LAUREA
AMMATTIKORKEAKOULU
Uuden edellä

Testauksen laadun varmistaminen

- Case testauksen mittarit

Pehtonen, Tarja

2013 Leppävaara

Laurea-ammattikorkeakoulu
Leppävaara

Testauksen laadun varmistaminen
- Case Testauksen mittarit

Tarja Pehtonen
Tietojenkäsittelyn koulutusohjelma
Opinnäytetyö
Marraskuu, 2013

Sisällys

Laurea-ammattikorkeakoulu **Tiivistelmä**
Laurea Leppävaara
Tietojenkäsittelyn koulutusohjelma

Tarja Pehtonen

Testauksen laadun varmistaminen: case Testauksen mittarit

Vuosi 2013 Sivumäärä 42

Tämä opinnäytetyö tehtiin suomalaisessa pankki- ja rahoitusalan yrityksen ICT-palveluyksikössä, joka kehittää itse tai jolle ulkopuolinen toimittaja tuottaa muutoksia ja uusia ominaisuuksia käytössä olevia järjestelmiä. Tapaustutkimuksessa tutkittiin erisuuruisten kehitysprojektien testauksen seurannalle määriteltyjä mittaristoja ja niiden antamien tulosten merkitystä testattavan järjestelmän laadulle. Ennen tutkimuksen alkua määriteltiin käytettävät mittarit ja testauksen läpiviennin aikana analysoitiin mittareiden tuloksia. Tulosten perusteella pyrittiin määrittelemään käytettävien mittareiden luotettavuutta määriteltäessä testauksen laatua ja testauksen aikaisen prosessin tehokkuutta.

Testausprosesseina tutkittiin sekä perinteisellä vesiputousmallilla, että ketterällä mallilla tehtävää tuotekehitystä. Kummallakin mallilla toteutettujen testattavien projektien testauksen seuranta tehtiin samalla työvälineellä Quality Centerillä.

Tutkimuksen tavoitteena oli selvittää, mitkä määritellyistä mittareista antavat luotettavinta tietoa testauksen aikana, jotta testattavan järjestelmän tuotantoon viemisen kypsyys saadaan mahdollisimman luotettavasti todennettua. Analyysi perustuu tutkimuksen tekijän omakohtaisiin analyyseihin saaduista tuloksista.

Parhaiksi mittareiksi organisaation testausyksikön laadun varmistukseen valittiin kolme mittaria, jotka todentavat järjestelmätestauksen kattavuuden, seuraavat testauksen etenemistä ja keräävät tietoa testauksen suorittamisen aikana havaituista virhekirjauksista. Valitut mittarit on tarkoitus edelleen esittää testausyksikön yhteisesti noudatettaviksi mittareiksi. Niiden käyttöönotosta yksikön mittaristoissa päätetään erikseen riippumatta tämän tutkimuksen antamista analyyseistä yksikön johdon toimesta.

Asiasanat: testaus, testausprosessi, mittarit, analysointi, tietojärjestelmä

Laurea University of Applied Sciences **Abstract**
Laurea Leppävaara
Bachelor's Degree Programme in Business Information Technology

Tarja Pehtonen

Quality assurance in testing: a case study of metrics in testing

Year 2013 Pages 42

This research was carried out in a Finnish banking and financial company which maintains and develops new features to systems itself or by using an outside contractor. This case study describes metrics defined for monitoring of testing and the implications metrics can give to the quality of testing. Prior to starting the study, the metrics used were defined and during the testing process the results given by the metrics were analysed. Based on the measurement results the purpose was to evaluate the reliability of the metrics used in defining the quality and effectiveness of the testing process.

Both agility and traditional waterfall models of testing were studied. The testing process carried out with either of the models was executed using the same tool: Quality Center.

The main objective of the study was to figure out, which of the defined metric s will give the most reliable information during the testing process in order to verify the maturity of the system tested. The analysis is based on the writer's own results.

As the best metrics for quality in testing process were chosen metrics measuring requirements coverage, progression of test execution and the number of defects detected. The three best metrics are supposed to be recommended for common use in the testing organization. Introduction of the presented metrics will be determined separately despite of this research by the management of the testing organization, regardless of this research.

Keywords: metrics, testing, quality, measuring, analysing, systems

1 Johdanto

Tässä opinnäytetyössä käsitellään testauksen laatua ja menetelmiä sen mittaamiseksi tutkimalla laadun varmistamisen ja raportoinnin mittareita. ITIL-prosessin mukaisesti laadun varmistuksen määritelmä on "Prosessi, joka vastaa siitä, että laadun, palvelun, prosessin tai muun palveluominaisuuden laatu tuottaa aiotun hyödyn" (ITIL 2011 Finnish Glossary. 2011). ITIL eli Information Technology Infrastructure Library on koonti ICT-palveluhallinnan käytettävistä ja hyväksi havaituista käytännöistä ja se edustaa kokemuksia maailman johtavilta palvelutuottajilta. ITILin mukaisia prosesseja käytetään yleisesti palvelukehittämisessä läpi koko organisaation.

Yleisen käsityksen mukaan testauksen perustehtävänä on varmistaa kehitettävän tuotteen laatu ennen tuotantokäyttöä. Laadunvarmistusta ei kuitenkaan tehdä ainoastaan testausvaiheessa, vaan koko tuotteen kehitysprosessin ajan: ideointi, määrittely, suunnittelu, toteutus ja käyttöönottovaiheissa. (Kaner. 2002, 9)

Laadukkaaseen toimintaan vaikuttaa ennen kaikkea organisaation johtamiskulttuuri ja kyky kehittää laatuprosessia systemaattisesti. Tärkeätä on myös viestittää laatutavoitteet koko organisaatiolle riittävän sekä varmistaa riittävä osaaminen peilaten laatuvaatimuksiin ja –odotuksiin. (Vuori, M. 2001)

Testauksella on mahdollista todentaa kehitettävän tuotteen laatu yksiselitteisesti ja luotettavasti. Yleisin tapa arvioida testauksen aikana tuotteen laatua on peilata sitä havaittuihin virheisiin. Virheet paljastavat puutteet kehitettävän järjestelmän toiminnallisuudessa, mutta voivat olla myös merkki siitä, että toteutus on tehty vastoin haluttuja tuotteen ominaisuuksia. Todellisten virheiden löytyminen ja mahdollisimman luotettavan lopputuotteen määritteleminen on yhtä tärkeätä kuin asiakkaan haluaman tuotteen käyttöönotto.

Jotta parhaimpaan mahdolliseen laatuun päästään toteutettavien palveluiden ja tuotteiden osalta, on toteutukselle sen eri vaiheissa pystyttävä asettamaan luotettavasti mitattavat kriteerit. Kriteeristön on muodostuttava todelliseen tietoon perustuviin mitattaviin faktoihin, joiden tulkinnassa ei saa syntyä erimielisyyksiä. Palvelun tai tuotteen lopullisen laadun määrittelee viimeistään sen loppukäyttäjä. (Haikala. 1998, 194–195)

2 Tutkimuksen tausta

Opinnäytetyön taustana on mielenkiinto testauksen laadulliseen mittaukseen. Testausprosessin aikana testauspäällikölle useimmiten esitetyt kysymykset koskevat testaukseen käytettävän ajan käyttöä, testauksen kypsyysastetta, testatun järjestelmän laatua, testauksen tulosta jne. Voidakseen vastata luotettavasti ja analyyttisesti testauspäällikkö tarvitsee välineekseen mitattavaa tietoa ja siihen suhteutettuja selkeitä mittareita. Hyvät mittarit ovat ymmärrettäviä ja ne on helppo perustella. Ymmärrettävyydellä tarkoitetaan sitä, että mittareiden antamat tulokset tulkitaan lukijasta riippumatta samalla tavalla ja eri aikaan tuotettujen mittaristojen arvot ovat keskenään vertailtavissa.

Mittariston tavoitteena on olla riittävän yksinkertainen jotta se olisi helposti hallittava. Mittareiden antaman tiedon perusteella on helppo ennustaa tulevaa kehitystä ja tehdä johtopäätöksiä tulevan tekemisen suunnalle ja tavoitteille (Kooman, T., van der Aaist, L., Broekman, B. & Vroon, M. 2006, 569 – 577).

Analysoitavat mittarit valittiin tutkimuksen kohteena olevan yrityksen testausstrategian mukaisista mittareista, joiden tavoitteena on mitata testauksen laatua, testauksen aikana havaittavien virheiden määrää ja ominaisuuksia, testausprosessin laatua sekä testauksen kohteena olevan projektin laatua (Yritys X:n testausstrategia. 2011). Mittareista valittiin tutkittavaksi kahdeksan mittaria, joista valittiin lopullisesti kolme ehdotettaviksi mittareiksi, jotka parhaiten tutkimuksessa tehdyn analyysin perusteella mittaavat testauksen laatua yksiselitteisesti.

Työn tuloksena saadaan analysoituja ehdotuksia yrityksen testausyksikön mittareiksi, joilla raportoidaan työn tilaajille ja testausprosessiin osallistuville testauksen etenemisestä suunnitellussa aikataulussa ja testattavan kehitystyön laadun tasosta prosessin aikana syntyneen datan perusteella. Käytettäessä yhteistä mittaristoa, joka perustuu testauksen aikana kerääntyvään tietoon, saadaan eri projektien ja kehitystöiden laadun arvioinnista yhteismitallisempaa. Lisäksi yhteiseksi määritellyt standardoidut mittarit helpottavat raportoijan työtä, kun mittareiden tiedot ovat ajettavissa valmiiden mallipohjien avulla milloin tahansa ja raporttien ja mittareiden tietosisältöä ei tarvitse aina projekti- tai työkohtaisesti määritellä uudelleen alusta.

Wikipedian mukaan metriikalla tarkoitetaan yleisesti mittajärjestelmää (fi.wikipedia.org/wiki/Metriikka). Metriikkojen tavoitteena on lisätä tuotekehityksen aikaista näkyvyyttä sovelluksen ja järjestelmän laatuun sekä läpinäkyvyyttä toimintatapoihin. Metriikat antavat faktoihin perustuvaa tietoa päätöksenteon tueksi ja antavat mahdollisuuden tuot-

teen laadun ja toimintatapojen parantamiseen. Metriikkojen systemaattinen käyttö on keskeisessä asemassa testauksen ja koko ohjelmistokehityksen kyvykkyyden parantamisessa.

Testauksen mittaristoilla voidaan mitata paitsi kehitettävän järjestelmän laatua myös testausprosessin laatua, esim. testauksen läpimenoaikaa tai virhekorjausprosessin tehokkuutta. Tuotteen laadun metriikoilla voidaan arvioida projektissa tai hankkeessa tuotetun palvelun, sovelluksen tai osajärjestelmän laatutasoa, kuten virheiden määrää. Testausprosessin laadun mittaamisessa keskitytään arvioimaan prosessien tehokkuutta ja organisaation kyvykkyyttä.

2.1 Tutkimuksen tavoite ja tutkimuskysymys

Opinnäytetyön tarkoituksena oli selvittää, miten parhaiten ja luotettavimmin voidaan mitata kehitettävän järjestelmän testauksen laatua ja valita luotettavimmat mittarit varmistamaan järjestelmätestauksen tuloksen laatua.

Tarkoituksenani on selvittää testausprojektien testauksen läpiviennin kautta mahdollisimman luotettavat mittarit kuvaamaan testauksen lopputuloksen laatua. Valittavien mittareiden tulee mahdollisimman pitkälle perustua testauksen suorittamisen aikana syntyneeseen tietoon ja mittareiden antaman tuloksen tulee olla mahdollisimman yksiselitteisesti analysoitavissa.

2.2 Tutkimuksen kohde ja rajaus

Tutkimus tehtiin Suomessa pankki- ja rahoitusalalla toimivassa yrityksessä (Yritys X), joka kehittää ja ylläpitää alalla käytettäviä tietojärjestelmiä. Yrityksen tietojärjestelmäkehitys toimii pääkaupunkiseudulla ja Oulussa ja se käyttää järjestelmäkehityksessään oman palveluyksikön kehittäjien, suunnittelijoiden ja testausasiantuntijoiden työpanosta. Osa järjestelmäkehityksestä tehdään myös palveluostoina ulkopuolisilta toimittajilta sekä käyttäen offshore palveluita. Järjestelmäkehityksessä käytetään sekä vesiputousmenetelmää että ketterää menetelmää.

Tutkimuksen kohteena on testausvaiheista pääasiassa järjestelmätestaus, jonka laajuuteen kuuluu toiminnallinen testaus, jota suoritetaan testausautomaatioon tarkoitetuilla työvälineillä tai manuaalisesti testaten. Järjestelmätestauksen kohteena on kehitettävä järjestelmä ja sen toiminnallisuus määrittelyjä ja kirjattuja vaatimuksia vastaan. Toiminnallisuuden varmistamisen lisäksi testausvaihe varmistaa testattavan järjestelmän suorituskyvyn ja tietoturvan tason (Haikala. 1998, 290).

Tutkimuksen tavoitteena on löytää mittarit mahdollisimman hyvän laadullisen tason mittaamiseksi. Analyysin perustana on kevään 2013 aikana valitut kehitysprosessit ja niiden testauk-

sen aikana tallennettu data. Analysoinnin kohteena ei ole kaikki organisaatiossa meneillään olevat kehitysprojektit, vaan kohteeksi on otettu kevään 2013 aikana meneillään olevat viisi projektia, joissa käyttöönotto tapahtuu syyskuun 2013 aikana, jotta tutkimusalue on saatu rajattua.

2.3 Tapaustutkimus

Tapaustutkimuksen avulla voidaan saada tietoa tutkimalla haluttua tapahtumaa tai ilmiötä empiirisesti. Tutkittava voi olla määritelty yksittäinen ilmiö tai koonti useamman tapahtuman muodostavasta joukosta. (Yin. 2009, 18). Kun tapaustutkimus on arvioitu parhaaksi lähestymistavaksi tutkimukselle päästään suunnitteluvaiheeseen, jossa päätetään miten tutkimuskysymyksen tavoitteisiin päästään. Suunnitteluvaihetta seuraa aineiston kerääminen mm. aikaisemmista tutkimuksista, tutkimusaiheesta julkaisusta kirjallisuudesta, tilastoista. Jos tarvittavaa aineistoa ei löydy valmiina, on se kerättävä ja muokattava itse (Yin. 2009, 27 – 30).

Tapaustutkimus tulkitsee kerätyn tutkimusaineiston perusteella tutkittavaa ilmiötä yleistämättä tehtyjä havaintoja ja tutkijan itse osallistumatta tutkittavaan tapahtumaan. Suuri osa työstä muodostui tiedon keräämisestä lopullista analyysia varten, sekä analyysien tekemisestä aineiston perusteella. Teorian ja käytännön tiedon keruun jälkeen työn viimeisteli johtopäätösten ja oman oppimisen arviointi tehdystä prosessista sekä mahdollisten jatkokehitysehdotusten kuvaaminen (Likitalo. 1998, 35).

3 Tutkimusmenetelmän valinta ja tutkimuksessa käytettävät välineet

Tutkimusmenetelmäksi valittiin tapaustutkimus, koska analysoitavaksi määriteltyjen mittareiden pohjatieto oli mahdollista hankkia todellisten, käynnissä olevien sekä ketterällä menetelmällä että perinteisesti vesiputousmallilla toteutettujen projektien testausten kautta. Näin tutkimuksen aineistosta saatiin mahdollisimman monipuolista, luotettavaa, kattavaa, tuoretta ja tarkoituksenmukaista (Likitalo. 1998, 32). Menetelmän valinnan varmistuttua seuraava vaihe tutkimuksessa oli aikataulullinen suunnittelu opinnäytetyön laatimiseen eri vaiheineen sisältäen aiheeseen liittyvään kirjalliseen materiaaliin perehtymisen, tarvittavan kirjallisuuden hankkimisen ja teoreettisen pohjan keräämisen.

Tutkimuksen aineiston keräämiseen käytettiin organisaation testausstrategian mukaista välinettä: HP:n Quality Centeriä, josta analysoitavat raportit oli mahdollista saada etukäteen kirjatun ja testauksen suorittamisen aikana syntyneen datan avulla. Ketterissä projekteissa enenevässä määrin käytössä on työväline, joka mahdollistaa ketterän työmenetelmän story boardien ja käyttäjätarinoiden hallinnoinnin. Linkitystä ketterän testauksen työvälineiden ja Quality Centerin tietojen välillä kehitetään jatkuvasti, jotta testauksen seurantaan tarvittavia

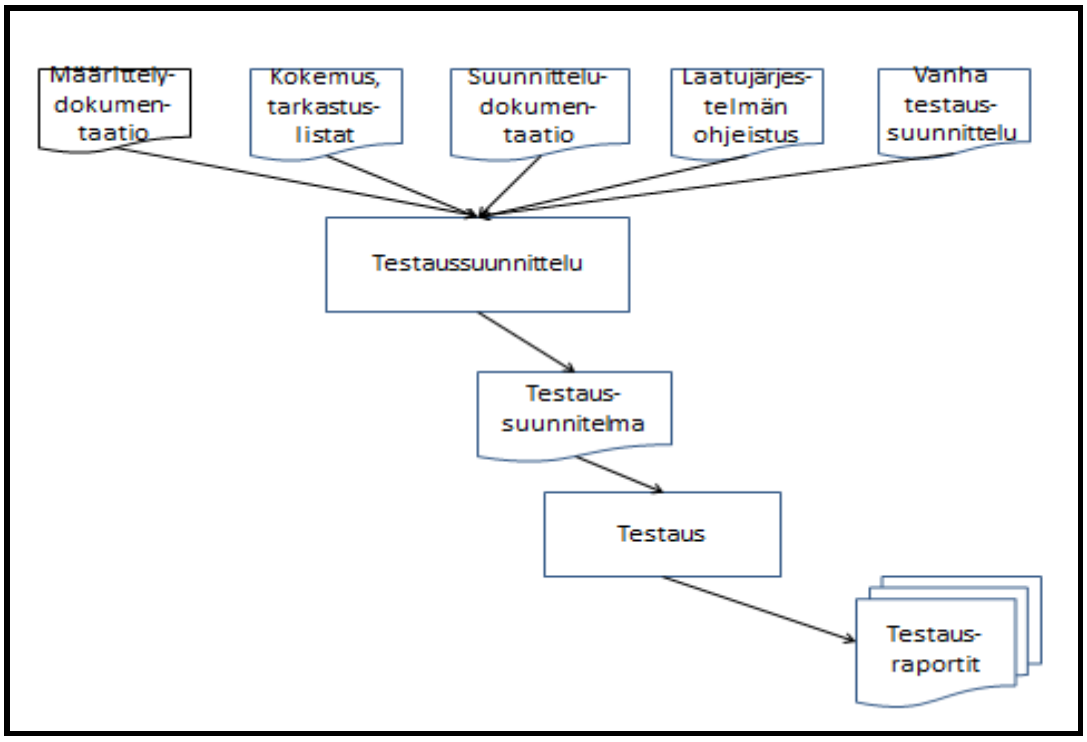
tietoja ei tarvitsisi kirjata kahdelle välineelle, vaan kertaalleen kirjoitetut tiedot olisivat mahdollisimman laajalti integroitavissa.

Projektin aikataulun seuranta, resurssien kiinnittäminen sekä työajan seurannan välineenä on Clarity, johon suunnitellaan projektin tehtävät ja kirjataan näille tehtäville kirjatut tunnit. Clarity on pääsääntöisesti projektipäälliköiden työväline, jossa seurataan koko projektin etenemistä ja suunnitelmien pitävyyttä.

3.1 Kerättävän pohjadatan ominaisuudet

Tutkimuksen pohjaksi kerättiin kehitysprojektien järjestelmätestauksen aikana syntynyttä dataa, joka muodostui testauksen suunnittelun ja suorittamisen aikana käytettävälle testauksen hallintavälineelle. Pohjatiedon oikeellisuus ja luotettavuus muodostui sitä kautta, että siihen ei pääse testausvälineen ohitse vaikuttamaan, vaan jokainen yksittäinen tieto tallentuu välineelle testaajan tekemien toimenpiteiden kautta ja on raportoitavissa eri kriteerein.

Kun testaussuunnittelu on tehty suunnitelmallisesti linkittäen todennettavat vaatimukset aina yksittäiseen testitapaukseen asti, on aukoton seuranta mahdollista vaatimusmäärittelystä aina kirjattuun virheilmoitukseen ja sen luokitteluun (Kuva 1). Tutkimukseen tarvittava data muodostui testaussuunnittelun pohjaksi määritellyistä vaatimuksista sekä niihin suunnitelluista yksittäisistä testitapauksista. Testitapauksien suunnittelun etenemisellä pystyttiin todentamaan, että kaikki kirjatut vaatimukset tulivat katettua ja ne oli priorisoitu vaatimusten tärkeysasteen mukaan.



Kuva 1: Testauksen suunnittelu (Haikala. 1998, 289)

Testauksen etenemisen seurantaan ja testattavien toimintojen laadun seurantaan tarvittiin tietoa suoritetuista testitapauksista ja niiden suorittamisen tuloksesta. Onnistuneet testitapaukset muodostivat tietoa testauksen etenemisestä ja samalla myös testauksen valmistumisasteesta verrattuna suunniteltuun käyttöönottopäivään. Epäonnistuneet virheeseen ja virhekorjaukseen päätyneet testitapaukset kerryttivät virhekantaa, josta oli mahdollista raportoida virheitä eri kriteereillä: vakavuusasteittain, toiminnoittain, virheen iän mukaan tai virhekirjausten trendin mukaan.

Testauksen kompleksisuudesta on mahdollista saada raportille tietoa ajettujen testikierrosten määrästä tai testitapauksen suorittamiseen kuluvasta ajasta. Nämä tiedot eivät kuitenkaan välttämättä kerro todellista tilannetta, testitapauksia voidaan ajaa useamman kerran myös varmistuksen vuoksi tai kun halutaan suorittaa samaa testiä käyttämällä vaihtuvaa testidataa.

Testitapauksen suorittamiseen kuluvaa aikaa ei voida myöskään suoraan ottaa analysoitavaksi tiedoksi, koska testaajat toimivat eri tavalla testejä ajaessaan eivätkä suorita konemaisesti testejä toisensa jälkeen.

3.2 Testauksen hallintaväline Quality Center

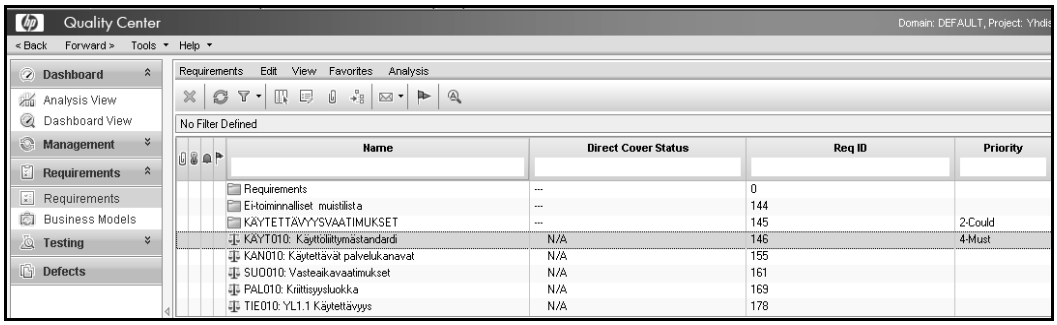
Quality Center (QC) on integroitu työkalu testausprosessin hallintaan. Sen sisältämät neljä osa-aluetta vaatimusten hallinta-, testaussuunnittelu-, testien ajo- ja virheiden kirjaus

osio mahdollistavat koko testausprosessin hallinnan keskitetyssä tietokannassa. Testauksen seurantavälineelle voidaan suunnitella sekä manuaalisia että automaattisesti ajettavia testejä ja sitä voidaan käyttää niin perinteisellä vesiputousmenetelmällä kuin ketterillä menetelmillä tehtävään suunnitteluun. Quality Centerin käyttö perustuu käyttäjien rekisteröintiin ja käyttötunnusten määrittelemiseen välineelle ennen sen käyttöönottoa. Väline on jokaiselle tarvittaessa asennettava ohjelmisto, jonka käyttö määräytyy käytössä olevien käyttäjätunnusten määrän perusteella.

Työvälineen käyttö on laajalle levinnyttä ja on myös useiden organisaation käyttämien alihankkijoiden ja toimittajien käytössä, joten testauksesta on mahdollista saada mahdollisimman läpinäkyvää, kun eri osapuolten suorittama testaussuunnittelu ja testaus ovat dokumentoitavissa samalle välineelle. Väline mahdollistaa myös linkitykset olemassa olevaan dokumentaatioon, esim. word, excel, power point yms. muodossa. Sen eri osioihin on myös mahdollista liittää kuvakaappauksia.

3.2.1 Quality Center: Vaatimukset

Vaatimusten hallintaosiossa (Requirements) kuvataan järjestelmälle asetetut vaatimukset sekä liitetään testitapaukset ja havainnot vaatimuksiin (Kuva 2). Lisäksi tarkastellaan miten kattavasti vaatimuksia on testattu ja tarkastellaan kunkin vaatimuksen kohdalla testauksen eteneminen vastaan virhetilanne.



Kuva 2: Vaatimusten hallinta

3.2.2 Quality Center: Testaussuunnittelu

Testaussuunnitteluosiossa (Test Plan) suunnitellaan testitapaukset sekä analysoidaan ja hallitaan testisuunnitelmia. Testitapausten suunnittelu voidaan aloittaa heti kun määrittelydokumentaatio on katselmoitu ja hyväksytty ja testauskohteet suunniteltu. Testitapaukset suunnitellaan pääsääntöisesti ennen testauksen aloitusta. Testauksen yhteydessä testitapauksia kirjoitetaan tarpeen vaatiessa lisää.

Testitapausten suunnittelun pohjana käytetään saatavilla olevaa määrittelydokumentaatiota sekä muuta lisätietoa, jota testaussuunnittelija on testattavasta kohteesta kerännyt. Suunnittelussa tulee huomioida erityyppiset testitapaukset, kuten toiminnalliset, kuormitus, tietoturva, poikkeustilanteet ja automatisointi. Testitapauksen askelissa, joita kirjoitetaan vähintään kaksi, kuvataan tarkemmalla tasolla, miten kuvaukseen kirjoitettu asia testataan (Kuva 3). Testitapaus tulisi kirjoittaa niin, että kaikki ymmärtävät, mitä testitapauksessa tulee tehdä.

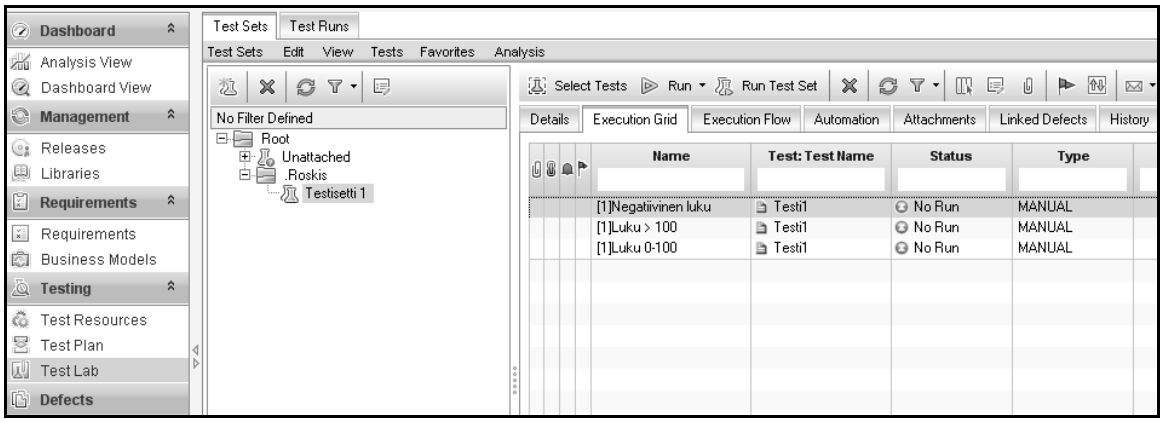
Askeleen nimi	**Kuvaus** yksi askel, jossa on kerrottu mitä testauksessa tehdään ja mikä on odotettu lopputulos	**Haluttu lopputulos**
Askel 1	Kirjaudu yrityksen verkkosivulle www.yritys.fi	Kirjautumissivu aukeaa
Askel 2	Anna käyttäjätunnus ja salasana	Käyttäjätunnus ja salasana on hyväksytty ja kirjautuminen onnistunut
Askel 3	Valitse haluttu toiminto menuvalikosta	Haluttuun toimintoon on siirrytty aloitussivulta

Kuva 3: Testitapauksen rakenne

Testitapauksia priorisoidaan suunnittelun yhteydessä. Priorisointia käytetään jatkossa suunniteltaessa testitapausten suoritusjärjestystä. Näin varmistetaan, että ainakin kriittisimmät testitapaukset on suoritettu tilanteessa, jossa testaukselle varattu aika loppuu kesken.

3.2.3 Quality Center: Testien ajaminen

Testien ajo-osiossa (Test Lab) suunnitellaan testijaksot (Test set), ajetaan suunnitellut testitapaukset ja analysoidaan testien tulokset (Kuva 4). Testaaja suorittaa kaikki sovitut testitapaukset, priorisoidussa järjestyksessä. Testaaja kirjaa jokaisen testitapauksen suorituksen tilanteen testauksen hallintavälineeseen: No Run = testiä ei ole ajettu (oletusarvo), Pass= testi meni odotetusti läpi, Failed = testin lopputulos poikkesi odotetusta, Not Completed = testin suorittaminen jäi kesken.



Kuva 4: Testien ajaminen

2.1.4 Quality Center: Virheiden kirjaaminen

Virheilmoitusosiossa (Defects) kirjataan testien aikana havaitut virheet sekä hallinnoidaan ja analysoidaan virhekirjauksia. Testijakson suorittamisen yhteydessä, testaaja kirjaa jokaisen tapauksen kohdalla ilmenevät poikkeavat havainnot. Jos poikkeamia on niin runsaasti, että ne haittaavat jaksolle asetetun tavoitteen saavuttamista, on suoritus keskeytettävä.

Virheilmoitus tehdään aina testitapauksen suorittamisen yhteydessä, jotta yhteys säilyy testitapaukseen, jossa virhe ilmeni. Mikäli virhekirjaus tehdään myöhemmässä vaiheessa, on se ehdottomasti liitettävä olemassa olevaan testitapaukseen.

2.1.5. Quality Center: Toteutuserien hallinnointi

Yllämainittujen neljän yleisimmin käytössä olevan osion lisäksi voidaan käyttää myös Release – osiota, jossa suunnitellaan eri käyttöönottoeriä sekä liitetään niihin testattava sisältö linkittäen suunnitellut testit.

Release-osiossa eri käyttöönottoerien valmistumista voidaan seurata ja puuttua riittävän aikaisin mahdollisiin ongelmiin toteutuksen etenemisessä. Lisäksi voidaan jo suunnitteluvaiheessa arvioida erillisen käyttöönottoerän kokoa ja tehdä mahdollisia priorisointeja aikataulun ja käytettävissä olevien resurssien suhteen.

4 Järjestelmäkehityksen eri menetelmät ja laatu

Järjestelmäkehityksen menetelmistä eniten käytössä on perinteinen vesiputousmalli, jossa testauksen eri vaiheet suunnitellaan ja toteutetaan peräkkäin. Yhä enenevässä määrin ohjelmistokehityksessä on siirrytty ketteriin kehitysmalleihin (RAD, SCRUM) tai inkrementaalisiin menetelmiin (Agile). Riippumatta kehitysmallista tietyt ohjelmistokehityksen vaiheet on ero-

teltavissa etenemisen aikana: määrittely, suunnittelu, toteutus ja testaus. (Pohjonen. 2002, 42)

Ohjelmistokehitys riippumatta käytettävästä kehitysmallista saa alkunsa asiakkaan tai tilaavan liiketoimintayksikön tarpeesta uuden toiminnallisuuden luomisesta tai olemassa olevan tuotteen parantamisesta tai vian korjauksesta. Kehityksen kohteena voi olla tuote, palvelu tai liiketoimintaprosessi. Useimmiten järjestelmäkehitys läpi viedään projektin muodossa tai muutoin muodollisesti, jotta tekemisen seuranta työn edetessä on mahdollista (McGregor, J. A & Sykes, D.A. 2001, 1 – 2).

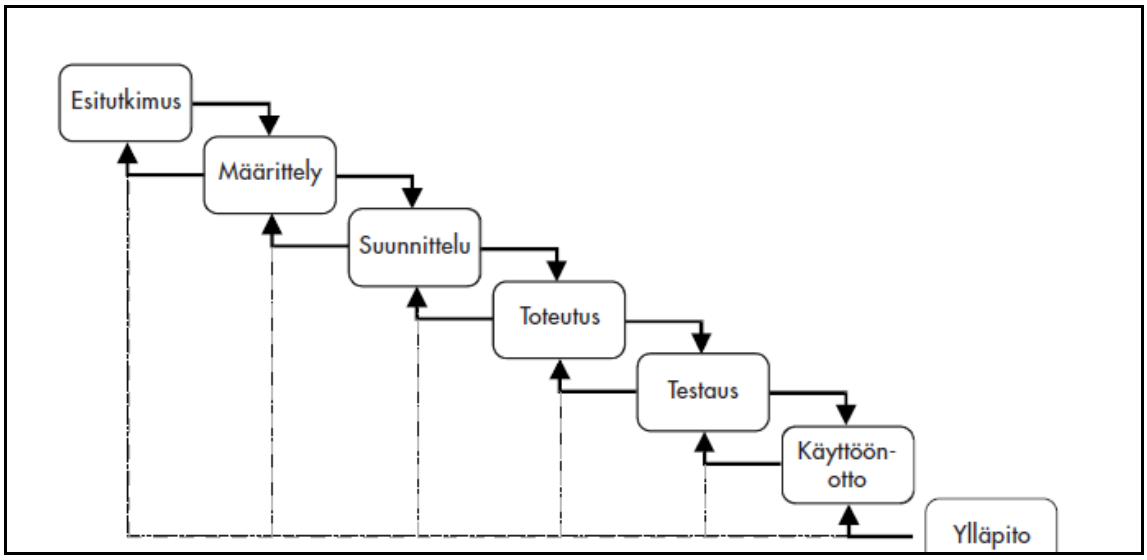
Järjestelmäkehityksen toteutuksen vastaavuutta haluttuihin ominaisuuksiin ja lopputuloksen laadullisen ja toiminnallisuuden varmistamiseksi järjestelmäkehityksen aikana suoritetaan erityyppistä testausta. Jotta testaus voidaan kohdistaa oikeisiin asioihin ja riittävällä laadulla on sen suorittamista seurattava. Tähän tarkoitukseen määritellään testaukselle omat mittarit, jotka voivat olla testausprosessia, testattavan tuotteen laatua tai testauksen projektimetriikkaa mittaavia. Mittari voi olla mikä tahansa systemaattinen tapa kerätä tieto järjestelmän kehityskaaren aikana liittyen dokumentaatioon, prosessiin tai kehitettävään koodiin (Wiegers. 2003, 548).

Testauksen ja laadunvarmistuksen tavoitteena on oikeiden asioiden testaus oikea-aikaisesti ja testauksen läpinäkyvä edistymisen raportointi. Testauksella varmennetaan järjestelmien toimintaa noudattamalla määriteltyjä laatukriteereitä ja palvelutasoja noudattamalla selkeitä testaussuunnitelmia ja sisäisiä sopimuksia. Kaiken perustana on liiketoimintaprosessien ymmärrys ja asiakaslähtöinen toimintatapa (Haikala. 1998, 284).

Testausmetriikoiden tavoitteena on lisätä tuotekehityksen aikaista näkyvyyttä sovelluksen ja järjestelmän laatuun sekä läpinäkyvyyttä toimintatapoihin. Metriikat antavat faktoihin perustuvaa tietoa päätöksenteon tueksi ja antavat mahdollisuuden tuotteen laadun ja toimintatapojen parantamiseen. Metriikkojen systemaattinen käyttö on keskeisessä asemassa testauksen ja koko ohjelmistokehityksen kyvykkyyden parantamisessa. Kohdeyrityksen osalta tavoite on nostaa metriikkojen käyttö vähintään alan keskimääräiselle tasolle.

4.1 Kehittäminen vesiputousmallilla ja laadunvalvonta

Vesiputousmallissa järjestelmäkehitys toteutetaan toisiaan seuraavissa peräkkäisissä vaiheissa, joissa aina edellisen vaiheen lopetuskriteerit ja dokumentaatio on seuraavan vaiheen aloituskriteerinä ja pohjadokumenttina (Kuva 5).



Kuva 5: Tietojärjestelmien kehittäminen vesiputousmallilla (Pohjonen. 2002, 42)

Laadunvalvonta vesiputousmallissa tapahtuu tarkistuspisteinä vaiheesta toiseen siirryttäessä. Laadunvalvontaa voidaan suorittaa erillisinä tarkistuspisteinä ja niiden avulla voidaan varmistaa kehitysprojektin laadukas eteneminen ja synkronoida eri vaiheiden tekemisen yhteen.

Esitutkimusvaiheessa asetetaan liiketoiminnalliset vaatimukset kehittävälle tuotteelle tai palvelulle ja sen tavoitteena on perustella kehitysprojektin tarpeellisuus. Ennen kehitysprojektin liikkeelle lähtöä on ymmärrettävä asiakkaan tarpeet riittävän hyvin, jotta on mahdollista lähteä arvioimaan, miten kehitystyötä lähdetään viemään eteenpäin: tarvittava osaaminen, aikataulutus, resurssien määrä, järjestelmäriippuvuudet (Pohjonen. 2002, 27 – 28).

Määrittelyvaiheessa esitutkimusvaiheen toteutettavat asiakasvaatimukset analysoidaan ja rajataan niistä lopulliset toteuttaviksi sovitut ohjelmiston ominaisuudet. Tälle lopulliselle toteutuskokonaisuudelle suunnitellaan toteutuksessa tarvittavat järjestelmä- ja ohjelmistovalinnat sekä kuvataan toteutettavan ohjelmiston toiminnallisuudet käyttötapauksiksi tai toiminnallisiksi määrittelyiksi. Määrittelyvaiheessa otetaan kantaa myös toteutuksen ei-toiminnallisiin vaatimuksiin, kuten suorituskykyyn, käytettävyyteen, tietoturvaan ja ulkoasustandardeihin (Haikala. 1998, 38 – 41).

Ohjelmiston toiminnallisten ja ei-toiminnallisten määrittelyjen pohjalta suunnitellaan toteutettava kokonaisuus. Suunnittelun lähtökohtana on arkkitehtuurisuunnittelu, joka luo asemakaavan eri järjestelmämoduulien kokonaisuudelle ja niiden välisille liittymille. Varsinainen ohjelmistosuunnittelu pitää sisällään järjestelmän komponenttien suunnittelun koodauksineen. Laadunvarmistuksena toteutusvaiheessa käytetään esim. koodikatselmointeja, joissa kehittäjät tarkistavat toistensa tekemää koodia parikatselmointien avulla. Paitsi laadun var-

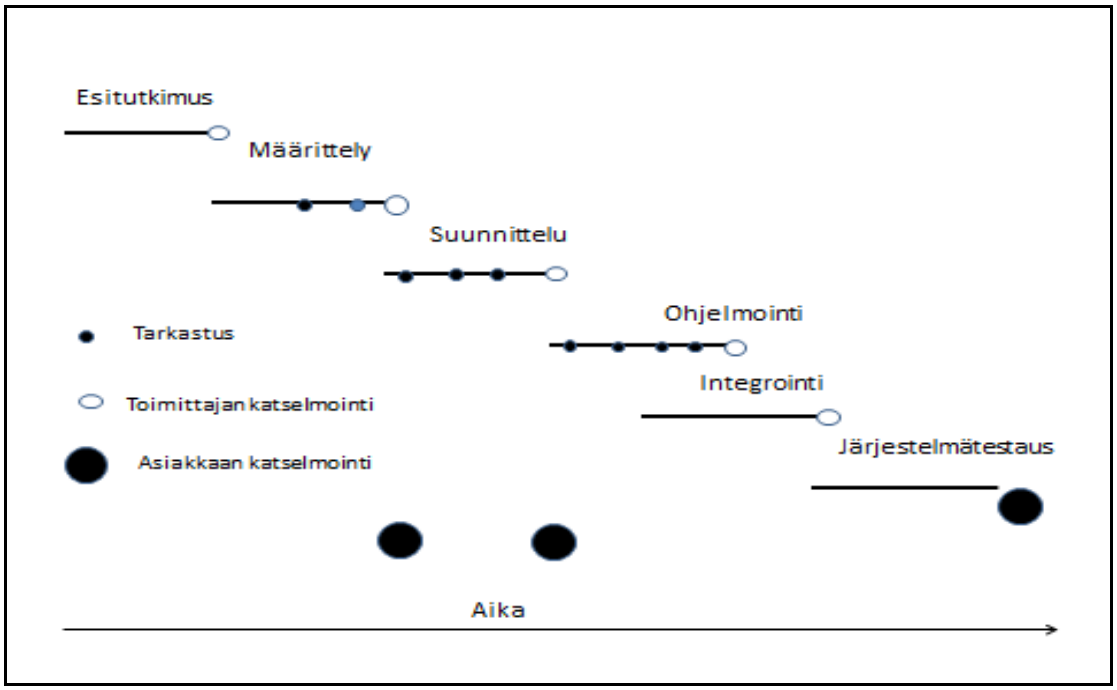
mistamisessa, parikatselmointi toimii loistavana keinona oppia toiselta toteuttajalta (Wiegers. 2003, 546 – 547).

Kun kehitettävä järjestelmä on suunniteltu ja toteutettu tehtyjen määrittelyjen mukaisesti, testauksella varmistetaan kokonaisuuden toiminnallisuus tehtyjä määrittelyjä ja asiakasvaatimuksia vasten. Testauksen lähtöaineiston varmistamista kannattaa tehdä jo osallistumalla vaatimusten ja toiminnallisten määrittelyjen katselmointeihin. Katselmoinnissa varmistetaan että määrittelyihin on kuvattu testauksen kannalta oleelliset asiat tarpeeksi yksiselitteisesti. Tarkastamalla testauksen lähtöaineisto, varmistetaan että testataan oikeita asioita. Jos lähtöaineisto on epäselvää, puutteellista tai sitä ei ole, ei voida olettaa että myöskään testaus pystyisi varmistamaan että järjestelmä toimii toivotulla tavalla.

Vesiputousmallissa, jossa toteutusvaiheet seuraavat toisiaan selkeinä osinaan laadunvalvontaa seurataan vaiheiden välissä määriteltyinä laatuportteina (Kuva 6). Laatuportteihin määritellään kehitettävän sovelluksen valmiusaste ja dokumentaatio, joka on oltava valmiina ennen seuraavaan vaiheeseen siirtymistä (Pohjonen. 2002, 42).

Laatuporteissa vaadittava dokumentaatio läpikäydään katselmoinneissa, joihin osallistuu henkilöitä liiketoiminnasta ja kehitysprojektista. Katselmointitilaisuudet ovat määrämuotoisia ja niiden perusteella tehtyjen katselmointipöytäkirjojen perusteella kehitettävä ohjelmisto voi jatkaa seuraavaan vaiheeseen tai siltä vaaditaan tarvittavat täydennykset puuttuvasta toteutuksesta tai dokumentaatiosta (Black. 2004, 546–547).

Katselmoinnin kohteena voi olla ohjelmistokoodi, arkkitehtuurikuvaukset, määrittelyt, projektisuunnitelmat, testaussuunnitelmat, testitapaukset yms.



Kuva 6: Katselmointimalli (Haikala. 1998, 52)

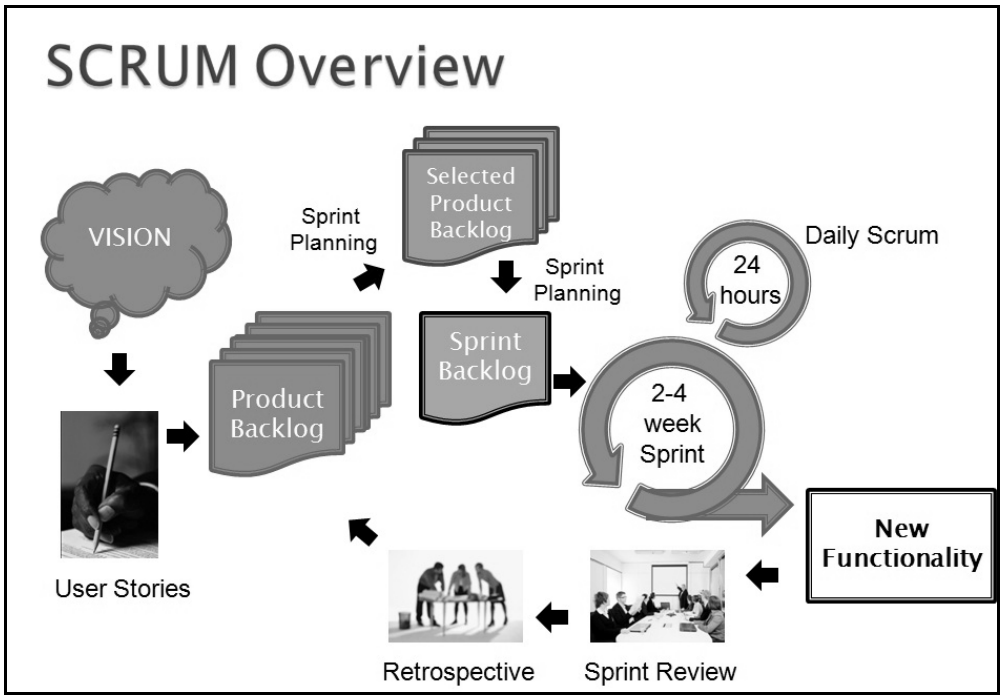
4.2 Kehittäminen ketterillä menetelmillä ja laadunvalvonta

Perinteisessä vesiputousmallissa eri kehitysvaiheet seuraavat toisiaan eri vaiheina, ketterissä menetelmissä kierrokset toteutetaan joko yhtä aikaa tapahtuvina ja nopeammassa syklissä. Mitä tahansa kehitysmallia käytetäänkin, lopullisen laadun mittari kehitettävälle tuotteelle on testaus ja sen toteama virheetön ja määrittelyjen mukainen tuote. Tästä johtuen myös ketteriä menetelmiä käytettäessä määriteltävä prosessiin tarkistuspisteitä, joissa laatua mitataan.

Jotta laatua voidaan mitata ketterien menetelmien kehittämisessä, on laadun arvioimiselle myös asetettava tavoitteet, joihin lopputulosta verrataan. Välineenä ketterän kehittämisen projektien testauksen seurannassa käytetään yleisesti käytössä olevaa testausvälinettä, jotta eri kehitysmenetelmillä toteutettavien projektien laadunvarmistuksen pohjalle muodostunut tieto on yhteismitallista.

4.2.1 Scrum-kehittäminen

Scrum-mallissa tilaajan tahtotila kerrotaan käyttäjätarinoina ja niistä kerätään toteutuskokonaisuudet tehtäviksi töiksi Product Back Logille, josta koostetaan kokonaisuuksia kehityssykleiksi eli Sprinteiksi (Kuva 7). Jokainen yksittäinen Sprint sisältää suunnittelu-, toteutus- ja testausvaiheet ja niiden päätteeksi voidaan esittää tilaajalle valmis kokonaisuus. Kehittämisen tahti nopeutuu ja kehitettävän tuotteen laatu on nähtävissä aikaisemmassa vaiheessa perinteiseen kehitysmalliin verrattuna.



Kuva 7: Scrum kehittäminen

Ketterissä menetelmissä toteutuksen pohjana oleva määrittelydokumentaatio yleisesti katselmoidaan edellä mainitulla tavalla. Useimmiten ketterissä menetelmissä vaatimusten sijaan määritellään käyttäjätarinoita, joiden pohjalta esim. testauksen skenaariot suunnitellaan, Ketterän kehityksen aikana määrittely, suunnittelu, toteutus ja testaus suoritetaan toisiaan seuraavissa sprinteissä, joihin kootaan loogisesti yhteenkuuluvia toiminnallisuuksia.

Ketterän sprintin lopetuksen laadunmäärittelynä on definition-of-done (DOD), joka määrittele tason, jolla sprintin tuotokset ovat valmiita käyttöönotettavaksi. DOD voidaan määritellä kehitettävän ominaisuuden, sprintin tai suuremman kokonaisuuden laadulle (Dhaval, P. 2008). Toinen agile-kehittämisen laadun määritelmä on definition-of-ready (DOR), joka määrittelee, missä vaiheessa kehitettävä tuote on valmis (Agile alliance and Institute Agile. 2011).

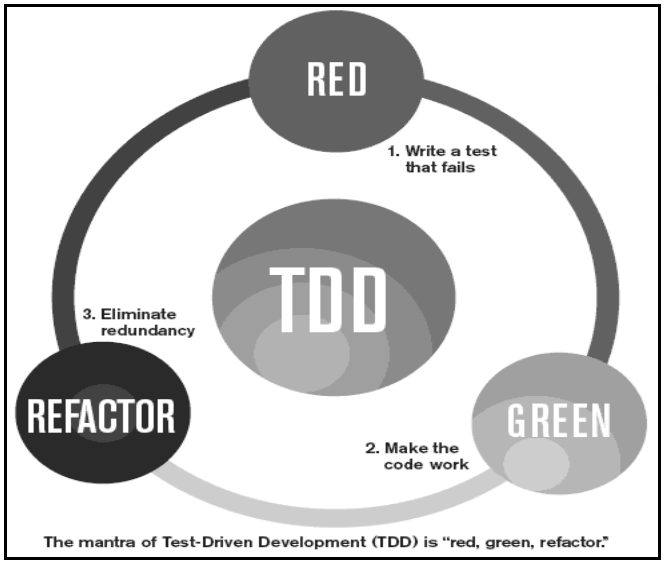
Esimerkkejä DOD:in määrittelyille:

- tarina on toteutettu Product Ownerin laatiman kuvauksen mukaisesti
- automaatiotestit on kirjoitettu ja testit on suoritettu hyväksytysti
- ylläpidon kannalta tarvittava ohjeistus on dokumentoitu

4.2.2 TDD Test Driven Development

Test-Driven-Develoment – menetelmässä testauksen perinteiset roolit laajenevat, koska jo kehittäjä tekee testausta. Toiminnallisuuden kehittäjä suunnittelee automaattiset testit ja iteroi kehitettävää toiminnallisuutta, kunnes testi saadaan suoritettu onnistuneesti (Kuva 8). Pyörittäessään iteraatioita testausautomaation avulla, koodiin tehtävät muutokset tallentuvat automaattisesti ja tehtävästä yksikkötestauksesta syntyy dokumentaatio käytettävälle välineelle. Samalla saadaan talteen myöhempää käyttöä varten automatisoidut regressiotestit, joilla tulevien muutosten jälkeen voidaan varmistaa kokonaisuuden toiminnallisuus.

Riskinä TDD-kehittämiselle on se, että kehittäjä suunnittelee vain testejä, jotka suoritetaan onnistuneesti, ns. happy-day – skenaariot ja negatiiviset tapaukset jäävät todentamatta. TDD sopii hyvin yksikkötestausvaiheeseen ja automaatiotestauksen pohjadokumentaation aikaan saamiseksi (Sogeti. 2009).

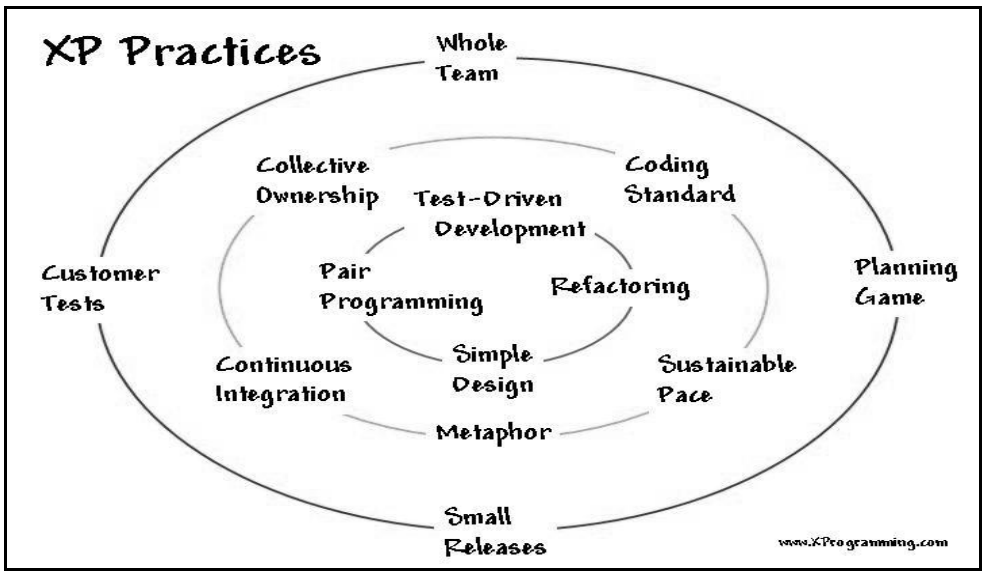


Kuva 8: Test-Drive-Development

4.2.3 eXtreme Programming XP

Iteratiivisesti kehitettävä XP metodi perustuu ketterän kehittämisen mallin mukaisesti moneen pienempänä kokonaisuutena toteutettavaan toiminnallisuuksien pakettiin, jotka seuraavat toisiaan (Kuva 9). Tämä malli sopii hyvin käytettäväksi erityisesti pienempiin projekteihin. Kehitettävää koodia parannellaan jokaisen iteraation kohdalla, vaikka tarkoituksena ei ole muuttaa varsinaisesti koodin toteuttamaan toiminnallisuutta vaan parantaa kirjoitetun koodin laatua iteraatiosta toiseen.

Kehityksen aikana tuotteen tai palvelun loppukäyttäjä on lähempänä koodin kehittäjiä ja voi antaa varhaisessa vaiheessa palautetta sekä toteutukseen että toiminnallisuuden testauksen antamiin tuloksiin. Näin kehityskohteen laadun valvominen on mahdollista heti ensimmäisen toimivan koodin testauksen tulosten perusteella ja mahdollisuus vaikuttaa sen parantamiseen syntyy jokapäiväisen kommunikaation pohjalta (Wells, Don. 2009).



Kuva 9: XP Menetelmä (Wells, Don. 2009)

5 Tuotteen laadun arviointi eri metriikoilla

Mittareiden antamalla tiedolla on oltava jonkinlainen arvo tai hyöty mitattavalle asialle. Laadunvarmistuksen näkökulmasta mittareiden antama tieto on arvokasta kehitettävän työn kypsyydelle, mutta myös sen avulla voidaan arvioida käytettävän prosessin tehokkuutta ja löytää kehityskohteita (Kooman, T., van der Aaist, L., Broekman, B. & Vroon, M. 2006, 569 – 570).

Monia laatustandardeja julkaissut kansainvälinen teknisen alan järjestö IEEE (Institute of Electrical and Electronics Engineering) on määritellyt testauksen olevan prosessi, joka analysoi järjestelmää löytääkseen eroavaisuuksia testattavana olevien ja määriteltyjen ominaisuuksien välillä (Hutcheson. 2003, 13).

Nämä eroavaisuuden ilmenevät virhehavaintoina, joiden esiintymistiheys testauksen aikana määrittelee testauskohteen laadun. Mittareiden avulla saadaan seurattavasta laadusta puolueetonta arviointia, koska virhekirjaus osoittaa selkeästi kohdan, jossa määrittelyt ja toteutettu ominaisuus eivät kohtaa. Ainoa seikka, jossa testaajan oma mielipide ratkaisee, on virheen haitta-aste ja sen korjauksen prioriteetin vaikutus testauksen etenemiselle. Puoluee-

tonta mittaamista testauksen etenemisen aikana kehitettävän järjestelmän osalta on mahdollista saada vain lahjomattomien mittareiden antamaa dataa hyödyntämällä.

Mittaamisen tarkoituksena on paitsi todentaa testattavan järjestelmän laatua verrattuna laadittuihin loppukäyttäjän vaatimuksiin, myös arvioida testausprosessin laatua (Wiegers. 2003, 399). Käytettäessä yhteistä testausvälinettä, voidaan olla varmoja siitä, että sovittua prosessia testauksen valmistelussa, suunnittelussa, suorittamisessa ja raportoinnissa käytetään. British Standards Institutionin standardin BS 4778 mukaan prosessina laadunvalmistukseen kuuluu kaikki suunnitellut ja systemaattisen prosessin mukaiset tarvittavat toiminnot, joilla varmistetaan, että kehitettävä tuote tai palvelu lopputuloksen laatu on määriteltyjen vaatimusten mukaista (Hutcheson. 2003, 25)

Tähän kokonaisprosessiin kuuluu testauksen lisäksi kaikki sitä ennen suoritetut laatua varmistavat toiminnot, kuten katselmoinnit, hyväksymiskriteerien määrittelyt, testauksen läpiviennin suunnittelu sekä varsinaisen testauksen etenemisen suunnittelu. Suunnitelmien ja sovittujen tarkistuspisteiden seuranta organisaatiossa, jossa vuosittain läpi viedään satoja erikokoisia projekteja ja niiden rinnalla

Seuraavissa kappaleissa on eri testauksen metriikkatyyppien jaotteluna käytetty soveltaen tutkittavan yrityksen testausstrategiaan mukaista jakoa.

5.1 Testauksen laatumetriikat

Testauksen laadun metriikoilla voidaan arvioida projektissa tai hankkeessa tuotetun palvelun, sovelluksen tai osajärjestelmän laatutasoa ja testauksen laadun trendit. Laadun metriikoita ovat mm. vikametriikat, joilla todennetaan testauksen aikana löydettyjen avointen virheiden määrä ja niiden vakavuusluokittelu.

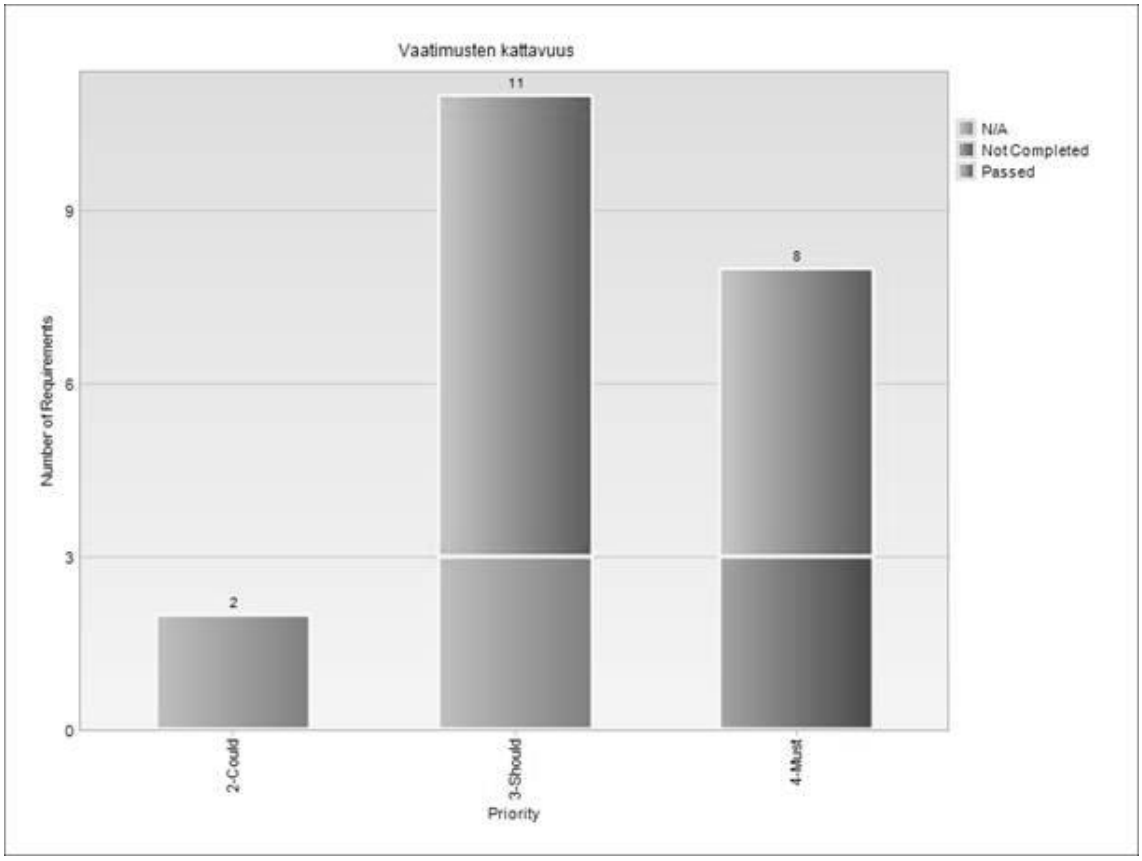
Lisäksi seurataan testauksen aikana löydettyjenvirheiden historiatrendiä, josta selvittää virhekirjausten tiheyden ja vakavuuden muutokset testauksen edetessä.

5.1.1 Testauksen laatumetriikka 1. Vaatimusten kattavuus

Laadun metriikkana käytetään myös vaatimusten kattavuuden seuraamista testaussuunnittelun aikana (Kuva 10). Testaussuunnittelulla on katettava ainakin kriittisimmät vaatimukset ja niiltä osin, kuin tehtyjä vaatimuksia ei voida kattaa testisuunnittelulla, on tilanteesta tehtävä merkintä kyseiseen vaatimukseen tai poistettava vaatimus kokonaan epävalidina.

Ennen testauksen aloittamista määritellään vaatimukset testattavalle järjestelmälle. Vaatimukset voivat olla joko järjestelmävaatimuksia, liiketoimintavaatimuksia tai ei-toiminnallisia vaatimuksia. Järjestelmätestauksen tärkein kohde on järjestelmävaatimusten testaus. Liiketoimintavaatimukset muodostavat hyväksymistestauksen pohjan ja ei-toiminnalliset vaatimukset ovat lähtökohtana mm. kuormitus ja tietoturvatestaukselle.

Käytettävällä testausvälineellä on mahdollista linkittää testauksen suunnitteluosioon suunnitellut testit vaatimuksiin, ja siten testaussuunnittelun edetessä seurata testikattavuutta



Kuva 10: Vaatimusten kattavuus

5.1.2 Testauksen laatumetriikka 1. Vaatimusten kattavuuden analysointi

Kehitettävän järjestelmän vaatimusten priorisointi on tärkeätä, jotta testauksen painopiste ja testauksen suunnittelu voidaan kohdistaa tärkeimpiin toiminnallisuuksiin ja varmuus kriittisten toimintojen laadunvarmistuksesta voidaan saada testauksella (McGregor, J.A. & Sykes, D.A. 2001, 76–77). Käytettävä testaussuunnitteluvälineen asteikko testattaville vaatimuksille on Must – Should – Could ja joskus myös käytetään alinta asteikkoa Would (MoSCoW). Korkeimmalla prioriteetilla on Must, joka pitää sisällään vaatimukset, jotka on ehdottomasti katettava testaussuunnittelulla, keskitasoa Should edustaa vaatimukset, jotka tulisi sisällyttää testaussuunnitteluun, mutta ne voivat olla todennettavissa muullakin tavalla. Alimman vaatimus-

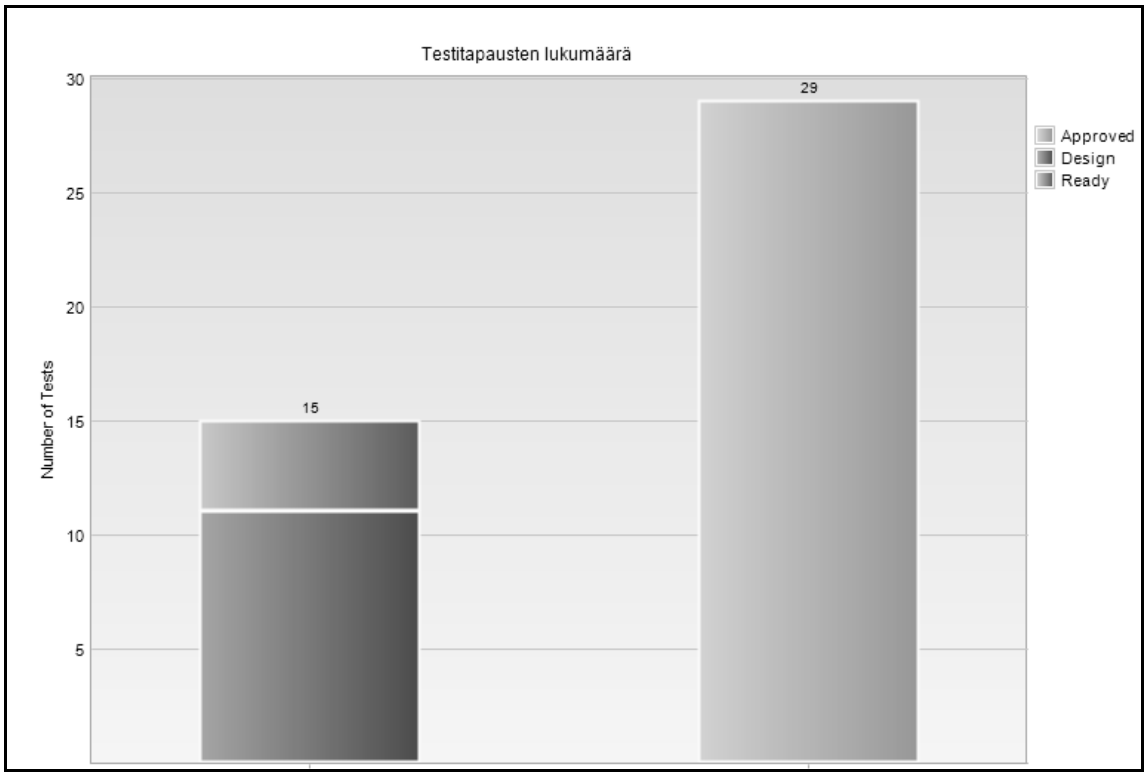
taso muodostavat Could-taso vaatimukset, jotka yleensä pitävät sisällään testitapaukset, jotka voidaan suorittaa, kun korkeamman tason vaatimukset on läpikäyty. Useimmiten tämän tason vaatimukset toteutetaan aikataulun niin salliessa (Gardner, B. 2009)

Kuvassa 9. raportoitu kehitysprojekti vaatimusten kattavuusjakauma on tyypillinen kehitysprojekteille, suurin osa vaatimuksista asettuu prioriteetiltaan keskitasoon. Testaussuunnittelu ja myös testauksen suorittaminen aloitetaan aina korkeimman tason vaatimuksista, kuvasta näkee että Must – prioriteetin testit ovat kaikki joko suoritettuja (Passed) tai niiden suoritus on aloitettu (Not Completed).

5.1.3 Testauksen laatumetriikka 2. Testaussuunnittelun eteneminen

Edellisessä luvussa todettiin, että testaussuunnittelu on aina aloitettava korkeimman tason vaatimuksista (Kuva 11). Testaussuunnittelu on mahdollista aloittaa heti, kun vaatimukset on määritelty ja kirjattu käytettävälle testausvälineelle ja testauksen pohjatiedoksi tehtävä määrittelydokumentaatio on sillä tasolla, että testitapauksia voidaan aloittaa suunnitteleminen (McGregor, J. A & Sykes, D.A. 2001, 337–339).

Paitsi vaatimuksille, myös testitapauksille suoritetaan katselmointi niiden valmistuttua, jotta voidaan varmistua testauksen laadusta. Niin kauan kun testitapausta ei ole katselmoinnissa hyväksytty on sen tila testausvälineellä Design, testitapaukset muutetaan Approved – tilaan heti hyväksynnän jälkeen. Ready-tilassa testitapaukset odottavat katselmointia tai muutoksia saatujen kommenttien perusteella.



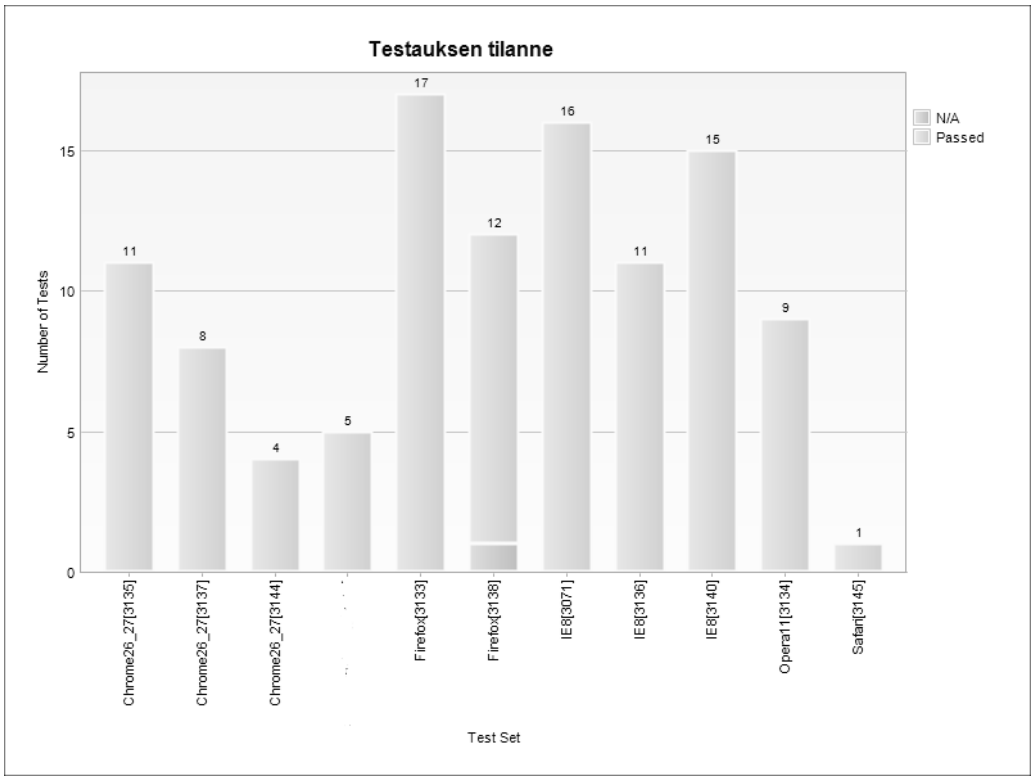
Kuva 11: Testitapausten määrä

Suunniteltujen testitapausten määrä on riippuvainen kehitettävän järjestelmän koosta tai siitä onko kyseessä yksittäisen toiminnon kattava suunnittelu vai isomman projektin tai hankkeen monimutkaisten toimintojen kattavasta testauksen suunnittelusta.

Katselmoitujen testitapausten määrän tulisi olla mahdollisimman lähellä 100 %:ia, jotta voidaan varmistua siitä, että testitapausten sisältö vastaa haluttua toiminnallisuutta ja sen korkeata laatutasoa.

5.1.4 Testauksen laatumetriikka 3. Testauksen suorittamisen eteneminen

Tärkeä tuotteen laadun metriikka on testauksen edistymisen seuranta, jota mitataan suoritettujen testitapausten määrä ja niiden testaustulos (Kuva 12). Tällä metriikalla havaitaan nopeasti kehitettävän sovelluksen ne toiminnallisuudet, joiden testaus on todennettu. Mittari osoittaa toisaalta myös ne toiminnallisuudet, joissa testitapauksia ei ole suoritettu tai ne ovat päätyneet virheeseen. Yksittäisen toiminnallisuuden osalta nähdään laadun taso verrattuna suoritettujen testien määrää virheellisiin tapauksiin ja nopeasti arvioida ne kohdat toteutuksessa, johon testauksen perusteella kohdistuu eniten virhekirjauksia.



Kuva 12: Testauksen tilanne

5.1.5 Testauksen laatumetriikka 3. Testauksen suorittamisen analysointi

Testauksen tilanne testattujen ja testaamattomien sekä virheeseen päätyneiden testitapausten osalta antaa kuvaa siitä, onko testattavien toiminnallisuuksien joukossa joitain ongelma-alueita, joiden testaus ei etene. Varsinkin jos korkealle priorisoidut toiminnallisuudet ja niihin kiinnitetyt testitapaukset eivät etene on syy selvitettävä pikaisesti. Vaikka testitapaukset pyritään priorisoimaan niiden liiketoiminnalle merkitsevyyden takia, testaajat usein haluavat lähestyä testattavaa järjestelmään helposti eli aloittaa testaamisen itselleen tutuista ja helpoista testitapauksista. Tämä johtaa useimmiten siihen, että testauksen alkuvaiheessa testitapauksia suoritettaan lukumääräisesti enemmän, jolloin myös mahdollisuus löytää virheitä. (Kaner. 2002, 27 – 28)

Mahdollista on myös, että priorisointia joudutaan muuttamaan tai osatoiminnallisuuksia poistamaan kokonaan toteutuksesta, jotta testauksessa päästään eteenpäin. Priorisointien muutokset tekee aina liiketoiminta tai kehitystyön tilaaja, jotta varmistutaan kokonaistoimituksen eheys ja tarkoituksen mukaisuus myös muutoksen jälkeen. Uudelleenpriorisoinnissa on myös muistettava varmistaa, että tarvittavat toteutusresurssit ovat käytettävissä aikataulujen ja toteutusten painoarvojen muuttuessa.

5.2 Testauksen vikametriikat

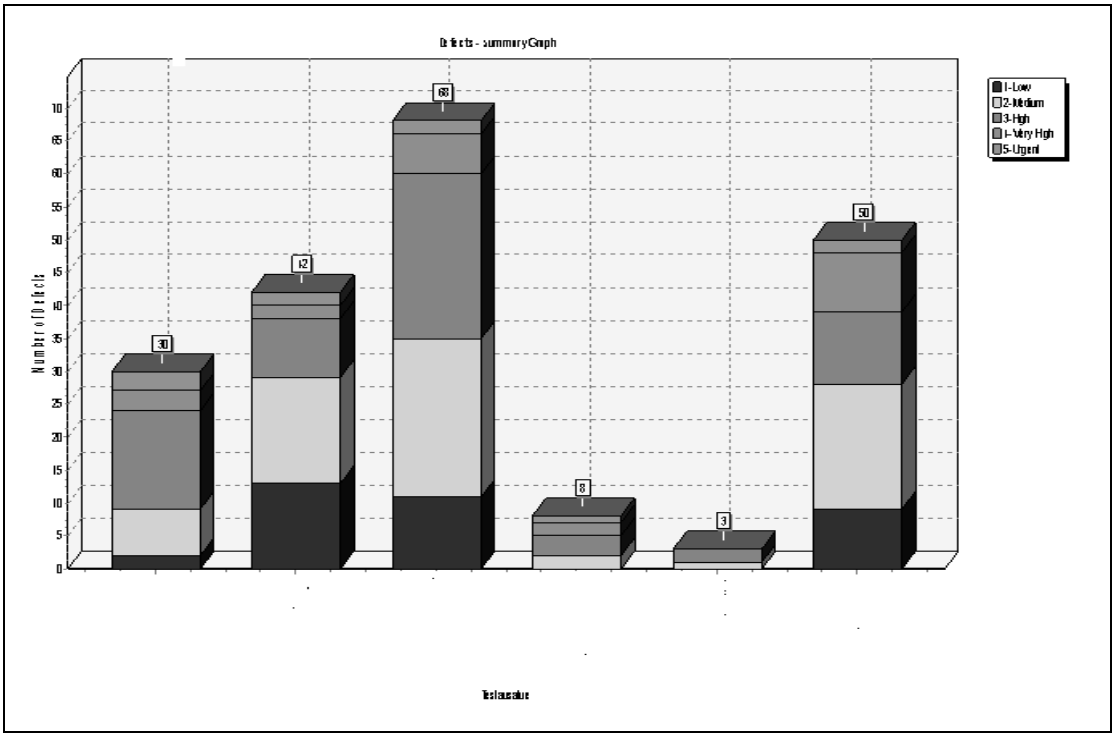
Vikametriikat, joilla todennetaan avointen virheiden määrä ja niiden vakavuusluokittelu on testauksen laadun perusmittareita. Mittareilla kerätään tietoa testattavan kokonaisuuden virheiden kokonaismäärästä vakavuusasteittain ja todetaan testattavan sovelluksen laatu eri testausvaiheissa. Mittaristolla löydetään myös sovelluksen eri kohtiin kertyviä virhekasaumia, kun virheilmoitukseen on raportoitu sovellus, johon virhe kohdistuu. Näin voidaan laadun parannustoimenpiteet ja resurssointi kohdistaa oikeaan toimintoon ja mahdollisesti parantaa koko tuotteen laatua osatekijän laadun parannuksella. Virheiden historiatietoa seuraamalla saadaan arvokasta tietoa virhekorjausprosessin tehokkuudesta ja laadusta.

Testauksen lähestymistavalla on myös merkitystä havaittujen virheiden lukumäärään. Jos testitapauksia suoritetaan noudattamalla tarkasti asetettuja prioriteetteja ja testaus suoritetaan ajamalla vain suunnitellut testit prioriteettijärjestyksessä, virhehavaintojen määrä useimmiten kasvaa testitapausten suorittamisen yhteydessä. Noudatettaessa tutkivaa testausta, jossa taustalla ei ole etukäteen suunniteltuja testitapauksia, vaan testaaja kokeilee järjestelmän ominaisuuksia, saatetaan löytää nopeammin sellaisia virheitä, jotka johdonmukaisessa testauksen etenemismallissa löytyisivät myöhemmin tai ei ollenkaan. Tutkivaa testausta käytetään parhaimmillaan testauksen alkuvaiheessa, jolloin vastaanottotestauksen tavalla tarkistetaan testaukseen saadun sovelluksen kypsyyttä testattavaksi. Tutkiva testaus voi olla myös suunnitelmallisen testauksen täydentäjä, jolloin se usein vaatii testaajalta pitkää testauskokemusta tai ainakin testattavan järjestelmän syväosaamista (McGregor, J. A & Sykes, D.A. 2001, 118 – 121).

5.2.1 Testauksen vikametriikka 1. Havaitut virheet vakavuusluokittain

Mitä tahansa testauksen lähestymistapaa noudatetaankin, tärkeätä virheiden havainnoinnille on se miten virheet luokitellaan korjauksen tärkeysjärjestykseen (Kuva 12). Korkeammalla prioriteetille määritellyt virheet korjataan nopeammin ja tavoitteena on saada ne mahdollisimman pikaisesti uudelleentestattavaksi ja edelleen käsitellyiksi (Kaner, 76). Huolestuttavaksi testauksen eteneminen muodostuu silloin kuin kriittiseksi määriteltyjen virheiden määrä kasvaa testauksen edetessä. Ennen johtopäätösten tekoa on tehdyt virheilmoitukset analysoitava ja tarkistettava oikea virheluokitus. Kriittisen virheen määritelmä voi olla erilainen eri testaajille, ja on varmistuttava siitä että arviot virheen vakavuudesta on oikeita ennen johtopäätösten tekemistä. Testausprojektissa virheiden kriittisyyden arvion tekee viimekädessä testauspäällikkö keskustellen virhekirjauksen tehneen testaajan kanssa ennen virheraportin analysointia.

Esimerkki kirjattujen virheiden määrästä jaoteltuna virheen vakavuusluokituksen perusteella testattavan sovellusalueen mukaisesti.



Kuva 13: Virheet vakavuusluokittain

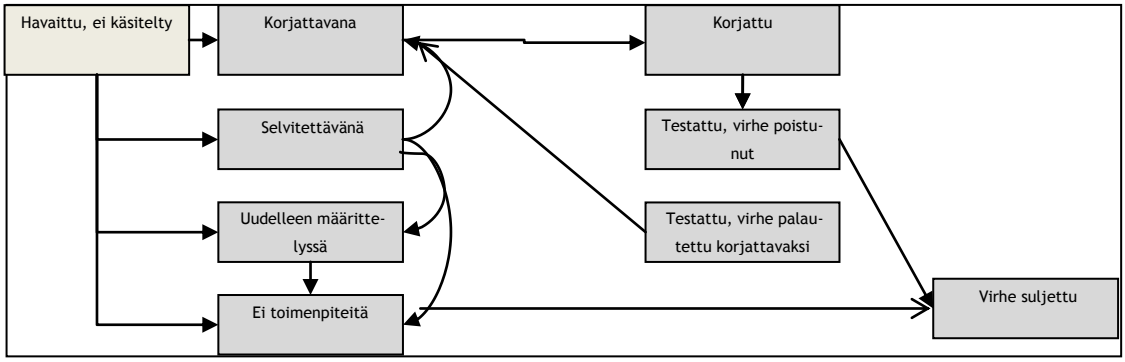
5.2.2 Testauksen virhemetriikka 1. Havaittujen virheiden analysointi

Analysoitaessa kirjattujen virheiden määrää ja niiden jakoa eri vakavuusasteiden mukaan on erityistä huomiota kiinnitettävä virheiden esiintymistiheyteen. Jos vakavien virheiden määrä kasvaa tai edes pysyy ennallaan testauksen edetessä, on syyt selvitettävä mahdollisimman nopeasti ja selvitettävä huonokuntoisen koodin laatu. Toteuttajan työn ja aikaansaadun koodin laadussa ei välttämättä ole mitään vikaa, mutta virheellisen toteutuksen perussyynä voi olla määrittelyjen epätarkkuus, josta johtuen toteutettu toiminnallisuus ei ole halutun laista. Virheen juurisyy voi johtaa aiempaan sovelluskehitykseen asti ja laatua saadaan parannettua jo prosessin alkuvaiheista lähtien kiinnittämällä huomiota parempaan määrittelyn laatuun (Kaner. 2002, 61).

Virheiden aiheuttajana voi olla myös toteutettavan sovelluksen kompleksisuus ja saattaa olla että toteuttajalta ei löydy riittävää osaamista tehtävään. Yhdistämällä sovelluskehityksen henkilöiden osaamista tai hankkimalla osaamista työryhmän ulkopuolelta pyritään ratkaisemaan tietotaidon varmistaminen ja siten laadukkaampi työn jälki. Virheiden analysoinnissa on pystyttävä riittävän avoimeen keskusteluun niiden juurisyistä ja pystyä välttämään vastakkainasettelua ja syyttävää asennetta virheen selvittelyssä, jotta saadaan selvitettyä virheiden syyt ja mahdollisimman pikaisesti ja oikeilla toimenpiteillä korjata niitä aiheuttavat olosuhteet (Black. 2004, 548, 2004)

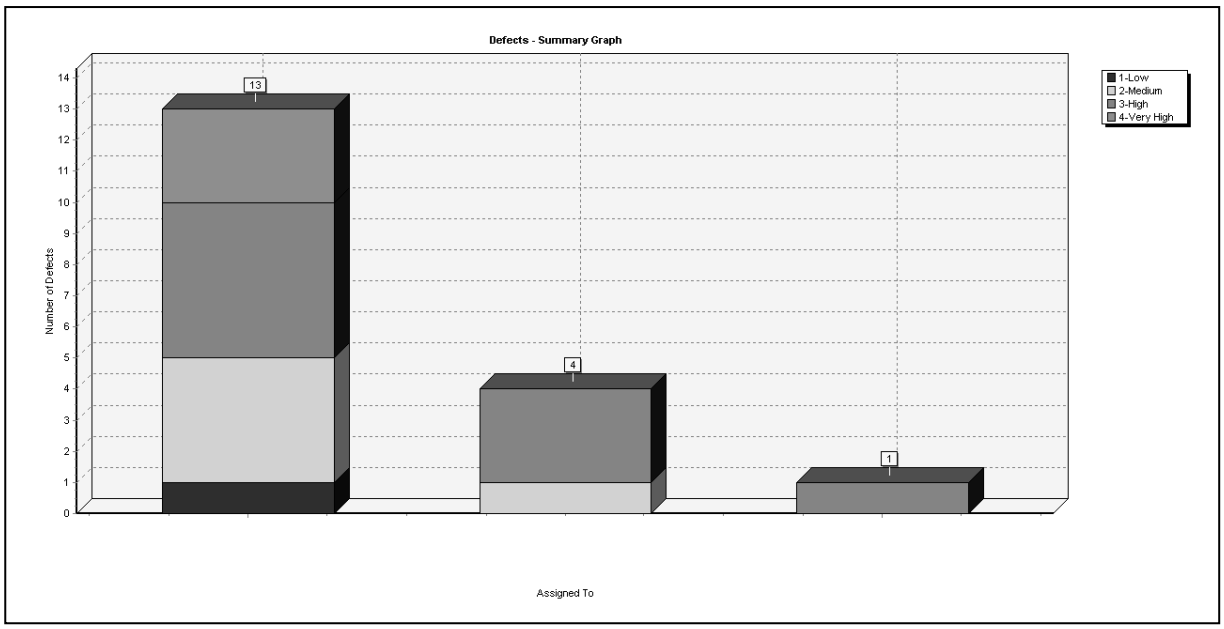
5.2.3 Testauksen virhemetriikka 2. Virheiden tilanmuutosten historiatrendi

Virhekirjauksen edetessä ja sitä eri käsittelijöille siirrettäessä virheen tila ja käsittelijän nimi muutetaan, jotta nähdään missä vaiheessa virheenkirjausprosessia virhe on ja kenellä se on viimeksi käsiteltävänä (Kuva 14). Uutta virhettä kirjattaessa virheen tilaksi tulee "Havaittu" ja jatkossa virhekirjausprosessi voi olla esim. seuraavanlainen:



Kuva 14: Virhekäsittelyprosessi ja tilamuutokset

Virheen tilaa muutetaan aina ennen seuraavalle käsittelijälle siirtämistä ja tilamuutosten avulla pystytään seuraamaan virheen käsittelyn etenemistä prosessin aikana (Kuva 15). Tilakoodin lisäksi virhe osoitetaan nimetyllä henkilölle (Assigned To), jolta on mahdollista saada lisäselvityksiä korjauksen etenemisestä.



Kuva 15: Eritilaisia virheitä osoitettuna käsittelijöille

5.2.4 Testauksen virhemetriikka 2. Virheiden tilamuutosten analysointi

Virheiden tilan muutosten historiatrendiä seuraamalla saadaan tietoa virhekorjauksiin käytetyn ajan kulumisesta ja samalla seurata virhekorjausprosessin tehokkuutta. Tehokkuuden mittaamisen lisäksi saadaan myös työnjohdollista tietoa eri virheiden korjauksen vastuuhenkilöille kertyvien korjattavien virheiden määrästä ja korjausaikataulusta. Helposti päästään tilanteeseen, jossa suurin osa virhekorjauksista kasautuu keskitetysti harvoille henkilöille ja työkuorman tasaamiseksi seuraamalla virhekertymää saadaan vastuita jaoteltua tasapuolisemmin

Virheen vakavuuden perusteella raportoidun virhejakauman juurisyyn selvittämisellä on mahdollisuuksia parantaa koko tuotekehityksen prosessia ja mahdollistaa laadun parantamista kautta koko kehityslinjan. Mittarin antamaan tietoon on suhtauduttava analyyttisesti, jotta sen tarjoamaa tietoa saadaan parhaiten hyödynnettyä. Parhaimmillaan analyysin tuloksena on mahdollista saada kiinni laajempia kehitysprosessin pullonkauloja tai kehityksen esteitä pelkän kliinisen numerotiedon lisäksi.

5.3 Testausprosessin metriikat

Heti kun testaussuunnittelu saadaan riittävän yksityiskohtaiselle tasolle, pystytään kokonaistestaus jakamaan pienempiin osiin: toimituserät. Kukin toimituserä muodostaa järkevän toimituskokonaisuuden, joka asennetaan testattavaksi sovitussa aikataulussa. Yleisimmin uusia toiminnallisuuksia tuodaan testattavaksi kerran viikossa, jos aikataulu niin vaati, voi rytmi olla tiheämpi.

Toimituseriin jaettaessa testaus on paremmin hallittaessa, kun pystytään osoittamaan pienempien kokonaisuuksien laatu toimituserä kerrallaan. Lisäksi toimituseriin jakaminen rytmittää testausta ja vie sitä kohti suurempia kokonaisuuksia. Lisäksi virheiden hallinta ja kohdistaminen helpottuu, kun pystytään jäljittämään, mihin testauskokonaisuuteen virheilmoitus kohdistuu.

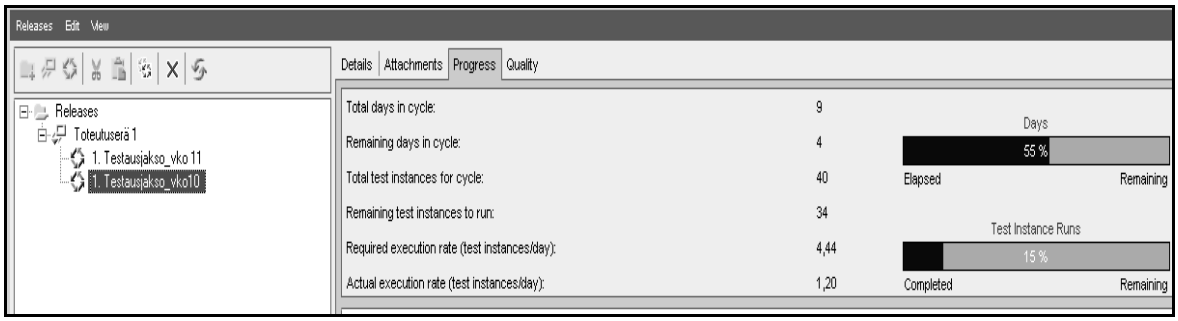
Jokaisesta toimituserästä julkaistaan toimituspaketti, joka sisältää kuvauksen testaukseen siirrettävästä toiminnallisuudesta, mahdollisista rajoituksista sekä luettelee virhekorjaukset, jota saadaan muutostietona tähän toimituserään.

5.3.1 Testausprosessin metriikat 1. Testausprosessin eteneminen

Jokaiselle testausjaksolle määritellään testaussuunnitelman mukaan niiden suorittamisen aikataulu ja kirjataan se Release – osioon tallennetun testausvaiheen osalle. Yksittäisessä jak-

sossa suunnitellut testitapaukset kiinnitetään linkittämällä oikeaan Testausjaksoon, esimerkissä Testausjakso 1_10 / Testausjakso 1_vko 11 / Testausjakso 2_vkot 12–13 (Kuva 16). Kun kaikki alkuarvot on määritelty ja kullekin testausjaksolle kiinnitetty siihen liittyvä testauksen sisältö, voidaan raportilta seurata kunkin testausjakson etenemisestä yhdeltä näkymältä.

Raporttisivun antaa tietoa testauksen etenemisestä, siihen käytetystä ajasta, jäljellä oleva testausaika ja jäljellä olevien testattavien asioiden määrä. Lisäksi välilehdeltä saadaan testauksen aikana kirjattujen virheiden määrä niiden tilaluokittelun mukaisesti testausjakson mukaan jaoteltuna.



Kuva 16: Testauksen eteneminen

5.3.2 Testausprosessin metriikat 1. Testausprosessin analysointi

Raporttia ei tarvitse tulostaa paperilla vaan näkymästä saadaan selville kertavilkaisulla testauksen valmiusaste sekä jäljellä oleva testattava määrä. Ylimmän kulutetun ajan palkin perusteella selviää, että käytettävissä olevasta ajasta on käytetty noin puolet (=55 %) ja testauksen valmiusaste on 15 %. Lisäksi saadaan selville, montako testiä on suoritettava jokaisena testausjaksosta jäljellä olevana päivänä (= 4,44 kpl). Tästä saadaan arviota siitä, saadaanko testausjakso suoritettua käytettävissä olevassa ajassa vai tarvitaanko mahdollisesti lisäresursseja. Raportilta ei saada selville, kuinka monta testaajaa on ollut suorittamassa testausta. Testaussuunnitelman mukaan testaukseen on kiinnitetty yksi päätoiminen testaaja, joten tavoitteena aikataulussa pysymiseksi on suorittaa vähintään 4 testiä päivässä ja tiedossa saattaa olla aikatauluongelma.

Arvioidaan kohtaan Actual execution rate, josta saadaan selville käytetyssä ajassa suoritettujen testien määrä verrattuna käytettyyn aikaan: tuloksena on 1,2 testiä.

Vielä ei voida vetää suoria johtopäätöksiä siitä, onko testaukselle aikatauluongelma, koska on ensin selvitettävä, minkä tyyppisiä ovat olleet jo suoritetut testit. Usein testaajalla on tavoitteena aloittaa testaus vaikeimmista ja monimutkaisimmista tapauksista, jotka vievät myös enemmän aikaa. Usein myös nämä vaikeimmat testit on priorisoitu ensimmäisiksi tehtäviksi.

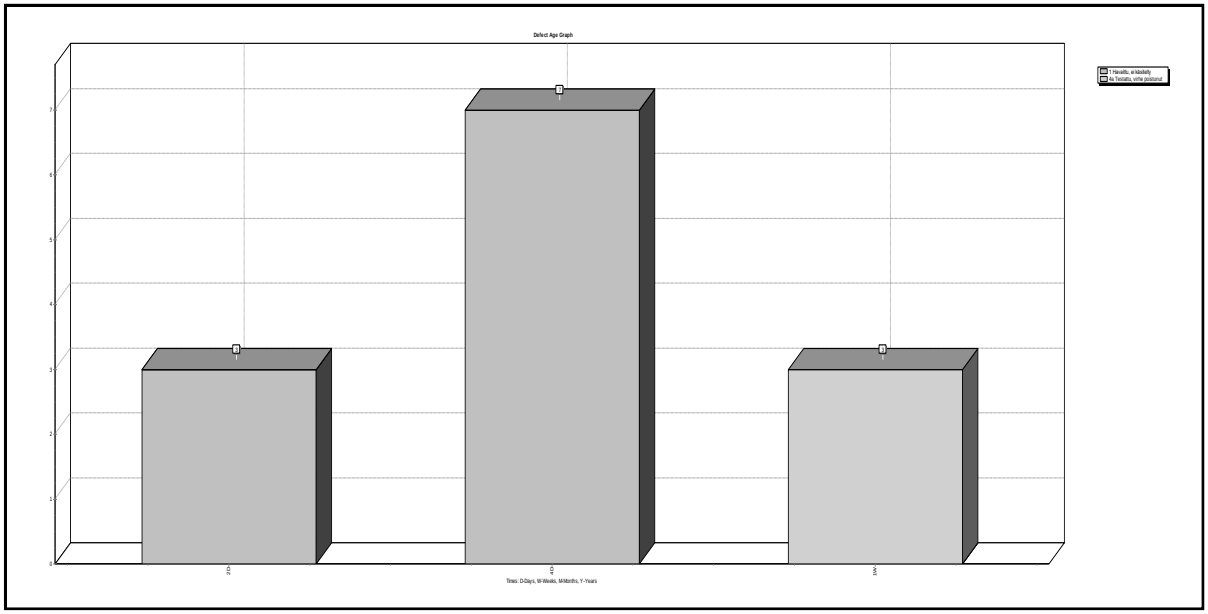
Haastattelemalla testaajaa selvitetään testien haasteellisuus ja samalla verrattavuus raportin antamiin lukuihin. Testaaja kertoo edellä mainittujen seikkojen pitävän paikkaansa, eli tavoitteena on ollut saada hankalimmat tapaukset ensin alta pois ja siksi niihin on kulutettu myös enemmän aikaa.

Suoran raportin antaman tiedon perusteella olisi voinut olettaa, että testaajalla on ongelmia ajankäytön kanssa ja testausta ei saada valmiiksi suunnitellussa aikataulussa. Tarkemman analyysin perusteella voidaan kuitenkin todeta, että toistaiseksi ei ole ongelmaa, koska tulevat suoritettava testit saadaan läpäistyä nopeammin niiden yksinkertaisemman rakenteen takia, ja testaajalla on hyvä mahdollisuus pysyä vaaditussa runsaan neljän testin päivävauhdissa.

5.3.3 Testausprosessin metriikat 2. Vikojen keskimääräinen ikä

Vikojen keskimääräinen ikä saadaan raportille vertaamalla virhekirjauksen tekopäivää päivämäärään, jolloin virhe otetaan käsittelyyn ja sen tilaa muutetaan (Kuva 17).

Testauksen etenemisen aikana saadaan eri tilassa olevista virheistä raportti, jossa seurataan niiden tilaa ja aikaa verrattuna kirjausaikaan. Virheiden ikä on jaoteltu luokkiin 1 päivä / 4 päivää / 1 viikko, jne.



Kuva 17: Virheiden ikäjakauma

Raportilla on nähtävissä testauksen aikana tehdyn virhehavainnon ikä sekä virhekorjaukseen ja uudelleen testaukseen kuluva aika. Virheet tilassa Havaittu, ei käsitelty ovat sellaisia, jotka testaaja kirjaa havaittuaan virheen. Tilassa Testattu, virhe poistunut olevat virheilmoi-

tukset ovat sovelluksen kehittäjän korjaamia ja ne on palautettu uudelleen testaukseen. Kun testaaja toteaa toistamalla testitapauksen virheen poistuneen, muutetaan virheen tilaa. Yllä olevassa tapauksessa koko virhekirjauksen sykli on yksi viikko, jota voidaan pitää hieman huolestuttavana, koska koko testausjakso on vain neljän viikon mittainen. Tilanteen selvittäminen vaatii lisätietoa, onko mahdollisesti kyse siitä, että virheen tila on jäänyt muuttamatta vai onko tilanteen taustalla muita syitä.

5.3.4 Testausprosessin metriikat 2. Vikojen keskimääräisen iän analysointi

Virheen korjaussyklin piteneminen voi johtua virheselvittelyn monimutkaisuudesta, jolloin on analysoitava virheen taustalla olevaa toiminnallisuutta. Onko se määritelty niin hyvin, että kehittäjän on ollut mahdollista toteuttaa toiminnallisuus mahdollisimman oikein ja eri riippuvuudet huomioiden. Tässä tapauksessa käydään tilannetta läpi sekä kehittäjän että määrittelijän kanssa ja pyritään saamaan toteutuksen määrittelyt ja niiden ongelmista käytävät selvittelyt mahdollisimman sujuviksi (Kaner. 2002, 88)

Toinen virhekorjauksen pitkittymistä aiheuttava syy voi olla kehittäjän työtilanne. Usein erityisosaaminen saattaa olla vain harvojen käsissä ja näitä osaajia tarvitaan yhtäaikaisesti useassa samaan aikaan meneillään olevassa projektissa. Resurssipulaa pystytään helpottamaan keskustelemalla kehittäjien esimiesten kanssa vastuiden jakamisesta useamman henkilön kesken. Lisäksi selviteltävänä on tekeillä olevien töiden ja virhekorjausten priorisoinnin läpikäynti.

Virhekorjausten priorisointi on havaintoa kirjattaessa tärkeä, koska ne otetaan työn alle tehdyn prioriteettiluokan mukaan. Siksi alemman prioriteetin virheissä korjausaikatauluna viikko on täysin hyväksyttävä. Tutkimalla tarkemmin virhekirjausten tilaa viikon kestävissä virhekorjauksissa päädytään tulokseen jossa kolmesta virheestä yksi on Medium-tasoa ja kaksi on Low tasoa (Kuva 18).

	1-Low	2-Medium	3-High	<total>
2D			3	3
4D			7	7
1W	2	1		3
<total>	2	1	10	13

Kuva 18: Virheiden korjausaika

5.4 Testauksen projektimetriikat

Testauksen projektimetriikoilla voidaan arvioida testauksen edistymistä verrattuna suunniteltuun aikaa, työmäärään tai suunniteltuihin kustannuksiin. Näitä mittareita ovat mm:

- Kustannusarvio vs. toteutunut kustannus
- Aikataulu vs. toteutunut aikataulu
- Työmääräarvio vs. toteutunut työmäärä
- Toteutuneet vs. arvioidut työmäärät vaiheittain

Yleisimmin aikataulu, resurssisuunnittelu ja työmäärien suunnittelu tehdään MS Project-työkalulla ja suunnitelma on osana projektisuunnitelmaa. Tehty kustannus ja resursointisuunnittelu muodostavat tärkeän kriteerin projektin kustannus-hyöty – arvioinnille ja siten muodostaan projektin aloituskriteerin ja koko sen keston ajan arvioitavan baselinen. Kokonaisuudessaan projekti muodostuu eri tehtävistä, kuten suunnittelu, määrittely, toteutus ja testaus. Testauksen osuutta seurataan tarkemmin, jos kyseessä on organisaation ulkopuolelta hankittujen testausresurssien kulujen seuranta. Myös oman testausorganisaation tehokkuutta on hyvä aika ajoin tarkastella kriittisesti.

Loppulaskun kehitettävästä järjestelmästä maksaa liiketoiminta ja se myös haluaa seurata sijoittamansa rahan hyötysuhdetta kehityksen eri vaiheissa. Tarkkana on myös oltava sen suhteen, että tehdään järkeviä ratkaisuja siinä vaiheessa kun aikataulu tai budjetti alkaa venyä. Näissä tilanteissa tehtävät ratkaisut on helpompi tehdä, kun päätöksen pohjana on tietoa ja niiden pohjalta mahdollisesti jo tehtyjä ratkaisuvaihtoehtoja. (Kaner. 2002, 94 – 95)

Koska suurin tekijä kehitysprojekteissa on henkilötyöstä muodostuva kustannus, on toteutuneita henkilötyökustannuksia helpointa seurata tehtyjen tuntikirjausten perusteella. Organisaatiossa tuntikirjausten työvälineellä on mahdollista eri hakutekijöillä saada selville halutun projektin kirjatut tunnit ja sitä kautta toteutuneet henkilötyökustannukset.

Testauksen projektimetriikat on rajattu tutkimuksen ulkopuolelle, koska niiden analysointi vaatisi toisen työkalun esittelyn ja perehtymisen. Lisäksi tämän metriikan antamat tulokset eivät suoranaisesti vaikuta laatuun.

6 Tutkimuksen kulku

Tutkimuksen edistymisen haasteena oli aiheen riittävän rajauksen löytäminen. Alusta asti tutkimuksen kohteeksi päädyin ottamaan testauksen laadun ja määrittelyn sen tutkimiselle. Lo-

pullinen rajaus testauksen laadun analysointiin mittaristojen kautta syntyi työpaikalla pidetyn testauksen mittaristojen laatua käsittelevän seminaarin yhteydessä, jolloin mielenkiinto konkreettiseen laadun mittaukseen syntyi. Testausprojektien läpiviennin yhteydessä mietintään jäänyt ajatus mittareiden käyttämisestä ja niiden antaman tiedon analysoinnista kyti edelleen. Tutkimuksen pohjaksi muodostuva aineisto syntyi testauksen läpiviennin yhteydessä automaattisesti ja osana jokapäiväistä työtäni oli sen raportoiminen ja analysoiminen sidosryhmille päätöksentekoa varten. Kirjallisuuden löytäminen testauksen laatumäärittelyistä osoittautui haastavaksi, mutta laadusta ja sen standardeista tietoa oli mahdollista saada. Laadun määrittelyt varsinkin testausten tulosten osalta ovat usein organisaation itse määrittelemiä, mutta jokaisen laatumäärittelyn taustana on tavalla tai toisella olemassa oleva laatumäärittelyt ja tavoitteena mahdollisimman tarkoituksenmukainen tuote tai palvelu loppukäyttäjälle.

Kokonaisuudessaan prosessi ensimmäisen idean synnystä tutkimuksen aloittamiseen kesti noin puoli vuotta, jonka jälkeen kirjallisuustutkimuksen jälkeen teoriaa tutustumisen yhteydessä opinnäytetyö alkoi muotoutumaan lopulliseen rakenteeseensa. Tutkimuksen pohjaksi kerääntyi tietoa automaattisesti testausprojektien edetessä, joten suurimmaksi työksi jäi edustavimman otoksen kerääminen tietomassasta.

Tutkimuksen tavoitteena oli löytää parhaimmat mittaristot testauksen laadun osoittamiseen. Käytettävällä testausvälineellä on mahdollista generoida sille tallennetusta tiedosta käyttämällä eri poimintakriteereitä lukematon määrä erilaisia raportteja ja graafeja, joten päätin rajata tutkittavien mittareiden määrää. Organisaation testausstrategiassa määritellyistä jokaisesta laatumetriikanlajista valitsin 2 – 3 mittaria, joita käytin kerätyn tutkimusaineiston analysointiin. Kolme valituista mittareita oli jo käytössä projektin testauksen raportointiin, mutta halusin tarkastella uusia tapoja analysoida testauksen etenemistä ja sen laadunvarmistusta.

Lopputuloksena analysoinnin perusteella valitut mittarit ja niiden antama tieto perustuu täysin omaan mielipiteeseeni niiden antaman mahdollisimman luotettavan laatuanalyysin takia. Päädyin valitsemaan lopullisiksi mittareiksi kolme mahdollisimman luotettavaa mittaria, joiden antama tieto on kiistattomasti analysoitavissa ja joiden tieto perustuu mahdollisimman luotettavaan tietoon.

6.1 Aineiston kuvaus ja aineiston kerääminen

Aineiston keräämiseen pyrin saamaan mukaan mahdollisimman monipuolisesti sekä ketterällä että vesiputousmallilla toteutettavia kehitysprojekteja. Tutkimuksessa käytettävistä aineiston

ominaisuuksista Likitalo mainitsee kirjassaan Tutkimusmenetelmät mm. luotettavuus, kattavuus, laajuus, tuoreus ja tarkoituksenmukaisuus.

Tutkittava aineisto syntyi oman testaukseni sekä testaustiimini suorittaman testauksen perusteella projektin edetessä. Aineisto kerättiin kehityshankkeiden järjestelmätestausvaiheesta, jolla testausstrategian mukaisesti suunnitellaan ja suoritetaan käytettävällä testausvälineellä. Aineiston muodostamiseen osallistui myös analysoitavien testausprojektien järjestelmätestaajat testausorganisaatiosta.

Testauksen etenemisen aikana otin valittujen mittareiden vaatimat raportit testausvälineeltä ja pyrin mahdollisimman puolueettomasti analysoida niiden antamaa tietoa. Analysoitavat mittarit on avattu ja niiden antama tieto analysoitu tutkimuksen kohdassa 5.1. – 5.4. Samalla kun analysoin mittareiden antamaa tietoa, valikoituvat lopulliset ehdokkaat tutkimuksen tavoitteena oleviksi mittareiksi.

6.2 Tutkimuksen aikana kohdatut haasteet

Haasteellisinta tutkimuksen sisällön määrittelyssä oli riittävän rajauksen tekeminen laatukäsitteestä. Laadusta sinänsä on kirjoitettu useitakin teoksia ja standardeja, mutta halusin tutkia nimenomaan testauksen laatua ja sen varmistamiseen liittyviä menetelmiä. Testauksen laadunvarmistuksesta ei varsinaisesti ollut löydettävissä lähdekirjallisuutta, mutta tietoa löytyi eri testauksen teoriasta kirjoitetuista teoksista ja soveltamalla laatumäärittelyjä testauksen maailmaan. Testauksen teoriakirjallisuus ei ole puhtaimmillaan noudatettuna tutkimuksessa käytetyssä organisaatiossa, joka on testauksen laatuteorioiden pohjalta kirjoittanut oman testausstrategiansa, ja johon on määritelty omat laadun tarkistuspisteet ja menettelytavat.

Tutkimuksen tekemisen aikana käynnissä olevista projekteista oli mahdollista saada paljon dataa, joten tämän käsiteltävän datan määrä oli sekä tutkimusta edistävä asia, mutta myös haaste. Päädyin rajaamaan analysoitavan tiedon viiteen erityyppiseen projektiin sen perusteella, että niiden käyttöönotto tapahtui samaan aikaan, joten edistymisen seuranta oli mahdollisen yhteismitallista.

7 Tutkimuksen tulosten arviointi

Tutkimukseni tavoitteena oli arvioida testauksen laadun määrittelyä ja arviointia seurattavaksi määriteltyjen mittareiden avulla Tutkimuksen pohjana oli keväällä 2013 käynnissä olleiden kohdeyrityksessä kehitettävien järjestelmien testauksen seuranta järjestelmätestauksen

osalta. Tutkimukselle asetettu tavoite oli määritellä luotettavimmat mittarit varmistamaan järjestelmätestauksen tuloksen laatua.

Tutkimuksen aikana pyrittiin saamaan selville kolme perusmittaria, joilla mahdollisimman luotettavasti pystytään arvioimaan järjestelmätestauksen aikana kehitettävän palvelun tai toiminnallisuuden laatua ja sen kypsyyttä hyväksymiselle tuotantokäyttöön.

Käynnissä olleiden projektin testauksen tuloksia peilattiin määriteltyjen mittareiden antamaan tietoon ja analyysin pohjalta päädyin valitsemaan tutkimuksen tuloksena valituiksi mittareiksi:

- Mittari 1. Testauksen laatumetriikka 1: Vaatimusten kattavuus
- Mittari 2. Testausprosessin metriikat 1: Testauksen eteneminen
- Mittari 3. Testauksen virhemetriikka 1: Kirjatut virheet vakavuusluokittain

7.1 Perustelut valitulle mittarille: Mittari 1

Vaatimusten kattavuus on koko testaussuunnittelun laajuuden arvioimisen perusta. Sen kautta on määriteltävissä, onko testaussuunnittelu riittävän laaja ja siten se muodostaa tärkeimmän pohjan koko testausprojektille. Riittävällä testaussuunnittelulla katetut vaatimukset antavat tuotteen tai palvelun tilaajalle tiedon siitä, että halutut toiminnallisuudet on testauksella varmistettu ja näiden määriteltyjen vaatimusten testauksen avulla todennettu laatu on tiedossa.

Mittarin antama tieto on yksiselitteistä: joko vaatimus on katettu testaussuunnittelulla tai ei ole katettu. Varsinainen mittari on nopeasti muodostettavissa ja sen antama tieto päivittyy reaaliaikaisesti, joten edistymisen raportointi projektiryhmälle on nopeata. Jos raportti paljastaa vaatimuksia, joille testaussuunnittelua ei ole tehty tai ei ole voitu tehdä, on analysoitava onko vaatimus tarpeellinen. Jos todetaan, että vaatimukselle ei voida suunnitella testejä, on se poistettava vaatimusluettelosta ja muutoshallintaan tallennettava tieto, miksi vaatimus on poistettu.

Lisäksi mittari antaa selkeää tietoa siitä, miten priorisoitujen vaatimusten testaussuunnittelu etenee. Suunnittelu on aina aloitettava korkeamman prioriteetin testitapauksista, ja poikkeamat priorisoinnista on havaittavissa myös reaaliaikaisesti, joten siihen on mahdollista puuttua aikaisessa vaiheessa, jotta testaussuunnittelu etenee keskittyen tärkeimpien toiminnallisuuksien varmistamiseen.

7.2 Perustelut valitulle mittarille: Mittari 2

Testauksen eteneminen on perustava mittari kehitysprojektin etenemisen kannalta. Projektille asetetuista tavoitteista määritelty aikataulu on kustannusseurannan kanssa yksi tärkeimmistä. Projektin osa-alueista testaus sijoittuu kehityskaaren loppuvaiheeseen ja sen aikataulun pidentyminen suunnitellusta on usein kriittisen tarkastelun kohteena. Testauksen läpivienti sitoo resursseja testaajien lisäksi myös kehittäjistä, koska heidän on varattava aikaa mahdollisesti löydettyjen virheiden korjaamiseen. Aikataulun venyessä resurssien vapautuminen uusiin tehtäviin vaarantuu ja saattaa syntyä resurssivarauksissa ristiriitatilanteita.

Etenemisen seuranta on tärkeätä myös määriteltäessä ne toiminnallisuudet, joissa testaus ei etene. Riskinä on mahdollisuus siihen, että nämä toiminnallisuudet ovat puutteellisten tai jopa virheellisten määrittelyjen perusteella toteutettu väärin ja eivät toimi kuten on haluttu. Pikaisesti puuttumalla niihin kohteisiin, joissa testaus ei syystä tai toisesta etene, auttaa aikataulussa pysymistä.

Raportin antamat tiedot muodostuvat vain suoritetusta testistä, joten raportoitu tieto on ajantasaista ja muodostuu todellisten suoritettujen testien perusteella. Mittarin tulokset ovat graafisena palkkina helposti yhdellä silmäyksellä todennettavissa, koska raporttinäkymältä saadaan tieto testauksen valmiusasteesta prosentteina, jäljellä olevien testien määrä sekä tieto montako testiä on suoritettava jäljelle olevina päiviä, jotta suunnitellussa aikataulussa pysytään. Raportin tietoja voidaan käyttää päivittäisten tai viikottaisten testaajapalaverien tilanneseurannassa, jossa etenemistä estävät tekijät voidaan selvittää. Samalla on mahdollista jakaa testaajaresursseja uudelleen tai suunnitella testauksen suorittamista raportin antamien faktojen perusteella.

7.3 Perustelut valitulle mittarille: Mittari 3

Kolmas valittu mittari perinteisimmin kertoo testattavan kehitystyön laadusta, koska sen antama tieto kohdistuu kirjattuihin virheilmoituksiin. Havaittujen virheiden määrä kertoo kehitettävän tuotteen tai palvelun kypsyydestä ja sen laadun tasosta. Virhekirjaukset syntyvät testitapausten suorittamisen yhteydessä, kun verrataan määriteltyä toiminnallisuutta testauksen lopputulokseen ja havaitaan poikkeamia halutusta.

Virheen havaitseminen on kiistaton tosiasia ja niiden kasautuminen tiettyyn toiminnallisuuteen antaa viitteitä siitä, että toteutuksessa on selkeitä eroavaisuuksia verrattuna haluttuihin ominaisuuksiin. Virheen vakavuusluokittelu on ainoa subjektiivinen asia virhekirjauksessa, josta voidaan keskustella projektin sisällä yhdessä ja päätyä eri tulokseen, kuin minkä testaaja on arvioinut. Testauksen etenemisen estävä critical-tasoinen virhe on tästä selvä poikkeus:

testitapauksen suorittamisessa ei päästä eteenpäin, on virhe sitä luokkaa, että sen vakavuusaste tai priorisoinnissa ei ole epäselvyyttä.

Kun kirjatut virheilmoitukset luokitellaan toiminnoittain, on yhdeltä raportilta nähtävissä eri toteutettujen toimintojen laadulliset erot ja on mahdollista lisätä panostusta virhealttiisiin toimintoihin joko osaamista tai resursseja lisäämällä. Pahimmassa tapauksessa on koko testaus keskeytettävä, jos raportti näyttää joka osa-alueelle kasautuvaa virhemäärää.

8 Jatkokehitysehdotukset

Koko tutkittavan organisaation testauksen tehostamisessa huomiota on kiinnitetty mitattavan tiedon luotettavuuteen ja niiden mittareiden määrittelemiseen, joilla mahdollisimman puolueettomasti voidaan varmistaa laatua testauksen aikana syntyvän datan perusteella. Mittaristosta on tarkoitus tehdä mahdollisimman yhteismitallinen käynnissä olevien projektien testauksen laadun mittaamiseksi. Toisena vaatimuksena on se, että mittareiden raportointiin ei kuluisi kohtuuttoman paljon aikaa, vaan niiden muodostaminen olisi mahdollista valmiiden mallipohjien avulla. Mallipohjia käyttämällä jokainen projekti raportoi mittarien antamat tiedot samanmuotoisena.

Käytettävät mittarit tallennetaan omaksi työpöytänäkymäkseen, josta vakiomittareiden raportit ovat ajettavissa ja josta myös voidaan omat kriteerit antamalla raportoida testauksen tuloksia myös muilla. Tällaisia valmiita työpöytänäkymiä on mahdollista tallentaa käytettävällä testausvälineelle, mutta valintana voi olla myös ostettava ratkaisu, jossa testausvälineen päälle integroidaan käyttöliittymä, jossa vakioraportteja voidaan hallita.

Tämän tutkimusraportin kirjoitushetkellä lopullista ratkaisua käytettävistä mittareista ei ole vielä tehty, mutta tutkimukseni pohjalta päädyn ehdottamaan valitsemiani mittareita tekemieni analyysien pohjalta. Lopullinen päätös valinnasta saattaa olla erilainen kuin tutkimukseni perusteella, mutta joka tapauksessa omien testausprojektieni osalta tulen käyttämään valitsemiani mittareita joko vakiomittariston lisäksi tai niiden joukossa.

Lähteet

Agile alliance and Institute Agile. 2011 [WWW-dokumentti].
<http://guide.agilealliance.org/guide/ready.html> (Luettu 16.8.2013).

Black, P., 2004. Critical testing processes: plan, prepare, and perform, perfect. Boston (Mass.): Addison-Wesley.

Dhaval, Panchal. 2008 [WWW-dokumentti].
<http://www.scrumalliance.org/community/articles/2008/september/what-is-definition-of-done-(dod)> (Luettu 16.8.2013).

http://fi.wikipedia.org/wiki/Metriikka (Luettu 4.11.2013)

Gardner, Brian. 2009. Testingminded [WWW-dokumentti].
<www.testingminded.com/2009/01/what-is-moscow-principle.html (Luettu 1.9.2013).

Haikala, Ilkka. & Märijärvi, Jukka. 1998. Ohjelmistotuotanto. Espoo. Suomen atk-kustannus.

Hutcheson, Mariel. 2003. Software testing fundamentals: methods and metrics Indianapolis: Wiley.

Kaner, Cem, Bach, James. & Pettichord, Bret. Lessons Learned in software testing. 2002. Wiley Computer Pubishing.

Kooman, Tim., van der Aaist, Leo, Broekman, Bart. & Vroon, Michiel. 2006. TMap Next for result-driven testing. Nederlands. UTN Publishers

Likitalo, Heikki. & Rissanen, Riitta. 1998. Tutkimusmenetelmät. Menetelmätietoutta tra-denomiopiske-lijoille. Opetusmoniste. Pohjois-Savon ammattikorkeakoulun julkaisut D 8/1998. Helsinki. Pohjois-Savon ammattikorkeakoulu.

McGregor, John & Sykes, David, 2001. A practical guide to testing object-oriented software. Pearson Education.

Pohjonen, Risto. 2002. Tietojärjestelmien kehittäminen. Jyväskylä: Docendo.

Sogeti. 2009. Maximizing the value of good testing practice in an Agile environment [WWW-dokumentti].
<www.uk.sogeti.com/en/Resources--Downloads/Case-Studies-Brochures-and-White-papers/whitepapers/Maximising-the-Value-of-Good-Testing-Practice-in-an-Agile-Environment> (Luettu 8.10.2013)

Vuori, Matti. 2001. Organisaatiokulttuurin ymmärtäminen laatutyön menestystekijänä [WWW-dokumentti].
http://www.mattivuori.net/julkaisuluettelo/liitteet/organisaatiokulttuurista_laatutoimijalle.pdf (Luettu 24.7.2013).

Wells, Don. Extreme Programming [WWW-dokumentti].
<http://www.extremeprogramming.org/> (Luettu 8.10.2013).

Wiegers Karl. 2003. Software Requirements. Second edition. Redmond: Microsoft Press.

Yin, Robert. 2009. Case study research. Design and methods. Fourth edition. Sage Publications Inc.

Yritys X: n testausstrategia, 2011.

Kuvat