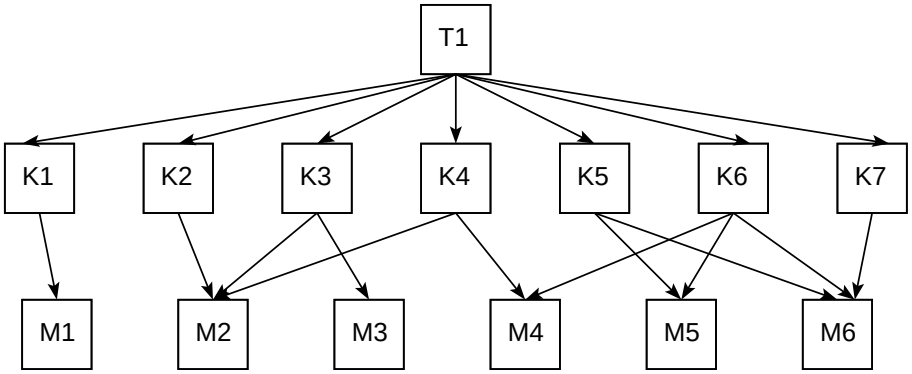
pitoon liittyvää aineistoa. Lisäksi aineistossa oli selvitetty asiakastyytyväisyyttä yleisellä tasolla kaikkien tuotteiden osalta. Pyrin huomioimaan nämä molemmat asiat mittareita suunniteltaessa. Kysymykset hahmottuivat, kun tutkin mittausaineissa esiintyviä ongelmakohtia. Kysymyksistä johdin mittareita, joiden avulla ylläpitoa voidaan seurata tarkemmin. Päädyin taulukossa 12 esitettyyn GQM-malliin. Lisäksi kuvasta 20 on nähtävissä tarkemmin, kuinka kysymykset ja mittarit linkittyvät toisiinsa.

Asiakastyytyväisyyden parantaminen on erittäin laaja tavoite, josta voidaan johtaa lukemattomia kysymyksiä ja mittareita. Lavazzan (2000) mukaan yhdestä tavoitteesta on mahdollista

Taulukko 12: Yrityksen GQM-mallin tavoite, kysymykset ja mittarit.

Tavoite:
T1. Asiakastyytyväisyyden parantaminen ylläpidossa
Kysymykset:
K1. Kerätäänkö virheistä riittävästi tietoa?
K2. Ovatko asiakkaat olleet tyytyväisiä ylläpitoon?
K3. Mikä on muutospyyntöjen korjausnopeus?
K4. Onko muutospyyntöjen korjaukset laadukkaita?
K5. Miksi kehityksen aikaisia virheitä ei ole esiintynyt, vaikka asiakkaat ovat ilmoittaneet virheistä?
K6. Miksi tuotteella on esiintynyt asiakkaiden ilmoittamia virheitä eniten, vaikka korjauksia on tehty huomattavasti enemmän kuin muille tuotteille?
K7. Miksi kaikkia raportoituja virheitä ei ole korjattu?
Mittarit:
M1. Puutteellisesti täytetyt muutospyynnöt
M2. Tuotekohtaiset asiakastyytyväisyyskyselyt
M3. Muutospyynnön korjausnopeus
M4. Virheellisten korjauksien lukumäärä
M5. Testauksen edistymisen ja laadun seuranta
M6. Tarpeettomat muutospyynnöt

luoda jopa yli 100 mittaria. Tämänkin GQM-mallin avulla kysymyksiä ja mittareita olisi voitu suunnitella enemmän, mutta koin järkevämmäksi keskittyä selkeään kokonaisuuteen. Goldpracticien (2005) mukaan mittareita ei tulisi kerätä kohtuuttomasti, sillä mittaustiedon kerääminen ja tulkinta vievät paljon resursseja. Lisäksi mittareita tulisi rajoittaa, jotta keskityttäisiin vain olennaisen asian tutkimiseen.



Kuva 20: GQM-mallin rakenne.

Koska yrityksen mittausaineisto painottuu korjaavaan ylläpitoon, otin mittareiden valinnassa myös huomioon Buckleyn & Chillaregen (1995) tutkimuksen. Buckley & Chillarege (1995) ovat ensimmäisiä tutkijoita, jotka ovat osoittaneet asiakastyytyväisyyden ja ylläpidon välisen yhteyden ohjelmistotuotannossa. He ovat tutkineet, mitkä korjaavan ylläpidon tekijät vaikuttavat asiakastyytyväisyyteen. He tulivat siihen tulokseen, että asiakastyytyväisyyden kannalta tärkeintä oli, että korjaukset olivat virheettömiä. Seuraavaksi tärkeimpänä he pitivät sitä, että ohjelmistoissa esiintyi vähän virheitä. Lisäksi korjausnopeudella oli vaikutusta asiakastyytyväisyyteen, mutta sen merkitys ei ollut niin suuri.

Jotta mittareiden tunnistaminen olisi helpompaa, muodostin Parkin & al. (1996) mukaisesti kysymyksistä K1-K7 olioita ja attribuutteja, jotka ovat esitetty taulukossa 13. Kysymyksien keskeisimmiksi attribuuteiksi nousi muutospyyntöjen määrä, vakavuus ja korjausaika. Näiden tietojen avulla huomasin tarpeen mittareiden M1-M6 muodostamiselle.

Mittari 1: Puutteellisesti täytetyt muutospyynnöt

Yrityksen luokittelu on ollut osittain epätäydellistä, sillä jokaisesta kehityksen aikaisesta tai asiakkaan ilmoittamasta virheestä ei ole saatavilla tarpeeksi tietoja muutospyyntöä varten. Siksi yritys voisi mitata, että kaikki vaadittavat tiedot kerätään, jotta kaikkien muutoksien tekeminen olisi mahdollista.

Taulukko 13: GQM-mallista muodostettuja olioita ja attribuutteja.

Kysymys	Olio	Attribuutti
K1	virheestä kerätty tieto	riittävyys
K2	ylläpito	virheiden määrä, korjausaika
K3	korjaamaton muutospyyntö	lukumäärä, korjausaika
K4	korjaamaton muutospyyntö	virheiden määrä, vakavuus
K5	asiakkaiden ilmoittamat virheet	virheiden määrä, vakavuus
K6	korjattu muutospyyntö	virheiden määrä, vakavuus
K7	korjaamaton muutospyyntö	lukumäärä suhteessa korjattuihin, vakavuus

Taulukossa 11 on esitetty tiedot, jotka tulisi kerätä jokaisesta virheestä. Jokaisen virheen osalta tulisi kerätä kohdat 1-10 ja 15-17, jotta mahdollinen muutospyyntö voidaan toteuttaa. Mikäli muutospyyntö toteutetaan, tulisi myös kohdat 11-14 olla täytettyinä korjauksen jälkeen.

Puutteellisesti täytettyjä muutospyyntöjä tulisi mitata erikseen korjaamattomista sekä korjatuista muutospyynnöistä.

$$k_0 = \frac{p_{k_0}}{mp} * 100\%, missä$$

$k_0 = korjaamattomat\, puutteellisesti\, täytetyt\, muutospyynnöt$

$p_{k_0} = puutteelliset\, muutospyynnöt,\ joiden\ \ kohdista\, 1-10\, ja\, 15-17\, ei\, ole\, kerätty\, tietoa$

$mp = kaikki\, korjaamattomat\, muutospyynnöt$

$$k_1 = \frac{p_{k_1}}{mp} * 100\%, missä$$

$k_1 = korjatut\, puutteellisesti\, täytetyt\, muutospyynnöt$

$p_{k_1} = puutteelliset\, muutospyynnöt,\ joiden\ \ kohdista\, 11 - 14\, ei\, ole\, kerätty\, tietoa$

$mp = kaikki\, korjaamattomat\, muutospyynnöt$

Mittareita tulisi käyttää kunnes puuttuvia tietoja ei enää esiinny. Tämän jälkeen mittaria voidaan käyttää satunnaisesti, jotta nähdään, kerätäänkö virheistä edelleen riittävästi tietoa.

Mittari 2: Tuotekohtaiset asiakastyytyväisyyskyselyt

Analysointivaiheessa todettiin, että yritys on tehnyt vuosittain asiakastyytyväisyyskyselyn, jossa kartoitetaan yrityksen tuotteiden yleistä tilaa. Kysely on siinä mielessä hyvä, että esimerkiksi yrityksen johto saa siitä suuntaa antavan kuvan yrityksen ohjelmistotuotteista. Jotta asiakastyytyväisyyskyselyjä voitaisiin hyödyntää enemmän ylläpidon mittaamisessa, tulisi kyselyjä tarkentaa.

Yritys on mitannut asiakastyytyväisyyttä järkevästi, sillä se on kysyttävän ominaisuuden lisäksi mitannut sen tärkeyttä ja pyytänyt asiakkaalta mielipiteitä kysyttävästä ominaisuudesta.

Westfall (2002) korostaa, että ominaisuuden täyttymisen ohella, tulee mitata, kuinka tärkeä ominaisuus on. Jos asiakas esimerkiksi on vastannut, että muutospyyntöjen korjaukseen kuluu paljon aikaa, niin useimmiten yritys pyrkii mahdollisuuksien mukaan kehittämään tätä ominaisuutta. Toisaalta, jos asiakas on ilmoittanut kyselyssä, että ominaisuuden täyttyminen ei ole niin tärkeää, vaan pitää esimerkiksi muutospyynnön laatua tärkeämpänä ominaisuutena, voi yritys keskittää resurssinsa laadun kehittämiseen.

Jos asiakastyytyväisyyskyselyssä käytetään portaittaista asteikkoa, tulisi tämän lisäksi kerätä asiakkaiden mielipiteitä vapaavalintaisessa kohdassa, jotta saataisiin enemmän yksityiskohtaista tietoa. Jotta asiakkaan mielipiteet olisi helpompi tulkita, tulisi ne pyytää oikessa asiayhteydessä eli jokaisen kysymyksen jälkeen eikä esimerkiksi vasta kyselyn lopussa.

Yrityksen asiakastyytyväisyyskysely kattoi osaksi ylläpidon kehittämisen kysymyksiä. Raportti numero 1 kattoi 37 kysymystä, joista kolme kysymystä olivat tarkoitettu koskemaan ohjelmistoa ja sen tukipalvelun laatua. Mielestäni kaksi näistä kysymyksistä liittyi käytettävyyteen ja

yksi käsitteli yleisellä tasolla ylläpitoa. Vaikka yritys ei ole mitannut pääsääntöisesti ylläpitoa, voidaan kahdeksaa kysymystä käyttää ylläpidon mittaamiseen.

Mielestäni kysymyksiä voitaisiin tarkentaa osittain hienojakoisemmiksi, jotta niiden avulla saataisiin tarkempaa tietoa, sillä asiakkaat eivät välttämättä kerro kysyttävän asian mahdollisista ongelmakohdista ellei niitä erikseen kysytä.

Yrityksen asiakastyytyväisyyskysely oli tehty monelle eri kohderyhmälle, jolloin tulokset saattavat vääristyä. Esimerkiksi lopputuotteen käyttäjät saattavat olla hyvin tyytyväisiä dokumentaatioon, mutta asentajat eivät, jos dokumentaatiota ei ole suunniteltu heidän tarpeisiinsa (Westfall, 2002). Toisaalta tuloksista voidaan nähdä eri kohderyhmien eroavaisuudet, mutta tämän tulkinta ei ole aina itsestään selvää. Täten yritys voisikin suunnitella oman asiakastyytyväisyyskyselyn ylläpitoon kuuluville kohderyhmille.

Tärkeimpänä ominaisuutena yrityksen asiakastyytyväisyydessä pidän tuotekohtaisia kyselyitä. Mittausaineistossa oli hyvin paljon variaatiota tuotteiden X ja Y kohdalla muun muassa kehitysten aikaisten virheiden korjausprosentin ja asiakkaan ilmoittaminen virheiden välillä. Täten tuotekohtaisten kyselyjen avulla voitaisiin saada yksityiskohtaisempaa tietoa jokaisesta tuotteesta, jolloin prosessin parantaminen voidaan keskittää oikeaan kohtaan.

Taulukkoon 14 on koottu yhteenveto uusista sekä tarkennetuista kysymyksistä, jotka ovat suunniteltu tuotteen loppukäyttäjälle. Ensimmäinen sarake kattaa kysymyksen, jonka jälkeen on tyytyväisyyssarake, jossa kartoitetaan, kuinka tyytyväinen asiakas on kysymyksessä esitettyyn ominaisuuteen. Sen jälkeen asiakkaan tulisi merkitä tärkeyssarakkeeseen, kuinka tärkeä kysymyksen ominaisuuden täyttyminen on hänelle. Mitä tyytyväisempi asiakas on ominaisuuteen, sitä paremman numeroarvon hän voi sille antaa. Numeroasteikon luokittelussa voidaan käyttää esimerkiksi kuvassa 4 esiintyvää asteikkoa.

Kysymyksissä kolme ja neljä mainitaan termit virhe ja häiriö, jotka Fenton ja Pfleeger (1997) määrittelevät seuraavasti: *Virhe* (error) on ihmisen aiheuttama toiminto, joka ilmenee ohjelmistossa *vikana* (fault) eli johtaa esimerkiksi ohjelmiston toiminnan epäonnistumiseen. *Häiriö* (fai-

lure) puolestaan syntyy, kun vika kohdataan. Tällöin ohjelmistomoduulilta voi loppua esimerkiksi suorituskyky vaaditun toiminnon toteuttamisessa. Häiriön vakavimmasta tilasta käytetään termiä *kaatuminen*, jolloin ohjelmisto lakkaa kokonaan toimimasta.

Westfall (2002) on huomannut, että kyselyjen tulokset saattavat heiketä, vaikka jatkuvaa parantumista tapahtuisikin. Useasti asiakkaiden vaatimukset kasvavat, kun tuotteet, palvelut ja prosessit paranevat. Siksi asiakkaan kasvavat odotukset tulisi myös huomioida kyselyjen tuloksissa.

Jotta asiakastyytyväisyyskyselyjen vastauksista olisi enemmän hyötyä, tulisi yrityksen mitata ainakin osittain kysyttyjä asioita, mikäli ne ovat tärkeitä asiakkaalle. Asiakastyytyväisyyskyselyssä pyrittiin muun muassa selvittämään sopivaa muutospyynnön korjaukseen kuluvaa aikaa. Jotta yritys tietäisi, pitääkö korjausnopeutta tehostaa, täytyy sitä mitata.

Mittari 3: Muutospyynnön korjausnopeus

Yrityksellä on jo olemassa valmiiksi tarvittavat tiedot muutospyynnön korjausnopeuden mittaamiseen. Basilin & al. (1994b) mukaan juuri tämänkaltaisen tietoa tulisi maksimoida, sillä mittaaminen on luotettavaa ja se ei vaadi kohtuutonta ponnistelua, vaan se voidaan aloittaa heti.

Muutospyynnön keskimääräinen korjausnopeus voidaan laskea seuraavalla kaavalla (ISO/IEC, 2002):

$$kn = \frac{\sum t}{n}, missä$$

$kn = muutospyynnön\ keskimääräinen\ korjausnopeus$

$\sum = kaikkien\ muutospyyntöjen\ korjaukseen\ kuluneiden\ aikojen\ yhteenlaskettu\ summa$

$t = muutospyynnön\ korjaukseen\ kulunut\ aika$

$n = asiakkaan\ ilmoittaminen\ virheellisten\ muutospyyntöjen\ lukumäärä$

Mitä pienempi on saatu tulos, sitä parempi korjausnopeus on. Muutospyynnön korjausnopeus on suhteellinen mittari, jonka antamaa tulosta voidaan arvioida tarkemmin, kun tiedetään, mikä on asiakkaan oletus muutospyynnön kestosta.

Tutkimuksen aikana mittausaineisto antoi viitteen siihen, että korjausten laadusta on saatettu tinkiä, sillä tuotteen Y virheistä oli korjattu vuoden 2005 aikana 96,2 %, mutta asiakkaiden ilmoittamia ohjelmavirheitä oli suhteellisen paljon korjausprosenttiin nähden. Kuten jo aiemmin mainitsin, asiakkaiden ilmoittamien ohjelmavirheiden lukumäärä voi johtua monesta tekijästä. Tuotteen Y virheiden lukumäärä on voinut nousta suhteessa korkeammaksi, jos tuotetta on käytetty vuonna 2005 enemmän kuin aikaisemmin. Lisäksi tuotteen X kohdalla muutospyynnöistä oli jätetty korjaamatta 73,7 %, vaikka asiakkaat olivat valittaneet ohjelmistovirheistä 115 kertaa. Tämän takia olisikin hyvä, jos yritys keskittyisi jatkossa mittaamaan virheellisten korjauksien lukumäärää, testauksen laatua ja tarpeettomia muutospyyntöjä. Näiden tietojen avulla yritys voisi saada enemmän tietoa korjauksen, testauksen ja muutospyyntöjen laadusta.

Mittari 4: Virheellisten korjauksien lukumäärä

Virheellisten korjauksien lukumäärä on Kanin (2003) mukaan tärkeä laatumittari ohjelmiston ylläpitovaiheessa, koska virheillä on suora vaikutus asiakastyytyväisyyteen. Virheellisellä korjauksella tarkoitetaan sitä, että korjaus ei korjaa alkuperäistä virhettä tai virhe aiheuttaa muita virheitä. Virheellisen korjauksien lukumäärä voidaan laskea seuraavalla kaavalla:

$$n = \frac{f}{t} * 100\%$$

$n = virheellisten\, korjauksien\, lukumäärä,\ f = virheellisesti\, korjatut\, muutospyynnöt$
$t = kaikki\, korjatut\, muutospyynnöt$

Kanin (2003) mukaan virheellisten korjauksien määrän pitäisi olla suuntaa näyttävä arvo, koska pieni virheellinen korjausprosentti voi antaa liian positiivisen kuvan korjauksien luotettavuudesta, jos jäljelle jääneet virheet ovat vakavia tai virheiden korjausten määrä on suuri. Tätä

mittaria käytettäessä tulisi myös muistaa, että virheiden löytyminen saattaa kestää kauan, joten mittari ei välttämättä näytä kovinkaan nopeasti todellista tulosta.

Mikäli virheellisiä korjauksia esiintyy paljon, tulisi korjauksen nopeutta ja testauksen laatua tarkkailla ja selvittää, johtuuko virheellisten korjausten määrä liian nopeasta korjaustahdista tai puutteellisesta testausprosessista.

Mittari 5: Testauksen edistymisen ja laadun seuranta

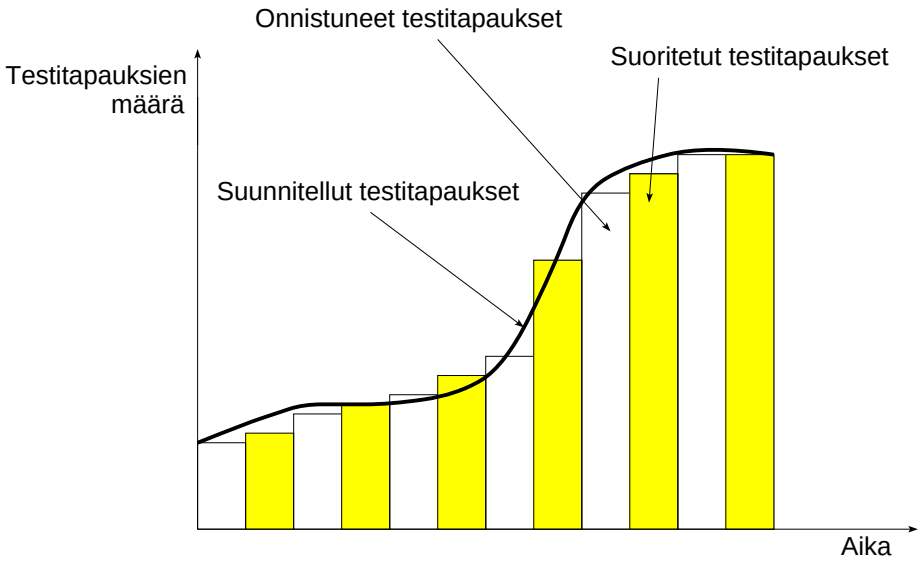
Yrityksen tulisi erityisesti panostaa hyväksymistestaukseen, sillä se toteuttaa ohjelmistot ja niiden testauksen alihankkijoillansa. Humphreyn (1989) mukaan hyväksymistestaus tulisi suorittaa ohjelmiston lopullisessa ympäristössä. Mikäli tämä ei ole mahdollista, tulisi suorittaa simulointi, joka jäljittelee mahdollisimman tarkasti lopullista ohjelmiston ympäristöä.

Koska yritys laatii itse ohjelmistovaatimukset, tulisi jokaisen vaatimusten täyttymiseen kiinnittää huomioita eri näkökulmista. Humphrey (1989) suosittelee käyttämään vaatimusten testaamiseen testattavien vaatimusten tarkastustaulukkoa (test requirements verification martix). Matriisiin listataan kaikki vaatimukset ja niiden testitapaukset, jotka osoittavat vaatimuksien täyttymisen. Testitapausten tulisi kattaa tietosisältö- ja toiminnallisten vaatimusten lisäksi yleiset, yhteensopivuus-, turvallisuus-, ulkoiset liittymä-, resurssi-, luotettavuus-, käytettävyys-, tehokkuus-, ylläpidettävyys- ja siirrettävyysvaatimukset.

Zuze (1998) painottaa, että testattaessa tulisi käydä kaikki mahdolliset vaihtoehdot lävitse, jolloin eri kombinaatioiden määrä saattaa kasvaa suureksi etenkin laajoissa ohjelmistoissa. Tällöin hyväksymistestauksen edistymistä tulisi seurata siihen soveltuvalla mittarilla. Kan (2003) ja Westfall (2004) ehdottavat testitapausten suunnittelun ja toteutumisen mittaamiseen *S-käyrää* (S-curve). Mittari on Kanin (2003) mukaan tärkein mittari testitapausten edistymisen sekä laadun seurannassa. Tämän lisäksi se soveltuu kaikkiin testausvaiheisiin.

Mittari on saanut nimensä graafisen kuvaajansa perusteella, sillä useasti se muistuttaa s-kirjainta tiedon kumulatiivisen käyttäytymisen takia. X-akselille asetetaan aikayksikkö ja Y-akseli ker-

too suunniteltujen, suoritettujen ja onnistuneiden testitapausten määrän. Mittarin avulla saadaan selville, kuinka monta testitapausta on suunniteltu aikajaksoa kohden. Lisäksi aikajaksolta nähdään, onko kaikki suunnitellut testitapaukset suoritettu ja kuinka moni testitapaus on onnistunut. Kuvasta 21 on nähtävissä S-käyrän idea.



Kuva 21: S-käyrä kertoo suunniteltujen, suoritettujen ja onnistuneiden testitapausten määrän aikajaksoa kohden.

Kanin (2003) mukaan S-käyrän avulla on helppo seurata testitapausten edistymistä vertaamalla suunniteltua ja suoritettua käyrää toisiinsa. Lisäksi mittarin avulla voidaan nähdä mahdolliset testauksessa löytyneet virheet, jos suoritettujen ja onnistuneiden testitapausten käyrät poikkeavat toisistansa. Mikäli käyrän jyrkkyydessä on 15 %:n ero, tulisi toimenpiteisiin ryhtyä, jotta ongelmakohtiin ehditään puuttua riittävän ajoissa. Mittarin vahvuutena voidaan pitää myös sitä, että testitapaukset suunnitellaan huolella etukäteen, jotta suunniteltu testikäyrä voidaan asettaa kaavioon etukäteen. Testitapausten suunnitteleminen ei ole välttämättä kovinkaan helppoa, sillä realistisen suunnitelman tekeminen on haastavaa. Etenkin, jos testitapaukset pitää johtaa keskeneräisestä lähdekielisestä ohjelmistosta. Toisaalta hyväksymistestauksessa testitapausten suunnittelun pitäisi olla helpompaa, sillä testitapaukset voidaan johtaa vaatimusmäärittelystä, jolloin testitapausten määrä tiedetään etukäteen.

Vaikka kaaviosta voidaan nähdä mittarin käyttäytyminen, suosittelee Kan (2003) myös taulukon ylläpitämistä. Taulukkoon tulisi merkitä S-käyrän tulokset, koska siitä voidaan nähdä tarkemmin testauksessa esiintyvät poikkeamat kuin silmämääräisesti katsottuna. Lisäksi hän muistuttaa, että suunniteltua testikäyrää tulisi ylläpitää, jotta se vastaisi mahdollisimman paljon todellista testausta. Mittaria tulisi myös käyttää useammalla eri tasolla, sillä kokonaisuudessaan mittaaminen saattaa näyttää hyvältä, jos jokin osa on aikataulua edellä. Tällöin se saattaa kompensoida jälkeenjääneitä testausosioita. Siksi olisikin hyvä, jos testausprosessi pilkottaisiin pienempiin kokonaisuuksiin, jotta mahdolliset jälkeenjääneet osiot havaittaisiin tarpeeksi ajoissa.

S-käyrä on oikeinkäytettynä hyödyllinen mittari, josta voidaan seurata testausprosessin etenemistä ja laatua. Sitä on käytetty sekä yksittäisille että lisääville tuotekohtaisille (release-to-release) projekteille. Mittarin avulla saadaan myös kerättyä hyödyllistä historiatietoa samankaltaisista testausprosesseista aikataulun ja ongelmakohtien osalta. Tällöin tulisi muistaa, että jokaiselle testausprosessille käytetään samaa aikayksikköä, kuten viikkoa, jotta mittaustulokset olisivat jatkossa suoraan vertailukelpoisia. Ennen mittarin käyttöönottoa tulisi muistaa, että sen todellinen hyöty saadaan esiin vasta satojen tai tuhansien testitapausten avulla, joten ihan pienimpiin testausprosesseihin sitä ei välttämättä kannata käyttää.

Mittari 6: Tarpeettomat muutospyynnöt

Mittausaineistoa tutkiessani kiinnitin huomiota tarpeettomien muutospyyntöjen määrään. Yrityksen periaatteisiin kuuluu, että kaikki muutospyynnöt käydään lävitse, jolloin valitaan korjattavat muutospyynnöt. Useasti kaikkia muutospyyntöjä ei ole järkevää korjata, sillä esimerkiksi kosmeettiset muutospyynnöt jätetään useasti korjaamatta, jotta tärkeimmille muutospyynnöille jäisi enemmän resursseja. Toisaalta jos vuoden aikana esiintyneistä muutospyynnöistä jätetään korjaamatta yli 70 %, kuten vuonna 2004 tehtiin tuotteen X kohdalla, voitaisiin kiinnittää tarkempaa huomiota muutospyyntöjen laatuun.

Korjaamattomien muutospyyntöjen osalta tulisi tarkastaa, ovatko ne olleet asiallisia, jolloin oh-

jelmiston mittaajien tulisi selvittää, johtuvatko asiakkaan ilmoittamat ohjelmavirheet korjaamattomista muutospyynnöistä. Mikäli ohjelmavirheet johtuvat korjaamattomista muutospyynnöistä, tulisi ne korjata välittömästi. Toisaalta jos ohjelmavirheet johtuvat korjatuista muutospyynnöistä, tulisi ohjelmiston laatuun kiinnittää parempaa huomiota esimerkiksi testaamisen avulla.

Korjaamattomat muutospyynnöt voivat myös olla epäselkeästi täytettyjä, jolloin muutosta ei voida suorittaa puuttuvan tiedon takia. Lisäksi ne voivat olla turhia, jolloin esimerkiksi asiakas ei ole osannut käyttää ohjelmistoa oikein ja asiakkaan virheestä johtuva käytös on korjattu vahingossa muutospyynnöksi. Näitä muutospyyntöjä voidaan mitata seuraavasti (Oinas, 2000):

$$tarpeettomat\,muutospyynnöt = \frac{toteutumattomat\,muutospyynnöt}{kaikki\,käsitellyt\,muutospyynnöt}, missä$$

$toteutumattomat\,muutospyynnöt = kaikki\,muutospyynnöt,\ jotka\,eivät\,ole\,aiheuttaneet\,korjaavaa\,toimintaa$

Oinaksen (2000) mukaan mittarin avulla voidaan tunnistaa lisäkoulutuksen tarve, jos tarpeettomia muutospyyntöjä esiintyy runsaasti. Tällöin muutospyynnön kerääjät tulisi kouluttaa paremmin, jolloin vältyttäisiin tarpeettomilta muutospyynnöiltä. Pahimmassa tapauksessa vakava virhe saattaa jäädä huomaamatta, jos muutospyyntö on kirjattu virheellisesti eikä sen avulla voida tunnistaa virheen aiheuttajaa.

Mittarin osoittajana ovat muutospyynnöt, jotka eivät ole aiheuttaneet korjaavaa toimintaa. Mielestäni tämä rajaus ei erottele tarpeettomia muutospyyntöjä esimerkiksi kosmeettisista muutospyynnöistä, sillä ne saatetaan korjata myöhemmin aikataulun salliessa. Siksi osoittajaa voitaisiin tarkentaa siten, että tarpeettomiksi muutospyynnöiksi laskettaisiin vain ne muutospyynnöt, joita ei tulla koskaan korjaamaan esimerkiksi puutteellisen tai väärinkirjatun tiedon takia. Tällöin koulutuksen todellinen tarve olisi paremmin nähtävissä. Koulutuksen todellinen hyöty tulee esille, kun muutospyyntöjä on useita satoja, jolloin turhista muutospyynnöistä voi koostua todellinen ongelma, sillä niiden selvittämiseen voi kulua paljon aikaa.

Tässä kohdin esiteltiin kuusi mittaria yrityksen ylläpidon kehittämiseen. Mittarit voidaan luokitella CMMI:n tasolle kaksi, sillä niiden avulla on tarkoitus arvioida jälkikäteen, kuinka mittauksen kohteena oleva asia on onnistunut (Kan,2003; SEI, 2006). Testauksessa käytettävä S-käyrä voidaan mielestäni luokitella CMMI:n tasolle kolme, sillä sen tarkoituksena on ohjata hyväksymistestauksen etenemistä. Pidemmällä aikavälillä S-käyrä saattaa toimia myös tason neljä mittarina, sillä sen avulla voidaan ennustaa vastaavanlaisien ohjelmistoprosessien hyväksymistestauksen kestoa ja mahdollisia ongelmakohtia. Toisaalta vastaavanlaisen historiatiedon kerääminen voi viedä hyvinkin paljon aikaa, sillä, juuri samantasoisia ohjelmistoprosesseja kehitetään harvoin, joten ennustettavuus voi olla hyvinkin epätarkkaa.

Mittaaminen on haastava prosessi, sillä jokaisen yrityksen täytyy luoda oma mittarikonsepti omiin erityistarpeisiinsa eikä vastaavanlaista mallia ole muualta saatavissa. Entisestään tätä prosessia vaikeuttaa se, että kirjallisuuden avulla saatavat mittaustiedot ovat melko suppeita ja mittareiden välinen syvällisempi vertailu jää usein mitättömäksi. Lisäksi ohjelmistotuotannossa ei ole vielä pystytty määrittelemään riittävällä tasolla, minkälaisia mittareita kullekin attribuutille tulisi käyttää. Tämä lisää mittareiden valinnan monimuotoisuutta. Siksi on tärkeää, että jokaisen käytettävän mittarin rajoitukset tunnetaan riittävän hyvin, jotta niihin voitaisiin varautua etukäteen. Jokaisen tulisi käyttää myös omaa intuitiotaan mittareiden valinnassa, sillä yhtä ja ainutta oikeaa ratkaisua tuskin on olemassa (Briand & al., 2002).

Taulukko 14: Tuotteen loppukäyttäjälle suunnitellut uudet, tarkennetut kysymykset.

Kysymys	Tyytyväisyys	Tärkeys
1. Onnistuuko tuotteen asentaminen vaivattomasti?	1 ⌊ ⌋ 5	1 ⌊ ⌋ 5
Lisätietoja:		
2. Onko asennusohje taydellinen ja virheetön?	1 ⌊ ⌋ 5	1 ⌊ ⌋ 5
Lisätietoja:		
3. Esiintyykö tuotteessa heti muutospyynnön toimituksen jälkeen virheitä tai häiriöitä?	1 ⌊ ⌋ 5	1 ⌊ ⌋ 5
Lisätietoja:		
4. Esiintyykö tuotteessa pitkällä aikavälillä virheitä tai häiriöitä?	1 ⌊ ⌋ 5	1 ⌊ ⌋ 5
Lisätietoja:		
5. Esiintyykö tuotteessa heti muutospyynnön toimituksen jälkeen kaatumisia?	1 ⌊ ⌋ 5	1 ⌊ ⌋ 5
Lisätietoja:		
6. Esiintyykö tuotteessa pitkällä aikavälillä kaatumisia?	1 ⌊ ⌋ 5	1 ⌊ ⌋ 5
Lisätietoja:		
7. Voiko asiakas käyttää tuotetta virheen tai häiriön ilmaantuessa?	1 ⌊ ⌋ 5	1 ⌊ ⌋ 5
Lisätietoja:		
8. Ratkaistaanko muutospyyntö riittävän nopeasti?	1 ⌊ ⌋ 5	1 ⌊ ⌋ 5
Kerro oletuksesi muutospyynnön kestosta:		
9. Vastaako korjaus muutospyynnössä esitettyä asiaa?	1 ⌊ ⌋ 5	1 ⌊ ⌋ 5
Lisätietoja:		
10. Osaako tekninen tuki ratkaista asiakkaan ongelman?	1 ⌊ ⌋ 5	1 ⌊ ⌋ 5
Lisätietoja:		
11. Onko teknisen tuen henkilökunta palvelualtista?	1 ⌊ ⌋ 5	1 ⌊ ⌋ 5
Lisätietoja:		

7 YHTEENVETO

Pyrin tutkielmassani sovittamaan tavoitekeskeisen mittaamisen teoriaa ja käytäntöä toisiinsa. Teoriaosuudessa käsittelin ohjelmistoprosessin parantamisen lisäksi GQM-menetelmän suunnittelu-, määrittely-, tiedonkeruu- ja tulkintavaiheita. Käytännön osuudessa GQM-menetelmää sovellettiin ohjelmiston ylläpitoon, jossa liiketoiminnan tavoitteena oli asiakastyytyväisyyden parantaminen. Liiketoiminnan tavoitteet johdettiin kirjallisuudessa esitettyjen esimerkkien avulla ensiksi liiketoiminnan kysymyksiksi, joista ryhmiteltiin alitavoite, oliot, attribuutit ja lopuksi mittaamisen tavoite, kysymykset ja mittarit.

Lisäksi tutkielmassa käsittelin suomalaisen mekaanisia ja sähkömekaanisia tuotteita valmistavan yrityksen mittausaineistoa. Mittausaineisto koostui pääasiallisesti asiakkaan ilmoittamista virheistä. Mittausaineiston pohjalta johdin GQM- menetelmää apuna käyttäen uusia mittareita yrityksen käyttöön. Mittareiden avulla on tarkoitus saada enemmän tietoa esimerkiksi ylläpitoon lähetetyistä muutospyynnöistä, asiakkaiden mielipiteitä ylläpitoprosessista ja tietoa testauksen laadusta. Pääasiallisesti mittareiden avulla saatavan tiedon tarkoitus on auttaa ymmärtämään, toimiiko yritys oikein ja voisiko se kehittää ylläpitoprosessia tehokkaammaksi. Mittarit voidaan pääasiallisesti luokitella CMMI:n tasolle 2, mutta testauksessa käytettävän mittarin on tarkoitus toimia CMMI:n tasolla 3 ja tulevaisuudessa mahdollisesti myös tasolla 4. Mittareiden lisäksi täydensin yrityksen virheluokittelua, jonka keskeisemmäksi ongelmaksi nousi virheen aiheuttajan uudelleen jäljittäminen. Mielestäni uudistetun luokittelun käyttöönotto olisi yrityksen kannalta merkittävä, sillä useasti suurin osa ylläpitoprosessin ajasta kuluu ongelmakohdan etsimiseen. Mikäli luokittelu ja sen käyttö saadaan tehokkaaksi, uskon, että se nopeuttaa ohjelmiston virheiden paikantamista.

Yritykselle johdettujen mittareiden toimivuudesta ei vielä voida vakuuttua, sillä GQM-menetelmää ei voida pitää täydellisenä, koska sen toteutus ei ollut niin laaja, kuin oikeassa mittausprosessissa tulisi olla. Mittareiden tavoitteet johdettiin yrityksen aikaisemman mittausaineiston ja yhteyshenkilön tietojen perusteella. Yrityksen toimintatapojen ja mitattavien prosessien

sekä prosessimallien tuntemuksen ja ylläpitäjille kohdistettujen haastattelujen avulla GQM-menetelmää olisi voinut soveltaa syvällisemmin. Tässä tutkielmassa GQM-menetelmä rajottui määrittelyvaiheeseen, josta jätettiin toteuttamatta GQM-, mittaus- ja analysointisuunnitelmat, sillä mittareiden on tässä vaiheessa tarkoitus toimia ehdotuksena eikä niiden lopullisesta käytöstä ole saatu yritykseltä varmuutta. Mikäli yritys aikoo käyttää sille suunniteltuja mittareita, tulisi mittausprosessia laajentaa toteuttamalla kyseiset dokumentaatiot. Mielestäni yritys voisi jatkaa mittausprosessia GQM-menetelmän mukaisesti, sillä se tarjoaa selkeän mallin prosessin loppuunsaattamiseksi. GQM-menetelmän tärkeimpiin osa-alueisiin kuuluu palautetilaisuuksien pitäminen, joissa projektin jäsenet keskustelevat mittaustuloksista ja mahdollisesti keksivät uusia tavoitteita, kysymyksiä ja mittareita tai karsivat vanhoja hyödyttömäksi havaittuja mittauskohteita. Ennen kaikkea palautetilaisuuksien tärkeimpänä tehtävänä on huomata uusia parannuskohteita.

Tämän lisäksi tutkielmassa käsiteltiin alustavaa kustannusmallia, joka mittausprosessilta kuluu GQM-menetelmää käyttämiseen. Useasti kustannukset nousevat suuriksi ja niiden arvioiminen on yksinkertaisempaa kuin mittausohjelmasta aiheutuvan hyödyn määrittäminen. Siksi mittaamisen tavoitteiden tulisikin aina perustua liiketoiminnan tavoitteisiin, jotta ylimmän johdon tuki olisi helpommin saatavissa mittausprosessille. Kirjallisuudessa esitettävät mallit keskittyvät usein ylimmän johdon tuen saamiseen, mutta mielestäni keskijohdon vakuuttaminen on vähintäänkin yhtä tärkeää, sillä heidän alaisensa suorittavat mittaamisen. Mikäli keskijohto ei ole sitoutunut mittaamisprosessiin, ei voida olettaa, että mittaustulokset olisivat luotettavia. Mielestäni mittausprosessissa tulisi erityisesti kiinnittää huomiota henkilöihin, jotka keräävät mittaustietoa. Heille tulisi kertoa, kuinka mittausprosessi tulee vaikuttamaan heidän työskentelymenetelmiin ja ettei mittaamisen avulla ei ole tarkoitus arvioida henkilöiden työtaitoja. Lisäksi mittausprosessi tulisi sisällyttää varsinaiseen prosessin kulkuun, jotta työntekijät eivät kokisi sitä lisätehtävänä.

Mielestäni mitään kirjallisuudessa esitettyä mallia, olipa se sitten TQM, CMMI, SPICE tai GQM, ei tulisi käyttää liian kirjaimellisesti hyväksi, sillä mallit tarjoavat vain teoreettisen nä-

kökulman asioiden tarkasteluun. Lisäksi malleja tulisi myös yhdistää keskenään, jotta niitä voitaisiin hyödyntää parhaalla mahdollisella tavalla.

Uskon, että yritykselle suunnitellut mittarit olisi voinut toteuttaa myös ilman GQM-menetelmää, mutta hyödyin menetelmän tavasta jalostaa tavoitteet kysymyksiksi ja lopulta mittareiksi. Mallit antavat käyttäjälleen vapauden toteuttaa omanlaisensa prosessin, mutta sama asia voi koitua usein ongelmaksi. Huomasin, että kirjallisuus tarjosi vähän käytännönläheistä tietoa GQM-menetelmästä ja se koitui tutkielmassa useasti ongelmalliseksi. Toivon, että tutkielmani auttaa osaltaan ihmisiä, jotka etsivät käytännönläheisempää kuvaa GQM-menetelmän soveltamisesta.

VIITTEET

Aaen, I. (2003) Software Process Improvement: Blueprints versus recipes. *IEEE Software* **20**(5), 86-93.

Arthur, L. J. (1993) *Improving software quality: an insider's quide to TQM.* John Wiley & Sons, New York.

Baldassarre, M. T., Caivano, D., Visaggio, G. (2003) Comprehensibility and Efficiency of Multiview Framework for Measurement Plan Design. *Proceedings of the 2003 International Symposium on Empirical Software Engineering (ISESE 2003)* (toim. Amschler Andrews, A. K.), IEEE Computer Society, Washington DC, 89-98.

Basili, V. R. (1992) *Software Modeling and Measurement: The Goal/Question/Metric Paradigm.* WWW-sivusto, `http://www.cs.umd.edu/~basili/publications/technical/T75.pdf` (18.8.2006).

Basili, V. R., Caldiera, G., Rombach, H. D. (1994a) *The Experience Factory.* WWW-sivusto, `http://www.cs.umd.edu/users/basili/publications/technical/T85.pdf` (15.3.2006).

Basili, V. R., Caldiera, G., Rombach, H. D. (1994b) *Goal Question Metric Approach.* WWW-sivusto, `http://www.cs.umd.edu/users/basili/publications/technical/T86.pdf` (15.3.2006).

Basili, V. R., Rombach, H. D. (1988) The TAME Project: Towards Improvement-Oriented Software Environments. *IEEE Transaction on Software Engineering* **4**(6), 758-773.

Basili, V. R., Weiss, D., M. (1984) A Methodology for Collecting Valid Software Engineering Data. *IEEE Transaction on Software Engineering* **SE-10**(6), 728-738.

Birk, A., van Solingen, R., Järvinen, J. (1998) Business Impact, Benefit, and Cost of Applying GQM in Industry: An In-Depth, Long-Term Investigation an Schlumberger RPS. *Proceedings*

Fifth International Software Metrics Symposium (Metrics 1998) (toim. Kelly, K.), IEEE Computer Society, Bethesda, 93-96.

Briand, L. C., Differding, C. M., Rombach, D. H. (1996) Practical Guidelines for Measurement-Based Process Improvement. *Software Process Improvement and Practice* **2**(4), 253-280.

Briand, L. C., Morasca, S., Basili, V. R. (2002) An Operational Process for Goal-Driven Definition of Measures. *IEEE Transactions on Software Engineering.* **28**(12), 1106-1125.

Buckley, M., Chillarege, R. (1995) Discovering relationships between service and customer satisfaction. *Proceedings of International Conference on Software Maintenance (ICSM 1995),* IEEE Computer Society, Los Alamitos, 192-201.

Cattaneo, F., Fuggetta A., Sciuto, D. (2001) Pursuing Coherence in Software Process Assesment and Improvement. *Software Process Improvement and Practice* **6**(1), 3-22.

Choudhury, S. (1988) *Project Management.* Tata McGraw-Hill, New Delhi.

Chrissis, M. B., Konrad, M., Shrum, S (2003) *Guidelines for Process Integration and Product Improvement.* Pearson Education, Boston.

Conradi, R., Fuggetta, A. (2002) Improving Software Process Improvement. *IEEE Software* **19**(4), 92-99.

Crosby, P. B. (1979) *Quality Is Free.* McGraw-Hill, New York.

Deming, W. E. (1986) *Out of the Crisis.* MIT Center for Advanced Engineering Study, Cambridge, Mass.

Differding, C., Hoisl, B., Lott, C. M., (1996) *Technology Package for the Goal Question Metric Paradigm.* WWW-sivusto, `http://www.chris-lott.org/work/pubs/1996-gqm-tp.pdf` (1.8.2006).

Ebert, C., Dumke, R. Bundschuh, M., Schmietendorf, A. (2005) *Best Practices in Software Measurement: How to use metrics to improve project and process performance.* Springer-Verlag

Berlin Heidelberg, Berlin.

El Emam, K., Drouin J-N., Melo, W. (1998) *SPICE : The Theory and Practice of Software Process Improvement and Capability Determination.* IEEE Computer Society, Los Alamitos.

Fenton, N. E., Pfleeger, S. F. (1997) *Software Metrics: A Rigorous & Practical Approach.* PWS Publishing Company, Boston.

Florac, W. A., Charleton, A. D. (1999) *Measuring the Software Process: Statistical Process Control for Software Process Improvement.* Addison-Wesley, Reading (Mass.).

Fuggetta, A. (2000) Software Process: A Roadmap. *Proceedings of the Conference on The Future of Software Engineering* (toim. Finkelstein, A.), Limerick, Ireland, ACM Press, New York, 25-34.

Fuggetta, A., Lavazza, L., Morasca, S., Cinti, S., Olando, G., Orazi, E. (1998) Applying GQM in an Industrial Software Factory. *ACM Transactions on Software Engineering and methodology (TOSEM)* **7**(4), 411-448.

Garcia, S. (1997) Evolving Improvement Paradigms: Capability Maturity Models and ISO/IEC 15504 (PDTR). *Software Process Improvement and Practice* **3**(1), 47-58.

Goethert, W., Siviy, J. (2004) *Applications of the Indicator Template for Measurement and Analysis.* WWW-sivusto, `http://www.sei.cmu.edu/pub/documents/04.reports/pdf/04tn024.pdf` (25.4.2006).

Goldpractices (2005) *Goal-Question-Metric Approach.* WWW-sivusto, `http://www.goldpractices.com/practices/gqm/index.php` (11.4.2006).

Guaspari, J. (1985) *I Know It When I See It: A Modern Fable About Quality.* American Management Association, New York.

Haikala, I., Märijärvi, J. (2000) *Ohjelmistotuotanto.* Satku - Kauppakaari, Pieksämäki.

Humphrey, W. S. (1989) *Managing the Software Process.* Addison-Wesley, Reading (Mass.).

IEEE (1998) *IEEE Standard for Software Maintenance.* IEEE Std 1219-1998, The Institute of Electrical and Electronics Engineerings, New York.

ISO/IEC (2001) *Software engineering - Product quality - Part 1: Quality model.* ISO/IEC 9126-1:2001, International Organisation for Standardisation, Switzerland.

ISO/IEC (2002) *Software Engineering - Product quality -Part 2: External metrics.* TR 9126-2:2002, International Organisation for Standardisation, Switzerland.

ISO/IEC (2005) *Information Technology - Process Assessment - Part 5: An exemplar Process Assessment Model.* ISO/IEC 15504-5:2005, International Organisation for Standardisation, Switzerland.

Kan, S. H. (2003) *Metrics and Models in Software Quality Engineering.* Addison-Wesley, Boston.

Kearney, J. K., Sedlmeyer, R. L., Thompson W. B, Gray, M. A., Adler, M. A (1986) Software complexity measurement. *Communication of ACM* **29**(11), 1044-1050.

Lavazza, L. (2000) Providing Automated Support for the GQM Measurement Process. *IEEE Software* **17**(3), 56-62.

Lavazza, L., Barresi, G. (2005) Automated Support for Process-aware Definition and Execution of Measurement Plans. *Proceedings of the 27th international conference on Software engineering 2005 (ICSE 2005)* (toim. Roman, G-C.), AMC Press, New York, 234-243.

Leffingwell, D., Widrig, D. (2003) *Managing Software Requirements: A Use Case Approach.* Addison-Wesley, Boston.

Mendonça, M. G., Basili, V. R. (2000) Validation of an Approach for Improving Existing Measurement Frameworks. *IEEE Transactions on Software Engineering* **26**(6), 484-499.

Oinas, A. (2000) Defining goal-driven fault management metrics in a real world environment: a case-study from Nokia. *Proceedings of the Fourth European Conference on Software Main-*

tenance and Reengineering (CSMR'00) (toim. Ebert, J., Verhoef, C.), IEEE Computer Society, Washington, 101-108.

Olsson, T., Runeson, P. (2001) V-GQM: A Feed-Back Approach to Validation of a GQM Study. *Proceedings, Seventh International Software Metrics Symposium (METRICS 2001)* (toim. Rombach, D.), IEEE Computer Society, Washington, 236-245.

Pall, G. A. (1987) *Quality Process Management.* Prentice Hall, Englewood Cliffs.

Park, R. E., Goethert, W. B., Florac, W. A. (1996) *Goal-Driven Software Measurement: A Guidebook.* WWW-sivusto, `http://www.sei.cmu.edu/pub/documents/96.reports/pdf/hb002.96.pdf` (6.3.2006).

Paulk, M. C., Curtis, B., Chrissis, M. B., Weber, C. V. (1993) *Capability Maturity Model for Software, Version 1.1.* WWW-sivusto, `http://www.sei.cmu.edu/pub/documents/93.reports/pdf/tr24.93.pdf` (27.1.2006).

Pigoski, T. M. (1997) *Practical Software Maintenance: Best Practices for Managing Your Software Investment.* John Wiley & Sons Inc.

Polo, M., Piattini, M., Ruiz, F. (2003) *Advances in Software Maintenance Management: Technologies and Solutions.* Idea Group Publishing, Hersley and London.

Pressman, R.S. (2000) *Software Engineering: A Practitioner's Approach.* McGraw-Hill Publishing Company, London.

Rombach, H. D., Ulery, B. T. (1989) Improving Software Maintenance Through Measurement. *Proceedings of the IEEE* **77**(4), 581-595.

Royce, W.W. (1970) *Managing the Development of Large Software Systems.* WWW-sivusto, `http://www.cs.umd.edu/class/spring2003/cmsc838p/Process/waterfall.pdf` (4.1.2006).

Senge, P. M., Kleiner, A., Roberts, C., Ross, R. B., Smith, B. J. (1994) *The fifth discipline: the art and practice of the learning organisation.* Doudleday, New York.

Shewhart, W. A. (1939) *Statistical Method from the Viewpoint of Quality Control.* Graduate School of the Depaturement of Agriculture, Washington. (Uudelleenjulkaistu (1986) Dover Publication, Inc., Mineola, New York.)

SEI (2002) *Capability Maturity Model Integration (CMMI), Version 1.1, Continuous Representation.* WWW-sivusto, `http://www.sei.cmu.edu/pub/documents/02.reports/pdf/02tr011.pdf` (27.1.2006).

SEI (2005) *Frequently Asked Questions.* WWW-sivusto, `http://www.sei.cmu.edu/cmmi/faq/15504-faq.html` (31.1.2006).

SEI (2006) *CMMI for Development, Version 1.2.* WWW-sivusto, `http://www.sei.cmu.edu/publications/documents/06.reports/06tr008.html` (7.3.2007).

Sommerville, I. (2000) *Software Engineering.* Addison Wesley.

SPICE (1998) *Phase 2 Trials Interim Report.* WWW-sivusto, `http://www.sqi.gu.edu.au/spice/docs/p2rp100pub.pdf` (31.1.2006)

SPICE Network (2007) *ISO.SPICE.* WWW-sivusto, `http://www.isospice.com` (7.3. 2007).

Springer (2001) *Software Process Improvement: Metrics, Measurement, and Process Modelling.* Berlin.

Storbacka, K., Blomqvist, R., Dahl, J., Haeger, T. (1999) *Asiakkuuden arvon lähteillä.* CRM Finland Oy ja Wsoy, Juva.

Tenhunen, V. (2006) Sähköpostiviesti 6.7.2006.

Tietotekniikan liitto ry:n sanastotoimikunta (2003) *Atk-sanakirja.* Talentum, Helsinki.

Pulford, K., Kuntzmann-Combelles, A., Shirlaw, S. (1995) *A quantitative approach to software management: the AMI handbook,* Addison-Wesley, Boston.

van Latum, F., van Solingen, R., Oivo, M., Hoisl, B., Rombach, D., Ruhe, G. (1998) Adopting GQM-Based Measurement in an Industrial Environment. *IEEE Software* **15**(1), 78-86.

van Solingen, R. (1995) *Goal-oriented software measurement in practice: Introducing software measurement in Schlumberger Retail Petroleum Systems.* Master's Thesis, Schlumberger.

van Solingen, R., Berghout, E. (1999) *The Goal/Question/Metric method: A Practical Guide for Quality Improvement of Software Development.* The McGraw-Hill Companies, London.

van Solingen, R., Berghout, E. (2001) Integrating Goal-Oriented Measurement in industrial Software Engineering: Industrial Experiences with and Additions to the Goal/Question/metric method (GQM). *Proceedings, Seventh International Software Metrics Symposium (METRICS 2001)* (toim. Rombach, D.), IEEE Computer Society, Washington, 246-258.

Westfall, L. (2002) *Software Customer Satisfaction.* WWW-sivusto, `http://www.westfallteam.com/Papers/Customer_Satisfaction_Surveys.pdf` (9.4.2007).

Westfall, L. (2004) *Metrics Report Definition*WWW-sivusto, `http://www.westfallteam.com/software_metrics,_measurement_&_analytical_methods.htm` (7.3.2007).

Wiegers, K. E. (1997) *Software Metrics. Ten Traps to Avoid.* WWW-sivusto, `http://www.processimpact.com/articles/metrics\_primer.html` (28.2.2006).

Ylikoski, T. (2001) *Unohtuiko asiakas?* KY-palvelu Oy, Keuruu.

Zeithaml, V., Bitner, M. (1996) *Services marketing.* McGraw-Hill, New York.

Zuse, H (1998) *A Framework of Software Measurement.* Walter de Gruyter, Berlin.

Åhlberg, M. (1997) *Jatkuva laadunparantaminen korkeatasoisena oppimisena.* Joensuun yliopisto Kasvatustieteiden tiedekunta, Joensuu.

LIITE 1

Alustava kustannusmalli on tärkeä osa GQM-mittausprosessia, sillä mallin avulla johto saa kuvan siitä, onko mittausprosessi taloudellisessa mielessä kannattava, ja päättää sen toteutumisesta. Lopullinen kustannus-hyötyanalyysi (cost-benefit analysis) tulee sisällyttää loppuraporttiin, jottei johto tai projektiryhmä tekisi omia päätelmiä projektin kannattavuudesta. Sen seurauksena jatkuva mittausprosessi saatetaan hylätä korkeiden kustannusten takia.

Tämän liitteen kustannusmalli on laadittu neljän vuoden kokemuksen perusteella kuutta erilaista teollista GQM-mittausprosessia apuna käyttäen (Birk & al., 1998). Kustannusmallissa on eroteltu kuusi eri toimintoa. Kunkin toiminnon kohdalta on laskettu henkilötyötunnit, jotka kuluvat johdolta, GQM- ja projektitiimilta mittausprosessin toteuttamiseen. Kustannukset saadaan kertomalla henkilötyötunnit yrityksen tuntikustannuksilla.

Taulukossa 11 on esitelty kustannusmalli, joka sisältää kaksi eri kustannusvaihtoehtoa: ensimmäisen (eka) GQM-mittausprosessin ja rutiininomaisen (rut) GQM-mittausprosessin. Mittausprosessien kustannukset ovat eroteltu toisistaan, koska ensimmäinen mittausprosessi vaatii enemmän panostusta ja täten tulee kalliimaksi kuin rutiininomainen mittausprosessi.

Taulukon 11 kustannusmallin henkilötyötunnit on laskettu seuraavien tietojen pohjalta: (Birk & al., 1998; van Solingen & Berghout, 1999):

- GQM-tiimi koostuu yhdestä jäsenestä ja projektitiimi kymmenestä ohjelmoijasta.
- Mittausprosessissa on vain yksi päätavoite, mutta kolme erilaista näkökulmaa: johdon, laadunvarmistuksen ja ohjelmoijan.
- Haastattelu täytyy suorittaa neljälle ohjelmoijalle ja yhdelle projektipäällikölle.
- Mittausprosessin työvälineet ja tiedonkeräysmenetelmät ovat valmiiksi suunniteltu ja erillistä koulutusta ei tarvitse järjestää, koska projektitiimin jäsenet tuntevat menetelmät jo entuudestaan.

- Palautetilaisuuksia tulee pitää keskimäärin viisi kertaa, jotta tavoitteet saavutettaisiin. Van Solingenin & Berghoutin (1999) mukaan palautetilaisuuksien määrästä ei tule tinkiä, vaikka ne tuottavat eniten kustannuksia. Palautetilaisuuksista saatava hyöty kattaa nopeasti kustannukset, koska ne ovat tärkein mittausprosessin osa, sillä palautetilaisuuksissa saadaan selville jatkotoimenpiteet prosessin parantamiseksi.

Taulukko 15: Kustannnusmallin sisältämät toiminnot ja niiden henkilötyötunnit ensimmäisessä sekä rutiininomaisessa mittausprosessissa (van Solingen & Berghout, 1999.)

Toiminnot	**GQM-asiantuntija**		**GQM-jäsen**		**Johto**		**Ohjelmoija**		**10 ohjelmoijaa**		**Yhteensä**	
Kustannus vaihtoehdot	eka	rutiini	eka	rutiini	eka	rutiini	eka	rutiini	eka	rutiini	eka	rutiini
Mittausprosessin suunnittelu	24	4	32	-	4	2	4	-	40	-	100	6
Tavoitteiden tunnistaminen ja määrittely	18	8	8	-	2	1	1	1	10	10	38	19
Haastattelujen suorittaminen	35	40	35	-	2	2	1	2	4	8	76	50
GQM-suunnitelmien tuottaminen	196	40	138	-	2	1	1	1	10	6	346	47
Mittaustiedon keruu	-	24	16	-	-	1	3	1,5	30	15	46	40
Mittaustiedon tulkinta/ palautetilaisuus	12	48	48	-	8	2	4	2	40	20	108	70
Mittaustietojen tulkinta/ 5 palautetilaisuutta	60	240	240	-	40	10	20	10	200	100	540	350
Yhteensä	333	356	496	-	50	17	30	15,5	294	139	1146	512

LIITE 2

Taulukko 16: Ylläpidon elinkaarimalli (IEEE, 1998).

	Problem identification	Analysis	Design	Implementation	System test	Acceptance test	Delivery
Input	MR = Modification Request	- Project/system document - Repository information - Validated MR	- Project/system document - Source code - Databases - Analysis phase output	- Source code - Product/system document - Results of design phase	- Updated software documentation - Test-readiness review report - Updated system	Test-readiness review report Fully integrated system Acceptance test • Plans • Cases • Procedures	Tested/ accepted system
Process	- Assign change number - Classify - Accept or reject change - Preliminary magnitude estimate - Prioritize	- Feasibility analysis - Detailed analysis - Redocument, if needed	- Create test cases - Revise • Requirements • Implementation plan	- Code - Unit test - Test-readiness review	- Functional test - Interface testing - Regression testing - Test-readiness review	- Acceptance test - Interoperability test	PCA Install Training PCA = Physical Configuration Audit
Control	- Uniquely identify MR - Enter MR into repository	- Conduct technical review - Verify • Test strategy • Documentation is updated - Identify safety and security issues	- Software inspection/review - Verify design	- Software inspection/ review - Verify • CM control of software • Traceability of design	- CM control of • Code • Listings • MR • Test documentation	- Acceptance test - Functional audit - Establish baseline	PCA VDD VDD = Version Description Audit
Output	- Validated MR - Process determinations	- Feasibility report (FR) - Detailed analysis report - Updated requirements - Preliminary modification list - Implementation plan - Test strategy	- Revised • Modification list • Detail analysis • Implementation plan - Updated • Design baseline • Test plans	- Updated • Software • Design documents • Test documents • User documents • Training material - Test-readiness review report	- Tested system - Test reports	- New system baseline - Acceptance test report - FCA report FCA = Functional Configuration Audit	PCA report VDD

LIITE 3

Taulukko 17: Laskelmia yrityksen mittausaineistosta.

	Kehityksen aikaiset virheet					Asiakkaan ilmoittamat virheet / yhteydenotot
	Lähetetyt	Korjatut	Korjaa-mattomat	Korjaa-mattomat %	Korjatut %	
X / 12 kk						
2002	259	189	70	27,0 %	73,0 %	ei aineistoa
2003	122	103	19	15,6 %	84,4 %	648 (52)
2004	57	15	42	73,7 %	26,3 %	440 (115)
Y / 12 kk						
2002	104	75	29	27,9 %	72,1 %	ei aineistoa
2003	257	221	36	14,0 %	86,0 %	260 (19)
2004	102	57	45	44,1 %	55,9 %	568 (47)
X / 7 kk						
2003	93	68	25	27,0 %	73,0 %	394 (32)
2004	19	6	13	68,4 %	31,6 %	437 (77)
2005	0	9	0	0,0 %	100,0 %	245 (19)
Y / 7 kk						
2003	110	104	6	5,0 %	95,0 %	45 (0)
2004	53	22	31	58,5 %	41,5 %	402 (36)
2005	53	51	2	3,8 %	96,2 %	308 (32)

() = ohjelmavirhe