TAVOITEPOHJAINEN OHJELMISTOPROSESSIN KEHITTÄMINEN

Esa Kröger

06.04.2001

Joensuun yliopisto
Tietojenkäsittelytiede
Pro gradu -tutkielma

TIIVISTELMÄ

Ohjelmistoprosessien kehittämisellä pyritään aikaansaamaan tehokkaampia ohjelmistojen kehitys- ja ylläpitoprosesseja ja siten laadukkaampia ohjelmistotuotteita mahdollisimman pienillä kustannuksilla. Ohjelmistoprosessien kehittämismenetelmät jakautuvat kolmeen eri suuntaukseen: arviointikehykset, formaali määrittäminen ja mittausperustainen kehittäminen. Tässä tutkielmassa käsitellään tarkemmin mittausperustaista kehittämistä tavoitteisiin perustuvan GQM-paradigman avulla. GQM-paradigman lähtökohtana on pyrkimys saavuttaa ennalta asetetut organisatoriset tavoitteet ja siten saada aikaan prosessien kehittymistä erityisen mittausohjelman avulla. Tutkielmassa esitellään tavoitteisiin perustuvan mittaamisen malli ja sitä tukeva mittausprosessi, jonka vaiheet käydään läpi käytännön esimerkkien avulla. Ohjelmistoprosessien kehittäminen ei ole kertaluonteinen, vaan jatkuva käytäntö, jolloin siitä saatavia hyötyjä ja kustannuksia arvioitaessakin pitää ottaa kehittämisprosesseissa saatu kokemus huomioon.

Avainsanat: GQM, mittausohjelma, ohjelmistoprosessin kehittäminen, tavoitepohjainen mittaaminen

SISÄLLYSLUETTELO

1 JOHDANTO

Ohjelmistot ovat monimutkaisia tuotteita, joita on hyvin vaikea kehittää ja testata. Hyvin usein ohjelmistoista löytyy odottamattomia ja epätoivottuja ominaisuuksia, jotka voivat pahimmillaan aiheuttaa pahoja ongelmia ja jopa tuhoja (Fuggetta 2000). Ohjelmistotuotanto on siis hyvin virhealtista toimintaa. Ohjelmistoprojektit ovat myös usein myöhässä aikataulusta, ylittävät niille laaditut budjetit tai vaativat enemmän resursseja kuin mitä alunperin oli suunniteltu.

Ohjelmistotuotannon kehittäminen on suuri haaste organisaatioille. Vaikka lähestymistapoja ja menetelmiä prosessien kehittämiseen on ollut olemassa ja on kehitetty jo vuosikymmeniä, ovat kehittämishankkeet olleet melko harvinaisia, koska ne kuluttavat resursseja ja epäonnistumisen riski on ollut suuri ilman selviä kehitystavoitteita. Yhtenä syynä voidaan pitää alalla vallitsevaa kovaa kilpailua ja siitä johtuvaa työvoimapulaa, jolloin henkilöstöresurssien käyttäminen prosessien kehittämiseen on jäänyt toissijaiseksi. Tästä syystä lähinnä suurilla ja keskisuurilla yrityksillä on ollut resursseja soveltaa kehitettyjä laadunparannusmenetelmiä.

Yksi tapa saada palautetta ja arvioida organisaation ohjelmistotuotantoa on ohjelmistoprosessien mittaaminen. Mittausteorian tuntemuksen ja kokemuksen lisääntyessä mittaamista onkin alettu pitää tehokkaana menetelmänä ohjelmistojen kehitys- ja ylläpitoprojektien ymmärtämiseen, valvontaan, ohjaamiseen, ennustamiseen ja kehittämiseen (Briand et al. 1996).

Mittaamisen avulla ei suoranaisesti voida ratkaista ohjelmistoprosesseihin liittyviä ongelmia, mutta sen avulla voidaan vastata moniin ohjelmistotuotantoa koskeviin kysymyksiin, joiden perusteella luodaan pohja prosessien kehittämiselle (Florac et al. 1997). Sen avulla voidaan esimerkiksi määrittää nykyisten prosessien ja tuotteiden heikkouksia ja vahvuuksia ja arvioida niiden laatua. Lisäksi mittaaminen auttaa projektin seurannassa, mittausten perusteella tehtyjen korjaustoimenpiteiden tekemisessä ja näiden toimenpiteiden vaikutusten seurannassa (Basili et al. 1994b).

Tavoitepohjaisen kehittämisen tarkoitus on mahdollistaa prosessien kehittäminen asetettujen liiketoiminnallisten tai muiden tavoitteiden avulla. Yhdistämällä tavoitepohjaisuus ohjelmistoprosessien mittaamiseen mahdollistetaan ohjelmistoprosessin mittaustoimien keskittäminen tavoitteiden kannalta olennaisiin seikkoihin ja siten estetään olennaisten tietojen poisjäänti ja turhan tiedon kerääminen.

Tässä tutkielmassa keskitytään ohjelmistoprosessien kehittämisen tarkasteluun ja Basilin et al. (1994b) kehittämän GQM-paradigman soveltamiseen tavoitepohjaisen mittausohjelman määrittelyssä. Luvussa 2 perehdytään ohjelmistoprosessin kehittämiseen yleisesti, tutkitaan yleisesti mistä ohjelmistotuotantoprosessit koostuvat, mitä ohjelmistoprosesseissa voidaan kehittää ja mitä eri suuntauksia ohjelmistoprosessien kehittämiseksi on syntynyt. Lisäksi luvun lopussa kerrotaan ohjelmistomittareista ja niiden luokittelusta.

Luvussa 3 perehdytään tavoitepohjaiseen ohjelmistoprosessin mittaamiseen GQM-paradigman mukaisesti. Ensin tutustutaan tavoitepohjaiseen mittausmalliin ja siihen kuinka se GQM-lähestymistapaa soveltaen muodostetaan tavoitteiden, kysymysten ja mittareiden avulla. Mittausprosessi käydään läpi vaihe vaiheelta ja luvun lopussa arvioidaan mittausohjelman kustannuksia ja hyötyjä eri organisaatioista kerättyjen, kirjallisuudessa esitettyjen, tietojen perusteella. Mittausprosessin kuvaamisen yhteydessä esitetään esimerkkejä puutetietojen keräämisestä ja tulkitsemisesta erään mittauksentukijärjestelmän (Kröger 1999) avulla.

Luvussa 4 perehdytään tarkemmin jatkuvaa laadun parantamista tukeviin järjestelyihin. Lopuksi luvussa 5 tehdään yhteenveto tarkastellusta mittausprosessista ja arvioidaan sen käyttökelpoisuutta.

2 OHJELMISTOPROSESSIN KEHITTÄMINEN

Ohjelmistoprosessin kehittämisellä (Software Process Improvement, SPI) tarkoitetaan toimia, joilla pyritään kehittämään organisaation ohjelmistotuotantoprosessia ja siten tuottamaan esimerkiksi laadukkaampia tuotteita matalammilla kustannuksilla. Kun suunnitellaan parannuksia ohjelmistoprosesseihin, on tärkeää ymmärtää, mitkä prosessitekijät ja asiakastarpeet vaikuttavat tuotteen laatuun (Mashiko ja Basili 1997). Ohjelmistoprosessien kehittämistoimenpiteet tulee perustaa prosessien ja tuotteiden ominaisuuksien sekä asiakastarpeiden ymmärtämiselle. Rakentamalla malleja näiden tekijöiden välisten suhteiden esittämiseksi, voidaan paremmin ymmärtää ohjelmistoprosessia. Mittaaminen on keskeinen asia näiden suhteiden ymmärtämisessä, jalostamisessa ja validoinnissa.

2.1 Ohjelmistotuotantoprosessi

Ohjelmistotuotannon kehittämiskohteita ovat ohjelmistoprosessit ja ohjelmistotuotteet. Tärkeimpiä kehittämistavoitteita ovat laadun parantaminen, projektin keston lyhentäminen sekä kustannusten ja riskien pienentäminen (Solingen ja Berghout 1999). Tavoitteiden saavuttamiseksi tutkijat ja ohjelmistotuottajat ovat yhä enemmän kiinnittäneet huomiota kehitettävien ohjelmistojen laadun parantamiseen. Ohjelmistoprosessien parantamiseksi on kehitetty erilaisia näkökulmia ja menetelmiä (Fuggetta 2000).

2.1.1 Ohjelmistotuotteet

Ohjelmistotuotteella tarkoitetaan tietokoneohjelmia toimintoineen, ohjelmiin kuuluvia dokumentteja ja tietoja, jotka on tarkoitettu toimitettavaksi käyttäjälle (Solingen ja Berghout 1999). Lisäksi ohjelmistotuotannon eri vaiheissa syntyy muita tuotoksia, kuten esim. prototyyppejä (Fenton ja Pfleeger 1997). Koska ohjelmistot ovat usein yhä suurempia ja monimutkaisempia, voivat myös ohjelmistoprosessit olla vaativampia ja monimutkaisempia, jolloin ohjelmistojen kehittäjät eivät välttämättä pysty tarjoamaan luotettavaa tietoa ohjelmistojensa laadusta. Tämä onkin yksi syy, miksi monet ohjelmistoprojektit ovat epäonnistuneet. Ohjelmistojen on pystyttävä täyttämään erilaisten käyttäjien vaatimukset. Ei riitä, että ohjelmistossa on kaikki tarvittavat toiminnot ja että se toimii halutulla tavalla, vaan sen pitää myös toimia loogisesti, olla hyvin dokumentoitu ja tuettu käyttäjien koulutuksella (Solingen ja Berghout 1999).

Ohjelmistot eivät ole staattisia, vaan niitä pitää kehittää koko ajan ympäristön muuttuessa. Ohjelmistojen joustavuutta pidetään yhtenä niiden tärkeimmistä vahvuuksista. Koska ohjelmistojen vaatimukset muuttuvat ajan myötä, etenkin ylläpidon aikana, on tärkeää, että ohjelmisto joustaa ja on helposti ylläpidettävissä. Toisaalta joustavuus on yksi ohjelmistojen suurimmista heikkouksista, koska ylläpidettäessä ohjelmisto voi muuttua niin paljon, että alkuperäinen arkkitehtuuri joutuu koetukselle (Solingen ja Berghout 1999).

2.1.2 Ohjelmistoprosessit

Yleisesti prosessilla tarkoitetaan ohjelmistotuotantoon liittyvien toimintojen kokoelmaa (Fenton ja Pfleeger 1997). Ohjelmiston elinkaareen kuuluu karkealla tasolla kaksi tällaista prosessia. Ensimmäisenä on ohjelmiston kehitysvaihe, jossa ohjelmisto alunperin luodaan. Toinen vaihe on käyttövaihe, jonka aikana ohjelmistoa käytetään ja tarvittaessa tehdään muutoksia ohjelmiston ylläpitämiseksi. Ohjelmiston kehitysprosessi määritellään kaikkina käyttäjien vaatimukset ohjelmistoiksi muuttavina toimintoina (Solingen ja Berghout 1999). Sekä kehitys- että ylläpitoprosessi voidaan jakaa aliprosesseiksi tuotannon ja tukitoimintojen osalta.

Prosessiin liittyy usein aikajana eli prosessien toiminnot liittyvät toisiinsa tai ovat usein järjestettyjä ajallisesti niin, että jokin toiminto on suoritettava ennen kuin toinen voi alkaa. Ajoitus voi olla eksplisiittinen, jolloin esimerkiksi suunnitteluvaiheen on alettava 31. lokakuuta, tai implisiittinen, jolloin esim. suunnitteluvaiheen on päätyttävä ennen kuin toteutus voi alkaa. Resurssit ja prosessit liittyvät läheisesti toisiinsa, sillä jokaisella prosessilla on resursseja, joita se käyttää ja tuotoksia, joita prosessissa syntyy. Näin jonkin prosessin aikaansaama tuotos voi olla toisen prosessin syöte. Esimerkiksi suunnitelma on suunnitteluprosessin tuotos ja samalla se toimii syötteenä toteutusprosessille (Fenton ja Pfleeger 1997).

Pfleegerin (1998) mukaan ohjelmiston kehitysprosessi koostuu seuraavista vaiheista: vaatimusten analysointi ja määrittely, järjestelmän suunnittelu, ohjelman suunnittelu, ohjelman toteutus, yksikkötestaus, integrointitestaus, järjestelmätestaus, järjestelmän toimitus ja ylläpito. Vaiheilla on yhteyksiä toisiinsa ja sekä analysoinnissa että suunnittelussa voidaan käyttää koerakentamista (prototyping).

Kehitysprosessit ovat laadun parantamisen kannalta tärkeitä, koska niiden aikana ohjelmistotuote konkreettisesti luodaan. Mitä aiemmin laatuun liittyvät ongelmat havaitaan, sitä helpommin ja sitä halvemmaksi niiden korjaaminen tulee. Ohjelmistoprosesseihin liittyviä tyypillisiä ongelmia ovat Solingenin ja Berghoutin (1999) mukaan: ohjelmistoprosesseja ei ole määritelty ollenkaan tai ne on määritelty huonosti, ohjelmistojen kehitys on suurelta osin riippuvaista tiettyjen yksilöiden taidoista ja ohjelmistotuotanto on vaikeasti hallittavissa.

Selkeästi määriteltyjen prosessien puuttumisen vuoksi ohjelmistotuotanto on paljolti riippuvainen yksilöiden kyvyistä, kokemuksista ja ammattitaidoista. Ohjelmistotuotannon riippuvuus yksilöistä tekee siitä vaikeasti hallittavan. Esimerkiksi tehtävien vaihtaminen kehittäjien kesken on vaikeaa, koska eri ihmiset tekevät asioita eri tavalla ja eri nopeudella (Solingen ja Berghout 1999).

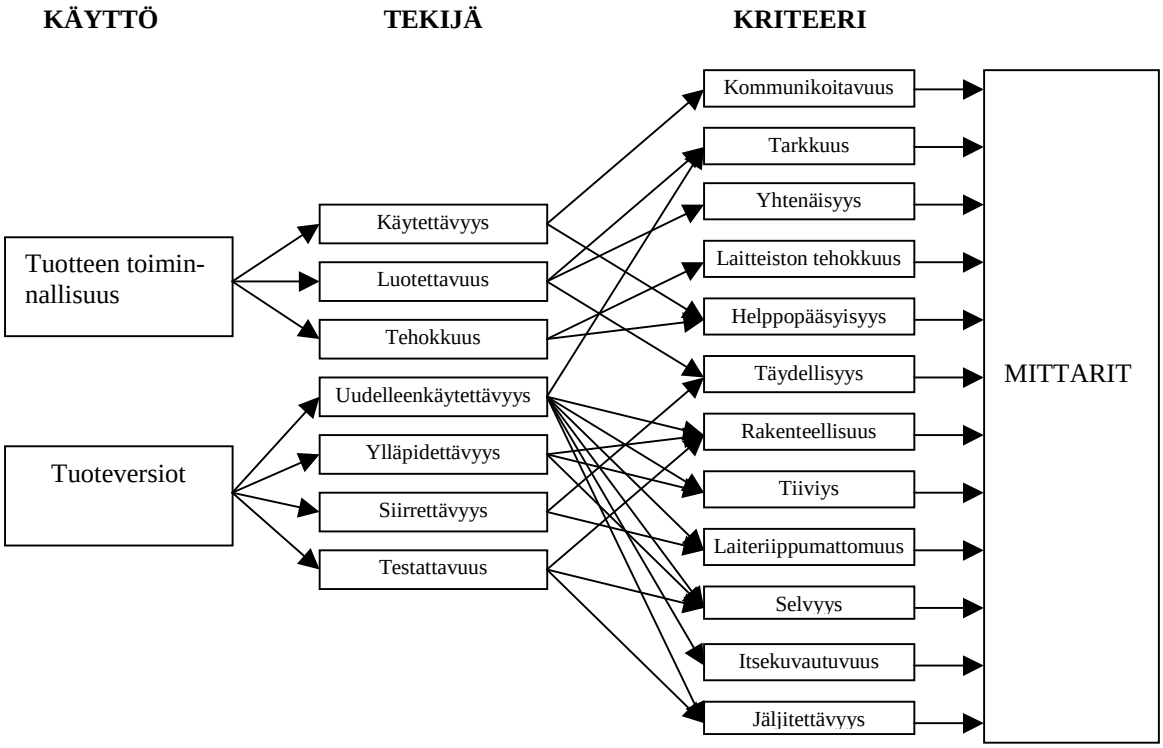
Jos kehitysprosesseja ei määritellä ja dokumentoida selkeästi, eikä yhteisiä pelisääntöjä ole siten olemassa, on myös toimitusaikojen arviointi vaikeampaa, sillä yksilöiden toiminnan kontrollointi ja ohjelmistoprojektin etenemisen seuranta ilman prosessien määrittelyä on mahdotonta. Tämä vaikuttaa projektin aikatauluun, joka on useimmiten liian tiukka, koska yrityksen liiketoiminnan kannalta on tärkeää, että projektit saadaan valmiiksi mahdollisimman nopeasti (Solingen ja Berghout 1999).

2.1.3 Ohjelmistojen laatu

Ohjelmiston laatu koostuu kaikista niistä tekijöistä, jotka vaikuttavat käyttäjän tyytyväisyyteen. Laadun voidaan ajatella koostuvan esimerkiksi toiminnallisuudesta, luotettavuudesta, käytettävyydestä, tehokkuudesta, testattavuudesta, ylläpidettävyydestä, uudelleenkäytettävyydestä ja siirrettävyydestä (Fenton ja Pfleeger 1997). Kuvassa 1 on esitetty esimerkki laadun osatekijöistä. Kuhunkin laatutekijään liittyvät tietyt kriteerit, joiden toteutumista voidaan arvioida mittareiden avulla.

Hinnan lisäksi laadulla on suuri merkitys, kun ohjelmistotuotetta aletaan markkinoida (Solingen ja Berghout 1999). Yrityksen pitää jollakin tavoin pystyä vakuuttamaan asiakkaat tuotteidensa laadusta. Tämä voidaan tehdä esimerkiksi käyttämällä selkeästi määriteltyjä, hyväksi todettuja ohjelmistoprosesseja, dokumentoimalla tuotteet hyvin ja tarjoamalla

käyttäjäkoulutusta. Kaikki nämä seikat antavat asiakkaalle kuvan yrityksestä, joka panostaa tuotteisiinsa ja niiden laatuun. Yritys, jolla tiedetään olevan hallittua ohjelmistotuotantoa, pärjää todennäköisesti paremmin kuin yritys, jonka ohjelmistotuotanto on kaoottista. Vaikkei tieto hallitusta ohjelmistotuotannosta sinällään kerrokaan mitään tuotteiden laadusta, on sillä varmasti selkeä vaikutus yrityksen imagoon.



Kuva 1. Ohjelmiston laatumalli (Fenton ja Pfleeger 1997).

Ohjelmiston laatuun liittyy kaksi perusongelmaa. Ensinnäkin ohjelmiston laatua on vaikea määrittää mitattavilla termeillä ja toiseksi on vaikeaa valita tehokkain kehitysprosessi, joka tuottaa halutun laatuisia tuotteita (Solingen ja Berghout 1999). Nämä perusongelmat ovat selvästikin sidoksissa toisiinsa.

Laatuun liittyy myös monia näkökantoja, jotka vaativat subjektiivisia päätöksiä. Esimerkiksi ohjelmiston käytettävyys on selvästi subjektiivisesti mitattava asia ja riippuu siten arvioijasta. Toisaalta ohjelmiston luotettavuutta voidaan mitata objektiivisesti esimerkiksi virheiden määrän perusteella (Solingen ja Berghout 1999).

Käytännössä määrittelyvaiheessa ei kaikkiin laatuseikkoihin välttämättä kiinnitetä huomiota, vaan huomio keskittyy lähinnä toiminnallisuuteen. Tällöin muut laatuun liittyvät tekijät jäävät kehittäjien kokemuksen, ammattitaidon ja henkilökohtaisen mielenkiinnon varaan (Solingen ja Berghout 1999).

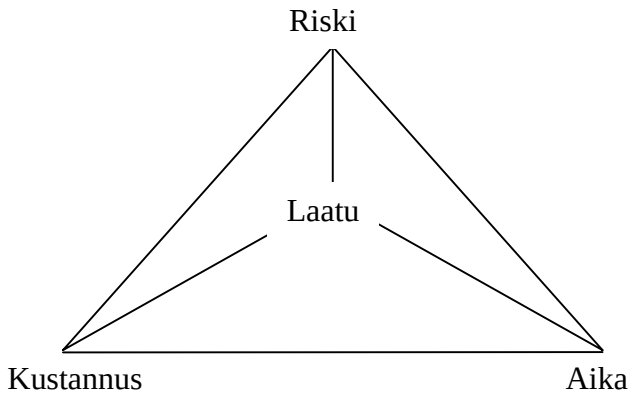
Vaikka laatu olisikin riittävän hyvin määritelty, ei sen saavuttaminen ole helppoa. Käytettävien menetelmien, tekniikoiden ja apuvälineiden vaikutusta ei useinkaan tunneta, jolloin sopivimman kehitysprosessin valitseminen on hyvin vaikeaa. Lisäksi kehitysprosesseissa käytetään riittämätöntä mittausta, jotta sen avulla voitaisiin tehdä korjaustoimenpiteitä (Solingen ja Berghout 1999).

2.2 Ohjelmistoprosessin kehittämisen tavoitteet

Organisaatioiden tulee määrittää selkeät kehitystavoitteet, jottei prosessien kehitystoimista tule yhtä kaoottisia kuin itse tuotantoprosesseista. Näiden kehitystavoitteiden tulee tukea liiketoimintatavoitteita mahdollisimman hyvin. Kehitystavoitteet voidaan määrittää monin eri tavoin. Ensinnäkin organisaation johto voi määrittää laatutavoitteet, joista kehitystavoitteet voidaan johtaa. Toiseksi organisaatio voi suorittaa ohjelmistoprosessin arvioinnin, jotta kehitettävät kohteet voidaan tunnistaa ja määrittää (Solingen ja Berghout 1999).

Liiketoimintatavoitteiden muovaaminen ohjelmistotuotannon kehitystavoitteiksi on aina vaikeaa, mutta tähän voi käyttää olemassa olevia menetelmiä ja kirjallisuutta (Paulk 1995, Solingen ja Berghout 1999). Myöhemmin luvussa 3 esiteltävä GQM-menetelmä soveltuu käytettävien mittareiden johtamiseen yrityksen johdon asettamista liiketoimintatavoitteista. Myös mittaamisen avulla voidaan löytää kehittämisen kohteita ohjelmistoprosesseista. Lisäksi kehitystavoitteet voidaan määrittää jostakin tietystä projektista saatujen tietojen perusteella (Solingen ja Berghout 1999).

Neljä tärkeintä, osin toisistaan riippuvaa kehitystavoitetta ovat kuvan 2 mukaisesti: tuotteiden laadun parantaminen, projektin keston lyhentäminen, projektin kustannusten pienentäminen ja projektin riskien vähentäminen.



Kuva 2. Prosessin parantamisen osatekijät (Solingen ja Berghout 1999).

2.2.1 Laadun parantaminen

Jos organisaatio haluaa parantaa tuotteidensa laatua, pitää tuotteen laadun kehitystavoitteet määritellä. Tämä lisää tarvetta määritellä paremmin toiminnallisuuden, luotettavuuden ja muiden laatuattribuuttien vaatimukset (Solingen ja Berghout 1999).

Jotta tuotteiden laadun parantaminen voidaan ottaa kehitystavoitteeksi, pitää organisaation käyttämät prosessit määrittää selkeästi. Tuotteiden laatu on ohjelmistojen tekijöiden vastuulla, joten heidän pitää olla selvillä syistä, miksi laadukkaampia tuotteita tarvitaan. Jotta projektiryhmät saadaan mukaan laadunparannukseen, on niihin kuuluvat henkilöt saatava ymmärtämään sen tärkeys organisaation liiketoiminnalle. Laadun parantamisen syy ei välttämättä ole se, että tuotteissa olisi jotain vikaa, vaan entistä paremman laadun saavuttaminen.

Laadun kehittäminen aloitetaan usein puutteiden mittauksella ja tulosten tallentamisella. Tällöin voidaan selvittää tärkeimmät tuotteiden ja prosessien kehityskohteet, kun tunnetaan puutteiden syyt (Solingen ja Berghout 1999). Laadun parantumista voidaan seurata ajallisesti peräkkäisissä projekteissa.

2.2.2 Projektin keston lyhentäminen

Projektin keston nopeuttaminen on yksi tärkeä kehitystavoite, jota toteutetaan käytännössä. Tämä aiheutuu markkinoiden ja kilpailun kovuudesta. Projektin elinkaarta tai kehitysaikaa voidaan lyhentää esimerkiksi lisäämällä tuottavuutta, rinnakkaiskehityksellä, koerakenta-

misella tai aikaisemmin luotujen ohjelmistojen uudelleenkäytöllä. Tuottavuutta voidaan lisätä esimerkiksi erikoistamalla ohjelman suunnittelua. Rinnakkaistuotantoa voidaan lisätä ohjelmiston modulaarisella suunnittelulla (Solingen ja Berghout 1999).

Projektin keston lyhentäminen on usein yksi syy määrittää kehitystavoite suhteessa ohjelmiston kehittäjien tuottavuuteen. Yksilöiden työn mittausta ja arviointia pidetään houkuttelevana ideana, koska näin saataisiin tietoja ihmisten tehokkuudesta ja kyvyistä. Tällaisen lähestymistavan käytön suhteen pitää kuitenkin olla varovainen, koska yksilön suorituskyvyn mittaaminen on vaikeaa. Jos lisäksi ohjelmistojen kehittäjät eivät tue lähestymistapaa, se voi tuhota koko parannusaloitteen. Valvonta voi helposti aiheuttaa ongelmia työmotivaatiossa, työilmapiirissä ja siten myös alentaa ohjelmoijien tuottavuutta (Solingen ja Berghout 1999).

Ohjelmistoprosessin kehittäminen on joka tapauksessa hyvä keino tutkia vaikuttaako uusien menetelmien ja välineiden käyttöönotto projektin kestoon lyhentävästi (Solingen ja Berghout 1999). Samantyyppisiä projekteja voidaan keston (ja laadun) suhteen verrata toisiinsa, kun projekteissa käytetään eri menetelmiä tai välineitä.

2.2.3 Kustannusten pienentäminen

Kustannusten pienentäminen tähtää lähinnä työmäärän pienentämiseen ja sitä kautta ohjelmistoprosessien tehokkaampaan suorittamiseen. Aluksi kannattaa tutkia, millaisia työmäärät ja kustannukset ovat mittauksen aloitushetkellä. Näitä voidaan ilmaista erilaisilla mittayksiköillä tutkimalla esim. mitä ovat kustannukset koodiriviä, toimintopistettä (function point), elinkaaren vaihetta tai vaiheen tuotosta kohden. Ohjelmiston mittaaminen soveltuu tähän työhön erinomaisesti. Näin voidaan varmistaa, ettei asioita tehdä pelkästään paremmin, vaan että tehdään oikeita asioita, koska eliminoimalla turhat toiminnot saadaan aikaan kustannussäästöjä. Kustannussäästöjä saadaan aikaan myös käyttämällä uudelleen ohjelmistojen osia tai dokumentteja (Solingen ja Berghout 1999).

Syy kustannusten pienentämiseen on luonnollisesti se, että organisaation johto on useimmiten kiinnostunut rahaan liittyvistä luvuista. Jos ohjelmistotuotannon kustannuksia saadaan alennettua, antaa se myös suuren edun kilpailtaessa uusista asiakkaista, koska tuotetta

voidaan tarjota asiakkaille edullisemmin (Solingen ja Berghout 1999). Kustannuksia ei kuitenkaan voi pienentää laadusta tinkimällä.

2.2.4 Riskien pienentäminen

Ohjelmistoprojekteihin liittyvien riskitekijöiden hallinta on tärkeää riskien pienentämiseksi (Pfleeger 1998). Tämä voidaan tehdä tunnistamalla riskit ja mittaamalla riskeihin liittyvien kohteiden nykytilaa ja tarvittaessa kohdistamalla korjaustoimenpiteet noihin kohteisiin. Nostettaessa prosessien *kypsyystasoa* (Paulk et al. 1995) prosesseihin liittyvien riskien määrä pienenee, sillä kehitysprosessien ongelma-alueet vähenevät prosessien kehitystoimien myötä. Riskien vähentäminen tekee prosesseista hallittavampia ja on siten sopiva keino ohjelmistoprosessien parantamiseen (Solingen ja Berghout 1999).

2.3 Ohjelmistoprosessin kehittämismenetelmiä

Ohjelmistoprosessien kehittämistä on tutkittu runsaasti ja paljon erilaisia malleja, kehyksiä ja teknologioita on kehitetty prosessien määrittelyä, arviointia ja kehittämistä varten. Nämä tutkimukset voidaan jakaa kolmeen eri suuntaukseen (Krishnan ja Kellner 1999): arviointikehykset, formaali määrittäminen ja mittausperustainen kehittäminen. Lähestymistavat eivät ole toisiaan täysin poissulkevia, mutta kullakin on omat erikoispiirteensä. Vaikka tutkimus näillä kolmella eri suuntauksella onkin ollut suurimmalta osin itsenäistä, on lähestymistapojen välillä havaittavissa myös yhteisiä piirteitä.

2.3.1 Arviointikehykset

Vuosikymmenten varrella on kehitetty *arviointikehyksiä* (assessment frameworks) ohjelmisto-organisaation prosessien *kypsyyden* (maturity) ja *kyvykkyyden* (capability) arviointiin. Arviointikehysten mukaisesti johdonmukainen, hyvin määriteltyjen ja mitattujen, prosessien soveltaminen yhdistettynä jatkuvaan prosessien kehittämiseen saa aikaan olennaista tuottavuuden ja laadun parannusta organisaation tuotannossa. Tällaisia kokonaisvaltaisia lähestymistapoja ovat esimerkiksi CMM (Software Capability Maturity Model), SPICE (Software Process Improvement and Capability dEtermination) ja ISO 9000 (Pfleeger 1998).

IBM:n ja SEI:n (Software Engineering Institute) kehittämän *kypsyysmallin* (CMM) tavoitteena on parantaa ohjelmistotuotannon hallittavuutta (Paulk 1995, Paulk et al. 1995, Pfleeger 1995). Tähän voidaan päästä tutkimalla ja kehittämällä ohjelmistotuotannon eri osa-alueita. Mitä korkeammalla *kypsyystasolla* organisaation ohjelmistotuotanto on, sitä paremmin sitä voidaan hallita ja kehittää. SEI on määritellyt monitasoisen kehyksen, jonka mukaan organisaation ohjelmistotuotannon *kyvykkyyttä* voidaan arvioida ja kehittää (Paulk et al. 1995). Kullakin tasolla edellytetään tietyt käytännöt. Ylemmillä tasoilla prosessien *näkyvyys* ja valvonta paranee, jolloin tuloksena pitäisi olla parempia ohjelmistotuotteita (Pfleeger 1995, The ami Handbook 1992).

SPICE on nimensä mukaisesti tarkoitettu prosessien parantamiseen ja kyvykkyyden määrittämiseen. Pfleegerin (1998) mukaan SPICE:n lähtökohtina ovat olleet CMM ja CMM:n pohjalta Euroopassa kehitetty BOOTSTRAP-menetelmä (Kuvaja et al. 1994). SPICE-arkkitehtuuri on kaksiulotteinen (Rout 1995) siten, että prosessit ja niiden ryhmittelyt muodostavat toisen ulottuvuuden ja kyvykkyystasot toisen ulottuvuuden. Arvioitavat prosessit on ryhmitelty alueisiin sen mukaan, mihin kukin prosessi parhaiten sopii. Oleellista on, että prosessien kyvykkyys voi kasvaa eri tahdissa, koska kunkin prosessin ominaisuudet arvioidaan erikseen.

Kansainvälinen standardointijärjestö ISO (International Standards Organization) on tuottanut standardikokoelman ISO 9000. Standardeista ISO 9001 soveltuu parhaiten ohjelmistotuotantoon (Fenton ja Pfleeger 1997, Pfleeger 1998). Se on kuitenkin kuvattu hyvin yleisellä tasolla, joten sitä tulkitaan ohjelmistotuotannossa standardin ISO 9003 avulla. Esimerkiksi kohta 4.2 standardissa ISO 9001 edellyttää organisaatiolta laatujärjestelmää. Standardi ISO 9003 selittää, kuinka laatujärjestelmä integroidaan ohjelmiston kehitysprojektiin.

2.3.2 Formaali määrittäminen

Toinen prosessien kehittämissuuntaus sisältää menetelmiä ja tekniikoita, jotka helpottavat prosessien formaalia määrittelyä ja kuvausta, analysointia ja automaatiota. Näitä tavoitteita tukemaan on tehty paljon tutkimusta formaalien ja semiformaalien prosessien mallinnuskielten kehittämiseksi (Krishnan ja Kellner 1999). Minkowitzin (1993) mukaan formaalit menetelmät tukevat hyvin prosessien mallinnusta. Eräänä suuntauksena ovat olleet meta-

mallit, joiden avulla on kuvattu prosessien rakenteita tavoitteena prosesseja tukevien ohjelmistoapuvälineiden toteuttaminen (Moser 1995).

Ohjelmistoapuvälineet käyttävät laadittuja prosessikuvauksia ohjaamaan ja tukemaan ohjelmoijia ottamalla osaa prosessiin ja mahdollisesti avustamalla joidenkin prosessien suorituksessa. Yksi tärkeimmistä haasteista on ollut prosessien *säätäminen* (enacting) (Christie 1994), jolla on pyritty parantamaan välineiden kykyä sietää odottamattomia tilanteita ja poikkeamia määritellyistä prosessimalleista. Tällaiset poikkeamat prosessimalleista aiheuttavat helposti epäjohdonmukaisuuksia todellisen prosessin ja prosessia tukevan järjestelmän välillä (Krishnan ja Kellner 1999).

2.3.3 Mittausperustainen kehittäminen

Arviointikehyksiin sisältyy osana myös ohjelmistotuotteen ja –prosessin mittaaminen. Prosessia on mahdollista mitata, kun se on riittävän hyvin määritelty. Prosessi pitää olla hallittu ja vakiinnutettu organisaatiossa. Pfleegerin (1998) mukaan CMM ja SPICE tukevat eksplisiittisemmin mittausta kuin ISO 9000 standardi.

NASA:ssa kehitetty *laadunparannusparadigma* (Quality Improvement Paradigm, QIP) keskittyy kehittämään ja löytämään parhaat menetelmät organisaatiossa ilman vahvaa riippuvuutta arviointikehyksiin. QIP sisältää uusien menetelmien ja teknologioiden kokeilua kohdeorganisaatiossa, tulosten mittaamista ja mittaustulosten määrällistä analysointia parhaiden menetelmien määrittämiseksi ja menetelmien käyttökohteiden selvittämiseksi (Krishnan ja Kellner 1999). Laadunparannusparadigmaa tukee luvussa 3 tarkasteltava Basilin et al. (1994b) kehittämä GQM-paradigma, jonka avulla mittausta voidaan suorittaa tavoitepohjaisesti. Laadunparannusparadigmaa tarkastellaan luvussa 4 osana jatkuvaa laadun parantamista

Humphrey (1997) on esittänyt mittausperustaiseen suuntaukseen rinnastettavan yksilökohtaisen menetelmän. Henkilökohtaisen ohjelmistoprosessin (Personal Software Process, PSP) avulla yksittäiset henkilöt voivat kehittää omaa ohjelmistoprosessiaan määriteltyjen prosessien, mittaamisen sekä tilastollisten prosessinhallintatekniikoiden avulla.

2.4 Ohjelmistomittarit

Ohjelmistotuotantoa tarkasteltaessa voidaan löytää kolme kohdeluokkaa, joiden attribuutteja voidaan mitata. Näitä ovat *tuotteet*, *prosessit* ja *resurssit*. Kunkin kohteen kohdalla voidaan erottaa sisäiset ja ulkoiset attribuutit. *Sisäisillä attribuuteilla* tarkoitetaan niitä kohteen ominaisuuksia, joita voidaan mitata tutkimalla itse tuotetta, prosessia tai resurssia huomioimatta sen käyttäytymistä. *Ulkoisilla attribuuteilla* tarkoitetaan puolestaan niitä ominaisuuksia, joita mitataan suhteessa siihen, kuinka tuote, prosessi tai resurssi liittyy ympäristöönsä. Tässä tapauksessa kohteen käyttäytyminen on tärkeää, ei siis itse kohde (Fenton ja Pfleeger 1997).

Sisäisten ja ulkoisten attribuuttien eroa voidaan havainnollistaa esimerkiksi ohjelman koodimoduulien avulla. Ilman ohjelman suorittamista voidaan määrittää monia tärkeitä sisäisiä attribuutteja kuten esimerkiksi koko, monimutkaisuus ja moduulien väliset riippuvuudet. Koodia luettaessa saattaa löytyä myös virheitä. Osa attribuuteista voidaan kuitenkin mitata vain suorittamalla ohjelmakoodi. Tällaisia attribuutteja ovat esimerkiksi käyttäjän löytämät virheet, tietokantahaun tekemiseen kuluva aika tai eri näyttöjen välillä liikkumiseen liittyvät ongelmat. Näistä attribuuteista voidaan helposti havaita, että ne riippuvat koodin käyttäytymisestä eli ne ovat ulkoisia attribuutteja (Fenton ja Pfleeger 1997).

Ohjelmistomittareilla tarkoitetaan ohjelmistotuotteiden, -prosessien ja -resurssien attribuuttien mittauksessa käytettäviä mittareita. Mittareiden avulla pyritään selvittämään erilaisten kohteiden ja niihin liittyvien attribuuttien arvoja. Mittauksessa kerätään tietoja näistä kohteista mittareiden avulla ja kuhunkin mittariin liittyy usein monia mittaustuloksia (Solingen ja Berghout 1999).

2.4.1 Ohjelmistomittareiden luokittelusta

Yleisimpiä ohjelmistomittareiden luokittelutapoja ovat (Solingen ja Berghout 1999):

- objektiiviset ja subjektiiviset mittarit
- suorat ja epäsuorat mittarit
- dynaamiset ja staattiset mittarit
- ennustavat ja selittävät mittarit

- absoluuttiset ja suhteelliset mittarit.

Objektiiviset mittarit mittaavat absoluuttisesti tuotetta tai prosessia siten, että näihin mittareihin ei vaikuta mittaajan näkökulma. Tällaisia mittareita ovat esimerkiksi ohjelmakoodirivien tai virheiden lukumäärä. *Subjektiiviset mittarit* sitä vastoin sisältävät mittaajan näkökulman, jolloin hänen subjektiivinen arvionsa vaikuttaa mittaustulokseen. Tällaisia mittareita ovat esim. odotettu monimutkaisuus, yhdenmukaisuuden aste ohjelmointistandardeihin nähden tai käyttöohjeen luettavuus (Solingen ja Berghout 1999).

Suorilla mittareilla tarkoitetaan mittareita, joiden arvoon minkään toisen mittarin arvo ei vaikuta. Grady ja Caswell (1987) käyttävät tällaisista mittareista nimitystä *primitiiviset* mittarit. Esimerkiksi virheiden lukumäärä tai prosessiin käytetty tuntimäärä ovat suoria mittareita. *Epäsuorat mittarit* sisältävät yhden tai useamman muun piirteen mittaamisen. Näistä mittareista Grady ja Caswell (1987) käyttävät nimitystä *lasketut* (computed) mittarit. Tuottavuus ja virhetiheys ovat esimerkkejä tällaisista mittareista. Epäsuorat mittarit sisältävät aina vähintään kaksi eri mittaria, joiden avulla kolmas, haluttu arvo lasketaan (Solingen ja Berghout 1999).

Dynaamiset mittarit sisältävät aikadimension eli niiden arvo voi olla erilainen riippuen mittausajankohdasta. Jos mitataan projektin etenemistä, saadaan projektin alussa erilaisia arvoja kuin projektin lopussa. *Staattiset mittarit*, kuten koodirivien määrä tai löytyneiden puutteiden lukumäärä, säilyvät muuttumattomina ajan suhteen (Humphrey 1995).

Ennustavat mittarit voidaan generoida etukäteen, kun taas *selittävät mittarit* saavat arvonsa vasta mittauksen jälkeen. *Absoluuttiset mittarit* säilyttävät arvonsa, vaikka uusia asioita lisättäisiin. Esimerkiksi ohjelman kokoon eivät vaikuta toisten ohjelmien koot, mutta *suhteellisiin mittareihin,* kuten kokojen keskiarvoon vaikuttavat muidenkin ohjelmien koot (Humphrey 1995).

Organisaation johto haluaa usein mitata ja ennustaa juuri ulkoisia attribuutteja epäsuorilla mittareilla, sillä esimerkiksi toimintojen kustannustehokkuus tai henkilöstön tuottavuus auttavat varmistamaan, että laatu pysyy korkealla ja että hinta pysyy samalla alhaisena. Myös käyttäjät ovat kiinnostuneita ulkoisista attribuuteista, sillä järjestelmän toiminta vaikuttaa suoraan heihin. Järjestelmän luotettavuus, käytettävyys ja siirrettävyys vaikuttavat

ylläpito- ja ostopäätöksiin. Ulkoiset attribuutit ovat kuitenkin useimmiten vaikeampia mitata kuin sisäiset, ja lisäksi niitä voidaan mitata vasta tuotantoprosessin myöhäisessä vaiheessa. Esimerkiksi ohjelmiston luotettavuutta ei voida mitata ennen kuin tuotantoprosessi on päättynyt ja tuote on valmis (Fenton ja Pfleeger 1997).

Taulukko 1. Ohjelmistomittauksen kohteita.

KOHDE	**ATTRIBUUTIT**	
	Sisäiset	**Ulkoiset**
Tuotteet		
Määrittelyt	Koko, uudelleenkäyttö, modulaarisuus, funktionaalisuus…	Käsitettävyys, ylläpidettävyys…
Suunnitelmat	Koko, uudelleenkäyttö, modulaarisuus, kytkennät, kiinteys, periytyvyys, toiminnallisuus…	Laatu, monimutkaisuus, ylläpidettävyys…
Koodi	Koko, uudelleenkäyttö, modulaarisuus, kytkennät, toiminnallisuus, algoritminen monimutkaisuus, ohjausvuon monimutkaisuus…	Luotettavuus, käytettävyys, ylläpidettävyys, uudelleenkäytettävyys…
Testiaineisto	Koko, kattavuus…	Laatu, uudelleenkäytettävyys…
…	**…**	**…**
Prosessit		
Määrittelyn tekeminen	Aika, työmäärä, muuttuneiden vaatimusten lkm…	Laatu, kustannus, stabiilius…
Yksityiskohtainen suunnittelu	Aika, työmäärä, löytyneiden määrittelyvirheiden lkm…	Kustannus, kustannustehokkuus…
Testaus	Aika, työmäärä, ohjelmointivirheiden lkm…	Kustannus, kustannustehokkuus, stabiilius…
…	**…**	**…**
Resurssit		
Henkilöstö	Ikä, hinta…	Tuottavuus, kokemus, älykkyys…
Tiimit	Koko, kommunikointitaso, rakenteellisuus…	Tuottavuus, laatu…
Organisaatiot	Koko, ISO-sertifikaatti, kypsyystaso…	Kypsyys, kannattavuus…
Ohjelmistot	Hinta, koko…	Käytettävyys, luotettavuus…
Laitteistot	Hinta, nopeus, muistin määrä….	Luotettavuus…
Toimistot	Koko, lämpötila, valaistus…	Mukavuus, laatu…
…	**…**	**…**

Joskus attribuuttien määrittäminen on vaikeaa sellaisella mitattavalla tavalla, josta kaikki ovat samaa mieltä. Esimerkiksi laadun määritteleminen tyhjentävästi yhdellä tavalla on vaikeaa. Tämän vuoksi näitä vaikeasti määritettäviä laajoja attribuutteja määritellään muiden mitattavissa olevien attribuuttien avulla, kuten laadun osalta esimerkiksi kuvan 1 mukaisesti (Fenton ja Pfleeger 1997).

Monissa tapauksissa ohjelmiston kehittäjät ja käyttäjät keskittyvät vain yhden laajan attribuutin osaan. Esimerkiksi jotkut mittaavat laatua (ulkoinen tuoteattribuuttia) formaalin testauksen aikana löydettyjen virheiden lukumäärällä (sisäinen prosessiattribuutti). Käyttämällä sisäisiä attribuutteja ulkoisten attribuuttien arvioimiseen voidaan joskus tehdä harhaanjohtavia johtopäätöksiä. Yksi ohjelmistomittauksen tutkimuksen tavoite onkin tunnistaa sisäisten ja ulkoisten attribuuttien välisiä suhteita, kuten myös löytää uusia ja käyttökelpoisia menetelmiä intressien mukaisten attribuuttien mittaamiseksi suoraan. Taulukossa 1 on esitetty yhteenveto ohjelmistomittauksen kohteista attribuutteineen (Fenton ja Pfleeger 1997, Fenton ja Neil 2000).

2.4.2 Tuotemittarit

Mitä tahansa ohjelmiston elinkaaren aikana syntyvää tuotosta voidaan mitata ja arvioida. Tällaisia mitattavia kohteita ovat itse tuotteen ja sen dokumentaation lisäksi esimerkiksi projektin aikana luodut prototyypit ja testausta avustamaan luodut apuvälineet. Tuotoksia mitataan usein mm. koon ja laadun määrittämiseksi (Fenton ja Pfleeger 1997).

Tuotemittarit voidaan jakaa sisäisiä ja ulkoisia attribuutteja mittaaviin mittareihin. Koska ulkoiset tuoteattribuutit riippuvat sekä tuotteen käyttäytymisestä että ympäristöstä, pitää ulkoisia mittareita käytettäessä ottaa nämä seikat huomioon. Jos halutaan tutkia esimerkiksi laadun attribuuttina ohjelmankoodin luotettavuutta, täytyy ensiksi ottaa huomioon tietokone, jossa ohjelmaa käytetään ja toiseksi on huomioitava ohjelman käytön muoto. Esimerkiksi henkilöllä, joka käyttää tekstinkäsittelyohjelmaa vain tekstin kirjoittamiseen, voi olla aivan erilainen käsitys ohjelman luotettavuudesta kuin henkilöllä, joka käyttää samaa ohjelmaa taulukoiden käsittelyyn ja linkittämiseen taulukkolaskentaan. Vastaavasti järjestelmän ylläpidettävyys voi olla riippuvainen ylläpitäjien taidoista ja käytössä olevista työvälineistä (Fenton ja Pfleeger 1997).

Muita ulkoisia laadun attribuutteja mitattaessa on otettava huomioon, että nämä attribuutit eivät kuvaa pelkästään ohjelmakoodia, vaan myös niitä dokumentteja, jotka tukevat kehitystyötä. Suunnitelmien ja määrittelyjen ylläpidettävyys, uudelleenkäytettävyys ja jopa testattavuus ovat yhtä tärkeitä kuin itse ohjelmakoodi (Fenton ja Pfleeger 1997).

Sisäisiä tuoteattribuutteja on periaatteessa helppo mitata. Tuotteen koko voidaan määrittää esimerkiksi koodirivien, sivujen tai käytettyjen sanojen lukumäärällä. Koska tuotteet ovat konkreettisia, voidaan koko helposti ymmärtää attribuuttina. Monet muut sisäiset attribuutit ovat vaikeammin mitattavissa, sillä niiden tarkoituksesta ja mittaustavasta ollaan usein eri mieltä. Esimerkiksi tuotteen *monimutkaisuus* (complexity) on yksi tällainen attribuutti (Fenton ja Pfleeger 1997).

Koska tuotteita on suhteellisen helppo tutkia automaattisesti, on pystytty määrittämään joukko yleisesti käytettäviä sisäisiä attribuutteja. Esimerkiksi määrityksiä voidaan arvioida niiden pituuden, toiminnallisuuden, modulaarisuuden, uudelleenkäytettävyyden, redundanssin ja syntaktisen oikeellisuuden perusteella. Formaaleja suunnitelmia ja ohjelmakoodia voidaan arvioida lisäksi esim. moduulien kytkentöjen ja kiinteyden sekä ohjausvuon monimutkaisuuden suhteen (Fenton ja Pfleeger 1997).

Käyttäjät hylkäävät usein sisäiset attribuutit, sillä he ovat lähinnä kiinnostuneita ohjelman perustoiminnallisuudesta, laadusta ja käytöstä. Kuitenkin sisäiset attribuutit auttavat määrittämään mitä ulkoisia attribuutteja voidaan mahdollisesti löytää (Fenton ja Pfleeger 1997). Esimerkiksi ihmisen tekemät virheet aiheuttavat puutteita dokumenteissa ja muissa tuotoksissa, ja voivat johtaa *häiriöön* eli *epäonnistumiseen* (failure) ohjelmistotuotetta käytettäessä. Näin ollen tuotoksista löytyvät puutteet voivat implikoida tuotteen luotettavuutta. Tarkastusprosessia voidaan muuttaa löydettyjen puutteiden lukumäärän perusteella, vaikka luotettavuus määräytyykin häiriöiden perusteella.

Tuotteet voivat koostua alituotteista. Esimerkiksi ohjelmistojärjestelmän suunnitelma voi muodostua useista moduulien suunnitelmista. Moduulien keskimääräinen koko, joka on siis järjestelmän suunnitelman attribuutti, voidaan muodostaa laskemalla kunkin moduulin koko ja jakamalla saatu summa moduulien lukumäärällä (Fenton ja Pfleeger 1997).

Fentonin ja Pfleegerin (1997) mukaan ohjelmien rakennetta kuvataan usein sisäisten attribuuttien, kuten *modulaarisuuden*, *kytkentöjen* ja *kiinteyden* tason, avulla. Usein oletetaan, että hyvä sisäinen rakenne johtaa hyvään ulkoiseen laatuun. Vaikka tämä kuulostaakin intuitiivisesti houkuttelevalta, ei yhteyttä sisäisten ja ulkoisten tuoteattribuuttien välille useinkaan ole muodostettu, koska kontrolloitujen kokeiden suorittaminen attribuuttien välisten yhteyksien vahvistamiseksi on vaikeaa. Ohjelmiston mittaaminen voi kuitenkin auttaa ymmärtämään ja vahvistamaan suhteet empiirisesti. Onnistumisen edellytyksenä on tarkat ja merkitykselliset mittarit sisäisistä attribuuteista. Lisäksi pitää ymmärtää hyvin mittauksen validoinnin merkitys ja pystyä tulkitsemaan oikein mittauksen tulokset.

2.4.3 Prosessimittarit

Mittaamista voidaan suorittaa kaikissa ohjelmistotuotannon tuotanto- ja tukiprosesseissa. Mittaamisen avulla voidaan vastata toimintoihin ja prosesseihin liittyviin kysymyksiin. Sen avulla voidaan selvittää esim. kuinka kauan prosessin suorittaminen kestää, paljonko se maksaa, onko prosessi tehokas vai tehoton ja kuinka hyvä se on verrattuna muihin vastaaviin prosesseihin. Kuitenkin vain muutamaa sisäistä prosessiattribuuttia voidaan mitata suoraan. Tällaisia attribuutteja ovat prosessin tai siihen liittyvien toimintojen kesto, prosessiin tai sen toimintoihin liittyvä työmäärä ja prosessin tai sen toiminnon aikana tapahtuvien määritellyn tyyppisten tapahtumien lukumäärä (Fenton ja Pfleeger 1997).

Esimerkkinä Fenton ja Pfleeger (1997) mainitsevat vaatimusten tarkastamisen ennen kuin ne annetaan suunnittelijoille. Tarkastusmenettelyn tehokkuuden mittaamiseksi voidaan mitata löytyneiden virheellisten vaatimusten määrää. Vastaavasti voidaan mitata integrointitestauksen aikana löydettyjen virheiden lukumäärää määrittämään, kuinka hyvin ollaan toimittu.

Kuten tuotteet voivat muodostua alituotteista, voivat prosessit koostua aliprosesseista. Joissakin tapauksissa halutaan mitata monista erillisistä aliprosesseista koostuvan prosessin ominaisuuksia. Tällainen prosessi on esimerkiksi testaus. Se muodostuu usein yksikkö-, integrointi-, systeemi- ja hyväksymistestauksesta. Prosessin jokaista komponenttia voidaan mitata ja siten voidaan määrittää, kuinka eri komponentit vaikuttavat kokonaisprosessin tehokkuuteen. Kussakin aliprosessissa löytyneiden virheiden määrän ja kunkin virheen

löytymiseen tarvitun ajan ja kustannusten perusteella voidaan päätellä, onko aliprosessi kustannuksiin nähden tehokas (Fenton ja Pfleeger 1997).

Kustannusten lisäksi mitataan usein prosessien *ohjattavuutta* (controllability), *seurattavuutta* (observability) ja *stabiiliutta* (stability). Nämä ovat tärkeitä ulkoisia prosessiominaisuuksia etenkin suurissa projekteissa. Esimerkiksi suunnitteluprosessin stabiilius voi riippua tietystä ajankohdasta tai valituista suunnittelijoista. Tällaisten attribuuttien mittaaminen objektiivisilla mittareille ei useinkaan onnistu, sillä niitä ei ymmärretä riittävän hyvin. Tällöin käytetään subjektiivisia mittareita. Tällaisten subjektiivisten tai huonosti määriteltyjen tutkimusten avulla voidaan kuitenkin luoda pohja tuleville objektiivisille mittauksille (Fenton ja Pfleeger 1997).

2.4.4 Resurssimittarit

Resurssimittarit mittaavat kaikkia niitä resursseja, joita ohjelmistotuotannossa tarvitaan tuotteiden aikaansaamiseksi. Resursseja ovat esimerkiksi henkilöstö (yksilöt, tiimit), materiaalit (toimistotarvikkeet), työkalut (laitteistot, ohjelmistot) ja käytettävät menetelmät. Resursseja mitataan niiden määrän, kustannusten ja laadun määrittämiseksi. Tämä auttaa ymmärtämään ja kontrolloimaan prosesseja ilmaisemalla, kuinka kukin prosessi käyttää ja muuntaa resurssit tuotoksiksi. Esimerkiksi jos tuotetaan laadultaan huonoja ohjelmistoja, voi resurssimittareiden avulla selvitä, että huono laatu aiheutuu esim. työntekijöiden vähyydestä tai sellaisten henkilöiden käytöstä, joiden kyvyt eivät sovellu tähän työhön (Fenton ja Pfleeger 1997).

Kustannuksia mitataan yleensä kaikenlaisista resursseista, jotta vastuuhenkilöt näkevät, kuinka syötteiden kustannukset vaikuttavat tulosteiden kustannuksiin. Esimerkiksi voidaan tutkia, vaikuttaako investointi CASE-välineisiin henkilöstön tuottavuuteen parantavasti tai auttaako se parantamaan tuotteiden laatua. Koska kustannukset määräytyvät yleensä useista komponenteista, ollaan yleensä kiinnostuneita niistä komponenteista, joiden vaikutus kustannuksiin on suurin (Fenton ja Pfleeger 1997).

Tuottavuus on aina tärkeä seikka ja siksi sitä ollaan usein mittaamassa ja innokkaasti parantamassa. Henkilöstön tuottavuus on ulkoinen resurssiattribuutti, sillä se riippuu olemassa olevista kehitysprosesseista. Tästä syystä tuottavasta työntekijästä voi tulla tuottamaton,

jos prosessi muuttuu (pätee myös toisinpäin). Tuottavuutta kuvataan yleensä siten, että tuotantomäärä jaetaan työmäärällä. Tämän resurssimittarin tapauksessa tuotemittarin arvo jaetaan siis prosessimittarin arvolla. Tuottavuuden lisäksi voidaan myös mitata henkilöstön kokemusta, ikää ja älykkyyttä, joilla kaikilla voi olla vaikutusta tuotteen laatuun. Myös kehitysryhmän koko ja rakenne sekä käytettävien viestintämenetelmien tehokkuus ovat tärkeitä (Fenton ja Pfleeger 1997).

Työvälineitä ja -menetelmiä voidaan myös luokitella ja analysoida. Ohjelmointikielet voivat olla proseduraalisia tai oliopohjaisia, tekniikat voivat olla automaattisia tai manuaalisia ja työvälineet voivat vaatia koulutusta tai kokemusta. Näiden attribuuttien avulla voidaan arvioida ja määrittää, kuinka menetelmiä ja välineitä voidaan käyttää tehokkaasti (Fenton ja Pfleeger 1997).

Koska kenelläkään ei ole aikaa mitata kaikkia mitattavissa olevia attribuutteja, on tärkeää keskittää mittaustoimenpiteet sellaisiin kohteisiin, jotka tarvitsevat eniten näkyvyyttä, ymmärrystä ja kehitystä. Siksi mittauksen kohteet pitää valita valittujen tavoitteiden mukaisesti (Fenton ja Pfleeger 1997).

3 TAVOITEPOHJAINEN OHJELMISTOPROSESSIN MITTAAMINEN

Ohjelmistomittauksella tarkoitetaan jatkuvaa tuote- ja prosessitiedon määrittelyä, keräystä ja analysointia prosessien ja tuotteiden laadun ja valvonnan parantamiseksi (Solingen ja Berghout 1999). Ohjelmistomittausta varten voidaan luoda erityinen *mittausohjelma* (Grady ja Caswell 1987), jonka onnistumisen edellytykset on lueteltu taulukossa 2 (Wieczorek 1997).

Taulukko 2. Mittausohjelman onnistumisen edellytykset (Wieczorek 1997).

Organisatoriset edellytykset:	**Menetelmälliset edellytykset:**
- selkeästi määritellyt vastuualueet - kaikkien osapuolten sitoutuminen mittaukseen - asianmukaisesti suunnitellut resurssit - asianmukaisesti koulutetut mittausohjelmaan osallistujat	- tavoitepohjaisen mittaamisen menetelmällisten periaatteiden noudattaminen - mittaukseen asianmukaisesti valitut projektit - todellisten kehitysprosessien olemassa olevat kuvaukset - strategiset kehitystavoitteet liittyvät projektitavoitteisiin - palaute

Briandin et al. (1996) mukaan tavoitepohjaisen mittaamisen avulla voidaan varmistaa mittaussuunnitelman ja siten myös tiedonkeräämisen sopivuus, johdonmukaisuus ja täydellisyys. Mittausohjelman suunnittelijan on pystyttävä käsittelemään paljon tietoa ja lukuisia vuorovaikutussuhteita. Jotta suunnittelija voi varmistua mittausten sopivuudesta, johdonmukaisuudesta ja täydellisyydestä, on hänen tiedettävä tarkalleen, miksi valittuja attribuutteja mitataan (esim. kokoa mitataan, jotta voidaan ennustaa projektin kustannuksia projektin alkuvaiheessa), mihin olettamuksiin ne perustuvat (esim. ei koodin uudelleenkäyttöä) ja mihin malleihin mittauksia on tarkoitus käyttää.

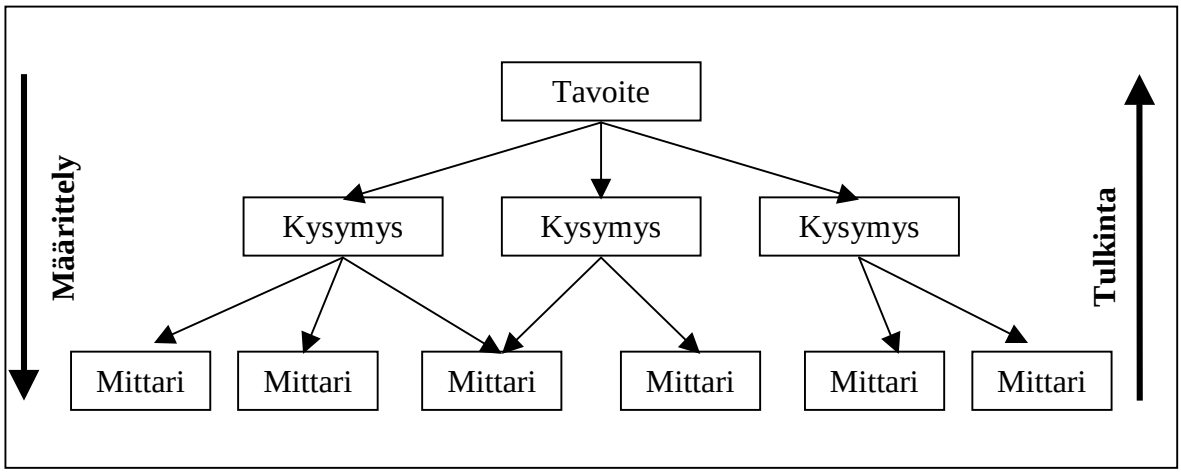
Tavoitepohjaisella mittaamisella voidaan paremmin hallita mittausohjelman monimutkaisuutta, kun valittavana on paljon erilaisia mitattavia attribuutteja ja kullekin niistä monia eri mittausasteikkoja. Lisäksi tapa, jolla mittaus suoritetaan, on riippuvainen asetetuista tavoitteista. Mittausohjelma ilman selvää tavoitteenmukaista rakennetta muuttuu nopeasti hallitsemattomaksi (Briand et al. 1996).

3.1 Tavoitepohjaisen mittaamisen malli

Tavoitepohjainen mittaaminen yhdistetään yleensä *GQM* (Goal/Question/Metric) –paradigmaan, jonka mukaisilla menetelmillä voidaan ohjelmistoprojekteista kerätä tietoa tavoitepohjaisesti (Basili et al. 1994b, The ami Handbook 1992).

Paradigma perustuu jäsentävään (top-down) lähestymistapaan, jolloin organisaation täytyy ensin määritellä itselleen ja projekteilleen tavoitteet, jotka se haluaa toteuttaa mittaamisen avulla. Tavoitteiden määrittelemisen jälkeen pitää tutkia, miten nuo tavoitteet voidaan saavuttaa eli mistä löytyy tarvittavat tiedot, joiden oletetaan määrittelevän asetetut tavoitteet. Tavoitteiden määrittelemisen lisäksi tarvitaan puitteet mittaustietojen tulkitsemiseksi ja käyttämiseksi ohjelmistoprosessin kehittämiseen.

GQM:n soveltamisen tuloksena saadaan mittausohjelman määrittely, joka kohdistuu tiettyihin mitattaviin asioihin ja sääntöihin mittaustulosten tulkitsemiseksi. Menetelmän käytöstä muodostuva GQM- eli mittausmalli (kuva 3) sisältää kolme tasoa. Koska jokainen organisaatio ja projekti ovat erilaisia, voi GQM-menetelmän soveltamisessa olla projekti- tai organisaatiokohtaisia eroja, jolloin myös muodostetut mittausmallit ovat erilaisia (Basili et al. 1994b).

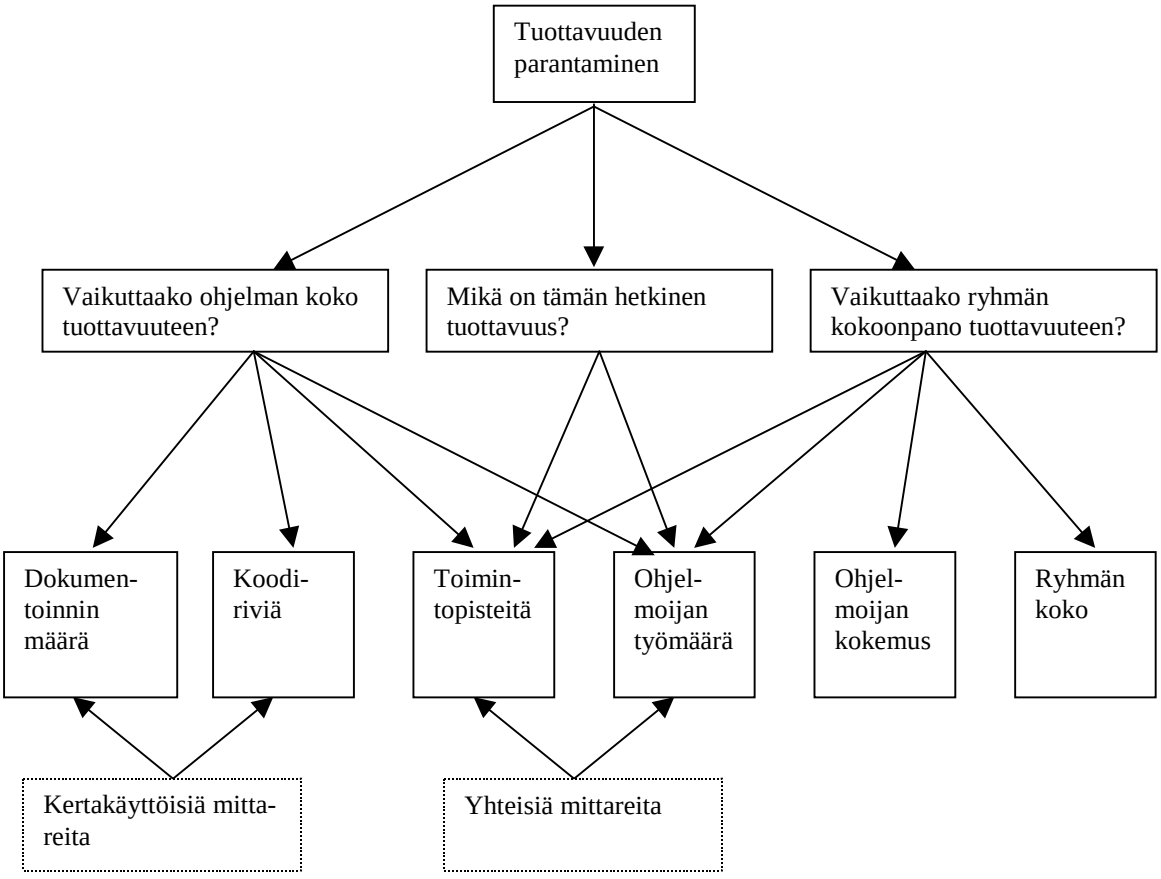


Kuva 3. Hierarkkinen GQM-malli (Basili et al. 1994b, Gray ja MacDonell 1997, Solingen ja Berghout 1999).

Käsitteellinen taso eli *tavoite* (goal) määrittelee mittauksen tarkoituksen, mittauskohteen, mitattavan asian ja näkökulman, josta mittaus suoritetaan. Mittauskohteita käsitteellisellä tasolla ovat tuotteet, prosessit ja resurssit.

Operatiivisella tasolla määritellään *kysymyksiä* (question), joita käytetään kuvaamaan tapaa, jonka avulla asetetut tavoitteet pyritään saavuttamaan tietyn kuvausmallin mukaisesti. Kysymykset pyrkivät kuvaamaan mittauskohdetta (tuote, prosessi, resurssi) valittuun laatutekijään nähden ja määrittelemään sen laatua tietystä näkökulmasta. Tavoitteesta muodostetaan useita kysymyksiä, jotka jakavat mitattavan asian komponenteiksi.

Kvantitatiivinen taso eli *mittari* (metric) sisältää informaation, joka vastaa kysymyksiin määrällisellä tavalla. Jokainen kysymys jalostetaan mittareiksi, jotka voivat olla joko objektiivisia tai subjektiivisia. Samaa mittaria voidaan käyttää myös saman tavoitteen alla oleviin muihin kysymyksiin. Eri GQM-malleilla voi olla myös samoja kysymyksiä ja mit-



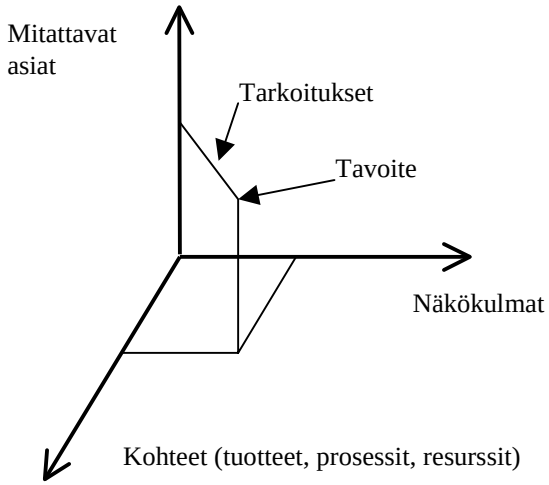
Kuva 4. Esimerkki GQM-mallista (Gray ja MacDonell 1997).

tareita, joiden avulla varmistetaan se, että eri näkökulmat otetaan mitattaessa huomioon (mittarilla voi olla eri arvo riippuen siitä, mistä näkökulmasta sen arvo otetaan).

Kuvassa 4 on esitetty esimerkki GQM-mallista. Esimerkissä tavoitteena on ohjelmistojen tuottavuuden parantaminen yrityksen näkökulmasta. Kuvassa 4 nähdään myös esimerkit *kertakäyttöisistä* (single use) ja *yhteisistä* (common) mittareista. Kertakäyttöisille mittareille voidaan myöhemmin lisätä muitakin tarkoituksia, mutta aluksi niillä mitataan vain yhtä kohdetta ja pyritään vastaamaan vain yhteen kysymykseen. Yhteisiä mittareita puolestaan käytetään useissa kysymyksissä ja malleissa, mutta tällöin niillä voi siis olla eri arvo riippuen näkökulmasta.

3.2 Tavoitteiden asettaminen

Tavoitteiden asettaminen on tärkeää GQM-menetelmän onnistuneen soveltamisen kannalta. Eri tavoitetyyppejä ovat *asiakastavoitteet*, *projektitavoitteet* ja *organisaatiotavoitteet*. Esimerkkejä asiakastavoitteista ovat asiakastyytyväisyys ja että tuotetussa ohjelmassa on vaaditut toiminnot. Esimerkkejä projektitavoitteista ovat ohjelman toimittaminen ajoissa ja laadukkaiden prosessien tarve. Organisaatiotavoitteisiin kuuluvat mm. parantuneet ohjelmistojen tuotantoprosessit ja myyvät ohjelmat (Basili et al. 1994b).



Kuva 5. Tavoitteen kolme koordinaattia (Basili et al. 1994b).

Basilin et al. (1994b) mukaan tavoitteen muodostamista voidaan havainnollistaa kolmen koordinaatin avulla kuvan 5 mukaisesti. Kuvan 4 esimerkin avulla ilmaistuna nämä koordinaatit on esitetty taulukossa 3.

Taulukko 3. Tavoitteen koordinaatit.

Koordinaatti	**Esimerkki**
Mitattava asia	Tuottavuus
Kohde	Ohjelmistot
Näkökulma	Yrityksen
Tarkoitus	Parantaa

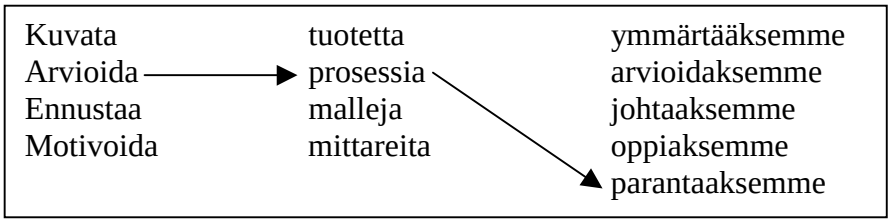
Kolmen ulottuvuuden vuoksi tavoitteen määrittäminen perustuu kolmeen tietolähteeseen. Ensimmäinen tietolähde koostuu organisaation menettelytavoista ja strategioista. Tutkimalla organisaation menettelytapoja ja suunnitelmia sekä haastattelemalla mittauksen kannalta olennaisia henkilöitä, saadaan selville *mitattava asia* ja mittauksen *tarkoitus*.

Toinen tietolähde sisältää organisaation tuotteiden ja prosessien kuvaukset. Näistä kuvauksista oleellisia ovat ainakin ne, joihin mittaamista on tarkoitus soveltaa. Jos siis halutaan arvioida jotain prosessia, tarvitaan kyseisen prosessin ja sen aliprosessien mallit. Muodostamalla prosessien ja tuotteiden mallit mahdollisimman tarkasti saadaan selville *mittauksen kohde*.

Kolmas tarvittava tietolähde on organisaatiomalli, josta selviää tavoitteen *näkökulma*. Koska kaikkia tuotteita ja prosesseja ei tarvitse tutkia kaikista mahdollisista näkökulmista, on ensin selvitettävä kuka tietoja tarvitsee, jolloin selviää näkökulma, josta mittaus suoritetaan. Tällä tavoin saadaan selville tavoitteiden määrittelyt, jotka ottavat huomioon organisaation rakenteen ja tavoitteet.

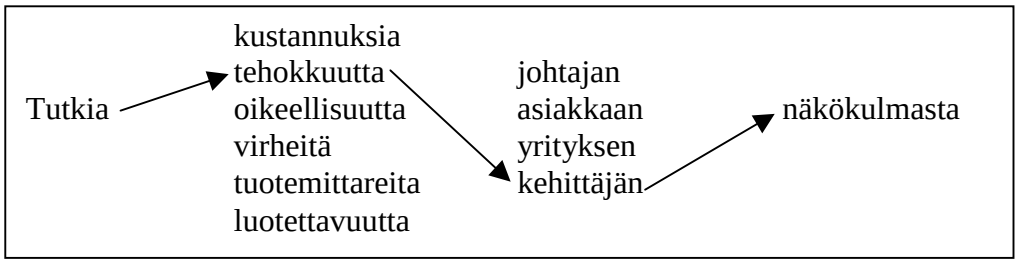
Tavoitteesta muodostetaan kysymyksiä, jotka luonnehtivat tavoitetta kvantitatiivisesti. Kysymysten määrittäminen asetettujen tavoitteiden mukaisesti ei ole helppoa, vaan siihen tarvitaan käytännön kokemusta asiasta. Tämän vuoksi on kehitetty tavoitteiden asettamista tukevia malleja ja ohjeistuksia kysymysten määrittämistä varten. Tällaisia malleja ja oh-

jeistuksia on kehitetty asiantuntijoiden kokemusten perusteella TAME-projektissa Marylandin yliopistossa (Basili ja Rombach 1988). Kuvassa 6 on kuvattu tavoitteen asettamismallia mittauksen *tarkoituksen* selvittämiseksi. Mallin avulla voidaan esimerkiksi arvioida ohjelman testausprosessia sen parantamiseksi.



Kuva 6. Tavoitteen asettamismalli: tarkoitus.

Kuvassa 7 on kuvattu tavoitteen asettamismallia tavoitteen *näkökulman* selvittämiseksi. Mallin avulla voidaan esimerkiksi tutkia tehokkuutta kehittäjän näkökulmasta.



Kuva 7. Tavoitteen asettamismalli: näkökulma.

3.3 Kysymysten asettaminen

Kysymykset voidaan jakaa tuote-, prosessi-, ja resurssipohjaisiin (Basili et al. 1994b), jolloin kysymysten avulla pyritään kuvaamaan joko tuotetta, prosessia tai resurssia. Prosessi- ja resurssipohjaiset kysymykset laaditaan vastaavasti kuin tuotepohjaiset kysymykset, joten seuraavassa esitetään ohjeita vain tuotepohjaisia kysymyksiä varten.

Jokaisella mitattavalla tuotteella on kolme osatavoitetta, jotka pitää selvittää (Basili ja Rombach 1988). Ensimmäinen osatavoite on *tuotteen määrittely*. Tähän kuuluvat tuotteen fyysiset attribuutit eli tuotteen määrällinen kuvaus fyysisten attribuuttien (koko, kompleksisuus jne.) avulla, kustannukset (työmäärä, tietokoneaika jne.), muutokset ja puutteet (vir-

heet, viat, parannukset, laajennukset jne.) ja konteksti eli asiakkaan ympäristön määrällinen arviointi.

Toinen osatavoite on *kohteen laatutekijöiden määrittely*. Tämä tarkoittaa sitä, että jokaista laatutekijää (esim. luettavuus ja käyttäjäystävällisyys) kohden valitaan kysymyksiä koskien käytettävien laatutekijöiden kvantitatiivisia määrittelyjä (laatumalleja). Kysymykset voivat koskea myös laatumallien sopivuutta kyseiseen projektiin, kerätyn tiedon oikeellisuutta, laatumallien tehokkuutta tai tulosten järkevyyttä eri näkökulmista.

Kolmas osatavoite on *palaute*. Tällöin valitaan kysymyksiä liittyen tuotteen parantamiseen laatutekijöitä ajatellen. Näihin kysymyksiin saadut vastaukset kertovat tuotteen laadusta, suurimmista ongelmista ja antavat parannusehdotuksia nykyisiin ja tuleviin projekteihin.

3.4 Mittareiden valinta

Mittarit vastaavat valittuihin kysymyksiin määrällisellä tavalla. Usein yhteen kysymykseen vastaamiseen tarvitaan useita mittareita, sekä objektiivisia että subjektiivisia. Mittareiden valintaan vaikuttavat seuraavat tekijät (Basili et al. 1994b):

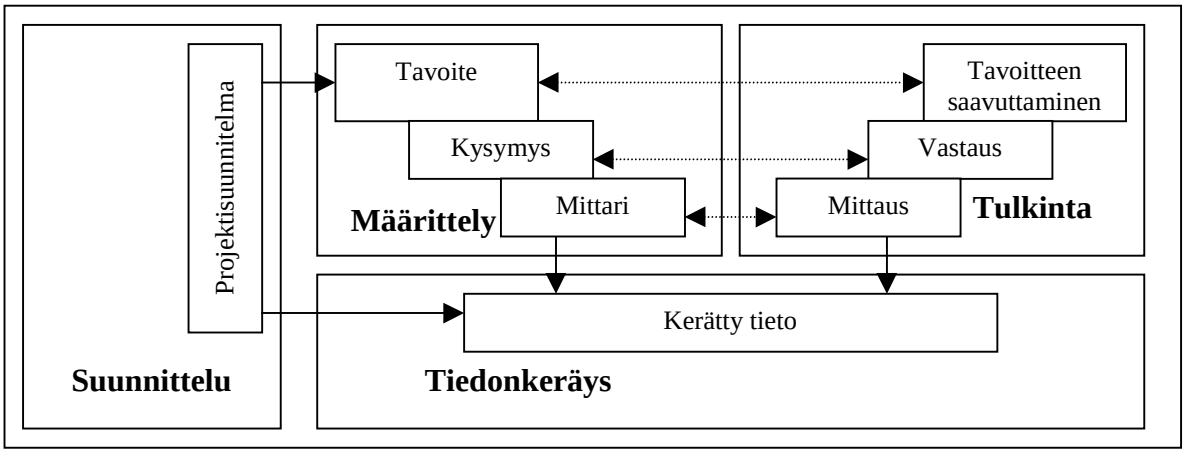
- *Olemassa olevan mittaustiedon määrä ja laatu:* jos tietoa on saatavana ja se on luotettavaa, sen käyttö pyritään maksimoimaan.
- *Mittauskohteen kypsyys:* objektiivisia mittareita käytetään kypsempien kohteiden arvioimiseen ja subjektiivisia mittareita käytetään, kun kyseessä on epävakaampi tai epämuodollisempi mittauskohde.
- *Oppimisprosessi*: koska GQM-mallia pitää hioa ja muuttaa aina tarvittaessa, pitää mittareiden auttaa arvioimaan sekä itse mittauksen kohdetta että arvioimiseen käytetyn mallin luotettavuutta.

Valittujen mittareiden perusteella valitaan sopivat mekanismit tiedon keräämistä ja tarkistamista varten. Kun GQM-malli on luotu ja tiedonkeräystekniikoista sekä –välineistä on sovittu, sovitetaan kerätty tieto luotuun malliin ja tulkitaan määriteltyjen käytäntöjen mukaisesti (Basili et al. 1994b). GQM-mallin avulla voidaan tulkita kerättyä tietoa tutkimalla luotua mallia kokoavasti alhaalta ylöspäin kuvan 3 mukaisesti. Kun mallia tutkitaan kokoavasti, löydetään kuhunkin mittariin liittyvät kysymykset ja lopulta tavoite, johon mittari

liittyy. Lisäksi tutkimalla mittareiden arvoja ja vertaamalla niitä tavoitteeseen, voidaan päätellä, onko tavoitteeseen päästy (Basili 1992).

3.5 Tavoitepohjainen mittausprosessi

GQM-prosessi voidaan jakaa kuvan 8 mukaisesti neljään osaan. Nämä vaiheet ovat suunnittelu, määrittely, tiedonkeräys ja tietojen tulkinta.



Kuva 8. GQM-menetelmän neljä eri vaihetta (Solingen ja Berghout 1999).

3.5.1 Suunnitteluvaihe

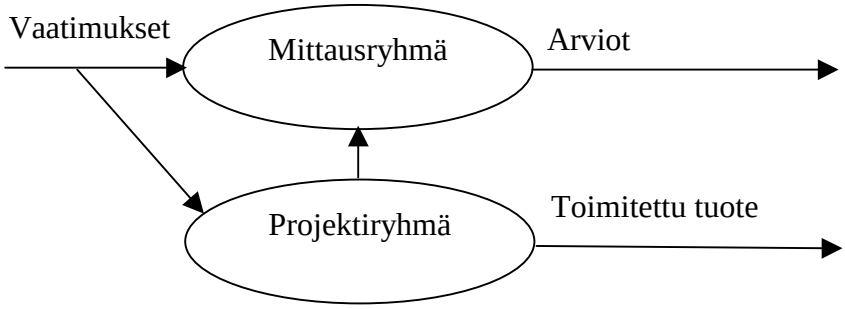
Suunnitteluvaiheen (planning phase) päätehtävä on luoda perusta onnistuneelle mittausohjelman esittelylle ja valmistella ja motivoida organisaation henkilöstöä mittausohjelmaan. Suunnitteluvaiheen aikana luotavaan projektisuunnitelmaan dokumentoidaan mittausohjelman käytännöt, aikataulut ja tavoitteet sekä projektihenkilöstön koulutussuunnitelma. Se tarjoaa myös perustan organisaation johdon hyväksynnälle. Suunnitteluvaihe koostuu seuraavista viidestä osavaiheesta eli askeleesta (Solingen ja Berghout 1999):

- mittausryhmän muodostaminen
- kehityskohteen valinta
- sovellusprojektin ja projektiryhmän asettaminen
- projektisuunnitelman luominen
- henkilöstön kouluttaminen.

Mittausryhmän muodostaminen

Basilin et al. (1994b) mukaan käytäntö on osoittanut, että itsenäinen mittausryhmä eli GQM-ryhmä on lähes välttämätön mittausohjelman jatkuvuuden kannalta, koska projektien määräaikojen lähestyessä mittaustoiminnot jäävät usein vähälle huomiolle ilman siitä vastuussa olevaa erillistä ryhmää. Latum et al. (1998) toteavat erillisen mittausryhmän ja projektiryhmän välisen luottamuksellisen ja tiiviin kanssakäymisen olevan mittausohjelman onnistumisen kannalta välttämätöntä.

DeMarco (1982) kuvaa erillisen mittausryhmän suhdetta projektiryhmään kuvan 9 mukaisesti. Mittausryhmän pitäisi siis olla riippumaton projektiryhmästä. Myös Grady ja Caswell (1987) korostavat vastuun kohdentamista organisaatiossa. Pienemmissä organisaatioissa tämä saattaa olla resurssien käytön kannalta vaikeampi toteuttaa kuin suurissa organisaatioissa. Pienissä organisaatioissa ei välttämättä kannata luoda erillistä mittausryhmää, vaan mittausryhmä voi olla ns. *looginen,* jolloin vastuu mittausryhmän toiminnoista on nimetyillä henkilöillä.



Kuva 9. Mittausryhmän suhde projektiryhmään (DeMarco 1982).

Luottamuksen ja yhteistyön tason säilyttämiseksi mittausryhmällä ei saa olla erityistä mielenkiintoa projektiryhmän keräämään mittaustietoon (Solingen ja Berghout 1999). Jotta mittausryhmä voi ohjata ja tukea mittausohjelmaa, pitää ryhmällä olla tarpeeksi taustatietoa mittaamisesta ja sen kohteina olevista prosesseista ja tuotteista. Tämä on tärkeä edellytys, koska mittausryhmän pitää pystyä keskustelemaan tulosten tulkinnoista projektiryhmän kanssa. Projektiryhmä kuitenkin ”omistaa” projektinsa, joten mittausryhmän pitää kunnioittaa projektiryhmän työskentelyä, vaikka se ei aina pystyisikään noudattamaan ennalta määrättyjä menettelytapoja.

Kehityskohteiden valinta

Kehityskohteen valinta voi perustua esimerkiksi tuotteissa tai prosesseissa havaittuihin ongelmiin, organisaation liiketoimintatavoitteisiin tai suoritettuun ohjelmistotuotannon arviointiin (Solingen ja Berghout 1999).

Kehityskohteeksi voidaan valita ongelma, jonka ohjelmistojen kehittäjä tai joku muu organisaation jäsen on tuotteessa tai prosessissa havainnut. Ongelmien löytämiseksi ja selvittämiseksi tarvitaan lisätietoa, jota saadaan haastattelemalla asianomaisia henkilöitä. Nämä havaitut ongelmat yleensä motivoivat kehittäjiä mittauksen suorittamiseen, sillä prosesseissa ja tuotteissa esiintyvät ongelmat liittyvät tai vaikuttavat heidän toimintaansa.

Kehityskohteiden tulee olla yrityksen kehitys- ja liiketoimintatavoitteisiin sidottuja. Park et al. (1996) tarkastelevatkin liiketoimintatavoitteiden jalostamista kehitystavoitteiksi ja mittaustavoitteiksi. Tällöin organisaation johdon asettamat liiketoimintatavoitteet jaetaan alitavoitteiksi, joista sitten johdetaan tavoitteita tukevia kysymyksiä ja mittareita. Yhdistämällä yrityksen kehitystavoitteet ja kuvassa 2 esitetyt neljä perusnäkökulmaa (kustannukset, aika, riskit ja laatu) saadaan kehitystavoitteita, kuten esimerkiksi asiakastyytyväisyyden parantaminen tai markkinaosuuden kasvattaminen (Solingen ja Berghout 1999). Grady ja Caswell (1987) korostavat johdon tuen riippuvan siitä, miten tavoitteet mittausohjelmalle annetaan.

Kehityskohteita voidaan löytää myös arviointikehyksiä soveltamalla (The ami Handbook 1992). Arvioinnin perusteella on tarkoitus saada selkeä kuva organisaation ohjelmistotuotannosta, sen ongelmista ja tehdä aloitteita sen kehittämiseksi (Solingen ja Berghout 1999).

Sovellusprojektin ja projektiryhmän asettaminen

Projektiryhmän rooli mittausohjelman onnistumisessa on suuri, sillä ryhmän jäsenet suorittavat mittaukset ja voivat joutua sopeuttamaan työprosessejaan mittauksen vuoksi (Solingen ja Berghout 1999). Projektiryhmän motivaation ylläpitäminen on siten tärkeää. Myös projektiryhmän kehitysideoiden ja mittaustavoitteiden linjaamiseen kannattaa panostaa. Mittausryhmän vastuulla on valvoa ja innostaa projektiryhmän panostusta mittaustoimintoja kohtaan.

Koska kehityskohde valitaan ennen projektiryhmän luomista, on mittausryhmällä merkittävä vaikutus mittausalueeseen. Mittausryhmä ei saa kuitenkaan olla kokonaan vastuussa mittausohjelmasta, koska kaikkien mittausten tulisi liittyä projektiryhmän tavoitteisiin. Jos projektiryhmä päättää lopettaa jonkun tietyn kohteen mittaamisen tai muuttaa mittauksen laajuutta, näin voidaan tehdä. Mittausryhmä voi kuitenkin vaikuttaa päätökseen, jos muutokset eivät ole mittausohjelman kannalta kannattavia.

Projektisuunnitelman laatiminen

Kun mittausryhmä ja projektiryhmä on asetettu ja kehityskohteet valittu, on aika tehdä projektisuunnitelma eli ehdotus mittausohjelmasta. Tämän projektisuunnitelman tekee pääosin mittausryhmä projektiryhmältä saatujen tietojen perusteella.

Suurin osa projektisuunnitelman sisällöstä muodostuu suunnitteluvaiheen aikana kerätyistä tiedoista. Solingenin ja Berghoutin (1999) mukaan tärkeää on myös se, että suunnitelmassa tuodaan esille organisaation johdon hyväksyntä projektille. Alustava kustannus-hyöty -analyysi perustuu arvioituihin mittaus- ja projektiryhmien työmääriin ja odotettuihin tuottoihin, jotka saadaan kehitystavoitteet saavuttamalla.

Kuten normaalistikin ohjelmistotyössä, pitää projektisuunnitelmaan tehdä realistisia suunnitelmia mittausohjelman tulevista vaiheista ja niiden ajankohdista. Suunnitelman tulisi myös ilmaista, kuinka paljon projektiryhmän jäsenet missäkin projektin vaiheessa ovat mukana. Lisäksi projektin virstanpylväät ja takarajat määritellään projektisuunnitelmassa, kuten myöskin lista kunkin vaiheen tuotoksista.

Projektisuunnitelman pääasiallinen tehtävä on toimia ehdotuksena organisaation johdolle ja saada johdon suostumus ja sitoutuminen projektiin. Kun suunnitelma on hyväksytty, sitä ylläpidetään mittausryhmän toimesta. Projektisuunnitelma todennäköisesti muuttuu ajan kuluessa, kun enemmän tietoa on saatavilla.

Henkilöstön koulutus

Suunnitteluvaiheen viimeisenä askeleena on projektiin osallistuvien henkilöiden koulutus (Solingen ja Berghout 1999). Koulutus on tärkeää, jotta projektiin osallistuvat henkilöt

ovat omistautuneita ja motivoituneita ja jotta heidän omistautuneisuutensa projektin tavoitteita kohtaan säilyy. Tämä voidaan saavuttaa pitämällä säännöllisesti koulutus- ja edistymistilaisuuksia.

Gradyn ja Caswellin mukaan (1987) mukaan tiedonkeruun tarkkuus riippuu henkilöstön sitoutumisesta. Henkilöstön pitää olla vakuuttuneita hankkeen tärkeydestä (The ami Handbook 1992). Koulutustilaisuuksilla voidaan vahvistaa liiketoiminnan ja projektin tavoitteiden yhteensopivuutta, jos myös johdon edustajat ovat paikalla.

GQM-menetelmän soveltaminen edellyttää, että projektiin osallistuvat hallitsevat sen. Tämän mittausryhmä voi organisoida pitämällä koulutusta mittauksen perusteista, GQM-paradigmasta ja GQM-menetelmästä käytännössä. Projektiryhmälle pitää selvittää, millaisia mittaustehtäviä he joutuvat tekemään, miksi he niitä tekevät, milloin ja miten tehtäviä suoritetaan, kuinka suuri työmäärä niihin liittyy, kuinka ne vaikuttavat heidän päivittäisiin rutiineihinsa ja mitä he hyötyvät ja oppivat mittauksesta.

3.5.2 Määrittelyvaihe

Määrittelyvaiheessa (definition phase) määritellään ja dokumentoidaan mittausohjelma formaalisti. Mittausohjelma sisältää tavoitteet, kysymykset, mittarit ja hypoteesit. Määrittelyvaiheessa luodaan GQM-, mittaus- ja analysointisuunnitelmat seuraavin askelin (Solingen ja Berghout 1999):

- mittaustavoitteiden määrittely
- ohjelmistoprosessimallien katselmus tai tuottaminen
- haastattelujen suorittaminen
- kysymysten ja hypoteesien määrittely ja tarkastaminen
- mittareiden määrittely ja eheyden ja johdonmukaisuuden tarkastaminen
- GQM-suunnitelman tuottaminen
- mittaussuunnitelman tuottaminen
- analysointisuunnitelman tuottaminen
- suunnitelmien tarkastaminen.

Mittaustavoitteiden määrittely

Mittaustavoitteiden määrittäminen tapahtuu suunnitteluvaiheessa asetettujen kehitystavoitteiden perusteella (Solingen ja Berghout 1999). Kaikkien mittausohjelmaan osallistuvien pitäisi ottaa osaa tavoitteiden määrittämiseen, jotta kaikille pysyisivät selvänä syyt, miksi mittausta tarvitaan. Koska mittaustavoitteet tulee määritellä ymmärrettävällä tavalla, on tätä varten olemassa kaavaimia, joiden avulla voidaan kuvien 6 ja 7 esimerkkien mukaisesti määrittää mittaustavoitteen tarkoitus, näkökulma ja konteksti. Kuvassa 10 on Basilin et al. (1994a) esittämä kaavain tavoitteiden määrittelyä varten. Kuvassa 11 on esimerkki erään projektin mittaustavoitteen määrittelystä.

Analysoi	mitattava kohde
Tarkoituksena	ymmärtää, valvoa tai kehittää kohdetta
Suhteessa	mittauksen kohteena olevan kohteen laatuun
Näkökulmana	ihmiset, jotka mittaavat kohdetta
Kontekstina	ympäristö, jossa mittaus tapahtuu

Kuva 10. Tavoitteen määrittelykaavain.

Tavoite ”luotettavuus”:

Analysoi toimitettu tuote ja kehitysprosessi tarkoituksena parempi ymmärtäminen suhteessa luotettavuuteen ja sen syihin näkökulmana ohjelmiston kehittämisryhmä kontekstina projekti XYZ

Kuva 11. Esimerkki tavoitteen määrittelystä.

Jos projektiryhmällä ei ole kokemusta tavoitteiden asettamisesta, voi mittausryhmä tehdä sen yksin ja esittää kehitystavoitteista johdetut mittaustavoitteet *katselmusten* (review) aikana muille projektiryhmän jäsenille. Kaikkia mittaustavoitteita ei kannata aina ottaa mukaan mittausohjelmaan, vaan niistä joudutaan valitsemaan osa. Tavoitteiden priorisointi ja valinta niiden välillä ei välttämättä ole helppoa. Tavoitteet pitää myös validoida ristiriitaisuuksien löytämiseksi (The ami Handbook 1992). Organisaation pitää myös pystyä esittä-

mään, kuinka mittaustavoitteet liittyvät liiketoimintatavoitteisiin ja tukevat liiketoiminnan kehittymistä.

Ohjelmistoprosessimallien katselmus tai tuottaminen

Ohjelmistoprosessien malleja tarvitaan tukemaan mittausohjelman mittareiden määrittämistä (Solingen ja Berghout 1999). Jos nämä prosessimallit on jo määritelty organisaatiossa, pitää ne käydä läpi ja tarvittaessa niitä pitää kehittää tukemaan mittausten määrittämistä. Mahdollisia tarkastustapoja ovat *formaalit katselmukset*, *aivoriihet*, *haastattelut* ja *esittelyt*. Kaikkien mittausohjelmaan osallistuvien tulisi olla yhtä mieltä prosessimalleista. Lisäksi prosessimallien tulisi kuvata prosessien todellista tilaa eli sitä, kuinka työt todellisuudessa tehdään, eikä siis sitä, kuinka ne tulisi tehdä. Jos asianmukaisia prosessi- tai tuotemalleja ei ole määritelty, tulee mittausryhmän kehittää ne. Kun kaikki olennaiset prosessimallit on luotu, tulee projektiryhmän hyväksyä ne. Kuvassa 12 on eritelty esimerkki kehitysprosessin toteutustoiminnon määrittelystä. Periaate on Humphreyn (1989) mukainen.

Pfleeger (1995) esittää artikkelissaan lähestymistavan, jossa otetaan huomioon CMM:n mukaiset prosessien kypsyystasot sekä organisaation ohjelmistotuotanto- ja ylläpitoprosessien mallit. Prosessien kypsyystasojen avulla huolehditaan, että jokainen mitattava kohde on näkyvä eli sitä voidaan mitata. Ohjelmistomittaus ja kypsyystasot liittyvät läheisesti toisiinsa; kehittäjä voi mitata vain sen mikä prosessissa on näkyvää. Prosessien kypsyystasojen avulla voidaan ohjelmistomittauksessa selvittää, mitä mitataan mittausohjelman alkuvaiheessa ja mitä voidaan myöhemmin mitata kokemuksen ja resurssien lisääntyessä. Tuotanto- ja ylläpitoprosessien malleja käytetään kuvaamaan kuka on vastuussa mittauksen suorittamisesta, milloin tietoa kerätään, kuinka usein mittaus suoritetaan, mistä ja miten tietoa kerätään. Prosessimallien tulisi sisältää seuraavat tiedot:

- kaikki mitattavat prosessit ja niihin liittyvät toiminnot
- kaikki mitattavat tuotteet ja resurssit
- kaikkien mitattavien prosessien syötteet ja tuotokset
- kaikki prosesseihin liittyvät rajoitteet, kuten budjetit, aikataulut ja standardit
- kussakin prosessissa käytettävät resurssit, kuten laitteistot, ohjelmistot ja henkilöstö.

Kun käytettävät mallit on luotu, voidaan niiden perusteella määrittää milloin ja miten (esim. käsin vai jollakin työvälineellä) kukin mittaus suoritetaan ja kuka sen suorittaa (Pfleeger 1995). Jako tuotanto- ja ylläpitoprosesseihin on järkevä, koska mittarien arvojen laskennassakin joudutaan joskus huomioimaan tämä prosessijako. Esimerkiksi toiminto-pistemittari voidaan toteuttaa jaon mukaisesti (Dreger 1989, Garmus ja Herron 2001, Kröger 1999).

Aloituskriteerit:
- vaaditut yksityiskohtaiset suunnitelmat ovat valmiit ja hyväksytty
- toteutussuunnitelmadokumentti on valmis ja hyväksytty

Lopetuskriteerit:
- koodi on valmis
- koodaus on suoritettu standardin mukaisesti

Tehtävät:
- koodaus
- moduulien toteutus
- koodin dokumentointi
- koodin katselmus

Mittaukset:
- toimintopisteet
- työmäärä tehtävittäin ja henkilöittäin
- kalenterikesto
- havaitut puutteet (lkm, aiheuttaja, komponentti)
- korjatut puutteet
- puutteiden korjaukseen käytetty aika

Kuva 12. Esimerkki prosessin kuvauksesta.

Haastattelujen suorittaminen

Projektiryhmän pitäisi olla koko ajan läheisesti mukana mittausohjelman suunnittelussa (Solingen ja Berghout 1999). Erityisen tärkeää on heidän osallistumisensa tavoitteiden, kysymysten, mittareiden ja hypoteesien määrittämiseen. Projektiryhmän jäsenet tietävät eniten mitattavista kohteista, joten määrityksissä tarvittavat tiedot tulee saada heiltä.

Tämän tiedon saamiseksi haastatellaan ryhmän jäseniä yksitellen. Näillä haastatteluilla pyritään saamaan projektiryhmältä mittaustavoitteisiin liittyviä määrityksiä, käsityksiä ja

malleja, ja pyritään muuttamaan projektiryhmän implisiittinen tieto eksplisiittiseksi. Vaikka useamman henkilön yhtäaikainen haastatteleminen voi tuntua tehokkaammalta, on Solingenin ja Berghoutin (1999) mukaan parempi suorittaa haastattelut yksitellen, koska näin saadaan tieto kerättyä ilman, että siihen vaikuttaisivat muiden läsnäolevien mielipiteet. Jos paikalla on useampia henkilöitä, ei välttämättä saada kaikkia tietoja eikä etenkään henkilön rehellisiä mielipiteitä.

Haastattelujen apuna voidaan käyttää *abstraktiolomakkeita (abstraction sheet)*, joiden käyttö tarjoaa tavan keskittyä tavoitteen kannalta olennaisiin seikkoihin ja estää olennaisten asioiden huomiotta jättämisen (Solingen ja Berghout 1999). Abstraktiolomakkeiden avulla voidaan haastattelujen tulokset kirjata ylös yhdenmukaisella tavalla. Abstraktiolomake esittää tavoitteen pääkohdat ja riippuvuudet GQM-suunnitelman mukaisesti ja jakautuu neljään osioon:

- *Laadun kohdennus* (quality focus): mitä mittareita voidaan projektiryhmän mukaan käyttää tavoitteen kohteen mittaukseen?
- *Perushypoteesi* (baseline hypothesis): mikä on ryhmän jäsenen tämänhetkinen tietämys näistä mittareista? Ryhmän jäsenen odotukset dokumentoidaan mittareiden perushypoteeseiksi.
- *Vaihtelutekijät* (variation factors): mitkä ympäristötekijät vaikuttavat ryhmän jäsenen mukaan mittareihin?
- *Vaihtelutekijöiden vaikutus perushypoteesiin*: kuinka vaihtelutekijät mahdollisesti vaikuttavat mittauksiin? Millaisia riippuvuuksia mittareiden ja vaihtelutekijöiden välillä mahdollisesti on?

Solingenin ja Berghoutin (1999) mukaan mittausryhmä voi käyttää abstraktiolomakkeita monin tavoin. Lomake voidaan täyttää yhdessä projektiryhmän jäsenen kanssa keskustelemalla ensin laadun kohdennuksesta ja perushypoteesista ja niiden jälkeen vaihtelutekijöistä ja niiden vaikutuksista. Kun kaikki osiot on täytetty, käydään lomaketta läpi iteroiden, kunnes lomake on hyväksyttävästi täytetty. Toinen tapa on kouluttaa projektin jäsenet lomakkeiden täyttöön ja antaa heidän täyttää ne itse. Tämä vaatii hieman investointeja, koska kouluttaminen lomakkeiden täyttöön ei ole helppoa.

Mittausryhmä voi täyttää lomakkeet myös etukäteen ennen haastatteluja. Näin ryhmä valmistelee haastattelun ja tuo esille vedoksen abstraktiolomakkeesta. Tällöin pitää kuitenkin olla varovainen, sillä tässä tapauksessa haastattelija tuo itse asiassa omat implisiittiset mallinsa esille. Haastattelu on siten haastattelijan tekemän hahmotelman validointia. Jotta tätä lähestymistapaa voidaan käyttää, tarvitaan tietoa mittaustavoitteen kontekstista ja aiheesta.

Haastattelujen tulokset voidaan järjestää rakenteisesti ja kopioida GQM-suunnitelmaan abstraktiolomakkeilta. Niitä voidaan käyttää myös mittaustiedon analysointiin, esitykseen ja tulkintaan palautekeskustelujen aikana. Periaatteessa abstraktiolomake on yhden sivun tiivistelmä GQM-suunnitelmasta. Kaikkia GQM-suunnitelmassa mainittuja suoria mittauksia ei esitellä abstraktiolomakkeilla, vaan vain perusmittaukset, jotka vaikuttavat tärkeimpiin mittareihin.

Kuvassa 13 on esimerkki abstraktiolomakkeesta. Lomakkeen osien sisällön pitää olla suhteessa toisiinsa siten, että jokaista laadun kohdennusta vastaa ainakin perushypoteesi ja mahdollisesti joitakin vaihtelutekijöitä. Lisäksi jokaista vaihtelutekijää vastaa sen vaikutukset perushypoteeseihin. Nämä suhteet voidaan tarkastaa johdonmukaisuuden ja eheyden osalta, koska keskinäiset suhteet osioiden välillä ovat olemassa.

Kysymysten ja hypoteesien määrittely

Kysymysten avulla pyritään helpottamaan tiedon tulkintaa mittaustavoitteiden mukaisesti (Solingen ja Berghout 1999). Kysymykset täsmentävät asetettuja abstrakteja tavoitteita käytännöllisemmälle tasolle, joten kysymyksiä asetettaessa pitää tarkastaa, että niiden avulla varmasti voidaan tehdä johtopäätöksiä tavoitteen saavuttamisesta.

Kysymysten asettamisessa pitää huolehtia siitä, ettei kysymyksistä tule liian abstrakteja eikä myöskään liian yksityiskohtaisia, koska näissä tapauksissa ei kerätyn tiedon tulkinta kysymysten avulla onnistu. Jotta kysymyksistä tulisi sopivia, kerätään projektiryhmän haastatteluissa muodostamat kysymykset ja ryhmitellään samanaiheiset kysymykset yhteen. Koska useimmiten haastatteluista saatavat kysymykset ovat joko liian abstrakteja tai yksityiskohtaisia, saadaan kysymysten ryhmittelyllä selville, millaisia kysymysten tulee olla.

Kohde:	**Tarkoitus:**	**Laadun kohdennus:**	**Näkökulma:**	**Konteksti:**
Tuote ja prosessi	Parempi ymmärtäminen	Luotettavuus ja sen syyt	Ohjelmiston kehitysryhmä	Projekti XYZ

Laadun kohdennus

Puutteiden lukumäärä:
- kokonaismäärä
- vakavuusasteen mukaan (vähäinen, vakava kriittinen)
- elinkaaritoiminnoittain
- komponenteittain
- havaintoryhmittäin (kehittäjä, käyttäjä, ohjaus)

Kustannus (työmäärä tunteina):
- kokonaismäärä
- elinkaaritoiminnoittain
- komponenteittain
- puutteiden korjaus

Vaihtelutekijät

Prosessin noudattaminen
- katselmusten aste

Kohdealueen tuntemus
- kehittäjien kokemus

Attribuutit
- koodausstandardien noudattaminen
- dokumenttien ristiriidattomuus

Perushypoteesit (estimaatit)

Puutteiden jakauma vakavuusasteittain:
- vähäinen 60%
- vakava 30%
- kriittinen 10%

Puutteiden jakauma havaintoryhmittäin:

	Kehittäminen	Käyttöönotto	Käyttö
- kehittäjä	90%	5%	0%
- käyttäjä	5%	90%	100%
- ohjaus	5%	5%	0%

TOP 10 puutteellisinta komponenttia (järjestyksessä): …
TOP 10 työläintä korjattua komponenttia (järjestyksessä): …

Puutejakauma aiheuttajan mukaan:

	VM		SUUN		TOT …	
	#	%	#	%	#	%
- VM	2	5	0	0	5	12,5
- SUUN	3	7,5	0	0	6	15
- TOT	1	2,5	2	5	2	5
- …						

Työmäärän jakauma:
- puutteet / toimintopiste 0.1
- puutteiden korjaus / toimintopiste 1.0

Vaihtelutekijöiden vaikutus hypoteeseihin

- paremman prosessin valvonnan seurauksena vähemmän puutteita
- liian tiukat määräajat lisäävät puutteiden lukumäärää
- kokeneet kehittäjät tuottavat vähemmän puutteita
- koodausstandardien noudattaminen:
  - vähentää puutteita
  - pienentää työmäärää
- dokumenttien päivittämättä jättäminen aiheuttaa puutteita myöhemmissä elinkaaritoiminnoissa

Kuva 13. Esimerkki abstraktiolomakkeesta.

Kysymysten lisäksi arvioidaan kuhunkin kysymykseen vastaukset silloisella hetkellä eli luodaan hypoteesit, joihin todellisia mittaustuloksia verrataan. Hypoteesien määrittäminen saa projektiryhmän miettimään tarkemmin sen hetkistä prosessien ja tuotteiden tilaa. Tärkeintä ei ole arvioida hypoteesien oikeellisuutta, vaan hypoteesien ja todellisen mittaustiedon vertaamisella pyritään löytämään ja analysoimaan syitä, miksi tulokset mahdollisesti poikkeavat hypoteeseista tai vastaavat asetettuja hypoteeseja. Hypoteesien asettamisella pyritään parantamaan mittauksen aikaansaamaa oppimisprosessia.

Asetettua tavoitetta (kuva 11) ja haastattelun tuloksia (kuva 13) vastaavia kysymyksiä on esitetty kuvassa 14.

Tuotekohtaisia kysymyksiä:

Q1	Mikä on tuotteen kokoonpano?
Q2	Kuinka hyvin lähdekoodi vastaa koodausstandardeja?
Q3	Mikä on toimitetun ohjelmiston koko?

Prosessikohtaisia kysymyksiä:

Q4	Mikä on puutejakauma vakavuusasteittain?
Q5	Mikä on puutejakauma havaintoryhmittäin?
Q6	Mikä on puutejakauma elinkaaritoiminnoittain?
Q7	Mikä on puutteiden korjaamisen vaatima työmäärä?
Q8	Mikä on puutteiden ja ohjelmiston koon välinen suhde?
Q9	Mikä on tukitoimintojen ja tuotantotoimintojen välinen suhde?
Q10	Mikä on projektiryhmän kokemus?

Kuva 14. Kysymysluettelo.

Asetetut kysymykset ja hypoteesit tarkistetaan, jotta ne varmasti ovat oikeita ja oikein muodostettuja (Solingen ja Berghout 1999). Kysymysten pitää olla muodostettu projektiryhmän haastattelujen perusteella ja siksi on tärkeää tarkistaa, ettei mittausryhmä ole tehnyt tulkintavirheitä tai muita virheitä kysymyksiä tulkitessaan. Koska hypoteesit muodostavat yhdessä kysymysten kanssa pohjan mittareiden määrittämiselle, pitää myös ne tarkistaa.

Mittareiden määrittely

Kun tavoitteet on tarkennettu kysymyksiksi, seuraavana vaiheena on mittareiden määrittäminen ja niiden johdonmukaisuuden ja eheyden tarkastaminen (Solingen ja Berghout

1999). Mittareiden avulla saadaan kerättyä tarvittava kvantitatiivinen tieto kysymyksiin vastaamiseksi. Mittareiden lisäksi pitää myös tunnistaa suoraan mittareiden arvoihin vaikuttavat tekijät, koska nämä tekijät vaikuttavat myös kysymyksien vastauksiin. Jos vaihtelutekijät jätetään huomioimatta, voi joistakin tulkinnoista tulla virheellisiä. Nämä vaihtelutekijät määritellään usein myös mittareiksi, jotta niiden vaikutusta voidaan tarkkailla. Kuvan 14 mukaisia kysymyksiä tukevia mittareita on lueteltu kuvassa 15.

Q1 Mikä on tuotteen kokoonpano?
- M.1.1 Luettelo elinkaaren dokumenteista.
- M.1.2 Luettelo alijärjestelmistä.
- M.1.3 Luettelo moduuleista

Q2 Kuinka hyvin lähdekoodi vastaa koodausstandardeja?
- M.2.1 Standardien noudattaminen moduulettain (hyvin/huonosti)

Q3 Mikä on toimitetun ohjelmiston koko?
- M.3.1 Ohjelmiston koko toimintopisteinä
- M.3.2 Ohjelmiston koko koodiriveinä
- M.3.3 Moduulien lukumäärä

Q4 Mikä on puutejakauma vakavuusasteittain?
- M.4.1 Vähäisten puutteiden lukumäärä
- M.4.2 Vakavien puutteiden lukumäärä
- M.4.3 Kriittisten puutteiden lukumäärä
- M.4.4 Puutteiden kokonaismäärä
- …

Kuva 15. Mittariluettelo.

Määriteltyjen tavoitteiden, kysymysten ja mittareiden tulee olla yhdenmukaisia ja eheitä mitattavan kohteen mallin mukaisesti. Tämän takia tarkistuksia pitää tehdä koko määritysvaiheen ajan. Jos määrittelyissä selviää epäjohdonmukaisuuksia, pitää joko prosessimalleja tai määrityksiä muuttaa.

Pfleegerin (1995) mukaan mittausohjelma kannattaa aloittaa pienellä mittarijoukolla ja laajentaa sitä, kun mittauksesta kertyy kokemusta, prosessit tulevat näkyvämmiksi ja resurssit sallivat. Kypsyystasojen kasvaessa prosessit tunnetaan paremmin ja tavoitteita voidaan lisätä, joten mittausta voidaan suorittaa yhä perusteellisemmin.

GQM-suunnitelman laatiminen

GQM-suunnitelma muodostuu kuvan 11 ja kuvien 13-15 mukaisesti tavoitteiden, kysymysten, mittareiden ja hypoteesien määrityksistä. Suunnitelmassa kuvataan tavoitteiden tarkentuminen kysymyksiksi ja kysymysten tarkentuminen mittareiksi. Osa mittareista on epäsuoria, joten suunnitelmassa kuvataan myös kuhunkin epäsuoraan mittariin kuuluvat suorat mittarit.

GQM-suunnitelma toimii myös ohjenuorana mittaustiedon tulkinnalle ja tarjoaa perustan mittaus- ja analysointisuunnitelmalle (Solingen ja Berghout 1999). GQM-suunnitelma dokumentoi siis mittausohjelman formaalisti ja toimii perustana *mittauksentukijärjestelmälle* (measurement support system). Mittauksentukijärjestelmä tukee mittausryhmää mittaustiedon käsittelyssä.

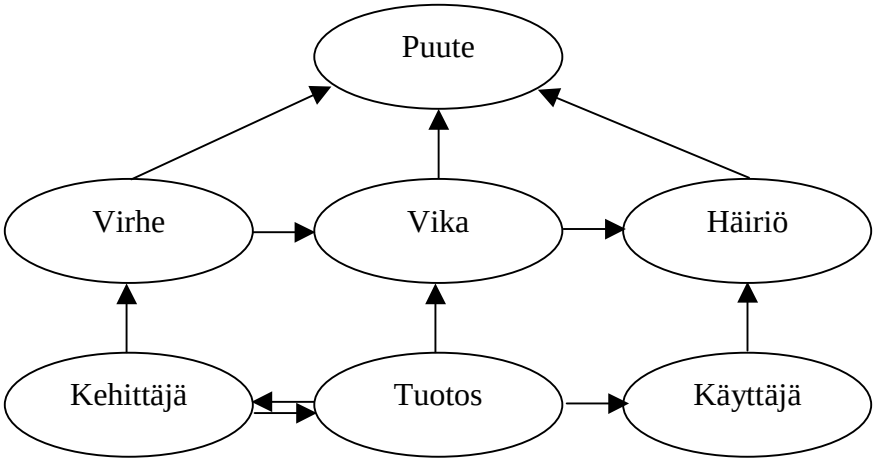
Mittaussuunnitelman laatiminen

Mittaussuunnitelmassa kuvataan kustakin suorasta mittarista seuraavat tiedot (Solingen ja Berghout 1999):

- mittarin formaali määritys
- mittarin tekstimuotoinen kuvaus
- mahdolliset mittaustulokset
- henkilö, joka kyseisen mittarin mittaukset tekee
- aika, jolloin henkilön tulisi mittaus suorittaa
- tapa (lomake, väline), jolla mittaustulos kerätään.

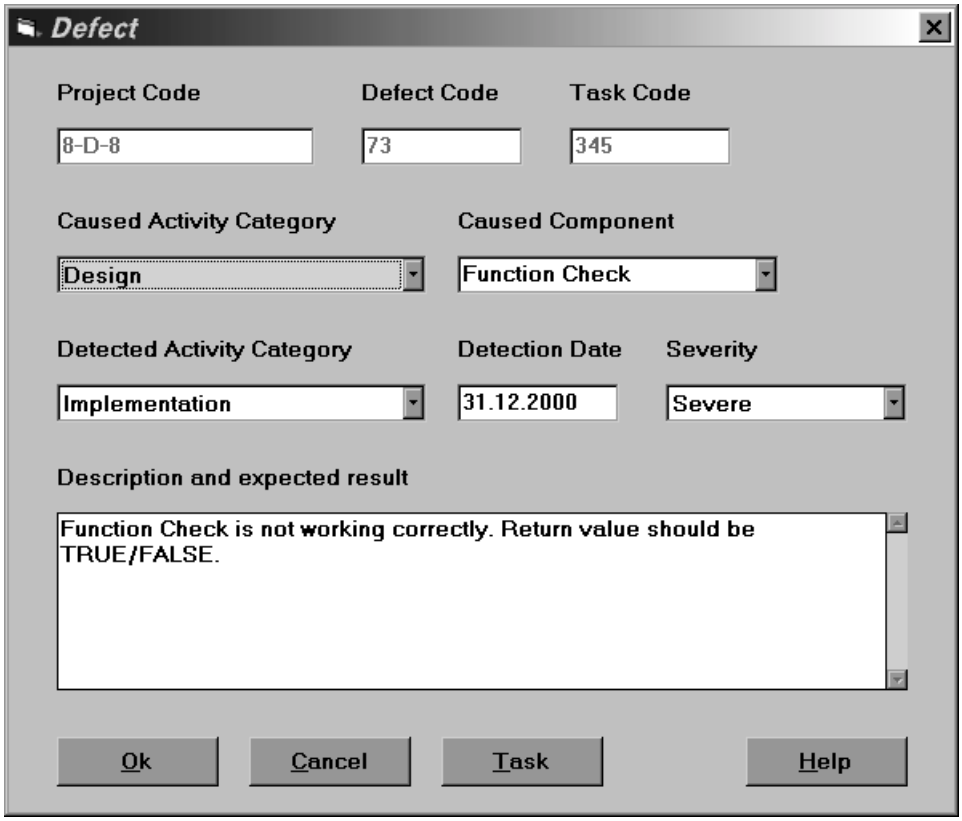
Kuvassa 16 on hahmoteltu *puutteen* (defect) muodostumista Fentonin ja Pfleegerin (1997) esittämän määritelmän perusteella. Puute voi olla *virhe* (error), *vika* (fault) tai *häiriö* (failure). Kehittäjä tekee virheitä, josta aiheutuu vikoja tuotoksiin. Viat tuotoksissa aiheuttavat uusia virheitä sekä synnyttävät häiriöitä käyttäjän käyttäessä tuotosta.

Puutteita voidaan kerätä kehittämisen aikaisissa katselmustilaisuuksissa tai käytön aikana. Kukin havaittu puute käsitellään havaitsijan toimesta esimerkiksi kirjaamalla mittauksentukivälineellä kuvan 17 mukaisesti (Kröger 1999). Kuvan 17 esimerkissä puutteesta kirja-



Kuva 16. Puutteiden synty.

taan toimintoluokka, jossa puute havaittiin, puutteen aiheuttanut toimintoluokka, kohdekomponentti, havaitsemispäivämäärä ja puutteen vakavuusaste. Lisäksi puutteesta kirjataan tarkka kuvaus.



Kuva 17. Puutteiden kirjaus (Kröger 1999).

Tuotoksessa havaittu puute aiheuttaa lisätyötä, jonka tekemiseksi projektiryhmälle määritellään työtehtävä (task). Tästä näkökulmasta puutteen voidaan ymmärtää koostuvan useammastakin virheestä ja viasta, jolloin työtehtävien hallinta on helpompaa projektin kannalta.

Mittaussuunnitelmassa määritellään myös, kuinka eri mittareiden tulokset voidaan helpoiten ja tehokkaimmin kerätä ja kenelle mittaustulokset toimitetaan. Lisäksi siinä kuvataan ja määritellään manuaaliset ja automaattiset tiedonkeräysmenetelmät.

Analysointisuunnitelman laatiminen

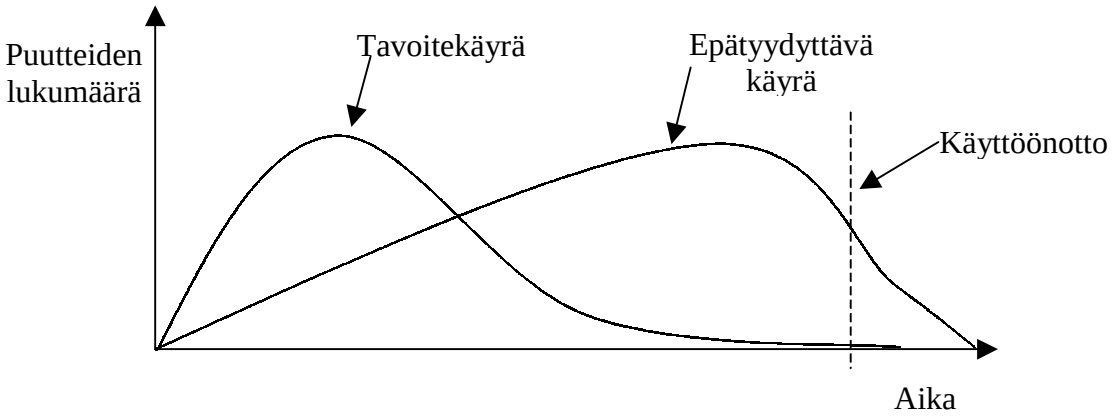
Analysointisuunnitelman tärkein tehtävä on kuvata se, kuinka mittaustiedot saadaan projektiryhmän kannalta helposti tulkittavaan muotoon. Solingenin ja Berghoutin (1999) mukaan tärkeä osa tulosten tulkintaa on hypoteesien vertaaminen todellisiin mittaustuloksiin. Analysointisuunnitelmassa on kuvaajia ja taulukoita, jotka validoivat hypoteesit. Lisäksi suunnitelmassa kuvataan, kuinka vaihtelutekijöitä käytetään tuloksiin. Analysointisuunnitelmasta projektiryhmä näkee, millaisia kaavioita he voivat myöhemmin odottaa oikeiden tulosten pohjalta. Taulukossa 4 on esitetty esimerkki mahdollisesta puutejakaumasta elinkaaritoiminnoittain.

Taulukko 4. Puutejakauma: aiheutuminen ja havaitseminen.

	Havaitseminen							
Aiheutuminen	Vaatimusten määrittely		Suunnittelu		…	Käyttö		Yhteensä	
	#	%	#	%		#	%	#	%
Vaatimusten määrittely	3	6	3	6	…	2	4	9	18
Suunnittelu	1	2	3	6	…	3	6	8	16
…	…	…	…	…	…	…	…	…	…
Käyttö	2	4	2	4	…	2	4	10	20
Yhteensä	10	20	12	24	…	11	22	50	100

Olettamuksena tulisi olla myös puutteiden havaitseminen mahdollisimman aikaisessa vaiheessa kuvan 18 periaatteen (Kan 1995) mukaisesti. Tämä vähentää aiempien vaiheiden tuotosten korjaustyötä ja tuottaa stabiilimman lopputuloksen.

Mittaustietoja analysoimalla voidaan selittää, kuinka vaihtelutekijät vaikuttavat mitattavaan kohteeseen (Solingen ja Berghout 1999). Tämän lisäksi hypoteesien asettaja sekä mittaustiedon kerääjä suorittavat todellisen ja hypoteettisen tiedon vertailun. Tiedot tulisi esittää sellaisessa muodossa, että se tukee projektiryhmän tulkintatehtävää. Mittausryhmän tulisi toimia tiedon tulkitsemisen ohjaajana, ei siis varsinaisesti tulkita mittaustietoa. Analysointisuunnitelma on myös lähtökohta palautemateriaalin tuotannolle.



Kuva 18. Puutteiden havaitseminen.

Ennen kuin tiedon kerääminen voi alkaa, pitää GQM-, mittaus- ja analysointisuunnitelmat hyväksyttää projektiryhmällä. Tällöin on tärkeää tarkistaa, että projektiryhmä on samaa mieltä määritellyistä tavoitteista, kysymyksistä ja mittareista ja ettei puuttuvia tai turhia määrityksiä enää löydy. Lisäksi palautemateriaalin tulee olla projektiryhmälle sopivaa. Mittaussuunnitelmasta on tärkeintä käydä läpi mittausvälineet ja -lomakkeet, koska kaikkien projektiryhmän jäsenten tulee ymmärtää niiden toiminta.

3.5.3 Tiedonkeräysvaihe

Tiedonkeräysvaiheessa kerätään tietoa mittausohjelman mukaisesti ja mittaustiedot tallennetaan mittauskantaan (Solingen ja Berghout 1999). Vaiheen käynnistäminen vaatii, että tarvittavat lomakkeet ja välineet ovat käytettävissä. Mittausohjelman tehokas toteuttaminen edellyttää myös mittauksentukijärjestelmää.

Tiedonkeräysmenetelmät

Tiedonkeräysmenetelmät on määritelty mittaussuunnitelmaan ja niitä sovelletaan sen mukaisesti. Mittaustiedon keräysmenetelmät voidaan jakaa kolmeen eri tapaan (Solingen ja Berghout 1999): manuaaliset tiedonkeräyslomakkeet, elektroniset lomakkeet ja automaattinen mittaustiedon kerääminen.

Manuaaliset tiedonkeräysmenetelmät ovat helppoja ja joustavia menetelmiä. Tästä syystä niitä käytetään usein alkuvaiheessa, kun tiedonkeräysmenetelmien luomisesta ei ole kokemusta. Manuaalisesti ylläpidettävät ja täytettävät lomakkeet koostuvat yleensä yhdestä sivusta, johon kerätään mittaustietoa GQM- ja mittaussuunnitelman mukaisesti.

Lomakkeen pitää olla käytön kannalta sellainen, että se voidaan täyttää peräkkäisesti. Suunnittelussa pitää varata tila eksplisiittisesti kaikille kerättäville arvoille. Jotta lomakkeen tietoja voidaan tarvittaessa tarkistaa, pitää lomakkeella olla myös tieto lomakkeen täyttäjästä. Lomakkeella tulisi olla myös osio, jossa selitetään etukäteen kaikki kerättävät arvot. Myös lisätietoja varten on hyvä olla tilaa, esimerkiksi lomakkeen täytön helppouden arvioimiseksi. Lomakkeita tulisi täyttää aina, kun jokin tietty mittausohjelmaan kuuluva tehtävä on suoritettu. Täytetyt lomakkeet toimitetaan säännöllisesti mittausryhmälle, joka tarkastaa ja syöttää mittaustulokset lomakkeilta mittauskantaan.

Tiedonkeräyslomakkeet eivät ole staattisia, vaan ne muovautuvat mittausohjelman aikana. Projektiryhmän täytyy tukea näitä muutoksia ja lisäksi kaikki muutokset pitää myös käsitellä GQM- ja mittaussuunnitelmassa.

Elektroniset tiedonkeräyslomakkeet voidaan toteuttaa esim. sähköpostilla, web-sivuilla tai tietokantasovelluksena (Solingen ja Berghout 1999). Elektronisen tiedonkeräämisen tärkein hyöty on se, että tietoa ei tarvitse syöttää uudelleen lomakkeilta, vaan tiedot ovat suoraan elektronisessa muodossa. Lisäksi jo tietoja syötettäessä voidaan tehdä tarkastuksia syötetyille arvoille ja tarvittaessa huomauttaa käyttäjää virheellisistä tai puuttuvista arvoista. Elektronisten tiedonkeräyslomakkeiden ylläpitäminen on helpompaa kuin manuaalisten, sillä ylläpito voidaan suorittaa keskitetysti palvelimella, josta lomakkeita suoritetaan.

Sähköpostin käyttöä tiedonkeräysmenetelmänä voidaan perustella sillä, että sähköpostin käyttö on monesti tuttua ja kuuluu ihmisten päivittäiseen rutiiniin. Lisäksi valmiit järjestelmät sähköpostia varten ovat jo olemassa. Sähköpostit voidaan ohjata sovellukselle, joka osaa kerätä, tarkastaa ja tallentaa tiedot mittauskantaan automaattisesti. Sähköpostin yhteydessä voidaan myös käyttää liitetiedostoina esimerkiksi taulukkolaskentataulukoita, joista tieto saadaan helposti siirrettyä mittauskantaan.

Tietokantasovelluksen käyttö vähentää mittausryhmän tiedontallennustyötä paljon, sillä tiedot tallentuvat suoraan tietokantaan (Solingen ja Berghout 1999). Tämän edellytyksenä on tietokantajärjestelmän asentaminen yritykseen, jos sellaista ei ennestään ole käytössä. Jos tiedonkeräyssovellusta ei tehdä yrityksessä itse, joudutaan tekemään investointi sellaisen hankkimiseksi. Hankinta voi tulla kalliiksi ja asennustyö voi kestää kauan, jos käyttäjiä on paljon ja sovellus joudutaan asentamaan jokaiselle työasemalle. Kuvassa 17 on esimerkki tietokantasovelluksen käytöstä puutteiden kirjaamiseksi. Puutteiden etsimiseksi tarvittu ajankäyttö ja puutteiden korjaamiseksi käytetty aika voidaan tietokantasovellusta käytettäessä kirjata kuvan 19 mukaisesti. Puutteiden etsiminen ja korjaaminen ovat siis etukäteen määriteltyjä tehtäviä, jotka pitää ajoittaa projektissa sopiviin ajankohtiin (Kröger 1999).

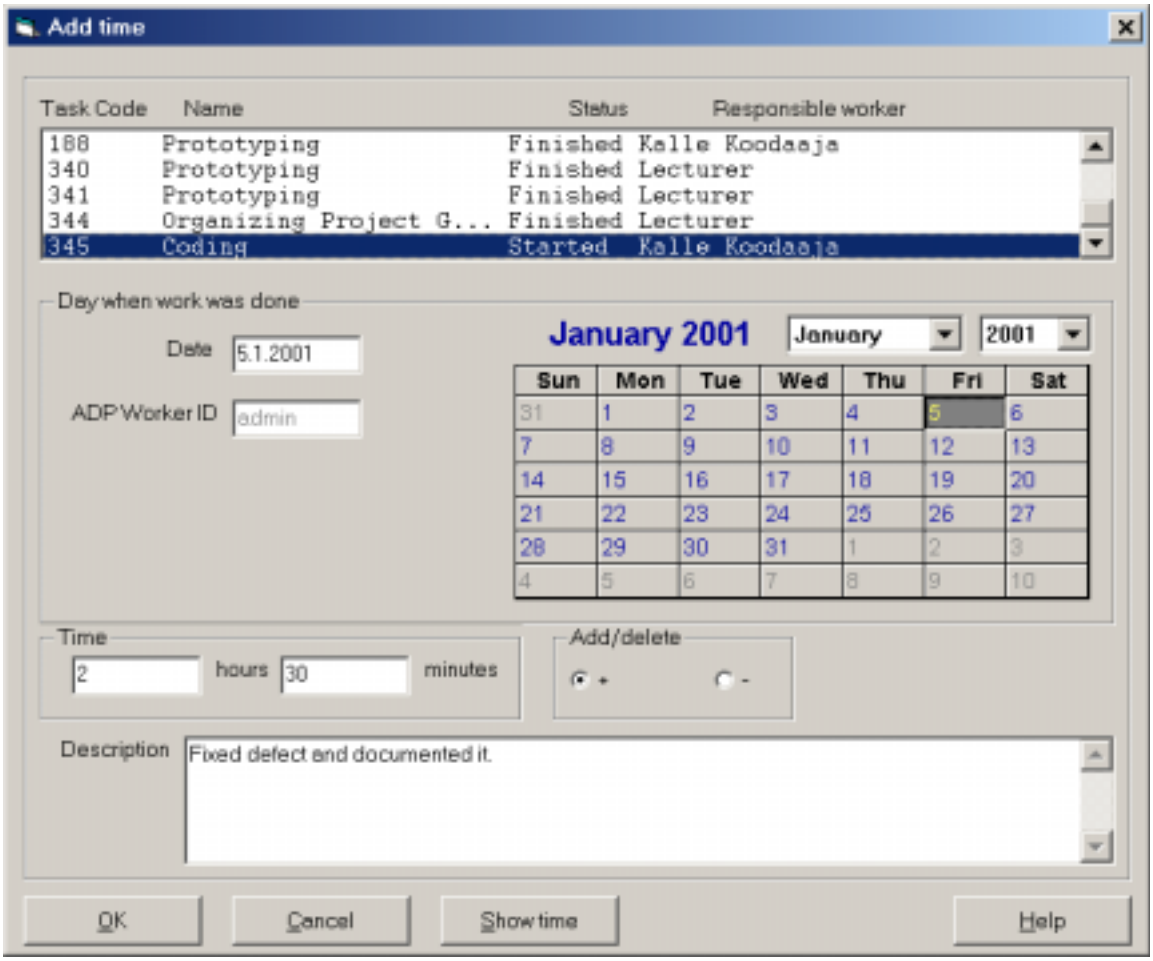
Internetin käyttö yhdistää sähköpostin ja tietokantasovelluksen parhaat puolet mahdollistamalla tietojen tallentamisen tietoverkkojen välityksellä suoraan mittauskantaan. Jos websovellusta voidaan käyttää Internetistä (ei siis pelkästään yrityksen sisäisestä verkosta), mahdollistaa se sovelluksen käytön esim. toisella paikkakunnalla olevasta toimipisteestä.

Tiedonkeräystä voidaan suorittaa osittain myös automaattisesti. Tällaisia *automaattisia tiedonkeräysvälineitä*, jotka keräävät mittaustietoa ennalta määriteltyjen algoritmien perusteella, voidaan rakentaa esimerkiksi ohjelmakoodin analysoimiseksi tai CASE-välineen ylläpitämän kuvauskannan analysoimiseksi. Yleensä ohjelmistotuotannon tukitoimintoja avustaviin välineisiin liittyy jonkinlainen raportointiosa.

Vaikka automaattisten tiedonkeräysmenetelmien käyttö voi olla tehokasta, on Solingenin ja Berghoutin (1999) mukaan hyvä muistaa, että tärkein tieto tulee ihmisiltä eikä välineiltä. Asetettujen tavoitteiden saavuttaminen on mittausohjelman tärkein tavoite. Jos manuaaliset

mittauskeinot riittävät tarvittavan mittaustiedon saamiseksi, ei erityisiä automaattisia mittausvälineitä kannata välttämättä ottaa käyttöön.

Tiedonkeräystä suunniteltaessa on mahdollista määritellä hyvinkin yksityiskohtaisia tietotarpeita. Esimerkiksi Solingen ja Berghout (1999) ehdottavat hyvin yksityiskohtaista tiedonkeruulomaketta katselmuksia varten: katselmoijan tietämyksen taso arvioidaan moneen kertaan asteikolla 0-10, katselmuksesta opitut asiat arvioidaan asteikolla 0-10, lasketaan puutteiden lukumäärä vakavuusasteittain, lasketaan katselmuksessa esitettyjen kysymysten määrä, jne.



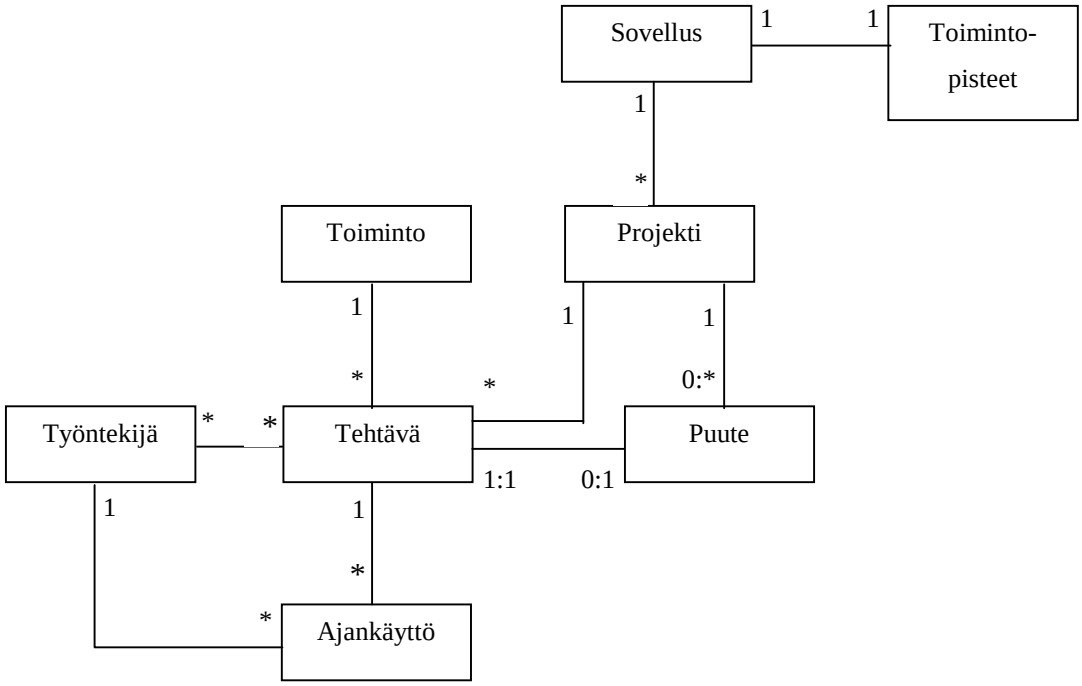
Kuva 19. Puutteen korjaamiseen käytetyn ajan kirjaaminen (Kröger 1999).

Yksityiskohtaisen tiedon keräyksessä tulee kuitenkin olla tarkkana, koska silloin hyvästä tavoitteesta huolimatta ei välttämättä saada objektiivista tai prosessin ja tuotteen parantamisen kannalta oleellista tietoa. Tästä syystä kerätyn tiedon hyödyntäminen pitää perustella hyvin suunnitelmissa. Esimerkiksi katselmuksissa on oleellista löytää ja kuvata puutteet

selkeästi, jotta voidaan arvioida niiden syyt ja merkitys projektin kannalta: aiheuttaja nykyisessä prosessissa, korjaustarve, korjaukseen kykenevät henkilöt, korjaustyön kesto ja korjaustyön vaikutus muihin projektin tehtäviin. Pressmanin (1994) mukaan katselmuksissa pitää keskittyä tuotteeseen eikä tuottajiin.

Mittauksentukijärjestelmä

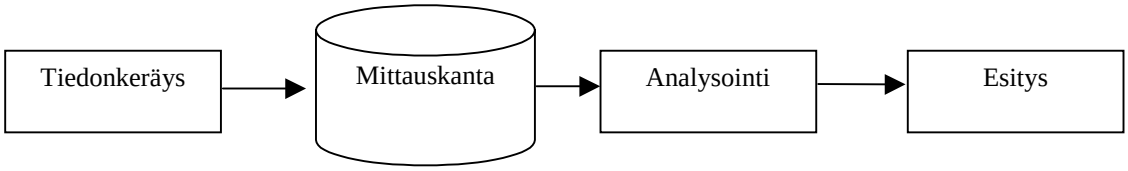
Mittauksentukijärjestelmän tarkoitus on tukea mittausohjelmaa kaikissa toiminnoissa. Sen perustana toimivat yleiset työvälineet, kuten taulukkolaskentaohjelmistot, tilastolliset työvälineet, tietokantasovellukset ja esitysohjelmistot. Nämä välineet ovat tarpeeksi joustavia mahdollisten muutosten suhteen. Joustavuus on tärkeää, jotta aiemmin kerättyä tietoa voidaan käyttää muutoksista huolimatta. Toinen tärkeä tukijärjestelmän ominaisuus on *saatavuus* (accessibility), etenkin katselmusten aikana, jolloin projektiryhmä voi tutkia mittaustietoja haluamallaan tavalla ja etsiä vastauksia kysymyksiin (Solingen ja Berghout 1999).



Kuva 20. PMS-järjestelmän mittaustietokanta (Kröger 1999).

Mittauskanta voidaan toteuttaa taulukkolaskentaohjelmistolla tai käyttämällä erillistä tietokantaa, jolloin tiedot voidaan siirtää tarvittaessa käsiteltäväksi taulukkolaskentaohjelmistoon. Se sisältää kaiken kerätyn raakatiedon, jonka mittausryhmä on sinne syöttänyt. Tiedot tulee tarkastaa oikeellisuuden ja eheyden suhteen.

Kuvassa 20 on esitetty puute-, toimintopiste- ja työaikatietoja sisältävän mittaustietokannan kaavio (Kröger 1999). Kaaviosta nähdään, että sovellukseen voi liittyä useampi projekti. Sovellukselle ylläpidetään toimintopistearvoa, josta voidaan johtaa erilaisia mittareita puutetietojen ja ajankäyttötietojen avulla. Puutteet kirjataan sellaisina kokonaisuuksina, että niitä voidaan ajoittaa tiettyyn toimintoon liittyvinä projektin tehtävinä työntekijöille.



Kuva 21. Mittauksentukijärjestelmä.

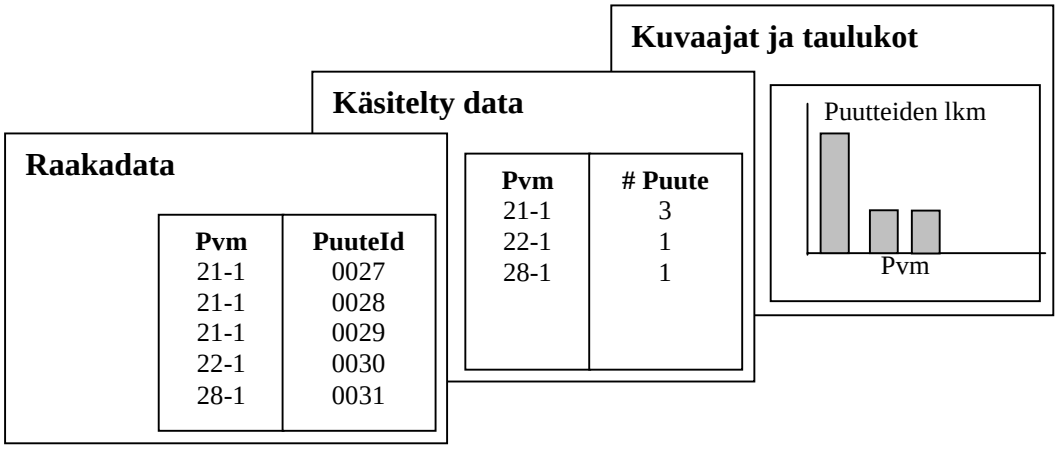
Solingenin ja Berghoutin (1999) mukaan mittauksentukijärjestelmän kaksi tärkeää osaa ovat analysointi- ja esitysohjelmisto (kuva 21). Analysointiohjelmiston tulisi olla riittävän joustava, koska mittausohjelma kehittyy ajan kuluessa ja mittaustavoitteita voidaan joutua lisäämään, muuttamaan ja poistamaan. Esitysohjelmiston tulee tukea sekä esityskalvojen että automaattisten esitysten luomista palautetilaisuuksia varten.

Mittauskannan tiedot ja niistä johdetut tiedot muodostavat kolme abstraktiotasoa (kuva 22). Ensimmäisellä tasolla on mittauskannan raakadata. Käsitelty data on raakadatasta valitsemalla, lajittelemalla ja laskemalla saatua tietoa, joka toimii pohjana seuraavan tason kuvaajien ja taulukoiden luomiselle. Nämä kolmannen tason kuvaajat ja taulukot soveltuvat tiedon esittämiseen ja tulkitsemiseen.

Koska GQM-suunnitelma määrittelee tavoitteiden, kysymysten ja mittareiden väliset suhteet, se määrittelee kehyksen mittaustiedon analysoinnille. Näin ollen analysointiväline tulee suunnitella GQM-suunnitelmassa määriteltyjen suhteiden mukaisesti.

Solingen ja Berghout (1999) ovat käyttäneet mittausohjelmissaan analysointijärjestelmänä taulukkolaskentaohjelmistoa sen joustavuuden vuoksi. Taulukkolaskentaohjelmistoa käytettäessä kannattaa jokaista mittaustavoitetta kohden luoda oma tiedostonsa, etenkin jos tavoitteisiin liittyy paljon mittaustietoa. Jokainen tiedosto sisältää seuraavat laskentataulukot:

- *Tavoitetaulukko,* jossa kuvataan GQM-suunnitelman mukaisesti tavoite ja siihen liittyvät kysymykset.
- *Datataulukko,* johon on kopioitu mittauskannasta tavoitteeseen liittyviin kysymyksiin vastaamiseen tarvittavat tiedot ja hypoteesit.
- Useita *kysymystaulukoita,* joissa kussakin on yksi tavoitteeseen liittyvä kysymys. Kysymystaulukossa mittaustieto käsitellään ja muutetaan taulukoiksi ja kuvaajiksi. Kysymystaulukot ovat siis kuvassa 22 esitettyjen kahden viimeisen abstraktiotason toteutuksia.



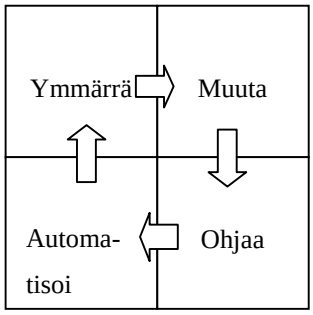
Kuva 22. Analysoinnin kolme abstraktiotasoa.

Kun tieto on käsitelty analysointilomakkeilla, kopioidaan kuvaajat ja taulukot kysymystaulukoista esitysohjelmistoon. Tietojen päivitystä varten luodaan linkit taulukkolaskentataulukoiden ja esitysohjelmiston välille, jotta esitysten kuvaajat ja taulukot säilyvät ajantasalla mittaustietojen muuttuessa (Solingen ja Berghout 1999).

3.5.4 Tiedon tulkintavaihe

Gradyn ja Caswellin (1987) mukaan prosessin muuttaminen edellyttää prosessin ymmärtämistä (kuva 23). *Tulkintavaihe* on mittausohjelman kannalta elintärkeä vaihe, koska tässä vaiheessa yritetään löytää vastaukset asetettuihin kysymyksiin ja ymmärtää, ollaanko asetettuihin tavoitteisiin päästy. Tämän jälkeen prosesseja on mahdollista muuttaa ja ohjata haluttuun suuntaan. Tulkintavaihe koostuu seuraavista toiminnoista (Solingen ja Berghout 1999):

- palautetilaisuuksien valmistelu
- palautetilaisuuksien pitäminen
- raportointi.



Kuva 23. Prosessin parantaminen.

Palautetilaisuuksien valmistelu

Mittausohjelman tuloksista keskustellaan palautetilaisuuksissa (Solingen ja Berghout 1999). Mittausryhmä valmistelee palautetilaisuudet luomalla kerätyistä mittaustiedoista esitysmateriaalin, joka tukee GQM-suunnitelmassa asetettuihin kysymyksiin vastaamista. Esitysmateriaalin ja GQM-suunnitelman perusteella pitää pystyä päättelemään, onko asetetut tavoitteet saavutettu. Vaikka mittausryhmä valmistelee tilaisuudet pääosin, voi projektiryhmän edustajien osallistumisesta valmisteluihin olla hyötyä.

Palautetilaisuuden valmistelu sisältää analysointitiedon päivittämisen mittauksentukijärjestelmällä. Ennen ensimmäistä palautetilaisuutta mittausryhmän pitää tehdä paljon palautemateriaalia. Jotta seuraavat palautetilaisuudet eivät vaatisi yhtä suurta työmäärää, pitäisi mittauksentukijärjestelmän tukea palautemateriaalin automaattista päivitystä.

Koska tavoitteet ja vaatimukset saattavat muuttua mittausohjelman aikana, on joskus tarpeen luoda uutta lisämateriaalia. Voidaan esimerkiksi selvittää, mikä on kunakin viikkona puutteiden lukumäärän suhde aiemmin saman tyyppisistä projekteista kalibroituun jakaumaan, jos puutteiden vaatiman työmäärän suhteellinen osuus on normaalia suurempi. Jos mittauksentukijärjestelmä antaa mahdollisuuden, voidaan analysoida syyt lisääntyneeseen puutemäärään. Solingenin ja Berghoutin (1999) mukaan lisämateriaalin tuottaminen ei kuitenkaan koskaan saa johtaa alkuperäisten tavoitteiden ja kysymysten hylkäämiseen.

Mittauksentukijärjestelmä voi mahdollistaa hyvinkin monipuolisen analysoinnin. Tästä syystä palautetilaisuudessa näytettävä esitysmateriaali pitää valita huolellisesti, jotta ei tehdä vääriä tulkintoja mittaustuloksista. Tämän tarkastuksen voi tehdä joku mittausryhmän jäsen yhdessä projektiryhmän jäsenen kanssa. Kaikki mittaustiedot ja kaikki luodut palautemateriaalit tallennetaan tulevaa käyttöä varten. Palautetilaisuudessa käsiteltävä palautemateriaali voidaan monistaa jaettavaksi joitakin päiviä ennen palautetilaisuuden pitoa. Näin osallistujat voivat perehtyä materiaaliin jo ennen varsinaista tilaisuutta ja siten itse palautetilaisuudesta tulee tehokkaampi.

Palautetilaisuudet

Palautetilaisuudet ovat muodoltaan Augustin (1991) kuvaaman JAD-menetelmän mukaisia kokoustyyppisiä istuntoja. Niihin osallistuvat kaikki projektiryhmän ja mittausryhmän jäsenet. Kokouksia tulisi pitää noin 6-8 viikon välein ja niiden tulisi olla kestoltaan 1,5-2 tuntia. Tilaisuuden järjestelyihin kuuluvat tilojen varaaminen, tarvittavien laitteiden hankkiminen materiaalin esittämistä varten ja osallistujien kutsuminen. Jos osallistujia on enemmän kuin viisi, nimetään etukäteen joku mittausryhmän jäsen tilaisuuden vetäjäksi. Hänen tehtävänään on esittää materiaalit käyttäen tietokonetta ja dataprojektoria tai piirtoheitintä ja johtaa ryhmän keskustelua. Keskustelun aikana vetäjän tulisi tarkkailla ryhmän suoritusta ja pitää huoli siitä, että kaikki ymmärtävät mistä kulloinkin on kyse (Solingen ja Berghout 1999).

On luonnollista, että istunnon vetäjänä toimii mittausryhmän jäsen, mutta projektiryhmän ja mittausryhmän roolien välillä on palautetilaisuuksissa selvä ero (Solingen ja Berghout 1999). Periaatteessa projektiryhmän jäsenten tehtävä on pitää palautetilaisuus ”käynnissä”. Koska he ovat mittauskohteen suhteen asiantuntijoita, he analysoivat, tulkitsevat ja vetävät johtopäätöksiä mittauksista ja kääntävät johtopäätökset *toimenpiteiksi* (action points). Projektiryhmän tulee siis keskittyä arvioimaan edellisten palautetilaisuuksien toimenpiteet, tulkitsemaan mittaustietoja suhteessa asetettuihin kysymyksiin ja tavoitteisiin ja muuntamaan tulkinnat johtopäätöksiksi ja toimenpiteiksi. Kuvassa 24 on esitetty periaate toimenpiteiden johtamiseksi tavoitteista ja mittaustuloksista.

Mittausryhmän jäsenten tulee välttää tiedon tulkintaa, jos he eivät ole mittauskohteen suhteen asiantuntijoita. Heidän tehtävä on esitellä projektiryhmälle tarvittaessa vaihtoehtoisia

tulkintoja. Tämän lisäksi he tekevät tilaisuudesta muistiinpanoja myöhemmin kirjoitettavaa raporttia varten.

Palautetilaisuudet ovat arkaluontoinen osa mittausohjelmaa, jossa osallistujien keskinäinen luottamus on olennaisen tärkeää. Keskustelut alkavat yleisellä keskustelulla keskittymällä tavoitteisiin, kysymyksiin ja mittareihin. Mittausryhmän on huolehdittava siitä, että keskustelu pysyy faktoissa ja kehitystoimenpiteissä. Analysointi ei saa keskittyä yksittäisten henkilöiden suorituksiin.

Tavoite: Analysoi korjaustyön osuus suhteessa kokonaistyöhön tarkoituksena osaamisen selvittäminen suhteessa luotettavuuteen ja sen syihin näkökulmana kehittämisryhmä kontekstina projektiryhmä ZYX.

Kysymyksiä: Mikä on kokonaistyömäärä?
Mikä on puutteiden korjaamiseen käytetty työmäärä?
Mikä on keskimääräinen kokemus välineestä?
…

Mittaus:

	Hypoteesi	Toteutunut
Kokonaistyöaika (h)/toimintopiste	4	6
Korjausaika (h)/puute	0,4	0,8
Keskimääräinen kokemus (kk)	4	6,5
…		

Toimenpiteet: Välinetuntemuksen parantaminen
Kohdetuntemuksen parantaminen
…

Kuva 24. Toimenpiteiden johtaminen.

Raportointi

Palautetilaisuuden jälkeen mittausryhmä kirjoittaa tilaisuudesta raportin, johon on kirjattu kaikki tilaisuudessa esille tulleet olennaiset tulkinnat, johtopäätökset ja toimenpiteet (Solingen ja Berghout 1999). Tämä raportti jaetaan kaikille projektiryhmän jäsenille. Projektiryhmä ”omistaa” mittaustiedon, joten ryhmä voi päättää jaetaanko mittaustietoja ja yksityiskohtaisia raportteja ylemmälle johdolle, joka saattaa tehdä virhetulkintoja annetuista

yksityiskohtaisista tiedoista. Ylemmälle johdolle mittausryhmä voi laatia yhteenvedon tuloksista projektiryhmän suostumuksella. Mittausohjelman tärkeimmistä tuloksista kannattaa informoida koko organisaatiota esim. ilmoitustaulujen tai julisteiden avulla tai julkistamalla tulokset elektronisessa muodossa sisäisessä verkossa.

Jotta mittaustuloksia ja kokemuksia voidaan käyttää uudelleen tulevaisuudessa, kannattaa ne dokumentoida siten, että ne ovat helposti saatavilla ja ymmärrettäviä. Kuvassa 25 on luonnos palautetilaisuuden raportista (Solingen ja Berghout 1999).

Projektin nimi: Projektinumero: Pvm:
Kokouspvm: Läsnä:
Toimittaja: Poissa:
Aihe:

Johdanto
- yleiskuvaus
- mittausohjelman tavoite
- mittauksen kohdennus (prosessin osa, tuotokset)

Mittauksen tulokset
- hypoteesit
- toteutuneet

Johtopäätökset

Toimenpiteet

Kuva 25. Esimerkkiluonnos raportista.

Tavoitteen saavuttaminen on luonnollisesti mittausohjelman kannalta tärkeä osa, mutta myös kustannusten ja hyötyjen analysointi kannattaa sisällyttää mittausohjelman loppuraporttiin. Kustannusten ja hyötyjen analysoinnin suorittaa mittausryhmä. Tärkeää on todeta ylittääkö saavutetut hyödyt kustannukset (Solingen ja Berghout 1999).

3.6 Kustannukset ja hyödyt

Gradyn ja Caswellin (1987) mukaan mittausohjelmasta saatavia hyötyjä ovat mm. parempi ohjelmistoprosessin ymmärtäminen, edistymisen mittaamisen oppiminen ja ohjelmisto-

tuotannon objektiivisempi hallinta. Saavutettujen hyötyjen mittaaminen ei kuitenkaan ole niin itsestään selvää kuin kustannusten mittaaminen.

3.6.1 Mittausohjelman kustannusmalli

Solingenin ja Berghoutin (1999) esittämän kustannus-hyöty –analyysin avulla nähdään, että kannattavan mittausohjelman laatiminen ei ole niin suoraviivaista kuin voisi olettaa. Heidän esittämä operationaalinen kustannusmalli on laadittu kuuden todellisen mittausohjelman perusteella. Suurin osa kustannuksista muodostuu työvoimakustannuksista mittausohjelman eri vaiheissa.

Kustannusmalli sisältää kaksi eri tilannetta, jotka esittävät mittausohjelman ensimmäistä käyttökertaa ja rutiinimittausta organisaatiossa. Ero näiden kahden tilanteen välille on tehtävä siksi, että ensimmäinen GQM:n käyttökerta vaatii huomattavasti enemmän työpanosta ja tulee siten myös kalliimmaksi kuin seuraavat. Kummassakin kustannusmallissa erotellaan seuraavat tyypilliset toiminnot:

1. *Mittausohjelman suunnittelun* aikana tunnistetaan olemassa olevat syötteet, edellytykset ja rajoitteet. Lisäksi otetaan käyttöön välineet ja menettelytavat, valitaan kehitettävä alue, valitaan projekti, suunnitellaan alustavasti mittausohjelma ja valmistellaan ja koulutetaan projektiryhmä.

2. *Tavoitteiden tunnistamisen ja määrittelyn* aikana kuvataan projekti ja organisaatio, tunnistetaan ja valitaan kehitystavoitteet, määritellään ja valitaan GQM-tavoitteet, mallinnetaan mittauksen kannalta olennaiset ohjelmistoprosessit ja tunnistetaan uudelleenkäytettävät tuotteet.

3. *GQM-haastattelujen suorittamisen* aikana tutustutaan dokumentaatioon, määritellään ja aikataulutetaan haastattelut sekä kutsutaan haastateltavat, tavataan projektiryhmä, suoritetaan haastattelut ja raportoidaan haastattelujen tulokset.

4. *GQM-tuotoksien kehittäminen* sisältää GQM-suunnitelman ja mittausohjelman määrittelyn, tarkastuksen ja parantelun, tiedonkeräysmenetelmien tunnistamisen ja määrittelyn ja analysointisuunnitelman kehittämisen.

5. *Tietojen keräysvaiheessa* testataan ja otetaan käyttöön tietojenkeräysmenetelmät ja –lomakkeet, tavataan projektiryhmä, käynnistetään mittausohjelma ja kerätään, validoidaan, koodataan ja tallennetaan mittaustiedot.

6. *Tietojen analysointi- ja tulkintavaiheessa* analysoidaan kerätty mittaustieto, valmistellaan esitysmateriaali, suunnitellaan ja pidetään palautetilaisuus sekä raportoidaan siitä.

Taulukossa 5 on listattu mittausohjelmiin liittyviä tyypillisiä kustannuksia ja hyötyjä.

Taulukko 5. Mittausohjelman kustannuksia ja hyötyjä (Solingen ja Berghout 1999).

Kustannukset	Hyödyt
- Mittausohjelman valmisteluun mittausryhmältä kulunut aika (palkat ja juoksevat kulut) - Projektiryhmän kanssa tapaamisiin kulunut aika - Tiedonkeräyslomakkeiden täyttöön projektiryhmältä kulunut aika. - Mittauksentukijärjestelmän kehittämiseen kulunut aika - Mittausohjelmaa tukevien laitteiden ja ohjelmistojen osto - Mittausdatan prosessointiin ja palautetilaisuuksien valmisteluun mittausryhmältä kulunut aika	- Laadun parantumisesta johtuva myynnin kasvu - Laadun parantumisesta johtuva myynnin pienenemisen välttäminen - Ajansäästö ohjelmistojen kehityksessä parantuneen kehitysprosessien tuntemuksen vuoksi - Kustannussäästöt parantuneen resurssienhallinnan kautta - Kustannusten välttäminen parantuneen resurssienhallinnan kautta

Kustannus-hyöty -analyysiä pidetään vaikeasti suoritettavana, koska laadun ja kustannusten vertailu on vaikeaa (Solingen ja Berghout 1999). Analyysin tarkoitus ei ole laskea investoinnin tarkkaa tuottoa, vaan tutkia, oliko projekti taloudellisessa mielessä kannattava. Markkinointiosastot osaavat varmasti kertoa, kuinka tarkkoja asiakkaat ovat laadusta ja kuinka parantuneeseen laatuun liittyvät tuotteen ominaisuudet saavat aikaan liikevaihdon kasvua. Jos kustannusten ja hyötyjen analysointia ei laiteta loppuraporttiin, saattavat johto ja projektiryhmä tehdä omia analyysejä, jotka eivät pohjaudu niin tarkkoihin tietoihin kuin mitä mittausryhmä voi käyttää analysointien tekemiseen. Tämä voi johtaa siihen, että mittausohjelma hylätään kannattamattomana.

3.6.2 Rutiinimittauskierroksen kustannusmalli

Rutiinimittauskierrosta voidaan pitää normaalina mittausohjelman soveltamisena, jossa pohjatyö mittausohjelmaa varten on jo tehty. Tästä syystä rutiinimittauskierroksen kustannusmalli kuvaa paremmin kustannuksia ja hyötyjä pidemmällä tähtäimellä kuin ensimmäisen mittauskierroksen kustannusmalli. Rutiinimittauskierroksen kustannusmallissa oletetaan olevan seuraavia piirteitä (Solingen ja Berghout 1999):

- Kustannusmalli perustuu tyypilliseen ohjelmistoprojektiin, jossa pitää haastatella neljää ohjelmoijaa ja yhtä projektipäällikköä. Mittausryhmässä on vain yksi henkilö.
- Mittausohjelmassa on vain yksi päätavoite, mutta kolme mahdollista näkökulmaa. Nämä näkökulmat ovat yrityksen johdon, laadunvarmistuksen ja ohjelmoijan.
- Projektiryhmässä on kymmenen ohjelmoijaa.
- Mittausohjelmaa tukevat välineet, esim. tietokoneohjelmat ja lomakkeet, ovat valmiina.
- Erityistä koulutusta tai ohjeistusta ei tarvitse järjestää, koska osallistujat tuntevat mittausmenetelmän ennestään.
- Palautetilaisuudet ovat tärkein (ja myös kallein) osa mittausohjelmassa ja keskimäärin viiden palautetilaisuuden ajatellaan riittävän tavoitteen saavuttamiseen.

Taulukossa 6 on kuvattu rutiinimittauskierroksen kustannusmalli (harmaalla taustalla olevat rivi ja sarake ovat yksikkökustannuksia). Annetut työmäärät on ilmaistu henkilötyötunteina ja kustannukset saadaan kertomalla henkilötyötunnit sopivilla yrityksen tuntikustannuksilla. Vaikka kustannusmalli onkin edellä kuvatun projektimäärittelyn mukainen, on siinä riittävästi tietoa, jotta sitä voidaan käyttää myös toisenlaisilla projektimäärityksillä.

Seuraavanlainen kustannusrakenne on tyypillinen rutiinimittauskierroksella:

- Noin 30% kustannuksista koostuu mittausohjelman määrittelystä ja loput 70% mittausohjelman jatkosta, lähinnä palautetilaisuuksista.
- Mittausryhmän osuus mittausohjelman työmäärästä on noin 70% ja projektiryhmä käyttää vain noin 30% mittausohjelman kokonaistyömäärästä.
- Projektiryhmän käyttämä työmäärä on alle 1% ryhmän kokonaistyöajasta.

Tyypillisen GQM-mittausohjelman työmäärä on noin kolme henkilötyökuukautta jakautuneena kalenterivuoden ajalle.

Taulukko 6. Rutiinimittauskierroksen työmäärämalli (Solingen ja Berghout 1999).

Tehtävä	**Mittausryhmä**	**Johtaja**	**1 ohjelmoija**	**10 ohjelmoijaa**	**Yhteensä**
GQM-mittausohjelman suunnittelu	4	2	-	-	6
GQM-tavoitteiden tunnistaminen ja määrittely	8	1	1	10	19
GQM-haastattelujen suorittamisen	40	2	2	8	50
GQM-suunnitelman kehittäminen	40	1	1	6	47
Tietojen keräysvaihe	24	1	1.5	15	40
Tietojen analysointi- ja tulkinta per palauteistunto	48	2	2	20	70
Tietojen analysointi- ja tulkinta (5 palauteistuntoa)	240	10	10	100	350
Yhteensä	356	17	15.5	139	512

Suunnittelu ja määrittely vievät normaalisti aikaa neljästä kuuteen viikkoa. Haastateltavien tavoitettavuus on tärkeää määrittelyn keston kannalta. Tiedonkeräyksen ja palautteiden kesto riippuu palautetilaisuuksien tiheydestä ja tavoitteen saavuttamisen nopeudesta. Solingen ja Berghout (1999) kokemusten perusteella tavoitteen saavuttaminen kestää noin vuoden.

Taulukosta 6 nähdään, että mittausohjelman asettaminen valmistelusta tiedonkeräysvaiheen alkuun kestää mittausryhmältä keskimäärin 11,5 päivää (92 h). Projektipäällikön (6 h) ja ohjelmoijien (4 h) osuudet ovat suhteellisen pieniä kaikissa määrittelyvaiheen osissa. Suurin osa työstä on siis tässä vaiheessa mittausryhmän harteilla.

Tiedonkeräysvaiheessa pidetään avauspalaveri jokaisen osallistujan kanssa. Palaveri kestää noin puoli tuntia henkilöä kohden. Mittausryhmä käyttää tässä vaiheessa yhden päivän palavereihin ja toiset kaksi päivää kerätyn tiedon validointiin.

Tulkintavaiheessa mittausryhmä tarvitsee tyypillisesti viisi päivää kunkin palautetilaisuuden valmisteluun ja kukin tilaisuus kestää noin kaksi tuntia osallistujaa kohden. Lisäksi ryhmä tarvitsee kuusi tuntia palautetilaisuuden raportointiin. Kun tavoite on saavutettu, pitää mittausohjelmasta saadut kokemukset dokumentoida.

Mittausohjelman työmäärään vaikuttavat monet seikat (Solingen ja Berghout 1999). Yksi kolmesta tärkeimmästä seikasta on GQM-tavoitteiden määrä. Tavoitteiden lisääminen lisää kaikkien toimintojen työmäärää. Lisäksi tavoitteiden lisääminen monimutkaistaa mittausohjelmaa nopeasti. Saatujen kokemusten perusteella kolme tavoitetta on ehdoton maksimi ja jo se monimutkaistaa mittausohjelmaa paljon. Toinen työmäärään vaikuttava tekijä on projektiryhmän koko. Projektiryhmän koon kasvaminen aiheuttaa sen, että yhä useampaa henkilöä pitää haastatella ja palautetilaisuuksiin menee enemmän aikaa. Näiden kahden työmäärään vaikuttavan tekijän lisäksi kolmas tekijä on palautetilaisuuksien määrä. Palautetilaisuudet ovat eniten aikaa kuluttava ja samalla tärkein osa mittausohjelmaa ja siten niiden määrä on tärkeä tekijä. Koska mittausohjelman hyödyt saadaan esille juuri palautetilaisuuksien avulla, on erityisen tärkeää, että niiden määrästä ei tingitä.

3.6.3 Ensimmäisen mittauskierroksen kustannusmalli

Ensimmäisen mittauskierroksen kustannusmalli sisältää samat perusvaiheet kuin rutiinikierroksen kustannusmalli, mutta kunkin vaiheen vaatimat työmäärät ovat huomattavasti suuremmat. Ensimmäisen mittauskierroksen kustannusmallissa oletetaan olevan seuraavia piirteitä (Solingen ja Berghout 1999):

- Ulkopuolinen GQM-asiantuntija kuuluu mittausryhmään ja toimii sekä projekti- että mittausryhmän ohjaajana.
- Ohjelmoijien määrä on pienempi kuin rutiinikierroksella. Viisi ohjelmoijaa on riittävästi mittausohjelman ensimmäisellä kerralla.

- Mitään erityisiä mittausta tukevia välineitä (tiedonkeräyslomakkeita tai tietokoneohjelmia) ei ole vielä olemassa. Tiedonkeräysjärjestelmät kehittyvät yleensä asteittain paperilomakkeista suorajärjestelmiin.

Taulukossa 7 on kuvattu ensimmäisen mittauskierroksen kustannusmalli. Seuraavanlainen kustannusrakenne on tyypillinen ensimmäisellä mittauskierroksella:

- Ensimmäinen mittausohjelma vaatii keskimäärin kahdeksan henkilötyökuukauden työmäärän. Tämä on huomattavasti enemmän kuin rutiinikierroksen kolme kuukautta. Ensimmäisellä kerralla esimerkiksi mittausryhmän koulutus vie aikaa.
- Noin 50% kokonaistyömäärästä menee mittausohjelman suunnitteluun. Tämä eroaa rutiiniohjelman 30%:sta huomattavasti. Etenkin GQM-suunnitelman määrittely ja mittaussuunnitelma vaativat enemmän työtä.
- Mittausohjelman työmäärästä 70% kuuluu mittausryhmälle ja 30% projektiryhmälle, joten tämä on suhteessa saman verran molemmilla kerroilla.
- Projektiryhmän työmäärä on kaksinkertainen ensimmäisellä mittauskerralla, mutta siltikin alle 2% heidän kokonaistyömäärästään.

Suunnittelu- ja määrittelyvaiheet vievät noin kolme kuukautta, joka on huomattavasti enemmän kuin rutiinikierroksen 6-8 viikkoa. Lisäaika menee lähinnä koulutukseen ja oppimiseen. Tiedonkeräykseen ja palautteeseen pätevät samat perussäännöt kuin rutiinisovelluksessakin eli työmäärään vaikuttavat palautetilaisuuksien määrä ja tavoitteen saavuttamisen nopeus. Kokemusten perusteella vuoden pitäisi riittää tavoitteen saavuttamiseen. Ensimmäiset palautetilaisuudet ovat osoittautuneet kuitenkin ongelmallisiksi joissakin projektiryhmissä, koska vuorovaikutteiset, avoimet ja oppimispainotteiset palautetilaisuudet voivat olla uutta monille ohjelmoijille. Siksi ei ole ihme, että ensimmäinen mittausohjelma voi kestää jopa pitempään kuin vuoden.

Solingenin ja Berghoutin (1999) mukaan ensimmäisellä mittauskierroksella on muitakin eroja mittauksen rutiinisovellukseen nähden. Mittausryhmä joutuu tekemään huomattavasti enemmän töitä. Koska ohjelmistoprosessien mittaaminen ja niiden kehittäminen yleensäkin voi olla monille organisaatioille tuntematonta, voi kehitystavoitteiden ja mittausohjelman määrittelyssä hyvä olla mukana mahdollisesti joku ulkopuolinen GQM-asiantuntija, jolla on kokemusta kehitystoimista. Mittausohjelman suunnittelu vie GQM-asiantuntijalta arvi-

olta kolme päivää ja mittausryhmän muilta jäseniltä neljä päivää. Mittaustavoitteiden tunnistamista ja määrittelyä varten mittausryhmä tarvitsee koulutusta ja selvennystä GQM-asiantuntijalta. GQM-tuotosten kehittämiseen menee paljon enemmän aikaa, koska GQM-ryhmälle pitää kertoa, kuinka suunnitelma luodaan. Myös mittausta tukevien tiedonkeräysmenetelmien, dokumenttien, välineiden ja lomakkeiden kehittäminen vaatii suuren työpanoksen.

Taulukko 7. Ensimmäisen mittauskierroksen työmäärämalli (Solingen ja Berghout 1999).

Tehtävä	**GQM-asiantuntija**	**GQM-jäsen**	**Johtaja**	**1 ohjelmoija**	**10 ohjelmoijaa**	**Yht.**
GQM-mittausohjelman suunnittelu	24	32	4	4	40	100
GQM-tavoitteiden tunnistaminen ja määrittely	18	8	2	1	10	38
GQM-haastattelujen suorittamisen	35	35	2	1	4	76
GQM tuotosten kehittäminen	196	138	2	1	10	346
Tietojen keräysvaihe	-	16	-	3	30	46
Tietojen analysointi- ja tulkinta per palauteinstunto	12	48	8	4	40	108
Tietojen analysointi- ja tulkinta (5 palauteistuntoa)	60	240	40	20	200	540
Yhteensä	333	469	50	30	294	1146

Tiedonkeräysvaiheessa tarvitaan arviolta yksi päivä mittauskannan ja tiedonkeräyslomakkeiden validointiin, niihin liittyvien ongelmien selvittämiseen ja henkilöiden valmennukseen. Lisäksi projektiryhmää pitää motivoida tiedonkeräysvaiheen alussa. Tietojen tulkintavaihe on mittausohjelman tärkein vaihe. Projektiryhmän täytyy oppia tulkitsemaan mittaustietoa. Tämän lisäksi on tärkeää oppia ymmärtämään, että tarkoitus ei ole arvioida yksilöiden suoritusta, vaan parantaa kehitysprosessia. Tulkintojen pitää keskittyä parannustoimenpiteisiin ja GQM-kysymyksiin vastaamiseen.

Myös ensimmäistä mittausohjelmaa aloitettaessa on tekijöitä, jotka vaikuttavat mittausohjelman työmäärään. Yhtenä tärkeimpänä on muutoksia kohtaan esiintyvä vastustus. Muutoksia vastustetaan usein, mutta tämä vastustus voidaan voittaa osoittamalla, että tarvetta muutoksiin esiintyy ja varmistamalla, että henkilökunta saa riittävästi koulutusta ollakseen valmis mittausohjelmaan. Myös projektin koko vaikuttaa mittausohjelman työmäärään. Jos ryhmässä on joku vähemmän kokenut jäsen, on projektiryhmän koon vaikutus yhä tärkeämpi. Kaikki vaiheet vievät tällöin enemmän aikaa ennen kuin mitään tuloksia mittausohjelmasta saadaan. Tämä voi vaikuttaa siten, että ihmiset saattavat menettää mielenkiintonsa mittausohjelmaa kohtaan. Siksi on suositeltavaa, että ensimmäisessä mittausohjelmassa olisi korkeintaan viisi ohjelmoijaa.

4 JATKUVA LAADUN PARANTAMINEN

Ohjelmistoprosessien kehittämisen kannalta on tärkeää, että kehitys- ja mittaustoimista tulee jatkuvia, normaaliin kehitys- ja ylläpitotyöhön integroituja osia. Tällöin saadaan aikaan jatkuvaa kehitystä ja laadun parantumista. Tällaista jatkuvaa kokemuksista oppimista ja kehitystyötä voidaan tarkastella käsitteiden laadunparannusparadigma ja kokemuspaja avulla.

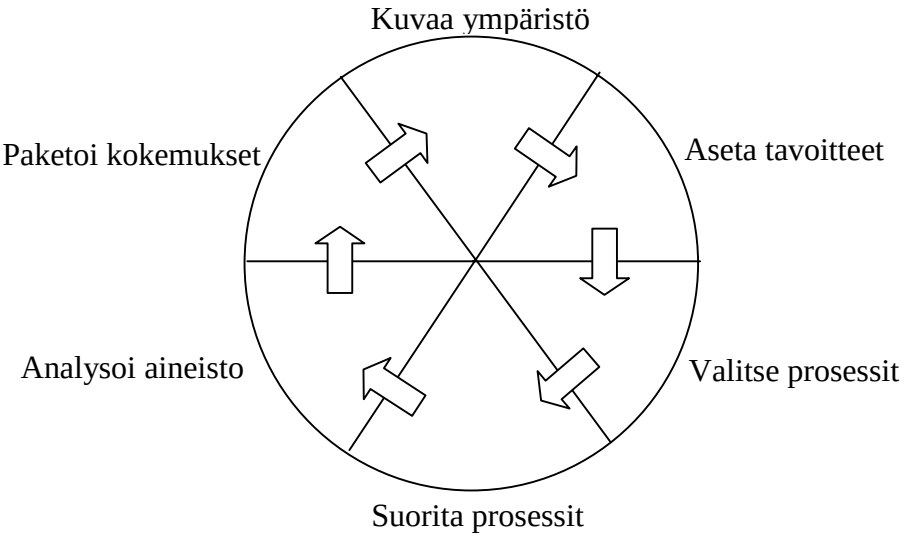
4.1 Laadunparannusparadigma

Laadunparannusparadigma (Quality Improvement Paradigm, QIP) on lähestymistapa, joka korostaa jatkuvaa kokemuksista oppimista sekä projekteissa että organisaatiossa (Solingen ja Berghout 1999). Kokemuksista oppiminen perustuu kokeiluihin ja mittauksen käyttöön, jolloin kaikista projekteista kerätään tiedot talteen uudelleenkäyttöä varten. Uudelleenkäyttöä voidaan soveltaa esimerkiksi prosesseista, tuotteista, ongelmista, menetelmistä tai resursseista saatuihin kokemuksiin.

QIP sisältää kuvan 26 mukaiset kuusi vaihetta (Basili et al. 1994c, Solingen ja Berghout 1999):

1. Ympäristön kuvaaminen olemassa olevien mallien, tietojen, intuition jne. avulla. Luodaan lähtökohdat organisaation liiketoimintaprosessien perusteella ja kuvataan niiden kriittisyys.
2. Asetetaan mitattavia tavoitteita projektin ja organisaation onnistuneelle suoritukselle ja kehitykselle. Edellisessä vaiheessa asetettujen strategisten lähtökohtien pohjalta luodaan järkeviä odotusarvoja tuloksille.
3. Valitaan sopivat kehitettävät prosessit ja projektia tukevat työkalut ja –menetelmät kuvatun ympäristön ja asetettujen tavoitteiden kannalta. Samalla varmistetaan, että valitut prosessit ja työkalut ovat yhteneviä asetettujen tavoitteiden suhteen.
4. Suoritetaan tuotantoprosessit, joista saadaan samalla projektipalautetta tavoitteiden saavuttamisesta.

5. Kunkin projektin lopussa analysoidaan aineistoa nykyisten käytäntöjen arvioimiseksi, ongelmien selvittämiseksi, havaintojen tallentamiseksi ja suositusten tekemiseksi tulevia projekteja varten.
6. Yhdistetään suoritetusta projektista ja aiemmista projekteista saadut kokemukset uusien tai päivitettyjen ja parannettujen mallien ja muiden esitystapojen muodossa ja tallennetaan ne uudelleenkäyttöä varten *kokemuskantaan* (experience base).



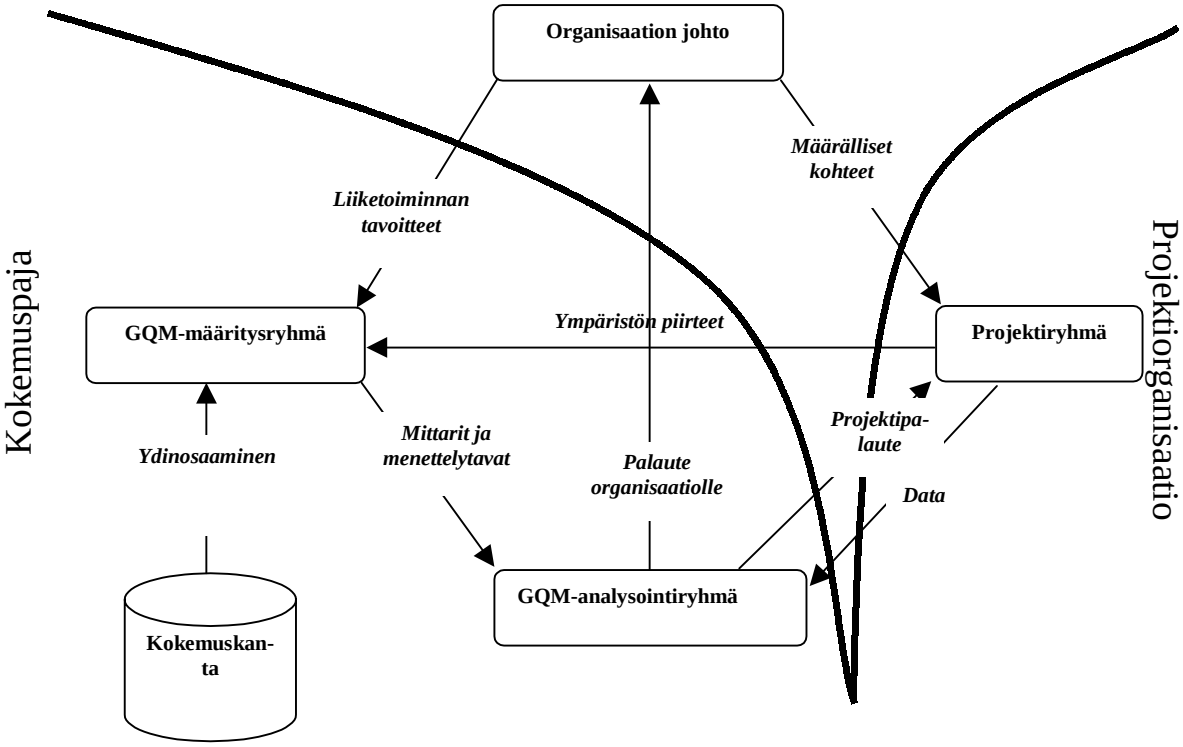
Kuva 26. Laadunparannusparadigma (Basili et al. 1994c).

QIP:ssä on kaksi *palautesykliä* (Basili et al. 1994c, Gresse et al. 1995, Solingen ja Berghout 1999). *Projektikohtainen palautesykli* tuottaa palautetta projektin aikana ongelmien ehkäisemiseksi ja korjaamiseksi sekä projektin valvomiseksi ja tukemiseksi. *Organisatorisesta palautesyklistä* saadaan projektin päätyttyä palautetta projektissa kerätyn tiedon yhtäpitävyydestä ja poikkeavuuksista asetettujen rajojen suhteen. Analysoinnin tuloksista saatavaa kokemusta voidaan hyödyntää tulevien projektien suorituksen parantamiseksi. Tästä syystä kerätty kokemus pitää tallentaa uudelleenkäyttöä varten sellaisessa muodossa, että se on helposti saatavilla.

4.2 Kokemuspaja

Tuotteiden, prosessien ja kokemuksien uudelleenkäyttöä pidetään toteuttamiskelpoisena ratkaisuna laadukkaampien tuotteiden tuottamiseksi halvemmalla. Laatua voidaan parantaa

käyttämällä ja muuntamalla samoja komponentteja yhä uudelleen ja uudelleen samalla parantaen niitä (Basili et al. 1994c).

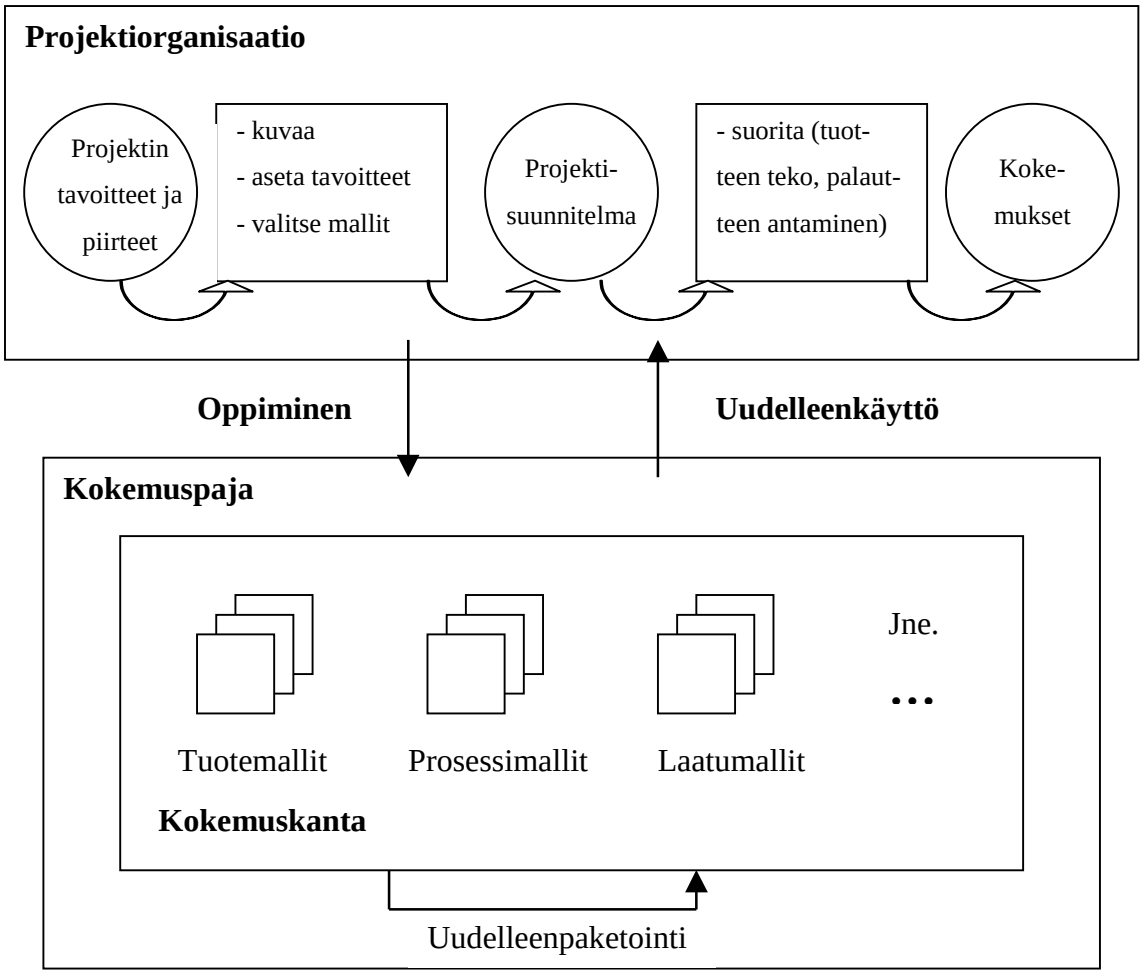


Kuva 27. Mittausohjelman osapuolet ja kokemuspaja (Solingen ja Berghout 1999).

Kokemuspaja (experience factory) on looginen ja/tai fyysinen organisaatio. Kokemuspajan käyttö edellyttää käytännössä kahta erillistä organisaatiorakennetta (kuva 27). Nämä rakenteet ovat itse kokemuspaja sekä projektiorganisaatio. Kokemuspajan toiminnot ovat systemaattinen oppiminen, uudelleen käytettävien kokemusten varastointi (paketointi) ja niiden toimittaminen tarvittaessa tuleviin projekteihin. Nämä toiminnot ovat riippumattomia projektiorganisaatiosta. Projektiorganisaation tehtävä on tuottaa, toimittaa ja ylläpitää tuotteita käyttäen tukena kokemuspajan toimittamia kokemuksia (Basili 1993, Basili et al. 1994c). Kuvassa 28 on esitetty laadunparannusparadigman ja kokemuspajan välinen suhde.

Kuvan 27 mukaisesti mittausryhmä voidaan jakaa kahteen erilliseen ryhmään. GQM-määritysryhmä tulkitsee organisaation johdolta saamansa liiketoimintatavoitteet ja jalostaa niistä projektiryhmältä saatavien vaatimusten ja aiempien kokemuksien mukaisesti mittareita ja menettelytapoja. GQM-analysointiryhmä puolestaan saa projektiryhmältä mittaustietoa, jonka se analysoi ja esittää projektiryhmälle ja mahdollisesti organisaation johdolle.

GQM-analysointiryhmä käsittelee mittausaineistoa ja tarjoaa palautetta projekteille, kun taas GQM-määritysryhmä ylläpitää tietoja, tekee siitä tehokkaasti saatavaa ja kontrolloi siihen pääsyä. Solingen ja Berghout (1999) käsittelevät näitä kahta ryhmää yhtenä mittausryhmänä, jonka tehtäviin kuuluvat sekä määrittely- että analysointitehtävät mittausohjelmissa.



Kuva 28. QIP ja kokemuspaja (Gresse et al. 1995).

Kokemuspaja tukee projektien suoritusta analysoimalla ja yhdistämällä kokemuksia projekteihin. QIP ja kokemuspaja tarjoavat yhdessä GQM-pohjaisen mittauksen kanssa kehyksen projektitavoitteiden asettamisen tukemiseksi suunnitteluvaiheen aikana, mittaustiedon analysoimiseksi ja palautteen antamiseksi meneillään olevaan projektiin. Lisäksi niiden avulla voidaan tallentaa suoritetut mittausohjelmat tulevaa käyttöä varten. Mittauksen tulisi siten olla huolellisesti suunniteltua ja projektin suoritukseen integroitua. Tavoitteena on siis mittausperustainen jatkuva käytäntö.

4.3 Parannuskohteiden etsiminen

Ohjelmistojen kehittämis-, ylläpito- ja tukiprosesseihin liittyy monia eri työryhmiä. Näiden ryhmien tulee käyttää mittausta kuvaamaan, kontrolloimaan ja kehittämään noita prosesseja. Jos organisaatiot kehittävät mittauskehyksiä ilman selviä tavoitteita, voi se johtaa huonosti määriteltyihin mittauskehyksiin ja kerätyn datan hyödyllisyys saattaa jäädä vähäiseksi (Mendonça ja Basili 2000).

Ei ole harvinaista, että organisaatioissa käytetään mittauskehyksiä, jotka keräävät ylimääräistä tietoa, tai tietoa, jota kukaan ei käytä tai tietoa, josta voisi olla hyötyä, mutta jonka olemassaolosta eivät tiedon mahdolliset käyttäjät tiedä. Näistä syistä mittauksen kehittäminen on tärkeää. Organisaatiot tarvitsevat menetelmiä, joiden avulla ne voivat paremmin ymmärtää, parantaa ja jäsentää käynnissä olevaa mittausta ja mittausdataa sekä tutkia paremmin aiemmin kerättyä mittausdataa.

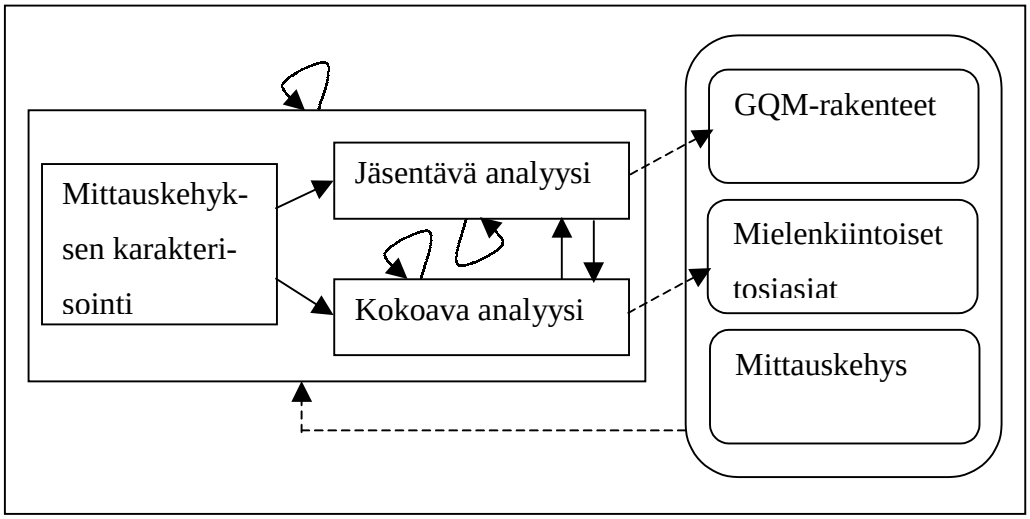
Mendonça ja Basili (2000) esittävät menetelmän, jota voidaan käyttää mittauskehysten parantamiseen, kun organisaatiossa jo kerätään tietoa monenlaisten mittareiden avulla. Menetelmä käyttää GQM-menetelmää käynnissä olevan mittauksen ymmärtämiseen ja jäsentämiseen sekä tiedon käyttäjien tavoitteiden tunnistamiseen. Nämä käyttäjien tavoitteet yhdistetään organisaation käyttämiin mittareihin, jolloin voidaan tunnistaa, mitkä mittarit ovat organisaation kannalta hyödyllisiä ja mitkä eivät. Lisäksi voidaan määrittää, voidaanko asetetut tavoitteet saavuttaa organisaatiossa kerätyn tiedon avulla.

GQM-menetelmän lisäksi Mendonçan ja Basilin (2000) menetelmä käyttää *attribuutinkohdennus* (attribute focusing, AF) –nimistä kokoavaa *tietämyksen muodostamistekniikkaa* (data mining). Tarkoituksena on etsiä jo kerätyistä mittaustiedoista sellaista hyödyllistä tietoa, jonka olemassaolosta tiedon mahdolliset käyttäjät eivät ole tietoisia. Attribuuttien arvojen perusteella lasketut *mielenkiintoiset tosiasiat* (interesting facts) esitetään sitten asiantuntijoille esimerkiksi helposti tulkittavina pylväsdiagrammeina.

Mielenkiintoisille tosiasiasioille on tunnusomaista attribuutin arvojen poikkeaminen jostain odotetusta jakaumasta tai odottamaton korrelaatio attribuuttijoukkojen arvojen välillä (Mendonça ja Basili 2000). Diagrammit lajitellaan mielenkiintoisuusasteen mukaan. *Mielenkiintoisuusasteella* tarkoitetaan numeerista arvoa, joka on laskettu kuvaamaan sitä,

kuinka mielenkiintoinen mikin diagrammi saattaa olla sitä tulkitsevan asiantuntijan mielestä. Mittauskehystä kehitettäessä asiantuntijoita ovat itse mittaustiedon käyttäjät, jotka voivat menetelmän avulla saada enemmän tietoa mitattavista asioista ja mittausprosessin komponenteista eli siitä, kuinka he mittausta suorittavat. Asiantuntija voi löytää uutta tietoa pohtiessaan diagrammien herättämiä kysymyksiä omien tietojensa ja kokemustensa perusteella. Nämä johtopäätökset saattavat johtaa uuden tiedon löytämiseen ja ongelmien korjaamiseen (Bhandari et al. 1993).

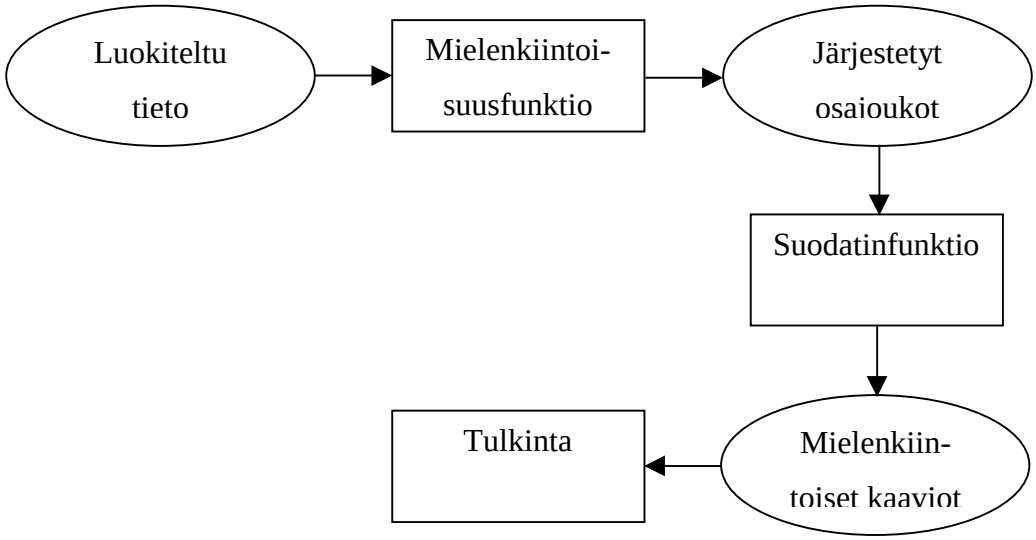
Mendonçan ja Basilin (2000) kehittämä menetelmä koostuu kolmesta vaiheesta, jotka on kuvattu kuvassa 29. Vaiheet ovat mittauskehyksen karakterisointi, jäsentävä analyysi ja kokoava analyysi. *Mittauskehyksen karakterisoinnilla* kuvataan nykyiset ja tulevat tietojen käyttäjäryhmät ja kuinka he käyttävät tai voisivat käyttää tietoja. GQM-paradigmaan pohjautuvan *jäsentävän analyysin* avulla määritetään tiedonkäyttäjien tavoitteet ja yhdistetään ne mittauskehyksen mittareihin ja dataan. *Kokoavan analyysin* avulla etsitään olemassa olevasta mittaustiedosta käyttökelpoista tietoja. Kokoava analyysi suoritetaan attribuutinkohdennuksen avulla. Attribuutinkohdennuksen periaate (Bhandari et al. 1993) on esitetty kuvassa 30.



Kuva 29. Mittauskehysten kehittäminen (Mendonça ja Basili 2000).

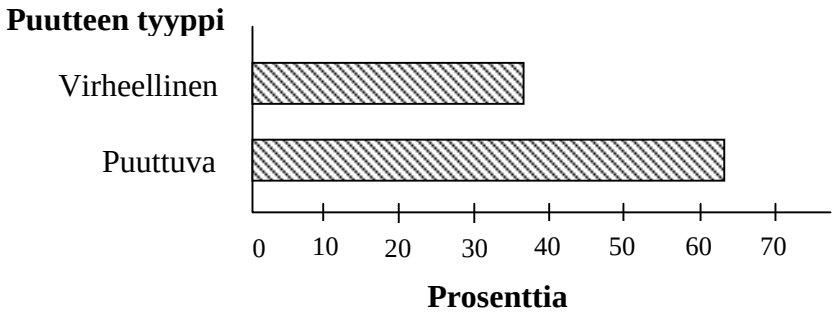
Luokiteltu tieto voi olla esimerkiksi puutteista kerättyä, valittujen attribuuttien avulla (tyyppi, havaitsemisvaihe, aiheutumisvaihe…) luokiteltua lukumäärätietoa. Luokitellusta tiedosta luodut kaaviot järjestetään mielenkiintoisuusfunktion avulla käyttäen soveltuvaa AF-tekniikan mukaista algoritmia. Bhandari et al. (1993) esittelevät lukumäärään ja assosi-

aatioon perustuvat laskentamallit. Mendonça ja Basili (2000) määrittelevät assosiaatioon perustuvan algoritmin. Saadusta järjestetystä joukosta valitaan suodatinfunktion avulla mielenkiintoisimmat kaaviot esitettäväksi asiantuntijoille, jotka sitten tulkitsevat kaavioita.



Kuva 30. AF-tekniikan periaate.

Kuvassa 31 on esitetty puutetiedoista luotu kaavio, jonka mukaan puute voi olla tyypiltään *virheellinen* tai *puuttuva* (Bhandari et al. 1993). Virheellinen tarkoittaa puutetta, joka vaatii korjauksen esim. olemassa olevaan dokumenttiin ja puuttuva merkitsee sitä, että kyseistä asiaa ei ole aiemmin esitetty dokumentissa ollenkaan ja se pitää siten lisätä sinne.



Puuttuva/virheellinen	Havaittu	Odotettu	Erotus
1. Virheellinen	37%	50%	-13%
2. Puuttuva	63%	50%	13%

Kuva 31. Puutteiden jakauma.

Kuvan 31 kaavio on yksi useista kymmenistä vastaavista kaavioista, jotka ovat luotu kerätyn datan avulla luokittelemalla puutteet valittujen attribuuttien avulla (Bhandari et al. 1993). Puutteen tyypin odotusarvo (odotettu) ilmoittaa prosentuaalisen osuuden, jos molemmat olisivat yhtä todennäköisiä. Mielenkiintoisuusfunktio perustuu tässä tapauksessa lukumäärän laskentaan. Mielenkiintoisuusfunktion tuottama poikkeama (erotus) odotetusta jakaumasta ilmaisee kaavion mielenkiintoisuuden ja toimii siten kriteerinä suodatinfunktiolle valittaessa asiantuntijoille esitettäviä kaavioita.

Mendonça ja Basili (2000) ovat validoineet lähestymistapaa käytännön sovelluksena erään organisaation mittauskehyksen kehityksessä ja verranneet sitä siellä aiemmin käytössä olleisiin mittauskehyksen kehitysprosesseihin. Käytännön sovellus osoitti, että GQM- ja AF-menetelmät täydentävät toisiaan. GQM:n avulla voidaan valita ja järjestää tietoja AF:n analyysejä varten. Analyysien tuloksia voidaan puolestaan käyttää GQM-rakenteiden parantamiseen.

5 YHTEENVETO

Ohjelmistoprosessien kehittämisellä saadaan aikaan parempia tuotanto- ja ylläpitoprosesseja. Kehittämistyö ei ole kuitenkaan yksinkertaista, vaan se vaatii tarkkaa suunnittelua ja prosessien tuntemista.

Tässä tutkielmassa on perehdytty tavoitepohjaiseen ohjelmistoprosessien kehittämiseen GQM-paradigman ja ohjelmistomittauksen avulla. Koska ohjelmistojen tuotantoprosessit ovat hyvin erilaisia eri organisaatioissa ja ne voivat olla huonosti määriteltyjä, voidaan tavoitepohjaisen kehittämisen avulla keskittää resurssit tehokkaammin juuri niihin kohteisiin, jotka vaativat eniten kehittämistä.

GQM-paradigma on ollut käytettävissä jo vuodesta 1994 alkaen, mutta käytännön tietoa tai esimerkkejä sen soveltamisesta on ollut vähän saatavilla. Tässä tutkielmassa keskeisenä lähtökohtana ollut Solingenin ja Berghoutin (1999) käytännöllinen näkökulma helpottaa mittausohjelman käyttöönottoa tapauksissa, joissa tavoitepohjaisen mittausohjelman taustalla olevat teoriat eivät ole tuttuja prosessien kehittämisestä vastaaville henkilöille.

Ohjelmistoprosessien kehittämisen aikaansaamaa taloudellista hyötyä on vaikea arvioida, mutta jonkinlaisen kuvan mittausohjelman hyödyistä ja kustannuksista saa luvussa 3 esitetyistä Solingenin ja Berghoutin (1999) kustannusmalleista. Kustannusten suuruus riippuu paljolti asetettujen tavoitteiden määrästä ja siten mittausohjelman laajuudesta. Koska kustannusten määrittäminen on helppoa suhteessa hyötyjen määrittämiseen, on kynnys kehitysaloitteen lopettamiseksi pieni. Tästä syystä on tärkeää, että mittausohjelma perustuu ennalta asetettuihin organisatorisiin tavoitteisiin.

Ohjelmistotuotannon kehittämisen yksi tärkeä painotettava asia on kehittämis- ja mittaustoimien jatkuvuus. Jo mittausohjelmaa luotaessa kannattaa ottaa huomioon mittausohjelman kehittäminen ja kehittämistoimista oppiminen. Mittausohjelman kehittämisellä ja eri vaiheiden automatisoinnilla voidaan kehitystoimissa pitkällä tähtäimellä saada aikaan kustannussäästöjä. Oleellista toiminnan seurannan ja ohjauksen kannalta on tarkoitukseen soveltuva mittauksentukiväline. Tässä tutkielmassa esimerkkinä käytetyn Krögerin (1999) toteuttaman mittauksentukijärjestelmän avulla voidaan määriteltyä prosessia noudattavista

projekteista kerätä puute-, toimintopiste- ja työaikatietoja sekä luoda seurantaraportteja kerättyjen tietojen perusteella. Tarkasteltuun välineeseen ei kuitenkaan ole toteutettu tavoitteiden asettamista eikä mittareiden johtamista kysymysten avulla, vaan nämä vaiheet pitää suorittaa manuaalisesti.

Tutkielman lopussa tutustuttiin laadunparannusparadigmaan, joka perustuu jatkuvaan kokemuksista oppimiseen. Jatkuvalle oppimiselle on tyypillistä, että tieto voidaan varastoida tulevaa käyttöä varten mahdollisimman tehokkaasti. Tällaisena varastona voidaan käyttää kokemuspajaa, jonka tehtävä on ylläpitää ja tarvittaessa tarjota talletettuja kokemuksia projekteihin.

Parantamalla mittauskehyksiä voidaan ohjelmistoprosessien kehittämistoimet keskittää kehittämisen kannalta olennaisiin seikkoihin. Kohdassa 4.3 tarkastellun Mendonçan ja Basilin (2000) laatiman mittauskehysten kehitysmenetelmän tarkoitus ei ole suoranaisesti parantaa mittauskehystä, vaan osoittaa, missä parannuskohteita on. Lisäksi menetelmä auttaa ymmärtämään kerättyä mittaustietoa, mittareita ja sitä, kuinka ne täyttävät tiedon käyttäjien tarpeet. Menetelmän käyttämän AF-tekniikan avulla voidaan aiemmin kerätyistä tiedoista löytää uutta ja mielenkiintoista tietoa mittauksen kohteista ja mittausprosesseista.

Suunnittelemalla mittausohjelma huolellisesti ja aloittamalla kehitystoimenpiteet pienistä, mutta sitäkin olennaisemmista kohteista, voidaan saada aikaiseksi jatkuva kehityskäytäntö, jonka aikaansaamat parannukset ohjelmistotuotannossa esiintyvät näkyvämpinä, paremmin dokumentoituina ja tehokkaampina kehitys-, ylläpito- ja tukiprosesseina. Parannukset näkyvät siten myös mahdollisesti laadukkaampina ja pienemmillä kustannuksilla tuotettuina ohjelmistotuotteina.

VIITELUETTELO

August J. H.: *Joint Application Design – The Group Session Approach to System Design*. Yourdon Press, New Jersey, 1991.

Basili V. R.: Applying the Goal/Question/Metric Paradigm in the Experience Factory, *10th Annual CSR Workshop, Application of Software Metrics and Quality Assurance in Industry*, Amsterdam, 1993. URL: http://www.cs.umd.edu/users/basili/papers.html (05.02.2001).

Basili V. R.: *Software Modeling and Measurement: The Goal/Question/Metric Paradigm*. CS-TR-2956, UMIACS-TR-92-96, University of Maryland, 1992.

Basili V. R., Briand L. ja Morasca S.: *Goal-Driven Definition of Product Metrics Based on Properties*. CS-TR-3346, UMIACS-TR-94-106, University of Maryland, 1994a. URL: http://www.cs.umd.edu/users/basili/papers.html (05.02.2001).

Basili V. R., Caldiera G. ja Rombach H. D.: The Goal Question Metric approach. *Encyclopedia of Software Engineering*, John Wiley & Sons, 1994b, 528-532. URL: http://www.cs.umd.edu/users/basili/papers.html (05.02.2001).

Basili V. R., Caldiera G. ja Rombach H. D.: The Experience Factory. *Encyclopedia of Software Engineering*, John Wiley & Sons, 1994c, 469-476. URL: http://www.cs.umd.edu/users/basili/papers.html (05.02.2001).

Basili V. R. ja Rombach H. D.: *The TAME Project: Towards Improvement-Oriented Software Environments*, CS-TR-1983, UMIACS-TR-88-8, University of Maryland, 1988.

Bhandari I., Halliday M., Tarver E., Brown D., Chaar J. ja Chillarege R.: A Case Study of Software Process Improvement During Development. *IEEE Transactions On Software Engineering*, 19(12), 1993, 1157-1170.

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

Christie A. M.: Enactable process specification in support of process-centred environments. *Information and Software Technology*, 36(11), Butterworth-Heinemann Ltd., 1994, 655-666.

DeMarco T.: *Controlling Software Projects: Management, Measurement & Estimation*. Prentice Hall, Inc., USA, 1982.

Dreger J. B.: *Function Point Analysis*. Prentice Hall, New Jersey, 1989.

Fenton N. E. ja Neil M.: Software Metrics: Roadmap. *The Future of Software Engineering* (ed. Finkelstein A.). 22nd International Conference on Software Engineering, Limerick, Ireland, 2000, 357-370.

Fenton N. E. ja Pfleeger S. L.: *Software Metrics: A Rigorous and Practical Approach*. PWS Publishing Company, USA, 1997.

Florac W. A., Park R. E. ja Carleton A. D.: *Practical Software Measurement: Measuring for Process Management and Improvement.* CMU/SEI-97-HB-003, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, 1997.

Fuggetta A.: Software Process: A Roadmap. *The Future of Software Engineering* (ed. Finkelstein A.). 22nd International Conference on Software Engineering, Limerick, Ireland, 2000, 27-34.

Garmus D. ja Herron D.: *Function Point Analysis: Measurement Practices for Successful Software Projects*. Addison-Wesley, Boston, 2001.

Grady R. B. ja Caswell D.L.: *Software Metrics: Establishing a Company-wide program*. Prentice-Hall, New Jersey, 1987.

Gray A. ja MacDonell S.G.: GQM++ A full life cycle framework for the development and implementation of software metric programs. *Proceedings of ACOSM'97 Fourth Australian Conference on Software Metrics.* Canberra, Australia, 1997, 22-35.

Gresse G., Hoisl B. ja Wüst J.: *A Process Model For GQM-Based Measurement.* STTI-95-04-E. Software-Technologie-Transfer-Initiative, Universität Kaiserslautern, Kaiserslautern, 1995.

Humphrey W. S.: *A Discipline for Software Engineering.* Addison-Wesley, Reading, Massachusetts, 1995.

Humphrey W. S.: *Introduction to the personal software process.* Addison-Wesley, Reading, Massachusetts, 1997.

Humphrey W. S.: *Managing the software process.* Addison-Wesley, Reading, Massachusetts, 1989.

Kan S. H.: *Metrics and models in software quality engineering*. Addison-Wesley, New York, 1995.

Krishnan M.S. ja Kellner M. I.: Measuring Process Consistency: Implications for reducing software defects. *IEEE Transactions on software engineering*, 25(6), 1999, 800-815.

Kröger E.: *Process Management System*. Erikoistyö, Tietojenkäsittelytieteen laitos, Joensuun yliopisto, 1999.

Kuvaja P., Similä J., Krzanik L., Bicego A., Koch G. ja Saukkonen S.: *Software Process Assessment ja Improvement, The BOOTSTRAP Approach*. Blackwell Publishers, UK, 1994.

Latum F. van, Solingen R. van, Oivo M, Hoisl B., Rombach D. ja Ruhe G.: Adopting GQM-based measurement in an industrial environment. *IEEE Software*, 15(1), 1998, 78-86.

Mashiko Y. ja Basili V. R.: Using the GQM Paradigm to Investigate Influential Factors for Software Process Improvement. *Journal of Systems Software*, 36, Elsevier Science Inc., 1997, 17-32.

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

Minkowich C.: Formal Process Modeling. *Information and Software Technology*, 35 (11/12), Butterworth-Heinemann Ltd., 1993, 659-667.

Moser S.: Metamodels for Object-Oriented Systems: A Proposition of Metamodels Describing Object-Oriented Systems at Consecutive Levels of Abstraction. *Software-Concepts and Tools*, 16, Springer-Verlag, 1995, 63-80.

Park R. E., Goethert W. B., ja Florac W. A.: *Goal-Driven Software Measurement: A Guidebook*. CMU/SEI-96-HB-002, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, USA, 1996.

Paulk M. C.: The Evolution of the SEI's Capability Maturity Model for Software. *Software Process - Improvement and Practice*, Pilot Issue, John Wiley & Sons And Gauthier-Villars, 1995, 3-15.

Paulk M. C., Weber C. V., Curtis B., Chrissis M. B.: *The Capability Maturity Model: Guidelines for Improving the Software Process*. Addison-Wesley, Reading, Massachusetts, 1995.

Pfleeger S. L.: Maturity, Models, and Goals: How to Build a Metrics Plan. *Journal of Systems Software*, 31, Elsevier Science Inc., 1995, 143-155.

Pfleeger S. L.: *Software Engineering: Theory and Practice*. Prentice Hall, USA, 1998.

Pressman R. S.: *Software Engineering, A Practitioner's Approach.* McGraw-Hill, Lontoo, 1994.

Rout T. P.: SPICE: A Framework for Software Process Assessment. *Software Process-Improvement and Practice*, Pilot Issue, John Wiley & Sons And Gauthier-Villars, 1995, 57-66.

Solingen R. van ja Berghout E.: *The Goal/Question/Metric Method: a practical guide for quality improvement of software development*. McGraw-Hill Publishing Company, 1999.

The ami Handbook: *A Quantitative Approach to Software Management*. The ami Consortium, South Bank Polytechnic, London, England, 1992.

Wieczorek I.: On the establishment of successful measurement programs. *Software Process - Improvement and Practice*, 3, John Wiley & Sons, 1997, 191-194.