TAMPEREEN
AMMATTIKORKEAKOULU

TESTAUSKÄYTÄNTÖJEN KEHITTÄMINEN LAUREAN TIETOJÄRJESTELMÄPROJEKTEISSA

Tuuli Rinne

Opinnäytetyö
Toukokuu 2016
Ylempi ammattikorkeakoulututkinto
Teknologiaosaamisen johtamisen
koulutusohjelma

TIIVISTELMÄ

Tampereen ammattikorkeakoulu
Teknologiaosaamisen johtamisen koulutusohjelma
Ylempi AMK-tutkinto

RINNE TUULI: Testauskäytäntöjen kehittäminen Laurean tietojärjestelmäprojekteissa

Opinnäytetyö 73 sivua, joista liitteitä 5 sivua
Toukokuu 2016

Tietojärjestelmän hyväksymistestaus ennen tietojärjestelmän tuotantokäyttöönottoa on yksi tärkeimpiä vaiheita käyttöönoton onnistumisessa. Tästä huolimatta tietojärjestelmän testaukselle varataan usein liian vähän aikaa sekä resursseja.

Tämän opinnäytetyön toimeksiantajana toimi monialainen Laurea-ammattikorkeakoulu, joka toimii usealla eri paikkakunnalla Uudellamaalla. Opinnäytetyön tarkoituksena oli selvittää Laurean tietojärjestelmäprojektien nykyisiä testauskäytäntöjä ja selvityksen pohjalta muodostaa kehittämiskohteita sekä ratkaisuehdotuksia testauksen kehittämiseksi. Tärkeimpänä tavoitteena oli muodostaa konkreettisia testauksen kehittämiseen liittyviä työvälineitä tietojärjestelmien pääkäyttäjille ja projektipäälliköille.

Tutkimusmenetelmänä käytettiin kvalitatiivista tutkimusta, joka toteutettiin teemahaastatteluilla. Haastateltaviksi valittiin yhdeksän tietojärjestelmien pääkäyttäjää ja kaksi projektipäällikköä. Haastatteluaineistoa analysoitiin käyttäen tematisointia. Tutkimuksen lopputuloksena syntyneet kehityskohteet ja ratkaisuehdotukset perustuivat haastatteluaineiston analysointiin sekä testauksen taustateorioihin.

Haastatteluaineisto ja kehityskohteet liittyivät neljään eri teemakokonaisuuteen: testauksen suunnittelu ja suorittaminen, testiympäristön käyttö testauksessa, uuden tietojärjestelmän tai ohjelmistoversion käyttöönotto, ja toimittajan suorittama testaus. Kuhunkin teemakokonaisuuteen havaittiin haastatteluaineiston analysoinnissa kehityskohteita, yhteensä kuusi kappaletta. Kehityskohteille muodostettiin yhteensä kahdeksan ratkaisuehdotusta. Opinnäytetyön tärkein tavoite, konkreettisten työvälineiden tuottaminen Laurean tietojärjestelmäprojekteille testauksen kehittämiseksi, onnistuttiin saavuttamaan muodostettujen ratkaisuehdotuksien kautta. Työvälineiden käyttöönotto ja jatkokehitys tapahtuvat erillisessä projektissa.

Asiasanat: testaus, prosessinkehitys, tietojärjestelmät

ABSTRACT

Tampereen ammattikorkeakoulu
Tampere University of Applied Sciences
Master's Degree in Strategic Leadership of Technology-Based Business

RINNE TUULI: Development of Testing Processes for Laurea IT-Projects

Master's Thesis 73 pages, appendices 5 pages
May 2016

Acceptance testing for IT-system is one of the most important phases when planning deployment of IT-system. Regardless of this, time and resources reserved for testing IT-system is usually too low.

The purpose of this study was to gather information about current testing practises in IT-system projects of Laurea University of Applied Sciences. The purpose was to find out areas which needed an improvement and to create solution proposals for those. The most important target was to create concrete tools for improving testing practises in IT-system projects.

This study was carried out as qualitative study using theme based interviews. The data were collected from nine IT-systems' main users and from two project managers. The data were analysed using theme based analysis method. Improvement areas found in analysis were related to following four themes: test planning and execution, usage of test environment in testing, deployment of new IT-system or new version of software and testing performed by the vendor.

Six improvement areas were found for these four themes. Altogether eight solution proposals were created for improvement areas. As a final result, based on solution proposals, you can say that targets of the study have been achieved. Implementation and further development of solution proposals will be carried out in a separate project.

Key words: testing, process development, it-systems

SISÄLLYS

1 JOHDANTO 7
2 TUTKIMUSASETELMA 9
2.1 Toimeksiantajan esittely 9
2.2 Tutkimuksen taustaa 10
2.3 Tutkimuksen tavoitteet ja rajaus 12
2.4 Tutkimuskysymykset 12
2.5 Tutkimusmenetelmä 13
2.5.1 Teemahaastattelu 13
3 OHJELMISTOTESTAUS 15
3.1 Mitä testaus on? 15
3.2 Testauksen suunnittelu 17
3.2.1 Asiakasvaatimukset 18
3.2.2 Käyttötapaukset ja käyttäjätarinat 21
3.2.3 Testitapaukset 23
3.3 Testiympäristön luonti 24
3.4 Testauksen suorittaminen 25
3.4.1 Staattinen ja dynaaminen testausmenetelmä 27
3.4.2 Musta- ja lasilaatikkotestaus 28
3.4.3 Virheiden tunnistaminen 29
3.5 Testituloksien raportointi 31
4 TESTAUSTASOT 32
4.1 Yksikkötestaus 34
4.2 Integraatiotestaus 35
4.3 Järjestelmätestaus 36
4.4 Hyväksymistestaus 38
4.4.1 Hyväksymistestauksen aloitus- ja lopetuskriteerit 39
4.4.2 Testaussuunnitelma 40
4.4.3 Käyttöönotto 42
5 TUTKIMUKSEN TOTEUTTAMINEN JA TULOKSET 44
5.1 Teemahaastattelut 44
5.2 Haastatteluaineiston litterointi ja analysointi 45
5.3 Tutkimuksessa havaitut kehityskohteet 46
5.3.1 Testauksen suunnittelu ja suorittaminen 46
5.3.2 Testiympäristön käyttö testauksessa 50
5.3.3 Uuden tietojärjestelmän tai ohjelmistoversion käyttöönotto 51
5.3.4 Toimittajan suorittama testaus 53

5.4 Ratkaisuehdotuksia kehityskohteisiin 55
5.4.1 Testauksen suunnittelu ja suorittaminen 55
5.4.2 Testiympäristön käyttö testauksessa 57
5.4.3 Uuden tietojärjestelmän tai ohjelmistoversion käyttöönotto..... 58
5.4.4 Toimittajan suorittama testaus 59
6 TUTKIMUKSEN LAATU JA LUOTETTAVUUS 60
6.1 Tutkimusmenetelmien arviointi 60
6.2 Tutkimuksen luotettavuus 60
6.3 Tutkimuksen eettisyys 63
7 JOHTOPÄÄTÖKSET JA POHDINTA 64
LÄHTEET 66
LIITTEET 69
Liite 1. Haastattelukysymykset pääkäyttäjille. 69
Liite 2. Haastattelukysymykset projektipäälliköille 71
Liite 3. Listaus kaikista haastatteluissa tehdyistä havainnoista 72

LYHENTEET JA TERMIT

IBM	International Business Machines
ISTQB	International Software Testing Qualifications Board
IT	Information Technology
JHS	Julkisen hallinnon suositus
LbD	Learning by Developing
MTBF	Mean time between failure
t & k	Tutkimus ja kehitys

1 JOHDANTO

Tietojärjestelmälle suoritettava suunnitelmallinen testaus ennen tuotantoon käyttöönottoa on yksi tärkeimpiä vaiheita tietojärjestelmän käyttöönoton onnistumisessa. Tietojärjestelmän testaus, niin uutta tietojärjestelmää hankittaessa kuin vanhaa päivitettäessäkin, jää kuitenkin usein liian vähälle huomiolle. Puutteellisesti testattu ja siten ennenaikaisesti tuotantokäyttöön viety tietojärjestelmä saattaa aiheuttaa merkittäviä lisäkustannuksia ja erityisesti hankaloittaa loppukäyttäjien arkea. Tästä huolimatta tietojärjestelmän testaukselle varataan usein liian vähän aikaa sekä resursseja.

Tämän opinnäytetyön aiheena on testauskäytäntöjen kehittäminen Laurea-ammattikorkeakoulun tietojärjestelmäprojekteissa. Opinnäytetyön puitteissa kartoitetaan tietojärjestelmäprojektien testaukseen liittyviä nykykäytäntöjä niin vahvuuksineen, heikkouksineen kuin kehityskohteineen. Selvityksen kautta saatuja tietoja analysoidaan ja tehdään päätelmiä mitkä osa-alueet tai toiminnot kaipaavat kehittämistä. Näiden päätelmien sekä ohjelmistotestaukseen liittyvien taustateorioiden pohjalta määritellään kehittämiskohteita ja ratkaisuehdotuksia, joiden avulla voidaan kehittää Laurean tietojärjestelmäprojektien testausta. Tavoitteena opinnäytetyössä on tuottaa konkreettisia työvälineitä Laurea-ammattikorkeakoulun tietojärjestelmäprojektien testauksen kehittämiseksi.

Seuraavassa luvussa kuvataan opinnäytetyön tutkimusasetelmaa sisältäen toimeksiantajan esittelyn sekä kuvaten tutkimuksen taustaa, tavoitteita että rajauksia. Luvun lopuksi esitellään opinnäytetyön tutkimuskysymykset ja kuvataan käytetty tutkimusmenetelmä. Luvut kolme ja neljä muodostavat opinnäytetyön teoriaosuuden. Luvussa kolme käydään läpi testauksen yleistä teoriaa. Luvussa on määritelty mitä testaus yleisesti tarkoittaa, mitä vaiheita koko testausprosessiin kuuluu ja mitä erilaisia testausmenetelmiä voidaan testauksessa hyödyntää. Luvussa neljä perehdytään tarkemmin erilaisiin testaustasoihin ja siihen, mitkä ovat kunkin testaustason tavoitteet ohjelmiston testauksessa.

Teoriaosuuden jälkeen luvussa viisi kuvataan kuinka tutkimus tietojärjestelmien testauskäytännöistä toteutettiin sekä miten saatua tutkimusaineistoa käsiteltiin ja analysoitiin. Luvussa viisi pääpaino on tutkimuksessa havaittujen kehittämiskohteiden ja niihin muodostettujen ratkaisuehdotuksien läpikäynnissä. Ratkaisuehdotuksien muodostamisessa on

tavoitteena ollut mahdollisimman konkreettisten ja helposti käyttöön otettavien ratkaisujen luominen. Lopuksi luvussa kuusi pohditaan tutkimuksen laatua, luotettavuutta ja eettisyyttä.

2 TUTKIMUSASETELMA

Opinnäytetyöni toimeksiantajana toimi työnantajani, Laurea-ammattikorkeakoulu (jatkossa Laurea). Tässä luvussa esitellään toimeksiantaja, tutkimuksen taustaa sekä kuvataan tutkimukselle asetettuja tavoitteita. Luvun lopussa käydään läpi opinnäytetyölle määritetyt tutkimuskysymykset sekä esitellään käytetty tutkimusmenetelmä.

2.1 Toimeksiantajan esittely

Laurea on Uudellamaalla toimiva uutta osaamista tuottava palveluinnovaatioiden ammattikorkeakoulu, jossa työelämäläheisyys on keskeisessä asemassa. Laureassa toteutetaan työelämäläheistä koulutusta, aluekehitystä ja t & k-toimintaa Learning by Developing (LbD) –toimintamallilla. Laureassa on lähes 8000 opiskelijaa ja henkilöstöä noin 500. (Laurea 2016)

Laurean yhtenä erityispiirteenä ovat rikosseuraamus- ja turvallisuusalan koulutukset, joita Laurea tarjoaa ainoana ammattikorkeakouluna Suomessa. Kaiken kaikkiaan ammattikorkeakoulututkintoja voi suorittaa kymmeneltä eri koulutusalalta. Lisäksi on mahdollisuus suorittaa englanninkielisiä ja ylempiä ammattikorkeakoulututkintoja sekä avoimen ammattikorkeakoulun opintoja. Laurea vaikuttaa laajalti Uudellamaalla, sillä se toimii seitsemällä eri kampuksella. Kampukset sijaitsevat Hyvinkäällä, Keravalla, Leppävaarassa, Lohjalla, Otaniemessä, Porvoossa ja Tikkurilassa. (Laurea 2016)

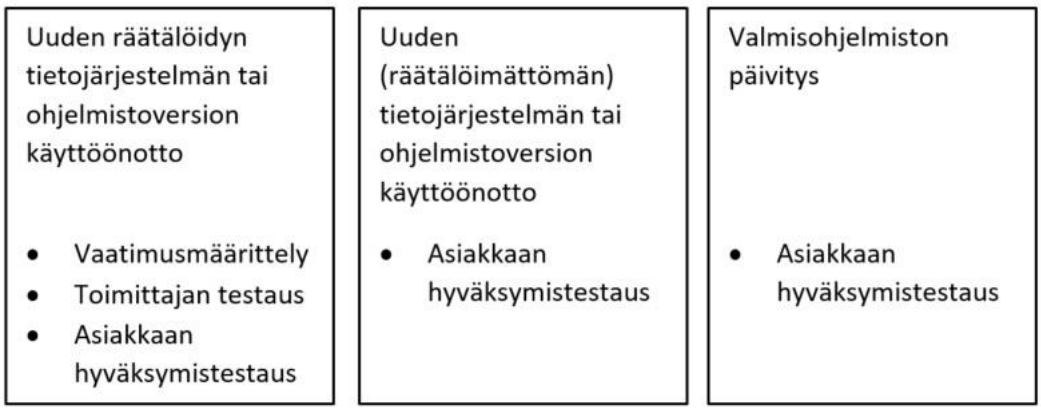
Laureassa on jo pitkään kehitetty ammattikorkeakoulumaista oppimista ja tuloksena on 2000-luvun alussa syntynyt kehittämällä oppimisen eli Learning by Developing –toimintamalli. Toimintamallin lähtökohtana on integroida opetus, tutkimus- ja kehitystyö sekä aluekehitys, ja parantaa näin oppimisen työelämäläheisyyttä. Kehittämispohjainen oppiminen on uutta luova toimintamalli, joka perustuu autenttisuuteen, kumppanuuteen, kokemuksellisuuteen ja tutkimuksellisuuteen. (Kallioinen 2008, 4, 7.)

2.2 Tutkimuksen taustaa

Laureassa uusien tietojärjestelmien hankintaprojekteja vetävät joko projektipäälliköt tai tulevan tietojärjestelmän pääkäyttäjä(t). Käytössä olevien tietojärjestelmien versiopäivityksistä vastaavat tietojärjestelmän pääkäyttäjät, jotka ovat usein myös itse tietojärjestelmän käyttäjiä. Pääkäyttäjän vastuulla on tietojärjestelmän ylläpito sekä kehitys- ja tukitehtävät, jotka voivat olla hyvinkin laajat. Yleensä kehitys- ja tukitehtäviin kuuluu muun muassa tietojärjestelmän käytettävyyden ja palvelujen kehittäminen, käyttöoikeuksien hallinta ja käyttäjien ohjeistaminen sekä kouluttaminen. Pääkäyttäjä on vastuussa myös ongelmatilanteiden selvittämisestä sekä ongelmien raportoinnista tietojärjestelmän toimittajalle.

Laureassa tietojärjestelmät hankitaan kilpailuttaen toimittajat kuhunkin hankintaan liittyen. Myös tietojärjestelmien kehitystyötä tehdään yhteistyössä toimittajien kanssa, sillä tietojärjestelmien ohjelmistokehitystä ei juurikaan tehdä Laureassa. Tietojärjestelmien kehittäminen yhdessä toimittajan kanssa vaatiikin tietojärjestelmien pääkäyttäjiltä aikaa ja aktiivista osallistumista.

Laureassa on kolmen tyyppisiä tietojärjestelmähankkeita: 1) Uuden räätälöidyn tietojärjestelmän tai -version käyttöönotto, 2) Uuden (räätälöimättömän) tietojärjestelmän käyttöönotto sekä 3) Valmisohjelmiston päivitys. Kuvassa 1 on kuvattu kullekin tietojärjestelmähankintatyypille tyypillisiä toimintoja. Toimittaja on tietojärjestelmän valmistaja ja asiakas tietojärjestelmän tilaaja. Tässä opinnäytetyössä asiakas on Laurean tietojärjestelmän pääkäyttäjä ja/tai projektipäällikkö.



KUVA 1. Erilaisia tietojärjestelmähankkeita Laureassa

Uuden räätälöidyn tietojärjestelmän tai ohjelmistoversion käyttöönottohanke lähtee käyntiin tietojärjestelmän vaatimusmäärittelyllä. Uutta tietojärjestelmää hankittaessa vaatimusmäärittelyssä määritellään muun muassa mitä toimintoja tietojärjestelmän pitäisi toteuttaa, minkälaisessa ympäristössä tietojärjestelmän pitäisi toimia ja miten käyttöoikeuksia hallitaan. Vaatimusmäärittely tarvitaan jo osaksi toimittajan kilpailutusta, jonka pohjalta toimittajakandidaatit muodostavat tarjouksensa tietojärjestelmän toteuttamisesta. Tietojärjestelmän käyttöönoton jälkeen tulee yleensä tarve kehittää tietojärjestelmän toimintoja tai havaitaan virhetilanteita, jotka pitäisi korjata. Näissä tilanteissa tehdään toimittajalle joko uusi vaatimusmäärittely halutuista toiminnallisista muutoksista tai virheraportteja havaituista virhetilanteista. On sitten kyse kokonaan uudesta tietojärjestelmästä tai vain ohjelmistoversiopäivityksestä, toimittaja toteuttaa ja testaa tietojärjestelmän tai tehdyt muutokset sekä niiden vaikutukset tietojärjestelmän muihin toiminnallisuuksiin. Tämän jälkeen alkaa asiakkaan hyväksymistestaus. Uuden räätälöidyn tietojärjestelmän tai ohjelmistoversion hankintaprosessi vaatimusmäärittelystä asiakkaan hyväksymistestaukseen vaatii hyvin onnistuakseen tiivistä yhteistyötä ja kommunikointia sekä selkeää yhdessä sovittua työnjakoa toimittajan ja asiakkaan välillä.

Uuden räätälöimättömän tietojärjestelmän käyttöönotto eroaa edellisestä siten, että tässä hanketyypissä tietojärjestelmä on valmis kokonaisuus, jota ei räätälöidä asiakkaan tarpeiden mukaan vaan se hankitaan ja otetaan käyttöön sellaisenaan. Tähän hanketyyppiin kuuluvia tietojärjestelmiä usein jatkokehitetään yhteistyössä toimittajan ja muiden ammattikorkeakoulujen kanssa. Ohjelmistoversiopäivitykset sisältävätkin siten muutoksia niin omiin kuin muidenkin korkeakoulujen muutostarpeisiin. Asiakkaan hyväksymistestaus on tässäkin hanketyypissä tärkeä vaihe ennen tietojärjestelmän tai uuden ohjelmistoversion käyttöönottoa.

Valmisohjelmiston päivitys tarkoittaa ohjelmiston toimittajan julkaisemia ohjelmistoversiopäivityksiä, jotka ovat saatavilla kaikille ohjelmiston käyttäjille. Päivityksiä ei siis tilata asiakkaan toimesta vaan ohjelmiston toimittaja julkaisee niitä tarvittaessa. Ohjelmiston julkaisutiedotteessa kuvataan yleensä tehdyt muutokset. Asiakkaalla on päätösvalta milloin nämä valmisohjelmiston päivitykset asennetaan asiakkaalla ohjelmiston loppukäyttäjille. Jotta tämä päätös voidaan tehdä, pitää asiakkaan suorittaa oma hyväksymistestaus ja varmistaa ohjelmistopäivityksen toimivuus omassa ympäristössä.

2.3 Tutkimuksen tavoitteet ja rajaus

Tämän opinnäytetyön tavoitteena on testauskäytäntöjen kehittäminen Laurean tietojärjestelmäprojekteissa. Tietojärjestelmäprojektien vetovastuullisia ovat joko projektipäälliköt tai tietojärjestelmien pääkäyttäjät. Yleensä tietojärjestelmän pääkäyttäjä on tietojärjestelmän asiantuntijana vastuussa ylläpidon ja tuen lisäksi tietojärjestelmän kehittämisestä sekä sen laadusta että yhteydenpidosta tietojärjestelmän toimittajaan. Empiirisessä osuudessa selvitetään millaisia testauskäytäntöjä tietojärjestelmien pääkäyttäjillä ja projektipäälliköillä on tällä hetkellä käytössä. Empiirisessä osuudessa kartoitetaan myös mitkä ovat tietojärjestelmän toimittajan vastuut testauksessa sekä miten valmistaudutaan ottamaan tietojärjestelmä tuotantokäyttöön. Toimittajat itsessään on rajattu tämän tutkimuksen ulkopuolelle. Toimittajille kuuluvat vastuut kartoitetaan tässä tutkimuksessa tietojärjestelmäprojektien vetovastuullisten näkökulmasta.

Opinnäytetyön tarkoituksena on tuoda esille tietojärjestelmän testauksen merkitys ja tärkeys käyttöönottoprojekteissa. Tavoitteena on kasvattaa projektipäälliköiden ja tietojärjestelmien pääkäyttäjien tietotaitoa tietojärjestelmien testauksen suhteen. Tämän tavoitteen täyttyminen mahdollistaisi osaltaan laadukkaampien tietojärjestelmäprojektien toteuttamisen. Tärkein tavoite opinnäytetyössä on kuitenkin tuottaa mahdollisimman konkreettisia työvälineitä tietojärjestelmäprojektien testauksen kehittämiseksi.

2.4 Tutkimuskysymykset

Opinnäytetyön tavoitteet voidaan purkaa seuraaviin tutkimuskysymyksiin:

1. Mitä konkreettisia työvälineitä voitaisiin ottaa käyttöön Laurean tietojärjestelmäprojekteissa testauksen kehittämiseksi?

2. Miten kehittää Laurean tietojärjestelmäprojektien vetovastuullisten tietotaitoa testaukseen liittyen?

Ensimmäinen tutkimuskysymys on opinnäytetyön haasteellisin tavoite. Siinä pohditaan miten käytännössä ratkaista havaittuja kehitystarpeita. Tavoitteena on, että opinnäytetyön tuotokset on helppo ottaa käyttöön tietojärjestelmäprojekteissa. Toinen tutkimuskysymys

keskittyy löytämään tapoja miten kasvattaa tietojärjestelmäprojektien pääkäyttäjien ja projektipäälliköiden tietämystä yleisesti testauksen merkityksestä ja siitä, kuinka testausta käytännössä tehdään erilaisissa tietojärjestelmähankkeissa.

2.5 Tutkimusmenetelmä

Tutkimusmenetelmänä käytettiin kvalitatiivista eli laadullista tutkimusta, joka toteutettiin teemahaastatteluilla. Haastatteluilla pyrittiin kartoittamaan haastateltavien näkemyksiä tutkimuksen kohteena olevien tietojärjestelmien testauksen nykykäytännöistä.

Laadullinen tutkimus on tieteellisen tutkimuksen menetelmäsuuntaus, jossa pyritään ymmärtämään kohteen laatua, ominaisuuksia ja merkityksiä kokonaisvaltaisesti (Jyväskylän yliopisto 2015). Laadullisessa tutkimuksessa tutkimussuunnitelma muokkautuu tutkimushankkeen edetessä. Tämä korostaa tutkimuksen eri vaiheiden – aineistonkeruun, analyysin, tulkinnan ja raportoinnin – kietoutumista yhteen. Tulkinta jakautuu koko tutkimusprosessiin ja sen pilkkominen eri vaiheisiin voi olla joskus haastavaa. Aineistonkeruun edetessä voi joutua tarkastelemaan uudestaan niin tutkimussuunnitelmaa kuin jopa tutkimusongelman asetteluakin. (Eskola & Suoranta 2000, 16.)

Haastattelu sopii tutkimusmenetelmäksi silloin, kun ei tiedetä, millaisia vastauksia tullaan samaan, tai kun vastaus perustuu haastateltavan henkilön omaan kokemukseen. Haastattelua käytetään myös silloin, kun halutaan syventää tietoa jostakin asiasta. (Hirsjärvi & Hurme, 2000, 35.) Haastattelun yhtenä etuna pidetään sitä, että siinä voidaan joustavasti eri tilanteiden mukaan huomioida haastateltavat ja kerätä tietoa. Haastatteluaiheiden järjestystä voi tarvittaessa muuttaa, ja se antaa mahdollisuuden myös tulkinnan tekemiseen. Haastattelu onkin ainutlaatuinen tiedonkeruumenetelmä, sillä siinä ollaan suorassa kielellisessä vuorovaikutuksessa tutkittavan kanssa. (Hirsjärvi, Remes & Sajavaara 2013, 204-205.)

2.5.1 Teemahaastattelu

Tässä tutkimuksessa haastattelumenetelmänä käytettiin teemahaastattelua, joka on puolistrukturoitu haastattelumenetelmä. Puolistrukturoidulle haastattelulle on ominaista, että

jokin haastattelun näkökohta on lyöty lukkoon, mutta ei kuitenkaan kaikkia. Teemahaastattelussa haastattelu kohdennetaan tiettyihin aihealueisiin. Haastattelu muistuttaa keskustelua, jolla on etukäteen määritelty tarkoitus. Teemahaastatteluun kuuluu, ja nähdään jopa keskeisenä, että ihmiset tulkitsevat ja antavat merkityksiä asioille. Teemahaastattelulle on myös ominaista, että haastateltavat ovat kokeneet tietyn tilanteen. Haastattelu suunnataankin haastateltavien henkilöiden subjektiivisiin kokemuksiin. (Tilastokeskus; Hirsjärvi & Hurme 2004, 47-48.) Teemahaastattelu valittiin haastattelumuodoksi, koska haastatteluissa keskityttiin tarkan kysymyslistan sijaan keskusteluun tiettyjen teemojen puitteissa. Haastateltavat saivat kysymyslomakkeet (liitteet 1 ja 2) ennen haastattelua valmistautuakseen haastatteluun, mutta itse haastattelu suoritettiin keskustellen kysymysten esittämistä teemoista, ei tarkasti kysymyslistaa noudattaen.

3 OHJELMISTOTESTAUS

3.1 Mitä testaus on?

Mayers, Badgett & Sandler (2011, 6) teoksessaan ”The Art of Software Testing” määrittelevät ohjelmistotestauksen käsitteen seuraavasti:

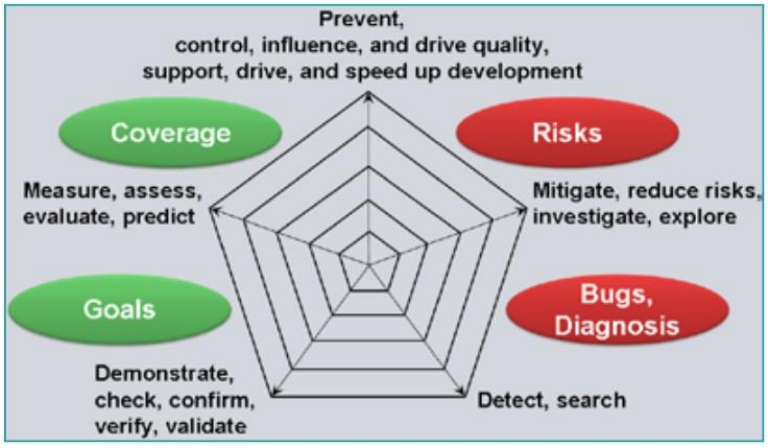
Testing is the process of executing a program with the intent of finding errors.

Testaus on siis prosessi, jossa suoritetaan ohjelmistoa tavoitteena virheiden löytäminen. Myös Patton (2006, 19) kuvaa testaajan tavoitetta samaan tapaan määrittelemällä testaajan tavoitteeksi löytää virheitä ja löytää ne mahdollisimman aikaisessa vaiheessa. Testauksen tavoitteena voidaan siis nähdä enemminkin virheiden löytäminen kuin osoittaminen, että ohjelmisto toimii. Testauksessa pitäisikin lähteä oletuksesta, että ohjelmisto sisältää aina virheitä ja testauksen tehtävä on löytää niistä mahdollisimman monta mahdollisimman nopeasti (Mayers, Badgett & Sandler 2011, 6).

ISTQB (International Software Testing Qualifications Board) (2016) määrittelee testauksen laajempana kokonaisuutena seuraavasti: ”The process consisting of all lifecycle activities, both static and dynamic, concerned with planning, preparation and evaluation of software products and related work products to determine that they satisfy specified requirements, to demonstrate that they are fit for purpose and to detect defects.” ISTQB:n määritelmän mukaan testaus on prosessi sisältäen testauksen suorittamisen ja virheiden löytämisen lisäksi koko testauksen elinkaaren vaiheet suunnittelu-, valmistelu- ja arviointivaiheineen. Testausprosessin avulla varmistetaan, että tuotteet vastaavat tehtyjä määrityksiä ja sopivat määritettyyn tarkoitukseen.

Peter Zimmerer artikkelissaan ”The Value of Testing in 5 Dimensions” (2010/09, 7-9) tuo esille, että testaus pitäisi kuvata laajempana kokonaisuutena, jotta testauksen merkitys ja arvostus osana ohjelmistokehitystä paranisi. Zimmerer (2010/09) on kokenut ohjelmistoalan insinööri Siemens AG-yhtiöstä, joka on toiminut yli 19 vuotta ohjelmistotestauksen ja laadunhallinnan parissa. Hän toimii konsulttina, valmentajana ja kouluttaja testauksen hallintaan ja testauskäytäntöihin liittyen. Zimmerer kuvaa testaukseen kuuluvaksi 5

ulottuvuutta (kuva 2), joiden tarkoitus on ilmaista paremmin testauksen tarkoitusta, motivaatiota, toimintoja ja arvoja.



KUVA 2. Testauksen viisi ulottuvuutta (Zimmerer 2010, 7)

Testauksen ensimmäinen tavoite (Goals) on osoittaa, että testattava kohde toimii, ainakin jossakin määrin. Tämän jälkeen testaus laajenee varmistamaan, että ohjelmisto toimii juuri vaatimusmäärittelyn mukaisesti. (Zimmerer, 2010, 7.)

Testauksen aikana (Coverage) kerätään tietoa ohjelmiston toimivuudesta ja mitataan muun muassa ohjelmiston suorituskykyä, luotettavuutta tai saatavuutta. Näiden tietojen perusteella tehdään muutoksia ohjelmistoon ja päätetään tuotteistamisesta sekä suunnitellaan tuotteen tulevaisuuden kehityslinjoja. (Zimmerer, 2010, 8.)

Yksi tärkeimpiä tavoitteita ohjelmistokehityksessä on estää virheiden syntyminen (Prevent). Testauksella voidaan edistää tätä tavoitetta. On tärkeätä analysoida tarkasti raportoidut virheet ja selvittää mikä on ollut perimmäinen syy siihen, että virhe on tullut ohjelmistoon. Analysoinnin perusteella voidaan parantaa ohjelmiston kehitysprosessia, jotta voidaan välttää vastaavanlaisien virheiden syntyminen tulevaisuudessa. Myös testiohjattua suunnittelua, ”Test-Driven Development”, käyttämällä voidaan ohjata ohjelmiston kehitystyötä ja mahdollistaa tehokkaampi ohjelmiston kehitysprosessi sekä virheiden korjaaminen. (Zimmerer, 2010, 8.)

Neljännessä (Risks) ja viidennessä (Bugs, Diagnosis) ulottuvuudessa testauksen tavoitteena on ohjelmiston riskien vähentäminen sekä virheiden havaitseminen mahdollisimman aikaisessa vaiheessa. Ohjelmiston jo tiedossa olevia riskejä pitäisi ottaa huomioon testitapauksia suunniteltaessa ja kohdentaa niihin osa testitapauksista. Myös virheiden

korjaaminen mahdollisimman aikaisin tuotekehityksessä vähentää ohjelmiston riskejä. (Zimmerer, 2010, 8.)

Testaus käsitteenä on siis erilaisten määritelmien mukaan hyvinkin laaja kokonaisuus. Ensisijaisesti testaamisella pyritään varmistamaan ohjelmiston laadukkuus löytämällä mahdollisimman aikaisessa vaiheissa virheitä ja varmistamaan, että ohjelmisto vastaa vaatimusmäärittelyä. Testauksen tavoitteita on kuitenkin vaikea saavuttaa, mikäli testaaja ei tunne testauksen perusperiaatteita, testauksen prosessia eikä ole koulutettu käyttämään testaukseen tarkoitettuja työkaluja ja erilaisia testaustekniikoita. (Perry 2006, 39.)

Tekijöitä, jotka vaikuttavat alentavasti testauksen laadukkuuteen, ovat muun muassa seuraavat asiat:

- Testaukseen ei ole käytettävissä tarpeeksi aikaa tai resursseja, jolloin kaikkia toteutettuja toiminnollisuuksia ei ehditä testaamaan
- Puutteellisen testausprosessin johdosta ei saavuteta parasta mahdollista lopputulosta testauksessa
- Testaajien testausosaaminen ei ole riittävä, jolloin testaajilla ei ole riittävästä tietotaitoa suorittaa testausta tavoitteiden mukaisesti. (Perry 2006, 39.)

Testaus on paljon muutakin kuin vain testauksen suorittamista. Testaukseen kuuluu muun muassa myös testauksen suunnittelua, testausympäristön valmisteluja ja testituloksien raportointia sekä analysointia. Seuraavissa alaluvuissa tarkastellaan näitä tarkemmin.

3.2 Testauksen suunnittelu

Laadukasta testausta ei tapahdu ilman suunnittelua. Ohjelmiston testaus on työmäärältään merkittävä osa ohjelmistoprojektia. Jo testaussuunnitelman rakentamiseen kannattaa varata aikaa, sillä hyvä suunnitelma takaa sujuvamman testauksen, analysoinnin ja raportoinnin. Testaussuunnitelmassa kuvataan mitkä ovat testauksen tavoitteet, tarvittavat aika-, henkilöstö- ja laiteresurssit, mahdolliset koulutustarpeet sekä miten testaus organisoidaan että hallinnoidaan. Testaussuunnitelmassa kuvataan minkälaista testausta ja millä testausmenetelmillä testaus tullaan suorittamaan ohjelmistolle. Testaussuunnitelma ottaa

kantaa myös siihen, millä kriteereillä testaus tullaan lopettamaan. (Perry 2006, 209; Tuovinen 2013b, 6, 9.)

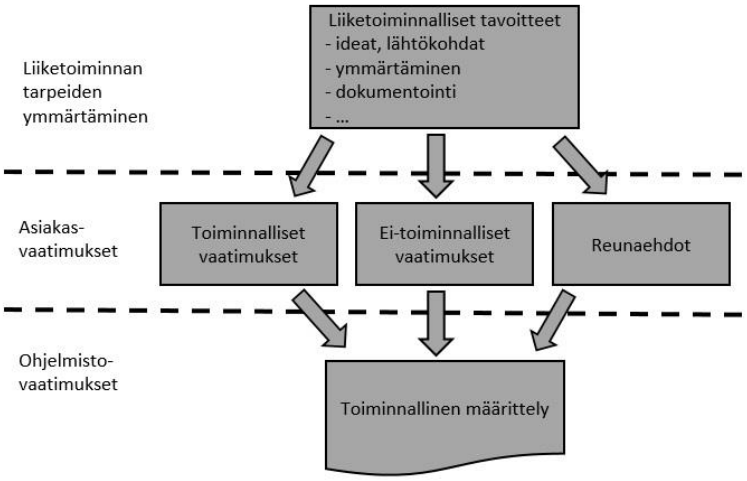
Testaussuunnitteluun vaikuttavat monet tekijät. Testauksessa käytettävät testausmenetelmät ovat yksi merkittävin testaussuunnitteluun vaikuttava tekijä. On hyvä huomioida myös miten hyvin ohjelmisto on jaettavissa testattaviin osiin. Kuitenkin testitapauksien suunnittelun tärkeimpänä lähtökohtana ovat ohjelmiston vaatimukset. Suunnittelun alkuvaiheessa on hyvä varmistaa onko vaatimukset kuvattu niin, että sen perusteella ymmärretään mikä on ohjelmiston odotettu käyttäytyminen/vaste kyseisessä tilanteessa. Lisäksi jäljitettävyys vaatimuksien ja konkreettisten testitapauksien välillä on tärkeätä, jotta voidaan muun muassa varmistaa, että kaikki vaatimukset on katettu testitapauksien suunnittelussa. (Tuovinen 2013b, 11-12, 14.)

Seuraavissa alaluvuissa perehdytään tarkemmin mitä vaatimukset ovat ja miten niitä voidaan kuvata sekä miten suunnitellaan testitapauksia.

3.2.1 Asiakasvaatimukset

Kattava ja hyvin dokumentoitu vaatimusmäärittely on yksi tärkeimpiä edellytyksiä onnistuneelle ohjelmistoprojektille. Puutteet vaatimuksissa, esimerkiksi puuttuvat ja virheelliset vaatimukset, ovat tänä päivänäkin yksi suurimmista syistä projektien epäonnistumiseen. (Paakki 2011, 3, 6.) Haikalan & Mikkosen (2011, 61) mukaan ”vaatimus on jotain, mitä tuotteella pystyy tekemään tai (laatu-) ominaisuus, joka tuotteella tulee olla”. Vaatimukset luokitellaan tavallisesti 1) toiminnallisiin vaatimuksiin, 2) ei-toiminnallisiin vaatimuksiin ja 3) reunaehtoihin. Näitä kutsutaan myös asiakasvaatimuksiksi. Kuvassa 3 kuvataan miten asiakkaan liiketoiminnalliset tavoitteet ovat pohjana asiakasvaatimuksille. Vaatimusmäärittelyn tavoitteena on selvittää tietojärjestelmän asiakasvaatimukset mahdollisimman kattavasti. Asiakasvaatimuksien toteuttaminen vaatii sitä vastoin tarkkaa ohjelmistovaatimuksien määrittelyä, jotka ovat käytännössä ohjelmiston erilaisia toimintoja. Vaatimusmäärittelyn lopputuotoksena syntyy yleensä toiminnallinen määrittely, joka sisältää sekä asiakas- että ohjelmistovaatimukset. Hyväksyttyihin vaatimuksiin kohdistuu yleensä muutostarpeita, joiden käsittely, analysointi ja hyväksyminen on myös osa

vaatimustenhallintaa. Vaatimustenkäsittely, sisältäen vaatimuksien määrittelyn ja hallinnan, on kokonaisuudessaan yksi perusedellytys ohjelmistoprojektin onnistumiselle. (Haikala & Mikkonen 2011, 61-62.)



KUVA 3. Asiakas- ja ohjelmistovaatimukset (Haikala & Mikkonen 2011, 62, muokattu)

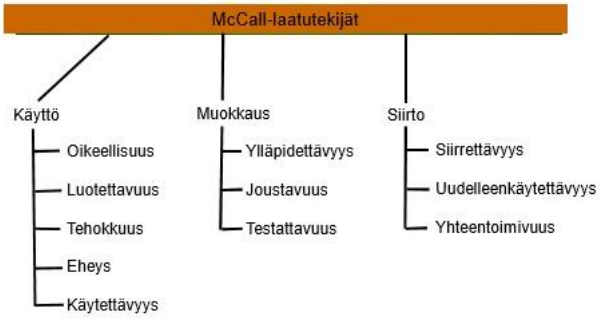
JHS 173 (JHS-suositukset 2012) määrittelee toiminnallisen ja ei-toiminnallisen vaatimuksen seuraavasti:

> **Toiminnallinen vaatimus**
> Vaatimus, joka määrittelee kehitettävän tai hankittavan järjestelmän käyttäytymistä tai toiminnallisuutta. Toiminnalliset vaatimukset määrittelevät, mitä palveluja ohjelmiston on tarjottava, miten ohjelmisto reagoi syötteisiin ja miten se käyttäytyy annetuissa tilanteissa. Voi olla joko käyttäjä- tai järjestelmävaatimus. (JHS-suositukset 2012.)
>
> **Ei-toiminnalliset vaatimukset**
> Ei-toiminnalliset vaatimukset määrittelevät rajoitukset ja reunaehdot toiminnallisille vaatimuksille. Ei-toiminnalliset vaatimukset eivät liity suoraan palveluihin vaan kertovat, mitä ehtoja järjestelmän on täytettävä, jotta toiminnalliset vaatimukset voidaan toteuttaa.

Reunaehdot rajoittavat myös tietojärjestelmän toimintaa, kuten ei-toiminnalliset vaatimukset. JHS 173:n (JHS-suositukset 2012) mukaan tietojärjestelmän tekniset reunaehdot määritetään esimerkiksi käyttäjien tarpeista ja organisaation tietohallinnon suosituksista tai vaatimuksista. Reunaehtona voi olla esimerkiksi: ”Palvelimina käytetään vain A-tyypin palvelimia, joissa käyttöjärjestelmänä on B.” (JHS-suositukset 2012)

Ohjelmiston laatua määrittelee se kuinka tyytyväinen asiakas on ohjelmistoon sitä käyttäessään. Mikäli asiakas on tyytymätön ohjelmiston toimivuuteen, tyytymättömyys ei useinkaan johdu siitä, että ohjelmistosta puuttuisi toiminnallisuuksia. Kaikki suunnitellut toiminnallisuudet voivat toimia kuten on määritelty. Tyytymättömyyttä aiheuttaa usein sen sijaan ohjelmiston hankala käyttö, ohjelmiston hitaus tai ohjelmiston vaikea ylläpidettävyys. Tämä tarkoittaa, että ohjelmiston ei-toiminnalliset vaatimukset eivät ole asiakkaan haluamalla tasolla. Ei-toiminnallisia vaatimuksia ei ole toteutettu määritetyllä tavalla, ne on kuvattu puutteellisesti vaatimusmäärittelyssä, tai jokin kirjaamaton ei-toiminnallinen vaatimus toteutuu ohjelmistossa. (Paakki 2014, 20-21.)

Voidaan siis sanoa, että ohjelmisto on laadukas, mikäli se täyttää sille asetetut toiminnalliset ja ei-toiminnalliset vaatimukset. Ei-toiminnallisia vaatimuksia ryhmitellään tyypillisesti erilaisiin laatuluokkiin Jim McCall:n jo 1977 muodostaman klassisen laatutekijämallin mukaisesti (kuva 4). McCall:n kolme laatutekijäluokkaa ovat: käytön laatutekijät, muokkauksen laatutekijät ja siirron laatutekijät. Ensimmäiset laatutekijät, kuten luotettavuus, vaikuttavat ohjelmiston käyttöön. Toiset laatutekijät, kuten ylläpidettävyys, vaikuttavat ohjelmiston ylläpitoon. Kolmannet laatutekijät, kuten yhteentoimivuus, vaikuttavat ohjelmiston toimivuuteen eri alustoilla. (Paakki 2014, 22-24.)



Kuva 4. McCallin laatutekijät (Paakki 2014, 25)

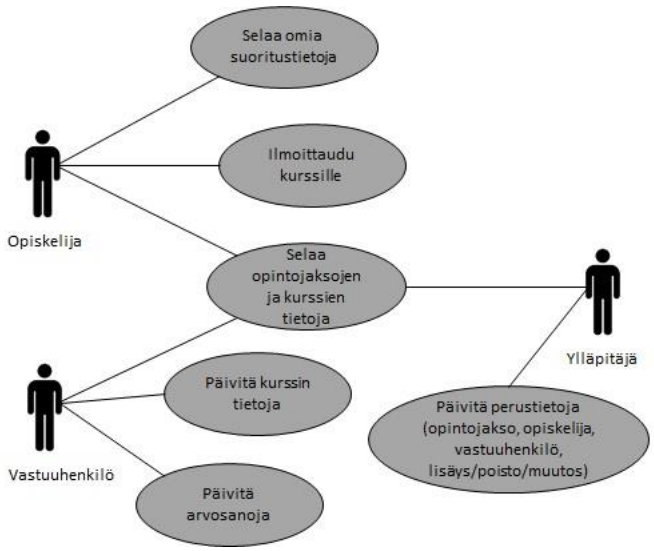
Luvussa 4 kuvataan tarkemmin toiminnallisten ja ei-toiminnallisten vaatimuksien huomioimista testauksessa.

3.2.2 Käyttötapaukset ja käyttäjätarinat

Hyvän vaatimuksen ominaisuuksia ovat virheettömyys ja selkeys. Siksi avainasemassa onkin vaatimuksen kuvaustapa ja dokumentointi. Vaatimuksia voidaan kuvata käyttäen esimerkiksi käyttötapauksia tai käyttäjätarinoita. Käyttötapaus on kuvaus tai kertomus siitä miten asiakasvaatimus kuvautuu ohjelmistovaatimukseksi. Käyttötapauksia voidaan dokumentoida esimerkiksi käyttötapauskaaviona tai kertomuksen muodossa. Käyttäjätarinoita käytetään yleisimmin ketterissä menetelmissä vaatimuksien dokumentointitapana. Ohjelmistoa ei kuitenkaan toteuteta suoraan käyttäjätarinoiden pohjalta vaan ne puretaan vielä tarkempiin tehtäväkuvauksiin, jotka määrittävät mitä ohjelmoijan tulee toteuttaa. (Haikala & Mikkonen 2011, 64, 79-84.)

Käyttötapauksien ja käyttäjätarinoiden avulla voidaan määrittää tietojärjestelmän toiminnallisuudet korkealla abstraktitasolla. Niitä voidaan käyttää myös vaatimuksien tarkentamiseen. Kuvatessa vaatimuksia käyttötapauksina tai käyttäjätarinoina voidaan varmistamaan, että asiakas ja toimittaja ymmärtävät vaatimukset samalla tavalla. Lisäksi käyttötapauskaaviota käytettäessä kuvaustapa auttaa tunnistamaan tietojärjestelmän sidosryhmät eli osapuolet, jotka käyttävät tietojärjestelmää. Käyttötapauskaavioita voidaan käyttää avuksi myös ohjelmistokehityksen suunnittelussa. Testitapauksien suunnittelussa käyttötapauskuvaukset toimivat hyvänä pohjana tarkemmalle testitapaussuunnittelulle. (Haikala & Mikkonen 2011, 64, 79-83.)

Kuvan 5 käyttötapauskaaviosta ilmenee mitkä roolit liittyvät käyttötapaukseen, mitä toimintoja käyttötapaukseen sisältyy ja miten käyttötapauksen kulku etenee. Kyseisestä käyttötapauskaaviosta voi päätellä, että joitakin toimintoja suorittaa vain jokin tietty rooli, kuten ”ilmoittaudu kurssille” ja joitakin toimintoja, kuten ”selaa opintojaksojen ja kurssien tietoja” voivat suorittaa kaikki käyttötapaukseen liittyvät roolit. Tämä kuvaus toimii myös hyvänä pohjana ohjelmiston testitapauksien suunnittelussa.



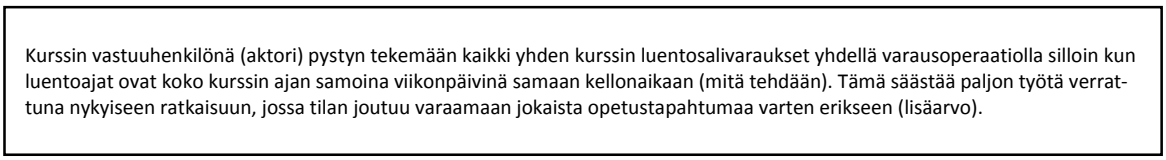
KUVA 5. Käyttötapauskaavio (Haikala & Mikkonen 2011, 77)

Käyttötapauksien kuvausmuotona käytetään käyttötapauskaavioiden lisäksi myös käyttötapauskertomuksia. Käyttötapauskertomukset sisältävät enemmän informaatiota suoritettavasta toiminnosta verrattuna käyttötapauskaavion kuvausmuotoon. Kuvassa 6 on käyttötapauskertomus opiskelijan ilmoittautumisesta kurssille. Käyttötapauskaavioon verrattuna käyttötapauskertomus määrittelee toiminnolle vaaditut tuloehdot ja odotetun lopputuloksen sekä mahdolliset poikkeukset. Tässä käyttötapauskertomuksessa on määritelty muut vaatimukset-kohdassa myös ei-toiminnallisia vaatimuksia suorituskykyyn liittyen.

Nimi: Ilmoittaudu kurssille
Versiohistoria: versio 1.0/ijh
Osallistujat: Opiskelija
Tuloehdot: Opiskelija, kurssi ja opintojakso on syötetty järjestelmään, kurssille ilmoittautuminen on avattu, opiskelija on kirjautuneena järjestelmään (KT:t Päivitä perustietoja, Päivitä kurssin tietoja)
Kuvaus: Opiskelija seuraa WWW-linkkiä, joka johtaa kurssin sivulle. Hän valitsee vaihtoehdon ilmoittaudu. Järjestelmä tarkastaa, että opiskelijalla on tarvittavat esitiedot [Poikkeus: esitietovaatimukset eivät täyty]. Järjestelmä varmistaa vielä OK/CANCEL-kyselyllä, että opiskelija todella haluaa ilmoittautua. Tämän jälkeen järjestelmä lisää opiskelijan kurssin osallistujaksi.
Poikkeukset: Esitietovaatimukset eivät täyty: opiskelijalle annetaan luettelo puuttuvista esitietokursseista.
Lopputulos: Opiskelija on rekisteröity kurssin osallistujaksi.
Muut vaatimukset: Päivittäin käsitellään kiireisimpänäkin tuntina enintään noin 50 varausta. Vastausajan on aina oltava alle 5 sekuntia.

KUVA 6. Käyttötapauskertomus (Haikala & Mikkonen 2011, 80)

Ketterien menetelmien suosimat käyttäjätarinat ovat kolmas tapa kuvata vaatimuksen sisältöä. Kuvan 7 käyttäjätarina kuvaa lausemuodossa toteutettavaa asiakasvaatimusta. Käyttäjätarinassa on tavoitteena mahdollisimman selkeä kuvaustapa, jotta asiakas ymmärtää vaatimuksen sisällön. Käyttäjätarinassa vältetään teknisiä yksityiskohtia. Siinä kuvataan kuka tekee (aktori), mitä tehdään ja mitä lisäarvoa se tuottaa aktorille (Haikala & Mikkonen 2011, 84).



KUVA 7. Käyttäjätarina (Haikala & Mikkonen 2011, 84)

3.2.3 Testitapaukset

Testitapauksien suunnittelua tapahtuu ohjelmistoprojektin eri vaiheissa. Yksikkötestauksen lasilaatikkotestaus perustuu ohjelmiston sisäiseen rakenteeseen, jolloin testitapaukset voidaan suunnitella vasta kun testattava koodi on olemassa. Järjestelmä- ja hyväksymistestauksen toiminnallinen testaus perustuu käyttötapauksiin, jolloin niihin liittyviä testitapauksia voidaan suunnitella heti kun käyttötapaukset on määritelty. (Tuovinen 2013b, 19.)

Huolellisesti suunnitellut testitapaukset ovat edellytys onnistuneelle testaukselle. Hyvän testitapauksen ominaisuuksia voidaan kuvata seuraavilla määritteillä:

- Testitapaus käyttää yhtä tiettyä toimintoa testattavasta tietojärjestelmästä
- Testitapaus kuvaa mitä testata ja miten, ja ohjaa testaajaa, joka suorittaa testauksen
- Ohjaus sisältää ohjeita testin alustamiseen ja suorittamiseen, sekä siihen mitä käyttäytymisiä tarkkailla ja kuinka arvioida lopputulos
- Testitapaukset on helpoin johtaa tietojärjestelmän toiminnallisista vaatimuksista, käytännössä käyttötapauksista. (Oppijan verkkopalvelut 2013.)

Taulukossa 1 on esimerkki testitapauksen pohjasta, joka noudattaa edellä listatun hyvän testitapauksen määritteitä. Testitapauksen dokumentoinnissa on tärkeätä testitapauksen yksilöiminen antamalla testitapaukselle nimi ja tunnus sekä jäljitettävyys siihen liittyvään käyttötapaukseen. Testitapauksen kriittisyydellä voidaan luokitella testitapaukset esimerkiksi testeihin, jotka testaavat tietojärjestelmän elintärkeitä toimintoja, tai testeihin, jotka testaavat harvemmin käytettäviä toiminnallisuuksia. Käyttäjäroolin määrittämisellä varmistetaan, että käyttöoikeusryhmät on määritelty, ja että rooleilla voi suorittaa vain kyseiselle roolille sallittuja toimintoja. Testitapauksen esiehdot on tärkeätä määritellä, jotta

ymmärretään mitä valmisteluja testitapauksen suorittaminen vaatii. Erilaisista työnkuluista on hyvä muistaa kuvata tyypillisen työnkulun lisäksi myös vaihtoehtoiset ja poikkeavat työnkulut. Erilaisia työnkuluja testaamalla saadaan myös testauksen kattavuutta kasvatettua. Tarkkojen testiaskelien kuvaus mahdollistaa testitapauksen identtisen toistettavuuden. Odotettu lopputulos määrittelee millä ehdoilla testitapaus voidaan määritellä hyväksytysti suoritetuksi. (Oppijan verkkopalvelut 2013.)

TAULUKKO 1. Testitapauksen pohja

Testitapauksen nimi	Lyhyt ja kuvaava nimi testitapaukselle, voi olla sama kuin käyttötapauksen nimi
Testitapauksen tunnus	Esimerkiksi TC0210, jossa 02 on testattavan osa-alueen tunnus ja 10 on testitapauksen juokseva numero
Käyttötapauksen nimi	Testitapauksen liittyvän käyttötapauksen nimi
Testitapauksen kriittisyys	Toiminnon kriittisyys
Käyttäjäroolit	Roolit, jotka voivat suorittaa testitapauksen
Esiehdot	Mitä tietojärjestelmässä pitää olla tehtynä, että testitapauksen voi suorittaa
Tyypillinen työnkulku	Kuvataan yleisin työnkulku
Vaihtoehtoinen työnkulku	Kuvataan tärkeimmät vaihtoehtoiset työnkulut
Poikkeavat työnkulut	Kuvataan tapahtumat, jotka keskeyttävät testitapauksen suorituksen
Testiaskel	Kuvaus mitä käyttäjä tekee
Odotettu tulos	Kuvaus mikä on tietojärjestelmän hyväksytty vastine käyttäjän toimenpiteeseen

3.3 Testiympäristön luonti

Ohjelmistotestauksen ensisijainen tavoite on löytää ohjelmiston käyttöä estävät tai haittaavat virheet ennen kuin ohjelmisto otetaan tuotantokäyttöön (Perry 2006, 38). Ohjelmiston testaajat varmistavat, että ohjelmisto täyttää sille asetetut vaatimukset vaatimusmäärittelyn mukaisesti. Tämän varmistamiseksi on oltava ympäristö, jossa ohjelmiston testausta voidaan suorittaa ennen tuotantokäyttöönottoa. Uutta tietojärjestelmää käyttöön otettaessa, voidaan rakentaa tuotantoympäristö ja tehdä siinä hyväksymistestaukset ennen tietojärjestelmän tuotantokäyttöönottoa. Testiympäristö kuitenkin tarvitaan ennemmin tai myöhemmin, sillä ohjelmistoon tulee yleensä käyttöönoton jälkeen virhekorjauksia tai halutaan toiminnallisia muutoksia, jotka pitää testata ennen ohjelmistopäivityksen siirtämistä tuotantoympäristöön. Tietojärjestelmän käyttäminen voi vaatia myös rajapintojen

rakentamista muihin asiakkaan tietojärjestelmiin. Tämä tarve tulee huomioida myös testiympäristöä suunniteltaessa. Aina ei ole kannattavaakaan rakentaa kaikkia tietojärjestelmän rajapintoja myös testiympäristöön, jolloin näiden rajapintojen osalta testaus tapahtuu vasta tuotantoympäristössä. Tällaisen tietojärjestelmän tuotantokäyttöönottoa suunniteltaessa on hyvä tehdä erityisiä suunnitelmia käyttöönoton aikatauluttamiselle ja rajapintojen testaamiselle tuotantoympäristössä.

3.4 Testauksen suorittaminen

Ohjelmistotestaajan tehtävä vaatii omia erikoistaitoja ja tänä päivänä monet yritykset määrittelevät ohjelmistotestaajan tehtävän teknisen insinöörin ammatiksi. Koulutetun ohjelmistotestaajan mukanaolo jo ohjelmiston kehitysvaiheessa mahdollistaa korkeampilaatuisen ohjelmiston rakentamisen. (Patton 2006, 20.) Teknisen osaamisen lisäksi hyvältä ohjelmistotestaajalta vaaditaan oikeata asennetta ja joskus tämä voi olla jopa tärkeämpi ominaisuus kuin tiukka testausprosessin noudattaminen. Hyvät ohjelmistotestaajat ovat kuin tutkimusmatkailijoita, jotka eivät pelkää heittäytyä tuntemattomiin tilanteisiin. He ovat myös ongelmanratkaisijoita sekä vakuuttavia, luovia ja sinnikkäitä perfektionisteja, omaten kuitenkin taidon lopettaa testauksen, kun ovat riittävän lähellä tavoitetta. (Patton 2006, 20-21.) Testauksen tavoite vaikuttaa myös testauksen suorittamiseen. Jos tavoitteena on osoittaa, että ohjelmisto toimii moitteetta, ohjelmistotestaaja ajautuu helposti valitsemaan testitapauksia, joilla on alhaisempi todennäköisyys aiheuttaa virhetilanne ohjelmiston suorittamisessa. Toisaalta, jos tavoitteeksi on asetettu osoittaa, että ohjelmistossa on virheitä, valituilla testitapauksilla on suurempi todennäköisyys paljastaa virhetilanteita. Voidaan sanoa, että onnistunut testitapaus on sellainen, jota suorittamalla aiheutetaan ohjelmistossa virhetoiminto. Lopullisena tavoitteena on kuitenkin saavuttaa testauksen avulla luottamus siihen, että ohjelmisto toimii vaatimusmäärittelyn mukaisesti ja ohjelmiston suorittamisen aikana ei tapahdu ei-toivottuja toimintoja. (Mayers, Badgett & Sandler 2011, 6, 8.)

Puhekielessä testaus -termillä tarkoitetaan usein lähes mitä tahansa kokeilemista. Esimerkiksi ohjelmiston umpimähkäinen koekäyttö jollakin syöttöaineistolla voidaan nähdä ohjelmiston testauksena. Hyvin suunnitelluissa ohjelmistoprojekteissa testaus määritellään kuitenkin hyvin suunnitelmalliseksi työvaiheeksi tarkoittaen virheiden suunnitelmallista

etsimistä ohjelmaa suorittamalla. Joskus muutaman tunnin testaus huolellisesti suunnitellulla testitapausjoukolla voi olla paljon tehokkaampaa kuin päiväkausien umpimähkäinen kokeilu. Testauksen tehtävä on siis löytää virheitä, mutta ohjelman virheettömyyttä sillä ei voida saavuttaa. Tosiasia on, että ohjelman testauksella voidaan kattaa vain osa ohjelmiston toiminnoista. Hyvästä testauskattavuudesta huolimatta, aina on mahdollista, että ohjelmaa käytettäessä tapahtuu virhetilanteita, jotka haittaavat tai jopa estävät ohjelmiston käytön. (Haikala & Mikkonen 2011, 205, 210.) Virheettömän ohjelmiston tavoittelu vie paljon aikaa ja siten nostaa kustannuksia. Käytännössä täysin virheettömän ohjelmiston saavuttaminen on lähes mahdoton tehtävä. Testauksella pyritäänkin varmistamaan, että ohjelmisto toimii hyvin yleisissä loppukäyttäjien käyttötapauksissa. (Eriksson 2012, 11-12.) Ohjelmistoprojekteissa kuluukin testaukseen, virheselvityksiin ja -korjauksiin tyypillisesti yli puolet ohjelmistoprojektin resursseista (Haikala & Mikkonen 2011, 205).

Testaussuunnitelmassa tulisi määritellä millä hyväksymiskriteereillä testaus lopetetaan. Järjestelmätestauksessa testauksen lopetuskriteerinä voi toimia tilanne, kun virheitä ei enää löydetä. Usein testausta ei voida käytännössä jatkaa näin pitkään, koska yleensä projektilla on kiinteät resurssit ja julkaisuaikataulu on lyöty kiinni. Toisaalta myös testausvaiheen keston arvioiminen on vaikeata, koska virhetilanteen tasaantumiseen tarvittavaa työmäärää ei voida arvioida. Testauksen lopettamiskriteereinä saattaakin olla testaukselle varatun ajan päättyminen, 100 % testitapauksista on suoritettu ja 100 % löydetyistä vioista on korjattu. (Haikala & Mikkonen 2011, 210, 216.)

Projektin aikataulutusvaiheessa testaukseen varattu aika ei useinkaan tule joko toteutumaan tai riittämään laadukkaan testauksen suorittamiseksi. Testausvaiheeseen kohdistuu painetta niin edeltävästä ohjelmiston toteutusvaiheesta kuin projektin julkaisupäivästä. Ohjelmiston toteutusvaihe voi kestää suunniteltua pidempään ja julkaisupäivä voi olla ”kiveen hakattu”. Testausajan lyhentyessä ollaan tilanteessa, että kaikkea ei ehditä testaamaan, jolloin on joko lykättävä tietojärjestelmän julkaisua, julkaistava tietoisesti heikosti testattu tietojärjestelmä tai karsia tietojärjestelmän julkaistavia toimintoja (Eriksson 2012, 13). Pyhäjärven & Pöyhösen (2006, 25) priorisointisääntö antaa hyvän ohjenuoran testauksen suorittamiseen: ”Priorisoi testejä siten, että koska tahansa lopettaessasi testauksen, olet tehnyt parasta mahdollista testausta saatavilla olevan ajan ja resurssien puitteissa.”

Seuraavissa alaluvuissa käydään läpi millä eri testausmenetelmillä testausta voidaan suorittaa ja mitä olisi hyvä huomioida virheiden raportoinnissa.

3.4.1 Staattinen ja dynaaminen testausmenetelmä

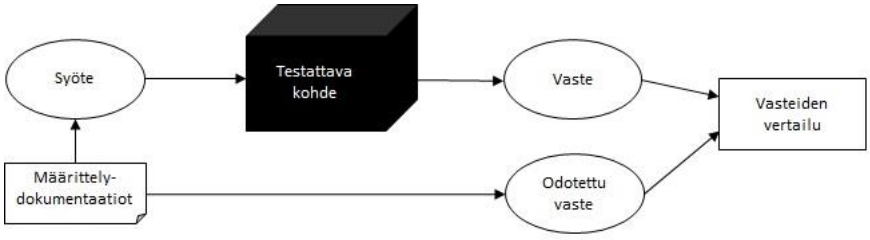
Ohjelmiston testausta voidaan kuvata käyttäen termejä staattinen ja dynaaminen testaus. Staattinen testaus on ohjelmakoodin ja dokumentaation staattista analysointia. Kuten Tuovinen (2013c, 3) kuvaa ”Testauksen kohdetta ei suoriteta, vaan sitä tutkitaan ihmismielen ja analyysityökalujen avulla.” Analysointitapana käytetään useimmiten katselmointia, joka tarkoittaa dokumenttien arviointia ja tarkastelua. Käytännössä asiantuntijat perehtyvät etukäteen katselmoitavaan dokumenttiin ja itse katselmointitapaamisessa käydään läpi asiantuntijoiden löydökset ja tehdään niiden perusteella johtopäätökset. Katselmointien tavoitteena on poistaa koodista ja dokumentaatiosta virheitä mahdollisimman aikaisessa vaiheessa ja siten nostaa ohjelmiston laatua. (Tuovinen 2013c, 2-6.)

Dynaaminen testaus on ohjelmiston tai sen osan suorittamista. Dynaaminen testaus suoritetaan tunnetuissa olosuhteissa ja ennalta määritetyillä syötteillä. Testauksen päätteeksi todetaan vastaako testauksen tulos odotettua tulosta. Ohjelmiston osittaista testausta tapahtuu tyypillisesti ohjelmiston yksikkö- ja integrointitestaustasoilla. Tällöin tiettyä ohjelmiston osaa, komponenttia, testataan käyttäen hyväksi testikehystä, joka simuloi testattavan komponentin suoritusympäristöä mahdollistaen testitapausten suorittamisen. Kokonaisen toimivan ohjelmiston testaus tapahtuu järjestelmä- ja hyväksyntätestaustasoilla. Tällöin testauksen suoritus tapahtuu käyttäen ohjelmiston käyttöliittymää tai järjestelmärajapintaa. Tässä vaiheessa kaikki ohjelmiston toiminnollisuudet on jo toteutettu, joten kaikkien testitapauksien suorittaminen onnistuu käyttäen testauksen kohteena olevaa ohjelmistoa. (Tuovinen 2013a, 3-5.)

3.4.2 Musta- ja lasilaatikkotestaus

Testausmenetelmät voidaan jakaa staattisen ja dynaamisen testauksen lisäksi musta- ja lasilaatikkotestaukseen.

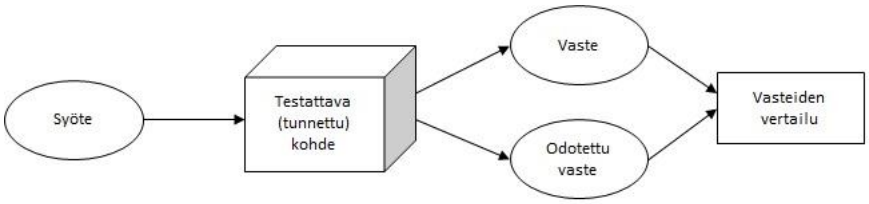
Mustalaatikkotestauksessa (engl. *black-box testing*) testauksen kohteena on ohjelmiston syöte-vaste-käyttäytyminen. Siinä tutkitaan ohjelmiston käyttäytymistä määrittelyihin ja arkkitehtuuriin verrattuna. Varsinaisesta ohjelmistokoodin toimintatavasta tai rakenteesta ei olla kiinnostuneita. Testaajalle testattava kohde voikin olla täysin tuntematon, eräänlainen musta laatikko (kuva 8). *Staattisella mustalaatikkotestauksella* testataan kirjoitettua määrittelydokumentaatiota. Tämä testausvaihe voidaan suorittaa jo aikaisessa vaiheessa ohjelmistoprojektia. Testaus, käytännössä katselmointi, voidaan suorittaa kun määrittelydokumentaatio on valmis. Virheiden löytäminen tässä vaiheessa ohjelmistoprojektia vähentää selkeästi virheiden korjauskustannuksia. (Tersa 2002, 77-78; Patton 2006, 55-56) *Dynaamisessa mustalaatikkotestauksessa* ohjelmistoa käytetään loppukäyttäjän näkökulmasta, tuntematta kuitenkaan ohjelmiston koodin rakennetta tai toimintaa. Testauksessa ohjelmistolle annetaan syötteitä, jonka jälkeen tarkastellaan ohjelmiston syötteille antamia vasteita. Dynaamista mustalaatikkotestausta kutsutaan myös käyttäytymistestaukseksi, koska testauksen tavoitteena on selvittää kuinka ohjelmisto käyttäytyy kun sitä käytetään loppukäyttäjän tapaan. (Saarinen 2008, 18.)



KUVA 8. Mustalaatikkotestaus (Kautto 1996, muokattu)

Lasilaatikkotestauksessa (engl. *white-box testing, glass-box testing*) testataan sitä vastoin ohjelmiston sisäistä rakennetta ja siten testitapauksien suunnittelu edellyttää tietämystä ohjelmiston toimintalogiikasta (kuva 9). (Patton 2006, 55-56; Äyrämö 2010, 17, 20.) Tähän viittaa myös testausmenetelmän nimi, eli testaaja näkee laatikon sisälle ja voi näin tarkastella ja analysoida koodia. *Staattisessa lasilaatikkotestauksessa* tarkastelun kohteena on ohjelmiston koodin rakenne ja toiminta. Tarkastelumetodina käytetään

yleensä katselmointia. Havaintojen perusteella voidaan arvioida ohjelmiston virhealttiita paikkoja ja keskittää testaus niihin. Riskinä on, että testaaja räätälöi testitapauksensa toimimaan niin, että testitapaukset vastaavat suoraan ohjelmistokoodin toimintoja. Staattista testausta tapahtuu kuitenkin varsin vähän ohjelmistoprojekteissa, sillä sitä ei pidetä tuottavana toimintona. (Tersa 2002, 61; Patton 2006, 55-56.) Kuitenkin mitä aiemmin virhe löydetään, sitä halvempaa sen korjaaminen on. *Dynaamisessa lasilaatikkotestauksessa* testitapaukset muodostetaan pohjautuen tietämykseen ohjelmiston sisäisestä rakenteesta. Testitapauksien suorittamisen jälkeen tuloksia verrataan odotettuihin tuloksiin ja tämän pohjalta tehdään analyyseja. (Tersa 2002, 66.)



KUVA 9. Lasilaatikkotestaus (Kautto 1996, muokattu)

3.4.3 Virheiden tunnistaminen

Virheet ovat poikkeamia asiakkaiden odotuksista ja vaatimuksista. Virheet voivat tulla esille vasta ajan myötä tai virhetilanne voi esiintyä tietyn järjestelmään annetun syötteen seurauksena. Virhettä kuvaavia termejä on useita eri käyttötarkoituksiin. Pyhäjärvi & Pöyhönen (2006, 96) määrittelevät virheen seuraavasti:

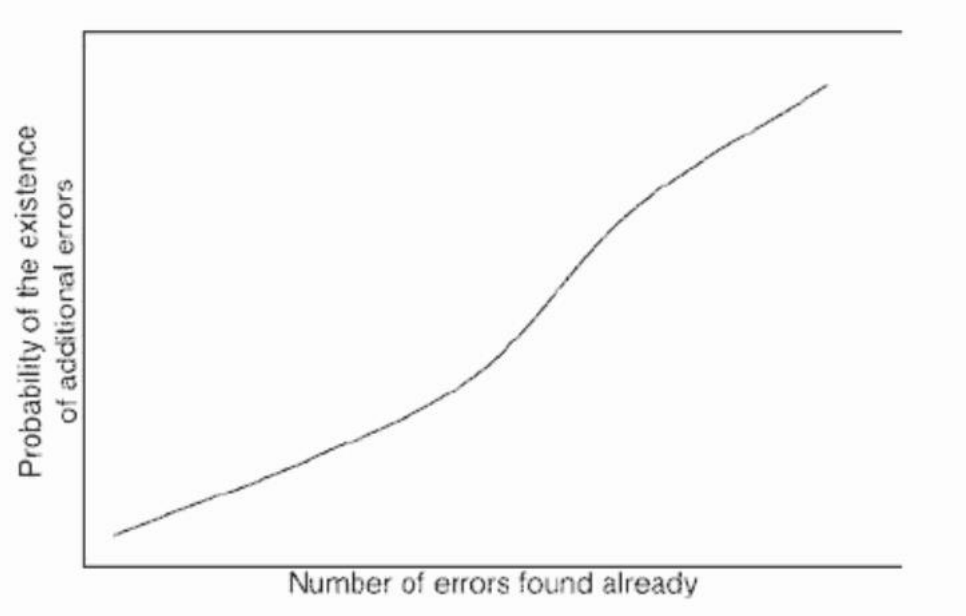
- Ihmisen toiminta, joka tuottaa väärän tuloksen on **virhe**
- Virheen ilmentymä ohjelmistossa on **vika**
- Ohjelmiston poikkeama odotetusta toimituksesta tai palvelusta on **häiriö**

Virhe-sanan käyttö normaalissa kielenkäytössä voi tarkoittaa niin virhettä, vikaa kuin häiriötäkin. Virhe sanaa käytetään normaalisti hyvin laajasti ja virhe -termiä käytetään myös tässä opinnäytetyössä ohjelmistosta löytyneessä viasta.

Miten tunnistaa virhetilanne? Pyhäjärven & Pöyhösen (2006, 113) sekä Pattonin (2006, 15) mukaan ohjelmistossa on virhe, jos ohjelmisto toimii seuraavasti:

- Ohjelmisto ei tee jotakin mitä sen määrittelyjen mukaan pitäisi tehdä
- Ohjelmisto tekee jotakin mitä sen määrittelyjen mukaan ei pitäisi tehdä
- Ohjelmisto tekee jotakin mitä määrittelyissä ei ole mukana
- Ohjelmisto ei tee jotakin mitä määrittelyissä ei ole mukana mutta pitäisi olla
- Ohjelmistoa on vaikea ymmärtää, käyttää, se on hidas tai – testaajan mielipiteen mukaan - ohjelmisto ei vain loppukäyttäjän mukaan toimi oikein.

Mayersin, Badgettin & Sandlerin (2011, 17) mukaan todennäköisyys, että ohjelmiston osassa on vielä löytämättömiä virheitä, on suoraan verrannollinen jo kyseisestä ohjelmiston osasta löydettyjen virheiden määrään (kuva 10). Heidän mukaansa virheet tuntuvat syntyvän ryhminä ja jotkut ohjelmiston osat ovat vain herkempiä virheille kuin toiset. Ohjelmistotestaajille tällainen ilmiö kertoo, että kyseisiin ohjelmiston osiin on syytä keskittää vielä lisätestausta, koska oletettavaa on, että virheitä löytyy vielä lisää. (Mayers, Badgett ja Sandler 2011, 17-18.)



KUVA 10. Jäljellä olevien ja löydettyjen virheiden välinen riippuvuus (Mayers, Badgett ja Sandler 2011, 17)

3.5 Testituloksien raportointi

Testauksen tarkoitus on löytää virheitä ohjelmiston laadun parantamiseksi. Mikäli havaittua virhetilannetta ei dokumentoida, havaintotieto katoaa, virhettä ei korjata ja siten ohjelmiston laatu ei parane. Tällöin ei voida myöskään välittää projektin johdolle tärkeätä tietoa projektin tilasta. (Pyhäjärvi & Pöyhönen 2006, 112.)

Mitä laadukkaampi virheraportti kirjoitetaan, sitä nopeammin ja varmemmin virhe tulee korjattua. Virheraportoinnissa on siis tärkeätä huomioida seuraavat asiat:

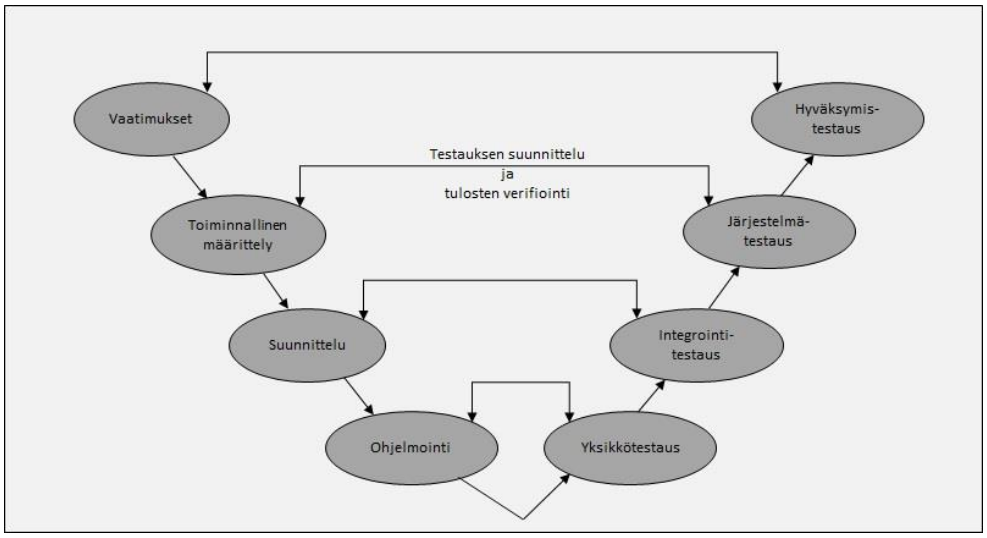
- raportoida virhe mahdollisimman pian
- kuvata virhetilanne mahdollisimman selkeästi
- kuvata tarkat askeleet, miten virhetilanne on saatu aikaiseksi
- kuvata ympäristö ja alusta
- kertoa raportissa kuka raportoi virheen ja koska
- raportoida vain yksi virhe per raportti. (Pyhäjärvi & Pöyhönen 2006, 120-123.)

Virhetilanteen toistaminen on edellytys virheen korjaamiselle. Siksi on erittäin tärkeätä, että virheen kuvaus sisältää tarkat askeleet miten virhetilanne saatiin aikaiseksi. Usein virhetilanteesta on hyvä liittää virheraporttiin kuva, mikäli se on testattavan ohjelmiston osalta mahdollista. Kuva virhetilanteesta helpottaa virheen analysointia ja oikean virhetilanteen korjaamista.

Testauksen kattavuutta ja löydettyjen virheiden määrää voidaan tarkastella testauksesta muodostettavien dokumenttien avulla. Tämä on yksi tärkeimpiä vaiheita indikoimaan testauksen laadukkuutta. (Eriksson 2012, 13.) Ilman kattavaa dokumentaatiota siitä miten testaus on suoritettu ja mitä havaintoja on tehty, on vaikea arvioida, onko tietojärjestelmä riittävän laadukas julkaistavaksi tuotantokäyttöön.

4 TESTAUSTASOT

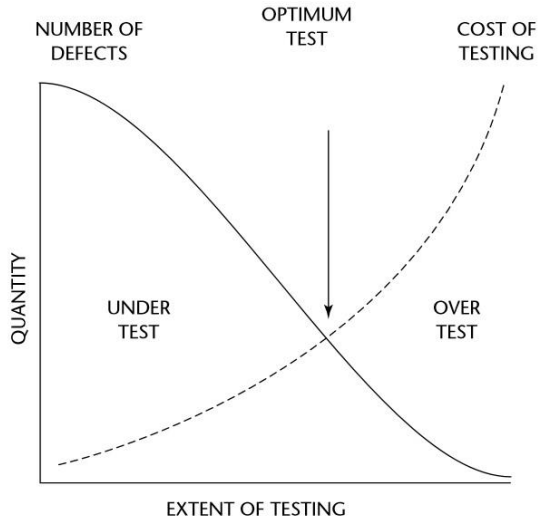
Perinteisen vesiputousmallin V-mallilla havainnollistetaan usein ohjelmiston kehitystyön ja testauksen suhdetta. Testaus tulee huomioida, resursoida ja aikatauluttaa heti ohjelmistoprojektin käynnistämisen yhteydessä. V-mallissa kukin testaustaso suunnitellaan vastaavaa ohjelmiston suunnittelutasoa vasten. Myös testaustulokset todennetaan niitä vastaaviin suunnitteludokumentteihin. (Haikala & Mikkonen 2011, 206.) Kuten V-malli kuvaa, testausta kannattaa suorittaa ohjelmiston kehittämisen jokaisessa vaiheessa, heti kehitystyön alusta lähtien. Jokaisella testaustasolla on omat tarkoituksensa ja tavoitteensa. V-mallin mukaisesti erillisiä testaustasoja ovat yksikkötestaus, integrointitestaus, järjestelmätestaus ja hyväksymistestaus (kuva 11) (Watkins & Mills 2011, 43). Alaluvuissa 4.1-4.4 käydään tarkemmin läpi testauksen eri tasoja ja, sitä mitkä ovat kunkin testaustason tavoitteet koko ohjelmiston testauksessa.



KUVA 11. V-malli ja testaustasot (Watkins & Mills 2011, 43, muokattu)

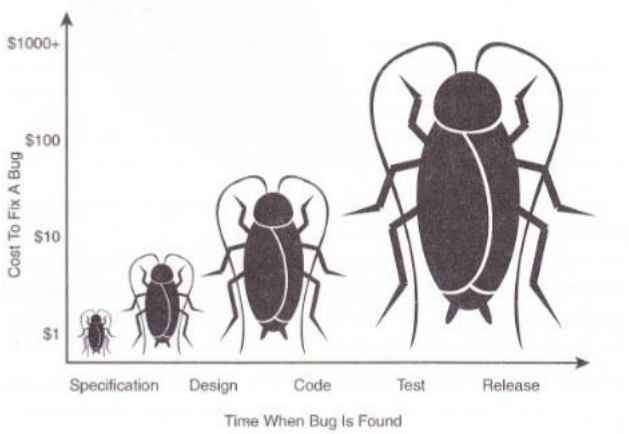
Ohjelmistoa voidaan testata joko liian vähän tai liian pitkään. Oikean testaustarpeen määrittäminen onkin haastavaa. Perry (2006, 47) kuvaa erään tietopalvelujohtajan sanoneen ”Too little testing is a crime, but too much testing is a sin.”. Hänen mukaansa liian vähäinen testaus on rikos, mutta liiallinen testaus on taas synti. On selvää, että liian vähäinen testaus jättää liikaa virheitä järjestelmään. Liiallinen testaus taas kuluttaa resursseja, kun ollaan tilanteessa, että ohjelmistossa ei ole juurikaan enää virheitä tai niiden vähäisten virheiden löytäminen vie enemmän resursseja kuin mitä on virheen korjaamisesta saatu

hyöty. (Perry 2006, 47.) Kuvassa 12 kuvataan löydettyjen virheiden ja testaukseen käytettyjen kustannuksien suhdetta. Kuvan optimitestauskohdassa testaukseen käytetyillä kustannuksilla ja löydetyillä virheillä on paras hyötysuhde. Tämän pisteen jälkeen testauksen kustannukset kasvavat ja löydettyjen virheiden määrä laskee.



KUVA 12. Testauksen kustannuksien ja löydettyjen virheiden suhde (Perry 2006, 48)

IBM:llä tehty tutkimus osoittaa, että virheiden korjauskustannukset nousevat mitä myöhemmin virhe havaitaan ja korjataan. Tutkimukset osoittavat, että on ainakin 10 kertaa kalliimpaa korjata virhe vasta ohjelmiston toteuttamisen jälkeen kuin jos virhe olisi havaittu ja korjattu ennen toteuttamisvaihetta. Jos virhe löydetään ja korjataan vasta tuotantovaiheessa, on virhekorjaus jo 100 kertaa kalliimpaa kuin virheen korjaaminen ennen ohjelmiston toteuttamista. (Perry 2006, 48.) Kuva 13 visualisoi virheiden, ”bugien”, korjauskustannuksien nousua mitä myöhemmin virhe löydetään.



KUVA 13. Virheen korjauskustannuksien kasvu ajan myötä (Patton 2006, 18)

Usein pääasiallinen testaus kuitenkin tapahtuu vasta ohjelmiston toteuttamisen jälkeen. Testaus voi tällöin kuluttaa jopa 50 % koko ohjelmiston kehitysbudjetista. Virheiden havaitseminen vasta ohjelmiston toteuttamisen jälkeen tarkoittaa, että virheeseen kuluu kustannuksia neljässä eri vaiheessa. Ensimmäinen kuluerä tulee, kun virhe toteutetaan ohjelmistoon joko väärin kirjoitetun määrittelyn tai suunnitteludokumentin muodossa tai ohjelmistoa toteutettaessa. Toinen kuluerä tulee, kun ohjelmistoa testataan ja virhe löydetään ja raportoidaan. Kolmas kuluerä on virheellisen määrittelyn tai ohjelmistokoodin paikallistaminen ja korjaus. Viimeinen, neljäs kuluerä, on ohjelmiston uudelleentestaus korjauksen jälkeen. Tutkimuksien mukaan peräti 64 % virheistä tapahtuu jo ohjelmiston suunnitteluvaiheessa, ja vain 36 % ohjelmistoa toteutettaessa. (Perry 2006, 52-53.)

Ennen ohjelmiston toteuttamista virheitä voidaan löytää käyttäen staattisia testausmenetelmiä, kuten dokumenttien ja ohjelmistokoodin katselmointeja. Vaatimusmäärittelyvaiheessa virheitä voidaan eliminoida varmistamalla, että määritellyt vaatimukset vastaavat asiakkaan tarpeita. Suunnittelu- ja ohjelmointivaiheessa voidaan varmistaa, että suunnitelma ja toteutettu ohjelmisto toteuttaa määritetyt vaatimukset. Tämän jälkeen alkaa ohjelmiston dynaaminen testaus, jossa tavoitteena on virheiden löytäminen.

4.1 Yksikkötestaus

Yksikkötestauksen tarkoituksena on varmistaa, että jokainen ohjelmistokomponentti, esimerkiksi luokka tai moduuli, toimii irrallisena kokonaisuutena (Watkins & Mills 2011, 48). Yksikkötestaus suoritetaan siis ennen kuin komponentit integroidaan isommiksi toimintakokonaisuuksiksi. Yksikkötestauksessa löydettyjen virheiden identifioiminen juuri tiettyyn komponenttiin on helpompaa, mikä siten nopeuttaa ja helpottaa virhekorjausta. (Naik & Tripathy 2008, 51.) Yksikkötestausta varten joudutaan usein rakentamaan keinotekoinen testiympäristö, koska komponentin testaukseen tarvittavat ympäröivät komponentit eivät useinkaan ole vielä valmiit (Tamres 2002, 220). Yksikkötestauksen suorittaa tyypillisestä komponentin ohjelmoinnista vastaava kehittäjä tai kehitystiimi (Watkins & Mills 2011, 48). Yksikkötestausta suoritetaan sekä lasi- että mustalaatikkotestausmenetelmin (Tamres 2002, 220).

4.2 Integraatiotestaus

Integraatiotestauksen tavoitteena on varmistaa ohjelmiston komponenttien yhteistoiminta. Integraatiotestaus etenee iteratiivisesti integroiden ja testaten komponentteja kohti isompaa ohjelmistokokonaisuutta. Uuden komponentin lisäyksen jälkeen alkaa integraatiotestaus, jolla varmistetaan, että ohjelmisto toimii edelleen. Hyväksytyn integraatiotestauksen jälkeen lisätään jälleen uusi komponentti. Integrointia ja testausta jatketaan iteratiivisesti, kunnes kaikki komponentit on integroitu ja integraatiotestit on suoritettu. Integraatiotestauksessa saadaan nopeasti palaute komponenttien yhteentoimivuudesta. Myös havaittujen virheiden paikallistaminen oikeaan komponenttiin on yleensä nopeaa, sillä mitä todennäköisemmin havaittu virhe liittyy juuri viimeksi lisättyyn komponenttiin. (Tamres 2002, 221.)

Komponenttien integraatioita voidaan toteuttaa eri tekniikoilla. Seuraavaksi kuvataan muutamia vaihtoehtoisia tekniikoita.

- Bottom-up
- Top-down
- Sandwich
- Big-bang

Tekniikan nimi kuvaa sitä, missä järjestyksessä ohjelmiston komponentteja integroidaan ja testaan koko ohjelmiston rakenteeseen nähden. **Bottom-up** -tekniikassa integroidaan ja testataan komponentteja iteratiivisesti lähtien alhaalta kohti ylemmän tason komponentteja ja **top-down** -tekniikassa päinvastoin, lähtien ylhäältä kohti alemman tason komponentteja. **Sandwitch** -tekniikassa yhdistetään kaksi edellistä integraatio- ja testaustekniikkaa, ja aloitetaan integraatio sekä alhaalta että ylhäältä ja edetään kohti keskustaa. **Big-bang** –tekniikassa integroidaan ensin kaikki komponentit ja vasta sen jälkeen suoritetaan koko ohjelmistolle integraatiotestaus. (Tamres 2002, 221.) Käytännössä tämä lähestymistapa tarkoittaakin ohjelmiston järjestelmätestausta. Etuna Big-bang -testauksessa on se, että silloin ei tarvitse rakentaa keinotekoista testiympäristöä komponenttien välisten viestien testaamiseen. Testaus käynnistyy vasta, kun kaikki komponentit on toteutettu ja integroitu ohjelmistokokonaisuudeksi. Haittapuolena on se, että palaute kom-

ponenttien toimivuudesta tulee vasta kun kaikki komponentit on integroitu ja integraatiotestaus on suoritettu. Siten myös havaittujen virheiden kohdistaminen oikeaan komponenttiin on usein aikaa vievää. (Pezzè & Young 2008, 408.)

4.3 Järjestelmätestaus

Järjestelmätestauksessa tarkastelun kohteena on koko järjestelmä ja testauksen tuloksia verrataan ohjelmiston toiminnalliseen vaatimusmäärittelyyn. Järjestelmätestausta suunniteltaessa testitapaukset yleensä priorisoidaan vaatimusmäärittelyssä olevan prioriteettijärjestyksen mukaisesti. Näin varmistetaan, että tärkeimmät ja kriittisimmät testitapaukset suoritetaan ensin. (ISTQB Exam Certification 2016a.) Järjestelmätestauksesta vastaa testaustiimi, jonka järjestelmätestaajat eivät ole osallistuneet ohjelmiston kehitystyöhön. (Watkins & Mills 2011, 63). Mikäli ohjelmiston toteuttajat testaisivat järjestelmää, huomio keskittyisi herkästi niihin ohjelmiston osiin, jotka tiedetään jo toimiviksi. Ulkopuolisen testaajan ennakkoluulottomuus mahdollistaa laajemman testauksen ja siten myös suuremman virhemäärän havaitsemisen. Järjestelmätestaukseen kuuluu myös kenttätestaus ja hyväksymistestaus. (Haikala & Mikkonen 2011, 208.) Hyväksymistestauksen sisältöä ja tavoitteita kuvataan tarkemmin seuraavassa alaluvussa.

Järjestelmätestauksessa havaittujen virheiden korjaaminen on paljon kalliimpaa kuin vastaavan virheen löytäminen jo yksikkö- tai integrointitestauksessa. Järjestelmätestauksessa havaittujen virheiden korjauskustannuksia kasvattaa virheanalysoinnin kesto, korjaustarpeen mahdollinen kohdistuminen useampaan komponenttiin ja siten uudelleen testaukseen tarvittava aika. Virhekorjauksen jälkeen komponenteille pitäisi suorittaa yksikkötestit ja tarpeen vaatiessa myös integraatiotestit sekä uudelleen järjestelmätestaus. (Haikala & Mikkonen 2011, 208.)

Järjestelmätestauksessa testataan järjestelmän toiminnallisten ominaisuuksien lisäksi myös järjestelmän ei-toiminnallisia ominaisuuksia. ISTQB Exam Certification:n (2016b) mukaan ei-toiminnalliseen testaukseen kuuluvat seuraavat testausalueet:

- Luotettavuustestaus
- Käytettävyystestaus
- Tehokkuustestaus

- Ylläpidettävyystestaus
- Siirrettävyystestaus
- Lähtötilannetestaus
- Yhdenmukaisuustestaus
- Dokumentaation testaus
- Kestävyystestaus
- Kuormitustestaus
- Suorituskykytestaus
- Yhteensopivuustestaus
- Tietoturvatestaus
- Skaalautuvuustestaus
- Paljoustestaus
- Rasitustestaus
- Palautettavuustestaus
- Kansainvälistämis- ja lokalisaatiotestaus

Luotettavuustestauksessa varmistetaan järjestelmän virheetön toimivuus yhtämittaisessa pitkäkestoisessa käytössä tai kuinka järjestelmä toipuu virhetilanteista. **Käytettävyystestauksessa** testataan, vastaako järjestelmä loppukäyttäjien käyttötarpeita ja onko järjestelmä ylipäänsä käytettävä. **Tehokkuustestauksessa** tutkitaan, kuinka monta testitapausta pystytään suorittamaan tietyn ajan kuluessa. Testaustuloksien avulla voidaan optimoida järjestelmän palvelujen vasteaikoja. **Ylläpidettävyystestauksen** avulla selvitetään kuinka ylläpidettävä järjestelmä on. Käytännössä tämä tarkoittaa sitä, kuinka helposti järjestelmää pystyy analysoimaan, muuttamaan tai testaamaan. **Siirrettävyystestauksessa** testataan, kuinka helposti järjestelmä voidaan siirtää toiseen ympäristöön (esimerkiksi Windows 7:sta Windows 10:een). **Lähtötilannetestauksessa** varmistetaan testitapauksien pohjautuvuus määrittely- ja muihin dokumentteihin. **Yhdenmukaisuustestauksella** varmistetaan, että järjestelmä noudattaa yrityksen määrittämiä standardeja. **Dokumentaation testauksella** varmistetaan, että järjestelmän testauksesta on tehty muun muassa testitapauksien määrittelydokumentti, testaussuunnitelma, testiraportit ja virheraportit. (Tamres 2002, 222; ISTQB Exam certification 2016b.)

Kestävyystestauksessa testataan järjestelmän käyttöä erityisen isolla kuormituksella pidemmän ajanjakson ajan. Kestävyystestauksella varmistetaan, että järjestelmässä ei ole

esimerkiksi muistivuotoja, jotka aiheuttaisivat järjestelmän toimimattomuutta tai satunnaisia virhetilanteita. **Kuormitustesteillä** varmistetaan järjestelmän toimivuus oletetussa tai sitä suuremmassa kuormitustilanteessa. Kuormitustestauksella tunnistetaan järjestelmän maksimitoimintakapasiteetti ja pullonkauloja sekä voidaan analysoida, mikä heikentää järjestelmän toimintakykyä. **Suorituskykytestauksella** varmistetaan, että yhden tehtävän vasteaika normaali- ja huippukuormituksessa on määritetyn mukainen. **Yhteensopivuustestauksella** testataan, että järjestelmä toimii suunnitellussa ympäristössä. **Tietoturvatestauksessa** varmistetaan muun muassa, että vain sallitut käyttäjät pääsevät käyttämään järjestelmää. **Skaalautuvuustestauksessa** testataan järjestelmän kyvykkyyttä kasvattaa esimerkiksi kuormituksen kestoa, tapahtumien määrää tai datamääriä. **Paljoustestauksessa** testataan järjestelmän toimintakykyä suurella datamäärällä. **Rasitustestauksessa** asetetaan järjestelmään kohtuuton kuormitus, jotta nähdään kuinka järjestelmä reagoi erityistilanteissa. Tavoitteena on varmistaa, että järjestelmä toimii hallitusti myös poikkeustilanteissa. **Palautettavuustestauksessa** varmistetaan, että järjestelmä pystyy palautumaan vakavista virhetilanteista, kuten ohjelmiston kaatumisesta. **Kansainvälistämis- ja lokalisaatiotestauksessa** testataan järjestelmän toimivuutta suunnitelluilla kielillä ja alueilla. (Tamres 2002, 222; ISTQB Exam certification 2016b.)

4.4 Hyväksymistestaus

Hyväksymistestauksen tarkoitus on varmistaa, että järjestelmä toimii asiakkaan vaatimuksien mukaisesti. Hyväksymistestit suoritetaan useimmiten asiakkaan omassa ympäristössä. Tuloksien perusteella asiakas päättää onko järjestelmä valmis tuotantokäyttöön. (Tamres 2002, 223-224.) Hyväksymistestaus eroaa muista testausvaiheista sillä, että tässä vaiheessa ei ole tarkoitus enää löytää virheitä ja saada niihin korjauksia vaan varmistaa tietojärjestelmän soveltuvuus todellisiin käyttötilanteisiin. Hyväksymistestaus käytännössä alkaa uudelleen, mikäli hyväksymistestauksessa löydetään virheitä ja niihin tehdään korjauksia. (Pöyhönen 2015, 46.)

Hyväksymistestauksen voi suorittaa myös toimittaja omissa tiloissaan. Tämä ei kuitenkaan ole varsinainen järjestelmän hyväksymistestaus vaan tällä testauksella toimittaja varmistaa, että järjestelmä tulee läpäisemään asiakkaan myöhemmin suorittaman varsinaisen hyväksymistestauksen. (Naik & Tripathy 2008, 450-451.) Järjestelmä- ja hyväksymistestauksessa suoritetaan samankaltaisia testejä, kuitenkin näillä testaustasoilla on

eri tavoite. Pöyhösen (2015, 39) mukaan ”järjestelmätestaus keskittyy järjestelmään toteutuskokonaisuutena ja pyrkii varmistamaan kaiken kehityksen aikana tehdyn työn lopputuloksen”, kun taas ”hyväksymistestaus keskittyy todelliseen liiketoimintaprosessiin ja järjestelmän soveltuvuuteen todellisiin käyttötilanteisiin”. Nämä testaustasot eroavat myös testiympäristöltään. Järjestelmätestauksessa käytetään toimittajan tiloissa olevaa testiympäristöä, mutta hyväksymistestauksessa testaus suoritetaan yleensä asiakkaan lopullisessa käyttöympäristössä.

Onnistuneen hyväksymistestauksen suorittamiseksi on tärkeätä määritellä testaussuunnitelman lisäksi hyväksymistestauksen aloitus- sekä lopetuskriteerit. Toimittajan ja asiakkaan on tärkeätä sopia yhdessä nämä kriteerit, jotta hyväksymistestausvaiheessa ei tule epäselvyyttä milloin testaus voidaan aloittaa tai millä perusteella tietojärjestelmän hyväksymistestaus voidaan hyväksytysti lopettaa.

4.4.1 Hyväksymistestauksen aloitus- ja lopetuskriteerit

Vaatimusmäärittelyn sisältämien toiminnallisten vaatimusten testaus kuuluu toimittajan suorittamaan järjestelmätestaukseen (Naik & Tripathy 2008, 453). Asiakkaan suorittama hyväksymistestaus kohdistuu myös toiminnallisten vaatimuksien testaamiseen, mutta tavoite on eri, kuten edellisessä luvussa todettiin. Ennen hyväksymistestauksen aloittamista, asiakkaan on hyvä varmistaa, että toimittaja on testannut toiminnalliset vaatimukset kattavasti omassa järjestelmätestauksessaan. Testausvastuut ja raportointikäytännöt on siis hyvä sopia jo sopimusvaiheessa. Hyväksymistestauksen aloittamiseen liittyy paljon myös muita valmisteluja, jotka on hyvä käydä läpi ennen testauksen käynnistämistä.

Pöyhösen (2015, 61) mukaan hyväksymistestaus voi alkaa, kun seuraavat aloituskriteerit täyttyvät:

- Virheraportointikäytännöstä on sovittu ja raportointiin käytettävä järjestelmä on käytettävissä
- Hyväksymistestausympäristö on valmis käytettäväksi ja testaajilla on siihen pääsy
- Kaikki määritetyt ominaisuudet on toteutettu ja toimittaja on tehnyt testauksen kaikille ominaisuuksille
- Virhekorjaukset on tehty ja kriittisiä, testauksen estäviä virheitä ei ole tiedossa

- Ohjelmistokoodi on versionhallinnan alla
- Toimittaja on tehnyt toimitusselosteen, johon on dokumentoitu julkaisun sisältö ja tiedossa olevat rajoitteet
- Sovellus on asennettu hyväksymistestiympäristöön ja sovittu aloitustesti on hyväksytysti suoritettu.

Hyväksymistestauksen lopetuskriteerien määrittämisessä lähtökohtana on "Mitkä kriteerit tietojärjestelmän pitää täyttää, jotta se voidaan hyväksyä tuotantokäyttöön?" Hyvä ohje lopetuskriteerien määrittämiseen on myös se, että kriteerien pitää olla vähintään mitattavissa, mutta mielellään myös laskettavissa. (Naik & Tripathy 2008, 451-452.)

Pöyhönen (2015, 63) määrittelee hyväksymistestauksen lopetuskriteereiksi seuraavat kohdat:

- Kriittiset ja vakavat virheet on korjattu
- Testauksesta raportoidut havainnot on käsitelty ja sovitut korjaukset on tehty
- Tuotantoon siirrettävälle julkaisulle on suoritettu kaikki suunnitellut testit ilman uusia havaintoja
- Testitulokset ja arvioidut riskit on esitelty johtoryhmälle.

4.4.2 Testaussuunnitelma

Hyväksymistestaussuunnitelman tarkoituksena on kuvata kuinka tietojärjestelmä tullaan testaamaan ennen tuotantokäyttöönottoa. Usein sopimusvaiheessa sovitaan, että toimittaja tuottaa myös hyväksymistestaussuunnitelman, jonka mukaisesti toimittaja suorittaa oman hyväksymistestauksensa ennen asiakkaan suorittamaa varsinaista hyväksymistestausta. Hyväksymistestaussuunnitelman tehtävä on varmistaa, että asiakkaan vaatimukset täyttyvät. Siksi suunnitelman kirjoittamiseen ja testauksen suorittamiseen on hyvä valita eri taustan omaavia henkilöitä, kuten laatuinsinöörejä, yrityksen liikekumppaneita ja asiakastuesta vastaavia henkilöitä. Asiakkaan suorittama hyväksymistestaus ei yleensä kestä kovin pitkään, isoillekin järjestelmille usein maksimissaan 6 viikkoa. (Naik & Tripathy 2008, 461-463.)

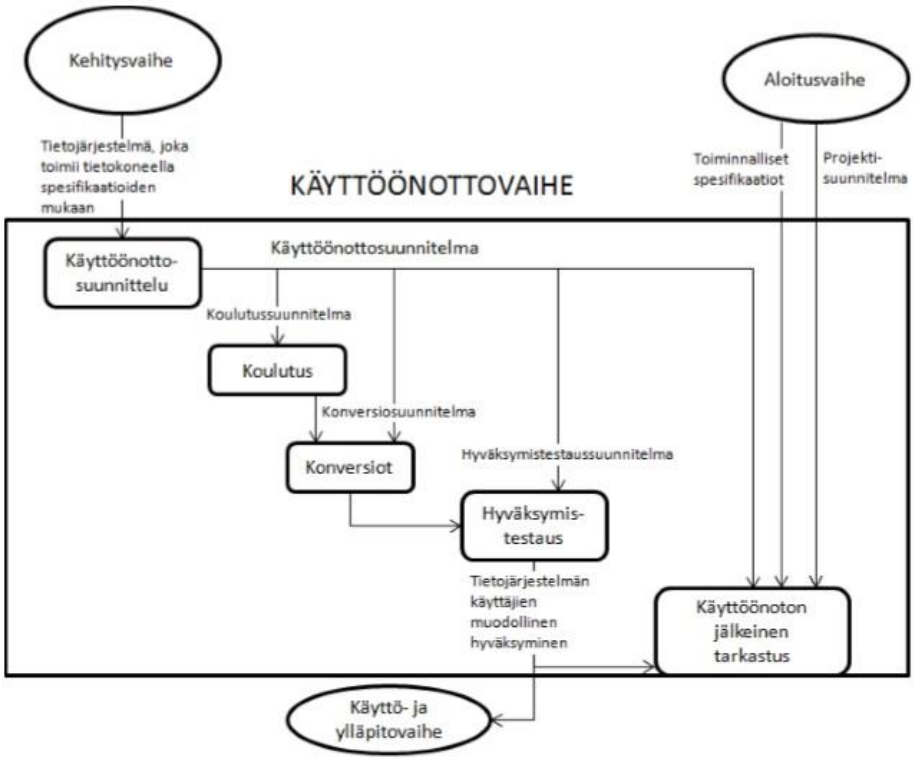
Hyväksymistestauksessa on tärkeätä varmistaa myös tietojärjestelmän ei-toiminnallisten vaatimuksien toimivuus. Naik & Tripathy (2008, 455) nostavat esille muun muassa tietojärjestelmän käytettävyyden, suorituskyvyn ja luotettavuuden testaamisen osana hyväksymistestausta. Hyväksymistestauksessa olisi tärkeätä varmistaa ”Kuinka helppoa on tietojärjestelmää käyttää tai kuinka helppoa on oppia sitä käyttämään? Onko tietojärjestelmän käyttöliittymä ja ohjeistus selkeä loppukäyttäjälle? Ovatko tietojärjestelmän toimintamallit loogisia loppukäyttäjälle?”. Hyvänkin tietojärjestelmän käytettävyyden voi pilata puutteet suorituskyvyssä. Jos tietojärjestelmän odotetaan palvelevan tiettyinä ajankohtina huomattavasti normaalikäyttöä suurempia käyttäjämääriä, on tärkeätä varmistaa suorituskyky hyväksymistestauksessa suorituskyky- tai rasitustestauksella. Tietojärjestelmän luotettavuutta määritellään sillä, kuinka pitkään tietojärjestelmä toimii virheettä. Mitä pitempään tietojärjestelmä toimii, sitä luotettavampi tietojärjestelmä on. Tietojärjestelmän virhetahtia kuvataan termillä MTBF, Mean time between failure eli virheiden esiintymisen välinen ajanjakso. Asiakas voi olla valmis hyväksymään useampia lievempiä virhetilanteita tietojärjestelmän toimivuudessa, mutta harvemmin ainuttakaan kriittistä virhettä. (Naik & Tripathy 2008, 455-456.)

Naikin ja Tripathyn (2008, 463-464) mukaan hyväksymistestauksen aloittaminen ja suoritus vaatii tietojärjestelmän kehittäjien saatavilla oloa. Ennen kuin asiakas aloittaa hyväksymistestauksen, toimittajan tulee huolehtia tietojärjestelmän kouluttamisesta asiakkaalle. Tietojärjestelmän kehittäjät käyvät myös läpi järjestelmätestauksen testitulokset asiakkaan hyväksymistestaajien kanssa. Tämän perusteella voidaan hienosäätää asiakkaan suorittaman hyväksymistestauksen painopisteitä. Lisäksi hyväksymistestauksen aikana tietojärjestelmän kehittäjien ja asiakkaan hyväksymistestaajien tulee tiiviissä yhteistyössä koordinoida testauksessa löydettyjen virheiden analysointi, virhekorjaus ja uudelleentestaus. Hyväksymistestauksen aikana tietojärjestelmän kehittäjät tukevat ja auttavat asiakkaan hyväksymistestaajia hyväksymistestien suorittamisessa. Hyväksymistestausvaiheessa havaitut virheet raportoidaan toimittajalle. Toimittaja korjaa raportoidut virheet ja testaa uudelleen ohjelmistoversion. Tämän jälkeen asiakas voi jatkaa omaa hyväksymistestaustaan. (Naik & Tripathy 2008, 464.) Hyväksymistestaus päättyy kun kaikki hyväksymistestauksen lopetuskriteerit on täytetty.

4.4.3 Käyttöönotto

Aikkilan & Saukon (2012, 6) mukaan ”Tietojärjestelmän käyttöönotolla tarkoitetaan uuden tietojärjestelmän säännönmukaisen käytön aloittamista tai vanhan järjestelmän toimintojen siirtämistä sen korvaavalle järjestelmälle.” Myös uuden ohjelmistoversion päivitys tuotantokäyttöön vaatii vastaavia käyttöönottovalmisteluja.

Tietojärjestelmien käyttöönotto on usein varsin haastava tehtävä yrityksissä. Koko tietojärjestelmän käyttöönottoprosessi, toteuttamisesta käyttöönottoon, kestää usein vuosia. Monimutkaisuutta käyttöönottoon tuo se, että prosessiin osallistuu useita eri tahoja erilaisin näkökulmin ja tehtävänkuvin. Tietojärjestelmän käyttöönottoprosessi onkin kokonaisuudessaan monivaiheinen ja monimutkainen prosessi. (Hyötyläinen & Kalliokoski 2001, 17, 20.) Itse käyttöönottovaiheeseen (kuva 14) voidaan lukea kuuluvaksi seuraavat vaiheet: käyttöönoton suunnittelu, loppukäyttäjien koulutus, datakonversiot vanhasta järjestelmästä uuteen järjestelmään, hyväksymistestaus ja uuden järjestelmän toimivuuden seuranta. Käyttöönottosuunnitelmassa määritetään muun muassa loppukäyttäjien koulutustarve ja -aikataulu, kuinka siirretään data olemassa olevasta järjestelmästä uuteen järjestelmään, mikä on viestintästrategia ja mikä on tietojärjestelmän käyttöönottoajankohta. (Aikkila & Saukko 2012, 8-9.)



KUVA 14. Tietojärjestelmän käyttöönoton vaiheet (Aikkila & Saukko 2012, 9)

Tietojärjestelmän käyttöönotto voidaan toteuttaa eri tavoin. Aikkilan & Saukon (2012, 6) mukaan käyttöönottoja voidaan toteuttaa neljällä eri tavalla: rinnakkainen siirtymä, vaiheittainen siirtymä, pilotointi ja suora siirtymä. *Rinnakkainen siirtym*ä tarkoittaa tilannetta, missä vanha ja uusi tietojärjestelmä toimivat jonkin ennalta määritetyn ajanjakson ajan rinnakkain. Tämä tuo turvallisuutta käyttöönoton onnistumiseen, mutta on myös kallis tapa. *Vaiheittaisessa siirtymässä* tietojärjestelmän osat otetaan käyttöön eri aikaan. Tämä on yleensä mahdollista isoissa järjestelmissä, joissa toisistaan erillään olevien toimintojen erilliskäyttö on mahdollista. *Pilotointiin* perustuvassa käyttöönotossa tietojärjestelmä otetaan ensin käyttöön vain osassa yritystä ja käyttökokemuksien myötä laajennetaan myöhemmin käyttöönotto koskemaan koko organisaatiota. *Suorassa siirtymässä* siirrytään välittömästi uuden tietojärjestelmän valmistuttua uuden tietojärjestelmän käyttöön, ja samalla vanha tietojärjestelmä poistetaan käytöstä. (Aikkila & Saukko 2012, 6.)

Tietojärjestelmän käyttöönottoprosessi on sitä haastavampi, mitä laajemmin tietojärjestelmän käyttöönotto vaikuttaa organisaation toimintaan. Avainasemassa onnistuneessa käyttöönottoprojektissa onkin projektiryhmä, joka vie prosessien ja järjestelmän vaatimaa muutosjohtamista eteenpäin muulle organisaatiolle. (Aikkila & Saukko 2012, 12.)

5 TUTKIMUKSEN TOTEUTTAMINEN JA TULOKSET

Luvussa viisi kuvataan miten teemahaastattelut käytännössä toteutettiin, miten haastatteluaineistoa käsiteltiin ja analysoitiin sekä minkälaisia päätelmiä haastatteluaineistosta tehtiin. Alaluvuissa 5.3 ja 5.4 käydään tarkemmin läpi tutkimuksessa havaittuja kehityskohteita sekä niihin muodostettuja ratkaisuehdotuksia.

5.1 Teemahaastattelut

Haastateltaviksi valittiin yhdeksän tietojärjestelmien pääkäyttäjää ja kaksi projektipäällikköä. Kaiken kaikkiaan Laureassa on noin neljäkymmentä erilaisten tietojärjestelmien pääkäyttäjää, ja Laurean tietohallinnossa on kolme projektipäällikköä tämän opinnäytetyön tekijä mukaan lukien. Tutkimukseen valikoitiin mukaan yhdeksän pääkäyttäjää sillä perusteella, että saatiin tutkimusaineistoa tietojärjestelmistä, joihin tulee automaattisesti versiopäivityksiä, ja tietojärjestelmistä, joihin Laurean asiantuntijat itse tilaavat uusia versioita. Lisäksi valintakriteerinä painotettiin sitä, että tietojärjestelmä on laajasti Laureassa käytössä. Luontevinta oli haastatella kahta projektipäällikköä, jotta tutkimukseen saatiin aineistoa myös heidän näkökulmastaan.

Haastattelujen pohjaksi luotiin kyselylomakkeet, oma versio pääkäyttäjille (liite 1) ja projektipäälliköille (liite 2). Haastattelut toteutettiin puolistrukturoidulla teemahaastattelumenetelmällä. Kyselomakkeet muodostettiin kattamaan tutkimuksen kohteena olevat teema-alueet. Kysymykset koskivat muun muassa toimittajan vastuulla olevaa tietojärjestelmän testausta, Laurean omia testauskäytäntöjä ja tietojärjestelmän tuotantokäyttöönottoa. Teemahaastattelujen tavoitteena oli selvittää näiden edellä mainittujen teema-alueiden nykyinen tilanne ja tämän hetken käytännöt. Kyselylomakkeet lähetettiin haastateltaville hyvissä ajoin ennen haastattelua, jotta he pystyivät valmistautumaan itse haastatteluun. Kyselylomaketta ei pyydetty kuitenkaan täyttämään ja palauttamaan ennen haastattelutilaisuutta, vaan haastateltavia ohjeistettiin ainoastaan tutustumaan halutessaan haastattelun aiheisiin kyselylomakkeen avulla.

Haastattelut suoritettiin pääosin kampuksilla. Ainoastaan yksi haastattelu suoritettiin hyödyntäen videoneuvotteluohjelmaa. Kullekin haastattelulle varattiin tunti aikaa, ja se tuntui

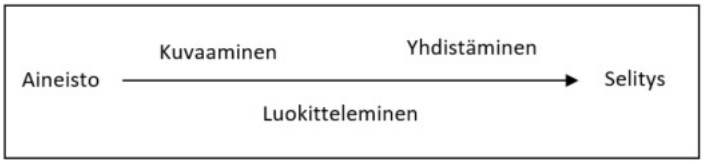
riittävän hyvin kaikkien haastateltavien kanssa. Kaikki haastattelut tallennettiin. Pääkäyttäjille suunnattua haastattelulomaketta täydennettiin ensimmäisen kolmen haastattelun jälkeen toimittajan testauskäytäntöihin liittyvillä kysymyksillä. Tästä aiheesta eräältä haastateltavalta pyydettiinkin jälkeenpäin vielä sähköpostitse täydennyksiä. Tutkimuksen kannalta saatiin riittävä otos toimittajateemasta, koska kuuden (6) pääkäyttäjän ja kahden (2) projektipäällikön haastatteluissa toimittajan testauskäytännöt olivat yksi keskeinen teema-aihe.

5.2 Haastatteluaineiston litterointi ja analysointi

Aineisto voidaan purkaa joko sanatarkasti puhtaaksikirjoittamalla eli litteroimalla tai se voidaan purkaa valikoiden, teema-alueita koodaten. Tutkimustehtävästä riippuen, on hyvä miettiä, kuinka tarkkaan litterointiin on syytä ryhtyä. (Hirsjärvi & Hurme 2004, 138-139.) Laadullisen aineiston analyysin tavoitteena on selkiyttää aineistoa ja tuottaa siten uutta tietoa tutkittavasta asiasta. Analyysin tarkoitus on aineiston tiivistäminen ilman, että kadotetaan aineiston sisältämää informaatiota. Oikeastaan analyysin tarkoitus on juuri päinvastainen eli tavoitteena on informaatioarvon kasvattaminen muodostamalla hajanaisesta aineistosta selkeää ja mielekästä. Usein kaikkein haastavin vaihe laadullisessa tutkimuksessa on juuri aineiston analyysi. (Eskola & Suoranta 2000, 137.)

Tässä tutkimuksessa haastatteluaineistoa ei litteroitu tallenteista sanatarkasti puhtaaksikirjoittamalla vaan kerätty aineisto purettiin ryhmitellen eri aihealueisiin. Haastateltavat käsiteltiin anonyymisti, ja haastattelujen tallenteet hävitettiin aineiston analysoinnin jälkeen. Aineisto purettiin suoraan tietokoneelle. Yhden tunnin haastattelun purkamiseen kului keskimäärin kaksi tuntia. Hirsjärven, Remeksen ja Sajavaaran (2013, 223) kuvaama vaiheittaisesti etenevä analyysi (kuva 15) vastaa hyvin tässä tutkimuksessa käytettyä analysointitekniikkaa. Lähtökohtana analysoinnissa voidaan pitää myös aineiston tematisointia (Eskola & Suoranta 2000, 174). Tallenteista kirjattiin (kuvaaminen) ensin teemahaastattelun eri aihealueiden aineisto. Ensimmäisessä työvaiheessa karsittiin pois tutkimusaiheeseen kuulumaton aineisto. Poistettavaa aineistoa oli vähän, sillä haastattelut pysyivät hyvin sovitussa aihepiirissä. Tämän jälkeen aineistoa analysoitiin edelleen, ja haastateltavien kommentteja ryhmiteltiin (luokiteltiin) tarkemmin eri aihealueisiin. Näin kyettiin havainnoimaan, kuinka moni haastateltava toi esille samoja asioita kyseisistä aihe-

alueista. Seuraavaksi eri aihealueita ja niihin liittyvää tutkimusaineistoa yhdistettiin isommiksi teemakokonaisuuksiksi. Lopuksi tutkimusaineiston analyysiä jatkettiin muodostamalla päätelmiä teemakokonaisuuksista (selitys) ja määrittämällä kehityskohteita. Seuraavassa luvussa kuvataan tutkimuksessa havaittuja kehityskohteita.



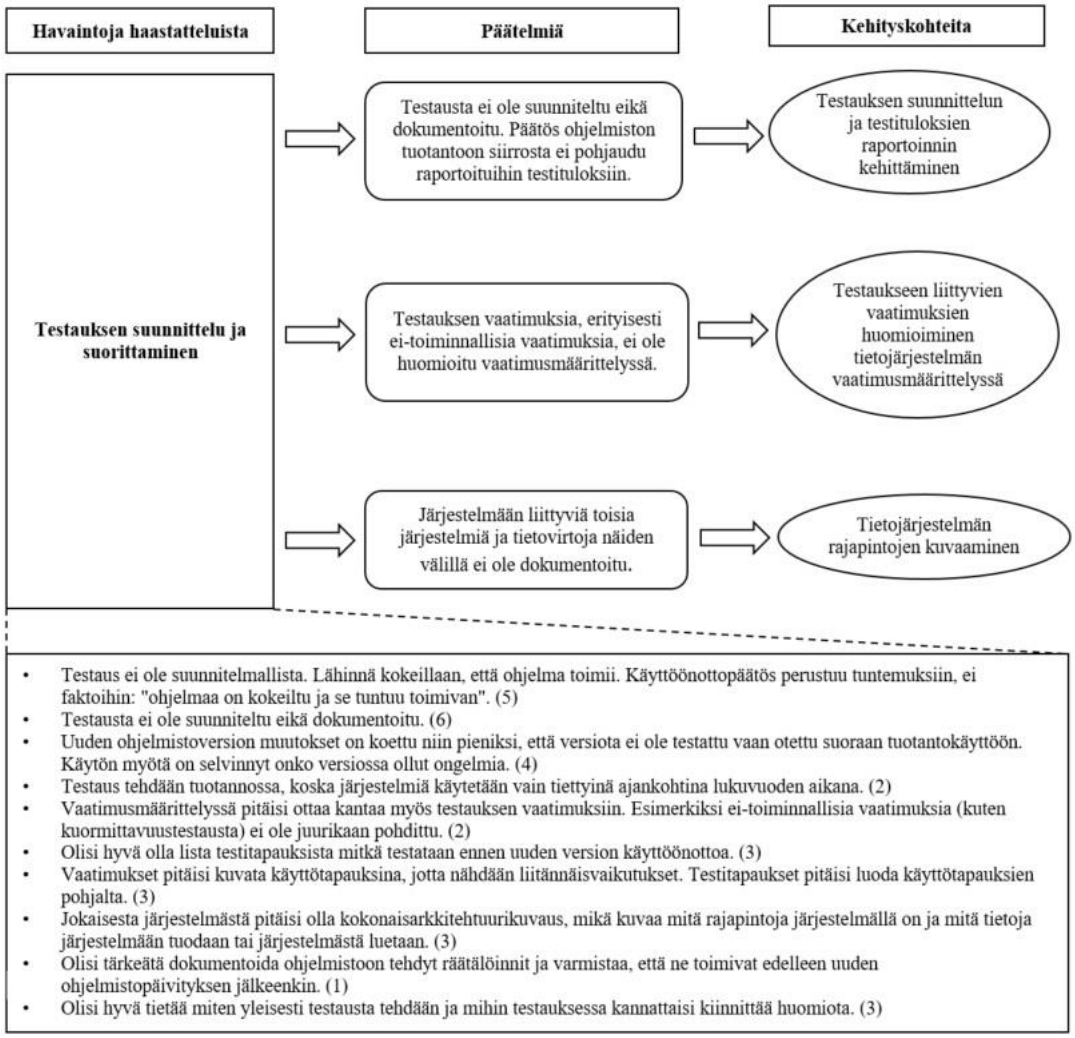
KUVA 15. Analyysi vaiheittaisesti etenevänä (Hirsjärvi, Remes & Sajavaara 2013, 223)

5.3 Tutkimuksessa havaitut kehityskohteet

Tutkimusaineiston analysoinnissa haastatteluaineistosta muodostuivat seuraavat teemakokonaisuudet: 1) Testauksen suunnittelu ja suorittaminen, 2) Testiympäristön käyttö testauksessa, 3) Uuden tietojärjestelmän tai ohjelmistoversion käyttöönotto sekä 4) Toimittajan suorittama testaus. Haastatteluissa tehdyt havainnot jaoteltiin oikean teemakokonaisuuden alle. Jokaisen havainnon yhteyteen merkittiin numeroin se, kuinka moni haastateltava toi saman asian esille haastattelussa. Haastatteluissa tuli esille myös useampia yksittäisiä havaintoja. Toisaalta useasta haastattelusta nousi esiin samoja havaintoja. Kuvissa 16-19 on listattu teemakokonaisuudet, niihin liittyvät havainnot lukumäärineen, havainnoista tehdyt päätelmät ja päätelmistä muodostetut kehityskohteet. Kuvien havaintolistaukseen on otettu mukaan vain tärkeimmiksi analysoidut havainnot, joiden katsotaan vaikuttavan tutkimuskysymyksiin. Kaikki haastatteluissa tehdyt havainnot on listattu liitteessä 3. Seuraavissa alaluvuissa käsitellään tarkemmin kuhunkin teemakokonaisuuteen liittyvät havainnot, päätelmät ja kehityskohteet.

5.3.1 Testauksen suunnittelu ja suorittaminen

Haastatteluaineistoa analysoitaessa testauksen suunnitteluun ja suorittamiseen liittyi eniten havaintoja, joiden pohjalta voitiin tehdä päätelmiä ja muodostaa kehityskohteita. Kuvassa 16 listataan tähän teemakokonaisuuteen liittyvät haastatteluhavainnot, päätelmät sekä kehityskohteet.



KUVA 16. Testauksen suunnittelu ja suorittaminen

Kaikissa 11 haastattelussa tuli esille yksi selkeä havainto: ennen käyttöönottoa tietojärjestelmille tehtyä testausta ei suunnitella eikä suoritettuja testejä dokumentoida. Testaus on lähinnä ohjelmiston summittaista käyttämistä. Haastateltavien sanoin, ”ohjelmaa on kokeiltu ja se tuntuu toimivan”. Tästä voidaan päätellä, että tietojärjestelmän käyttöönottopäätös perustuu ennemminkin tuntemuksiin tietojärjestelmän toimivuudesta, kuin suunnitelmalliseen testaukseen ja siihen pohjautuvaan testituloksien raportointiin. Kuten luvussa 3.5 todetaan, testauksen kattavuutta ja löydettyjen virheiden määrää voidaan tarkastella ainoastaan testauksesta muodostettujen dokumenttien avulla. Ilman kattavaa dokumentaatiota testauksen tuloksista, on vaikea tehdä päätöstä onko tietojärjestelmä riittävän laadukas julkaistavaksi tuotantokäyttöön. Testauksen suunnittelemattomuus tarkoittaa sitä, että kukin testaukseen osallistuva henkilö kokeilee tietojärjestelmän toimivuutta omasta näkökulmastaan ja käyttökokemuksestaan lähtien. Loppukäyttäjän kannalta kaikki oleelliset käyttötapaukset eivät välttämättä tule siten testatuksi ennen käyttöönottoa. Lisäksi näin toteutettu testaus ei ole identtisesti toistettavissa. Ilman täsmällistä ja

yksityiskohtaista testausohjedokumentaatiota – mitä testataan, millaisin käyttöoikeuksin ja missä ympäristössä - testausta on vaikeata toistaa samankaltaisesti kerrasta toiseen.

Kuusi haastateltavaa kertoi, että tietojärjestelmän uutta ohjelmistoversiopäivitystä ei testata ollenkaan ennen käyttöönottoa vaan uusi versio otetaan suoraan tuotantokäyttöön ja vasta käyttökokemukset näyttävät, onko versiossa ongelmia vai ei. Haastatteluissa tuli esille kaksi syytä, miksi testausta ei ole tehty ennen tuotantokäyttöönottoa. Ohjelmistoversion muutokset on koettu niin pieniksi, että version testaamista ei ole pidetty tarpeellisena ennen käyttöönottoa tai tietojärjestelmää käytetään vain tiettyinä aikoina lukuvuodesta. Muina ajankohtina versiopäivityksiä voi vastaavasti testata turvallisesti suoraan tuotantoympäristössä. Pienillä ja vähäpätöisiltä kuulostavilta ohjelmistomuutoksilla voi kuitenkin olla sivuvaikutuksia toimintoihin, joihin näiden ei olettaisi vaikuttavan. Testauksen suorittamista käsittelevässä luvussa 3.4 tuotiinkin esille, että täysin virheettömän ohjelmiston saavuttaminen on lähes mahdoton tehtävä, ja että testauksella pyritään varmistamaan, että ohjelmisto toimii hyvin yleisissä loppukäyttäjien käyttötapauksissa. Ongelmien havainnointi vasta tuotantokäytössä aiheuttaa yleensä lisäksi yllättäviä käyttökatkoksia loppukäyttäjille. Tästä voidaan havaita, että jokaisen tietojärjestelmän pääkäyttäjän olisi hyvä määritellä sellaiset testitapaukset, jotka varmistavat loppukäyttäjän perustoimintojen toimivuuden. Nämä testitapaukset toimisivat tuotantoon siirron hyväksymistestitapauksina, joiden toimivuus olisi tärkeätä testata aina ennen uuden ohjelmistoversion käyttöönottoa. Kolme haastateltavaa esittikin kehitysideana hyväksymistestitapauksien määrittelyn ja käytön.

Käyttötapauksiin perustuva testitapaussuunnittelu nousi kehitysideana esiin kolmessa eri haastattelussa. Testitapauksia suunniteltaisiin siis vaatimusmäärittelydokumenttiin pohjautuen niin, että testitapaukset kuvaavat loppukäyttäjien tietojärjestelmässä tekemiä toimintoja. Tällä tavoin varmistutaan siitä, että ohjelmisto vastaa loppukäyttäjien tarpeita. Luvussa 3.2.2 käyttötapauksien taustaa käsiteltäessä, nostettiin yhdeksi käyttötapauksien eduksi se, että käyttötapaukset auttavat myös kommunikoinnissa tietojärjestelmän toimittajan kanssa. Toimittajan on helpompi ymmärtää, mitä esimerkiksi jokin haluttu muutos konkreettisesti tarkoittaa loppukäyttäjälle, jos se kuvataan käyttötapauksena loppukäyttäjän näkökulmasta. Tästä voidaankin päätellä, että testitapaussuunnittelun olisi hyvä pohjautua käyttötapauksiin kuvaten siten oikeita loppukäyttäjien toimintoja.

Tietojärjestelmää hankittaessa ja tietojärjestelmän vaatimusmäärittelyä rakennettaessa, testaukseen liittyvät vaatimukset tuntuvat haastattelujen perusteella jäävän usein vähälle huomiolle. Erityisesti tietojärjestelmän ei-toiminnallisia vaatimuksia ei ole juurikaan pohdittu, eikä näitä myöskään ole huomioita vaatimusmäärittelyssä. Vaikkakin vain kaksi haastateltavaa toi esille tämän puutteen, on tämä aiheen tärkeyden vuoksi hyvä ottaa mukaan kehityskohteisiin ja varmistaa, että testaukseen kiinnitetään jatkossa paremmin huomiota jo uuden tietojärjestelmän vaatimusmäärittelyvaiheessa.

Tietojärjestelmän riippuvuus mahdollisiin muihin tietojärjestelmiin nousi esille kolmessa haastattelutilanteessa. Tällä tarkoitettiin kuvausta siitä, mitä rajapintoja tietojärjestelmällä on muihin järjestelmiin, ja mitä tietoja rajapintojen kautta tietojärjestelmään joko talletetaan tai tietojärjestelmästä luetaan. Tietojärjestelmän rajapintojen ja niiden tietovirtojen tunteminen on tärkeätä tietojärjestelmän pääkäyttäjälle. Ilman tätä tietämystä, pääkäyttäjä ei voi tietää mihin mikäkin suunniteltu muutos tietojärjestelmässä vaikuttaa ja miten muutos mahdollisesti näkyy loppukäyttäjille. Olisi siis tärkeätä dokumentoida tietojärjestelmän rajapinnat ja niihin liittyvät tietovirrat. Tämä mahdollistaisi muun muassa laadukkaamman tietojärjestelmien muutoksien hallinnan.

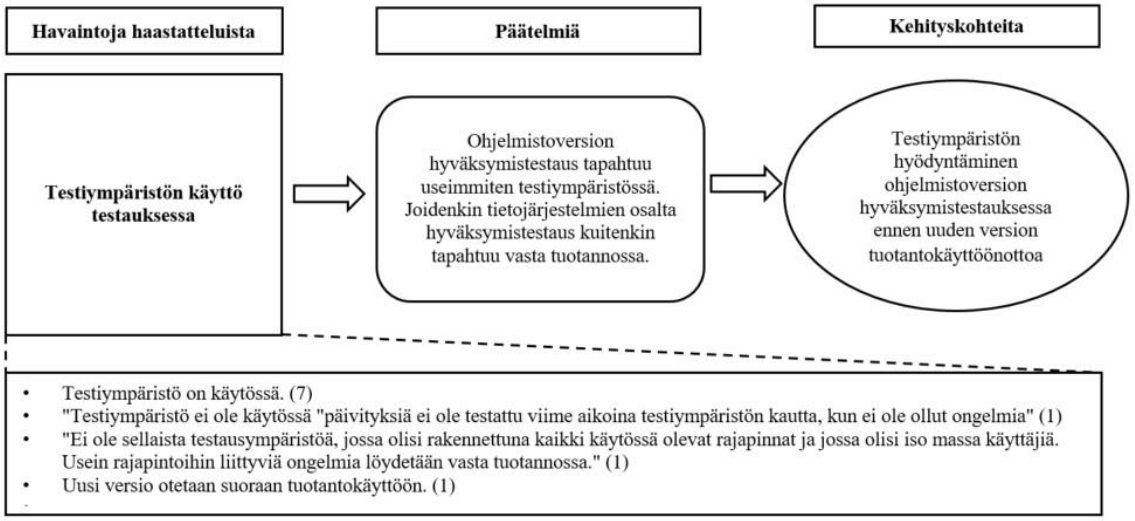
Kolmessa haastattelussa tuli esille tarve saada lisätietoa testauksen suorittamisesta ja sen merkityksestä tietojärjestelmien käyttöönotossa. Kuten jo tämän luvun alusta voidaan päätellä, testauskulttuuri ei ole kovin vahva tietojärjestelmien käyttöönotossa tai ohjelmistoversioiden päivityksessä. Voidaan päätellä, että testauksen suunnittelusta ja sen toteuttamisesta ei tiedetä riittävästi. Yleisen testaustietämyksen kasvattaminen onkin yksi selkeä tavoite tälle opinnäytetyölle.

Teemakokonaisuuteen *Testauksen suunnittelu ja suorittaminen* muodostuivat seuraavat kehityskohteet:

1) Testauksen suunnittelun ja testituloksien raportoinnin kehittäminen
2) Testaukseen liittyvien vaatimuksien huomioiminen tietojärjestelmän vaatimusmäärittelyssä
3) Tietojärjestelmän rajapintojen kuvaaminen

5.3.2 Testiympäristön käyttö testauksessa

Toiseksi teemakokonaisuudeksi haastatteluissa nousi testiympäristön käyttö testauksessa. Aiheen yksityiskohtaisuuden vuoksi kovin suurta havaintomäärää ei tähän teemakokonaisuuteen liittynyt. Kuvasta 17 nähdään tähän aihealueeseen liittyvät päätelmät ja kehityskohteet.



KUVA 17. Testiympäristön käyttö testauksessa

Suurimmalla osalla haastateltavista, seitsemällä yhdestätoista, oli kokemusta testiympäristön käytöstä uusien ohjelmistoversioiden testauksessa. Osa toi selkeästi esille, että uusi päivitys asennetaan aina ensin testiympäristöön ja vasta hyväksymistestauksen jälkeen siirretään tuotantoympäristöön. Haastatteluissa tuli myös päinvastaisia kommentteja, kuten ”uusi versio otetaan suoraan tuotantokäyttöön”, tai ”ei ole sellaista testiympäristöä, jossa olisi rakennettuna kaikki käytössä olevat rajapinnat…”. Yksi haastateltava kommentoi, että ”päivityksiä ei ole testattu viime aikoina testiympäristön kautta, kun ei ole ollut ongelmia”.

Haastatteluaineiston havainnoista voidaan päätellä, että yli puolella pääkäyttäjistä testiympäristö on siis käytössä, ja lopuilla pääkäyttäjillä testiympäristö ei ole joko käytössä tai sellaista ei ole käytettävissä. Ohjelmistoversioita päivitettäessä on ohjelmiston testaus kuitenkin isossa roolissa ennen ohjelmistoversion siirtoa tuotantokäyttöön. Mikäli tietojärjestelmälle ei ole käytettävissä testausympäristöä, on tietojärjestelmän testaus hankittava esimerkiksi tietojärjestelmän toimittajalta tai sitten päätetään jättää testausvaihe te-

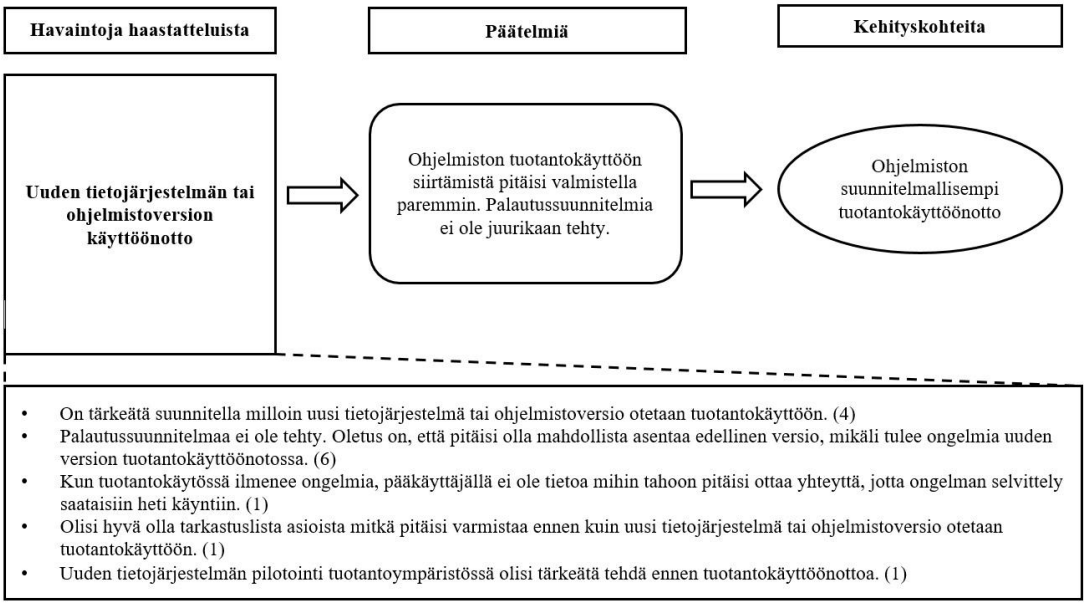
kemättä ennen tuotantoon siirtoa. Tässäkin tapauksessa testausvaihe kuitenkin tulee tapahtumaan, mutta vasta tuotantoympäristössä loppukäyttäjien käyttäessä tietojärjestelmää. Edellä mainituista vaihtoehdoista: testiympäristössä testaus, toimittajan suorittama testaus tai testaus tuotannossa loppukäyttäjien toimesta, on ohjelmiston laadukkuutta ja loppukäyttäjien tyytyväisyyttä ajatellen paras vaihtoehto testiympäristössä testaus. Aina ei ole kuitenkaan kannattavaa rakentaa täysin tuotantoympäristön kaltaista testiympäristöä. Päätös rakentaa tai olla rakentamatta testiympäristö huomioiden kummankin vaihtoehdon hyödyt ja haitat, olisi tärkeätä kuitenkin tehdä jokaisen tietojärjestelmän kohdalla.

Teemakokonaisuuteen *Testiympäristön käyttö testauksessa* muodostettiin seuraava kehityskohde:

1) Testiympäristön hyödyntäminen ohjelmistoversion hyväksymistestauksessa ennen uuden version tuotantokäyttöönottoa

5.3.3 Uuden tietojärjestelmän tai ohjelmistoversion käyttöönotto

Haastatteluissa selvitettiin nykytilannetta ja mahdollisia kehitysideoita uuden tietojärjestelmän tai ohjelmistoversion käyttöönottoon liittyen. Kaksi aihetta, suunnitelmallisempi valmistautuminen tuotantokäyttöön ja palautussuunnitelman puuttuminen, tulivat haastatteluissa esille useamman haastateltavan kanssa. Kuvassa 18 ovat tähän teemakokonaisuuteen liittyvät havainnot, päätelmät ja näistä muodostetut kehityskohteet.



KUVA 18. Uuden tietojärjestelmän tai ohjelmistoversion käyttöönotto

Uuden tietojärjestelmän tai ohjelmistoversion siirtäminen tuotantokäyttöön vaatii hyvää suunnittelua ja valmistautumista. Se, että ohjelmisto läpäisee hyväksymistestit, ei riitä valmisteluksi tuotantoon siirrolle. Uuden tietojärjestelmän tai ohjelmistoversion tuotantokäyttöönotossa pitää tarkkaan suunnitella muun muassa ajankohta milloin tuotantoon siirto tehdään, miten varaudutaan mahdollisiin ongelmiin tuotantoon siirtymisessä, ja miten tiedotetaan ja ohjeistetaan loppukäyttäjiä. Neljässä haastattelussa haastateltavat nostivat esille sen, kuinka tärkeätä on suunnitella juuri ajankohta milloin uusi tietojärjestelmä tai uusi ohjelmistoversio viedään tuotantokäyttöön. Ajankohtaa suunniteltaessa pitää varmistaa, että toimittajan tuki on kyseisenä ajankohtana saatavilla ja selvittää mikä olisi loppukäyttäjien kannalta paras ajankohta muutokselle.

Palautussuunnitelma kuvaa miten toimitaan, kun uuden ohjelmistoversion kanssa on tuotantokäytössä niin vakavia ongelmia, että uutta versiota ei voidakaan pitää tuotantokäytössä. Yksi ratkaisu on varautua palauttamaan edellinen ohjelmistoversio takaisin tuotantokäyttöön, varsinkin jos näyttää, että havaittuja ongelmia ei saada nopeasti korjattua. Haastatteluissa kuuden haastateltavan osalta tuli esille, että tällaista palautussuunnitelmaa ei ole tehty. Oletus on, että edellinen versio pystytään palauttamaan ongelmitta, mikäli siihen tulee tarvetta. Nyt tällaisissa tilanteissa on toimittajalta pyritty saamaan mahdollisimman nopeasti korjaus ongelmatilanteeseen.

Lisäksi haastatteluissa tehtiin yksittäinen havainto liittyen epäselvyyteen miten saada tukea ongelmatilanteissa. Muut haastatteluissa tehdyt havainnot olivat haastateltavien esille tuomia kehitysideoita teemaan liittyen. Muun muassa yksi haastateltava näki tärkeäksi, että olisi tarkastuslista uuden tietojärjestelmän tai ohjelmistoversion käyttöönottoon liittyvistä valmistelutehtävistä. Toinen haastateltava toi esille pilotoinnin tärkeyden. Pilotoinnilla tarkoitetaan uuden tietojärjestelmän tai ohjelmistoversion käyttämistä pienemmän loppukäyttäjäjoukon toimesta ennen varsinaista käyttöönottoa. Pilotoinnin tavoitteena on saada loppukäyttäjiltä palautetta ja mahdollisia kehitystarpeita tietojärjestelmän käyttöön liittyen. Pilotointipalautteen perusteella päätetään onko tietojärjestelmä valmis jo laajempaan tuotantokäyttöön vai onko vielä syytä tehdä muutoksia ennen virallista tuotantokäyttöönottoa.

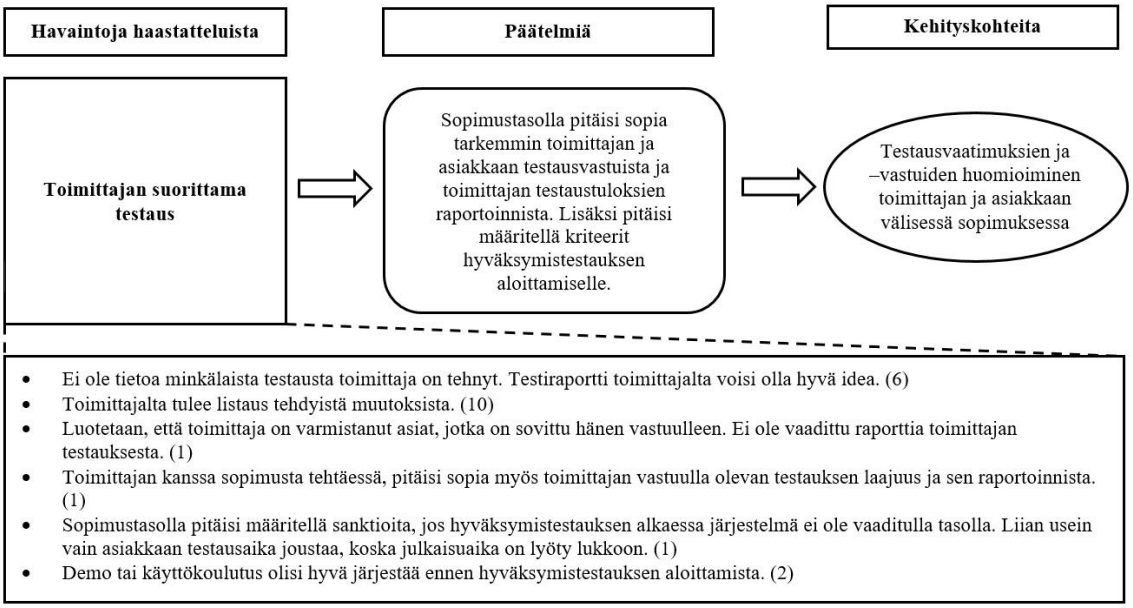
Haastatteluhavainnoista voidaan päätellä, että uuden tietojärjestelmän tai ohjelmistoversio tuotantokäyttöönottoa pitäisi suunnitella paremmin. Suunnitelmassa pitäisi ottaa kantaa siihen miten toimitaan, jos tuotantoon siirtyessä tulee ongelmia, ja varmistaa, että ongelmien selvittelytuki on saatavilla.

Teemakokonaisuuteen *Uuden tietojärjestelmän tai ohjelmistoversion käyttöönotto* muodostettiin seuraava kehityskohde:

1) Ohjelmiston suunnitelmallisempi tuotantokäyttöönotto

5.3.4 Toimittajan suorittama testaus

Toimittajan suorittama testaus oli neljäs teemakokonaisuus haastatteluissa. Toimittajalla on yleisesti vastuu varmistaa, että määritetyt toiminnallisuudet toimivat vaatimusmäärittelyn mukaisesti. Kuvassa 19 on kuvattu tähän teemakokonaisuuteen liittyvät haastatteluhavainnot sekä niistä muodostuneet päätelmät että kehityskohteet.



KUVA 19. Toimittajan suorittama testaus

Yli puolet haastateltavista toivat esille, että heillä ei ole tietoa minkälaista testausta toimittaja tekee tietojärjestelmälle tai ohjelmistoversiolle ennen kuin he saavat sen omaan hyväksymistestaukseensa. Toimittajan tuottama testiraportti omasta testauksestaan nousikin haastateltavilta yhdeksi kehitysideaksi tuleviin tietojärjestelmäprojekteihin. Toimittajan tekemät muutokset ohjelmistoversioon olivat lähes kaikilla tiedossa. Toimittajan

listaus tehdyistä muutoksista oli haastateltavien mielestä hyvä ja toimiva tapa informoida mitä muutoksia ohjelmistoversio sisälsi. Tätä listausta he pystyivät hyödyntämään omassa hyväksymistestauksessaan testatessaan ohjelmiston sisältämiä muutoksia.

Muut haastatteluhavainnot olivat yksittäisiä haastateltavien esille tuomia ongelmia ja kehitysideoita. Yhden haastateltavan mielestä luotetaan liikaa siihen, että toimittaja varmistaa ja toteuttaa asiat, jotka on sovittu toimittajan vastuulle. Toimittajalta ei ole kuitenkaan vaadittu mitään dokumentaatiota toimittajan suorittamista tehtävistä. On käynyt myös niin, että toimittaja ei ole tehnyt vastuullaan olevia asioita, mutta myöskään asiakas ei ole omassa hyväksymistestauksessaan varmistanut niiden toimivuutta. Tällöin puutteet ja ongelmat on havaittu vasta ohjelmiston tuotantokäyttöönoton jälkeen. Haastatteluissa nousi esille kehitysideoita testausvastuiden tarkempaan sopimiseen toimittajan ja asiakkaan välillä sekä aloituskriteereiden määrittäminen hyväksymistestauksen aloittamiselle. Myös sanktioiden määrittäminen sopimuksessa nähtiin yhdeksi vaihtoehdoksi, mikäli ohjelmisto ei ole vaaditulla tasolla hyväksymistestaukseen luovutettaessa. Esille tuli myös kehitysidea tietojärjestelmän demon tai käyttökoulutuksen järjestämisestä asiakkaalle ennen kuin asiakas aloittaa hyväksymistestauksen. Demolla tarkoitetaan tietojärjestelmän toimintojen esittelyä asiakkaalle, jolloin asiakas näkee käytännössä kuinka tietojärjestelmää käytetään loppukäyttäjän näkökulmasta. Yleensä toimittajan edustaja suorittaa demon ja asiakkaan edustajat vain seuraavat demoa. Myös käyttökoulutus antaa vastaavaa tietoa asiakkaalle. Tosin asiakas on silloin itse käyttämässä tietojärjestelmää loppukäyttäjän näkökulmasta, jolloin asiakkaalla on paremmat mahdollisuudet perehtyä tietojärjestelmän toimivuuteen juuri niistä näkökulmista kuin asiakas haluaa.

Haastatteluhavainnoista toimittajan suorittamaan testaukseen liittyen voidaan päätellä, että sopimustasolla pitäisi ottaa paremmin huomioon toimittajan vastuulla oleva testaus sekä sopia mitä dokumentaatiota, kuten mahdollisesti testiraportteja, vaaditaan toimittajan suorittamasta testauksesta. Erityisesti asiakkaan hyväksymistestaukseen liittyen olisi tärkeätä määritellä sopimuksessa kriteerit hyväksymistestauksen aloittamiselle ja mahdolliset sanktiot mikäli tietojärjestelmä ei täytä kriteereitä. Haastattelujen mukaan liian usein ollaan tilanteessa, että asiakkaan pitää joustaa omassa hyväksymistestausajassaan, koska julkaisuaikataulu on lyöty lukkoon, vaikka tietojärjestelmä ei ole vielä valmis asiakkaan hyväksymistestaukseen.

Teemakokonaisuuteen *Toimittajan suorittama testaus* muodostettiin seuraava kehityskohde:

1) Testausvaatimuksien ja -vastuiden huomioiminen toimittajan ja asiakkaan välisessä sopimuksissa

5.4 Ratkaisuehdotuksia kehityskohteisiin

Seuraavaksi esitetään ratkaisuehdotuksia kuhunkin edellä kuvattuun kehityskohteeseen. Ratkaisuehdotukset ovat syntyneet teoriaan perehtymisen sekä haastatteluissa esille tulleiden kehittämistarpeiden että kehittämisideoiden pohjalta. Ratkaisuehdotuksien muodostamisessa on tavoitteena ollut mahdollisimman konkreettisten ratkaisujen luominen. Yhteensä kuuteen kehityskohteeseen on muodostettu kahdeksan ratkaisuehdotusta. Ratkaisuehdotuksien toteuttamisessa pitää huomioida Laurean erityyppisten tietojärjestelmähankkeiden (kuva 1) tarpeet. Seuraavissa alaluvuissa käsitellään tarkemmin teemakokonaisuuksittain ratkaisuehdotuksia.

5.4.1 Testauksen suunnittelu ja suorittaminen

Verrattaessa muihin teemakokonaisuuksiin, tutkimuksen tuloksena muodostui eniten kehityskohteita testauksen suunnitteluun ja suorittamiseen. Taulukkoon 2 on listattu tähän teemakokonaisuuteen liittyvät kehityskohteet ja niille muodostetut ratkaisuehdotukset.

Ensimmäiseen kehityskohteeseen *Testauksen suunnittelun ja testituloksien raportoinnin kehittäminen* muodostuivat ratkaisuehdotukset R1, R2 ja R3. Haastatteluissa tuli vahvasti esille, että testauksen peruskäytännöt, testauksen suunnittelusta testauksen suorittamiseen ja raportointiin, eivät olleet useimmille haastateltaville tuttuja toimintatapoja. Osa haastateltavista toikin haastattelussa esille, että olisi hyvä tietää miten testausta käytännössä tehdään ja mihin testauksessa kannattaisi kiinnittää huomiota. Ratkaisuehdotuksessa R1 onkin tavoitteena kasvattaa tietojärjestelmien pääkäyttäjien sekä projektipäälliköiden tietotaitoa testauksen perusteista, kuten miksi testausta tehdään, mitkä ovat testauksen tavoitteet ja minkälaisia käytäntöjä laadukkaaseen testaukseen kuuluu. Ratkaisuehdotuksissa R2 ja R3 on tarkoituksena luoda konkreettisia apuvälineitä; mallipohjia ja ohjeis-

tuksia testauksen suunnitteluun ja testituloksien raportointiin sekä testitapauksen suunnitteluun. Mallipohjien ja ohjeistuksien luomisessa tulee ottaa huomioon Laurean erityyppisten tietojärjestelmähankkeiden (kuva 1) tarpeet. Lähtökohtana ohjeistuksen muodostamiseen ovat luvussa 3.2 kuvatut testauksen suunnittelussa huomioitavat asiat.

Toiseen kehityskohteeseen *Testaukseen liittyvien vaatimuksien huomioiminen tietojärjestelmän vaatimusmäärittelyssä* muodostui ratkaisuehdotus R4. Haastattelujen perusteella erityisesti ei-toiminnalliset vaatimukset jäävät usein lähes huomiotta tietojärjestelmän vaatimusmäärittelyvaiheessa. Ratkaisuehdotuksessa R4 on tarkoituksena muodostaa tietopaketti siitä mitä tarkoittavat toiminnalliset ja ei-toiminnalliset vaatimukset.

Kolmanteen kehityskohteeseen *Tietojärjestelmän rajapintojen kuvaaminen* syntyi ratkaisuehdotus R5, jonka sisältää mallipohjan ja ohjeistuksen luomisen tietojärjestelmän rajapintojen ja niihin liittyvien tietovirtojen kuvaamiseksi. Näiden tietojen dokumentointi auttaa pääkäyttäjää hallitsemaan laadukkaammin tietojärjestelmänsä muutoksia.

TAULUKKO 2. Ratkaisuehdotukset testauksen suunnitteluun ja suorittamiseen

Teemakokonaisuus	Kehityskohde	Ratkaisuehdotus (R)
Testauksen suunnittelu ja suorittaminen	Testauksen suunnittelun ja testituloksien raportoinnin kehittäminen	R1: Tietopaketti testauksen tavoitteista, peruskäytännöistä ja hyvistä toimintatavoista R2: Mallipohja ja ohjeistusta testauksen suunnitteluun ja testituloksien raportointiin R3: Mallipohja ja ohjeistusta testitapauksien suunnitteluun (perustuen mm. käyttötapauksiin)
	Testaukseen liittyvien vaatimuksien huomioiminen tietojärjestelmän vaatimusmäärittelyssä	R4: Tietopaketti toiminnallisista ja ei-toiminnallisista vaatimuksista
	Tietojärjestelmän rajapintojen kuvaaminen	R5: Mallipohja ja ohjeistusta tietojärjestelmän rajapintojen ja tietovirtojen kuvaamiseksi

Esitettyjä ratkaisuehdotuksia tukevat myös testauksen taustateoriat. Perry (2006, 39) korostaa yleisesti testausosaamisen merkitystä tavoitellessa laadukasta testausta. Sekä Perry (2006, 209) että Tuovinen (2013b, 6, 9) näkevät erityisesti testauksen suunnittelun yhtenä

tärkeimpänä edellytyksenä laadukkaalle testaukselle. Huolellisesti suunnitellut testitapaukset ovat myös Oppijan verkkopalvelujen (2013) mukaan edellytys onnistuneelle testaukselle. Haikala & Mikkonen (2011, 64, 79-83) esittävät käyttötapauksien tai käyttäjätarinoiden hyödyntämistä testitapauksien suunnittelussa. Testituloksien nopeaan ja laadukkaaseen raportointiin kannustavat Pöyhönen & Pyhäjärvi (2006, 112). Ratkaisuehdotuksien R1, R2 ja R3 muodostamista ovat tukeneet edellä mainittujen tekijöiden esittämät kuvaukset hyvistä testauksen käytännöistä.

Paakin (2011, 3, 6; 2014, 20-21) korostama vaatimusmäärittelyn tärkeys onnistuneessa ohjelmistoprojektissa, ja erityisesti ei-toiminnallisten vaatimuksien huomioiminen, toimivat pohjana ratkaisuehdotuksena R4 muodostamiselle. Ratkaisuehdotuksen R5 tärkeyttä puoltaa Koskimiehen & Mikkosen (2005, 58) näkemys siitä, että rajapintojen huolellinen suunnittelu ja kuvaus ovat edellytys tietojärjestelmän ylläpidettävyydelle ja testattavuudelle.

5.4.2 Testiympäristön käyttö testauksessa

Testiympäristön käyttö testauksessa –teemakokonaisuuteen tuli haastatteluissa varsin vähän erilaisia havaintoja. Niinpä tähän teemakokonaisuuteen muodostui vain yksi kehityskohde ja samoin yksi ratkaisuehdotus, jotka on kuvattu taulukossa 3.

TAULUKKO 3. Ratkaisuehdotus testiympäristön käyttöön testauksessa

Teemakokonaisuus	Kehityskohde	Ratkaisuehdotus (R)
Testiympäristön käyttö testauksessa	Testiympäristön hyödyntäminen ohjelmiston hyväksymistestauksessa ennen uuden version tuotantokäyttöönottoa	R6: Tietopaketti testiympäristön merkityksestä ja hyödyllisyydestä hyväksymistestauksessa

Ratkaisuehdotuksessa R6 on tarkoitus muodostaa pääkäyttäjille tietopaketti siitä, mitä tarkoitetaan testiympäristöllä, mikä on testiympäristön merkitys hyväksymistestauksessa ja mitä kannattaa ottaa huomioon, kun tehdään päätöstä rakennetaanko testiympäristö vai ei.

Ratkaisuehdotus R6 pohjautuu Perryn (2006, 38) esittämään ohjelmistotestauksen tavoitteeseen löytää ohjelmiston käyttöä estävät tai haittaavat virheet ennen kuin ohjelmisto

otetaan tuotantokäyttöön. Tämä on mahdollista, mikäli on olemassa ympäristö, jossa uuden ohjelmistoversion testaus voidaan suorittaa ennen ohjelmistoversion tuotantokäyttöönottoa.

5.4.3 Uuden tietojärjestelmän tai ohjelmistoversion käyttöönotto

Uuden tietojärjestelmän tai ohjelmistoversion käyttöönottoon liittyen nousi haastatteluissa yksi selkeä kehityskohde: Tuotantokäyttöönottoa pitäisi suunnitella ja valmistella nykyistä paremmin. Taulukossa 4 on esitetty ratkaisuehdotus tähän kehityskohteeseen.

TAULUKKO 4. Ratkaisuehdotus uuden tietojärjestelmän tai ohjelmistoversion käyttöönottoon

Teemakokonaisuus	Kehityskohde	Ratkaisuehdotus (R)
Uuden tietojärjestelmän tai ohjelmistoversion käyttöönotto	Ohjelmiston suunnitelmallisempi tuotantokäyttöönotto	R7: Tarkastuslista ohjelmiston tuotantokäyttöönoton tehtävistä

Ratkaisuehdotuksen R7 tavoitteena on luoda tietojärjestelmien pääkäyttäjille tarkastuslista yleisimmistä tehtävistä, jotka olisi hyvä huomioida valmistellessa uuden tietojärjestelmän tai ohjelmistoversion siirtoa tuotantokäyttöön. Näin tuotantoon siirtymisestä tulisi suunnitelmallisempaa ja todennäköisimpiin ongelmatilanteisiin olisi etukäteen varauduttu.

Hyötyläisen & Kalliokosken (2001, 20) mukaan tietojärjestelmän käyttöönotto on kokonaisuudessaan monivaiheinen ja monimutkainen prosessi, joka usein epäonnistuu. Hyvällä käyttöönottosuunnitelmalla on iso rooli käyttöönoton onnistumisessa. Ratkaisuehdotuksella R7 pyritäänkin parantamaan ohjelmistojen käyttöönottovalmisteluja huomioimalla muun muassa Hyötyläisen ja Kalliokosken (2001) sekä Aikkilan ja Saukon (2012) listaamia käyttöönottoa valmistelevia tehtäviä.

5.4.4 Toimittajan suorittama testaus

Toimittajan suorittamaan testaukseen liittyen päähavainto haastatteluista oli haastateltavien tietämättömyys toimittajan suorittamasta testauksesta. Toimittajan luovuttaessa uuden tietojärjestelmän tai ohjelmistoversion asiakkaan hyväksymistestaukseen luotetaan, että toimittaja on hoitanut vastuullaan olevan testauksen ilman, että tiedetään tarkasti mitä testausta toimittaja on tehnyt. Taulukossa 5 on kuvattu tähän teemakokonaisuuteen liittyvä kehityskohde ja ratkaisuehdotus.

TAULUKKO 5. Ratkaisuehdotus toimittajan suorittamaan testaukseen

Teemakokonaisuus	Kehityskohde	Ratkaisuehdotus (R)
Toimittajan suorittama testaus	Testausvaatimuksien ja –vastuiden huomioiminen toimittajan ja asiakkaan välisessä sopimuksessa	R8: Tarkastuslista testaukseen liittyvistä asioita, jotka tulisi huomioida sopimusvaiheessa

Ratkaisuehdotuksessa R8 ehdotetaan, että muodostetaan tarkastuslista asioista, jotka olisi hyvä huomioida toimittajan ja asiakkaan välisessä sopimuksessa. Tietojärjestelmään tai ohjelmistoversioon liittyvät testausvaatimukset olisi tärkeätä dokumentoida sopimuksessa sekä sopia testausvastuut että raportointikäytännöt. Haastattelujen perusteella juuri näkyvyys toimittajan suorittamaan testaukseen puuttui, joten testaustuloksien raportoinnista olisi hyvä sopia jatkossa toimittajan kanssa. Tarkastuslistalle olisi hyvä ottaa mukaan aloituskriteerien määrittäminen asiakkaan hyväksymistestauksen aloittamiselle. Myös mahdollisesta tietojärjestelmän tai ohjelmistoversion demosta tai käyttökoulutuksesta sopiminen olisi hyvä listata tarkastuslistalla, jotta asia huomioidaan jo sopimusvaiheessa.

Naik & Tripathy (2008, 455-456; 463-464) listaavat asioita, jotka olisi syytä huomioida hyväksymistestauksessa, sekä asiakkaan ja toimittajan välisessä sopimuksessa. Naikin & Tripathyn listaamiin huomioitaviin asioihin pohjautuu myös ratkaisuehdotus R8.

6 TUTKIMUKSEN LAATU JA LUOTETTAVUUS

6.1 Tutkimusmenetelmien arviointi

Opinnäytetyön tavoitteena oli testauskäytäntöjen kehittäminen Laurean tietojärjestelmäprojekteissa. Tutkimusosuuden tarkoituksena oli selvittää millaisia testauskäytäntöjä Laurean tietojärjestelmien pääkäyttäjillä ja projektipäälliköillä on tällä hetkellä sekä minkälaisia mahdollisia kehityskohteita he näkivät testauskäytännöissä. Tutkimuksessa kartoitettiin myös mitkä ovat tietojärjestelmän toimittajan vastuut tietojärjestelmän testauksessa sekä miten Laurean pääkäyttäjät valmistautuvat tietojärjestelmän tuotantokäyttöönottoon.

Valittu tutkimusmenetelmä, teemahaastattelu, sopi hyvin keskusteluun pohjautuvaan haastattelutilanteeseen, koska tavoitteena oli selvittää ennalta määriteltyjen teema-alueiden tämän hetken käytännöt ja niihin liittyvät mahdolliset kehityskohteet. Myös aineiston analyysitapana käytettiin teemoittelua. Haastatteluista saatua aineistoa ryhmiteltiin ja lopulta aineisto jaettiin eri teemakokonaisuuksiin. Tällainen analysointitapa helpotti kuvaamaan tutkimusongelmien kannalta oleellista tietoa sekä muodostamaan tuloksia tutkimusosuuden tavoitteisiin nähden.

6.2 Tutkimuksen luotettavuus

Tutkimuksen tarkoituksena on tuottaa mahdollisimman luotettavaa tietoa. Luotettavuutta arvioidaan usein validiteetin – tutkimuksessa on tutkittu sitä, mitä on luvattu – ja reliabiliteetin – tutkimustulosten toistettavuus – käsitteillä. Kuitenkin laadullisen tutkimuksen luotettavuutta arvioitaessa, näiden käsitteiden käyttöä on usein kritisoitu. Yhtenä syynä kritiikkiin on ollut se, että nämä käsitteet ovat lähtöisin määrällisestä tutkimuksesta ja ne vastaavat lähinnä määrällisen tutkimuksen tarpeita. Yleisin kritiikki onkin, että ne perustuvat oletukseen yhdestä konkreettisesta todellisuudesta, jota tutkimuksessa tavoitellaan. (Tuomi & Sarajärvi 2009, 136.) Hirsjärvi, Remes & Sajavaara (2013, 232) tuovat esille, että laadullisessa tutkimuksessa ”tapaustutkimuksen tekijä voi aiheellisesti ajatella, että kaikki ihmistä ja kulttuuria koskevat kuvaukset ovat ainutlaatuisia, ettei ole kahta saman-

laista tapausta, joten perinteiset luotettavuuden ja pätevyyden arvioinnit eivät tule kysymykseen.” Tutkimuksen luotettavuudesta kertoo myös tutkimustuloksien samankaltaisuus eri mittauskerroilla ja tilanteessa, kun tutkimuksen suorittaa kaksi eri tutkijaa. (Hirsjärvi, Remes & Sajavaara 2013, 231.) Hirsjärven ja Hurmeen (2004, 186) mukaan laadullisessa tutkimuksessa näihin edellä mainittuihin tavoitteisiin ei päästä. On oletettavaa, että ihmiselle on ominaista ajassa tapahtuva muutos, joten samaa henkilöä tutkittaessa ei saada aivan identtistä tutkimustulosta kahdella eri tutkimuskerralla. Lisäksi jokainen yksilö tekee omien kokemuksiensa perusteella tutkittavasta kohteesta oman tulkintansa, joten on epätodennäköistä, että kaksi tutkijaa ymmärtäisi tutkittavan sanoman täysin samalla tavalla. (Hirsjärvi & Hurme 2004, 186.) Vaikka näitä käsitteitä ei suoraan käytettäisikään, kuitenkin myös laadullisen tutkimuksen luotettavuutta ja pätevyyttä tulisi arvioida. Mikä sitten kertoo laadullisen tutkimuksen luotettavuudesta?

Hirsjärvi, Remes & Sajavaara (2013, 232) nostavat esille seuraavia asioita, jotka heidän mielestään nostavat laadullisen tutkimuksen luotettavuutta ja pätevyyttä:

- tutkijan tarkka selostus tutkimuksen toteuttamisesta
- haastattelututkimuksessa aineiston keräykseen liittyvien olosuhteiden ja paikkojen kuvaus sekä haastatteluun käytetty aika, mahdolliset häiriötekijät, virhetulkinnat ja tutkijan oma itsearviointi tilanteesta
- analyysissa käytetty luokittelu ja sen perusteet
- tulosten tulkinnassa tuotu esille perusteet tulkinnoille.

Eskola & Suoranta (2000, 210) korostavat myös laadullisen tutkimuksen luotettavuuden arvioinnissa koko tutkimusprosessia, koska tutkimuksen lähtökohtana on tutkijan avoin subjektiviteetti ja myöntäminen, että tutkija on tutkimuksensa keskeinen tutkimusväline. Tämän opinnäytetyön tutkimusosuuden luotettavuutta arvioitaessa on siis hyvä käydä läpi miten tutkimus toteutettiin.

Tutkimus suoritettiin laadullisena tutkimuksena käyttäen teemahaastattelua, jossa haastatteluun valituille pääkäyttäjille ja projektipäälliköille muodostettiin omat haastattelulomakkeet. Haastattelulomakkeen kysymyksien testaamisella ennen haastatteluja pyrittiin vahvistamaan tutkimuksen aineiston laadukkuutta. Haastattelulomake muodostui avoimista kysymyksistä ja lomake lähetettiin haastateltaville ennen haastattelua. Haastateltavia ei kuitenkaan pyydetty vastaamaan kysymyksiin etukäteen vaan he saivat halutessaan tutustua etukäteen kysymyksiin ja haastattelun teema-aiheisiin.

Haastateltavaksi valittiin pääkäyttäjiä, joiden tietojärjestelmää käytetään laajasti Laureassa. Haastateltavina olivat myös tietohallinnon projektipäälliköt tuoden kokemuksia ja näkemyksiä projektinhallinnan näkökulmasta. Lähes kaikkien haastateltavien osalta oli tilanne, että haastattelija ja haastateltavat tunsivat toisensa etukäteen ja olivat toimineet yhdessä samoissa projekteissa. Ainoastaan yhden haastateltavan kohdalla haastattelutapaaminen oli ensimmäinen kasvokkain tapahtuva tilaisuus. Kaikki haastateltavat olivat kokeneita oman vastuualueensa asiantuntijoita.

Haastattelut toteutettiin yksilöhaastatteluina. Haastatteluissa oli rento ja avoin ilmapiiri. Tähän varmasti vaikutti myös haastattelutilanteen huolellinen valmistelu, kuten haastattelukysymyksien lähettäminen etukäteen haastateltaville sekä joustavuus haastatteluajankohdan sopimisessa haastateltavan muihin tehtäviin nähden. Haastattelun aluksi kerrottiin tutkimustyön tavoite ja, että haastattelu tullaan nauhoittamaan. Haastatteluaineiston varmistamiseksi käytettiin tupla-tallennusta ja varauduttiin varapatterein nauhoitusvälineistön toimivuuteen. Nauhoitteet siirrettiin kahteen eri tallennusmediaan pian haastattelun jälkeen, jotta voitiin varmistaa aineiston pysyvyys. Haastattelujen aikana pyrittiin välttämään haastateltavien johdattelua. Kukin haastattelu saatiin suoritettua yhden tunnin aikana.

Ensimmäisen kolmen haastattelun jälkeen huomattiin tietojärjestelmän toimittajiin liittyvän teema-alueen liian kapea käsittely haastattelukysymyksissä. Kolmen haastattelun jälkeen haastattelulomakkeita päivitettiin tältä osin ja jo haastatelluille lähetettiin lisäkysymyksiä sähköpostitse toimittaja teema-alueeseen liittyen. Haastatteluvaiheen jälkeen haastattelut purettiin nopeasti ryhmitellen eri aihe-alueisiin.

Tutkimuksen objektiivisuus on tärkeää tutkimuksen lopputuloksen kannalta. Tutkimuksen objektiivisuudella tarkoitetaan sitä, että tutkija ei sekoita omia uskomuksiaan, asenteitaan ja arvostuksiaan tutkimuskohteeseen (Eskola & Suoranta 2000, 17). Täydellinen objektiivisuus ei kuitenkaan ole mahdollista. Tähdättäessä riittävään objektiivisuuteen, tutkijan tulee pyrkiä aktiivisesti tunnistamaan omat esioletuksensa ja asenteensa, ja koittaa toimia niin, että ne eivät vaikuta tutkimukseen liiaksi. (Eskola & Suoranta 2000, 17.) Tutkimusaineistoa analysoitaessa päätelmät ja kehityskohteet muodostettiin haastatteluaineiston pohjalta. Testauksen taustateorioiden, haastatteluissa esille tulleiden kehitysideoiden ja oman testauskokemuksen pohjalta muodostettiin ratkaisuehdotuksia kehityskohteille.

Tämän tutkimuksen luotettavuus koostuu kaikista aiemmin kuvatuista alueista ja siitä, että tutkimuksella on tutkittu sitä mitä pitikin. Tutkimuksen luotettavuutta vahvistaa myös se, että tutkimukseen on valittu suhteellisen laaja haastateltavien joukko ja, että haastateltavien valinta on tehty tietyin perustelluin kriteerein. Erilaisella haastateltavien otoksella olisi voinut olla vaikutusta lopputuloksiin, mutta tältä pohjalta tehty haastateltavien valinta edustaa tärkeimpien ja käytetyimpien tietojärjestelmien pääkäyttäjien näkemystä.

6.3 Tutkimuksen eettisyys

Tässä opinnäytetyössä on pyritty huomioimaan hyvä tieteellinen käytäntö eri lähteitä käyttäen. Lähteiksi on valittu mahdollisimman uutta ja ajankohtaista lähdemateriaalia. Käytettävien lähteiden luotettavuutta ja oikeellisuutta, erityisesti verkkomateriaalien osalta, on arvioitu kriittisesti. Haastatteluun valittujen henkilöiden anonymiteetti varmistettiin aineistoa käsiteltäessä ja nauhoitteet tuhottiin litteroinnin jälkeen. Haastattelut toteutettiin rauhallisessa erillisessä tilassa. Haastattelujen nauhoittamisesta ilmoitettiin aina haastattelun alussa. Lisäksi haastatteluun pyydettäessä henkilöille painotettiin vapaaehtoista osallistumista. Tutkittavan aiheen ja haastatteluun osallistuvien osalta pyydettiin työnantajalta tutkimuslupa suorittaa tähän opinnäytetyöhön suunniteltu tutkimusosuus.

7 JOHTOPÄÄTÖKSET JA POHDINTA

Opinnäytetyön tavoitteena oli Laurean tietojärjestelmäprojektien nykyisten testauskäytäntöjen kartoittaminen ja kehittämiskohteiden sekä ratkaisuehdotuksien muodostaminen. Tavoitteena oli kasvattaa projektipäälliköiden ja tietojärjestelmien pääkäyttäjien tietotaitoa tietojärjestelmien testauksen suhteen sekä tuottaa heille mahdollisimman konkreettisia työvälineitä testauksen kehittämiseksi.

Lähdeaineistoa taustateorioiden tutkimiseen löytyi laajasti niin ulkomaisten kuin kotimaistenkin testauksen asiantuntijoiden piiristä. Suurimmat haasteet opinnäytetyön edistämisessä liittyivät työn aikatauluttamiseen ja laajuuden arvioimiseen. Kuten testauksessa myös tässäkin tutkimus- ja kirjoittamistehtävässä auttoi suunnitelmallinen lähestymistapa ja suoritettavien tehtävien dokumentointi ja aikataulutus.

Haastattelujen perusteella voidaan todeta, että testauskulttuuri ei ole kovin vahva Laurean tietojärjestelmäprojekteissa. Testausta yleisesti ottaen kyllä suoritetaan, mutta se ei ole suunnitelmallista eikä testauksen tuloksia dokumentoida. Päätökset tietojärjestelmän tuotantokäyttöönotosta perustuvat enemmän tuntemuksiin kuin dokumentoituun testausdataan. Näihin havaintoihin liittyen pyrittiin löytämään ratkaisuehdotuksia ja mielestäni tavoitteessa myös onnistuttiin.

Opinnäytetyön tavoite purettiin seuraaviin tutkimuskysymyksiin:

1. Mitä konkreettisia työvälineitä voitaisiin ottaa käyttöön Laurean tietojärjestelmäprojekteissa testauksen kehittämiseksi?

2. Miten kehittää Laurean tietojärjestelmäprojektien vetovastuullisten tietotaitoa testaukseen liittyen?

Mielestäni opinnäytetyössä onnistuttiin hyvin vastaamaan tutkimuskysymyksien asettamiin tarpeisiin. Taulukossa 6 on kuvattu opinnäytetyön tutkimuskysymykset ja niihin tutkimuksen myötä syntyneet ratkaisuehdotukset. Tutkimuksen lopputuloksena syntyi siis yhteensä 8 ratkaisuehdotusta kahteen tutkimuskysymykseen liittyen. Opinnäytetyön tär-

kein tavoite konkreettisten työvälineiden tuottaminen Laurean tietojärjestelmäprojekteille testauksen kehittämiseksi, onnistuttiin saavuttamaan muodostettujen ratkaisuehdotuksien kautta.

TAULUKKO 6. Tutkimuskysymykset ja niihin liittyvät ratkaisuehdotukset

Tutkimuskysymys	Ratkaisuehdotus (R)
Mitä konkreettisia työvälineitä voitaisiin ottaa käyttöön Laurean tietojärjestelmäprojekteissa testauksen kehittämiseksi?	R2: Mallipohja ja ohjeistusta testauksen suunnitteluun ja testituloksien raportointiin R3: Mallipohja ja ohjeistusta testitapauksien suunnitteluun (perustuen mm. käyttötapauksiin) R5: Mallipohja ja ohjeistusta tietojärjestelmän rajapintojen ja tietovirtojen kuvaamiseksi R7: Tarkastuslista ohjelmiston tuotantokäyttöönoton tehtävistä R8: Tarkastuslista testaukseen liittyvistä asioita, jotka tulisi huomioida sopimusvaiheessa
Miten kehittää Laurean tietojärjestelmäprojektien vetovastuullisten tietotaitoa testaukseen liittyen?	R1: Tietopaketti testauksen tavoitteista, peruskäytännöistä ja hyvistä toimintatavoista R4: Tietopaketti toiminnallisista ja ei-toiminnallisista vaatimuksista R6: Tietopaketti testiympäristön merkityksestä ja hyödyllisyydestä hyväksymistestauksessa

Ratkaisuvaihtoehtojen käyttöönotto ja jatkokehitys erilaisten tietojärjestelmäprojektien tarpeisiin jatkuu omana projektina tämän opinnäytetyön valmistuttua. Ratkaisuvaihtoehtojen toteuttamisessa on tärkeätä huomioida Laurean erilaiset tietojärjestelmähankkeet (luku 2.2). Esimerkiksi *Tarkastuslista ohjelmiston tuotantokäyttöönoton tehtävistä (R7)* on erilainen eri hanketyypeille. Uuden räätälöidyn tietojärjestelmän tuotantokäyttöönotto vaatii erilaisia valmisteluja kuin valmisohjelmiston päivitys tuotantoon. Erilaiset hanketyypit lisäävätkin työn laajuutta yllä listattujen ratkaisuehdotuksien toteuttamisessa. Tämä kannattaa huomioida toteutusprojektin aikataulua ja työmäärää suunniteltaessa.

LÄHTEET

Aikkila, P. & Saukko, T. 2012. Tietojärjestelmän käyttöönotto ja ylläpito. Lappeenrannan teknillinen yliopisto. Teknillistaloudellinen tiedekunta. Tuotantotalouden osasto. Kandidaatintyö.

Eriksson, H. 2012. Testausprosessin kehittäminen osana laatutyöskentelyä: case Dimenteq Oy. Tietojenkäsittelyn koulutusohjelma. Turun ammattikorkeakoulu. Opinnäytetyö.

Eskola, J. & Suoranta, J. 2000. Johdatus laadulliseen tutkimukseen. Tampere: Vastapaino.

Haikala, I. & Mikkonen, T. 2011. 12. painos. Ohjelmistotuotannon käytännöt. Helsinki: Talentum.

Hirsjärvi, S. & Hurme, H. 2004. Tutkimushaastattelu. Teemahaastattelun teoria ja käytäntö. Helsinki: Yliopistopaino.

Hirsjärvi, S., Remes, P. & Sajavaara, P. 2013. 15.-17. painos. Tutki ja kirjoita. Helsinki: Tammi.

Hyötyläinen, R. & Kalliokoski, P. 2001. Tietojärjestelmien käyttöönottoprosessi. Teoksessa Kettunen, J. & Simons, M. (toim.) Toiminnanohjausjärjestelmän käyttöönotto pk-yrityksessä: Teknologialähtöisestä ajattelusta kohti tiedon ja osaamisen hallintaa. 2001. Espoo: Valtion teknillinen tutkimuskeskus. VTT-julkaisuja, 17-39.

ISTQB Exam Certification. 2016a. What is Functional testing (Testing of functions) in software? Luettu 17.4.2016. http://istqbexamcertification.com/what-is-functional-testing-testing-of-functions-in-software/

ISTQB Exam Certification. 2016b. What is Non- functional testing (Testing of software product characteristics)? Luettu 17.4.2016. http://istqbexamcertification.com/what-is-non-functional-testing-testing-of-software-product-characteristics/

ISTQB 2016. Glossary. Luettu 20.4.2016. http://astqb.org/glossary/

JHS-suositukset. 2012. JHS 173 ICT-palvelujen kehittäminen: Vaatimusmäärittely. Päivitetty 5.10.2012. Luettu 1.5.2016. http://docs.jhs-suositukset.fi/jhs-suositukset/JHS173/JHS173.pdf

Jyväskylän yliopisto. 2015. Laadullinen tutkimus. Päivitetty 23.4.2015. Luettu 18.2.2016. https://koppa.jyu.fi/avoimet/hum/menetelmapolkuja/menetelmapolku/tutkimusstrategiat/laadullinen-tutkimus

Kallioinen, O. (toim.) 2008. Oppiminen Learning by Developing-toimintamallissa. Laurea Publications A61. Helsinki: Edita Prima Oy.

Kautto, T. 1996. Ohjelmistotestaus ja siinä käytettävät työkalut. Luettu 1.5.2016. http://www.mit.jyu.fi/opiskelu/seminaarit/ohjelmistotekniikka/testaus/#RTFToC19

Koskimies, K. & Mikkonen, T. 2005. Ohjelmistoarkkitehtuurit. Helsinki: Talentum.

Laine, H. & Paakki, J. Ohjelmistotuotanto. Ohjelmistojen testaus 1. Luettu 16.4.2016. https://www.cs.helsinki.fi/u/paakki/ohtuk03-luento12-bw.pdf

Laurea. 2016. Laurea työnantajana. Luettu 5.5.2016. https://www.laurea.fi/

Mayers, G.J., Badgett, T. & Sandler, C. 2011. 3. painos. The Art of Software Testing. Hoboken, New Jersey: John Wiley & Sons, Inc.

Naik, K. & Tripathy, P. 2008. Software Testing and Quality Assurance: Theory and Practice. Hoboken, New Jersey: John Wiley & Sons, Inc.

Oppijan verkkopalvelut. 2013. Testitapausten teko-ohjeet. Päivitetty 11.2.2013. Luettu 2.5.2016. https://confluence.csc.fi/display/oppija/Testitapausten+teko-ohjeet

Paakki, J. 2011. Ohjelmistojen vaatimusmäärittely. Helsingin yliopisto. Luettu 1.5.2016. https://www.cs.helsinki.fi/u/paakki/Vaatimus-11-Luentokalvot-1.pdf

Paakki, J. 2014. Ohjelmistoprosessit ja ohjelmistojen laatu. Helsingin yliopisto. Luettu 1.5.2016. https://www.cs.helsinki.fi/u/paakki/Laatu-14-Luentokalvot-2.pdf

Patton, R. 2006. 2. painos. Software Testing. USA: Sams Publishing.

Perry, W. J. 2006. 3. painos. Effective Methods for Software Testing. USA: Wiley Publishing, Inc.

Pezzè, M. & Young, M. 2008. Software Testing and Analysis: Process, Principles, and Techniques. Hoboken, New Jersey: John Wiley & Sons, Inc.

Pyhäjärvi, M. & Pöyhönen, E. 2006. Ohjelmistojen testaus. Luettu 25.4.2016. http://users.jyu.fi/~kolli/testaus2006/materiaali/Maaret_27102006.pdf

Pöyhönen, E. Lead Test Manager Tieto. 2015. Hyväksymistestaus v5. Kurssi. Hyväksymistestaus 12.-13.11.2015. Talentum Events. Helsinki.

Saarinen, J. 2008. Testaus osana ohjelmistoprojektia. Luettu 1.5.2016. http://theseus.fi/bitstream/handle/10024/10224/Saarinen.Jaakko.pdf?sequence=2

Tamres, L. 2002. Introducing Software Testing. Great Britain: Pearson Education Limited.

Tersa, T. 2002. Testausmenetelmien käyttö sovelluksen systeemitestausvaiheessa. Tietotekniikan laitos. Jyväskylän yliopisto. Luettu 1.5.2016. http://www.mit.jyu.fi/opetus/opinnayte/gradu/systeemitestaus/Tiina_Tersa.pdf

Tilastokeskus. Haastattelutavat - Teemahaastattelu. Luettu 19.2.2016. https://www.stat.fi/virsta/tkeruu/04/03/

Tuomi, J. & Sarajärvi, A. 2009. 5.painos. Laadullinen tutkimus ja sisällönanalyysi. Helsinki: Tammi.

Tuovinen, A-P. 2013a. Dynaaminen analyysi 1. Helsingin yliopisto. Luettu 25.4.2016. https://www.cs.helsinki.fi/u/aptuovin/testaus/Ohj_testaus_2013_6.pdf

Tuovinen, A-P. 2013b. Ohjelmistotestauksen perusteita 2. Helsingin yliopisto. Luettu 1.5.2016. https://www.cs.helsinki.fi/u/aptuovin/testaus/Ohj_testaus_2013_2.pdf

Tuovinen, A-P. 2013c. Staattinen testaus. Helsingin yliopisto. Luettu 25.4.2016. https://www.cs.helsinki.fi/u/aptuovin/testaus/Ohj_testaus_2013_5.pdf

Watkins, J. & Mills, S. 2011. 2nd edition. Testing IT: An Off-the-Shelf Software Testing Process. New York, USA: Cambridge University Press.

Zimmerer, P. 2010/09. The Value of Testing in 5 Dimensions. Testing Experience. The Magazine for professional testers. Luettu 20.4.2016. http://www.istqb.org/images/Articles/zimmerer_The%20Value%20of%20Testing%20in%205%20Dimensions.pdf

Äyrämö, S. 2010. TIES546 Ohjelmistotestaus. Tietotekniikan laitos. Jyväskylän yliopisto. Luettu 1.5.2016. http://users.jyu.fi/~samiayr/testaus2010/L1-2_testaus_johdanto.pdf

LIITTEET

Liite 1. Haastattelukysymykset pääkäyttäjille.

1. Minkä tietojärjestelmän/-järjestelmien pääkäyttäjänä toimit?
   Kuinka pitkään olet toiminut tietojärjestelmän pääkäyttäjänä?

2. Kuinka usein tietojärjestelmään keskimäärin tehdään muutoksia?

3. Minkälaisia muutoksia versiopäivitykset yleensä sisältävät/koskevat?

4. Kun määritellään ja tilataan Laurealle räätälöity uusi tietojärjestelmä tai versiopäivitys, miten testausvastuista on sovittu toimittajan kanssa?

5. Minkälaista testausta toimittaja on tehnyt tietojärjestelmälle/versiopäivitykselle?

6. Miten toimittajan testauksen laatu on varmistettu?

7. Onko mielestäsi toimittajan vastuulla olevassa testauksessa kehitettävää? Jos on, miten itse kehittäisit sitä?

8. Entä minkälaista testausta Laureassa on tehty uudelle tietojärjestelmälle/versiopäivitykselle?

9. Miten testausta käytännössä tehdään: Kuinka testaus suunnitellaan? Miten itse testaus suoritetaan? Entä kuinka testitulokset raportoidaan? Ketkä ovat osallistuneet testaukseen?

10. Onko mielestäsi Laurean tekemässä testauksessa kehitettävää? Jos on, miten itse kehittäisit testausta?

11. Koetko, että tarvitsisit apua testauksessa? Pystytkö määrittelemään tarkemmin minkälaista apua?

12. Miten uusien tietojärjestelmien/versiopäivityksien tuotantokäyttöönotto on onnistunut?

13. Jos tuotantokäyttöönotossa olisi mielestäsi kehitettävää, miten muuttaisit sitä?

14. Saatko tukea tuotantokäyttöönottoon liittyvissä ongelmissa? Keneltä saat tukea tai keneltä tarvitsisit enemmän tukea?

Liite 2. Haastattelukysymykset projektipäälliköille

1. Kun määritellään ja tilataan Laurealle räätälöity tietojärjestelmä, miten testausvastuista on sovittu toimittajan kanssa?

2. Minkälaista testausta toimittaja on tehnyt tietojärjestelmälle?

3. Miten toimittajan testauksen laatu on varmistettu?

4. Onko mielestäsi toimittajan vastuulla olevassa testauksessa kehitettävää? Jos on, miten itse kehittäisit sitä?

5. Entä minkälaista testausta Laureassa on tehty tietojärjestelmälle?

6. Miten testausta käytännössä tehdään: Kuinka testaus suunnitellaan? Miten itse testaus suoritetaan? Entä kuinka testitulokset raportoidaan? Ketkä ovat osallistuneet testaukseen?

7. Onko mielestäsi Laurean tekemässä testauksessa kehitettävää? Jos on, miten itse kehittäisit testausta?

8. Miten tietojärjestelmän tuotantokäyttöönotto on onnistunut?

9. Jos tuotantokäyttöönotossa on kehitettävää, miten muuttaisit sitä?

Liite 3. Listaus kaikista haastatteluissa tehdyistä havainnoista

Kuinka testausta tehdään nyt:

- "Testausta ei ole etukäteen suunniteltu vaan testausta tehdään tuntuman mukaan." (1)
- Ei tehdä suunnitelmallista testausta, lähinnä kokeillaan että ohjelma toimii. (1)
- Omaa testausta ei ole suunniteltu eikä dokumentoitu. (5)
- Enemmän Ad-Hoc testausta. Jokainen testaaja kokeilee omasta näkökulmastaan ohjelman toimivuutta. (2)
- Ei erityisesti suunniteltu eikä dokumentoitu testausta. Ei testiraporttia. Päätös tehdään siltä pohjalta, että "ohjelmaa on kokeiltu ja se tuntuu toimivan". (1)
- "Uusi ohjelmistoversio asennetaan muille asiantuntijoille testikäyttöön, mutta ei tietoa ovatko testanneet uutta versiota. Jos mitään ei kuulu niin kaikki on hyvin." (2)

- Varsinaista testausta ei tehdä, paitsi jos kyse on merkittävistä uusista toiminnallisuuksista. (1)
- Uuden ohjelmistoversion muutokset nähty niin pieninä, että versiota ei ole testattu vaan otettu suoraan tuotantokäyttöön. Käytön myötä selviää onko ongelmia. (1)
- "Pääkäyttäjä on vaihtelevasti kokeillut uutta versiota, kun se on asennettu. Niin iso ohjelma, paljon toiminnallisuuksia, että kaiken testaus veisi monta päivää. " (1)
- "Ongelmia ollut sen verran vähän, että otettu se riski, että jätetään testaus tekemättä." (1)
- Testausta tehdään eri ympäristöissä. (1)
- Varmuuskopiot järjestelmästä otetaan, mutta ei ole testattu saadaanko järjestelmä palautettua varmuuskopioista. (1)
- "Onkohan varmuuskopiointi toiminnassa?" (1)
- Loppukäyttäjiä mukana testauksessa. (3)

Testiympäristön käyttö testauksessa:

- Testiympäristö on käytössä. (7)
- "Testiympäristö ei ole käytössä "päivityksiä ei ole testattu viime aikoina testiympäristön kautta, kun ei ole ollut ongelmia"" (1)
- Ei ole sellaista testausympäristöä, jossa olisi rakennettuna kaikki käytössä olevat rajapinnat ja jossa olisi iso massa käyttäjiä. Usein rajapintoihin liittyviä ongelmia löydetään vasta tuotannossa. (1)
- Uusi versio otetaan suoraan tuotantokäyttöön. (1)
- Testaus tehdään tuotannossa, koska näitä järjestelmiä käytetään vain tiettynä ajankohtina lukuvuoden aikana. (2)

Uusi ohjelmisto/-versio tuotantoon:

- Tärkeätä suunnitella milloin otetaan uusi ohjelmistoversio tuotantokäyttöön. (4)
- Palautussuunnitelma ei ole tehty. Oletus on, että pitäisi olla mahdollista asentaa edellinen versio, jos tulee ongelmia uuden version viennissä tuotantoon. (6)
- Kun tuotantokäytössä ilmenee ongelmia, pääkäyttäjällä ei ole tietoa mihin tahoon pitäisi ottaa yhteyttä, jotta ongelman selvittely saataisiin heti käyntiin. (1)

Toimittajan suorittama testaus:

- Ei ole tietoa minkälaista testausta toimittaja on tehnyt. (6)
- Toimittajalta tulee listaus tehdyistä muutoksista. (10)
- Luotetaan, että toimittaja on varmistanut asiat, jotka on sovittu hänen vastuulleen. Ei ole vaadittu raporttia toimittajan testauksesta. (1)
- Epätietoisuutta mikä on Otaverkon vastuulla palvelimen suhteen ja mikä ei. (1)

Haastatteluissa esille tulleita kehitysideoita:

- Olisi hyvä olla tarkastuslista asioista mitkä pitäisi varmistaa ennen kuin uusi versio otetaan tuotantokäyttöön. (1)
- Testiraportti toimittajalta voisi olla hyvä idea. (5)
- Toimittajan kanssa sopimusta tehtäessä, pitäisi sopia myös toimittajan vastuulla olevan testauksen laajuus ja sen raportoinnista. (1)
- Vaatimusmäärittelyssä pitäisi ottaa kantaa myös testauksen vaatimuksiin. Esim. ei-toiminnallisia vaatimuksia (kuten kuormittavuustestaus) ei juurikaan ole pohdittu. (2)
- Voisi dokumentoida mitkä ovat ne perustestit mitkä olisi hyvä testata aina kun uusi versio tulee. (3)
- Testauskuormaa voisi jakaa pääkäyttäjien esken. (2)
- Testitapaussuunnitelman luominen määrittelyn pohjalta eli käyttötapauksien kuvaus ja näiden katselmointi. Testituloksien raportointi suunnitelman pohjalta. (3)
- Vaatimukset pitäisi kuvata käyttötapauksina, jotta nähdään liitännäisvaikutukset. (1)
- Rajapintakuvaus, mitä tietoja tietoja omasta järjestelmästä liikkuu minnekin. Ymmärrys mihin tehdyt muutokset vaikuttavat. (3)
- Voisimme tehdä listan meille tehdyistä ohjelmistoräätälöinneistä ja varmistaa, että ne toimivat edelleen uuden ohjelmistopäivityksen jälkeenkin. (1)
- Jos hyväksymistestauksessa löytyy kriittinen ongelma, hyväksymistestaus pysähtyy. (1)
- Demo tai käyttökoulutus ennen hyväksymistestauksen aloittamista. (2)
- Uuden tietojärjestelmän pilotointi tuotantoympäristössä olisi tärkeätä tehdä ennen tuotantokäyttöönottoa. (1)
- Sopimustasolla sanktioita, jos hyväksymistestauksen alkaessa järjestelmä ei ole sillä tasolla kuin pitäisi olla. Liian usein vain asiakkaan testausaika joustaa, koska julkaisuaika on fiksattu. (1)
- Integraatiot ja data migraatio pitää huomioida. Näiden läpikäynti toimittajan kanssa ja huomioiminen testauksen suunnittelussa. (1)
- Jos ei ole räätälöity järjestelmä, pitäisi varmistaa että toimittaja on testannut ohjelmiston eri ympäristöissä ja eri selaimilla. (1)
- Voisi olla hyvä tietää miten testausta tehdään ja mihin testauksessa kannattaisi kiinnittää huomiota. (3)