LAUREA

Mainline-testauksen parantamisehdotus Process Visionille

Pettersson, Jarno

2009 Leppävaara

Laurea-ammattikorkeakoulu
Laurea Leppävaara

Mainline-testauksen parantamisehdotus Process Visionille

Jarno Pettersson
Tietojenkäsittely koulutusohjelma
Opinnäytetyö
Toukokuu, 2009

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

Jarno Pettersson

Mainline-testauksen parantamisehdotus Process Visionille

Vuosi 2009 Sivumäärä 46

Opinnäytetyön aihe tuli kohdeyritys Process Visionin taholta. Kohdeyritys Process Vision valmistaa ohjelmistoja energia-alan yrityksille. Tehtävänanto sisältää Mainline-testaustiimin toiminnan parannusehdotuksia. Tarkoituksena on löytää tapoja, joilla testaustiimin toimintaa voidaan tehostaa ja parantaa.

Opinnäytetyö on toimintakeskeinen tutkimus ja tutkimusmenetelmänä on kvalitatiivinen tutkimusmenetelmä.

Ohjelmistotestaus on kehittynyt huomattavasti viime vuosina, että mitään ohjelmia ei voida markkinoille tuoda ilman testausta. On olemassa erikokoisia yrityksestä riippuen muutaman henkilön tai kokonaisen tiimin kokoisia testaustiimejä. On myös lukuisia eri tapoja testata ohjelmaa, mutta yleensä ohjelmaa testataan tietyn kaavan mukaan erilaisilla testitapauksilla. Testitapaukset voivat testata ohjelman tiettyä toimintoa, erilaisten pienten toimintojen toimivuutta keskenään tai koko ohjelman toimivuutta.

Mainline tarkoittaa Generis-tuotteen pääversiohaaraa, joka sisältää kaiken viimeisimmän tuotekehityksen ja virhekorjaukset. Mainline-versiohaarasta julkistetaan kuukausittain pääversio, jonka testauksesta vastaa erillinen testaustiimi. Testaustiimillä on käytössään tarkoitukseen varattu testitietokanta ja testipalvelimet, joille tiimi päivittää uusimman version.

Yrityksen tuote on Generis, jota käyttävät eri energia-alan yritykset. Ohjelma on erilaisten mittaus-, lukema- ja laskutustietojen hallintaa varten soveltuva sovellus. Ohjelma on suunniteltu sähkö-, kaasu-, kaukolämpö- ja vesimittauksiin. Tietoa voi syöttää ohjelmaan sekä automaattisesti että manuaalisesti.

Parannusehdotuksia varten tehtiin tutkimuksia ja haastatteluita kohde yrityksessä. Kerättyjen tietojen perusteella suunniteltiin erilaisia parannusehdotuksia. Ensimmäisenä on testitapausten automatisointi, joka on tällä hetkellä ollut vähäistä testaajien aikataulun ja AutoTester ohjelman takia. Toisena tulee testitapausten dokumentaation ja kuvausten puutteellisuus, joka johtuu vanhasta tavasta tehdä testitapauksia. Kolmantena tulee moduulitestaus, joka on ollut vähäistä koko yrityksessä. Ainoastaan tietyt ohjelmoijat suorittavat moduulitestausta säännöllisesti ja Mainline-testaustiimistä voitaisiin hyödyntää täyspäiväisesti yhtä työntekijää moduulitestaukseen. Neljäntenä tulee projektitiimien toiminnan tuominen lähemmäksi Mainline-testaajien toimintaa Mainline-testauksessa. Viidentenä tulee Mainline-testauksen työkalujen laadukkuus ja toimivuus. Kuudentena tulee Mainline-testauksen työntekijöiden määrän lisääminen. Tällä hetkellä testaajilla on runsaasti töitä, etenkin erilaisten projektien kanssa.

Avainsanat: Mainline, testaustiimi, testitapaukset, automaatiotestaus

Laurea University of Applied Science **Abstract**
Laurea Leppävaara
Information Technology Programme

Jarno Pettersson

Mainline test team development plan for Process Vision

Year 2009 Pages 46

This thesis was assigned by a target company called Process Vision. The target company produces software for energy companies. This assignment consists of a development plan for the Mainline testing team. The idea is to find ways to make the testing team more effective and better.

The thesis is operational based research and the research method is qualitative.

Software testing has developed considerably in the last few years so that there is no software that could be brought to the market without testing it. It depends on the size of the company how many testers there are; there may be a few or a team. There are numerous ways to test software, but usually there is a specific way to test software with test cases. The test cases can test one module or a number of modules working together or the whole system.

Mainline refers to the target company's product Generis main version branch that contains the latest product development and bug fixes. Every month the main version of Mainline is published and there is a separate testing team that tests it. The testing team has there its own test database and test server, on which the team updates to the latest version.

The company's product is Generis that is used by different energy companies. The product is for different metering, measurement and billing systems. It is designed for electricity gas, district heating and water. Information can be input automatically or manually.

For the development plan some research was carried out as well as interviews in the target company. From the information gathered various development plans were created. The first was the automation of test cases that has been low, because of the lack of time that testers have and also because of the software AutoTester. The second is making the test case documentation and description better. This is because of the old way that the test runs were carried out. The third is module testing that is low in the target company. Only certain programmers perform module testing regularly and in the Mainline testing one person could be assigned to carry out module testing fulltime. The fourth is to bring project teams' work closer to the Mainline teams' work in Mainline testing. The fifth is the quality and functionality of the tools that are used in Mainline testing. The sixth is increasing the number of people working in Mainline testing. There is a great deal of work for the testers at the time, especially with projects.

Key Words: Mainline, testing team, test cases, automated tests

Sisällys

1 Johdanto

Tehtävänanto opinnäytetyölle tuli kohdeyrityksen Process Visionin puolelta. Process Vision on johtava energia-alan IT-ohjelmistotoimittaja. Opinnäytetyön tarkoituksena on ensimmäisenä selvittää kohdeyrityksen tämän hetkiset testausmenetelmät ohjelmistotestauksessa. Työn pohja tapahtuu tutkimalla ja selvittämällä testausmenetelmien heikkouksia ja vahvuuksia. Näiden tutkimusten perusteella parannusehdotuksia luodaan, jonka jälkeen analysoidaan ja otetaan mahdollisuuksien mukaan käytäntöön. Tarkoituksena on saada aikaan konkreettisia tuloksia, joilla voidaan tehostaa Mainline-testaustiimin toimintaa. Toiminnallisen tutkimuksen tutkimusmenetelmänä käytän kvalitatiivista tutkimusmenetelmää, joka hyödyntää haastatteluita ja keskusteluita opinnäytetyön selvityksessä. Kvalitatiivinen tutkimusmenetelmä on laadullinen tutkimus.

Työn ydin tulee olemaan tarkan tutkimuksen ja analyysin tekeminen kohdeyrityksen Mainline-testaustiimin testausmenetelmistä. Näiden tulosten jälkeen tulee parannusehdotukset, joiden tarkoitus on parantaa toimintaa. Ennen kaikkea parannusten tulisi tuoda mukanaan testausmenetelmien tasaisuutta, työn ja kustannusten tehokkuutta. Työn tarkoitus on siis luoda avaimet parempaan testausmenetelmään.

Työstä tulevat hyötymään monet sidosryhmät kohdeyrityksessä. Työ ei ainoastaan tule parantamaan testaustiimin työtä, vaan parantamaan koko yrityksen toimintaa. Testaustiimi on yksi yrityksen keskeimmistä toimista, joten sitä parantamalla parantuvat muut muuttujat kohdeyrityksessä, kuten projektitiimit, ohjelmoijat ja myyjät.

2 Tutkimusmenetelmä

Opinnäytetyö tulee olemaan toimintakeskeinen tutkimus. Toimintakeskeinen tutkimus tarkoittaa käytännönläheistä tutkimusta. Opinnäytetyössä tämä tarkoittaa kohdeyrityksen Mainline-testaus tiimin toiminnan parannusehdotusta.

Tutkimusmenetelmänä opinnäytetyössä käytetään kvalitatiivista tutkimusmenetelmää. Kvalitatiivinen tutkimus tarkoittaa laadullista tutkimusta. Kvalitatiivisessa tutkimuksessa hyödynnetään mielipiteitä, sekä niiden syitä ja seurauksia. Kvalitatiivinen tutkimus vastaa kysymykseen mitä, miksi ja kuinka. Kvalitatiivinen tutkimus on tyypillisesti arvioivaa ja luovaa. Kvalitatiivisessa tutkimusmenetelmässä tarkastellaan asiaa objektiivisesti. Opinnäytetyössä tutkin Mainline-testauksen toiminnan parantamista tutkimalla ja haastattelemalla. (Hirsjärvi, Remes & Sajavaara 2001, 152.)

3 Kohdeyritys

Process Vision on yksi johtavista energia-alan IT-ohjelmistovalmistajista Euroopan energiamarkkinoilla. Yritys on täysin keskittynyt energia-alan yrityksiin ja tarjoaa yrityksille kokonaisratkaisuja mittauksista laskutukseen. Vuodesta 1993 yritys on kehittänyt suurta osaamista ja energia markkinatietoisuutta useista eri alueista, mukana avainasiakkaiden kaikissa segmenteistä. Process Vision tuottaa asiakkailleen IT-ohjelmistoja, jotka tukevat heidän energiamarkkinoitaan. (Process Vision 2008, 3.)

Yrityksessä on 126 työntekijää useissa eri paikoissa: Pääkonttori on Helsingissä, R&D yksikkö Jyväskylässä, sekä Suomen lisäksi Ruotsissa ja Hollannissa on myyntipiste. Tämän lisäksi yrityksellä on partneri Saksassa, Itävallassa, Sveitsissä ja Uudessa-Seelannissa. Yrityksen missio on luoda korkeatasoisia energia-alan tietojärjestelmiä energia-alan yrityksille. Yrityksen visio on olla yksi avaintuottajista energia-alan tietojärjestelmille Euroopan markkinoilla ja samalla olla pohjoismaiden markkinoiden johtaja
(Process Vision 2008, 3.)

4 Ohjelmistotestaus

Viime vuosina yrityksissä on aloitettu panostaa yhä enemmän ohjelmistotestaukseen. Asiakasyritykset ja käyttäjät tietävät yhä enemmän ohjelmiston tekniikasta ja toiminnallisuudesta, joka luo uuden haasteen ohjelmistontuottajalle. Ohjelmistossa ei suvaita virheitä tai puutteellisuuksia. Toinen suuri syy on yhteensopivuus erilaisten toimintaympäristöjen kanssa. Erilaiset toimintaympäristöt kehittyvät niin paljon, että erilaiset sovellukset eivät välttämättä toimi samalla tavalla eri alustoissa. Syy voi olla ohjelmaa luotaessa, ohjelmoija ei ole ottanut huomioon tiettyjä ohjelmia tai on tarkoituksella suunnannut ohjelman ainoastaan toimimaan tietyn ohjelman kanssa. Melkein joka toinen testattava ohjelmisto epäonnistuu ja siksi juuri

testausta harrastetaan paljon ja usein. Tämän lisäksi laitteistopuolella voi olla puutteita; jokin ohjelma vaatii toimiakseen tietyn tyyppisen laitteiston. (Kautto 1996.)

On sanomattakin selvää, että erilaiset ohjelmistot vaativat eri määrän testausta. Suuret ja vaativat ohjelmat testataan useammin kuin perustoimintoja tuottavat ohjelmat. Hyvän ohjelman piirteitä ovat: vastaa käyttötarkoitusta, helppokäyttöinen ja yksinkertainen, helposti kontrolloitavissa, turvallinen ja suorituskykyinen. (Kautto 1996.)

Testauksen laatua mitataan usein sillä, kuinka paljon ohjelmaa tai sen toimintoja on testattu. Ei ole olemassa täydellistä testausta, joka olisi niin sanotusti takuuvarma. Testauksen suunnittelu on yhtä luovaa ja vaativaa kuin sen tuottaminen. Ohjelmiston testaus on valmis, kun määränpää on saavutettu. Tämä määränpää voi olla ajassa mitattavissa tai määrällisesti testejä. Testaamisella halutaan viestittää ohjelmiston sidosryhmille tuotteen laadusta. Kaikkia asioita ei välttämättä tulla testaaman, sillä olemassa ei ole täydellistä testausta, eikä testaaja välttämättä tiedä kaikkia asioita, joita tulisi testata. (Kautto 1996.)

Testauksen laatu ja ohjelman jatkuva testaus on tärkeää. Ohjelman elinkaaren lopussa havaittava virhe on huomattavasti työläämpää ja kalliimpaa kuin sellainen testaus, joka etenee rauhassa ja testaa säännöllisesti toimivuutta, koska täydellistä testausta ei ole olemassa, joudutaan testauksessa joskus tyytymään siihen, että ohjelmasta ei löydetä virheitä, sen sijaan että niitä väkisin etsittäisiin ja korjattaisiin. (Kautto 1996.)

4.1 Testaustiimi

Testaustiimi on tiimi, joka on määritelty testaamaan jotakin tuotetta. Testaustiimissä voi olla eri määrä jäseniä. Voi olla isompia testaustiimejä ja joskus vain muutamia testaajia. Testaustiimin koosta huolimatta on hyvä suunnitella vastuut ja tehtävät tiimin jäsenille. Tällaiset päätökset tulisi dokumentoida niin, että niitä voidaan myöhemmin tutkia ja tarkastella. Vastuiden ja roolien jako auttaa erityisesti siinä, että muut jäsenet tietävät, kuka hoitaa mitä ja onko jäsen itse jossakin projektissa mukana. Ennen kaikkea tämä auttaisi uutta tai uusia testaajia, jotka liittyvät tiimiin. Tämän avulla uudet testaajat tietävät keneltä kysyä ja mitä. Selkeiden roolijakojen jälkeen on helpompi esimerkiksi jakaa testejä eri henkilöille. On tärkeää käyttää hyväkseen testaajien vahvuuksia ja sitä myöten jakaa rooleja ja testejä. Henkilöt, jotka osaavat tietyn asian hyvin suorittavat testit yleensä rutinoituneesti ja nopeasti. Tehokkain testaustiimi on sellainen, jossa on erilaista osaamista paljon. Kierrättäminen testien ajossa lisää varmuutta testikierrosten jatkuvuudesta tulevaisuudessa. Kierrättäminen ehkäisee ongelmia sairaustapauksina ja testaajan yrityksestä lähtemisen jälkeen jättämän aukon. Testejä luodessa olisi hyvä hyödyntää asiantuntevaa testaajaa, koska hänellä on vahva näkemys asioista ja toiminnallisuuksista, joita tulisi testata. (Dustin 2003, 63–91.)

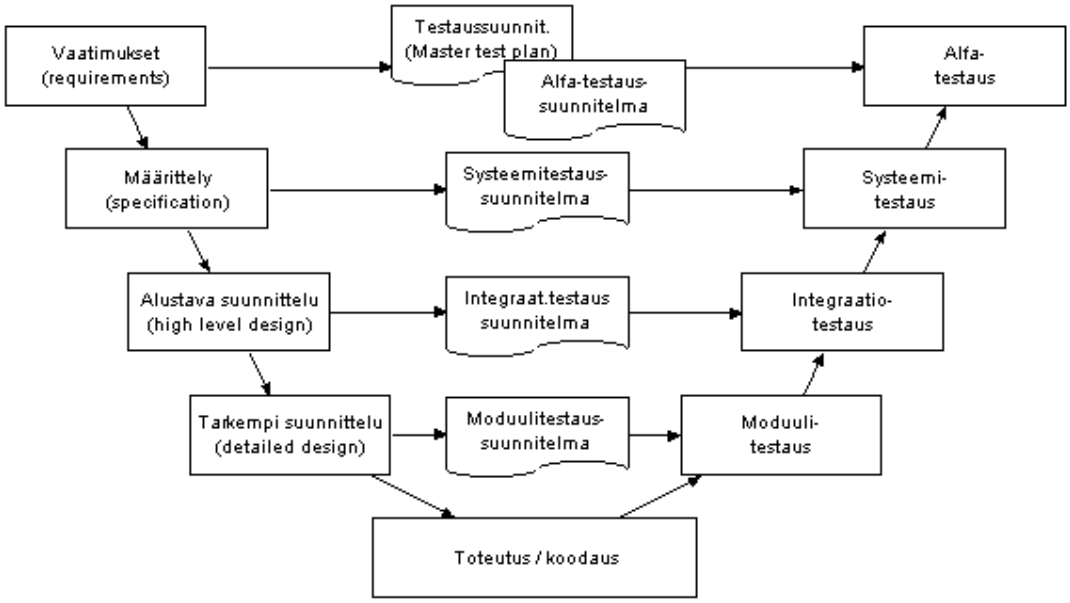
Testaustiimiä voidaan tarkkailla ja arvioida. On tärkeää tutkia testaustiimin toimintaa ja uudistaa sitä mahdollisuuden tullen. Testaustiimissä havaittujen ongelmien parannustoimenpiteet tulisi toteuttaa mahdollisimman nopeasti. Erilaisia strategioita voidaan luoda: testaustiimin jäsenet voivat luoda omia strategioita ja testaustiiminvetäjä voi suunnitella testaustiimille strategian. Tärkeintä on kuitenkin se, että noudatetaan tarkasti jotakin tiettyä strategiaa. Strategia voi pohjautua tapaan tehdä tai suorittaa testejä. Testaajien arviointi on vaikeaa. Tämän takia uuden testaajan palkkaamisessa tulisi kiinnittää huomiota hänen lähtötasoonsa testauksessa ja siihen mitä hän tulee tarkalleen tekemään. Kehityksen uudessa testaajassa havaitsee helpommin kuin rutinoituneesta testaajasta. Testaajia voidaan arvioida aikataulujen, testien määrien ja muiden tapahtumien mukaan. Jos testaaja pysyy aikataulussa, tekee kaikki suunnitellut testit ja noudattaa hänelle annettuja yleisiä säädöksiä, on hän suoriutunut työstään hyvin. (Dustin 2003, 63–91.)

4.2 Testausprosessi

Testausprosessia kuvastaa ohjelmistotestauksen koko elinkaari. Testausprosessin alussa suunnitellaan ja määritellään toiminnot, jotka tulee toimia ohjelmassa. Testausprosessi päätetään, kun ohjelmassa ei ole virheitä ja asiakas on tyytyväinen.

Ensimmäinen askel on asettaa ohjelmalle toimivuusvaatimukset eli vaatimusmäärittely. Vaatimusmäärittely sisältää erilaisia toimintoja, joita ohjelmalta voi odottaa tai odotetaan. Tämän jälkeen tehdään suunnitelma, kuinka ohjelmaa tullaan testaamaan. Tämä vaihe on erittäin tärkeä, sillä siinä tulee ottaa huomioon monia eri asioita, joita testauksen avulla täytyy todeta toimivaksi. Ensimmäisellä asteella testataan pieniä ominaisuuksia eli moduuleja. Nämä moduulit voivat olla yksinkertaisia toimintoja. Seuraava askel on integraatiotestaus, jossa testataan eri moduulien toimivuutta keskenään. Tämän jälkeen tehdään systeemitestaus, mikäli kaikki tähän asti suoritetut askeleet on tehty. Systeemitestaus tarkastelee koko ohjelman toimivuutta. Viimeisenä tulee alfa- ja beeta -testaus, jossa testataan asiakkaalle valmista ohjelmaa.

Yksi tunnetuimmista testaussuunnitelmista on V-malli (kuvio1) (Kautto 1996).



Kuvio 1. Testauksen V-malli (kautto 1996).

4.2.1 Moduulitestaus

Moduulitestauksessa testataan pieniä osia ohjelmasta, kuten funktiota. Moduulitestauksessa on tavoitteena löytää virheitä ja korjata ne. Virheiden korjaus ehkäisee lopulliseen testaukseen tulevaa virheellistä koodia. Mitä aikaisemmin vika havaitaan, sitä parempi ja edullisempi ratkaisu se on yritykselle. (Kautto 1996.)

Moduulitestauksessa pyritään löytämään kokonaisuuksia joita voidaan testata. Kokonaisuudet ovat normaalisti alle 1000 riviä koodia. Yleensä moduulitestausta tekevät henkilöt, jotka ovat suunnitelleet moduulin. Moduulitestauksessa huomataan myös eri moduulien yhteensopivuudet. Moduulitestausta ei voida pelkästään käyttää ohjelmiston laadun mittarina, sillä moduulitestauksessa tutkitaan vain pieniä yksittäisiä toimintoja. (Software Business Compitence.)

Moduulitestausta kohdeyrityksessä harrastetaan, jonkin verran CMS-tiimin toimesta. Moduulitestausta on toivottu toteutettavan enemmänkin, mutta vielä moduulitestaus on jäänyt hieman vähäiseksi.

4.2.2 Integraatiotestaus

Integraatiotestaukseksi kutsutaan testausmenetelmää, jossa kootaan pienet osat yhdeksi isoksi kokonaisuudeksi. Tässä tarkoituksena on testata, että eri ohjelmiston osat toimivat toisten osien kanssa. Tarkoituksena on keskittyä moduulien rajapintojen toimintaan. On olemassa muutama eri tapa suorittaa integraatiotestaus eli alhaalta ylös tai ylhäältä alas tapa. Alhaalta

ylös tarkoittaa, että lähdetään testaamaan ohjelman alimmistatoiminnoista ylöspäin ja ylhäältä alas tarkoittaa testausta, jossa aloitetaan käyttöliittymä tasolta ja mennään syvemmälle. (Kautto 1996.)

4.2.3 Systeemitestaus

Systeemitestaus toteutetaan sitten, kun koko ohjelmisto on kasattu yhteen ja tarkoitus on osoittaa, että toiminnot toimivat juuri niin kuin ne suunniteltiin. Tämä tarkoittaa sitä, että kaikki integraatiotestit on suoritettu onnistuneesti ennen siirtymistä systeemitestaukseen. Tässä testissä selvitetään myös ohjelman suorituskyky, rasitustesti ja turvallisuus. (Kautto 1996.)

4.2.4 Alfatestaus

Tässä testauksessa testataan, että ohjelmisto täyttää asiakkaan vaatimukset ja sitä parannellaan niin kauan, kunnes asiakas on tyytyväinen. Ohjelmisto on käyttövalmis, vaikka vielä keskeneräinen. (Kautto 1996.)

Alfa testaus suoritetaan uuden version alkuvaiheessa. Kohdeyrityksessä alfatestaus suoritetaan ennen uuden version julkaisemista eli normaalista testausperiaatteesta poiketen.

4.2.5 Beetatestaus

Tässä testaus menetelmässä annetaan ohjelmistotestattavaksi potentiaalisille asiakkaille, jotka raportoivat ohjelmiston toimivuudesta. Ohjelmistosta pyritään etsimään erilaisia virheitä ja ongelmia. (Kautto 1996.)

Myös ohjelmiston tekijän testaajat tekevät Beetatestausta. Beetatestauksia voi olla yksi tai useampia. Beetatestauksessa testataan ohjelmiston uusia ominaisuuksia.

4.3 Virheenjäljitysprosessi

Virheenjäljitysprosessia tehdään testitapauksissa, jotka pyrkivät paljastamaan virheitä, jotka pyritään korjaamaan saman tien. Ongelmia on vaikea havaita ja useimmiten kaavailtu ongelma ei ole syy vaan jokin toinen ongelma. Näiden ongelmien korjaus voi aiheuttaa lisää virheitä ohjelmaan, kun ohjelmakoodia muutetaan. (Kautto 1996.)

Tarkoituksena on luoda testitapauksia, joilla odotetaan syntyvän virheitä. Testauksessa havaitut ongelmat pyritään määrittelemään mahdollisimman tarkasti. Tämän jälkeen virheilmoitus

ilmoitetaan ohjelmoijille, jotka luovat ongelmaan korjauksen ja tämän jälkeen testataan uudestaan, kunnes ongelma on poistunut.

Periaatteessa Mainline-testauksen perimmäinen tarkoitus on jäljittää ja korjata virheitä järjestelmästä. Mainline-testauksessa nähdään asia parempana, mikäli ongelmia ja virheitä löytyy, kuin että olisi täysin virheetön.

4.4 Black-box-testaus

Tässä testauksessa tarkkaillaan ohjelman ja sen komponenttien syöttö- ja tulostuskäyttäytymistä. Testauksessa ei välitetä ohjelman sisällöstä tai sen rakenteesta. Testauksessa annetaan erilaisia syötteitä, joiden reaktiota tarkkaillaan. Testaus tapahtuu niin sanotusti ulkopuolisin silmin eli tuntematta ohjelman sisältöä. (Kautto 1996.)

Tarkoituksena ei ole siis luoda ennalta tarkalleen määritellyn materiaalin pohjalta testausta, vaan testauksessa testataan ohjelman toimivuutta. Testissä nähdään, toimivatko kaikki ominaisuudet niin kuin on suunniteltu.

4.5 White-box-testaus

White-box-testauksessa tutkitaan verrattuna Black-box-testaukseen, että jokainen ohjelman komponentti ja toiminto toimivat, kuten on suunnitelmassa määritelty. Tässä testitavassa vaaditaan testaajalta ohjelmointitaitoja. Testaus tehdään niin sanotusti läheltä ja tutuin silmin, kun Black-box-testauksessa ohjelmaa ei tarvinnut tuntea niin hyvin. (Kautto 1996.)

4.6 Ad hoc -testaus

Ad hoc -testaus on testaus, joka suoritetaan ilman tarkkaa suunnitelmaa, kuten elokuva ilman käsikirjoitusta. Ad hoc -testauksessa testataan testaajan oman tavan mukaan ohjelman eri toimintoja. Ongelmia havaittaessa niistä ilmoitetaan ohjelmoijille ja pyritään korjaamaan. (Arguss & Johnson 2000.)

Ad hoc -testauksen tarkoitus on käyttää ja kokeilla erilaisia toimintoja ohjelmistolla. Ad hoc -testaus on valmis, kun testaaja on mielestään testannut kaikkia erilaisia mahdollisuuksia, joita käyttäjä voisi tehdä. Ad hoc –testaus on vapaamuotoinen testaus, josta testaaja vastaa itse suunnittelemalla ja toteuttamalla.

Ad hoc -testausta harrastetaan jokaisessa testitapauksen toiminnossa. Ad hoc -testaus tehdään viimeisenä testimuotona testitapauksissa.

4.7 Testitapaukset

Testitapaukset ovat tapauksia, jotka suunnitellaan sitä varten, että voidaan todeta vastaako ohjelmisto vaatimuksia, joita ohjelmistolle on määritelty. Useimmiten ohjelmiston testaus vaatii useita testitapauksia ennen kuin voidaan todeta, että ohjelmisto toimii oikein. Testitapauksia on kahdenlaisia: on olemassa automaatiotestejä ja manuaalisia testejä. Automaatiotestissä testi tehdään suorittamalla scripti eli valmiiksi luotu komento, joka suorittaa tiettyjä toimintoja ohjelmassa. Manuaalisessa testissä testin suoritus on kuvattu ohjeiden avulla. Jokaista ohjelmiston toimintoa kohtaan on oltava testitapaus tai testitapauksia riippuen toiminnon laajuudesta. Testitapauksissa on useimmiten neuvottu tarkasti, kuinka testi tulisi suorittaa. Testitapauksissa on askeleet, jotka kuvastavat toimia, jotka testaaja joutuu tekemään testin onnistumiseksi. On olemassa erilaisia liitetiedostoja, joita voidaan hyödyntää testauksessa. Viimeisenä ovat testauksen tarkat tiedot, joissa kuvataan mitä ollaan tekemässä ja miksi, mitä sinulla täytyy olla tai täytyy tehdä ennen testiä ja mitä odotetaan testiltä testin lopussa. Automaattisessa testauksessa liitetiedostoissa on useimmiten scripti ja ohje scriptin käyttöä varten. (Kaner 2003.)

Kohdeyrityksessä testitapaukset ovat luotu testaaman aina yhtä tiettyä toiminnallisuutta, kuten laskun syntymistä. Eli laskutusmekanismia varten on useampia pienempiä ja erilaisia testitapauksia.

4.8 Automaatiotestaus

Automaatiotestauksessa käytetään valmiiksi luotuja scriptejä eli komentoja, joita ajetaan ohjelmassa. Scriptit suorittavat niille annetun komennon mukaan toimintoja käyttöliittymässä. On useita syitä, minkä takia kannattaa suosia automaatiotestausta. Kuten moni muu asia, joka on muuttunut automaatioksi niin myös ohjelmistotestaus. Suurin ja painavin tekijä on varmasti se, että manuaalisesti ajettuna testiajo voi kestää todella kauan, joskus jopa liian kauan. Tämän takia automaatiotestaus säästäisi aikaa ja vaivaa testaajalta. Joskus tietyt ohjelmiston suoritusta mittaavat testit vaativat automaatiotestiä. Jotkut testit joudutaan ajamaan yöllä ja juuri silloin automaatiotestausta tarvitaan. Isojen testien testausten kanssa automaatiotestaus voi kesken testin ilmoittaa havaituista virheistä. Automaatiotestauksia (scriptejä) kirjoittaessa voidaan havaita joitakin ongelmia. Automaatiotestaus voi vähentää tietokannan täyttymistä. (Farrell-Vinay 2008, 97.)

Automaatiotestaus on monella tapaa suotava asia. Ajan säästyminen ja oikeanlainen testaustapa syntyvät automaatiotestin avulla. Automaatiotesti suorittaa testitapauksen kokonaan tai osittain.

Automaatiotestauksessa on myös huonoja puolia. Automaatiotestaus voi joskus pysähtyä rakennevirheisiin sen sijaan, että etsisi vikoja koodista. Automaatiotestaus ei huomaa kauneusvirheitä ohjelmasta, kuten testaaja manuaalitestauksessa huomaa. Vastaavasti testaajilla voi olla liian vähän ohjelmointikokemusta. (Farrell-Vinay 2008, 98.)

Liian vähäinen ohjelmointi kokemus voi olla haitta automaatiotestiä tulkittaessa ja niitä luodessa. Automaatiotesti suorittaa testin niin kuin testin luoja on testille tarkoituksen luonut. Automaatiotesti ei kiinnitä siis huomiota mihinkään muuhun, kuin tietyn tuloksen saamiseen, jonka testin tekijä on määritellyt.

Tällä hetkellä Process Visionilla Mainline-testaustiimissä käytetään automaatiotestien(scriptien) tekemiseen PSPad nimistä ohjelmaa. Tällä ohjelmalla voidaan niin luoda, kuin suorittaa AutoTester scriptejä. AutoTester on ohjelma, joka on Process Visionin itse suunnittelema ja toteuttama. Ohjelma sisältää valmiin kirjaston erilaisia komentoja, joilla voidaan luoda ja suorittaa automaatiotestejä. AutoTesterillä suoritetaan automaatiotestit. AutoTesterin avulla voidaan TestTrack-ohjelman puolella suorittaa automaatiotestejä linkittämällä AutoTesterin automaatiotestejä TestTrack-ohjelmaan. TestTrack-ohjelmaan tallennetaan testienkuvaukset ja -tulokset. Yrityksen kannalta automaatiotestit ovat todella tärkeitä, sillä ne nopeuttavat testien suoritusta. Tämä taas antaa testaajalle aikaa muihin tehtäviin, kuten projekteihin tai sitten testaamisen kehittämiseen. Automaatiotestien rakentaminen vie testaajalta eri määrän aikaa, koska testit ovat hyvin persoonallisia. Manuaalisen testin lisäksi 10-50% lisää aikaa suhteessa manuaali testin tekemiseen. Aina ei ole mahdollista luoda automaatiotestejä tai kokonaisia automaatiotestejä erilaisten syiden takia, kuten jokin tietty toiminto, jonka testaaja joutuu itse suorittamaan. Tämän hetken AutoTesterin valmiit komennot tai toiminnot, eivät ole täydellisiä ja tämä estää automaatio testien tekemisen. Tällä hetkellä testaajilla ei ole tarpeeksi aikaa suorittaa automaatiotestejä niin monta, kuin haluaisivat.

4.9 Milloin testaus on valmis?

Testauksen voi lopettaa silloin, kun tarpeeksi vikoja on löytynyt. Toinen tekijä on se, että kaikki testiajot on ajettu. Kolmantena on se, että voidaan todistaa kaikkien ohjelmiston toimintojen toimivuus. Neljäntenä voidaan todeta, että ohjelma on vakaa ja toimii normaalisti. Viimeisenä on kaikkien testien suorittaminen hyväksytysti loppuun. (Farrell-Vinay 2008, 20.)

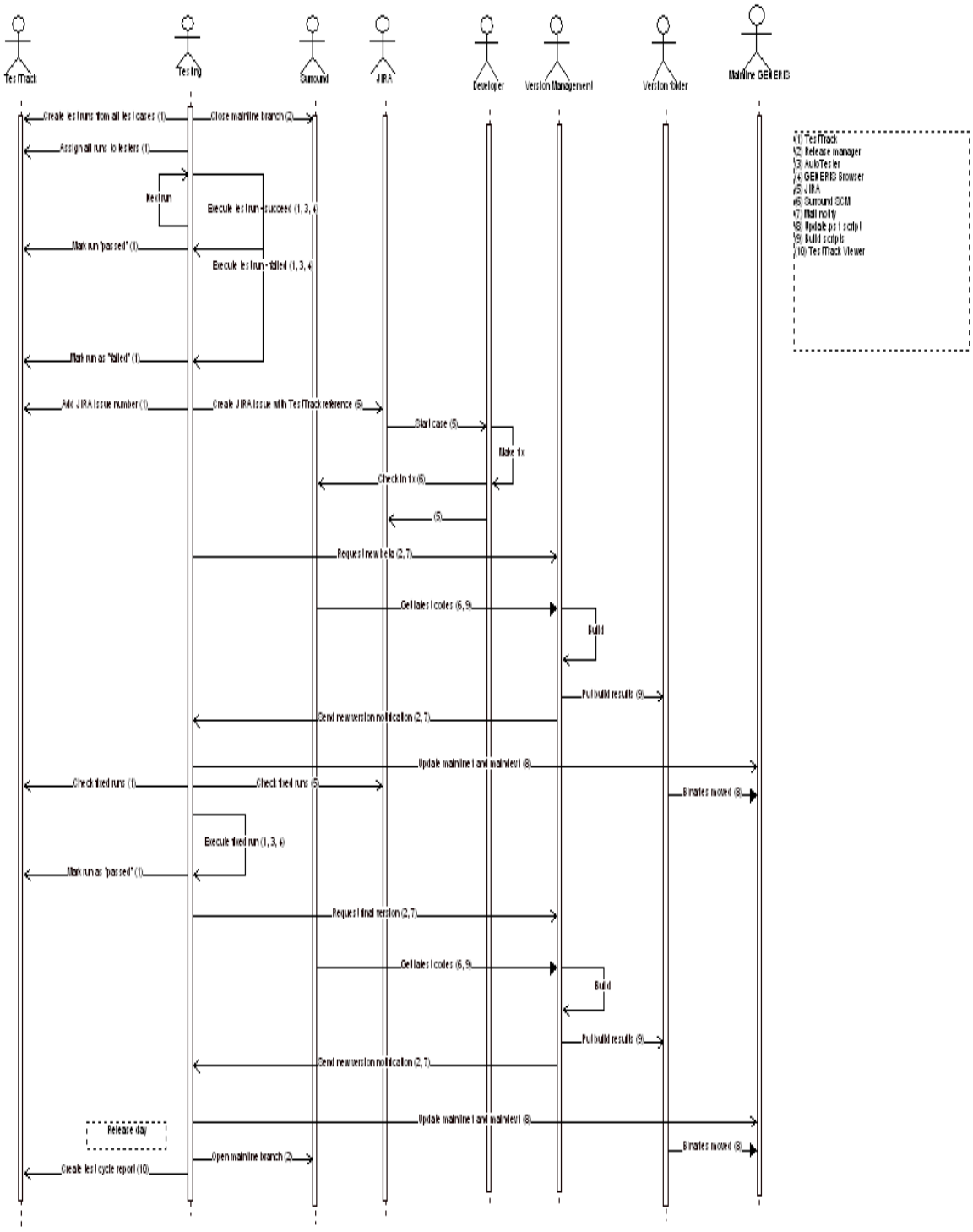
Erilaisten versioiden elinkaari tulee valmiiksi, kun tietyt kriteerit täyttyvät. Beetatestaus ja alfatestaus tehdään niin useita, että voidaan olla varmoja ohjelman laadusta.

5 Mainline

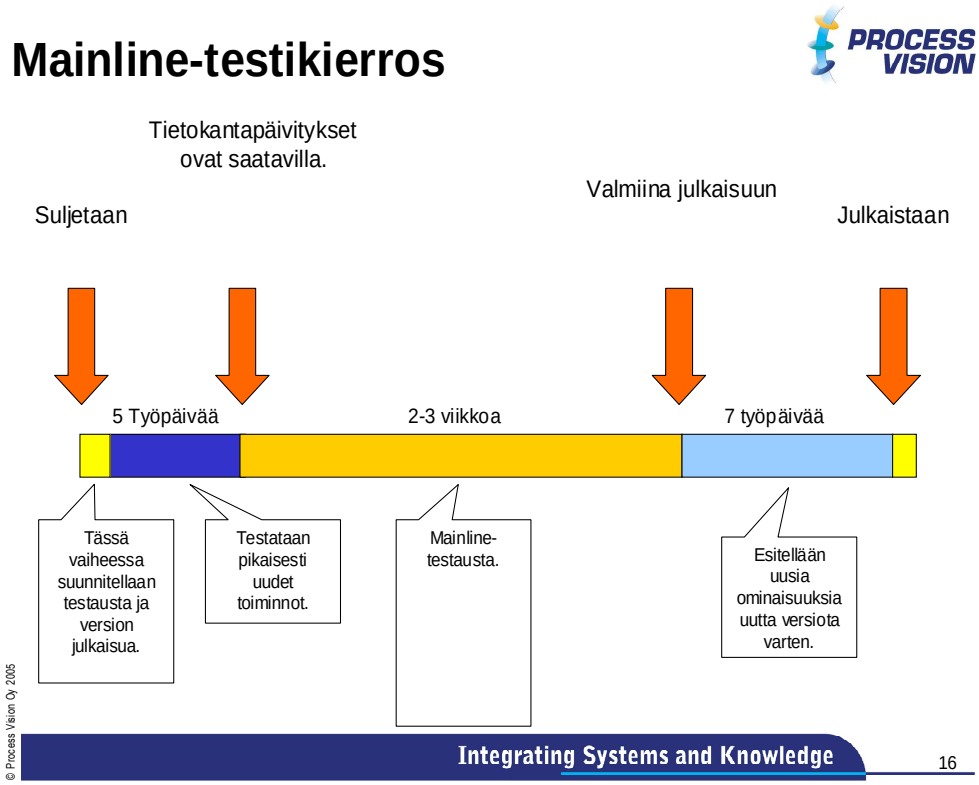
Mainline tarkoittaa kohdeyrityksessä viimeisimmän tuotantoversion sijaintia eli palvelinta, johon versio on laitettu. Mainline on omalla palvelimella, jossa sitä testaavat tai käyttävät Mainline-testaajat. Mainline-palvelimella on siis aina viimeisin tuotantoversio. Ohjelmoijilla on oma vastaava palvelin eli Mainlinedev, jota he käyttävät omiin kehitys ja testaus tarkoituksiin, mutta vastaa kuitenkin Mainline-palvelinta. Molemmat palvelimet päivitetään aina samaan versioon. Mainline on palvelin, jonne ohjelmoijat asentavat viimeisimmät koodit ja virhekorjaukset. Mainlinedev on pelkästään ohjelmoijien käytössä, vaikka Mainline-testaajat sitä päivittävät. Projektitiimeillä on myös omat palvelimet, jossa testaavat asiakkaiden eri versioita. Mainline-testaajat joutuvat projekteissa testaamaan kyseisiltä projektitiimien palvelimilta toimintoja. Ohjelmoijat päivittävät näihin versioihin ongelma tai virhe korjaukset.

5.1 Mainline-testausprosessi

Mainline-testausprosessia voidaan kuvata parhaiten muutamalla kuvalla, jotka avaavat asiaa paremmin. Kuva 2 osoittaa koko testausprosessin tällä hetkellä. Aikajana kuluu ylhäältä alas ja kuvassa kuvastetaan, kuinka ensiksi luodaan testitapauksia TestTrack-ohjelmassa, joita testataan. Onnistunut testitapaus merkataan onnistuneeksi. Epäonnistunut testitapaus kirjataan Jira-ohjelmaan ja samalla kirjataan TestTrack ohjelmaan syyn kera, miksi testitapaus epäonnistui. Ohjelmoija katsoo Jiraan tulleet vikaraportit ja korjattuaan viat, vie uudet päivitykset Surround-versionhallintajärjestelmään. Ohjelmoija ilmoittaa myös Jiraan, josta lähtee viesti testaajalle, että korjaus on tehty. Testaaja pyytää seuraavaksi uutta versiota. Uuden version koodit käännetään uutta versiota varten versionhallinnassa, josta tallennetaan versio kansioon. Uudesta versiosta tulee ilmoitus, että on valmis. Testaaja saa tiedon uudesta versiosta ja päivittää uuden version Mainline:n ja Mainlinedev:n. Tämän jälkeen tarkastetaan korjatut testitapaukset. Korjattujen testitapausten testauksen jälkeen pyydetään uutta versiota ja päivitetään jälleen Maineine ja Mainlinedev.



Kuvio 2. Mainline-testausprosessi



Kuvio 3. Mainline-testauskierros

Parannusehdotuksen tarkoituksena on saada Kuvassa 2 ylhäältä alaspäin katsottuna matka lyhenemään ja Kuvassa 3 tarkoituksena on saada Mainline-testaus osuus lyhennettyä ajallisesti.

5.2 Mainline-testaustiimi

Mainline-testaustiimi koostuu tällä hetkellä kahdeksasta henkilöstä. Suurin osa Mainline-testaajista sijaitsee Helsingin toimipisteellä ja yksi testaaja Jyväskylässä. Mainline-testaustiimin tavoite on löytää mahdollisimman paljon ongelmia ja virheitä Generiksestä. Sitä enemmän, kun ongelmia ja virheitä löydetään, sitä varmempia voidaan olla ohjelman toimivuudesta, kun ongelmat ja virheet on korjattu. Useimmiten Mainline-testauksessa testataan useita pieniä kokonaisuuksia. Pienillä kokonaisuuksilla tarkoitetaan erilaisia toimintoja. Isompia kokonaisuuksia olisi tarkoitus testata, mutta toistaiseksi ei ole otettu testaukseen mukaan. Mainline-testaustiimi saa tietoa uusista testattavista kohteista mm. projektitiimeiltä ja sitä varten testaajat luovat uusia testitapauksia. Suurimman osan testattavista kohteista tulee erilaisten projektien yhteydessä, jossa testataan asiakkaalle tehtyä ohjelmaa. Virheen löytämisen jälkeen pyritään ratkaisemaan ongelma mahdollisimman pian. Ratkaisun löydettyä testitapaus testataan ja merkitään testitapaus hyväksytyksi. Mikäli testaus onnistuu hyvin, sillä on tulevaisuuden kannalta ennalta ehkäisevä vaikutus. (Hulttinen 2006.)

Tavoite on myös kehittää testaajia siihen suuntaan, että heistä tulisi asiantuntijoita ohjelman parissa ja voisivat alkaa työskentelemään mukana projektitehtävissä. Projektitehtävät, jotka ovat useimmiten asiakkaan suunnalta tulleita realistisia ja akuutteja tehtäviä. Tällä hetkellä testaajien kysyntä erilaisiin projektitiimeihin on noussut suureksi. Testaajien työskentely erilaisten projektien parissa on kasvanut melkein täysipäiväiseksi työksi. Tämä taas hidastaa itse Mainline-testauksen toiminnan kehittämistä. (Hulttinen 2006.)

Mainline-testauksessa ajetaan niin sanottuja testaussyklejä eli testauskierroksia. Kaikki testattavat osat testataan ennen seuraava versiota. Yhden syklin aika on 2-3 viikkoa. Syklejä on melkein poikkeuksetta joka kuukausi. (Hulttinen 2006.)

Kaikki luodut ja suoritetut testitapaukset ovat kaikkien kohdeyrityksen työntekijöiden nähtävissä. Tämä taas helpottaa tiedonkulkua ja virheidenkorjausta. Myös esimiehillä on helppo seurata testaajien toimintaa. (Hulttinen 2006.)

Testitapausten luonti on aika monimutkainen asia ja testitapauksesta tulee aina persoonallinen. Testitapauksen luo yksi testaajista ja testiä saattaa testata jokin tai jotkut muut testaajat. Tämän takia testitapaus täytyy olla selkokielellä ja muutenkin tarkkaan ohjeistettu. Mainline-testauksessa harrastetaan niin sanottua kierrätystä, jossa jokainen testaaja vuorollaan saa oman testipaketin, joka sisältää tiettyjä testejä. Eli jokaisessa testisyklissä tulee tiettyyn pisteeseen saakka eri testiajoja, kunnes alkaa kierros alusta. Tästä johtuen testien dokumentaatiot täytyy olla laadukkaasti ja huolella kirjoitetut. Mikäli dokumentaatiossa on puutteita, sitä tulisi korjata niin, että seuraavalla testaajalla olisi helpompi suorittaa testi. Testin dokumentaation korjaamisesta on vastuussa testaaja, joka havaitsee testiä suorittaessa puutteellisia kohtia. (Hulttinen 2006.)

Mainline-testauksessa osa testeistä on automatisoituja eli sisältävät scriptin, joka ajaa testin lävitse ja ilmoittaa tuloksen testaajalle. Mikäli automaatiotestaus epäonnistuu, tulee testaajan tarkistaa testi manuaalisesti. Automaatiotestaus nopeuttaa testaajien työtä useimmiten ja tämän takia sen suosiminen on hyvä asia, mutta vielä automaatiotestausta on melko vähän. Automaatiotestien tekeminen on yksi Mainline-testauksen suurista haasteista lähitulevaisuudessa. (Hulttinen 2006.)

5.3 Testitapausten kierrätys vai erikoistuminen

Tällä hetkellä yksi mielenkiintoisimmista kysymyksistä on Mainline-testaustiimissä testitapausten kierrättäminen sykleissä. Hetki sitten takaisin käyttöön otettu toiminta on hyvä asia monelta kannalta. Testitapausten kierrättäminen eri testaajalle on tapa, jolla nähdään testit uudessa valossa ja samalla voidaan havaita uusia virheitä, mutta ennen kaikkea tällä ehkäistään vaara, joka on testaajan lähteminen Mainline-testaustiimistä toisiin tehtäviin. Vanha

tapa oli, että jokainen teki ainoastaan tiettyjä testitapauksia ja erikoistuvat, jonkin tietyn testitapauksen asiantuntijaksi. Tämä toi tietenkin tehokkuutta testaukseen. Nykyinen tapa on parempi tulevaisuutta ajatellen, mikäli tulee vaihtuvuutta ja samalla saadaan uusia näkökulmia tiettyjen testitapausten testaamiseen. Vanha tapa oli tehokas, mutta tavallaan riskialtis pitkällä tähtäimellä.

5.4 Mainline-testaustiimin toiminnan parantaminen

Varmasti jokaisessa työpaikassa ja jokaisessa tiimissä on parantamisen varaa maailmanlaajuisesti. Niin on myös Mainline-testaustiimissä. Tutkimuksien ja haastatteluiden pohjalta luotiin parannusehdotuksia Mainline-testaustiimiille.

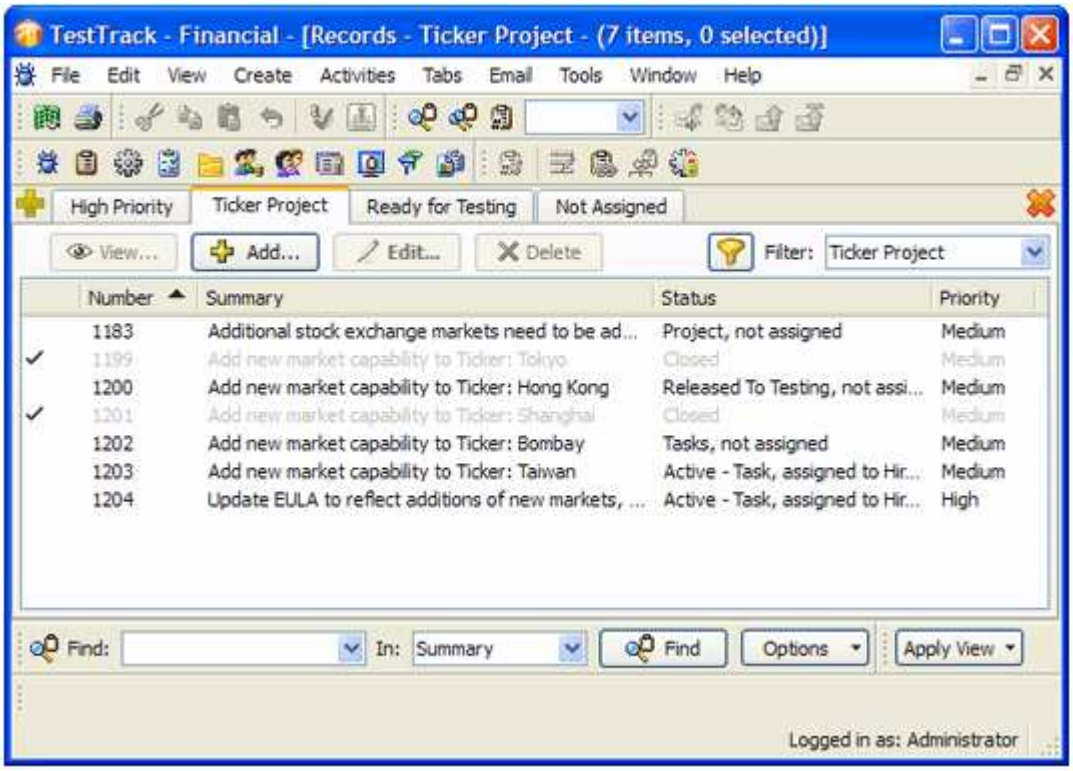
Parannusehdotuksia tuli jonkin verran ja niitä tulee varmaan tulevaisuudessakin. Ensimmäisenä parannusehdotuksena on testien automatisointi. Tämä takaisi testien testaamiseen kuluvan ajan lyhenemistä ja toiminnan nopeutumista. Toisena ehdotuksena tuli testitapausten dokumentointi ja kuvaukset. Testitapausten heikko dokumentointi ja kuvaus tekevät uuden testaajan testien suorittamisesta todella hankalaa. Kolmantena ehdotuksena olisi yksikkötestauksen käyttöönotto eli moduulitestaus. Tämä toisi selvempää koodia ohjelmiin ja samalla luotu koodi testattaisiin saman tien tai mahdollisimman nopeasti. Koodi voitaisiin korjata myös mahdollisimman nopeasti. Tämä vaatisi testaustiimin yhdeltä valitulta jäseneltä lisää työtä ja mahdollisesti uuden testaajan palkkaamista testaustiimiin. Neljäntenä olisi projektitiimien omien realististen testausten tuominen mahdollisimman lähelle Mainline-testaustiimin toimintaa. Mainline-testauksessa testattaisiin entistä realistisempia testejä. Viidentenä olisi uusien testaustyökalujen tarkkailu, eli onko markkinoilla vieläkin parempia työkaluja. Kuudentena on testaajien määrä eli tarvitaanko testaustiimiin lisää testaajia. Parannusehdotuksista lisää kappaleessa 11. Parannusehdotuksista on kerrottu tarkemmin kappaleessa 11.

6 Mainline-testauksessa käytettävät ohjelmat

Mainline-testauksessa testataan yrityksen tuotetta eli Generistä. TestTrack nimistä ohjelmaa käytetään itse testitapausten suorittamiseen. TestTrack-ohjelma on tavallaan erillään Generiksestä ja tavallaan sidoksissa. Sidoksissa ohjelma on silloin, kun ajetaan automaatiotestejä, sillä silloin TestTrack-ohjelman puolelta käynnistetään automaatiotestiajo, joka suorittaa toiminnot Generiksessä. TestTrack toimii muuten niin sanottuna testaajia tukevana alustana, jossa on kaikkien testien tiedot ja ohjelmaan voidaan luoda uusia testitapauksia. PSPad on ohjelma, jota käytetään luomaan uusia automaatiotestejä. AutoTester on ohjelma, jota hyödynnetään automaatiotesteissä. AutoTester-ohjelmaan on luotu valmiita automaatiotestien komentoja eli funktiota, joita voidaan hyödyntää PSPadilla. Lopuksi valmiit automaatiotestit asetetaan TestTrack-ohjelmaan ja Surround-versionhallintaohjelmaan. Surround-versionhallintaohjelma on Seapine-yrityksen tuottama, kuten TestTrack.

6.1 TestTrack

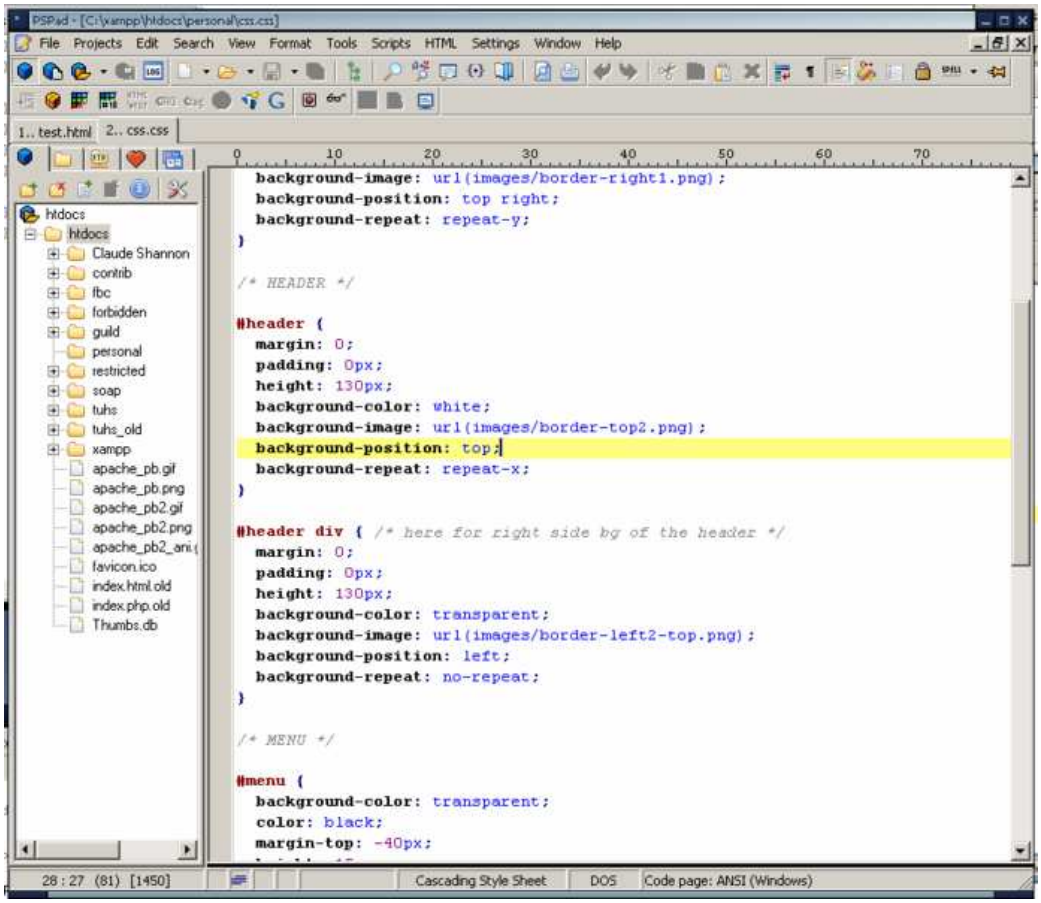
kohdeyrityksessä käytetään TestTrack Pro -nimistä ohjelmaa. Ohjelmalla suoritetaan testiajoja (test runs) ja tehdään uusia testitapauksia (test case). Ohjelma on melko helppo käyttää ja ohjelma on monipuolinen. Ohjelmalla voi tehdä kaikenlaisia erilaisia testiajoja. Ohjelmalla voi tehdä manuaalisia testiajoja tai sitten käyttää toisilla ohjelmilla luotuja automaattisia testiajoja (scriptejä), jotka ovat itse testaajien tekemiä. TestTrack on myös hyödyllinen väline säilyttämään tietoja testeistä ja testiajoista. Näitä voidaan hyödyntää, mikäli törmätään ongelmiin esimerkiksi testienajossa.(Seapine Software 2009.)



Kuva 1. TestTrack Pro (Seapine Software 2009).

6.2 PSPad

PSPad on ohjelma, jota käytetään yrityksessä Mainline-testauksessa AutoTester-scriptien eli komentojen kirjoittamiseen. Ohjelmaa käyttävät useimmiten ohjelmoijat. Ohjelman ominaisuuksia ovat mm. alleviivaus eli ohjelma esittää tietyn kohdan koodista korostetusti. Ohjelmassa on myös automaatio syöttö eli kirjoittaa muutaman kirjaimen ja ohjelma täydentää itse. Ohjelma osaa myös käyttää useita eri ohjelmointi kieliä. Ohjelma on ilmainen. (PSPad 2009.)



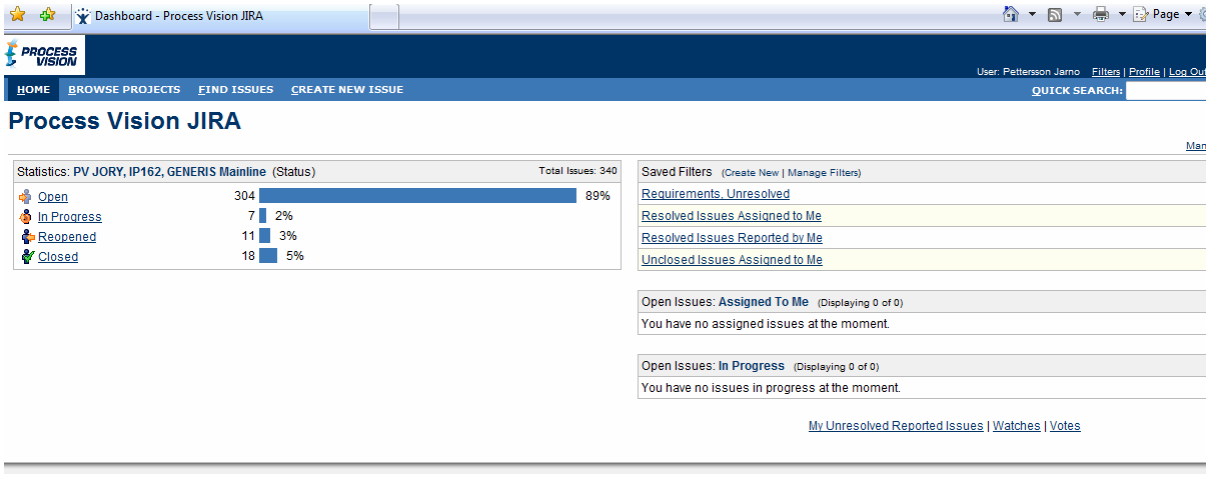
Kuva 2. PSPad (PSPad 2009).

6.3 AutoTester

AutoTester on kohdeyrityksen kehittämä ja luoma ohjelma. AutoTester-ohjelmalla sisältää valmiin kirjaston erilaisista toiminnoista ja funktioista. AutoTester on ohjelma, jota käytetään testien automatisoinnissa. AutoTesterillä suoritetaan automaatiotestejä, jotka linkitetään TestTrack ohjelmaan.

6.4 Jira

Jira on kohdeyrityksessä käytössä. Jiraa käytetään erilaisten ongelmien ja tehtävien hallintaan. Mainline-testaukessa Jira-ohjelmaa käytetään vikaraportointiin. Jira-ohjelmaa voidaan käyttää ohjelmisto virheiden raportointiin, help-desk työlistana, projektien hallintaan. (Jira 2009.)



Kuva 3. Jira

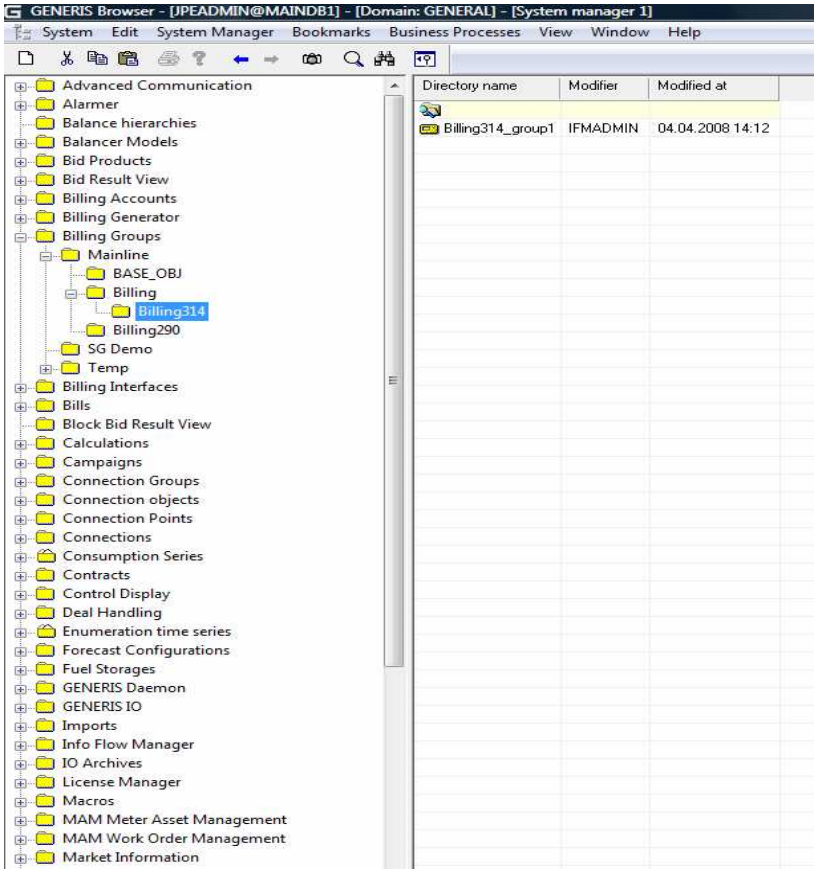
Kohdeyrityksessä Jira on erittäin tärkeässä roolissa, koska kaikenlaisten ongelmien ja virheiden raportointi tapahtuu Jiran kautta. Jira on hyvä ohjelma, koska ohjelmaan voidaan määritellä erilaisia asioita. Ohjelman runsaiden ominaisuuksien ja mahdollisuuksien myötä mm. ohjelmistojen vikaraportointi on tärkeä saada kuvattua mahdollisimman hyvin, jotta ohjelmoijalla on mahdollisuus korjata virheet tai ongelmat ohjelmasta.

7 Testattava ohjelma Generis

Mainline-testauksessa testaajat testaavat yrityksen tuotetta eli Generistä. Generis on ohjelma, jota käyttävät eri energia-alan yritykset. Generis on erilaisten mittaus-, lukema- ja laskutustietojen hallintaa varten soveltuva sovellus, jossa voidaan niin varastoida, kun käsitellä tietoa. Generis on suunniteltu sähkö-, kaasu-, kaukolämpö- ja vesimittauksiin. Tietoa voi syöttää Generikseen automaattisesti sanomien avulla ja manuaalisesti käyttäjän tahosta. Ohjelma on toteutettu Windows käyttöjärjestelmälle ja ohjelma käyttää Oraclen tietokantoja. Generis on helposti muunneltavissa, joten tuotetta voidaan muokata asiakkaan tarpeiden mukaan. (Process Vision 2009.)

Uudessa energiavertikaalissa Generistä käytetään yhdessä Microsoftin CRM:n ja Axaptan kanssa. Ideana on luoda laskutus- ja kirjanpitojärjestelmä. CRM toimii asiakaspalvelun välineenä eli sopimuksia hoidettaessa, Generis on puhtaasti tiedon varastoinnin ja laskutusaineiston tuottamiseen ja Axapta hoitaa kirjanpidon.

Alla oleva kuva on otettu yrityksen tuotteesta Generiksestä. Kuvassa näkyy generiksen puunäkymä, josta voi valita mitä haluaa tehdä. Olen esimerkissä menossa laskutusryhmään. Laskutusryhmässä voi luoda laskuja ja lähettää kirjanpitoon.



Kuva 4. Generis

8 Kultainen Keskitie

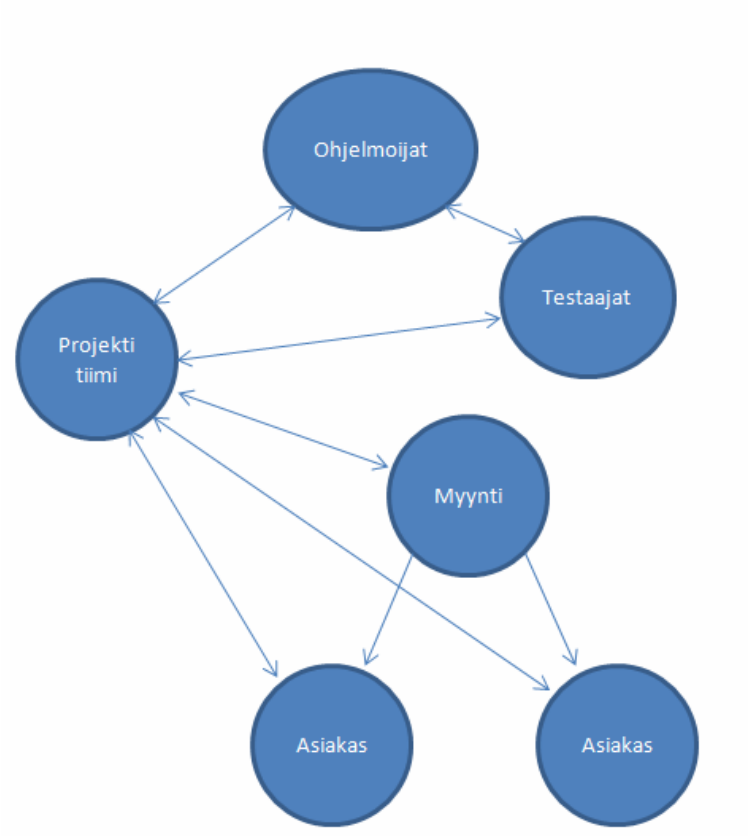
Kultaisella keskitiellä tarkoitetaan yrityksen eri toimijoiden parempaa juottamista yhteen ja toimivuuden tehostamista. Kohdeyrityksen sidosryhmien toiminta menee niin, että kiireisin sidosryhmä ajaa toisten sidosryhmien tehtävien ohi. Tietenkin tämä on selvää, koska yritys haluaa menestyä ja olla tuottoisa.

Toimijat ovat: Testaajat, jotka testaavat uutta ohjelmakoodia ja uusia ominaisuuksia uudessa versiossa. Testaajat suunnittelevat testitapauksia heille annettujen ohjeiden mukaan ja sitten testaavat niitä. Tällä hetkellä toiminta menee niin, että jokaisen kuun toisella viikolla tiistaina aloitetaan testaus eli sen jälkeen uusia ohjelmapäivityksiä ei saa versioon laittaa ellei saa erityistä lupaa, virhekorjauksia saa laittaa, mikäli testaaja niin pyytää. Näiden testien suorittaminen kestää 2-3 viikkoa. Tämän testauksen aikana testaajat ovat tekemisissä ohjelmoijien kanssa, mikäli havaitsevat ongelmia testitapauksissa, kuten pyytävät ongelma tai virhe korjauksia. Testaajat ovat myös mukana projektitiimien kanssa eli testaavat asiakkailla havaittuja ongelmia asiakkaiden versioissa ja pyrkivät ratkomaan syyn ongelmiin. Suurimman osan testaajien ajasta kuluu erilaisten projektien kanssa. Tämä vähentää aikaa kehittää testaustiimin toimintaa.

Toinen toimijaryhmä on myyjät, jotka myyvät yrityksen tuotteita uusille mahdollisille asiakkaille ja tämä taas tuottaa työtä projektitiimille. Projektitiimit aloittavat tuotteen suunnittelun asiakkaalle. Asiakkailla on erilaisia tarpeita ja niiden mukaan joudutaan elämään ja suunnittelemaan asioita.

Kolmantena toimijana ovat projektitiimit. Projektitiimit ovat tekemisissä pääosin asiakkaiden kanssa. He pyrkivät vastaamaan täysipainoisesti, siitä että asiakas saa alusta alkaen sellaisen tuotteen mitä ovat kaavailleet. Tämä tietysti tuottaa lisätyötä ohjelmoijille, sillä tiettyjen hienosäätöjen ja pienten muutosten tekeminen tuottaa lisää työtä ja testausta. Tämä taas johtuu asiakkaan erilaisista tarpeista ja muista työkaluista ja toimijoista asiakkaan päässä. Projektitiimit myös suorittavat omia testejä asiakkaiden ohjelmiin, mutta tukeutuvat testaajiin tarpeen mukaan.

Viimeisenä toimijana ovat ohjelmoijat, joiden pääasiallinen tarkoitus on luoda uutta koodia ja kehittää yrityksen tuotteita. Tämän lisäksi he joutuvat tekemään muokkauksia eri ohjelmaversioihin, joita asiakkaat käyttävät. Projektitiimit sopivat erilaisista muutoksista ohjelmiin ja sen jälkeen ohjelmoijat tekevät muutoksen koodiin. Ohjelmoijat myös tekevät ongelma tai virhe korjauksia, mikäli niitä havaitaan testaajien suunnalta.



Kuvio 4. Kultainen keskitie

9 Parannusehdotus haastattelu

Taustatietoa kartoitettiin opinnäytetyöhön vapaamuotoisella haastattelulla. Haastattelun muoto oli kvalitatiivinen tutkimus. Kvalitatiivinen tutkimus tarkoittaa vapaa muotoista mielipide haastattelua, jossa tavoitteena on saavuttaa tuloksia keskustelun pohjalta. Kvalitatiivista tutkimus tapaa käytetään useimmiten markkinataloustutkimuksissa. (Taloustutkimus Oy 2007.)

Tutkimusta lähdettiin tekemään Mainline-testaukseen haastattelun kautta. Haastatteluita suoritettiin niin, että haastateltiin eri henkilöitä eri puolilta yritystä. Haastattelusta sai koottua erilaisia näkökulmia Mainline-testauksesta. Kohdeyrityksessä haastateltiin 13 eri henkilöä. Haastateltavat: ohjelmistokehityksen vetäjä, Mainline-testauksen kehitysvastaava, ohjelmointitiimistä haastattelin CMS-tiiminvetäjä ja .Net-tiiminvetäjää, projekti-insinööriä ja projektipäällikköä, asiakaspalveluinsinööriä ja Mainline-testauksen-tiiminvetäjää ja viittä eri testaajaa Mainline-testauksesta.

9.1 Mitä teet tai mikä on toimenkuvasi?

- Testaaja A: Toimii kohdeyrityksessä testaajana. Tehtäviin kuuluu luoda ja suorittaa testitapauksia. Hän on Sesam-järjestelmän asiantuntija ja implementoija. Hän tuottaa myös runsaasti dokumentteja.

- Testaaja B: Toimii kohdeyrityksessä testaajana. Tehtäviin kuuluu luoda ja suorittaa testitapauksia. Viime aikoina päätoiminen työkuva on ollut mukana erilaisissa projekteissa.

- Testaaja C: Toimii kohdeyrityksessä testaajana. Tehtäviin kuuluu luoda ja suorittaa testitapauksia. Toimii pääasiassa projekteissa.

- Testaaja D: Toimii kohdeyrityksessä testaajana. Tehtäviin kuuluu luoda ja suorittaa testitapauksia. Hän suunnittelee myös yksikkötestejä.

- Testaaja E: Toimii kohdeyrityksessä testaajana. Tehtäviin kuuluu luoda ja suorittaa testitapauksia.

- Ohjelmistokehityksen vetäjä: Hänen toimiin kuuluu useampi eri asia kohdeyrityksessä. Hän hoitaa ohjelmointitiimin resursseja(suunnittelu/implementaatio), versiohallintaa, dokumentaatiota ja vastaa kokonaisvaltaisesti Mainline-testauksesta.

- Mainline-testauksen vetäjä: Vastaa Mainline-testauksen päivittäisten asioiden hoitamisesta. Luo ja suorittaa myös testitapauksia.

- Projekti-insinööri: Hänen tehtäviinsä kuuluu hoitaa Keski-Euroopan asiakkaiden asioita ja toimii käytännönviestinnässä.

- Asiakaspalveluinsinööri: Toimii erilaisten tukihommien ja juoksevien asioiden lisäksi IT-tuki hommissa. Näissä tehtävissä testataan käytännössä asiakkailla ilmenneitä virheitä.

- Projektipäällikkö: Toimii ja vastaa erilaisista asiakasprojekteista.

- Mainline-testauksen kehitysvastaava: Toimii kohdeyrityksessä Mainline-testauksessa. Tehtäviin kuuluu luoda ja suorittaa testitapauksia ja yksikkötestejä.

- .Net-tiiminvetäjä: Toimii .Net-ohjelmointitiiminvetäjänä ja vastaa tiimin toiminnasta.

- CMS-tiiminvetäjä: Toimii CMS-ohjelmointitiiminvetäjänä ja vastaa tiimin toiminnasta.

9.2 Miten hyvin tunnet Mainline-testauksen?

- Testaaja A: Tuntee Mainline-testauksen todella hyvin, koska on työskennellyt tiimissä 1,5 vuotta.

- Testaaja B: Tuntee Mainline-testauksen hyvin, sillä on työskennellyt tiimissä kaksi vuotta.

- Testaaja C: Tuntee Mainline-testauksen perusteet ja on työskennellyt tiimissä vajaan vuoden.

- Testaaja D: Ollut Mainlie-testauksen toiminnan käynnistymisestä asti mukana toiminnassa, joten tuntee Mainline-testauksen hyvin.

- Testaaja E: Tuntee Mainline-testauksen hyvin, sillä ollut alusta alkaen Mainline-testauksessa mukana.

- Ohjelmistokehityksen vetäjä: Tuntee Mainline-testauksen hyvin, sillä oli ensimmäinen Mainline-testauksen vetäjä.

- Mainline-testauksen vetäjä: Osaa Mainline-testauksen perusteet ja ollut Mainline-testauksessa mukana reilun vuoden.

- Projekti-insinööri: Osaa Mainline-testauksen perusteet.

- Asiakaspalveluinsinööri: Osaa Mainline-testauksen perusteet.

- Projektipäällikkö: Tuntee Mainline-testauksen toiminnan, vaikka ei ole ikinä ollut testaamassa.

- Mainline-testauksen kehitysvastaava: Tuntee Mainline-testauksen todella hyvin ja työssä käytettävät työkalut.

- .Net-tiiminvetäjä Ei oikeataan tunne Mainline-testauksen toimintaa.

- CMS-tiiminvetäjä: Tuntee Mainline-testauksen erilaisten vikakorjausten pohjalta, sillä Mainline-testauksessa ilmenevien vikojen korjausta delegoi CMS-tiiminvetäjä.

9.3 Miten olet tekemisissä Mainline-testauksen kanssa?

- Testaaja A - E: Työskentelevät tiimissä.

- Ohjelmistokehityksen vastaava: Toimii Mainline-testaustiimin valvojana ja suunnittelijana.

- Mainline-testauksen vetäjä: Työskentelee tiimissä.

- Projekti-insinööri: On yhteydessä Mainline-testaustiimiin, mikäli törmää ongelmiin. Useimmiten ongelman syynä ovat Keski-Euroopan asiakkaat.

- Asiakaspalveluinsinööri: On yhteydessä Mainline-testaukseen erilaisten vikojen takia.

- Projektipäällikkö: Toimii niin sanotusti konsultin asemassa Mainline-testaustiimille projekteihin liittyvissä asioissa.

- Mainline-testauksen kehitysvastaava: Työskentelee tiimissä.

- .Net-tiiminvetäjä: Ei ole oikeastaan tekemisissä Mainline-testauksen kanssa.

- CMS-tiiminvetäjä On tekemisissä Mainline-testauksen kanssa erilaisten vikakorjausten takia.

9.4 Kuinka paljon aikaa kuluu Mainline-testauksen parissa?

- Testaaja A: Mainline-testauksen testitapausten tekemiseen 40 % ajasta ja 60 % menee testitapausten suorittamiseen.

- Testaaja B: Viimeinen vuosi on mennyt tiiviisti projektien kanssa, joten on vaikea sanoa, kuinka aikaa kuluu, mutta yksi viikko testitapausten suorittamiseen ja kolme viikkoa testitapausten tekemiseen.

- Testaaja C: Suurin osa ajasta menee testitapausten luomiseen.

- Testaaja D: Testitapausten luomiseen menee aikaa muutamasta tunnista – yhteen viikkoon.

- Testaaja E: 90 % ajasta menee testitapausten luomiseen ja loput ajasta menee ylläpitohommissa.

- Ohjelmistokehityksen vetäjä: Suurin osa ajasta menee Mainline-testauksen toiminnan kehittämisen ja valvonnan parissa.

- Mainline-testauksen vetäjä: Aikaa kuluu täyspäiväisesti Mainline-testauksen parissa. Testaamisessa testitapausten luomisessa kuluu 60 % ajasta ja testitapausten suorittamiseen kuluu 40 % ajasta.

- Projekti-insinööri: On loppujen lopuksi melko vähän tekemisessä Mainline-testauksen kanssa ja ajallisesti n. yksi tunti viikossa keskimäärin kuluu Mainline-testauksen kanssa.

- Asiakaspalveluinsinööri: Silloin tällöin on tekemisessä Mainline-testauksen parissa.

- Projektipäällikkö Toimii satunnaisesti Mainline-testauksen parissa.

- Mainline-testauksen kehitysvastaava: Normaaleihin rutinoituneisiin Mainline-testaustiimin tehtävien kuluu aikaa viikossa neljästä – viiteen tuntia.

- .Net-tiiminvetäjä: Ei ole tekemisissä Mainline-testauksen kanssa.

- CMS-tiiminvetäjä: On satunnaisesti tekemisissä Mainline-testauksen kanssa.

9.5 Miten yhteistyö toimii Mainline-testauksen kanssa?

- Testaaja A-E: Työskentelevät Mainline-testauksessa.

- Ohjelmistokehityksen vetäjä: Työskentelee aktiivisesti Mainline-testaustiimin kanssa. Yhteistyö toimii Mainline-testauksen kanssa hyvin.

- Mainline-testauksen vetäjä: Työskentelee Mainline-testauksessa.

- Projekti-insinööri: Yhteistyö Mainline-testaustiimin kanssa sujuu hyvin. Saa aina palvelua ja tukea ongelmiin.

- Asiakaspalveluinsinööri: Yhteistyö toimii Mainline-testaustiimin kanssa hyvin.

- Projektipäällikkö: Mainline-testaustiimin kanssa yhteistyössä on hiomista. Mainline-testauksessa tulisi tehdä enemmän projekteissa esiintyviä testitapauksia.

- Mainline-testauksen kehitysvastaava: Työskentelee Mainline-testaustiimissä.

- .Net-tiiminvetäjä: Ei ole yhteydessä Mainline-testaustiimiin.

- CMS-tiiminvetäjä: Toiminnassa ei ole valittamissa.

9.6 Mitä hyvää tai huonoa havaitset omassa tiimissä tai Mainline-testauksessa?

- Testaaja A: Positiivisena asiana voidaan todeta tiimin ilmapiiri ja se, että saa apua, kun tarvitsee.

- Testaaja B: Positiivisena asiana Mainline-testauksessa on systemaattisuus ja uusi tapa kierrättää testitapauksia. Testaaminen on muutenkin tehty helpommaksi ja automatisointia on lisätty. Negatiivista sanottavaa Mainline-testauksesta sai testitapauksien kuvaukset, jotka eivät ole tarpeeksi laadukkaat.

- Testaaja C: Positiivisena asiana näkee toiminnassa ilmapiiriin, esimiehen ja työtoverit. Negatiivisena asiana näkee toiminnassa dokumentaation, joka on puutteellista ja ennen kaikkea englanninkielisten dokumenttien vähäisyys.

- Testaaja D: Positiivisena asiana Mainline-testauksessa on esimies ja käytössä olevat työkalut ja negatiivisena asiana näkee esimiehen liian suuren työmäärän.

- Testaaja E: Positiivista Mainline-testauksessa on kokonaiskuva, versionhallinta ja virheiden suodatin. Negatiivista on pitkät testaus syklit ja toiminnan jäykkyys.

- Ohjelmistokehityksen vetäjä: Negatiivisena asiana näkee perustestaus toiminnan liiallisen työvoiman käyttämisen ja toisaalta taas vaativimpien asioiden testaamisessa vaaditaan enemmän testaajia. Kaikkia mahdollisia asioita ei vieläkään ole testattu.

- Mainline-testauksen vetäjä: Positiivisena asiana näkee uusien testitapausten syntymisen ja negatiivisena liian vähäisen automaatiotestien määrän.

- Projekti-insinööri: Positiivisena asiana näkee Mainline-testauksessa työnjaon, joka on tehty hyvin ja selkeästi.

- Asiakaspalveluinsinööri: Positiivisena asiana esiintyy testitapausten kierrättäminen.

- Projektipäällikkö: Positiivisena asiana nousee esille toimiva kokonaisuus, sillä Mainline-testauksella on toimiva konsepti. Negatiivisena asiana nousee esille tiettyjen testitapausten puuttuminen kokonaan ja testikierroksen kesto on liian pitkä.

- Mainline-testauksen kehitysvastaava: Positiivisena asiana esille nousi yleisilme ja toiminta, joka on hyvää ja koordinoitua. Toisena asiana nousevat esille laajat testikirjastot. Negatiivisena asiana on se, että toiminta ei näytä kovinkaan hyvältä ulkopuolisille toimijoille. Testitapausten kuvaukset ovat heikot ja projektitiimit määrittelevät testaajien aikatauluja liikaa.

- .Net-tiiminvetäjä: Positiivisena asiana voi sanoa sen, että toiminta toimiva ja negatiivisena sen, että se ei näy muualla yrityksessä.

- CMS-tiiminvetäjä: Positiivista Mainline-testauksessa on se, että on yksi alusta, jota testataan. Negatiivisena asiana voidaan nähdä toiminnan jäykkyys.

9.7 Mitä parantaisit tai toivoisit parannusta?

- Testaaja A: Kehittämisen varaa olisi testitapausten kierrättämisellä.

- Testaaja B: Testitapausten kuvauksia tulisi korjata tai parantaa tarpeen tullen. Testitapaukset tulisivat olla enemmän realistisempia tai ainakin ympäristö, jossa testitapauksia suoritetaan. Testitapausten automatisointi työkalu ei ole tarpeeksi laadukas. Työkalua tulisi parantaa tai sitten katsoa markkinoilta parempaa työkalua.

- Testaaja C: Ainoana parannusehdotuksena tuli pitkän pohdinnan jälkeen testitapausten automatisoinnin lisääminen.

- Testaaja D: Suoranaisia isompia parannusehdotuksia ei tullut esille.

- Testaaja E: Testitapausten kuvauksiin ja dokumentointiin tulisi panostaa enemmän testitapausten luontivaiheessa.

- Ohjelmistokehityksen vetäjä: Testitapausten automatisointia tulisi lisätä, sillä automatisointi lisäisi testitapausten suoritusten nopeutumista.

- Mainline-testauksen vetäjä: Testitapausten automatisointia tulisi lisätä runsaasti.

- Projekti-insinööri: Testitapausten dokumentaatiota tulisi parantaa huomattavasti ja uuden työkalun hankinta olisi paikallaan, jossa projektitiimit voisivat suoraan laittaa uusia ominaisuuksia ylös, joita tulisi testata Mainline-testauksessa.

- Asiakaspalveluinsinööri: Mainline-testaustiimin toiminnassa ei löytynyt mitään suurempia parannuksen kohtia.

- Projektipäällikkö: Mainline-testaukseen tulisi saada enemmän isompia ja realistisia asiakaslähtöisiä testitapauksia.

- Mainline-testauksen kehitysvastaava: Keskustelimme pitkään erilaisista tapauksista ja totesimme, että jo esille tulleet asiat ovat olleet myös haastateltavan henkilön mielessä. Ensimmäisenä ja suurimpana parannusehdotuksena esille nousi testien automatisoinnin lisääminen, joka on ollut vähäistä testaajien tiukan aikataulun ja testauksen automatisointi työkalun takia. Toisena tuli testitapausten kuvausten ja dokumentaation parantaminen. Testitapauksien dokumentaatiossa on puutteita. Yksikkötestauksen lisääminen nousi uutena asiana esille, jota voitaisiin alkaa tehdä Mainline-testauksessa. Neljäntenä asiana nousi esille projektitiimin toiminnan tuominen mahdollisimman lähelle Mainline-testausta. Viidentenä nousi esille Mainline-testauksessa käytettävät ohjelmistot. Ovatko ohjelmistot tarpeeksi hyvät Mainline-testauksessa

käytettäväksi? Kuudentena asiana esille nousi työntekijöiden määrä Mainline-testauksessa, joka työmäärään nähden melko suuri.

- .Net-tiiminvetäjä: Aluksi ei tullut esille parannusehdotuksia, mutta pienen keskustelu tuokion jälkeen esille nousi mahdollisuus, että Mainline-testauksessa voitaisiin tehdä yksikkötestejä.

- CMS-tiiminvetäjä: Ensiksi olisi tärkeintä testata ainoastaan olennaisia testitapauksia. Toinen parannusehdotus on saada Mainline-testaajille parempi kuva, siitä mitä testaavat, jotta voivat ratkaista ongelman jo puoliksi ennen kuin ongelma saapuu ohjelmoijille. Yksikkötestauksen käyttöönotto olisi hyvä asia myös.

9.8 Yhteenveto

Karkeasti jaettuna haastateltava joukko oli testaajia, projektihenkilöitä ja ohjelmoijia. Ohjelmoijat olivat selkeästi tekemisissä Mainline-testauksen kanssa erilaisten vikailmoitus kanssa. Ohjelmoijat pitivät tärkeänä Mainline-testausta ja sitä, että kuinka paljon tulisi panostaa Mainline-testaukseen. Mainline-testaus oli ohjelmoijien mielestä yrityksen tärkein tai ainakin yksi tärkeimmistä tekijöistä.

Projektihenkilöt olivat Mainline-testauksen kanssa tekemisissä erilaisten asiakasprojektien takia. Projektipuolella saatiin mielenkiintoisia näkemyksiä, sillä toiminta on hyvin organisoitua, mutta toiminnan sisällössä olisi parantamisen varaa. Projektitiimit toivoisivat parannusta toimintaan, jotka pitävät sisällään projektitiimien asioita.

Testaajat olivat itse todella tyytyväisiä omaan toimintaan isommassa kokonaiskuvassa, mutta aina löytyy pieniä parannuskohteita. Parannuskohteita oli laitteistossa ja toimintatavoissa. Testaajat kokivat yhteistyön muiden yrityksen sidosryhmien kanssa suurimmaksi osaksi hyväksi asiaksi tai ainakin toimivaksi kokonaisuudeksi.

10 Mainline-testauksen vahvuudet ja heikkoudet

Vahvuuksien ja heikkouksien löytämiseen käytettiin kvalitatiivista haastattelua. Vahvuuksia ja heikkouksia varten sovelsin pros-cons -menetelmää. Pros-cons –menetelmä tarkoittaa vahvuuksien ja heikkouksien löytämistä tietystä aiheesta. Aihe voi olla mikä tahansa, josta saa etsittyä asiaa puoltavan kuvan ja vastaväitteen. (Smith 2009.)

Vahvuudet	**Väite**	**Vahvistaminen**	**Heikkoudet**	**Väite**	**vähentäminen**
Ilmapiiri	Tiimin jäsenet auttavat ja tukevat toisiaan	Testaustiimin yhteistyön säilyttäminen	Testitapausten kuvaukset ja laatu	Testitapausten kuvaukset ovat puutteelliset tai tasoltaan heikot	Testitapausten korjaaminen ja huolellisempi suunnittelu
Systemaattisuus	Toiminta on hyvin organisoitua ja tarkasti valvottua	Testaustiimin tehtävien jaon ja seurannan tarkka toteuttaminen	Testikierrosten pituus	Testikierrokset kestävät tällä hetkellä liian kauan	Testauksen toiminnan kehittäminen ja testien automatisointi
Testien kierrättäminen	Tapa, jossa testitapausten paketit kiertävät testaajalta toiselle	Sopivien testitapausten ryhmittely ja kierrätyksen järjestelmällisyys	Automaatiotestauksen vähäisyys	Testaajien aikataulu on kiireellinen ja automaatiotestaus työkalulla ei voida kaikkea automatisoida	Työntekijöiden lisääminen ja automatisointi työkalun kehittäminen
Osallistuminen projektityöhön	Vahvistaa asiakkaan järjestelmän asiantuntijuutta	Mainline-testaus saa uusia testitapauksia testaukseen	Osallistuminen projektityöhön	Testaajien aikaa kuluu erilaisten projektien kanssa runsaasti	Työntekijöiden lisääminen ja testaajien jakaminen niin, että toiset keskittyvät Mainline-testaukseen ja toiset projekteihin

Taulukko 1. Vahvuudet ja heikkoudet

11 Parannusehdotukset

Tutkimuksen kautta todettiin, että näitä asioita voitaisiin parantaa Mainline-testauksessa. Parannukset ovat sekä lyhyelle, että pitkälle aikavälille suunniteltuja. Toiset parannukset ovat helpommin ja toiset vaikeammin toteutettavissa. Kappaleessa käsitellään parannusehdotuksia painoarvon mukaan. Ensimmäisenä tulee tärkein ja seuraavana tulee toiseksi tärkein parannusehdotus.

Parannusehdotus	Kohteet	Tila	Toimet
1. Testitapausten automatisoinnin lisääminen	Testaajien ajan käytön suunnittelu	Testaajien työmäärä on suuri ja testitapausten suunnittelulle ei jää aikaa	Testaajien ajan suunnittelulle tulee tehdä aikaa
1. Testitapausten automatisoinnin lisääminen	Testitapausten automatisointi työkalu(AutoTester)	Testitapausten automatisointi työkalun funktioiden vajaa toiminnallisuus	Testitapausten automatisointi työkalun funktioiden tutkiminen ja kehittäminen
2. Testitapausten kuvauksien ja dokumentaation parantaminen	Testitapaukset	Testitapausten kuvaukset ja dokumentaatiot ovat heikot tai puutteelliset	Testitapausten kuvauksien korjaaminen laadukkaammiksi ja testaaja kollega tarkastaisi kollegan luoman testitapauksen
3. Moduulitestauksen eli yksikkötestauksen käyttöönotto	Moduulitestit	Moduulitestien määrä on vähäinen	Moduulitestausta tulisi aloittaa tekemään Mainline-testauksessa
4. Projektitiimien ja Mainline-testaustiimin yhteistyön tiivistäminen	Testitapaukset	Tällä hetkellä Mainline-testauksessa ei testata tarpeeksi laajoja ja realistisia testitapauksia	Asiakasprojektien realististen ja ajankohtaisten testitapausten kuvaaminen Mainline-testaukseen
5. Uusien työkalujen hankinta	Mainline-testauksessa käytettävät työkalut	Tällä hetkellä ollaan melko tyytyväisiä työkaluihin, mutta automaatiotestauksen työkalu voisi olla parempi	Testitapausten automatisointi työkalun kehittäminen, mahdollisten uusien työkalujen hankinta
6. Työntekijöiden lisääminen	Työntekijät	Tällä hetkellä Mainline-testauksen työntekijöillä on runsaasti töitä ja testaamisen kehitystyöt kärsivät siitä	Uusien Mainline-testaajien palkkaaminen

Taulukko 2. Parannusehdotukset

11.1 Testien automatisoinnin lisääminen

Tällä hetkellä Mainline testauksessa ei luoda, eikä käytetä testauksen automatisointia niin paljon, kuin toivottaisiin. Automatisointi, joka nopeuttaisi testaajan työtä huomattavasti. Tämä ehkäisi myös väärien testaustapojen mahdollisuuden. Syy vähäiseen automaatiotestaukseen johtuu useammasta eri syystä. Ensimmäisenä voidaan pitää testaajien aikataulua, eli ei ole aikaa tehdä automaatiotestejä. Tämä ei kuitenkaan ole koko totuus, sillä testaajat voisivat saada enemmän automaatiotestejä luotua, mikäli suunnittelisivat aikataulunsa paremmin. Tähän kohtaan varmasti löytyy ratkaisu testaajasta itsestään ja jos apua aikataulun suunnittelemiseen tarvitaan niin, sitä varmasti esimies tarjoaa. Testejä joutuu tällä hetkellä ajamaan käsin ja tämä vie paljon aikaa testaajalta, joka taas aiheuttaa sen, että on vähemmän aikaa luoda automaatiotestejä ja kehittää Mainline testauksen toimintaa. Tilannetta voisi varmasti helpottaa uusien testaajien palkkaaminen, jotta työmäärää voitaisiin vähentää nykyisiltä testaajilta. Toinen syy on automaatiotestauksessa käytettävä ohjelma. AutoTester vaatisi muutoksia testattavan ohjelmiston koodiin ja näin AutoTester-ohjelmalla ei voida tällä hetkellä toteuttaa kaikkia haluttuja toimintoja. Kolmantena syynä ovat projektitiimit, joiden asiakasprojektien testaus ajaa priorisoinnissa normaalien Mainline testauksien ohi. Tämä vie testaajien automaatiotestien suoritusajasta pois paljon aikaa. Syy automaatiotestauksen vähäisyyteen on ohjelmistossa ja työntekijöissä.

Tällä hetkellä automaatiotestien tekemiseen käytetään niin paljon aikaa, kuin testaajalla on testien ajojen ja erilaisten projektien jälkeen. Automatisointia varten on luotu AutoTester ohjelma, joka pitää sisällään kirjaston erilaisia komentoja, joita hyödynnetään esimerkiksi PSPad:lla, jolla voidaan kirjoittaa automaatiotestejä AutoTesterin kirjaston tukemista komennoista. AutoTester-ohjelma on luotu sisäisesti kohdeyrityksessä ja se on luotu testaamaan tietokantoja, kuin käyttöliittymää. Erilaiset kaupalliset tuotteet, eivät ole välttämättä yhteensopivia yrityksen muiden tuotteiden kanssa. Erilaisia tuotteita on tutkittu markkinoilta. PSPad:lla luodaan itse automaatiotesti AutoTesterin komentojen pohjalta. Valmis automaatiotesti liitetään TestTrack-ohjelmaan, josta sitä ajetaan testauksessa. PSPad ja TestTrack ovat ulkopuolisen toimittajan luomia ohjelmia.

Automaatiotestien luomiseen kuluu aikaa testaajalta määrittelemätön määrä aikaa. Eli tarkkaa keskiarvoa ei voida sanoa, sillä kaikkia testejä ei voida automatisoida ja osa testeistä on toisia testejä monimutkaisempia. Automaatiotestauksen parantamistoimenpide voidaan toteuttaa niin, että AutoTesterin -komennoista ja sen toiminnasta osaava henkilö alkaa laajentaa ja muokkaamaan kirjastoa, jossa on valmiit komennot.

11.1.1 Mainline-testaajien ajankäyttö

Mainline-testaustiimissä tiimin jäsenten ajankäytön suunnittelu ei ole toteutettu tarpeeksi hyvin. Tämä johtaa siihen, että automaatiotestaus on vähäistä siihen nähden, mitä se voisi olla. Testaajilla on mahdollisuus suunnitella automaatiotestit niin, että nykyisillä funktiolla saadaan osa automaatiotesteistä toimimaan. On olemassa automaatiotestejä, joita voidaan automatisoida, mikäli testin suunnitteluun käytettäisiin enemmän aikaa. Tämä tapa vaatii testaajalta suunnittelua ja testaamista, jotta voidaan todeta, että funktioiden muokkaaminen tai yhdistäminen tuottaa sellaisen tuloksen, jota testitapaukselta halutaan.

11.1.2 Autotester–ohjelman puutteellisuudet

AutoTester–ohjelman funktiot eli toiminnot ovat puutteellisia. Funktiot, eivät vastaa niin laajasti toiminnallisuutta, jota Generiksessä halutaan testata. Tämän takia osa suunnitelluista testeistä ei voida automatisoida tai sitten voidaan automatisoida osittain. Tällä hetkellä on tapauksia, jossa automaatiotestejä toteutetaan niin, että yhtä testitapausta varten on useampia automaatiotestejä, jotta saadaan haluttu lopputulos.

Esimerkki puutteellisuudesta on ikkunassa olevien välilehtien alivälehtien käyttö. Eli jos testitapauksessa halutaan testi automatisoida ja käyttää ikkunassa olevien välilehtien alivälilehtiä. Syy tällaisen toiminnallisuuden puutteellisuuteen on AutoTester–ohjelman funktioissa, mutta on myös mahdollista, että muutos koodissa Generiksessä. Tällaisen toiminnallisuuden korjaaminen vaatii siis Generiksen tutkimista ja tämän jälkeen mahdollisesti AutoTester funktioiden korjaamista. Toinen ongelma on sellainen, jossa Generiksessä avataan ikkuna, jossa halutaan muokata tietoja, jonka jälkeen tallentaa muutokset ja sulkea ikkuna. Eli tallenna tietoa ja sulje ikkuna automaattisesti ei onnistu. Kolmantena esimerkki ongelmana on Generiksen ikkunan sisällä avautuvan toisen ikkunan kontrollointi. Tämä tarkoittaa, sitä että avataan ikkuna Generiksessä ja kyseisessä ikkunassa avataan toinen ikkuna ja tässä ikkunassa ei voida tehdä mitään. Korjaustoimenpiteen kestoa on vaikea analysoida, koska tarvittaisiin aikaa asiantuntevalle henkilölle, joka tutkisi Generistä ja tämän jälkeen tekisi AutoTesteriin korjatut funktiot. Kesto voisi olla päivistä – viikkoihin, mikäli asiantunteva henkilö saataisiin tutkimaan rauhassa asiaa. Tällä hetkellä ongelmat, eivät ole vakavia, mutta tiettyjen testien rakentaminen ei onnistu.

AutoTesteriä on aloitettu muokkaamaan ja muokkauksia on tehnyt Jyväskylän toimipisteen testaustiimin jäsen. Ensimmäinen parannusominaisuus on saatu valmiiksi. Parannusominaisuus on AutoTester-funktio, jolla voidaan käynnistää IFM-jobeja. Suurin osa vanhemmista automaatiotesteistä on muokattu vastaamaan uuttaa päivitystä. Lisäksi kohdeyritys on suunnitellut ohjelman AutoTester.BatchRunner, jolla voidaan ajaa useampia IFM-jobeja yhtä aikaan. Vielä ei ole kuitenkaan integroitu TestTrack-ohjelmaan, mikä on tarkoitus toteuttaa.

11.2 Testitapausten kuvauksien ja dokumentaation parantaminen

Testien dokumentaatiolla tai kuvauksilla tarkoitetaan kaikkea mahdollista mitä testin luoja voi ohjeistaa tai auttaa materiaalin kautta testin suorittajaa, kuten missä, miten ja milloin. Testien kuvaukset ja kommentit luodaan TestTrack-ohjelmaan, joka on ainoa paikka mistä näitä tietoja voi muokata tai katsella. Testitapauksissa voi olla erilaisia liitetiedostoja, kuten kuvankaappauksia, dokumentteja tai scriptejä, nämä tiedostot tullaan tallentamaan yrityksen versionhallintaohjelmaan Surroundiin. Kohdeyrityksessä on luotu esimerkkimalli, jonka pohjalta testitapaukset tulisi muodostaa. Yrityksen esimerkkimallia luoda testejä tulisi jäljitellä mahdollisimman hyvin, jotta testin suorittaja tietää, mistä etsiä tarvittavat tiedot testin suorittamiseen. Esimerkiksi missä ympäristössä testi tulee suorittaa ja millaisia asioita tulee ottaa huomioon ennen testitapauksen suorittamista.

Tällä hetkellä on useita testejä joiden kuvaukset ja dokumentaatiot ovat heikot. Varsinkin uuden testaajan on erittäin vaikea saada selville testin suorittamisesta tai siitä mitä tullaan testaamaan. Syy heikkoihin testitapausten kuvauksiin on selkeä, testin tekijä on luonut testin kuvaukset suurin piirtein omiksi muistiinpanoiksi, sillä ennen kohdeyrityksessä toimittiin niin, että testaaja teki testitapauksen ja suoritti ainoastaan omia testitapauksia. Tämä vanha tapa aiheutti tilanteen, jossa testaaja siirtyisi muihin tehtäviin tai sairastuisi ja tätä myöten testitapauksen asiantuntemus katoaisi. Testitapauksen luontivaiheessa ei ole panostettu tarpeeksi testin kuvauksiin ja dokumentaatioon. Uuden tavan myötä testejä kierrätetään niin, että jokainen testaaja saa mahdollisimman paljon eri testejä testi syklissä. Testitapauksen tulisi olla niin selkeä, että kuka vain, joka osaa edes suurin piirtein ohjelmistoa käyttää, tietää mitä tehdä.

Testien dokumentaatiota voidaan parannella ja pitääkin parannella mikäli testin suorittaja huomaa puutteellisuuksia. Tämä tulee testien suorituksen aikana viemään hieman enemmän aikaa, sillä testin suorittamisen lisäksi testaajalle tulee testin dokumentaation ja kuvauksen parantaminen, mikäli epäselvyyksiin törmätään. Olemme suunnitelleet mahdollisuutta, jossa tulevaisuudessa toinen testaaja tarkistaisi toisen testaajan luoman testitapauksen välittömästi testitapauksen luonnin jälkeen niin, että testitapaus on helposti ymmärrettävissä ja kaikki olennainen tieto on saatavilla. Testaaja on siis koko parannustoimesta vastuussa. Testitapauksen itse luontivaiheessa laadukkaan testikuvauksen ja dokumentaation tekeminen ei luo lisätyötä tai vie testaajan aikaa.

Tavoite on, että muutamalla muutoksella ja asian tärkeyden painottamisella saamme luotua hyviä ja laadukkaita testejä kuvauksineen. Tällainen operaatio tulee kuitenkin viemään aikaa ja, sitä myöten kärsivällisyyttä vaaditaan testin osanottajilta, jotta parannus olisi mahdollinen. Tämä on erittäin tärkeä asia testien automatisointia silmällä pitäen. Testien kuvaukset,

dokumentaatio ja mahdolliset automaatiotestit tulee muokata muotoon, jossa niitä voidaan tulevaisuudessa automatisoida sellaisiksi, että saadaan useita testejä kerrallaan suoritettua. Tämä tulee olemaan mahdollista, mikäli automaatiotestauksen AutoTester-työkalua saadaan muokattua haluttuun suuntaan.

11.3 Moduulitestaus eli yksikkötestaus käyttöönotto

Moduulitestaus eli yksikkötestaus tarkoittaa ohjelman koodin testaamista. Ohjelmoijat luovat yksikkötestejä luodun ohjelmakoodin funktiota varten. Yksikkötestit ajetaan ja tutkitaan, kuinka ohjelmakoodin funktiot toimivat. Tällä hetkellä yrityksessä moduulitestausta tehdään kyllä, mutta se on vähäistä. Moduulitestaus on täysin riskitöntä toimintaa, sillä siinä ei itse asiassa itse ohjelmaa testata, vaan luodun koodin toiminnallisuutta. Kohdeyrityksen CMS-tiimi tekee aktiivisesti moduulitestausta, mutta kokonaisuudessa testaus on melko vähäistä.

Moduulitestin tekeminen vie henkilöltä aikaa tunneista-päiviin. Testien tekemisen kesto riippuu testattavan koodin koosta. Testaus kuitenkin selkeyttää koodia. Tavoite yksikkötestauksella on saada virheettömämpää koodia Mainline-testaukseen. Tämän moduulitestauksen voisi toteuttaa niin, että toinen testien tekijä keskittyisi alempiin toimintoihin ja toinen itse käyttöliittymäpuolen testien tekemiseen.

Moduulitestauksen hyöty on siis suuri, sillä virheiden ennalta ehkäiseminen säästää aikaa Mainline-testaajilta ja yritykseltä rahaa. Ohjelmoijat voivat itse suorittaa moduulitestejä tai sitten testejä suorittaa erillinen henkilö, joka on nimitetty moduulitestaukseen.

Asiaa on hieman esisuunniteltu ja yrityksen sisältä löytyy osaavia henkilöitä näihin tehtäviin, mutta näiden henkilöiden tilalle joudutaan mahdollisesti palkkaamaan 1-2 uutta työntekijää. Uudet työntekijät korvaisivat toisiin tehtäviin siirtyneitä henkilöitä. Asia on menossa eteenpäin, mutta hitaasti, sillä tällä parannuksella olisi tarvetta, mutta hyötyä on vielä vaikea osoittaa numeroiden kautta. Tieto on kuitenkin siitä, että virheen löytyminen asiakkaalta on paljon kalliimpaa kohdeyritykselle, kuin panostaminen moduulitestaamiseen.

Tällä hetkellä yrityksessä on tehty muutoksia ja yksi Mainline-testaustiimin jäsenistä on siirtynyt yksikkötestauksen pariin. Hän toimii niin Mainline-testaustiimissä, kuin yksikkötestauksen parissa. Hänen tilalleen ei ole palkattu uutta tai uusia työntekijöitä.

Parannustoimessa huomioon tulisi ottaa asioita, joita tulisi tehdä moduulitestaukseen siirryttäessä. Ensimmäisenä tulisi asentaa tuotantotietokannat testipalvelimelle, jonka jälkeen testaaja voisi aloittaa koodin korjaamisen erillisellä palvelimella. Korjaamisella tarkoitan erilaisten testien luomista ja testaamista, jotta koodi toimii niin, kuin on tarkoitus ja koodista tulee

samalla myös selkeämpää. Tilanne on vielä tällä hetkellä aivan alkumetreillä, mutta muutoksia yksikkötestausta kohtaan ja toiminnan parannusta kohtaan on otettu.

11.4 Projektitiimien ja Mainline-testaustiimin yhteistyön tiivistäminen

Projektitiimeillä tarkoitetaan yrityksessä työskenteleviä henkilöitä, jotka palvelevat asiakkaita, joille yrityksen tuotteita myydään. Projektitiimit kohtaavat, kuitenkin ongelmia asiakkaiden tarpeiden kanssa ja joitakin toimintoja ei tiedetä toimivatko oikein tai onko syytä toimia tietyllä tavalla ja tämän takia projektitiimit tuovat ongelmansa testattavaksi Mainline-testaustiimin jäsenille. Mainline-testaustiimin tuotantoversio ei kuitenkaan suoranaisesti vastaa asiakkaalla olevaa mahdollisesti vanhempaa versiota ja tämä tuottaa runsaasti työtä Mainline-testaustiimin jäsenille, koska testausta joudutaan tekemään testipalvelimella vanhemmalle versiolle. Tämän lisäksi joudutaan mahdollisesti myös Mainline-versiolle tekemään testausta, jotta nähdään toimiiko ominaisuus Mainline-versiossa, josta asiakas on ilmoittanut projektitiimeille.

Mainline-testaustiimi ja projektitiimit tekevät yhteistyötä niin paljon, kuin mahdollista ja suurimmaksi osaksi tarve yhteistyölle syntyy projektitiimien myötä. Keskimäärin kolmasosa Mainline-testaajan päivästä menee projektitiimien projektin kanssa, jossa testaaja antaa tukea testaamalla ja analysoimalla projektitiimin ongelmia. Tämä on Mainline-testaustiimin silmistä katsottuna melko paljon arvokasta työaikaa, joka kuluisi normaalisti oman tiimin tehtävien kanssa. Projektitiimien tarve ohittaa Mainline-testaustiimin omien arkisten tehtävien tarpeet. Projektitiimillä on maksavan asiakkaan kanssa sopimus tuotteen toimittamisesta ja sen tuesta. Tämän takia olisi hyvä saada mahdollisimman nopea ongelman selvitys. Mitä lähemmäksi projektitiimien ongelmat saataisiin kuvattua Mainline-testaustiimille niin, sitä parempi, koska ongelmia voitaisiin ratkoa hyvissä ajoin ja tehokkaasti uutta asiakasta tai tuotetta silmällä pitäen. Tämä palvelisi projektitiimiä ja Mainline-testaustiimiä. Uusien projektien kanssa voidaan tehdä niin, että tuodaan Mainline testaukseen kaikki sellaiset testitapaukset, jotka tulisi olla asiakkaalla kunnossa ja toimia oikein. Vanhemmat versiot, joita asiakkaat käyttävät ovat erilaisia verrattuna Mainline versioon, tämän takia ei voida suoraan verrata tai yhdistää asiakkaan versiota ja Mainline versiota keskenään.

Yksi ajatus on rauhoittaa Mainline-testaustiimin toimintaa, olisi jakaa testaajat kahteen ryhmään eli toiset olisivat rauhoitettu täysin testaamiseen ja toinen osa, jota voitaisiin hyödyntää eri projekteissa. Tämä tapa olisi todella hyvä, sillä saataisiin selkeyttä siihen kuka tai ketkä testaajista on missäkin projektissa ja ketä voidaan hyödyntää ja mihin. Toiminnan parhaina puolina ovat Mainline toiminnan kehittämisen mahdollisuus ja projektitiimeillä olisi valmiiksi tietyt testaajat, joita voisivat hyödyntää erilaisissa projekteissa.

Ongelmia havaitessa asiakkaan versiossa korjataan ongelmat asiakkaan versioon. Asiakkaan versiossa havaittu ongelma korjataan myös Mainline-versioon, jos sellaisia havaitaan Mainline-versiossa. Mainline-versiossa havaitut ongelmat, eivät kuitenkaan päivitetä asiakkaan versioon vaan asiakas saa aikanaan uuden version mukana korjatun version.

11.5 Uusien työkalujen hankinta

Tällä hetkellä Mainline-testaustiimissä käytetään versionhallinnassa ohjelmaa nimeltä Surround. Itse testien luomiseen ja seurantaan käytetään ohjelmaa nimeltä TestTrack. Testien automatisointiin käytetään ohjelmaa nimeltä AutoTester, joka on yrityksen oma automatisointia varten luotu ohjelma. PSPad-ohjelmaa hyödynnetään Mainline-testauksessa automaatiotestien tekemiseen. Jira-ohjelmaa hyödynnetään erilaisten vika ja ongelma tapausten raportointiin.

TestTrack ja Surround ovat ohjelmia, jotka ovat ihan hyviä tai ohjelmista ei löydy huonoja kohtia. Ohjelmat ovat vakaita ja suorittavat hyvin ohjelmille annetut tehtävät. PSPad-ohjelma on Mainline-testaajien käytössä tarpeeksi laadukas käyttötarkoitukseen nähden. PSPad palvelee Mainline-testaajan tarpeita automaatiotestejä luodessa tai niitä suorittaessa.

AutoTester on yrityksen luoma ohjelma, jota käytetään, kun automatisoidaan testejä. AutoTester sovellusta hyödyntäen ei tällä hetkellä pystytä automatisoimaan kaikkia mahdollisia testitapauksia. Tällä tarkoitetaan erilaisten funktioiden rajoittunutta käyttömahdollisuutta. Ohjelma on kuitenkin yritykselle tärkeä, koska on alusta loppuun asti kohdeyrityksen luoma ohjelma. Markkinoilla varmasti on maksullisia sovelluksia, jotka pystyisivät parempiin ratkaisuihin tai toisenlaisiin ratkaisuihin, mutta oman sovelluksen etuna on muunneltavuus ja yhteensopivuus yrityksen toiminnan ja tuotteiden kanssa.

Jira on ohjelma, jota käytetään yrityksessä vika ja ongelma raportointiin. Ohjelma on monipuolinen ja sillä voi tehdä monenlaisia raportteja tai seuranta tietoja. Ohjelma on hyvä ja toimii hyvin Mainline-testauksen vikaraportointiin, mutta ohjelma sisältää lukuisia ominaisuuksia ja tämä saattaa sekoittaa käyttäjän. Tämä johtaa siihen, että jotain tärkeää tietoa viasta tai ongelmasta voi jäädä puuttumaan. Ohjelmoijat haluavat varsin tarkat kuvaukset ongelmista ja vioista, jotta pystyvät ratkomaan ongelman tai vian. Yksi syy siihen, että vikoja tai ongelmia ei pystytä ratkomaan on vikaraportoinnin puutteelliset tai väärät tiedot.

Uutena ohjelmana käyttöön ollaan ottamassa uutta itse kehittelemää ohjelmaa nimeltä AutoTester.BatchRunner, jolla voidaan suorittaa useita eri automatisoituja testitapauksia yhtäaikaisesti, toisistaan täysin riippumatta. AutoTester.BatchRunner tullaan integroimaan TestTrack -ohjelmaan, jonka jälkeen AutoTester.BatchRunnerilla voidaan ajaa useita testitapauksia

samanaikaisesti ja tulokset saadaan TestTrack-ohjelmaan. Testejä voidaan suorittaa yöllä, joka helpottaisi testaajien työtä.

11.6 Työntekijöiden lisääminen

Tällä hetkellä Mainline-testaustiimissä työskentelee viisi vakituista työntekijää mukaan luettuna tiiminvetäjä. Yksi vakituisista työntekijöistä työskentelee Jyväskylässä. Tiimissä on myös kolme osa-aikaista. Ensimmäinen osa-aikainen on osaksi töissä energiayhtiö Fortumilla, toinen osa-aikaisista tekee diplomityötä ja kolmas osa-aikainen, joka suorittaa työssäharjoittelua ja opinnäytetyötä Mainline-testaustiimissä.

Tiimien henkilöiden työmäärä on suuri. Testaajan oman tiimin testitapausten suorittamisen ja luomisten lisäksi testaaja voi olla monessa eri projektissa mukana. Projektien määrää ei tällä hetkellä voida kontrolloida. Mikäli projektitiimit haluavat testaajan ammattitaitoa tai apua asiakasprojektissa, silloin hänen oma työnsä jää vähäiseksi tai siirtyy myöhempään ajan kohtaan.

Ainoa ratkaisu olisi palkata Mainline-testaustiimiin 2-4 uutta testaajaa. Testaajia ei tulisi palkata yhdellä kertaa, koska Mainline-testaustiimillä ei ole resursseja kouluttaa kaikkia yhtä aikaa. Uusien testaajien palkkaaminen tulisi tapahtumaan pidemmällä aika välillä, mutta tärkein asia on saada uusia testaajia niin, että jatkossakin yrityksen ohjelmisto tulee olemaan huolella testattua ja laatu korkealaatuista.

Uusien testaajien palkkauksessa tulee harrastaa porrastamista. Porrastamisella tarkoitan uusien työntekijöiden palkkaamisessa. sitä että palkataan työntekijä kerrallaan. Syy tähän on testaajan kouluttamisessa ja perehdyttämisessä. Uusi työntekijä täytyy saada ensiksi kunnon perehdytyksen Mainline-testauksessa käytettäviin työkaluihin ja testauksen kohteeseen Generikseen. Tämän jälkeen uusi työntekijä täytyy kouluttaa ohjelmien ja ennen kaikkea Generiksen asiantuntijaksi.

12 Parannusehdotuksen tuoma tehokkuus Mainline-testaukseen

Mainline-testauksen viimeisessä testauskierroksessa syntyi 51 uutta virheilmoitusta, joista pystyttiin kohtuullisessa ajassa korjaamaan 22 vikailmoitusta. Jäljelle jäi 29 vikailmoitusta, jotka ovat vielä työn alla. Parannusehdotuksien myötä on havaittavissa, että testikierroksen kesto lyhentyisi ajallisesti puoleen nykyisestä, joka antaa mahdollisuuden ohjelmoijille luoda nopeammin vikailmoituksen korjauksia. Alla olevasta kuvasta näkyy, kuinka viime testikierros meni. Onnistuneesti saatiin 245 testitapausta suoritettua ja epäonnistuneita testitapauksia oli 53, joista saatiin 51 vikailmoitusta aikaan. Testitapauksista 21 jäi statukseen epäselvä eli ei voitu määritellä lopputulosta. Jokaisen testikierroksen jälkeen määritellään epäonnistuneiden

testitapausten vakavuus. Vakavan tai vakavien vikailmoitusten havaittaessa ohjelmistoa ei laiteta eteenpäin asiakkaalle.

State	Count	Percentage
Passed	245	76,80 %
Failed	53	16,61 %
Unclear	21	6,58 %
In Progress	0	0,00 %
Waiting	0	0,00 %
On Hold	0	0,00 %
Total	319	100,00 %

Taulukko 3. Testikierroksen yhteenveto

13 Yhteenveto

Toiminta Mainline-testaustiimissä on jäänyt hieman liian vähäiselle huomiolle ja tämän takia tulisi kohdeyrityksen yhdestä tärkeimmästä osasta saada parannusehdotuksia. Tutkimus ja taustatyö tuottivat tuloksen, että Mainline-testaustiimi on työntekijämäärältään alimitoitettu. Kohdeyrityksen asiakkaiden määrän ja tuotteen laajennuksen lisääntyessä työt kasvavat ja tämän takia tulisi palkata lisää työntekijöitä Mainline-testaustiimiin. Työmäärää voi työntekijöiden palkkauksen lisäksi pienentää erittäin tärkeällä testitapausten automatisoinnin lisäämisellä. Automatisointityökalun kirjasto on suppea eikä sillä voida luoda niin monipuolisia ja laajoja testitapauksia kuin toivottaisiin. Testien dokumentaation ja kuvausten heikkous on yksi parannusehdotuksen kohteista, sillä useimmat vanhemmat testitapaukset on luotu alun perin vanhaa testikierrosta tukemaan. Vanhassa testausmenetelmässä sama testaaja loi ja testasi testitapaukset. Uudessa testausmenetelmässä testitapaukset kiertävät testaajalta toiselle ja jokaisen täytyy osata ohjeiden mukaan suorittaa testit. Kohdeyrityksen tuotteen ja Mainline-testaustiimin tulevaisuutta silmällä pitäen tulisi myös Mainline-testaustiimissä panostaa yksikkötestaukseen, jossa yksi Mainline-testaustiimin jäsenistä alkaa tutkia ohjelmoijilta tulevaa koodia ja tekisi luodusta koodista testitapauksia ja testaisi koodin toimivuutta. Tämän lisäksi koodi olisi selkeämpää ja tulevaisuudessa helpommin käsiteltävää. Toinen kohdeyrityksen tulevaisuuden kannalta tärkeä asia olisi saada mahdollisimman realistisia ja ajankohtaisia testitapauksia Mainline-testaukseen. Tällaisia tapauksia voidaan saada projektitiimeiltä, jotka tekevät asiakkaiden kanssa tiivistä yhteistyötä. Realististen ja ajankohtaisten testitapausten pohjalta voidaan ennalta ehkäistä mahdollisia tulevia ongelmia asiakkaiden versioiden kanssa. Mainline-testauksessa käytettävät testaajien työkalujen toimivuudet tutkittiin ja todettiin suurimmaksi osaksi toimiviksi, mutta varauksella tutkittiin mahdollisuuksia siirtyä uusien ohjelmistojen käyttöön.

Lähteet

Painetut teokset:

Process Vision 2008. For a new employee. Administrative handbook.

Farrell-Vinay, P. 2008. Manage software testing. Auerbach Publications.

Dustin, E. 2003. Effective software testing. Pearson education inc.

Hirsjärvi, S., Remes, P. & Sajavaara, P. 2001. Tutki ja kirjoita. Helsinki: Tammi.

Sähköiset lähteet:

Kautto T. 1996. Ohjelmistotestaus ja siinä käytettävät työkalut. Viitattu 19.1.2009. http://www.mit.jyu.fi/opiskelu/seminaarit/ohjelmistotekniikka/testaus/

Seapine Software 2009. TestTrack Pro 2008. 21.1.2009. Viitattu http://www.seapine.com/ttpro.html

Kaner C. 2003. What Is a Good Test Case? Viitattu 4.5.2009. http://www.kaner.com/pdfs/GoodTest.pdf

Arguss C. & Johnson B. 2000. Ad Hoc Software Testing. Viitattu 4.5.2009. http://www.testingcraft.com/ad_hoc_testing.pdf

Software Business Compitence. Viitattu 27.1.2009. http://www.oamk.fi/sbc/testaus/testaustasot.htm

Hulttinen 2006. Mainline tiimin filosofia. Viitattu 28.1.2009. http://pvintranet/sites/intranet/mltest

Process Vision 2009. Generis. Viitattu 5.5.2009. http://www.processvision.fi/

PSPad 2009. PSPad. Viitattu 11.4.2009. http://www.pspad.com/

Taloustutkimus Oy 2007. Kvalitatiivinen. Viitattu 13.5.2009. http://www.taloustutkimus.fi/tuotteet_ja_palvelut/tiedonkeruuratkaisut_ja_monitila/kvalitatiivinen_tutkimus/

S.E. Smith 2009. Pros-cons. Viitattu 14.5.2009. http://www.wisegeek.com/what-are-pros-and-cons.htm

Jira 2009. Jira. Viitattu 14.5.2009. http://www.atlassian.com/software/jira/

Kuvaotsikkoluettelo

Kuvio-otsikkoluettelo

Taulukkoluettelo