Six Sigma ohjelmistokehityksessä

Harri Pirttinen

Tampereen yliopisto
Tietojenkäsittelytieteiden laitos
Tietojenkäsittelyoppi
Pro gradu -tutkielma
Ohjaajat: Eleni Berki, Timo Poranen
Marraskuu 2010
Tutkielman tarkoituksena on selvittää, mikä Six Sigma on ja miten sitä voisi hyödyntää ohjelmistokehityksessä ohjelmistotuotantoprosessin ja laadun kehittämiseen. Six Sigma on levinnyt muilla teollisuudenaloilla prosessinkehityksessä ja laadunkehityksessä, eikä sen soveltamisesta ohjelmistokehitykseen ole tehty kattavaa tutkimusta.

Tutkimuksessa perehdytään Six Sigmaan kirjallisuuden ja toisten tutkijoiden löydettyjen perusteella. Ohjelmistokehityksen sovellustapoja kootaan lähteistä ja niiden käytön hyödyllisyttä arvioidaan.

Työssä havaittiin, että Six Sigma on sovellettavissa tietyiltä osilta ohjelmistokehitykseen, mutta se ei ole hopealuoti parempaan ohjelmistolaatuun. Se voi kuitenkin ajaa organisaatiota tilaan, jossa organisaatio on tietoinen tuotettujen virheiden syistä ja täten auttaa organisaatiota tekemään oikeita korjausliikkeitä. Tutkielmassa tehtiin havaintoja Six Sigman hyödyntämisestä Scrum-kehitysmalliin ja yksikkötestaukseen.

Tutkielmassa käsitellystä aiheesta on vähän tutkimustyötä ja tässä tutkielmassa esille tulleet ehdotukset ja pohdinnat vaatisivat lisätutkimusta, jotta niiden todellinen hyöty voitaisiin todentaa.

Avainsanat ja -sanonnat: Six Sigma, ohjelmistolaatu, ohjelmistotuotantoprosessin kehitys, CMMI, Scrum, PSP/TSP, yksikkötestaus, tilastollinen analyysi.
Sisällysluettelo

1. Johdanto .................................................................................................................................................. 1

2. Ohjelmiston laatu .................................................................................................................................. 2
   2.1. Ohjelmistolaatu ............................................................................................................................. 2
   2.2. Ohjelmistotuotantoprosessin kehittämisyn ja ohjelmistolaadun suhde ........................................... 3
   2.3. Prosessimittarit ja prosessin mitä ................................................................................................ 4
   2.4. SIPOC-analyysi ............................................................................................................................ 5
   2.5. Virhekausaalianalyysi (Defect Causal Analysis) .......................................................................... 6
       2.5.1. Five Whys ............................................................................................................................. 6
       2.5.2. Ishikawa-kaavio .................................................................................................................... 6
       2.5.3. Pareto-analyysi ..................................................................................................................... 7
   2.6. Kano-malli .................................................................................................................................... 8
   2.7. Katselmointi ja auditointi ............................................................................................................. 9
   2.8. Koodirivimäärä mittarina ............................................................................................................. 10
   2.9. Tilastotieteen käsitteitä ................................................................................................................ 10
       2.9.1. Keskiarvo .................................................................................................................................. 10
       2.9.2. Normaalijakauma .................................................................................................................. 10
       2.9.3. Keskihajonta ......................................................................................................................... 11
       2.9.4. T-testi ..................................................................................................................................... 11

3. Ohjelmistojen kehitysmallit ja laatujärjestelmät ............................................................................... 12
   3.1. Total Quality Management .......................................................................................................... 12
   3.2. Lean-malli .................................................................................................................................... 13
   3.3. CMM ja CMMI ............................................................................................................................ 13
       3.3.1. Capability Maturity Model .................................................................................................... 13
       3.3.2. Capability Maturity Model Integration ................................................................................... 16
       3.3.3. SW-CMM -mallin ja CMMI-mallin ero .................................................................................... 19
   3.4. Henkilökohtainen ja ryhmäohjelmistoprosessi, PSP/TSP ............................................................ 20
   3.5. Vesiputouksmalli .......................................................................................................................... 23
   3.6. Scrum ........................................................................................................................................... 24
3.7. Automaattinen yksikkötestaus .................................................................................. 25

4. Six Sigma ..................................................................................................................... 27
4.1. Six Sigma toimintamallit .......................................................................................... 29
4.1.1. Define, Measure, Analyze, Improve, Control (DMAIC) ........................................ 29
4.1.2. Design For Six Sigma (DFSS) .............................................................................. 33
4.2. Six Sigma –metodologian keskeiset mittarit ja analyysi ......................................... 35
4.2.1. Sigma-taso ......................................................................................................... 35
4.2.2. Prosessin suorituskyky .................................................................................... 38
4.2.3. Six Sigma –mittareiden vaatimukset ................................................................. 39
4.3. Six Sigma -projektin roolit .................................................................................... 40
4.3.1. Champion ........................................................................................................ 40
4.3.2. Black Belt ......................................................................................................... 40
4.3.3. Green belt ...................................................................................................... 41
4.3.4. Yellow belt ...................................................................................................... 41

5. Six Sigma ja ohjelmistotuotanto .................................................................................. 42
5.1. Kritiikkiä Six Sigman soveltuvuudesta ohjelmistotuotantoon ................................. 42
5.2. Six Sigman soveltaminen ohjelmistotuotantoon ....................................................... 44
5.2.1. Six Sigman soveltaminen raportoitujen virheiden perusteella ............................... 45
5.2.2. Laadunkehitys henkilökohtaisen ja ryhmäohjelmistoprosessin perusteella ............ 48
5.2.3. Asiantunnetta prosessin kehittämisen ............................................................... 50
5.3. Six Sigma ja CMMI ................................................................................................ 54
5.4. DFSS-toimintamallin soveltaminen ohjelmistoprojektiin ......................................... 55
5.5. Six Sigma ja ketterät menetelmät .......................................................................... 56
5.6. Six Sigma ja automatisoitu yksikkötestaus .............................................................. 58
5.7. Johtopäätöksiä Six Sigman soveltamisesta ohjelmistokesitykseen ....................... 59

6. Yhteenveto ................................................................................................................... 61
1. Johdanto


2. **Ohjelmiston laatu**

Laadun kehittämiseen on kehitetty useita mittareita ja työkaluja. Tässä luvussa esitellään ohjelmistolautaua ja sen kehittämiseen liittyviä käsitteitä.

2.1. **Ohjelmistolautau**


Ohjelmiston laatuun sisältyy muitakin kuin aistittavia ominaisuuksia. Ohjelmiston laadun ominaisuuksia on määritelty erilaisten standardien muodossa. Esimerkkinä mainittakoon ISO 9126, jossa määritellään laatukäsitteet, kuten siirrettävyys (portability), tehokkuus (efficiency), luotettavuus (reliability), toiminnallisuus (functionality), käytettävyys (usability), ylläpidettävyys (maintainability) ja laajennettavuus (extensibility) [Al-Kilidar et al., 2005].


Luotettavuus liittyy järjestelmän toimintakykyyn tietyllä aikavälillä, jonka aikana järjestelmän tulisi toimia mahdollisimman virheettömästi ja toipua virheistä menettämättä toimintakykyään.

Käytettävyyys liittyy järjestelmän käytettävyyteen liittyviin tekijöihin. Käytettävyyteen kuuluvat mm. käytön ymmärrettävyys, oppimisnopeus, käytön tehokkuus ja käyttöliittymän puoleensavetävyys.

Ylläpidettävyyys liittyy järjestelmään tehtävien muutosten työmääriään ja vaikeuteen. Mitä ylläpidettävämpi järjestelmä on, sitä nopeammin järjestelmään voidaan tehdä muutos, joka ei tuo mukanaan uusia järjestelmävirheitä.

Laajennettavuus liittyy järjestelmään tehtävien muutosten työmääriään ja vaikeuteen. Mitä laajennettävämpi järjestelmä on, sitä mahdollisimman pienemmäksi virhetta voidaan tehdä.

Käytettävyyksia voidaan mittata: käytön ymmärrettävyys, oppimisnopeus, käytön tehokkuus ja käyttöliittymän puoleensavetävyys.

Ylläpidettävyyksia voidaan mittata: järjestelmään tehtävien muutosten muuttoa vaikeuteen. Mitä ylläpidettävämpi järjestelmä on, sitä mahdollisimman pienemmäksi virhetta voidaan tehdä.

Laajennettavuus voidaan mittata: järjestelmään tehtävien muutosten muuttoa vaikeuteen. Mitä laajennettävämpi järjestelmä on, sitä mahdollisimman pienemmäksi virhetta voidaan tehdä.

Laajennettavuus voidaan mittata: järjestelmään tehtävien muutosten muuttoa vaikeuteen. Mitä laajennettävämpi järjestelmä on, sitä mahdollisimman pienemmäksi virhetta voidaan tehdä.

Laajennettavuus voidaan mittata: järjestelmään tehtävien muutosten muuttoa vaikeuteen. Mitä laajennettävämpi järjestelmä on, sitä mahdollisimman pienemmäksi virhetta voidaan tehdä.

Laajennettavuus voidaan mittata: järjestelmään tehtävien muutosten muuttoa vaikeuteen. Mitä laajennettävämpi järjestelmä on, sitä mahdollisimman pienemmäksi virhetta voidaan tehdä.

Laajennettavuus voidaan mittata: järjestelmään tehtävien muutosten muuttoa vaikeuteen. Mitä laajennettävämpi järjestelmä on, sitä mahdollisimman pienemmäksi virhetta voidaan tehdä.
Ohjelmistotuote ei ole kosketeltavissa, koska se on immateriaalituote. Tämän takia tuotetta ei voi tutkia tai mitata fyysisesti. Tämä tekee myös tuotteen todellisen valmiasasteen määrittämisestä vaikeaa.

Ohjelmistotuotantolle ei ole standardia ohjelmistotuotantoprosessia. Ohjelmistotuotantoprosessit ovat organisaatiokohtaisia ja voivat poiketa toisistaan huomattavasti.

Suuremmat ohjelmistotuotantoprojektit ovat ainutlaatuisia ja ainutkertaisia. Suurista projekteista opittuja asioita ei pakosta voi hyödyntää toisissa projekteissa, johtuen niiden ainutlaatuisuudesta.


Koska korkea ohjelmistolaatu on ohjelmistokehityksessä kilpailuetu, on ohjelmistokehitysprosessin kehittäminen kohti korkeampaa laatua perusteltua. Tämä seikka on tunnistettu yleisesti ja ohjelmistotuotantoprosessin kehittämiseen on kehitetty useita eri malleja.

Ohjelmistotuotantoprosessi on mallinnettavissa, jaksoettavissa ja ennen kaikkea, mitattavissa. Mittarit ovat yksiselitteisiä havaintoja prosessin toiminnasta, eli ne kertovat prosessin tehokkuuden ja sen kehittymisen tai rappeutumisen.

2.3. Prosessimittarit ja prosessin mitaus


Prosessimittarit ovat yksiselitteisiä numeerisia arvoja, jotka tuottavat prosessivaiheen toiminnasta analysoitavaa dataa. Prosessimittareista on tunnistettu kolme keskeistä ryhmää: laadun kannalta kriittiset (Critical To Quality), kustannusten kannalta kriittiset (Critical To Cost) ja aikataulun kannalta kriittiset mittarit (Critical To Schedule) [Keller, 2005, s. 87–93]. Laadun kannalta kriittiset mittarit riippuvat...

2.4. SIPOC-analyysi


<table>
<thead>
<tr>
<th>Toimittajat</th>
<th>Syötteet</th>
<th>Prosessi</th>
<th>Tulosteet</th>
<th>Asiakkaat</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Tarvittavien resurssien toimittajat)</td>
<td>(Prosessin vaatimat resurssit)</td>
<td>(Ylimmän tason kuvaus toiminnasta)</td>
<td>(Prosessista syntyneet tuotokset)</td>
<td>(Tuotosten vastaanottajat)</td>
</tr>
<tr>
<td>Ohjelmistokehittäjät</td>
<td>Henkilöstö, kalusto, suunnitelmat, määrittely ja vaatimukset</td>
<td>Ohjelmistotuotantoprosessi</td>
<td>Ohjelmistotuote, dokumentaatio</td>
<td>Loppukäyttäjät</td>
</tr>
</tbody>
</table>

Taulukko 1. SIPOC-analyysi [Redzic and Baik, 2006].
2.5. Virhekausaalianalyysi (Defect Causal Analysis)

Virheistä oppiminen on kehittymisen edellytys ihmisen toiminnassa. Jotta virheestä voi oppia, se on tunnistettava ja selvitettävä syy, josta virhe syntyi. Tästä syy-seuraussuhteen analysoinnista on kehitetty projektityökalu nimeltä virhekausaalianalyysi.

Virhekausaalianalyysi keskittyy syy-seuraussuhteiden ymmärtämiseen. Tällä systemaattisella menetelmällä on tarkoitus tunnistaa ja selvitää syy, josta virhe syntyi. Tästä syysuhteellisuuden analysoinnista on kehitetty projektityökalu nimeltä virhekausaalianalyysi.

Cardin [2006] tunnisti kolme ehtoa syy-seuraussuhteelle:

1. Ehdotetulla syyllä ja seurauksella tulee olla assosiaatio tai korrelaatio.
2. Syyn tulee edeltää seurausta ajallisesti.
3. Syyn ja seurauksen yhdistävämekanismi pitää pystyä tunnistamaan.


2.5.1. Five Whys


2.5.2. Ishikawa-kaavio

Virhekausaalianalyysin ehkä tunnetuin työkalu on Ishikawa-kaavio, joka tunnetaan myös nimellä kalanruotokaavio (fishbone diagram) [Kelley, 2000]. Ishikawa-kaaviossaan ongelma merkitään vaakaviivalla, johon liitetään viistoilla viivoilla havaitut pääsyyt

**Kuva 1. Ishikawa-kaavio.**

2.5.3. Pareto-analyysi


**Kuva 2. Pareto-kaavio.**
2.6. **Kano-malli**


1. **Pakollinen laatu (Must Be)** sisältää ominaisuuksia, joiden huono toiminta tai puuttuminen laskee laatua. Ominaisuuksien keskimääräistä parempi toteutus ei kuitenkaan nosta asiakastyytyväisyyttä.
2. **Yksiulotteinen laatu (One-dimensional)** sisältää ominaisuuksia, joissa asiakastyytyväisyys kasvaa lineaarisesti toteutuksen laadun kanssa.
4. **Yhdentekevä (Indifference)** laatu sisältää ominaisuuksia, joiden keskimääräistä parempi toteutus ei nosta asiakastyytyväisyyttä.
5. **Käänteinen (Reversal)** laatu liittyy ominaisuuksiin, joiden laadukkaampi toteutus laskee asiakastyytyväisyyttä.

<table>
<thead>
<tr>
<th>Ominaisuuden täysi toteutus</th>
<th>V1</th>
<th>V2</th>
<th>V3</th>
<th>V4</th>
<th>V5</th>
</tr>
</thead>
<tbody>
<tr>
<td>V1</td>
<td>L6</td>
<td>L1</td>
<td>L1</td>
<td>L1</td>
<td>L2</td>
</tr>
<tr>
<td>V2</td>
<td>L5</td>
<td>L4</td>
<td>L4</td>
<td>L4</td>
<td>L3</td>
</tr>
<tr>
<td>V3</td>
<td>L5</td>
<td>L4</td>
<td>L4</td>
<td>L4</td>
<td>L3</td>
</tr>
<tr>
<td>V4</td>
<td>L5</td>
<td>L4</td>
<td>L4</td>
<td>L4</td>
<td>L3</td>
</tr>
<tr>
<td>V5</td>
<td>L5</td>
<td>L5</td>
<td>L5</td>
<td>L5</td>
<td>L6</td>
</tr>
</tbody>
</table>


Chenin ja Chuangin [2008] mukaan ominaisuuksien laatukartoitus toteutetaan kahdella kyselyllä, joista ensimmäisessä oletetaan, että tuote sisältää halutun
ominaisuuden tai asiakasvaatimuksen. Asiakas vastaa kysymyksiin asteikolla "tyytyväinen" ("satisfied"), "asian pitäisi olla näin" ("it should be that way"), "yhdentekevää" ("I am indifferent"), "pystyn elämään asian kanssa" ("I can live with it") ja "tyytymätön" ("dissatisfied"). Toisessa kyselyssä asettelu käännetään päinvastoin, eli asiakas vastaa samalla asteikolla oletuksiin, että ominaisuus tai asiakasvaatimus puuttuu tuotteesta tai on vaillinainen. Kyselyjen tulokset analysoidaan kano-taulukolla, jonka avulla ominaisuudelle voi asettaa laatukategorian. Kano-taulukko on kuvattuna kuvassa 3. Taulukossa kahden eri kyselykierroksen tulokset paljastavat laatukategorian siten, että riveillä on ensimmäisen kyselyn vaihtoehdot, missä ominaisuus on toteutettu ja sarakkeissa toisen kyselyn vaihtoehdot, missä ominaisuuden toteutus on vaillinainen. Esimerkiksi, jos asiakas on tyytyväinen toteutukseen (V1) ja tyytyväinen siihen, että toteutus on vaillinainen (V1), niin kyseessä on ristiriitainen laatukategoria (L6). Laatukategoria on sama (L6), jos asiakas vastaa kummallakin kierroksella olevansa tyytymätön (V5).

2.7. Katselmointi ja auditointi
Katselmointi on laadunhallintaan kuuluva aktiviteetti, jossa katselmoinnin kohteena oleva toteutus tutkitaan ja validoidaan ohjelmistokehittäjien ja muiden tarpeellisten asianomaisten osalta. Katselmoinnit ovat muodollisia tilaisuuksia, joiden tarkoituksena on löytää virheet toteutuksesta, varmistaa asiakasvaatimuksen täyttyminen karkasteltavan toteutuksen osalta, varmistaa, että ohjelmistotuote täyttää määrätyt standardit, varmistaa ja että ohjelmisto kirjoitetaan yhtenäisellä tavalla [Pressman, 2005, s. 754].


- Toteutus hyväksytään ilman muutoksia.
- Toteutus hylätään vakavien virheiden takia (vaatii uuden katselmoinnin).
- Toteutus hyväksytään pienin korjausvaatimuksin (ei vaadi uutta katselmointia).

Tiedonjakamisen kannalta katselmoinnit toimivat kehittejille tilaisuuksina oppia tuntemaan järjestelmän toiminnallisuudet, joita he eivät ole olleet toteuttamassa.

2.8. Koodirivimäärä mittarina

2.9. Tilastotieteen käsitteitä
Tässä kappaleessa esitellään tilastotieteen käsitteitä, joiden ymmärtäminen on edellytyksenä Six Sigman metodologian periaatteiden käsittämiselle. Tässä kappaleessa esiteltävät käsitteet kuuluvat suurimmaksi osaksi tilastotieteen perusteihin, joten käsitteiden ymmärtäminen ei vaadi syvempää tilastotieteen tuntemusta.

2.9.1. Keskiarvo

2.9.2. Normaalijakauma
Normaalijakauma on jatkuva jakauma, joka on kaikkein yleisimmin käytetty ja tämä jakauma tilastotieteessä [Grönroos, 2003]. Syy yleisyyteen on se, että jakauman käyttö voidaan perustella keskeisellä raja-arvolauseella. Tämän takia normaalijakaumaan
perustuvia malleja käytetään aina, kun niiden käyttö voidaan perustella empiirisesti tai teoreettisesti [Grönroos, 2003]. Suurin hankaluus, joka normaalijakaumaan liittyy, on se, että jakauman kertymäfunktio ei ole esitettävissä suljetussa muodossa. Tunnettuja riittävän tarkkoja laskukaavoja on useita, joten tietokoneita hyödyntämällä todennäköisyydet ovat helposti laskettavissa. Yleisimmän hyödynnetyn normaalijakauman, standardinormaalijakauman, arvot löytyvät taulukkokirjoista positiivisten pisteiden osalta.

2.9.3. Keskihajonta
Keskihajonta on luku, joka kuvaa arvojen ryhmittymistä keskiarvonsa ympärille. Mitä lähemmäs keskiarvoa ja siis myös toisiaan arvot ovat, sitä pienempi on keskihajonta. Hajallaan sijaitsevien, eli keskenään kovin eri suurien lukujen keskihajonta on iso. Tilastotieteessä keskihajontaa merkitään kreikkalaisella kirjaimella kirjaimella sigma (σ), mutta myös kirjaimella s. Keskihajonnan kaava on kuvattu kaavassa 1.

\[
\sigma = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \mu)^2}{n-1}}
\]

\(\mu=keskiarvo\)
\(N=population\ koko\)


2.9.4. T-testi
Studentin t-testi on tilastollinen testi, joka noudattaa t-jakaumaa, kun nollahypoteesi on voimassa [Grönroos, 2003]. Nollahypoteesi on tilastotieteellisessä tutkimuksessa tehtävä oletelmus, jonka mukaan tutkittavien ryhmien välillä ei ole perustavaa eroa, vaan ryhmien välillä olevat pienet erot johtuvat otannan satunnaisvaihtelusta. Vaikka tässä tutkielmassa ei paneuduta edellä mainitun syvemmin, t-testin käytöstä voidaan todeta, että sillä voi testata normaalijakautuneiden satunnaismuuttujien keskiarvoja. Yksi käyttötapa on verrata kahden riippumattoman otoksen keskiarvojen yhtäsuuruutta.
3. **Ohjelmistojen kehitysmallit ja laatujärjestelmät**

Kehitysmallit ja organisaatiota ohjaavat toimintamallit ovat tärkeitä tuotettavan laadun kannalta, koska ne ohjaavat ohjelmistotuotantoprosessin suoritusta. Tässä luvussa esitellään kehitysmalleja, jotka liittyvät suoraan ohjelmistotuotantoprosessiin tai muuten organisaation toiminnan ohjaamiseen.

3.1. **Total Quality Management**

Total Quality Management on integroitu lähestymistapa, jolla tavoitellaan lopputuotteen korkean laadun saavuttamista ja laadun ylläpitämistä keskittämällä prosessien huoltoon, jatkuvaan kehittämiseen ja virheiden ennaltaehkäisyyn kaikilla tasoilla ja organisaation toiminnassa, jotta asiakasvaatimukset voidaan täyttää tai ylittää [Boaden, 1997].


- Asiakaslähtöisyys.
- Ylimmän johdon vastuu laadunkehitysprosessista.
- Työntekijöiden ja heidän tietämyksen kunnioittaminen.
- Työntekijät ovat aktiivisesti mukana laadunkehitysprosessissa.
- Prosessi- ja tuotehajonnan vähentäminen.
- Työntekijöiden jatkuvan koulutuksen ja harjoittelun mahdollistaminen.
- Tilastollisen ajattelutavan sisäänajo ja tilastollisten metodien hyödyntäminen koko organisaatiossa.
- Painoarvo virheiden ehkäisyllä tunnistamisen sijaan.
- Toimittajien näkeminen pitkääikaaisina yhteistyökumppaneina.
- Suorituskyvyn mittarit, jotka ovat johdonmukaiset organisaation tavoitteiden kanssa.
- Pitäytyminen parhaissa toimintatavoissa tehtävien suorittamiseen.
- Painoarvo laadulla tuotteiden ja palveluiden suunnittelussa.
- Organisaation kaikkien toimintojen yhteistyö ja osanotto.
- Sisäisten asiakkaiden tarpeiden tiedostaminen.
- Huomattava organisaatiokulttuurillinen muutos.
3.2. Lean-malli


Lean-mallin liittyvyys arvovirtojen kartoittaminen, jossa tunnistetaan tekijät, jotka organisaation toiminnassa lisäävät arvoa lopputulokseen [Chaneski, 2002]. Jos sisäinen hukka vähenee, hukkahuutattavat toiminnot ja niihin liittyvät kustannukset vähenevät ja asiakkaalle tarjolla oleva kokonaisarvo lisääntyy [Hines et al., 2004].

3.3. CMM ja CMMI

Software Engineering Institute on kehittänyt kaksi tunnettua kypsyysmallia, Capability Maturity Model –kypsyysmallin (CMM) sekä Capability Maturity Model Integration – kypsyysmallin (CMMI). Tässä kappaleessa esitellään edellä mainitut kaksi kypsyysmallia ja niiden keskeiset erot.

3.3.1. Capability Maturity Model

CMM-kypsyysmalli julkaistiin vuonna 1989 ja mallia kehitettiin vuoteen 1997 asti. CMM:n alkuperäinen tarkoitus oli toimia työkaluna ohjelmistotoimittajien toimituskyvyn arvioinnissa. CMM-mallia on erityisesti käytetty ohjelmistotuotantoprosessin kypsyysmallina. CMM-mallista kuitenkin kehittyi muille aloille erikoistuneita mallia, joten ohjelmistokehitykselle luotiin oma malli, jolle käytettiin tunnistetta SW-CMM (SoftWare-CMM).

Malli sisältää viisi kypsyystasoa, jotka ovat kuvattuna kuvassa 4. Kuvasta ilmenee, mitkä avainprosessialueet pitää toteuttaa eri tasoja tavoiteltaessa. Kypsyystasot koostuvat avainprosessialueista (Key Process Area), joita CMM-mallissa on yhteensä 18 kappaletta [SEI, 2002]. Jokaiselle avainprosessialueelle määritellään tavoitteet (Goals), sitoutuminen (Commitment), kyvykkys (Ability), mittaaaminen (Measurement) ja todentaminen (Verification). Avainprosessialueiden sisällöt ovat seuraavat [SEI, 2002]:

- **Konfiguraatiohallinta**: Kaikkien projektin tuotteiden eheyden ylläpitäminen. Tämä tehdään seuraamalla ja hallinnoimalla artefaktien muutoksia ja versioiden hallintaa.
- **Laadunvarmistus**: Tuotteiden laatua on varmistettava, jotta ne täyttävät laatuvaatimuksen ennen kuin ne luovutetaan asiakkaille. Tämä prosessialue kattaa laadunvarmistuksen kaiken tuotettavan osalta.
Alihankkijoiden hallinta: Alihankkijasuhteita hallinnoidaan yllätysten ja riskien välttämiseksi. Lisäksi on tärkeää varmistaa, että alihaankija on kykenevä toimittamaan sovitusti, eli varmistaa aikataulullisten riippuvuuksien toteutuminen.

CMM taso 5, optimointi
- Vikojen ehkäisy (Defect Prevention)
- Teknologian muutoshallinta (Technology Change Management)
- Prosessien muutoshallinta (Process Change Management)

CMM taso 4, johdettu prosessi
- Kvantitatiivinen prosessinhallinta (Quantitative Process Management)
- Ohjelmistokehityksen laadunhallinta (Software Quality Management)

CMM taso 3, määritelty prosessi
- Organisatorisiin prosesseihin keskittyminen (Organizational Process Focus)
- Organisatoristen prosessien määrittäminen (Organizational Process Definition)
- Kouluutusohjelma (Training Program)
- Integroidu ohjelmistohallinta (Integrated Software Management)
- Ohjelmistotuotteen suunnittelu (Software Product Engineering)
- Ryhmien välisen yhteistyön toteutaminen (Intergroup Coordination)
- Katselmoinnit (Peer Review)

CMM taso 2, toistettavissa oleva prosessi
- Vaativuus hallinta (Requirements Management)
- Ohjelmistoprosessin suunnittelu (Software Project Planning)
- Projektiseuranta ja hallinta (Project Monitoring and Control)
- Alihankkijaiden hallinta (Software Subcontract Management)
- Laadunvarmistus (Software Quality Assurance)
- Konfiguraatiohallinta (Software Configuration Management)

CMM taso 1, aloitus
- Ei prosessialueita

Kuva 4. SW-CMM -mallin kypsymystasot ja tasojen avainprosessialueet [SEI, 2002].


Ohjelmistoprosesin suunnittelu: Projektiin suunnittelu, missä arvioidaan tarvittava aika ja kustannus projektille. Työ pilkotaan pienempiin kokonaisuuksiin ja jokaiselle kokonaisuudelle tulisi määrittää resurssit ja aikataulu.
Vaatimusten hallinta: Tunnistettuja vaatimuksia tarvitsee hallinnoida. Tehtävän työn, projektisuunnitelman ja vaatimusten välillä ei saa olla epäjohdonmukaisuuksia.

Katselmoinnit: Verifioidaan, että toteutus vastaa määritettyjä vaatimuksia jokaisessa projektiin vaiheessa.

Ryhmien välinen yhteistyö: Kaikkien olennaisten asianomaisten sekä projektiin jäsenten on pystytettävä tekemään tarpeen mukaan yhteistyötä. Tämä prosessialue kattaa yhteistyön mahdollistamisen kaikissa projektiin eri vaiheissa.

Ohjelmistotuotteen suunnittelu: Siisältää asiakasvaatimuksia täyttävien ratkaisujen suunnittelun. Tämä sisältää korkean ja matalan tason suunnittelun, eli modulisuunnittelusta arkkitehtuurisuunnitteluun.

Integroitu ohjelmistohallinta: Tarkoituksena on räätälöidä organisaation ohjelmistotuotantoprosessista ja vahvuusista prosessi, jossa ohjelmistosuunnittelu ja hallintotoiminnat integroituvat yhdessä.

Koulutusohjelma: Tarkoitus on tunnistaa koulutustarpeet yksilötasolla organisaatiotasolle ja tarjota tarvittava koulutus.

Organisatoristen prosessien määrittäminen: Tarkoitus on kehittää ja ylläpitää käytettävää joukko ohjelmistotuotantoprosessiin liittyviä vahvuuksia, jotka kehittävät prosessin tehokkuutta.

Organisatorisiin prosesseihin keskittyminen: Tarkoituksena on perustaa organisatoriset vastuut ohjelmistotuotantoprosessin toiminnaille, jotka kehittävät koko organisaation kyvykkyyttä.

Ohjelmistokehityksen laadunhallinta: Laadunhallintaan kuuluu sekä laadunohjaus, että laadunvarmistus. Laadunohjaussa tarkoitetaan niiden teknikoiden ja toimintojen joukkoa, joita käytettävänä pyritään täyttämään laatuvaihtoimukset. Laadunvarmistuksen tavoitteena on saavuttaa sekä organisaation sisäisten että ulkopuolisten asiakkaiden ja viranomaisten luottamus tavoitteiden toteutumiseen. Tämän prosessialueen tarkoitus on määrittää ja ylläpitää laadunhallinnan toteutumista.

Kvantitatiivinen prosessinhallinta: Tarkoituksena on perustaa suorituskykytavoitteet, suorituskyvyn mittaminen ja analyysitapa ja näiden toimintojen pohjalta ohjata prosessia siten, että sen suorituskyky säilyy tai paranee.

Prosessien muutoshallinta: Tarkoituksena on jatkuvasti kehittää olemassa olevaa ohjelmistotuotantoprosessia. Prosessialueen tehtäviin kuuluvat mm. kehitystavoitteiden määrittäminen ja kehityskohteen tunnistaminen, arviointi ja toteuttaminen ohjelmistotuotantoprosessissa.

Teknologian muutoshallinta: Tarkoituksena on tunnistaa hyödyllisiä uusia teknologioita ja siirtää ne hallitusti organisaation toimintaan. Prosessialueen
tehtäviin kuuluu uusien teknologioiden tunnistaminen, valinta, evaluointi ja teknologian ajaminen organisaatioon.

- **Vikojen ehkäisy:** Tarkoituksena on estää virheitä tai vikoja syntymästä. Prosessialueeseen kuuluu aikaisempien virheiden analysointi juurisyyn tunnistamiseksi ja toimenpiteiden määrittely, joiden avulla virhe ei toistuisi enää tulevaisuudessa.

Organisaatio pystyy siis määrittelemään, miten se täyttää avainprosessialueiden vaatimukset. Kypsyystaso muuttuu ulkoisen sertifiointin kautta, missä tarkastava osapuoli arvioi tavoiteltavan kypsyystason avainprosessialueiden vaatimusten täyttymisen.

3.3.2. **Capability Maturity Model Integration**


- **Konfiguraatiohallinta:** Prosessialue kattaa kaikkien projektin tuotteiden eheyden ylläpitämisen.
- **Prosessien ja tuotteiden laadunvarmistus:** Laadunvarmistukseen liittyvä työ, jolla varmistetaan, että tuote vastaa vaatimuksia ja standardejä.
- **Mittausanalyysi**: Prosessialue kattaa tehtäviä, kuten mittareiden määrittäminen, mittausdatan säilöminen, raportointi ja analyysi. Tämä prosessialue tukee tarpeellisen tiedon keräämistä, joka auttaa päätöksenteossa.

<table>
<thead>
<tr>
<th>CMMI taso 5, optimointi</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Organisaationaalinen innovaatio ja toimeenpano (Organizational Innovation and Deployment)</td>
</tr>
<tr>
<td>• Kausaalialalyysi (Causal Analysis and Resolution)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CMMI taso 4, kvantitatiivisesti hallittu</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Organisatoristen prosessien suorituskyky (Organizational Process Performance)</td>
</tr>
<tr>
<td>• Kvantitatiivinen projektinhallinta (Quantitative Project Management)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CMMI taso 3, määritetly</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Vaatimusten kehitys (Requirements Development)</td>
</tr>
<tr>
<td>• Tekninen ratkaisu (Technical Solution)</td>
</tr>
<tr>
<td>• Tuoteintegraatio (Product Integration)</td>
</tr>
<tr>
<td>• Varmentaminen (Verification)</td>
</tr>
<tr>
<td>• Valdointi (Validation)</td>
</tr>
<tr>
<td>• Organisatorisiin prosesseihin keskittyminen (Organizational Process Focus)</td>
</tr>
<tr>
<td>• Organisatoristen prosessien määrittäminen (Organizational Process Definition)</td>
</tr>
<tr>
<td>• Organisatorinen koulutus (Organizational Training)</td>
</tr>
<tr>
<td>• Integroitu projektinhallinta (Integrated Project Management)</td>
</tr>
<tr>
<td>• Riskienhallinta (Risk Management)</td>
</tr>
<tr>
<td>• Päätöksenteon analyysi (Decision Analysis and resolution)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CMMI taso 2, hallittu</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Vaatimustenhallinta (Requirements Management)</td>
</tr>
<tr>
<td>• Projektisuunnittelu (Project Planning)</td>
</tr>
<tr>
<td>• Projektiseuranta ja hallinta (Project Monitoring and Control)</td>
</tr>
<tr>
<td>• Toimittajenhallinta (Supplier Agreement Management)</td>
</tr>
<tr>
<td>• Mittausanalyyysi (Measurement and Analysis)</td>
</tr>
<tr>
<td>• Prosessien ja tuotteiden laadunvarmistus (Process and Product Quality Assurance)</td>
</tr>
<tr>
<td>• Konfiguraationhallinta (Configuration Management)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CMMI taso 1, aloitus</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Ei prosessialueita</td>
</tr>
</tbody>
</table>

Kuva 5. CMMI-mallin kypsystasot ja tasojen avainprosessialueet [Siviy et al., 2008].
Toimittajenhallinta: Prosessialue kattaa alihankkijasuhteiden hallinnon. Tähän liittyy sopimuksien hallinta, kuten myös alihankkijoiden hankkimiseen liittyvien riskien hallinta.

Projektiseuranta ja hallinta: Prosessialue kattaa projektien etenemisen seurannan mm. aikataulun ja resurssien kannalta. Tämä alue määrittää toimintatavat projektiseurannalle.

Projektisuunnittelu: Projektisuunnittelun muodostaminen.

Vaatimustenhallinta: Vaatimustenhallintaan liittyvä työ.

Päätöksenteon analyysi: Päätöksenteon määrittely. Tähän liittyy päätöksentekijän tunnistaminen päätöksentekotilanteissa ja formaalin päätöksentekotavan muodostaminen.

Riskienhallinta: Potentialisten riskien, eli asioiden, jotka voivat estää projektia saavuttamasta tavoitteensa, tunnistaminen ja niihin varautuminen. Lisäksi tämä prosessialue sisältää riskien kehittymisen seurannan, eli muuttuvatko riskit todennäköisemmiksi projektien edessä.

Integroitut projektinhallinta: Prosessialueen tehtävänä on integroida projektin asianomaiset projektiin mukaan siten, että kaikki voivat tuoda oman osuutensa projektiin.

Organisatorinen koulutus: Prosessialue kattaa organisaation koulutussuunnittelun. Tarkoituksena on tunnistaa organisaation henkilöstön henkilökohtaiset koulutustarpeet tehtävien mukaan.


Organisatorisiin prosesseihin keskittyminen: Määrittää organisaatioon elimen, joka vastaa organisaation toimintaprosessin kehittämisestä. Tämä elin ajaa parannuksia prosessiin syvän ymmärryksen ja tiedon kautta.

Validointi: Varmistetaan, että tuote on rakennettu siten, että se täyttää alkuperäisen käyttötarpeen.

Varmentaminen: Prosessialue määrittää, miten jokaisessa tuotteen tuotantovaiheessa varmennetaan, että tuote on rakennettu oikein.


Tekninen ratkaisu: Prosessialue kattaa teknisen toteutuksen suunnittelun, eli miten ja millä teknisellä ratkaisulla asiakasvaatimukset saadaan täytettyä.

Kvantitatiivinen projekinhallinta: Prosessialue kattaa organisaation tavan edetä kohti tiedon ajamasta, missä jokainen päätös perustuu mitattuun tietoon. Prosessialue määrittelee mittausprosessit ja tavan, jolla mittausprosessi liittyy projektin ja liiketoiminnan suorituskykyyn ja laatuun.

Organisatoristen prosessien suorituskyky: Prosessialue kattaa organisaation prosessien suorituskyvyn kvantitatiivisen mittauksen. Prosessialue kattaa myös kerätyn tiedon tallentamisen, tiedon jakamisen ja prosessien mallintamisen.

Kausaalianalyysi: Prosessialue kattaa juurisyiden tunnistamisen ongelmille, kuten myös menestystekijöille. Tämä prosessialue on keskeisesti yhteydessä kvantitatiivisen tiedon käyttämiseen päätöstenkäsittelyyn.


3.3.3. SW-CMM -mallin ja CMMI-mallin ero

SEI esittelee SW-CMM -mallista CMMI-malliin päivitymisen keskeisimmät edut seuraavasti [SEI, 2004]:

- Hallinnollisten aktiviteettien sekä kehitysaktiviteettien yhdistäminen liiketoiminnallisiin tavoitteisiin.
- Parantunut näkyyvyys tuotteen elinkaareen sekä kehitysaktiviteettieihin, jotta tuote tai palvelu täyttää asiakasväestön tarpeet.
- Lisätehon hakeminen lisäprosessialueiden parhaista toimintatavoista, kuten esim. mittauksesta, riskienhallinnasta ja toimittajien hallinnoinnista.
- Vankemmatt korkeampien kypsyystasojen toimintatavat.
- Parantunut näkyyvyys organisaation toimintoihin, jotka ovat kriittisiä tarjottujen tuotteiden tai palveluiden kannalta.
- Tiukempi pariuttaminen keskeisten ISO standardien kanssa.
3.4. Henkilökohtainen ja ryhmäohjelmistoprosessi, PSP/TSP


- Jokainen kehittäjä on erilainen. Ollakseen mahdollisimman tehokas, jokaisen kehittäjän on itse suunniteltava työnsä ja suunnitelman pitää perustua henkilökohtaiseen toimintatehokkuuteen.
- Kehittäjäkseen jatkuva perusteltava tehokkuutta, kehittäjien on käytettävä hyvin määriteltyjä ja mitattuja prosesseja.
- Tuottakseen laadukkaita tuotteita, kehittäjien pitää tuntea henkilökohtaisista vastuuta laadusta.
- On halvempaa löytää ja korjata virHEET aikaisemmassa ohjelmistotuotantoprosessin vaiheessa verrattuna myöhempään.
On tehokkaampaa ehkäistä virheiden syntyä kuin etsiä ja korjata niitä.
Oikea tapa suorittaa tehtävää on nopein ja halvin tapa.


Taso 1.0 esittelee arvioinnin, siten että kehittäjän on arvioitava uusien ohjelmien koot. Arvioiden tulee perustua edellisten projektien henkilökohtaisiin arvioihin.

Kuva 7. PSP-prosessin vaiheet [Humphrey, 2000a].
Kaikkien projektien toteutuneet koot ja kestot tallennetaan. Tasoa 1.0 jatketaan 1.1-tasolla, jossa arviointeja aletaan hyödyntää tehtäväsuunnittelussa ja aikataulujen suunnittelussa.


- Ryhmän jäsenet muodostavat yhteisen tavoitteen ja määrittävät omat roolinsa.
- Ryhmää muodostaa yhteisesti hyväksytyn strategian ja määrittää yleisen prosessin, joka ohjaa heidän työtä.
- Kaikki jäsenet ottavat osaa suunnitelman tuottamiseen ja jokainen tuntee oman henkilökohtaisen roolin suunnittelussa.
- Ryhmä esittelee suunnitelman johdolle ja johto katselma ja hyväksyy, mahdollisesti neuvottellen, suunnitelman.
- Ryhmä tekee työnsä suunnitelman mukaan.
- Ryhmä kommunikoi vapaasti ja usein, josta pitäisi seurata koheesiotila ja tehokkaampi yhteistyö.
- Ryhmän ohjelmistokehittäjät tuntevat tilanteensa Ja saavat palautetta työstään.
- Ryhmää johdetaan tukemalla heidän motivaatiota.

TSP-ryhmä muodostetaan ja laukaistaan toimintaan yhteisen suunnitelman perusteella. Tähän suunnitelmaan on määritetty tehtävät ja niiden arvioidut koot ja aikataulut. TSP on iteratiivinen menetelmä, sillä yhden aloituksen sijaan, prosessi laukaistaan alusta väliajoin. Uudelleenlaukaisut ovat tarpeen, jotta projektin eri vaiheet voidaan suunnitella edellisten vaiheiden datan perusteella. Lisäksi uudelleenlaukaisun
yhteydessä ohjelmistokehittäjät päivittävät henkilökohtaiset PSP-suunnitelmat ja arviot, jotka ovat tarkkoja vain muutaman kuukauden ajan.

3.5. Vesiputousmalli

Vesiputousmalli on ollut yleisesti käytössä ohjelmistokehityksen alkuajoista asti. Ajan kuluessa asiakas- ja järjestelmävaatimukset ovat muuttuneet monimutkaisemmiksi ja kompleksisuus on lisääntynyt. Vesiputousmalli on kohdannut kritiikkiä nykyaikaisen ohjelmistokehityksen vaatimusten edessä. Pressman [2005, s. 79–80] mainitsee kolme ongelmia, jotka voi kohdata vesiputousmallin käytössä:

1. Harvat projektit voivat seurata lineaarista mallia. Vaikka malli tukee iterointia, se ei onnistu luonnollisesti.
2. Asiakkaan on usein mahdotonta lausa kaikkia vaatimuksia eksplisiitissesti projektin alussa. Vesiputousmalli vaatii tätä ja mallilla on vaikea hallita luonnollista epävarmuutta, joka on olemassa projektien alkuvaiheissa.

3.6. Scrum


kehittämisen näkökulmasta tunnistamaan mikä sprintin aikana meni hyvin ja mikä meni huonosti. Projektin etenemistä seurataan jokaisesta sprintistä kerätyin velocity-arvon perusteella, joka kertoo, kuinka monta pistettä projektiiryhmä pystyy toteuttamaan sprintissä. Tämän jälkeen seuraava sprint suunnitellaan ja uusi iteratio alkaa alusta.


3.7. Automaattinen yksikkötestaus

Yksikkötestaus on yksi testausstrategia, joka perustuu komponenttien ja moduulien testaukseen, siten että testin laajuus rajoittuu komponentin sisäisen prosessin ja tieterakenteiden testaamiseen [Pressman, 2005, s. 394–395]. Yksikkötesti ohjelmoidaan siten, että se on riippumaton kaikesta testikohdteen ulkopuolisesta toteutuksesta, sekä sen ajamista ei rajoita mikään, eli sen voi ajaa nopeasti, milloin vain. Tämä mahdollistaa yksikkötestien nopean ja riippumattoman toistettavuuden.


Testaukseen ja erityisesti yksikkötestaukseen liittyy käsite testikattavuus. Testikattavuudella tarkoitetaan prosenttiosuutta ohjelmakoodin rinnalle pääntävissä, jotka ajetaan yksikkötesteissä. Sataprosenttinen testikattavuus tarkoittaa, että jokainen ohjelmakoodin rinnalle pääntävä käydään läpi yksikkötestien yhteydessä. Testikattavuus osoittaa yleensä järjestelmän arkkitehtuuritasoa tai komponentteja, joiden testikattavuus ei ole kehittynyt yksikkötestejä kirjoitettaessa.

Varsiokin ketterissä menetelmissä käytetään ns. jatkuvan integroinnin palvelua (Continuous Integration) [Duvall et al., 2007]. Yleensä tämä palvelu liitetään versionhallintaan siten, että kun versionhallintaan siirretään uusi versio jostain rakennettavan järjestelmän osasta, jatkuvan integroinnin palvelu ajaa kaikki yksikkötestit versionhallinnan koodipohjasta vasten ja ilmoittaa, jos versionhallinnassa on rikkoinen versio rakennettavasta järjestelmästä. Tämä mahdollistaa nopean reagoinnin virheeseen ja estää virheen leviämisen. Lisäksi jatkuvan integroinnin palveluihin voi liittää esim. automaattisen testikattavuuden määrittämisen ja muita
automaattisesti laskettavia koodiin liittyviä mittareita, esim. koodirivien määrän laskemisen. Esimerkkinä jatkuvan integraation palvelusta mainittakoon Hudson [Hudson, 2010], joka on avoimeen lähdekoodiin perustuva jatkuvan integraation palvelin, johon voi liittää erilaisia mittauskomponentteja.
4. Six Sigma

Six Sigma on eri asiayhteyksissä yhdistetty mm. laadunhallintaan, projektinhallintaan ja liiketoiminnan tehostamiseen, joista yhtäkään se ei yksittäisesti edusta. Six Sigma on myös saanut tilastollisen menetelmän leiman, johtuen metodologiassa hyödynnettävistä tilastollisista työkaluista. Kyseessä on kokonaisuuus, joka tulee helposti luokittelua käytön tartoitusperään tai tavoitteiden perusteella. Six Sigmasta on olemassa lukuisia kuvauksia, kirjallisuutta, itseopiskelumateriaalia, mutta ei kattavaa empiiriseen tutkimuksen tuottamaa pohjaa. Schroeder ja muut [2008] tutkivat Six Sigman piirteitä ja ominaisuuksia muodostakseen perusmääritelmän, joka sisältäisi Six Sigman teoreettiset näkökulmat. Heidän määritelmänä suomennettuna on seuraava:

Six Sigma on järjestelmällinen rinnakkaisrakenne, jolla on tarkoitus vähentää organisatoristen prosessien variaatiota, käyttämällä kehittämiseen erikoistunutta henkilöstöä, rakenteista metodia ja suorituskykyjen mittauksesta strategisten tavoitteiden saavuttamiseen.

KYSEINEN MÄÄRITELMÄ KATTAA KESKEISET PIIRTEET METODOLOGIASTA. SIX SIGMA – METODOLOGIALLA ON OMAT ROOLINSIA JA HENKILÖSTÖHIERARKKIA, JOISTA SEURAAN RINNACKAISUUS ORGANISAATORIKANGAN KANSSA. ROOLIT VAATIVAT KOULTUSTA, JOKA ERIKOISTAA HENKILÖSTÖN METODOLOGIAN VAATIMIIN TEHTÄVIIN. SIX SIGMA ON VAIHEISTETTU, RAKENTEEN OMAAVA METODOLOGIA, JOKA HYÖDINYTTÄ MUITTA TULOKSIKSIA KEHITYKSEN VARMISTAMISESSA JA PÄÄTÖSTEN TEossa. ZU JA MUUT [2008] KÄYTÄVÄT SIX SIGMASTA SEURAAVAA MÄÄRITELMÄÄ:

Six Sigma on organisoitu, systemaattinen metodi strategiselle prosessin kehittämiselle, uusien tuotteiden, sekä palveluiden kehittämiselle, joka luottaa tilastotieteen metodeihin ja tieteelliseen metodiin tehdä dramaattisia vähennyksiä asiakkaan määrittämien kohteiden virhemääriisissä.


olemassa olevat teknikat uudessa paketissa. Mitä uutta Six Sigma toi sitten mukanaan? Schroeder ja muut [2008] esittelevät neljä keskeistä Six Sigman tuomaa lisäystä suhteessa TQM-hallintostrategiaan:


Six Sigma vaatii jäsenenyn metodin tarkkaa seuraamista, täysiaikaisenSix Sigma –rooleissa toimivien henkilöiden intensiivistä koulutusta ja tilastollisten ja muiden metodologian sukulaisen liiketoiminnan integrointia organisaation toimintaan. Erityisen kehitysprosessin käyttäminen kehittämiseen erikoistetulla henkilöstöllä vaatii organisaatioilta tukea ja sitoutumista metodologian käyttämiseen.

Six Sigma vaatii jäsenenyn metodin määritettyjä mittareita. Nämä erityiset mittarit, kuten sigma-taso ja DPMO (Defects Per Million Opportunities), konkretisoivat parannustulokset ja prosessin kunnon metodologian tuomien tulosten kautta. Mittarit korostavat prosessin parantamista ja rohkaisevat vaativiin, mutta realistisiin tavoitteisiin. Lisäksi, Six Sigman mittarit varmistavat, että asiakasvaatimukset huomioidaan, kun kehitysprosessia ruvetaan käyttämään. Six Sigman keskeiset mittarit esitellään luvussa 4.2.


Six Sigma roolirakenne
Six Sigma jäsentynyt kehitysprosessi
Six Sigma keskeinen mittaaminen

<table>
<thead>
<tr>
<th>Ylemmän johdon tuki</th>
<th>+</th>
<th>+</th>
</tr>
</thead>
<tbody>
<tr>
<td>Työvoiman johto</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Laatuinformaatio</td>
<td></td>
<td>+</td>
</tr>
<tr>
<td>Tuote- ja palvelusuunnittelu</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Prosessinhallinta</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Six Sigma roolirakenne</td>
<td>+</td>
<td></td>
</tr>
<tr>
<td>Six Sigma keskeinen mittaaminen</td>
<td>+</td>
<td></td>
</tr>
</tbody>
</table>

Taulukko 2. Six Sigma ja perinteisen laadunhallinnan käytäntöjen positiiviset suhteet [Zu et al., 2008].

4.1. Six Sigma toimintamallit


4.1.1. Define, Measure, Analyze, Improve, Control (DMAIC)


Ennen Six Sigma –projektin aloittamista, olemassa oleva prosessi tutkitaan ja sen kehitysmahdollisuudet arvioidaan. Jos arvioiden mukaiset säästöt tai muut tavoitteet

Määrittely (Define)


Käytössä oleva tuotantoprosessi kartoitetaan korkealla tasolla. Tässä vaiheessa on tärkeää tunnistaa prosesin ja sen aliprosessien rajat, syötteet, tulokset ja asianomaiset. Prosessikartoituksen tarkoituksena on auttaa projektin määrittelyä, sekä projektiryhmän muodostamista.

muuttaminen eksplisiittiseksi tiedoksi vaikuttaa positiivisesti Six Sigma -projektin onnistumiseen. Tämä on järkeenkäyppää, kun ottaa huomioon, että jokaisessa organisaatioyksikössä on oma tietosisältö.

**Mittaus (Measure)**


**Analyysi (Analyze)**

jossa kysymyksen vastaus asetetaan kyseenalaiseksi toistuvasti, kunnes pohjimmainen syy tunnistetaan.

Projektiryhmän tulee keksiä ratkaisuehdotuksia juurisyiden ratkaisemiseen. Ratkaisuehdotusten tulee poistaa tai vähentää juurisyyn aiheuttamaa hajontaa prosessissa. Kehitetyjä ratkaisuehdotuksia verrataan toisiinsa ja arvioidaan, mitkä ehdotukset kehittävät tuotantoprosessia siten, että se tuottaa asiakasvaatimusten mukaista tuotetta mahdollisimman tehokkaasti.

**Kehitys (Improve)**

Kehitysvaiheessa projektiryhmällä on analyysivaiheessa tunnistetut ratkaisuehdotukset. Ratkaisuehdotuksista on tarkoitus muodostaa konkreettisia prosessimuutoksia, jotka tähtäävät optimoituun tuotantotoimintaan.


Yksi yritysmaailmassa yleisesti tunnettu tapa parantaa omaa prosessia on verrata omaa tuotantoprosessia alan johtavien organisaatioiden tuotantoprosesseihin ja hakea sieltä kehitysajatukseja. Tällaista toimintaa kutsutaan termillä benchmarking [Stewart, 2010]. Benchmarking ei vaadi Six Sigmaa, vaan-parametrisiä analyysityökaluja, mutta Six Sigma tuo mukana työkalut, joilla voidaan mitata ja varmistaa muutos.

**Valvonta (Control)**


Valvontavaiheessa prosessin kuntoa mitataan lyhyemmillä aikavälein. Lyhyen aikavälin tulokset kertovat tuotantoprosessin muutoksesta nopeasti, jolloin sitä voidaan
vielä ohjata oikeaan suuntaan. Kun tuotantoprosessin toiminta alkaa vakiintua, voidaan mittausten aikaväliä suurentaa.

Dokumentointi kuuluu valvontavaiheeseen olellisena osana. Vaiheen aikana tulisi dokumentoida asioita, joista voidaan oppia jatkossa, sekä kehittää tulevien valvontavaiheiden toteuttamista. Valvontavaiheen dokumentointiin tulisi kuulua ainakin seuraavat asiat [Keller, 2005, s. 164]:

- Projektin perustuskirja kertoo lukijalle projektin ongelman, tavoitteet ja suunnitelman.
- Jokaisesta DMAIC-vaiheesta tulisi olla yhteneveto tavoitteista ja tuloksista.
- Liitteeksi tulisi liittää kerätty raakadata, sekä datasta tehdyt analyysit aikajärjestyksessä.
- Projektin toteuttamiseen tarvitut menoerät tulisi listata. Tähän listaukseen kuuluvat esim. väliaikaisesti menetety suorituskyky, materiaali- ja työvoimakulut.
- Projektin tuloksena syntyneet säästöt tulisi kirjata kuluneeseen päivään asti, sekä arvioida vaikutukset tuleville lähivuosille.
- Projektin nykyinen tilanne ja valvontasuunnitelma.
- Suositukset tuleville projekteille, jotka liittyvät kyseiseen prosessiin.

4.1.2. Design For Six Sigma (DFSS)


Optimointivaiheessa (Optimize) tuotantoprosessin suorituskyky optimoidaan. Optimointi suoritetaan parametrisoimalla tuotantoprocesi ja tutkimalla eri parametrien vaikutusta tuotantoprosessiin. Tätä kautta tuotantoprosessin määrittely teoreettiset, optimaaliset parametrit, joilla tuotantoprosessin pitäisi toimia tehokkaimmillaan myös käytännössä.

Tarkistusvaiheessa (Verification) kehitetty tuote ja tuotteen valmistusprosessi todennetaan valmistamalla tuotteesta prototyyppi ja pilootimalla tuotantoproessi. Tuotteen prototyyppiä käytämmällä tutkitaan, täyttääkö tuote sille asetetut asiakasvaatimukset. Tuotantoproessin pilotoimilla varmistetaan, että tuotantoproessilla voidaan luoda tuotetta asiakasvaatimusten mukaiseksi mahdollisimman tehokkaasti.
4.2. Six Sigma –metodologian keskeiset mittarit ja analyysi

4.2.1. Sigma-taso
Sigma (σ) on kreikkalainen kirjain, joka tilastotieteessä on keskihajonnan symboli. Sigma-taso kertoo, kuinka paljon mitatun kohteen tulos vaihtelee ja kuinka prosessin mitattava kohde pysyy sille määritetyllä kontrollialueella.

Keskihajonta ei itsessään esitä konkreettista näkökulmaa prosessin kunnosta. Sigma-taso on rinnastettu virheiden suhteelliseen määrään miljoonasta virheen mahdollisuudesta. Virheellä tarkoitetaan prosessin toimintaa, joka ei pysy prosessille määrittyjen rajojen sisällä. DPMO-mittari (Defects Per Million Opportunities) tuo näkökulman, josta saa keskihajontaa selkeämmän kuvan, kuinka usein prosessi tuottaa virheitä. Taulukossa 3 on kuvattu, miten Sigma-tasot vastaavat virheiden määriä miljoonassa virheen mahdollisuudessa.


![Kaava 2. Sigma-tason laskukaava.](image)
Sigma-taso lasketaan mittausulosjoukosta, joka on tässä yhteydessä tilaston populaatio. Populaatiosta lasketaan keskiarvo μ, sekä keskihajonta σ. Määrittelyrajojen, populaation keskiarvon ja keskihajonnan avulla voidaan laskea Sigma-taso z ylemmälle ja allemmalle määrittelyrajaalle kaavoilla, jotka ovat kuvattuina kaavassa 2.

Arvolla $Z_u$ voidaan tutkia, kuinka suuri osa koko populaatiosta on todennäköisesti pienempi kuin USL tai zi-arvolla suurempi kuin LSL. Aivan kuten tilastotieteessä, myös Six Sigman yhteydessä normaalijakauman todennäköisyyskaavat tutkitaan valmiiksi lasketuista taulukoista. Yleisen normaalijakauman todennäköisyystaulukosta voidaan hakea todennäköisyysarvo $z$-arvon avulla, joka on arvoltaan välillä [0,1].

Sigma-taso ei kuitenkaan tule aivan suoraan tuloksesta z. Motorolan projekteista kerättyjen tietojen perusteella metodologian kehittäjät tekivät huomion, että Sigma-taso ei pysy mitatulla tasolla pitkällä aikavälillä [Keller, 2005, s. 4]. Normaalijakauman keskitetyn +/-6σ sisään kuuluu 99,999998 % populaatiosta, eli Sigma-tasolla 6 DPMO on 0,002 eikä usein kirjallisuudessa esitetty 3,4 (esim. [Keller, 2005, s. 4] ja [Pressman, 2005, s. 761]). Motorolan havaintojen mukaan pitkällä aikavälillä prosessin suorituskyky on 1,5σ heikompi. Motorola on selittänyt siirtymää esimerkiksi materiaalivariaatioilla, valmistusmenetelmissä ja luonnollisella kulumisella. Toisin sanoen, keskitettävä tulosta siirretään 1,5σ heikompi. Motorola on selittänyt siirtymää esimerkiksi materiaalivariaatioilla, valmistusmenetelmissä ja luonnollisella kulumisella. Toisin sanoen, keskitettävä tulosta siirretään 1,5σ heikompi. Motorola on selittänyt siirtymää esimerkiksi materiaalivariaatioilla, valmistusmenetelmissä ja luonnollisella kulumisella. Toisin sanoen, keskitettävä tulosta siirretään 1,5σ heikompi. Motorola on selittänyt siirtymää esimerkiksi materiaalivariaatioilla, valmistusmenetelmissä ja luonnollisella kulumisella. Toisin sanoen, keskitettävä tulosta siirretään 1,5σ heikompi. Motorola on selittänyt siirtymää esimerkiksi materiaalivariaatioilla, valmistusmenetelmissä ja luonnollisella kulumisella. Toisin sanoen, keskitettävä tulosta siirretään 1,5σ heikompi. Motorola on selittänyt siirtymää esimerkiksi materiaalivariaatioilla, valmistusmenetelmissä ja luonnollisella kulumisella. Toisin sanoen, keskitettävä tulosta siirretään 1,5σ heikompi. Motorola on selittänyt siirtymää esimerkiksi materiaalivariaatioilla, valmistusmenetelmissä ja luonnollisella kulumisella. Toisin sanoen, keskitettävä tulosta siirretään 1,5σ heikompi. Motorola on selittänyt siirtymää esimerkiksi materiaalivariaatioilla, valmistusmenetelmissä ja luonnollisella kulumisella. Toisin sanoen, keskitettävä tulosta siirretään 1,5σ heikompi. Motorola on selittänyt siirtymää esimerkiksi materiaalivariaatioilla, valmistusmenetelmissä ja luonnollisella kulumisella. Toisin sanoen, keskitettävä tulosta siirretään 1,5σ heikompi. Motorola on selittänyt siirtymää esimerkiksi materiaalivariaatioilla, valmistusmenetelmissä ja luonnollisella kulumisella. Toisin sanoen, keskitettävä tulosta siirretään 1,5σ heikompi. Motorola on selittänyt siirtymää esimerkiksi materiaalivariaatioilla, valmistusmenetelmissä ja luonnollisella kulumisella. Toisin sanoen, keskitettävä tulosta siirretään 1,5σ heikompi. Motorola on selittänyt siirtymää esimerkiksi materiaalivariaatioilla, valmistusmenetelmissä ja luonnollisella kulumisella. Toisin sanoen, keskitettävä tulosta siirretään 1,5σ heikompi. Motorola on selittänyt siirtymää esimerkiksi materiaalivariaatioilla, valmistusmenetelmissä ja luonnollisella kulumisella. Toisin sanoen, keskitettävä tulosta siirretään 1,5σ heikompi. Motorola on selittänyt siirtymää esimerkiksi materiaalivariaatioilla, valmistusmenetelmissä ja luonnollisella kulumisella. Toisin sanoen, keskitettävä tulosta siirretään 1,5σ heikompi. Motorola on selittänyt siirtymää esimerkiksi materiaalivariaatioilla, valmistusmenetelmissä ja luonnollisella kulumisella. Toisin sanoen, keskitettävä tulosta siirretään 1,5σ heikompi. Motorola on selittänyt siirtymää esimerkiksi materiaalivariaatioilla, valmistusmenetelmissä ja luonnollisella kulumisella. Toisin sanoen, keskitettävä tulosta siirretään 1,5σ heikompi. Motorola on selittänyt siirtymää esimerkiksi materiaalivariaatioilla, valmistusmenetelmissä ja luonnollisella kulumisella. Toisin sanoen, keskitettävä tulosta siirretään 1,5σ heikompi. Motorola on selittänyt siirtymää esimerkiksi materiaalivariaatioilla, valmistusmenetelmissä ja luonnollisella kulumisella. Toisin sanoen, keskitettävä tulosta siirretään 1,5σ heikompi. Motorola on selittänyt siirtymää esimerkiksi materiaalivariaatioilla, valmistusmenetelmissä ja luonnollisella kulumisella. Toisin sanoen, keskitettävä tulosta siirretään 1,5σ heikompi. Motorola on selittänyt siirtymää esimerkiksi materiaalivariaatioilla, valmistusmenetelmissä ja luonnollisella kulumisella. Toisin sanoen, keskitettävä tulosta siirretään 1,5σ heikompi. Motorola on selittänyt siirtymää esimerkiksi materiaalivariaatioilla, valmistusmenetelmissä ja luonnollisella kulumisella. Toisin sanoen, keskitettävä tulosta siirretään 1,5σ heikompi. Motorola on selittänyt siirtymää esimerkiksi materiaalivariaatioilla, valmistusmenetelmissä ja luonnollisella kulumisella. Toisin sanoen, keskitettävä tulosta siirretään 1,5σ heikompi. Motorola on selittänyt siirtymää esimerkiksi materiaalivariaatioilla, valmistusmenetelmissä ja luonnollisella kulumisella. Toisin sanoen, keskitettävä tulosta siirretään 1,5σ heikompi. Motorola on selittänyt siirtymää esimerkiksi materiaalivariaatioilla, valmistusmenetelmissä ja luonnollisella kulumisella. Toisin sanoen, keskitettävä tulosta

<table>
<thead>
<tr>
<th>Sigma-taso</th>
<th>DPMO 1,5 siirtymän jälkeen</th>
<th>Virheettömien osuus 1,5 siirtymän jälkeen</th>
<th>DPMO ilman 1,5 siirtymää</th>
<th>Virheettömien osuus ilman 1,5 siirtymää</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>933 193</td>
<td>6,68%</td>
<td>500 000</td>
<td>50,00%</td>
</tr>
<tr>
<td>1</td>
<td>691 462</td>
<td>30,85%</td>
<td>158 655</td>
<td>84,13%</td>
</tr>
<tr>
<td>2</td>
<td>308 537</td>
<td>69,15%</td>
<td>22 750</td>
<td>97,72%</td>
</tr>
<tr>
<td>3</td>
<td>66 807</td>
<td>93,32%</td>
<td>1349</td>
<td>99,87%</td>
</tr>
<tr>
<td>4</td>
<td>6 210</td>
<td>99,38%</td>
<td>31</td>
<td>99,9997%</td>
</tr>
<tr>
<td>5</td>
<td>233</td>
<td>99,9767%</td>
<td>0,29</td>
<td>99,99997%</td>
</tr>
<tr>
<td>6</td>
<td>3</td>
<td>99,9997%</td>
<td>0,001</td>
<td>99,99999990%</td>
</tr>
</tbody>
</table>

Taulukko 3. Sigma-tasojen ja DPMO-määrien vertailu.

Puolentoista sigman siirtymä ei ole pieni asia. Sigma-tasolla 3 keskitetyn populaation DPMO on 1349 ja 1,5σ siirtymän jälkeen DPMO on 66807. Six Sigma on saanut kritiikkiä kyseisestä siirtymästä, koska se yleistetään metodologiaan Motorolan projekteissa havaituista siirtymistä. Siirtymää ei voi tilastotieteen kautta yleistää, joten siirtymää ei voi perustella yleistettäväksi tieteellisesti. Siirtymän eroa on kuvattu 1,5...
sigma-tasolla taulukossa 3. Taulukosta ilmenee, kuinka suuri merkitys siirtymällä on esim. virheettömien osuuteen.

**Esimerkki 1**

Prosessille on määritetty mittari, jonka ylempi määrittelyraja \( z_u \) on viisi ja alempi määrittelyraja \( z_l \) on yksi. Prosessin toiminnasta otetaan sadan mittauksen otos, jonka tulokset ovat kuvattuina taulukossa 4. Taulukon tuloksissa esimerkiksi arvo 0 on ilmentynyt kerran ja arvo 2 on ilmentynyt 25 kertaa.

<table>
<thead>
<tr>
<th>Arvo</th>
<th>Frekvenssi</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>25</td>
</tr>
<tr>
<td>3</td>
<td>47</td>
</tr>
<tr>
<td>4</td>
<td>16</td>
</tr>
<tr>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
</tr>
</tbody>
</table>

Taulukko 4. Esimerkin populaatio.

Kuva 10. Esimerkin 1 tulos.
Otokselle lasketaan keskiarvo $\mu$, sekä keskihajonta $\sigma$. Tässä vaiheessa tiedetään kaikki tekijät sigma-tason laskemista varten. Sigma-taso lasketaan sekä lyhyelle, sekä pitkälle aikavälille. Sigma-taso määrittelyvälille $[1,5]$ on kuvattuna kuvassa 10.

Esimerkki 2
Toisessa esimerkissä käytetään samaa populaatiota, mutta tarkoituksena on kuvata määrettä, joka ei saa negatiivisia arvoja. Ylempi määrittelyraja $z_u$ on 5, eli kaikki arvon 5 saaneet mittaustulokset ja sitä pienemmät ovat hyväksytyjä. Tämä esimerkki eroaa edellisestä siten, että pienempää määrittelyrajaa ei käytetä normaalijakauman todennäköisyyslaskentaan. Sigma-taso määrittelyvälille $[-\infty, 5]$ on kuvattu kuvassa 11.

![Kuva 11. Esimerkin 2 tulos.](image)

4.2.2. Prosessin suorituskyky
Six Sigma -organisaatiot käyttävät kahta indeksiä kuvaamaan prosessin suorituskykyä. Suorituskykyindeksi $C_p$ kuvaa prosessin kykyä tuottaa yhdenmukaisen tulosta. Mitä yhdenmukaisempaa prosessin toiminta on, sitä suurempi $C_p$ on. Suorituskykyindeksi $C_{pk}$ kuvaa kuinka hyvin prosessin tulokset täyttävät tavoitteensa. Mitä paremmin prosessin tulokset sopivat ylemmän ja aleman määrittysrajat väliin, sitä lähempänä

\[
\sigma = \sqrt{\frac{\sum_{i=1}^{n} (x_i - \mu)^2}{n-1}} \\
C_p = \frac{USL - LSL}{6\sigma} \\
C_{pk} = \min \left[ \frac{USL - \mu}{3\sigma}, \frac{\mu - LSL}{3\sigma} \right]
\]
C\textsubscript{pk} on arvoa 1,5. Prosessi on Sigma-tasoltaan n. 6 kun C\textsubscript{p} >= 2,0 ja C\textsubscript{pk} = 1,5. Suorituskyynindeksien C\textsubscript{p} ja C\textsubscript{pk} arvot on laskettu sigma-tason esimerkkeissä, kuvissa 10 ja 11. Arvojen C\textsubscript{p} ja C\textsubscript{pk}–arvojen kaavat on esiteltä kaavassa 3.

\[
    \text{USL} = \text{ylempi määrittelyraja} \\
    \text{LSL} = \text{alempi määrittelyraja} \\
    \mu = \text{keskiarvo} \\
    \sigma = \text{keskihajonta} \\
    C_p = \frac{\text{USL} - \text{LSL}}{6\sigma} \\
    C_{pk} = \min\left(\frac{\text{USL} - \mu}{3\sigma}, \frac{\mu - \text{LSL}}{3\sigma}\right)
\]

Kaava 3. Prosessin suorituskyvyn laskentakaava.

4.2.3. Six Sigma –mittareiden vaatimuksia


Ylempi johto ei kerkeä perehtymään yksittäisten mittareiden tuloksiin, vaan he tarvitsevat laajemman kokonaiskuvan prosessista. Mittareiden tulisi siis tuottaa johdolle dataa, josta selviää prosessin kokonaistehokkuus, sisältäen heikkoudet ja vahvuudet. Prosessin voidaan asettaa mittareita, jotka kertovat heikkouden tai vahvuuden omalta alueelta. Prosessista voidaan muodostaa funktion, jossa koko prosessin tehokkuus muodostuu osaprosessien tehokkuusarvoista. Six Sigma projekteissa puhutaan tällöin Isoista ja pienistä y-kirjaimista. Yleensä abstraktiotasoa nostava funktion kirjataan muodossa \( Y = \{y_1, y_2, y_3, \ldots, y_n\} \), jossa jokainen mittarin tulos vaikuttaa funktion. Tietyn tason isot \( Y : t \) voidaan kuvata uutena funktiona korkeammalle organisaatiotasolle. Tällä lailla ylemmälle organisaatiotasolle voidaan antaa helpommin tulkittavaa dataa, jonka pohjalta on helpompia tehdä päätöksiä. Prosessin valvonnan kannalta on tärkeää, että mittarin tuottama data voidaan käsitellä heti tulkittavaksi, jolloin prosessia voidaan nopeasti ohjata toivottuun suuntaan.
4.3. Six Sigma -projektin roolit


![Kuva 12. Six Sigma roolihierarkia.](image)

4.3.1. Champion


4.3.2. Black Belt


4.3.3. Green belt


4.3.4. Yellow belt
5. **Six Sigma ja ohjelmistotuotanto**

Six Sigma hyödynnetään laajasti perinteisessä tuotannossa [SixSigmaCompanies, 2010], missä samaa tuotetta tuotetaan massoittain vakaan tuotantoprosessin kautta. Tuotantoprosessin kautta syntyvien tuotteiden tulee täyttää tuotteelle asetetut vaatimukset ja laatumääritykset, sekä tuotantokulujen tulisi pysyä samana tuotekappaleiden välillä. Tässä luvussa tutkitaan, miten Six Sigma soveltuu ohjelmistotuotantoon.

**5.1. Kriitikkiä Six Sigman soveltuvuudesta ohjelmistotuotantoon**


Motorolan havaintojen perusteella materiaalivariatioriott, valmistusmenetelmät ja luonnollinen kuluminen aiheuttavat prosessin suorituskyvyn heikkenemisen pitkällä


Nicoletten [2005] mielestä Six Sigma soveltuu vain sellaisille aloille, joissa toiminta muistuttaa luonteeltaan teollista tuotantoa. Nicolette perustelue rii tavoin, että ohjelmistokehitys muistuttaa luonteeltaan rakennusalaan. Hyvänä esimerkkinä ominaisuuksien samanlaisuudesta ovat suunnittelumallit (design patterns) [Gamma et al., 1994], jotka on huomattu hyvikiä työkaluiksi ratkaisemaan ohjelmistokehityksen ongelmia. Suunnittelumalleja voidaan käyttää äärettömän monta kertaa eri ongelmissa,
joihin ne sopivat, mutta aina erityisellä tavalla. Samalla lailla vuosituhansien saatossa rakennusalalla on opittu parhaita toimintatapoja, joilla voidaan luoda toimivia ratkaisuja rakentamisen ongelmien.


Ohjelmistotuotanto vaatii ohjelmistokehittäjiltä logisti päättelykykyä ja ongelmanratkaisukykyä. Ohjelmistotuotantoprosessi koostuu pitkälti inhimillisestä luovasta toiminnasta, joka on yksi kehittäen määrittävää. Luovan toiminnan prosessia on vaikea kartoittaa tai mitata. Siksi, jo pelkästään ihmisen kognitiivisessa prosessissa tapahtuvaa variaatio hankaloittaa Six Sigman soveltamista ohjelmistotuotantoprosessiin.

5.2. Six Sigmaan soveltaminen ohjelmistotuotantoon

Six Sigmaan sovellustapoja on monia, mutta niitä yhdistää ohjelmistotuotantoprosessin mittaaminen, mittaustulosten analysointi ja
ohjelmistotuotantoprosessin kehittäminen. Tässä kappaleessa kuvataan raportoituja Six Sigman sovellustapoja ohjelmistokehityksessä.

5.2.1. Six Sigman soveltaminen raportoitujen virheiden perusteella


järjestelmä toimii virheettömästi. Tyydyttävät tekijät ovat ominaisuuksia, joita asiakas odottaa ja joiden toteuttaminen kasvattaa asiakastyytyväisyyttä. Ilahduttavat tekijät ovat ominaisuuksia, joita asiakas ei erikseen osannut vaatia, eikä siten myös maksa, mutta joiden toteutus täyttäisi jälkikäteen tulevia tarpeita. Liiketoiminnan ääni määrittä kiintiöiden virheiden lukumäärän, jota kehityksen onnistuessa yksikään osasto ei yllättäisi.

Nepesiin voitohiin eroikoinnut ryhmä tutki ohjelmistotuotantoprosessiin kuuluvien osastojen prosesseja. Tarkoituksena oli löytää ns. matalalla roikkuvaa hedelmää, eli prosesseja, joita olisi helppo ja nopea kehittää tuottamaan korkean arvon suhteessa sijoitukseen.


katselmoinneilla, joissa tarkistettiin mittareiden tulokset, mahdollisten ongelmien tunnistaminen, sekä pakollisten muutosten priorisointi. Tuotantoprosessin laatu katselmoitiin myös kuukausittaisissa laatukatselmoinneissa.


5.2.2. Laadunkehitys henkilökohtaisen ja ryhmäohjelmistotransessin perusteella

CMM/CMMI-kypsystasomallin mukaan prosessin mittaaminen ja valvonnanalainen hallinta ovat perustat jotkuvan prosessikehitykselle. Ongelmana on, että kypsystsomalli ei kerro, miten mallia sovelletaan käytännössä. Eräs tapa on soveltaa PSP/TSP–menetelmää, jonka käyttö on kuvattu ja se on yhteensopiva CMM:n tavoitteiden kanssa.


1. Ohjelmistokehittäjien tulee arvioida ohjelmakoot tehtävistä, sekä tallentaa arviointi ja toteuma.
2. Ohjelmistokehittäjien tulee tallentaa käytetty aika kaikista tehtävistä.
3. Ohjelmistokehittäjien tulee tallentaa kaikki viat, sekä seurata niiden elämänkaarta.

Kuva 13. Six Sigma kehyksen arkkitehtuuri [Pan et al., 2007].


5.2.3. Asiakkaan prosessin kehittäminen

soveltuu IT-osaston toimintaan paremmin kuin ulkoisen ohjelmistoyrityksen toimintaan, koska ulkoinen ohjelmistoyritys ei todennäköisesti pystyisi seuraamaan ja tutkimaan asiakkaan prosesseja jatkuvasti pitkällä aikavälillä. Tayntorin kuvaus on pitkä, joten koko mallin kuvaamisen sijaan tässä kohdassa on tarkoituksena kertoa piirteistä, jotka Six Sigma tuo mukanaan perinteiseen vesiputousmalliin.


<table>
<thead>
<tr>
<th>Vesiputousmallin vaihe</th>
<th>Six Sigman DMAIC-toimintamalli</th>
</tr>
</thead>
<tbody>
<tr>
<td>Projektin aloitus</td>
<td>Määrittely, mittaus, analyysi</td>
</tr>
<tr>
<td>Järjestelmäanalyysi</td>
<td>Määrittely, mittaus, analyysi</td>
</tr>
<tr>
<td>Järjestelmäsuunnittelu</td>
<td>Analyysi</td>
</tr>
<tr>
<td>Ohjelmointi</td>
<td>Kehitys</td>
</tr>
<tr>
<td>Testaus</td>
<td>Kehitys</td>
</tr>
<tr>
<td>Käyttöönotto</td>
<td>Kehitys ja hallinta</td>
</tr>
</tbody>
</table>

Taulukko 5. DMAIC-toimintamallin vaiheistus vesiputousmallissa [Tayntor, 2007].


ohjelmistokehitykseen. Six Sigma ei myöskään vaikuta ohjelmistoarkkitehtuurin tai komponenttisuunnittelun.


Testausvaihe on aina ollut laadunhallinnan kannalta oleellinen piste todentaa, että ohjelmisto täyttää toiminnalliset vaatimukset sekä laatuvaatimukset. Tayntorin mallissa ei kuitenkaan esitellä Six Sigmaan liittyviä toimenpiteitä, jotka muuttaisivat yleisiä vesiputousmallin testaustapoja.


5.3. Six Sigma ja CMMI

CMMI:n kolme ensimmäistä kypsyystasoa sisältävät lähes kaikki prosessialueet projektinhallinnasta, teknisestä osaamisesta ja tukitoimista. Toisin sanoen, CMMI kypsyystason 3 saavuttaminen edellyttää omaa huolellisesti suunnitellun laatu- ja kehitystulemusten suorittamista ja vakaata ohjelmistoprosessin kehittämiseen. CMMI:n kolmannen kypsyystason saavuttaminen ei ole itsestäänselvyyssä, eivätkä kaikki kypsyystasoa tavoittelevat organisaatiot sitä saavuta.


Organisaatiot, jotka huomaavat hyötyvän Six Sigman käyttöönotosta, hyödyntävät kaikkia Six Sigman työkaluja, menetelmiä ja filosofiaa.

Hyvä päätöksenteko korreloi Six Sigman käytön kanssa organisaatioissa.

Six Sigma tarjoaa yhdistävän kielen, jonka kautta ylempi johto, jotka eivät tunne ohjelmistokehitystä ja ohjelmistokehittäjät ymmärtävät tapoja päästä todellisiin liiketoimintatavoitteisiin.

Six Sigma mahdollistaa CMMI:n käyttöönoton sekä siirtymisen CMM- mallista CMMI-malliin.

Six Sigma nopeuttaa CMMI:n käyttöönottoa.

Six Sigma mahdollistaa CMMI:n käytön tavalla, jolla sitä on tarkoitettu käytettävän, sillä Six Sigma vaatii organisaatiota yhdistämään kaiken toiminnan liiketoimintaan ja asiakkaihin.

Six Sigma työkalujen ja menetelmien kautta saadut tulokset antavat kuvan siitä, mitä tarvitaan CMMI:n korkeaa kypsyyteen.

CMMI:hin pohjautuvat organisaatioresurssit mahdollistavat Six Sigma – projekteihin pohjautuvan opitun tiedon jakamisen ohjelmistokeskeisissä organisaatioiden kesken ja tätä kautta mahdollistaa Six Sigma institusionalisoinnin.


5.4. DFSS-toimintamallin soveltaminen ohjelmistoprojektiin

Ohjelmistokehityksessä on yleisesti tiedossa, että mitä aikaisemmassa vaiheessa ohjelmistotuotantoprosessia virheet tunnistetaan, sitä vähemmän niiden korjaus tulee maksamaan. Varsinkin vaatimustenhallinnalla on suuri vaikutus alkuvaiheessa tehtävien virheiden tunnistamiseen. DFSS-toimintamallin soveltaminen suoraan ohjelmistokehitykseen voi olla hankalaa, sillä osa asiakasvaatimuksista voi muuttua ohjelmistotuotantoprosessin aikana, tai osa vaatimuksista tulee esille myöhemmissä vaiheissa. Ketteriä menetelmiä käytettäessä asiakasvaatimukset vakiintuvat kehitysiteraatioiden mukaan. Voidaan siis olettaa, että kaikkia asiakasvaatimuksia ei todennäköisesti voida ottaa huomioon DFSS-toimintamallia käytettäessä
ohjelmistokehitykseen. Jos kuitenkin osa määrittelyyn esille tulleista asiakasvaatimuksista on erityisen keskeisiä tuotantoprosessille, niin DFSS-analyysin voisi mahdollisesti tehdä keskeisimmillä asiakasvaatimuksilla. Analyysi voisi esimerkiksi vaikuttaa kehitysmallin, kuten vesiputousmallin tai sopivan ketterän mallin valintaan.

5.5. Six Sigma ja ketterät menetelmät


Ohjelmistoprototyypit ovat luonteen ongelmia, jonka takia myös eri projektiin voi olla erilaiset ongelmia. Lisäksi projektiin vaihtuessa projektiryhmät muuttuvat. Tässä kappaleessa kuvatun soveltamisen edunsa voidaan laskea, että projektitehtä vohaitut kehitystoimenpiteet siirtyvät kehittäjien mukana seuraaviin projekteihin. Lisäksi, projekti ei ole riippuvainen edeltävästä projekkeesta, sillä se voi sopeutua omaan yksilölliseen projektiin iteratioiden aikana koettujen havaintojen kautta, jolloin ketteryys säilyy.

Tässä kappaleessa esitellyssä sovellustavassa tosin sivutetaan Six Sigman keskeiset mittari ja toimintamallit. Voidaan myös kyseenalaistaa, että onko toimintatavassa sovellettu Six Sigmaa laiskaan. Toisaalta, yhtäläilailla voidaan myös kyseenalaistaa DPMO:n, sigma-tason ja prosessisykyindeksien arvokkuus ketterässä ohjelmistokehityksessä.
5.6. Six Sigma ja automatisoitu yksikkötestaus

Perinteinen testaus on hankala kohde Six Sigmalle, koska olemassa olevien virheiden kokonaismäärää ei voida tietää ja täten testauksen todellinen tehokkuus ja onnistuminen jäävät arvioiksi. Yksikkötestaus sen sijaan on sopiva mittauskohde Six Sigmalle, sillä testi joko läpäistään hyväksytysti, tai ei. Lisäksi testiajon voi toistaa automaattisesti ja nopeasti, mikä mahdollistaa päivittäiset ajot. Päivittäiset ajot taas luovat dataa ohjelmiston eheydestä ja itse testauksesta koko kehitysvaiheen ajan. Tämä tarjoaa dataa erilaisille Six Sigma –mittareille, jotka kertovat ohjelmistotuotantoprosessin ja tuotettavan ohjelman laatusta.

Selkein mittari yksikkötestaukselle on testien läpäisy, esimerkiksi siten, että jokaisen yksittäisen testin ajo on tuotantoyksikkö ja testinläpäisy määrittelee virheellisyden. Tällainen mittari kertoo siitä, kuinka usein ohjelmistoon ohjelmoitu logiikka täyttää testille määritetyt ehdot. Yleensä virhe yksikkötestauksessa osoittaa sen, että jossain komponenttissa tapahtuneen koodimuutoksen vaikutuksia muihin komponentteihin ei ole huomioitu, tai sitten logiikka on muuttunut, eikä testiä ole muistettu päivittää uuden toteutuksen mukaiseksi.


menettämistä. Tällä tavoin korjautoimenpiteitä voidaan suunnitella ja toteuttaa ennen kuin tilanne on huomioitavissa käyttäjäkokemuksissa tai järjestelmän vakaudessa.

5.7. **Johtopäätöksiä Six Sigman soveltamisesta ohjelmistokehitykseen**


Toki Six Sigmmaa voi hyödyntää muihinkin mittauskohteisiin sekä perinteisiin ohjelmistotuotannon mittareihin, mutta tällöin kannattaa huomioida, että sigma-taso kertoo hajonnasta ja että mittauksia tulisi tehdä pitkällä aikavälillä. Sigma-tason tulisi olla arvokas tieto projektipäällikölle ja johdollle verrattuna esimerkiksi prosenttiosuuksien arvoon.

Six Sigman käyttöä ei ole määritelty tiukasti, minkä takia organisaatiot voivat soveltaa sitä omien toimintatapojen ja tarpeiden mukaan. Organisaatioiden toimintatavat eivät ole julki tietoa, sillä ne saattavat liittyä kilpailukykyn. Todennäköisesti näiden kahden syyn takia Six Sigmasta ei ole dokumentoitua de facto.


Six Sigma vaikuttaa valmiilta paketilta, joka oikeissa käsissä kehittää prosesseja kohti täydellisyyttä. Six Sigma ei kuitenkaan ole hopealuoti parempaan ohjelmistolaatuun. Six Sigma ei ratkaise syitä, joiden takia ohjelmistoviat syntyvät tai ohjelmistoprojektit epäonnistuvat. Six Sigma voi kuitenkin ajaa organisaatiota tilaan, jossa se keksii ratkaisuja organisaatiokohtaisiin ongelmiin, jotka ovat juurisyitä huonoon ohjelmistolaatuun.
6. Yhteenveto

Six Sigma –metodologia on ollut perinteisessä teollisuudessa 80-luvulta asti osana organisaatioiden laadun- ja prosessinhallintaa. Myös osa ohjelmistoyrityksistä on ottanut Six Sigmaa osaksi organisaationa toimintaa ja samalla osaksi ohjelmistotuotantoprosessia.

Vaikka Six Sigma on kirjoitettu paljon oppaita ja dokumentoitu kokemuksia, siitä ei ole tehty kattavaa empirististä tutkimusta. Tämän tutkielman tehtävänä oli selvittää, miten Six Sigmaa käytetään ja miten se soveltuu ohjelmistokehitykseen.

Six Sigma keskeisin piirre on tarjota toimintatapa, jonka avulla prosessia voidaan mitata syntyneiden virheiden määrää sekä selvittää virheiden juurisyyt. Näiden tietojen avulla prosessia voi muokata kohti tilaa, jolloin prosessin hajonta pienenee ja prosessi tuottaa vähemmän virheitä.

Perinteisessä teollisuudessa virheitä mitataan saman tuotantolinjan tuotantoyksiköistä, joiden hyväksymiskriteerit ovat samat ja ne yleensä liittyvät tuotantoyksikon fyysisiin piirteisiin. Ohjelmistokehityksessä vastaavan mittauskohteen määrittäminen on hankalaa, sillä ohjelmakoodilla ei ole fyysisiä ominaisuuksia ja koodiriveihin liittyvät eri vaatimukset, jolloin niillä ei ole yhteistä hyväksymiskriteeriiä. Six Sigma ei siis sovellu ohjelmistokehitykseen aivan ongelmittaa. Six Sigmaa pitää soveltaa ohjelmistokehitykseen.


CMMI-kehys sisältää osa-alueita, jotka vaativat tilastollista analysiä ja prosessien toimintakyvyn mittaamista. Tilastollisen analyysin ja mittaamisen osalta Six Sigma ja CMMI toimivat synergiaassa. Sekä CMMI että Six Sigma pyrkivät stabiloihman tuotantoprosessin ja parantamaan sen tehokkuutta sekä tuotettua laatua. CMMI ja Six Sigma kummatkin vaativat organisaatiolta lisäresursseja kehitystoiminnan kunnolliseen ylläpitämiseen. Tämän takia synergia on todennäköisempää tavoittaa suurissa ohjelmistokehitysorganisaatioissa.

Jotta Six Sigma yleistyisi ohjelmistokehityksessä, sitä tulisi pystyä käyttämään ketterien menetelmien yhteydessä. Six Sigma soveltamisesta ketteriin menetelmiiin ei ollut tutkimuksen tekovaiheessa tarpeeksi tutkimustietoa, mutta tässä tutkielmassa
tunnistettiin tapoja hyödyntää Six Sigman työkaluja Scrum-projekteissa. Scrum-projektista löytyi mittareita, joiden seuranta onnistuu Six Sigman kannalta ja ne kertovat prosessin sekä tuotettavan tuotteen laadusta, joten tiedon perusteella laadunkehityksen tulisi olla mahdollista.

Six Sigma ei sovellu suoraan perinteiseen testaukseen, koska Six Sigman pitäisi tietää, kuinka monta virhettä järjestelmässä on. Tässä tutkielmassa esiteltiin Six Sigma–mittareita yksikkötestaukseen, joka on myös yleinen testausstrategia ketterien menetelmien yhteydessä. Six Sigmalta voisi tukea koko kehitysvaiheen kestävää testikattavuuden mitattavista tai yleistä testien läpäisyastetta.

Tässä tutkimuksessa esiteltiin soveltamistapoja, joiden todellinen vaikutus laadun kehittämiseen vaatii lisätutkimusta. Jotta Six Sigma vaikutus ohjelmistoprojektin laadun kehittämisessä esitetyjen toimintatapojen osalta voitaisiin todentaa, soveltamistapojen vaikutus tulisi todentaa ja dokumentoida kattavassa otoksessa, joka koostuisi eri organisaatioiden ohjelmistoprojekteista.
Viiteluettelo


