RAPORTTIEN KEHITTÄMINEN KETTERÄSSÄ YLLÄPITOPROSESSISSA

Marko Nuutinen

Opinnäytetyö

Ylempi ammattikorkeakoulututkinto

Savonia
ammattikorkeakoulu

SAVONIA-AMMATTIKORKEAKOULU **OPINNÄYTETYÖ**
Tiivistelmä

Koulutusala Tekniikan ja liikenteen ala			
Koulutusohjelma Hyvinvointiteknologian koulutusohjelma			
Työn tekijä(t) Marko Nuutinen			
Työn nimi Raporttien kehittäminen ketterässä ylläpitoprosessissa			
Päiväys	25.8.2010	Sivumäärä/Liitteet	56
Ohjaaja(t) Lehtori Jussi Koistinen			
Toimeksiantaja/Yhteistyökumppani(t) Release Coordinator Markku Hurskainen/Tieto Healthcare & Welfare Oy			

Tiivistelmä

Tämän opinnäytetyön aiheena oli tutkia, millaisia raportointitarpeita on eri sidosryhmillä Tieto Healthcare & Welfare Oy:n potilastietojärjestelmän jatkuvan ylläpitoprosessin eri vaiheissa, ja kehittää ketterän ylläpitoprosessin käyttöön soveltuvia raporttimalleja. Työssä on kuvattu ohjelmistotuotantoa erityisesti ketteriä kehitysmenetelmiä hyödyntävän kehitysprosessin kannalta.

Työssä toteutettiin kvalitatiivinen tutkimus, jossa perehdyttiin ylläpitoprosessissa käytettävän Visual Studio Team System -järjestelmän raportoinnin vakiotoimintoihin, haastateltiin sidosryhmiä sekä etsittiin tietoa järjestelmän ominaisuuksista kirjallisuudesta sekä internetistä.

Tutkimuksessa kävi ilmi, että ylläpitoprosessin raportoinnin kehittämiselle on selkeästi tarvetta. Halutut raportit ovat kuitenkin monimutkaisia toteutettavia, ja raportoinnin kehittämiselle tulisikin varata resursseja.

Varsinainen raporttien kehitys ei kuulunut tämän työn piiriin. Tutkimuksen perusteella valittiin kehitettäväksi tärkeimmät raportit ja luotiin niistä jatkokehityksen mahdollistavat mallit.

Avainsanat
ketterä ohjelmistokehitys, VSTS, raportointi

SAVONIA UNIVERSITY OF APPLIED SCIENCES **THESIS**
Abstract

Field of Study Technology, Communication and Transport			
Degree Programme Degree Programme in Welfare Engineering			
Author(s) Marko Nuutinen			
Title of Thesis Development of Reports for the Agile Maintenance Process			
Date	25.8.2010	Pages/Appendices	56
Supervisor(s) Mr Jussi Koistinen, Lecturer			
Project/Partners Mr Markku Hurskainen, Release Coordinator/ Tieto Healthcare & Welfare Oy			

Abstract

The aim of this thesis was to research what kind of reporting features the stakeholders of Tieto Healthcare & Welfare Oy's patient information system need in continuous maintenance process. Another task was to develop report models that are suitable for the agile maintenance process.

The research was carried out as a qualitative research. The material of this research was collected by interviewing stakeholders, analyzing Visual Studio Team System's default reporting functions and studying system features from literature and the internet.

The findings of the research showed that there is a definite need for report development. Desired reports are complex and designated resources should be reserved for reporting development.

Based on the research, the most important reports were selected and models of the reports were made for further development.

Keywords
agile software development, VSTS, reporting

SISÄLTÖ

1 JOHDANTO

Laajojen tietojärjestelmien ylläpito on useimmiten haasteellista riippumatta järjestelmän toimialasta. Erityisen haasteellista on ympärivuorokautisessa käytössä olevan kriittisiä ja salassa pidettäviä tietoja sisältävän järjestelmän ylläpito. Tieto Healthcare & Welfare Oy:n (jatkossa Tieto) valmistamaa potilastietojärjestelmää käytetään potilastietojen hallintaan suurissa sairaaloissa sekä pienemmissä terveyskeskuksissa ympäri Suomea. Käytössä olevien työasemien määrät vaihtelevat terveyskeskusten muutamista kymmenistä sairaaloiden useisiin tuhansiin. On selvää, että terveydenhuollon tietojärjestelmien on oltava käytettävissä jatkuvasti ympärivuorokautisesti ja päivitysten aiheuttamien käyttökatkojen on oltava mahdollisimman lyhyitä. Lisäksi on äärimmäisen tärkeää, että ohjelmistojen päivitykset toimivat siten, kuin niiden on tarkoitettu toimivan.

Potilastietojärjestelmä on käytössä kymmenillä erisuuruisilla organisaatioilla, minkä vuoksi päivitystahti on vaihtelevaa ja kaikki asiakkaat eivät käytä täsmälleen samoja versiota järjestelmän osasovelluksista. Tämä aiheuttaa tietojärjestelmän ylläpidolle entisestään lisää haasteita, koska samoja korjauksia ja kehitystoiveita joudutaan toteuttamaan useaan eri osasovellusversioon.

Virhekorjausten ja kehitystoiveiden hallintaan on kehitetty OpenUP-prosessimallia hyödyntävä ylläpitoprosessi, jonka avulla tuotetaan asiakkaille kuukausittain uusia versioita potilastietojärjestelmän osasovelluksista. Tuotettavista julkaisuista käytetään nimeä Monthly Fix. Tärkein ylläpitoprosessin hallintaan käytettävä työkalu on Microsoft Visual Studio Team System 2008 (VSTS) -järjestelmä. VSTS:n muutospyyntöjen hallinta perustuu niin sanottuihin työkortteihin, joiden sisältöä, tilaa ja osoituksia seurataan ja päivitetään Monthly Fix -prosessin mukaisesti.

Opinnäytetyö on tehty omana kehitystehtävänä, kuitenkin osana Release & Quality -tiimin jokapäiväistä työtä. Release & Quality -tiimi vastaa Monthly Fix -prosessista sekä prosessin kehittämisestä. Tässä opinnäytetyössä tutkimusongelmana on selvittää, millaisia toiminnan tehokkuuden ja laadun mittarointia tukevia raporttimalleja tarvitaan Monthly Fix -kehitysprosessin eri vaiheisiin.

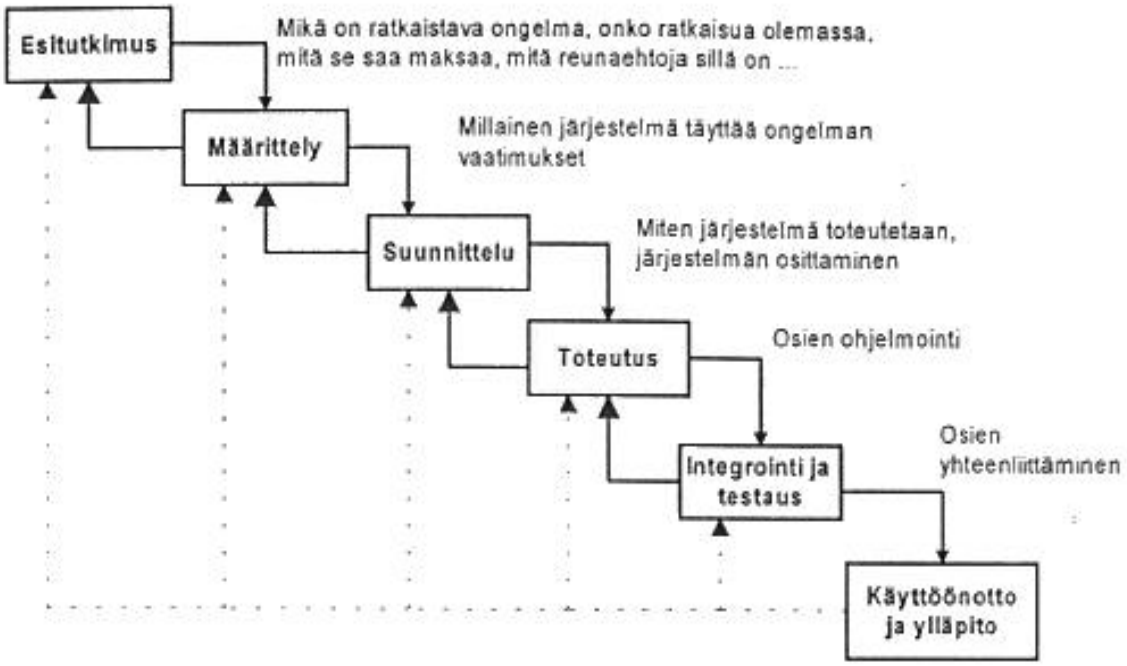
2 TEOREETTINEN TAUSTA JA TOIMINTAYMPÄRISTÖ

Tässä luvussa esitellään ohjelmistotuotannon perusteita, toimintaympäristö, taustalla oleva prosessimalli, käytettävät työkalut sekä organisaatio.

2.1 Ohjelmiston elinkaari

Ohjelmistotuotanto voidaan jakaa useisiin osa-alueisiin sen mukaan, millaisen ohjelmistokokonaisuuden tuotannosta on kyse. Yleensä tuotantoa ohjaa yrityksen laatujärjestelmä, joka määrittelee yrityksen toimintaprosessit ainakin yleisellä tasolla. Varsinainen ohjelmistotuotanto tapahtuu ohjelmistoprojekteissa, jotka voivat myös muodostaa yhdessä laajempia tuotekehityshankkeita. Kehitysprosessista voidaan yleensä erottaa selkeät vaiheet, jotka toistuvat ohjelmistoprojektista toiseen: määrittely, suunnittelu, ohjelmointi ja testaus. Näiden vaiheiden jälkeen tyypillisesti seuraa ohjelmiston käyttöönotto sekä ylläpito. Ohjelmistoprojektiin liittyy myös koko projektin ja ohjelmiston elinkaaren ajan kestäviä tukitoimintoja, tärkeimpinä laadunvarmistus, tuotteenhallinta ja dokumentointi. (Haikala ja Märijärvi 2004, 35.)

Ohjelmiston elinkaarella tarkoitetaan aikaa, joka kuluu ohjelmiston kehittämisen aloittamisesta sen käytöstä poistamiseen. Vaihejakomallilla puolestaan tarkoitetaan tapaa, jolla ohjelmiston kehitystyö tai koko elinkaari jaetaan vaiheisiin. Tavallisin vaihejakomalli on niin sanottu vesiputousmalli (Kuvio 2.1), jossa kehitysprosessin vaiheet seuraavat toisiaan järjestyksessä ja uuden vaiheen alkaminen edellyttää aina edellisen vaiheen päättymistä. Kaikkiin vaiheisiin liittyy laadunvarmistustehtäviä, kuten tarkastuksia, katselmuksia ja testausta. Tarkastusten ja testauksen avulla pyritään poistamaan kehitettävän järjestelmän virheet mahdollisimman aikaisessa vaiheessa. Katselmuksia pidetään yleensä vaiheiden päätteeksi. Katselmuksissa todetaan projektin tilanne ja varmistutaan, että kaikki vaiheelle asetetut tavoitteet on täytetty ja vaiheeseen liittyvät tehtävät tehty. (Haikala ja Märijärvi 2004, 35 - 36.)



KUVIO 2.1. Perinteinen ohjelmistotuotannon vesiputousmalli (Haikala ja Märijärvi 2004, 36).

Esitutkimus

Esitutkimuksen tehtävänä on asettaa yleiset järjestelmätason vaatimukset, joita usein kutsutaan myös asiakasvaatimuksiksi. Tällaiset vaatimukset määrittelevät asiakkaan tarpeet, mutta eivät ota lainkaan kantaa siihen, millainen järjestelmä täyttää asiakkaan vaatimukset. Esitutkimuksessa olennaisinta on selvittää asiakkaan todelliset tarpeet ja ymmärtää ne perusteellisesti. Jos asiakasvaatimukset ymmärretään tai käsitellään väärin, lopputuloksena ei voi olla asiakkaan kannalta hyvä järjestelmä. Sen vuoksi esitutkimus onkin ohjelmiston elinkaaren tärkein vaihe. Esitutkimus mielletään usein myös osaksi määrittelyvaihetta, koska asiakastarpeiden analysointi ja tarkentaminen yleensä jatkuu koko määrittelyvaiheen ajan. Toisaalta esitutkimus voidaan ymmärtää myös osaksi vaatimuksenhallinnaksi kutsuttua tukitoimintoa, erityisesti jos kyseessä on jatkuvaan tuotekehitykseen perustuva ympäristö. (Haikala ja Märijärvi 2004, 37.)

Määrittely

Määrittelyvaiheessa analysoidaan asiakasvaatimuksia ja niistä muodostetaan ohjelmistovaatimukset, jotka määrittelevät toteutettavan järjestelmän. Ohjelmistovaatimuksia nimitetään usein myös järjestelmävaatimuksiksi tai yksinkertaisesti ominaisuuksiksi. Kaikilla näillä nimityksillä kuitenkin tarkoitetaan asioita, joita järjestelmään aiotaan kehittää. Määrittelyssä kuvataan toiminnot, ei-toiminnalliset vaatimukset sekä

rajoitukset. Toiminnoissa määritellään ohjelmistolla toteutettavat ominaisuudet, järjestelmän kommunikointi muiden järjestelmien kanssa sekä käyttöliittymä. Ei-toiminnallisia vaatimuksia ovat esimerkiksi käytettävyys, suoritusteho ja vasteaika. Rajoituksiin luokitellaan esimerkiksi toteutuksessa käytettävä ohjelmointikieli. (Haikala ja Märijärvi 2004, 38 - 39.)

Suunnittelu

Suunnitteluvaiheessa suunnitellaan, kuinka toteutetaan määrittelyvaiheessa kuvatut toiminnot. Suunnitteluvaihe jaetaan usein useampiin tasoihin. Aluksi järjestelmä jaetaan itsenäisiin, toisista riippumattomiin osiin, niin sanottuihin moduuleihin. Tästä vaiheesta käytetään nimitystä arkkitehtuurisuunnittelu. Seuraavassa vaiheessa, moduulisuunnittelussa, suunnitellaan jokaisen yksittäisen moduulin sisäinen rakenne. (Haikala ja Märijärvi 2004, 40.)

Usein määrittelyn ja suunnittelun suhdetta havainnollistetaan siten, että määrittelyvaiheessa kuvataan, mitä järjestelmä tekee, ja suunnitteluvaiheessa kerrotaan, miten järjestelmä tehtävänsä suorittaa.

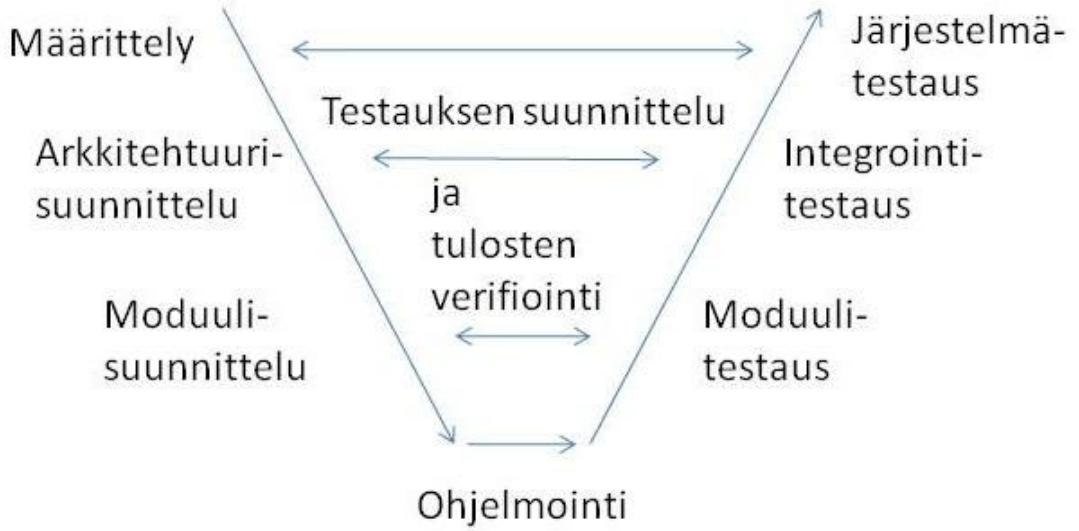
Toteutus

Ohjelmistotuotannossa toteutusvaihe tarkoittaa ohjelmiston kirjoitusvaihetta. Vaiheen päätös voi olla määritelty esimerkiksi ensimmäiseen virheettömään käännökseen. (Haikala ja Märijärvi 2004, 40.)

Testaus

Testauksen tarkoitus on löytää ohjelmistosta virheitä. Yleisesti testaus ohjelmistotuotannossa tapahtuu monella tasolla, niin sanotun testauksen V-mallin mukaisesti. V-mallissa testaus jaetaan moduulitestaukseen, integrointitestaukseen ja järjestelmätestaukseen (Kuvio 2.2).

Moduulitestauksessa testataan yksittäisen moduulin toimintaa ja etsitään siitä virheitä, integrointitestauksessa testataan moduulien välistä yhteistoimintaa ja järjestelmätestauksessa testaus keskittyy koko järjestelmän toimintaan ja suorituskykyyn (Haikala ja Märijärvi 2004, 40).



KUVIO 2.2. Testauksen V-malli (Haikala ja Märijärvi 2004, 289).

V-mallin mukaisessa testauksessa järjestelmätestauksen suunnittelu tehdään alustavasti osana ohjelmiston määrittelyä ja testauksessa verrataan valmista järjestelmää määrittelydokumentaatioon. Vastaavalla tavalla integrointitestaus suunnitellaan arkkitehtuurisuunnittelun yhteydessä ja moduulitestaus moduulisuunnittelun yhteydessä. (Haikala ja Märijärvi 2004, 40.)

Ylläpito

Ylläpitovaiheessa ohjelmistosta ratkotaan asiakkaan ilmoittamia ongelmia, korjataan virheitä ja muutetaan ohjelmistoa vaatimusten muuttuessa. Ohjelmistoon voidaan myös lisätä kokonaan uusia piirteitä, tosin yleensä isot muutokset ja ominaisuuksien lisäys toteutetaan tuotteen seuraavaan versioon tähtäävässä projektissa. (Haikala ja Märijärvi 2004, 41.)

2.2 Ketterät menetelmät

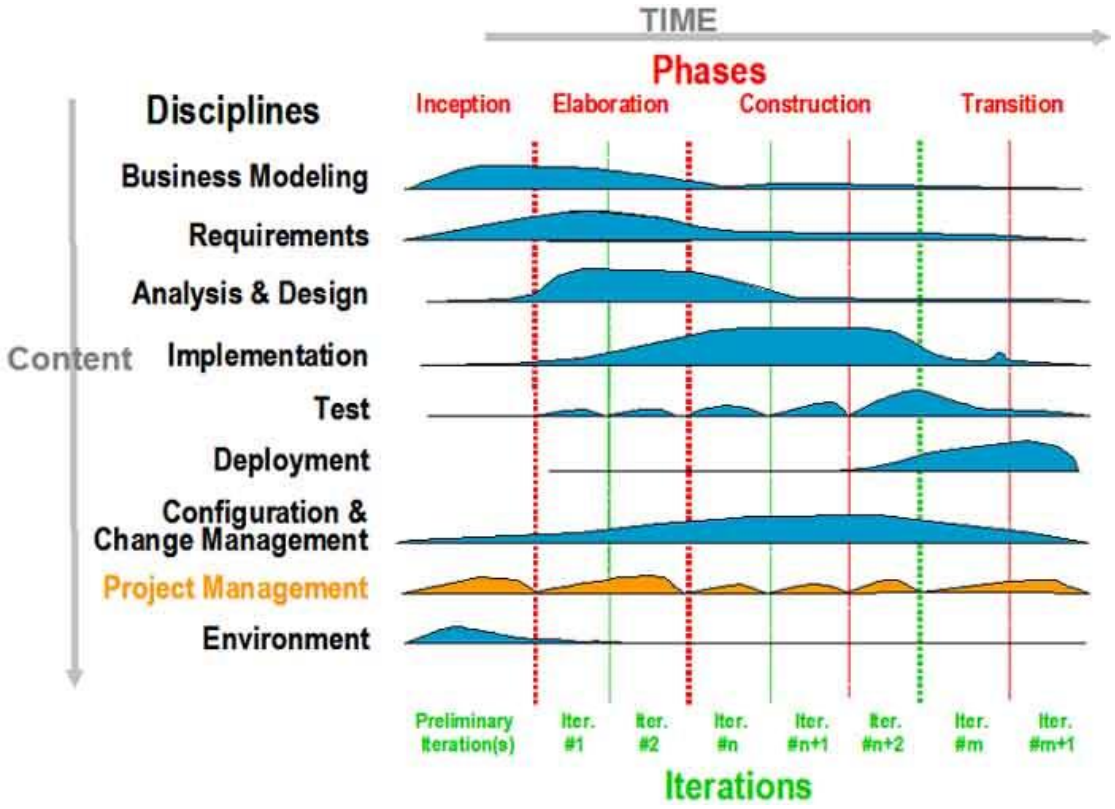
Ohjelmistokehitys on siis perinteisesti seurannut niin sanottua vesiputousmallia, jossa kehitys etenee vaiheittain ja tiukassa loogisessa järjestyksessä. Mallin ominaispiirteisiin kuuluu tarkat suunnitelmat dokumentteineen ja määrätyt hyväksyntäkriteerit jotka täytyy täyttyä ennen seuraavaan vaiheeseen siirtymistä. (Deemer ym. 2010, 3.)

Vesiputousmallin vahvuus on loogisuus ja tarkasti organisoitu prosessi, suurin heikkous se, että mallia on toteuttamassa ihmiset. Ihmisillä on tapana keksiä uusia ideoita

asioiden ratkaisemiseksi myös kehitysprosessin aikana, jopa kehityksen loppuvaiheessa, mutta siihen vesiputousmalli ei pysty mukautumaan, koska kaikki on suunniteltu jo prosessin alussa. (Deemer ym. 2010, 3.)

Ketterät kehitysmallit ovat kehittyneet lähemmäksi ihmisen todellisia käyttäytymismalleja. Ketterissä kehitysmenetelmissä pyritään tekemään jo projektin alkuvaiheessa toiminnallisia kokonaisuuksia, ohjelmistotekniikassa siis toimivia ohjelmien osia, eikä keskitytä kirjoittamaan tarkkoja määrittelyjä koko järjestelmän toiminnasta. Lisäksi niissä keskitytään nopeasti toistettaviin iteraatioihin ja jatkuvaan kehityksen aikaiseen asiakaspalautteeseen. (Deemer ym. 2010, 4.)

Rational Unified Process (RUP) (Kuva 2.1) oli ensimmäinen yleisesti käytetty iteratiivinen ohjelmistokehitysprosessi, mutta sen kompleksisuus ja koko tekee sen vaikeasti käyttöönotettavaksi. Ketterät kehitysprosessit kuten Scrum tai Extreme Programming (XP) ovat kevyempiä kuin RUP, mutta niiden erilainen luonne ja vajavaisempi dokumentaatio johtivat aluksi usein johtoportaan vastustukseen. (Gustafsson 2010, 1.)



KUVA 2.1. IBM:n Rational Unified Processin perusteet (Cottrell 2004).

Vuonna 2001 julkaistu Agile Manifesto (Agile Manifesti 2001) toi täysin uusia ideoita ohjelmistokehitysprosessien maailmaan. Vaikka RUP-prosessin syrjäyttäminen ei ollutkaan päätarkoitus, ketterät menetelmät vaikuttivat monien mielestä hyviltä. (Gustafsson 2010, 1.)

2.3 Agile Manifesto

Agile Manifesto syntyi vuonna 2001, kun 17 merkittävää ketterän kehityksen puolestapuhujaa kokoontui keskustelemaan menetelmiensä yhteisestä perustasta. Tarkoituksena oli luoda yhteistä pohjaa ketterille menetelmille ja edistää ketterän ajattelun leviämistä. Kokoontumisen tuloksena julkaistiin Agile Manifesto (Ketterä Manifesti), jota pidetään ketterän kehityksen perusmääritelmänä. Manifesto määrittelee ketterille menetelmille neljä tyypillistä arvoa sekä 12 periaatetta (Agile Manifestin Periaatteet 2001), joita menetelmät noudattavat.

Ketterä Manifesti:

”Me etsimme parempia keinoja ohjelmistojen kehittämiseen tekemällä sitä itse ja auttamalla siinä muita. Tässä työssämme olemme päätyneet arvostamaan

Yksilöitä ja vuorovaikutusta enemmän kuin prosesseja ja työkaluja
Toimivaa sovellusta enemmän kuin kokonaisvaltaista dokumentaatiota
Asiakasyhteistyötä enemmän kuin sopimusneuvotteluita
Muutokseen reagoimista enemmän kuin suunnitelman noudattamista.

Vaikka oikeallakin puolella on arvoa, me arvostamme vasemmalla olevia asioita enemmän.” (Agile Manifesti 2001).

2.4 OpenUP-menetelmä

OpenUP on minimaalinen ohjelmistokehitysprosessi, jossa on määritelty vain keskeiset asiat. Se on kehitetty osana avointa Eclipse Process Framework (EPF) –projektia, ja sen perustana on IBM:n Rational Unified Process (RUP), joka on paljon laajempi ja monipuolisempi prosessimalli. (Kroll 2007.)

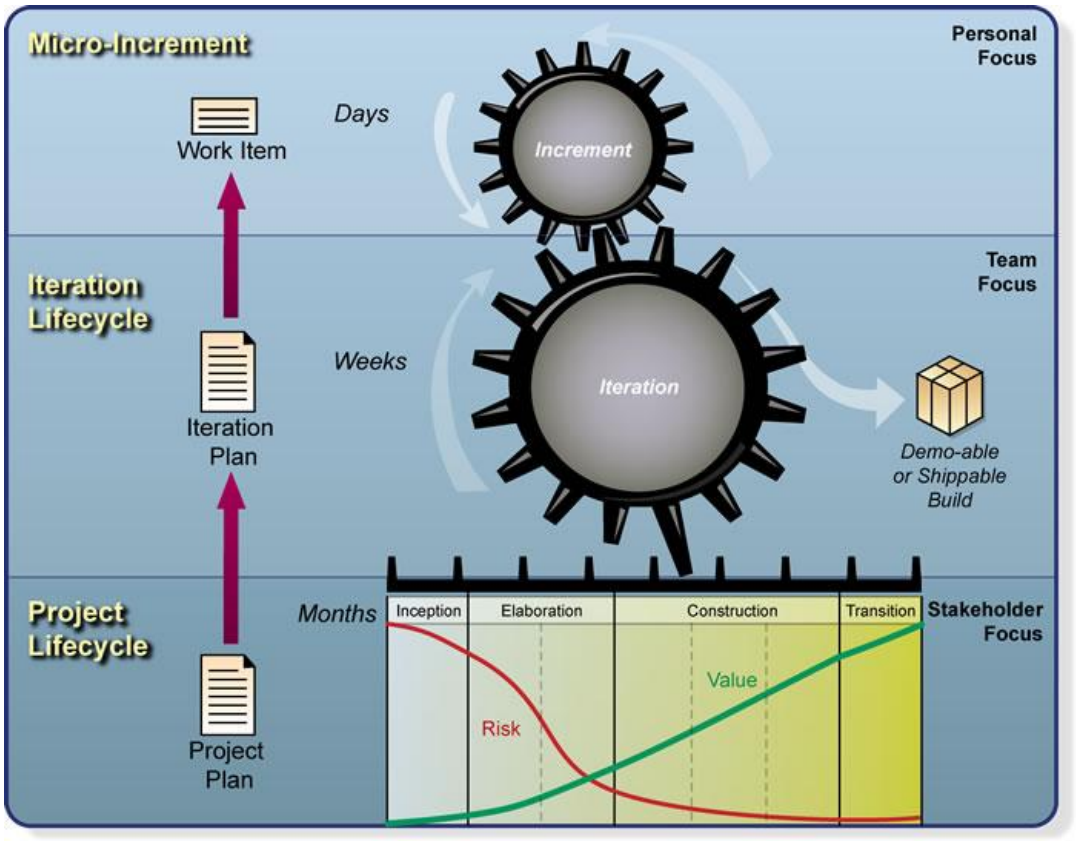
Vain keskeiset asiat sisältävänä OpenUP ei tarjoa ohjeita kaikkiin aiheisiin, joita projekteissa joudutaan käsittelemään, esimerkiksi sopimuksiin tai käytettäviin teknologioihin. OpenUP on kuitenkin täydellinen siinä mielessä, että sen avulla voidaan havainnollistaa koko ohjelmiston rakentamiseen tarvittava prosessi. OpenUP-prosessin sisältöä on mahdollista muokata myös kattamaan asioita, joita siihen ei valmiiksi kuulu, joten se on hyvä perusta ketterälle ohjelmistoprosessille. (Eclipse Foundation 2007, 1.)

2.4.1 OpenUP:n historiaa

OpenUP-kehityksen lähtökohtana oli kehittää kevyempi, ketterä versio RUP:sta hyödyntäen samalla muiden ketterien prosessien, kuten Scrumin ja XP:n, parhaita puolia. OpenUP:n kehitystyö aloitettiin IBM:ssä, mutta nopeasti todettiin, että tarvittiin laajempaa näkemystä kehitykseen ja kehitys siirrettiin Eclipse-säätiölle osaksi EPF-projektia. (Kroll 2007.)

2.4.2 OpenUP-kerrokset

OpenUP-prosessissa yhdistetään iteratiivinen ja inkrementaalinen lähestymistapa rakenteiseen elinkaarimalliin. OpenUP kattaa käytännöllisen, ketterän filosofian, joka keskittyy ohjelmistokehityksen yhteistyöluonteeseen. OpenUP-prosessi muodostuu kuvassa 2.2 näkyvistä kerroksista, Projektin elinkaaresta, Iteraation elinkaaresta ja Mikroinkrementeistä. (Kroll 2007.)

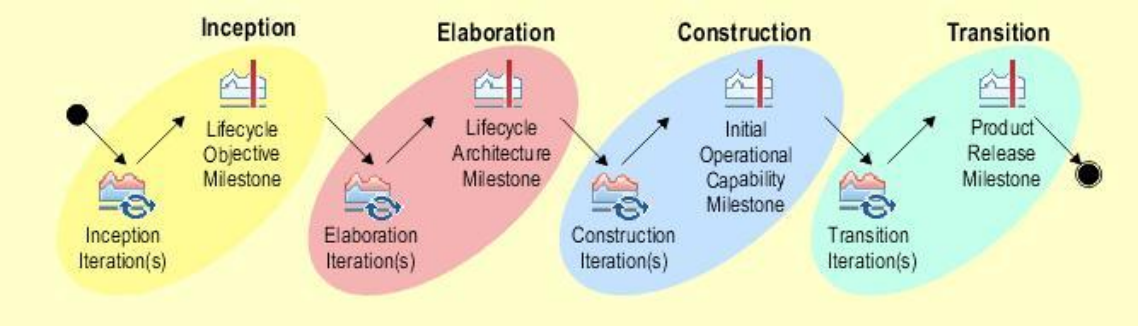


KUVA 2.2. OpenUP-kerrokset (Eclipse Foundation 2008).

Projektin elinkaari

Projektin elinkaarimalli (Kuva 2.3) tarjoaa projektin sidosryhmille mahdollisuuden valvoa ja ohjata projektin kuluja, laajuutta, riskejä ja muita projektin läpiviennin kannalta olennaisia asioita. OpenUP-prosessissa iteraatiot organisoidaan projektin vaiheiksi. Jokainen vaihe päättyy tarkastuspisteeseen, jonka tarkoitus on luoda sidosryhmälle mahdollisuus valvoa projektin etenemistä vastaamalla tärkeisiin kysymyksiin ja nostamalla esiin uusia:

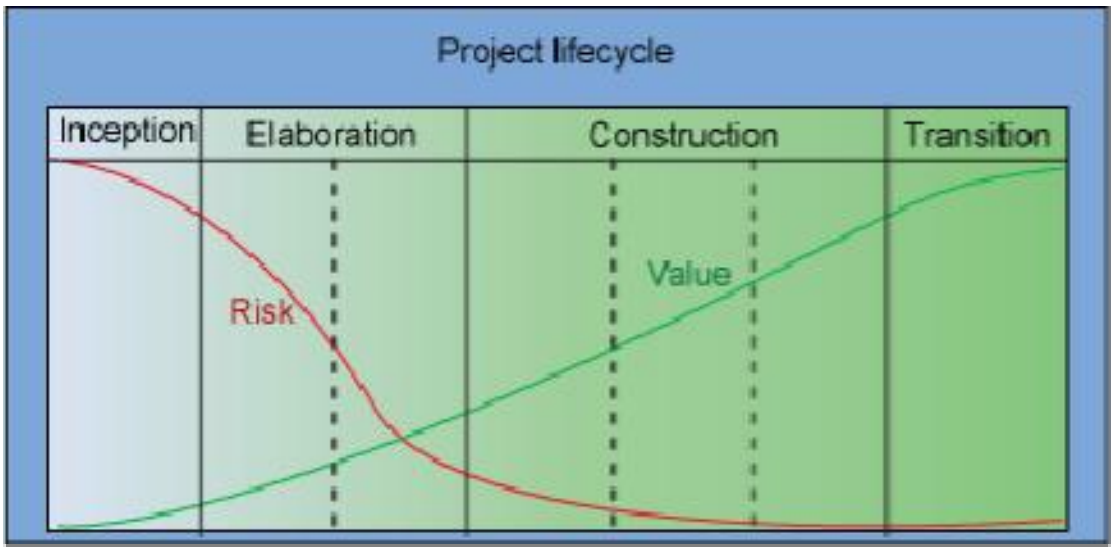
- **Aloitusvaihe** (Inception): Ollaanko projektin laajuudesta ja päämäärästä yksimielisiä ja pitäisikö projektin edetä seuraavaan vaiheeseen vai ei?
- **Tarkennusvaihe** (Elaboration): Ollaanko sovelluskehityksen arkkitehtuurista yksimielisiä? Onko tähän mennessä tuotettu arvo ja jäljellä oleva riski hyväksyttävällä tasolla?
- **Rakennusvaihe** (Construction): Onko tuotettu sovellus, joka on riittävän lähellä julkaisua, että voidaan siirtää kehityksen painopistettä viimeistelyyn ja sujuvan käyttöönoton varmistamiseen?
- **Siirtymävaihe** (Transition): Onko sovellus valmis julkaistavaksi?



KUVA 2.3. Projektin elinkaarimalli (Eclipse Foundation 2008).

Jos vastaukset kysymyksiin vaiheen katselmoinnissa ovat myönteisiä, projekti jatkuu. Jos vastaukset ovat kielteisiä, vaihe viivästyy, kunnes päästään haluttuun lopputulokseen tai sidosryhmät päättyvät projektin lakkauttamiseen. Vaiheen viivästyminen toteutetaan yleensä lisäämällä vaiheeseen lisäiteraatio, jossa toteutetaan vaiheen päättämisen mahdollistavat asiat. (Kroll 2007.)

Yksi elinkaarimallin päätavoitteista on keskittyä sidosryhmien tärkeimpiin kiinnostuksenkohteisiin: riskien pienentämiseen ja lisäarvon tuottamiseen (Kuva 2.4). OpenUP-vaiheet keskittävät tiimiä riskien pienentämiseen vaiheiden lopussa vastattavien kysymysten näkökulmasta ja samalla seuraamaan lisäarvon tuottamista. (Kroll 2007.)



KUVA 2.4. Projektin vaiheet riskin ja lisäarvon suhteessa (Kroll 2007).

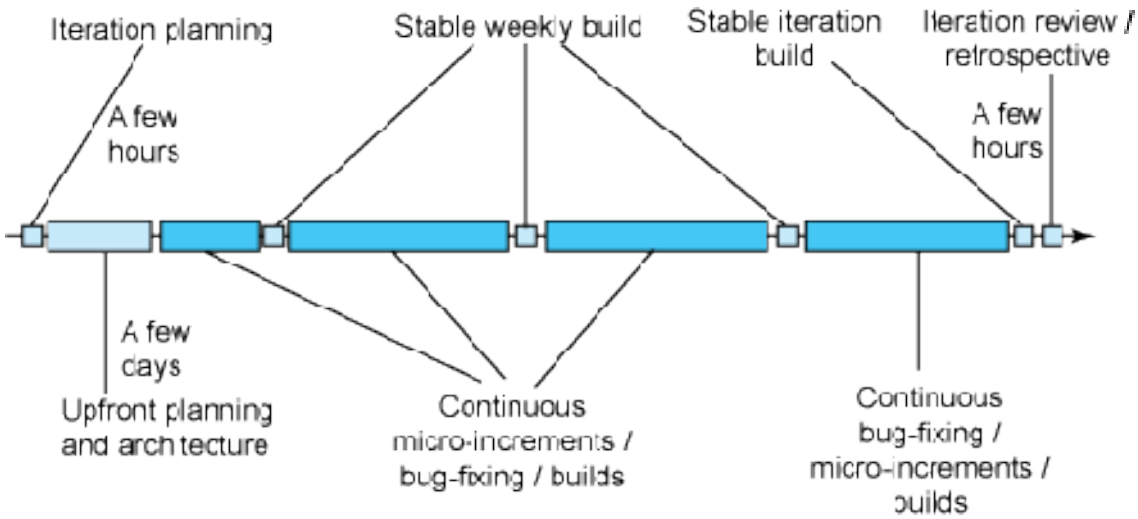
Elinkaarimallin avulla on mahdollista kiinnittää huomiota projektin laajuuteen ja ratkaisuihin jo alkuvaiheessa ennen täysimittaisen toteutusvaiheen aloittamista. (Kroll 2007.)

Iteraation elinkaari

OpenUP jakaa projektin iteraatioihin. Iteraatiolla tarkoitetaan suunniteltua, tiettyyn aikaikkunaan sidottua toistuvaa tehtäväkokonaisuutta, jota voi ajatella pienenä ohjelmistoprojektina projektin sisällä. Yhden iteraation kesto mitataan yleensä viikoissa. Jokainen iteraatio hyödyntää edellisen iteraation tuotoksia ja siirtää projektia askeleen lähemmäksi lopullista päämäärää. Iteraatiot ovat aikasidonnaisia, joka tarkoittaa, että iteraation aikataulun täytyy ajatella olevan muuttumaton. Sen sijaan iteraation sisällön valmistumista seurataan aktiivisesti ja sisältöä muutetaan tarvittaessa vastaamaan aikataulua. (Kroll 2007.)

Iteraatioiden tarkoitus on saada tiimi tuottamaan vähitellen lisäarvoa projektille, tällöin saadaan tuotettua toiminnallisia kokonaisuuksia jo projektin varhaisessa vaiheessa. Iteraatiosuunnitelma määrittelee mitä iteraation aikana pitäisi toteuttaa. Jokaisessa iteraatiossa pitäisi valita toteutettavaksi projektin kannalta kriittisimmät asiat ja korkeimmalla prioriteetilla määritellyt tehtävät. Tällöin varmistutaan, että jokainen iteraatio tuottaa mahdollisimman paljon hyötyä sidosryhmille. (Kroll 2007.)

Iteraation lopputuloksena on yleensä demottava tai toimituskelpoinen käännös ohjelmistosta. Kuvassa 2.5 esitellään yhden iteraation sisältämiä vaiheita. (Kroll 2007.)



KUVA 2.5. Iteraation elinkaari (Kroll 2007).

Mikroinkrementti

Yksittäisen projektihenkilön osalta OpenUP-projekti organisoidaan mikroinkrementteina. Mikroinkrementit edustavat lyhyttä muutamasta tunnista muutamaan päivään kestävää työtehtävää, joiden valmistuminen tuottaa tasaisen, mitattavan tahdin ku-

vaamaan kokonaisprojektin etenemistä. Prosessi lisää tiivistä yhteistyötä, koska järjestelmän on asteittain kehittänyt sitoutunut, itseohjautuva tiimi. (Kroll 2007.)

Mikroinkrementit mahdollistavat erittäin nopean palauteketjun, joka ohjaa mukautuvaa päätöksentekoa jokaisen iteraation sisällä. Ne auttavat myös tiimin yksittäisen jäsenen osittamaan oman työnsä pienempiin osiin siten, että jokainen tuottaa mitattavissa olevaa arvoa tiimin kokonaispanokseen. (Kroll 2007.)

Mikroinkrementit täytyy olla tarkasti määriteltyjä ja jokaisen mikroinkrementin edistymistä täytyy pystyä seuraamaan päivittäisellä tasolla. (Kroll 2007.)

2.5 Vaatimustenhallinta

Riippumatta ohjelmistotuotantoon käytetystä menetelmästä, perimmäinen tavoite on saada tuotettua asiakasvaatimusten perusteella asiakasvaatimukset ja sopimukset täyttävä ohjelmisto. Tuon tavoitteen täyttymisen varmistamiseen liittyviä toimenpiteitä kutsutaan yhteisesti vaatimustenhallinnaksi. Vaatimustenhallinnan avulla varmistetaan, että lopputuote vastaa asiakkaan vaatimuksia ja että tuote sisältää kaikki halutut ominaisuudet ja ei mitään ylimääräistä. (Haikala ja Märijärvi 2004, 91.)

Yleensä vaatimustenhallintaa ei käsitellä erillisenä tukitoimintona, vaan siihen kuuluvat asiat sisällytetään ohjelmistotuotannon muihin osa-alueisiin, pääasiassa määrittelyn ja esitutkimukseen sekä tuotteen- ja projektinhallintaan. Oppikirjamaiset näkemykset systemaattisista prosesseista eivät kuitenkaan käytännössä useimmiten toteudu. Kehitysprosessin aikana esille tulevia vaatimusmuutoksia on tietenkin pyrittävä välttämään mahdollisimman paljon, koska kehityksen aikaiset muutokset vaatimuksiin aiheuttavat suuria lisäkustannuksia. Käytännössä on kuitenkin varauduttava siihen, että kaikkia asiakasvaatimuksia ei voida etukäteen tuntea ja ymmärtää tarpeeksi hyvin, ja osa vaatimuksista muuttuu joka tapauksessa projektin aikana. (Haikala ja Märijärvi 2004, 92.)

Suurissa tuotekehityshankkeissa on yleensä paljon asiakasvaatimuksia ja ne muuttuvat useaan kertaan projektin aikana. Käyttäjäkunnan vaatimukset saattavat lisäksi olla erilaisia ja ristiriidassa toistensa kanssa. Ohjelmistotuotannossa tuotteen elinkaari jatkuu usein tuotteen seuraavien versioiden kautta, tällöin vaatimustenhallinta ei voi olla pelkästään projektin esitutkimus- ja määrittelyvaiheen toiminto, vaan ohjelmiston koko elinkaaren ajan kestävää jatkuvaa toimintaa. (Haikala ja Märijärvi 2004, 92.)

Muutostenhallinta

Asiakkaan vaatimukset ja niistä muodostetut ohjelmistovaatimukset selvitetään pääasiassa projektin esitutkimus- ja määrittelyvaiheessa, mutta tavallisesti vaatimuksiin tulee muutoksia koko projektin ajan. Viimeistään muutoksia tulee ylläpitovaiheessa ja ohjelmiston uutta versiota tehtäessä. Muutoksiin johtavia syitä on muun muassa:

- Asiakasvaatimusten väärin ymmärtäminen
- Asiakasvaatimusten huomiotta jättäminen
- Projektin aikana tehdyt toimintaympäristön muutokset
- Aikataulu
- Markkinatilanne, kilpailevat tuotteet

Esimerkiksi näistä syistä vaatimusten muutostenhallinta on keskeinen osa vaatimustenhallintaa. Projektissa onkin oltava sovitut menetelmät muutostenhallintaa varten. Muutosten yleinen ongelma on, että yhteen projektin osaan tehty muutos vaikuttaa yleensä myös johonkin toiseen osaan, ja projektissa täytyy selvittää minne vaikutus kohdistuu ja millä tavalla. Muutosten jäljitettävyys eteen- ja taaksepäin on muutostenhallinnan kannalta olennaisen tärkeää. (Haikala ja Märijärvi 2004, 98 - 99.)

Muutostenhallinnan suurin etu on, että sen avulla tehdään mitä on suunniteltu tehtäväksi ja suojellaan projektia tarpeettomilta muutoksilta. Tällaiseen tilanteeseen pääseminen edellyttää sovittua muutostenhallintamenettelyä, jossa muutokset käsitellään systemaattisesti harkiten. Tarpeettomien ominaisuuksien lisääminen ohjelmistoon on suuri ja tarpeeton riski ohjelmiston toiminnan kannalta. Turhat ominaisuudet lisäävät ohjelmiston kompleksisuutta ja heikentävät sen vakautta. Ne myös korottavat kustannuksia ja venyttävät projektin aikataulua. Muutostenhallinta parantaakin projektissa tehtävien päätösten laatua, koska menettelyssä valvotaan, että kaikki asianosaiset ovat mukana tekemässä päätöksiä. Lisäksi muutoksista tiedotetaan kaikkia asianosaisia paremmin, jolloin projektiryhmän kyky seurata projektin etenemistä paranee. (McConnell 1998, 76 - 77.)

2.6 Kohdeorganisaatio

Tiedon potilastietojärjestelmän ylläpito ja kehitys tapahtuu useassa maantieteellisesti hajautetussa yksikössä. Sähköisten järjestelmien ja apuvälineiden avulla hajautunut sijoittuminen ei kuitenkaan aseta esteitä tai tuo ylimääräisiä ongelmia toiminnalle.

Organisaatio on jaettu tuotekehitys- ja asiakaspalveluorganisaatioihin. Monthly Fix -prosessin kannalta molempien organisaatioiden työpanos on merkittävässä osassa.

2.6.1 Tuotekehitys

Tuotekehityksen tehtäviin kuuluu muutospyyntöjen toteutus ja Monthly Fix -prosessin koordinointi.

2.6.2 Asiakaspalvelu

Asiakaspalvelun tehtävät muodostuvat Monthly Fix -prosessissa muutosten testauksesta, asiakastoimituksista huolehtimisesta ja muutosten priorisoinnista.

2.7 MF-prosessi

Potilastietojärjestelmän ylläpitoon on kehitetty Monthly Fix -prosessi, jonka avulla tuotannossa olevien versioiden muutokset saadaan toteutettua hallitusti aikataulussa, joka on asiakkaille riittävän nopea mutta jonka avulla muutokset saadaan toteutettua laadukkaasti ja niiden vaikutukset kokonaisuuteen saadaan testattua riittävän kattavasti.

Monthly Fix -prosessissa tuotepäälliköt seuraavat jatkuvasti oman tuotteensa tilaa ja tuotteeseen tulevia uusia muutospyyntöjä. Tuotepäälliköt priorisoivat Monthly Fixiin tulevat muutospyynnöt tärkeysjärjestykseen, ja noin viikkoa ennen uuden Monthly Fix -kierroksen aloittamista pidettävissä Product Council -palavereissa suunnitellaan, mitkä muutospyynnöistä toteutetaan tulevan Monthly Fix -kierroksen aikana.

Monthly Fix -prosessi on johdettu ketterästä OpenUP-menetelmästä, ja siinä on erotettavissa kaksi iteraatiota, joilla on selkeä elinkaari. Kehitysiteraation kesto on kuusi viikkoa. Se alkaa kahden viikon valmisteluajalla, jonka aikana tuotekehittäjät pystyvät suunnittelemaan tehtävien muutosten toteutusta. Seuraavien neljän viikon aikana tuotekehityksen tuotekehittäjät toteuttavat muutokset sovellusten lähdekoodiin ja esittelevät korjauksen asiakaspalvelun sovellusasiantuntijalle, joka hyväksyy tai hylkää toteutuksen. Samaan aikaan testisuunnittelijat laativat tuotekohtaisen regressiotestisuunnitelman, jossa huomioidaan kehitysiteraation aikana tuotteeseen tehtävät muutokset. Kehitysiteraation päättyessä kaikki suunnitellut muutokset tulee olla toteutet-

tuna ja testattuna. Mikäli muutos ei ole valmistunut, se siirretään tehtäväksi seuraavassa Monthly Fixissa. Kehitysiteraation aikataulu ei iteraation elinkaarimallin mukaisesti siis muutu, vain sisältö muuttuu tarvittaessa.

Kehitysiteraation päättyessä alkaa laadunvarmistusiteraatio. Iteraation aikana keskitytään testaamaan kokonaisuutta eikä enää painoteta yksittäisen muutoksen osuutta. Regressiotestaus tehdään tuotteittain, kehitysiteraation aikana laaditun regressiotestisuunnitelman mukaisesti. Mikäli laadunvarmistuksen aikana löydetään uusia virheitä, ne raportoidaan normaalisti VSTS-järjestelmään ja arvioidaan tilanteen mukaan, korjataanko ne välittömästi vai vasta tulevissa Monthly Fix -julkaisuissa. Myös laadunvarmistusiteraatio kestää kuusi viikkoa. Kolmen viikon regressiotestin jälkeen julkaisu on mahdollista toimittaa asiakastestaukseen. Iteraation lopussa varmistetaan, että julkaisukriteerit täyttyvät ja kokonaisuus siirretään toimitettavaksi asiakkaille.

2.8 Potilastietojärjestelmä

Tiedon potilastietojärjestelmä on käytössä kymmenillä asiakkailla eri puolilla Suomea. Järjestelmää on rakennettu useita vuosia, ja sen ylläpitoon sekä kehitykseen liittyy monenlaisia haasteita.

2.8.1 Tekninen ratkaisu

Teknisessä mielessä potilastietojärjestelmä on tyypillinen asiakas-palvelin-ratkaisu, jossa keskitetyllä tietokantapalvelimella olevasta tietokannasta haetaan tietoja työasemilla käytettävien sovellusten avulla. Kokonaisuutena potilastietojärjestelmä koostuu kymmenistä osasovelluksista, joiden välillä on eriasteisia teknisiä riippuvuuksia.

Osasovellusten ylläpidossa haasteita aiheuttaa esimerkiksi päivitystahti. Jotkin potilastietojärjestelmän osasovelluksista eivät yksinkertaisen luonteensa vuoksi päivity juuri koskaan, toiset laajat osasovellukset puolestaan muuttuvat lähes jokaisessa Monthly Fixissa. Kaikkien muutosten hallinta ja kokonaisuuden toiminnan varmistaminen vaatii paljon työtä suunnittelussa, toteutuksessa ja testauksessa. Osasovellusten välisten riippuvuuksien tietäminen on ratkaisevaa tilanteissa, joissa yhtä osasovellusta muutettaessa vaikutus saattaa näkyä ei-toivotulla tavalla toisessa osasovelluksessa.

2.8.2 Asiakkaat

Potilastietojärjestelmän asiakaskunta on todella laaja. Joukossa on suuria sairaaloita, joissa potilastietojärjestelmä on käytössä lukuisilla osastoilla ja työasemien määrä nousee useisiin tuhansiin. Toisaalta asiakkaina on myös esimerkiksi pieniä terveysasemia, joissa työasemia on vain muutamia.

Pitkän historian vuoksi potilastietojärjestelmästä on ollut tuotannossa jo useita versioita, joista vanhimmat ovat jo poistuneet ylläpidosta. Edelleen eri asiakkailla on kuitenkin käytössä useita eri järjestelmäversioita. Lisäksi eri asiakkaiden sovelluskokonaisuus vaihtelee paljon: joillakin asiakkailla on käytössään vain muutamia perussovelluksia ja toisilla puolestaan kaikki mahdolliset järjestelmään kuuluvat osasovellukset. Asiakkailla on luonnollisesti käytössään myös monia muita sovelluksia, joita varten on rakennettu lukuisia erilaisia liittymiä potilastietojärjestelmään.

2.9 Ohjelmistokehitysympäristö - VSTS

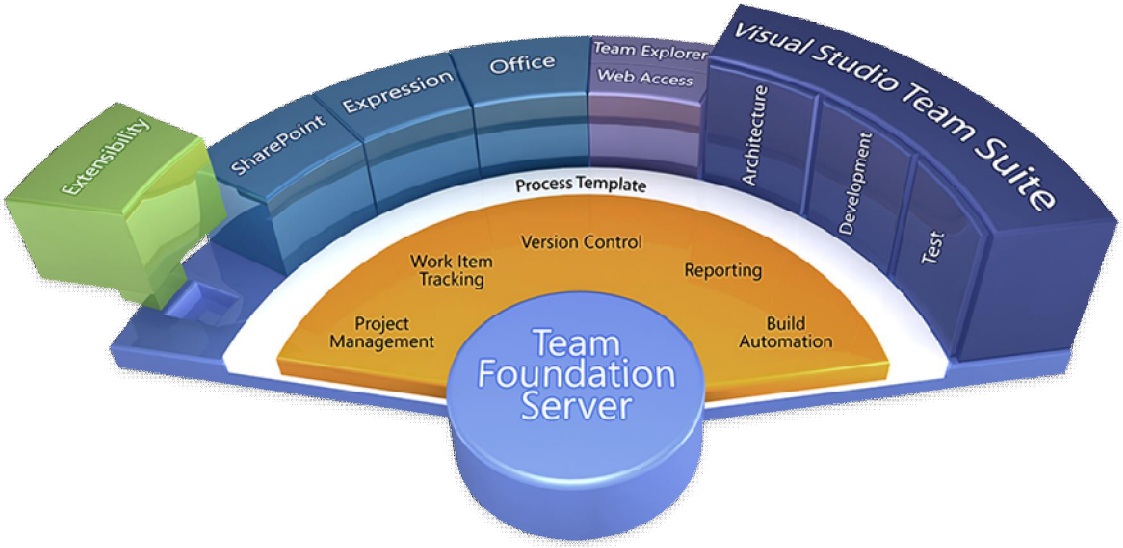
Potilastietojärjestelmän tuotekehitys- ja muutostenhallintaympäristönä käytetään Microsoft Visual Studio Team System 2008 Team Foundation Server -järjestelmää. Järjestelmä sisältää kaikki muutospyynnöt sekä osasovellusten lähdekoodit. Koko Monthly Fix -prosessin ohjaus ja seuranta tapahtuukin VSTS-järjestelmän avulla.

2.9.1 VSTS 2008

VSTS-järjestelmä on tarkoitettu kaikille, jotka toimivat osana ohjelmistojen kehitystiimiä: ohjelmistokehittäjät, testaajat, arkkitehdit ja projektipäälliköt. Se on kokonaisuutena ohjelmistokehitysympäristö, jonka avulla kehitysprosessi etenee sujuvasti. VSTS integroi kaikki kehitystiimin tarvitsemat työkalut yhteen yhteiseen ympäristöön, joka kokoaa kaikki projektin asiat yhteen paikkaan. Se tukee koko ohjelmistokehityksen elinkaarta vaatimusten analysoinnista, suunnittelun, ohjelmoinnin, ohjelmakoodin kääntämisen ja testauksen kautta ohjelmistojen julkaisemiseen. (Stott ja Newkirk 2007, 14).

VSTS-järjestelmästä on useita kaupallisia versioita, jotka poikkeavat hieman toisistaan. Kaikkien versioiden perusta on kuitenkin palvelimelle asennettavassa Team Foundation Server -ohjelmistossa (TFS) (Kuva 2.6). Palvelimella olevia tietoja tarkas-

tellaan ja muokataan työasemalle asennettavan Visual Studion Team Explorer -laajennuksen avulla. Myös internet-selaimen avulla päästään tarkastelemaan ja muokkaamaan TFS:lle tallennettuja tietoja, jos käytettävissä on palvelimelle asennettava Team System Web Access.



KUVA 2.6. VSTS 2008 -järjestelmän ominaisuudet (Pagels 2008).

Microsoftin oman määritelmän mukaan ”Microsoft Visual Studio Team System 2008 Team Foundation Server on kehitysorganisaation yhteistyöympäristö, joka keskittää tiimiportaalin, versionhallinnan, työkorttien seurannan, automaattisen käännös-, linkkaus- ja testiympäristön hallinnan, prosessien ohjauksen sekä liiketoimintatiedot samaan palvelimeen” (Microsoft Corporation 2008, 5).

2.9.2 Team Project

Team Projct on Team Foundation Serverillä oleva seuraavaksi suurin käsiteltävä kokonaisuus. Yhdellä Team Foundation Serverillä voi olla satoja Team Projecteja.

Team Project on kokoelma työkortteja, lähdekoodia, testejä, työtuotoksia, mittareita ja kaikkea muuta, jota kehitystiimi käyttää seuratakseen ja hallinnoidakseen yhteistä työkohdettaan. Loogisesti Team Project on yhtenäinen perusrakenne, joka kattaa kaikki erilliset välineet ja ohjelmiston kehityksen elinkaaren aikana käytetyt tekijät. Team Project onkin yksinkertaisesti säiliö, joka eristää sovelluksen kehityksessä käytetyt työkalut ja asiat, kuten lähdekoodin, työkortit ja dokumentit, erilleen siten, että ne eivät sekoitu muiden Team Projektien asioihin. (Microsoft Corporation 2010 .)

Team Project on keskeinen käsite, joka pitää yhteiseen päämäärään tähtäävää tiimiä koossa. Tiimin jäsenille Team Project luo selkeän ympäristön, jossa ei ole sekoittavia tekijöitä muista kehitysprojekteista. Koska Team Project kokoaa yhteen useita kehitystiimin työvälineitä, saadaan myös helpommin luotua raportteja projektin eri osa-alueiden välisistä suhteista eikä tarvita hankalia työkalujen välisiä konversioita. (Microsoft Corporation 2010.)

Team Project luodaan käyttämällä valmista Process Templatea, joka on eräänlainen pohjatietoja sisältävä malli Team Projektin sisältämistä asioista. Process Template määrittelee esimerkiksi Team Projectin käytettävissä olevat työkorttien tyypit, työkorttien sisältämät kentät ja Team Projectin valmiit raporttimallit. Oletusarvoisesti valittavissa on Agile tai CMMI -mallit, Microsoft Solutions Framework (MSF) for Agile Software Development tai MSF for Capability Maturity Model Integration (CMMI) Process Improvement. Lisämalleja esimerkiksi SCRUM-prosessimallin mukaisen Team Projectin perustamiseen voi ladata internetistä (Codeplex 2008). Lisäksi Process Templatea voi muokata myös itse haluamansa kaltaiseksi. (Microsoft Corporation 2010.)

2.9.3 Versionhallinta

Versionhallinta on perusvaatimus kaikille tiimimuotoisille ohjelmistokehitysympäristöille. Ohjelmistokehitystiimi tuottaa paljon kehityksen aikana paljon tietoa, josta vain pieni osa päätyy osaksi lopullista tuotetta. Usein uusia ratkaisuja löydetään vain yrityksen ja erehdyksen kautta ja se johtaa moniin versioihin lähdekoodien tiedostoista. Jokaisen ohjelmistokehitysprojektin, joka sisältää enemmän kuin muutaman tiedoston, joita vain yksi henkilö muuttaa, tulisi käyttää muutostenhallintaan soveltuvaa ohjelmistoa projektin lähdekoodien ylläpitoon. (Stott ja Newkirk 2007, 127 - 128.)

Versionhallintajärjestelmä tarjoaa kehitystiimille keskitetyn säilytyspaikan tiedostojen varastointiin ja niihin tehtyjen muutosten hallintaan. Tyypillisesti säilö sijaitsee palvelimella, jonne kaikilla kehitystiimin jäsenillä on pääsy. Versionhallintatyökalussa on toimintoja, joilla palvelimella olevat tiedostot haetaan muokattavaksi kehitystiimin jäsenten omille työasemille ja muokkauksen jälkeen palautetaan palvelimelle. Järjestelmä pitää automaattisesti kirjaa esimerkiksi palvelimella olevien muutosten sisällöstä, muutosten tekijästä ja muutosten ajankohdasta. Kaikki nämä tiedot ovat jokaisen kehitystiimin jäsenen näkyvillä reaaliaikaisesti. (Stott ja Newkirk 2007, 129 - 130.)

Yllä mainitut perusasiat pätevät myös VSTS versionhallintaan. Lisäksi VSTS järjestelmän versionhallinta on osa Team Projectia, jonka vuoksi versionhallinnan toimintojen lisäksi saadaan pelkkiä lähdekoodimuutoksia laajempi näkymä kokonaisuuteen, ja päästään hyödyntämään myös VSTS:n projektinhallinnan ominaisuuksia. Esimerkiksi muutoksen jäljitettävyys asiakkaan kirjaamalta muutospyynnöltä tuotekehittäjän tekemään lähdekoodimuutokseen on VSTS:n versionhallinnan perusominaisuus. (Levinson ja Nelson 2006, 59.)

2.9.4 Työkortit

Työkorttien (Work Items) voi sanoa olevan Team Foundation järjestelmän valuutta. Työkortti on Team Foundation Serverin tietokannassa oleva tallenne, jonka avulla työtehtävää pystytään jäljittämään ja seuraamaan sen tilaa. Työkorttityypit ovat määritelmiä tietyille työkorteille, jotka sisältävät esimerkiksi tietokenttiä, lomakkeita, tiloja ja tilasiirtoja. (Levinson ja Nelson 2006, 29.)

Työkorttivalikoima on määritelty Team Project -kohtaisesti, ja käytettävissä olevat työkortit valitaan Team Projectin luomisen yhteydessä kun valitaan Team Projectissa käytettävä Process Template. (Levinson ja Nelson 2006, 29.)

2.9.5 Kyselyt

Kyselyt (Query) ovat ennalta määriteltyjä näkymiä työkorttien tietokantaan. Jokaisessa Team Projectissa on Process Templaten mukaiset valmiit kyselyt, joilla käyttäjä löytää esimerkiksi omat työkorttinsa (All My Team Project Work Items), tai kaikki kyseisen Team Projectin työkortit (All Work Items). Kyselyjen toiminta perustuu työkorteilla olevien tietokenttien nimiin ja niiden sisältöön. Tietokenttiä käytetään parametreina kyselyitä tehdessä. (Levinson ja Nelson 2006, 125.)

Käyttäjä voi helposti luoda myös omia kyselyitä ja tallentaa niitä itselleen järjestelmään, jolloin kyselyt ovat ajettavissa samassa muodossa jatkossakin. Kyselyt voi tallentaa myös Team Queryina, siten että muut käyttäjät pääsevät käyttämään niitä (Levinson ja Nelson 2006, 127 - 128.)

2.9.6 Raportit

Team Foundation Server käyttää kaikkien tietojen tallennukseen Microsoft SQL Server 2008 tietokantaa. Tämä mahdollistaa, että raporttien luomiseen voidaan suoraan käyttää Microsoftin SQL Server Reporting Services (SSRS) ohjelmistoa, jonka käyttö on integroitu Visual Studion käyttöliittymään. (Levinson ja Nelson 2006, 162.)

Team Project sisältää myös tietyt vakioraportit esimerkiksi virheiden prioriteetin mukaan (Bugs by Priority) tai projektin jäljellä olevan työmäärän kuvaamiseen (Remaining Work). Sisältö vaihtelee käytetyn Process Templaten mukaan. (Levinson ja Nelson 2006, 162 - 164.)

2.10 Virheiden raportointijärjestelmä

Asiakkailla on käytettävissä sähköinen raportointijärjestelmä, jonka avulla ongelmat, kehitystoiveet ja muutospyynnöt välitetään ensin asiakaspalveluun ja sitä kautta tarvittaessa tuotekehitykseen saakka.

Asiakkaiden käyttämästä järjestelmästä kaikki tuotekehityksen toimenpiteitä vaativat asiat siirtyvät automaattisesti VSTS-järjestelmään, jossa niistä muodostuu uusia virheilmoituksia, kehitystoiveita tai selvityspyyntöjä, sen mukaan miten tärkeästä tai suuresta muutoksesta on kyse.

3 TUTKIMUKSEN TOTEUTTAMINEN

Tässä luvussa esitellään tutkimusmenetelmä ja tiedonkeruutavat sekä tutkimuksen tausta ja tavoitteet.

3.1 Tausta ja tavoitteet

Tutkimuksen kohdeorganisaatio on potilastietojärjestelmän kehitykseen ja ylläpitoon liittyvä organisaatio Tieto Healthcare & Welfare Oy:ssa. Tutkimus kohdistuu potilastietojärjestelmän tuotekehityksessä käytettävän Microsoft Visual Studio Team System 2008 -ohjelmiston tuottamien raporttien tarpeisiin organisaatiossa, erityisesti Monthly Fix -prosessin mukaisesti tehtyjen muutosten seurannassa. Tutkimuksessa selvitetään organisaation nykytilannetta VSTS:n käyttötapojen osalta sekä käytännön työssä esille tulleita ongelmia, joita raportoinnin kehittämisellä voitaisiin vähentää.

3.2 Tutkimusmenetelmä

Tutkimusmenetelmänä on kvalitatiivinen tapaustutkimus, jossa tarkastellaan yhtä tapausta. Kvalitatiivisen eli laadullisen tutkimuksen lähtökohtana on todellisen elämän kuvaaminen. (Hirsjärvi ym. 2005, 152.) Kvalitatiivisen tutkimuksen tyypillisinä piirteinä Hirsjärvi, Remes ja Sajavaara (2005, 155) listaavat esimerkiksi tutkimuksen luonteen kokonaisvaltaisuuden, ihmisten suosimisen tiedon keruun kohteena, tutkittavien näkökulmat esille tuovien laadullisten metodien käyttö sekä tarkoituksenmukaisen kohdejoukon valinnan tutkimuksen kohteeksi.

Tapaustutkimuksesta saadaan yksityiskohtaista tietoa yksittäisestä tapauksesta tai pienestä joukosta toisiinsa suhteessa olevista tapauksista. Tapaustutkimuksen tyypillisiä piirteitä katsotaan usein olevan prosessit ja yksittäistapausta tutkitaan yhteydessä ympäristöönsä, lisäksi aineistoa kerätään useita metodeja käyttämällä. (Hirsjärvi ym. 2005, 125 - 126.)

Kvalitatiivisessa tutkimuksessa tavoitellaan tutkimuskohteen ymmärtämistä, usein tutkimus aloitetaan kartoittamalla kenttää, jossa tutkija toimii (Hirsjärvi ym. 2005, 170).

Edellä mainittujen seikkojen perusteella kvalitatiivinen tapaustutkimus sopii hyvin tässä tutkimuksessa käytettäväksi menetelmäksi, koska tutkimuksella ei tavoitella yleistettävyyttä tai laajaa kattavuutta. Tutkimuksen tavoitteena on enemmänkin selvittää tietyissä rooleissa toimivien ihmisten näkökulmia ja heidän päivittäisessä työssään esille tulleita asioita sekä saada selville onko jossain ohjelmiston ylläpitoprosessin vaiheissa erityisiä ongelmia joita voitaisiin helpottaa raportointia kehittämällä.

Kuten tapaustutkimuksessa yleensä, tässäkin käytettiin useita tiedonkeruutapoja, keskeisenä tiedonkeruutapana olivat haastattelut. Haastatteluksi mielletään tutkimuksen tiedonkeruutapaa, jossa henkilöltä kysytään mielipiteitä tutkimuksen kohteesta ja vastaus saadaan puhutussa muodossa. Haastattelu on siis ennalta suunniteltua päämäärähakuista toimintaa joka tähtää informaation keräämiseen. (Hirsjärvi ja Hurme 2000, 41 - 42.)

Haastattelu on tiedonkeruumenetelmänä ainutlaatuinen, koska siinä ollaan suorassa kielellisessä vaikutuksessa tutkittavaan kohteeseen. Tästä seikasta aiheutuu etuja, että haittoja. Suurimpana etuna pidetään yleensä aineiston keräämisen joustavuutta. (Hirsjärvi ym. 2005, 193.)

Tutkimuksen tiedonkeruumenetelmien valinnan tulee olla perusteltua, joten haastatteluakaan ei tule valita menetelmäksi pohtimatta sen soveltuvuutta kyseisen ongelman ratkaisemiseen. Haastattelun etuna muihin tiedonkeruumuotoihin verrattuna on se, että siinä voidaan kerätä aineistoa joustavasti tilanteen ja haastateltavien edellyttämällä tavalla. Haastatteluaiheiden järjestystä voi säädellä ja vastausten tulkintaan on enemmän mahdollisuuksia kuin esimerkiksi lomakekyselyssä. Etuna on myös, että vastaajiksi suunnitellut henkilöt saadaan yleensä mukaan tutkimukseen ja heidän vastauksiaan on helppo täydentää tai tarkentaa myöhemminkin. (Hirsjärvi ym. 2005, 194 - 195.)

Monet haastattelun hyvät puolet sisältävät myös ongelmia. Haastattelut vievät aikaa, niiden tekeminen vaatii huolellista etukäteissuunnittelua ja vastausten purkaminen on usein hidasta. Haastatteluun katsotaan sisältyvän myös useita virhelähteitä, joista osa aiheutuu haastattelijasta, osa haastateltavasta ja osa myös haastattelutilanteesta kokonaisuutena. Haastattelun luotettavuutta voi myös heikentää se, että haastattelussa ihmisten on taipumus vastata sosiaalisesti suotuisalla tavalla, eikä kaikkia epäkohtia välttämättä tuoda esille. Lisäksi haastateltava saattaa antaa tietoja jostain tietyistä aiheista vaikka haastattelija ei kysyisikään. Haastatteluaineisto on myös kon-

teksti- ja tilannesidonnaista, eli haastateltavat saattavat puhua haastattelutilanteessa toisin kuin jossain toisessa tilanteessa. Tulosten tulkinnassa tulisikin ottaa huomioon nämä seikat ja tulosten yleistämistä ei pitäisi liioitella. (Hirsjärvi ym. 2005, 195 - 196.)

Haastattelujen lisäksi tietoa kerättiin tutkimalla VSTS-järjestelmään kirjattuja työkortteja sekä järjestelmään kuuluvia vakioraportteja.

3.3 Tiedonkeruutavat

Tiedonkeruu suoritettiin seuraavilla tavoilla:

1. Tehtiin tutkimushaastatteluja kohdeorganisaatiossa.
2. Tutkittiin nykyiseen VSTS-järjestelmään kirjattuja työkortteja ja niiden rakennetta.
3. Tutustuttiin VSTS-järjestelmän vakioraportteihin.

Olennaisimpana tiedonkeruutapana olivat haastattelut, joita tehtiin yhteensä 10. Haastattelut toteutettiin puolistrukturoituina teemahaastatteluina, joissa haastateltavat saivat ennalta tutustua haastattelurunkoon. Tyypillisesti puolistrukturoidussa teemahaastattelussa käytetään haastattelurunkoa tai teemaluetteloa, joilla varmistetaan, että haastattelun aikainen keskustelu kohdentuu oikeisiin ja tutkittavien kohteiden kannalta olennaisiin asioihin. (Hirsjärvi ja Hurme 2000, 103.)

Tässä tutkimuksessa puolistrukturoitu tutkimushaastattelu oli luontevin valinta tiedonkeruutavaksi. Strukturoiduista kysymyslomakkeista olisi ollut hankalaa saada riittävän kattavia ilman, että kysymysten määrä olisi kasvanut huomattavan suureksi, mistä olisi todennäköisesti seurannut alhainen vastausprosentti. Myöskään nyt haastatteluun valittua laajemmalle otannalle ei nähty tarvetta, vaan kaikki olennaisimmat roolit saatiin mukaan kyselyyn. Haastattelujen oli tarkoitus kattaa laadittavien raporttien kannalta olennaisimmat roolit organisaatiossa. Etukäteen oli selvää, että suurimmat tarpeet raporttien käyttöön on niin sanotulla päällikkötasolla, henkilöillä, joiden työnkuvaan kuuluu kokonaisuuksien hallintaa, resursointia ja asiakaskommunikointia. Tuotekehityksessä haastateltaviksi valittiin tiimipäälliköitä, tuotepäälliköitä sekä kehitysprosessissa muutospyyntöjä koordinoivia henkilöitä. Asiakaspalvelun organisaatiosta haastateltiin aluepalvelupäälliköitä sekä asiakasryhmäpäälliköitä. Haastateltaviksi valittiin jokaisesta roolista kaksi henkilöä. Kvalitatiivisen aineiston keruussa aineiston riittävyyttä kuvataan saturaation käsitteellä (Hirsjärvi ym. 2005, 171). Tässä

tutkimuksessa oletettiin lähtökohtaisesti, että useampien haastateltavien valinta jokaisesta roolista ei olisi tuonut lisäarvoa tutkimukseen.

Kaikkien haastattelujen tavoitteena oli selvittää, mitä tietoja haastateltavat ensisijaisesti haluaisivat raportteina saada nähtäväksi. Lisäksi haastateltavilta pyrittiin saamaan tietoa, mitkä tarpeelliset asiat ovat vaikeimmin löydettävissä nykyisestä järjestelmästä, jotta VSTS-järjestelmän kehityksen painopistettä pystyttäisiin jatkossa ohjaamaan oikeaan suuntaan.

Haastatteluaineiston laadukkuutta haluttiin parantaa laatimalla ennen haastatteluja haastattelurunko (Hirsjärvi ja Hurme 2000, 184). Haastattelurungon kysymyksiä mietittiin Monthly Fix -prosessin tarpeiden pohjalta ja kysymyksistä laadittiin melko yksityiskohtaisia, koska haastatteluista haluttiin tietoa tarkasti rajatusta asiasta. Keskeisintä haastatteluilla oli selvittää millaisia asioita eri rooleissa toimivat henkilöt haluaisivat saada raporttien muodossa VSTS-järjestelmästä ja millä tavalla he haluaisivat seurata muutospyyntöjen tilanteita. Haastattelussa pyrittiin lisäksi selvittämään, onko jossain tietyssä Monthly Fix -prosessin vaiheessa erityinen tarve jollekin määrätyntyyppiselle raportille.

Haastateltavat saivat haastattelurungon tutustuttavakseen noin kaksi työpäivää ennen haastattelua. Haastattelujen alussa haastateltaville kerrottiin tutkimuksen tarkoituksesta ja haastattelun kohderyhmä. Haastattelussa seurattiin haastattelurungon kysymyksiä, mutta haastateltavia kehotettiin miettimään asioita myös laajemmin Monthly Fix -prosessin kannalta.

Kaikki haastattelut käytiin puhelinneuvotteluina käyttämällä Microsoft Live Meeting -ohjelmistoa, jonka avulla haastattelun aikana jaettiin yhteisen työpöytänäkymän kautta ennalta lähetetyt kysymykset sekä tarkasteltiin työkorttien tietokenttiä ja niiden tietoja. Haastattelut tallennettiin teemahaastattelun tyypillisen luonteen mukaisesti. Tällä tavalla haastattelu sujuu nopeammin ja ilman taukoja vastausten kirjoittamisen vuoksi. Lisäksi nauhoitteista saadaan jälkeenpäin purettaessa analysoitua tärkeitä haastatteluun sisältyneitä vivahteita (Hirsjärvi ja Hurme 2000, 92). Haastattelujen aluksi haastateltavilta kysyttiin lupaa haastattelujen tallentamiseen. Tallentamiseen käytettiin Microsoft Live Meeting -ohjelmistoa, jonka avulla nauhoitettiin keskustelun lisäksi myös jaettu työpöytänäkymä.

Haastattelut kestivät noin 30 minuutista noin 60 minuuttiin. Kaikki haastattelut toteutettiin kahden peräkkäisen työviikon aikana. Haastattelunauhoitusten litterointi tehtiin haastatteluja seuraavan kahden viikon aikana. Nopeasti tehty litterointi parantaa myös haastattelujen laatua, erityisesti kun tutkija haastattelee ja litteroi itse (Hirsjärvi ja Hurme 2000, 185).

Toisena keskeisenä tiedonkeruutapana käytettiin VSTS-järjestelmän työkortteja, jotka on muodostettu virheilmoituksista ja kehitystoiveista. Työkorteilta tutkittiin käytettyjä tietokenttiä ja selvitettiin järjestelmän sisäisten hakutoimintojen perusteella olennaisimpien tietokenttien käytettävyyttä raporttien muodostamiseen. Lisäksi tutkittiin työkorttien historiatietoja, joista selvitettiin niiden tilojen muutoksia ja sitä, kuinka muutospyynnöt olivat edenneet organisaatiossa.

Tutkimuksessa tutustuttiin lisäksi VSTS-järjestelmän tuottamiin vakioraportteihin. Järjestelmän asennuksen yhteydessä käyttäjille muodostuu raporttipohjia, joilla voi suoraan ajaa raportteja vakiomuotoisilta työkorteilta.

4 TUTKIMUKSEN TULOKSET

Tämä luku käsittelee tapaustutkimuksen tuloksia. Aluksi esitellään tutkimushaastattelujen tulokset, minkä jälkeen kerrotaan VSTS-järjestelmän työkorteista tehtyjä havaintoja sekä esitellään VSTS-vakioraporttien soveltuvuutta Monthly Fix -prosessin käyttöön.

4.1 Haastattelujen tulokset

Tutkimuksen keskeisin tiedonkeruu tehtiin haastatteluilla. Haastatteluissa tuli selvästi esille, että VSTS-järjestelmään oltiin yleisesti varsin tyytyväisiä ja järjestelmän käytettävyys ja sopivuus koettiin organisaation tarpeisiin hyväksi.

Haastateltavat käyttivät pääsääntöisesti muutamia tunteja viikossa Monthly Fix -muutospyyntöjen käsittelyyn VSTS:ssä. Poikkeuksena olivat Monthly Fix -prosessia koordinoivat henkilöt, joiden työajasta muutospyyntöjen käsittely vei selvästi suuremman osan, enimmillään jopa lähes puolet kokonaistyöajasta. Myös käsiteltävien muutospyyntöjen määrä vaihteli käytetyn työajan mukaan; yleisesti haastateltavat käsittelivät omien arvioidensa mukaan keskimäärin noin kymmentä muutospyyntöä viikossa. Monthly Fixia koordinoivat henkilöt puolestaan arvioivat käsittelevänsä VSTS:n avulla noin sataa muutospyyntöä viikossa. Suuri ero roolien välillä oli tiedossa jo ennen haastatteluja, koska työtehtävät poikkeavat toisistaan merkittävästi. Tämän vuoksi koordinaattorien raporttitarpeet eroavat muiden haastateltujen esittämistä tarpeista jonkin verran.

Lähes kaikissa haastatteluissa tuotiin esille VSTS-järjestelmän vahvuutena, että käytössä on järjestelmä, jossa kaikki tieto muutoksista on kerättynä yhteen, varmennettuun paikkaan ja josta tietojen hakeminen on mahdollista myös pidemmältä aikaväliltä.

Kaikki haastateltavat olivat yhtä mieltä siitä, että automaattisesti toimitettaville raporteille ei ole tarvetta. Erityisen huonona vaihtoehtona koettiin sähköpostiin toimitettavat raportit tai ilmoitukset, koska ylimääräistä sähköpostia tulee muutenkin liikaa ja raportit hukkuisivat muun sähköpostin sekaan. Sen sijaan haastateltavien mielestä raporttien tulisi olla ajettavissa itsenäisesti silloin, kun niille on tarvetta. Raporteille johtavien linkkien sijainnille ei esitetty erityisiä vaatimuksia. Useimmat haastateltavat olivat

valmiit ajamaan raportit suoraan VSTS-järjestelmästä, muutamien mielestä raportit voisivat avautua myös tiimien dokumenttienhallintajärjestelmän web-sivustoilta.

Tuotekehityksen henkilöiden mielenkiinto kohdistui haastattelujen perusteella ensisijaisesti tuotekohtaiseen raportointiin. Kaikki tuotekehityksen haastateltavat toivat esille tuotekohtaisen raportoinnin tärkeyden, mutta myös sen, että nykyinen järjestelmien tuoterakenne ei ole kaikissa tapauksissa täysin paikkaansa pitävä. Tuotteiden välillä on rajatapauksia, jotka vääristävät ja hankaloittavat tilannetta. Tuoterakennetta olisi kuitenkin kaikkien mielestä lähes mahdotonta saada täysin aukottomaksi johtuen ohjelmistokokonaisuuden monimutkaisen luonteen ja sovellusten välisten riippuvuuksien vuoksi. Tuotteen tilannetta halutaan haastattelujen perusteella seurata monella eri tavalla. Tuotteen kokonaistilanne avoimien ja toteutettujen muutospyyntöjen osalta oli kiinnostavin asia, lisäksi tuotteen kokonaistilanteen muutoksia kuvaava trendi kiinnosti useimpia. Kokonaistilanteen lisäksi haluttiin erikseen seurata kriittisiksi luokiteltujen virheiden tilannetta tuotteissa. Myös muutospyyntöjen toteutukseen käytettyä aikaa haluttiin seurata raporteilla. Käytetty aika pitäisi pystyä myös luokittelemaan tuotteittain ja virheluokkien mukaan.

Yhtenä tärkeimmistä asioista tuotekehityksen henkilöiden haastatteluissa tuli esille testauksista takaisin tuotekehittäjälle palautuneiden virheiden seuraaminen. Palautuneet virheet miellettiin liian työllistäviksi ja prosessia rikkoviksi ja niistä haluttaisiin päästä eroon. Seuraamalla palautuneita virheitä voitaisiin jatkossa arvioida tietyntyyppisten korjausten työmäärää ja resursointia tarkemmin. Jokainen palautunut virhe keskeyttää yleensä jonkin toisen muutoksen toteuttamisen ja vaati uuden korjauksen lisäksi myös uuden testauksen. Lisäksi tuotekehityksessä haluttaisiin helposti nähdä, mistä syystä palautuneet virheet aiheutuivat; johtuiko virheen palautuminen väärästä tai epätäydellisestä korjauksen määrittelystä, jostain toisesta virhekorjauksesta, vai oliko kyseessä enää ollenkaan sama virhe, jota alkuperäisessä korjauksessa oli korjattu.

Tuotekehityksen tiimipäälliköiden mielestä tärkeimpiä seurattavia kohteita olivat luonnollisesti oman tuotekehitystiimin asiat. Nykytilanteessa koettiin hankalaksi oman tiimin henkilöiden kokonaiskuormituksen ja tekeillä olevien tehtävien selvittäminen. Tähän oli syynä se, että samat henkilöt tekevät korjauksia Monthly Fix -prosessin puitteissa mutta myös kehitystehtäviä tuotekehitysprojekteissa. Lisäksi tiimipäälliköt halusivat seurata Monthly Fixin muutospyyntöjen edistymistä tiimin tasolla sekä omalle tiimille kuuluvien tuotteiden virheiden määrän kehittymistä pidemmällä aikavälillä.

Tuotekehityksessä tuotepäälliköinä toimivilla henkilöillä oli haastatelluista kaikkein läheisin näkökulma tuotteisiin ohjelmistojen lähdekoodin tasolla. Heidän tiedonsaantitarpeensa kohdistui kaikista lähimmäksi yksittäistä muutospyyntöä ja sen vaikutusta ohjelmistoon teknisessä mielessä. Tuotepäälliköiden näkökulmasta oman tuotteen kokonaistilanne oli kiinnostavin ja sitä pitäisi pystyä seuraamaan koko tuotteen elinkaaren näkökulmasta, mutta myös Monthly Fix -kierroksen muutosten tasolla. Lisäksi esille tuli tarve seurata omaan tuotteeseen läheisesti liittyvien muiden tuotteiden vaikutusta. Koska useiden tuotteiden välillä on riippuvuuksia lähdekooditasolla, tuotepäälliköt haluaisivat seurata raporttien avulla, mikäli heidän oman tuotteensa lähdekoodeihin tulisi muutoksia jonkin toisen tuotteen kautta.

Sekä asiakaspalvelun, että tuotekehityksen haastattelussa tuli esille tarve jonkinlaiselle ilmoitukselle tai raportille pitkään järjestelmässä avoimena olleista muutospyynnöistä. Useassa haastattelussa tuli esille huoli, että muutospyyntöjen määrän ollessa jo melko suuri, jää järjestelmään kirjattuja kiireettömiä tapauksia roikkumaan toisinaan pitkäksikin aikaa. Monet tapauksista ovat sellaisia jotka ovat investigation-tilassa. Niihin on pyydetty tarkempaa selvitystä esimerkiksi tuotteen asiantuntijalta tai tuotevastaavalta, mutta kiireettöminä tapauksina ne ovat jääneet henkilöiden työjonoon selvitettäviksi jopa useiksi kuukausiksi.

Useissa haastatteluissa esitettiin myös tarpeita raporteille joista nähtäisiin tuotekohtaisia sekä pääversiokohtaisia trendejä asiakkailta ilmoitettujen ja omissa testeissä löydettyjen virheiden suhteesta korjattujen virheiden määrään useamman viikon ajalta. Erityisesti Monthly Fixin edistymistä seuraavat koordinaattorit toivat esille tarpeen seurata koko sovelluskokonaisuuden virhekorjausten lukumäärän kehittymistä verrattuna testeissä löydettyihin virheiden määrään pidemmällä aikavälillä.

Asiakaspalvelun organisaatioon kuuluvien henkilöiden haastatteluista tärkeimmäksi kehityskohteeksi nousi selkeästi asiakaskohtaisten muutospyyntöjen raportoinnin tarve. Erityisesti haluttiin seurata asiakaskohtaisesti kuinka tietyn asiakkaan ilmoittamat muutospyynnöt edistyivät, koska jokainen asiakas on kiinnostunut ensisijaisesti itse raportoimistaan virheistä ja kehitystoiveista.

Asiakaspalvelussa kaivattiin myös mahdollisuutta seurata tietylle asiakkaalle luvattujen muutosten edistymistä. Asiakkaiden ohjelmistopäivitykset on usein sovittu tehtäväksi tietyllä viikolla tai jopa tiettynä päivänä jo useita viikkoja aikaisemmin. Niissä on

usein luvattu toimittaa tiettyjä asiakkaan vaatimia muutoksia, joita ilman koko asennettava päivitys voi olla asiakkaan näkökulmasta lähes hyödytön. Sen vuoksi olisi tärkeää pystyä seuraamaan kuinka tietyn asiakkaan muutokset etenevät tuotekehityksessä, ja ehditäänkö kaikki asiakkaan vaatimukset toteuttaa, testata ja hyväksyä julkaistavaksi hyvissä ajoin ennen asiakkaan kanssa sovittua päivitystä. Asiakkailta tulevien muutospyyntöjen käsittelyssä koettiin lisäksi hankalaksi tilanteet, joissa useampi kuin yksi asiakas oli ilmoittanut samasta asiasta tai vastaavasti yksi muutospyyntö pitäisi pystyä tarkastelemaan useamman asiakkaan näkökulmasta. Työkorteille merkitään asiakastietoon vain se asiakas joka on ensimmäisenä asiasta raportoinut, vaikka sama muutos kohdistuu usealle muullekin asiakkaalle.

Asiakaspalvelun asiantuntijat osallistuvat Monthly Fix -prosessissa testaukseen ja asiakaspalvelun tiimipäälliköt haluaisivat sen vuoksi seurata oman tiiminsä osalta testausten edistymistä ja oman tiimin henkilöiden työtilannetta.

Erityisesti asiakaspalvelun haastateltavat kaipasivat myös raporttia, jossa näkyisi koko asiakkaalle toimitettavan sovelluskokonaisuuden kaikkien osasovellusten tarkat versiotiedot tietyn Monthly Fix -julkaisun aikaan, niin sanottua konfiguraatiolistaa. Lisäksi olisi tarvetta myös historiatiedolle, josta helposti nähtäisiin mitä versioita kustakin sovelluksesta asiakkaalla on ollut tietyn toimituksen aikaan.

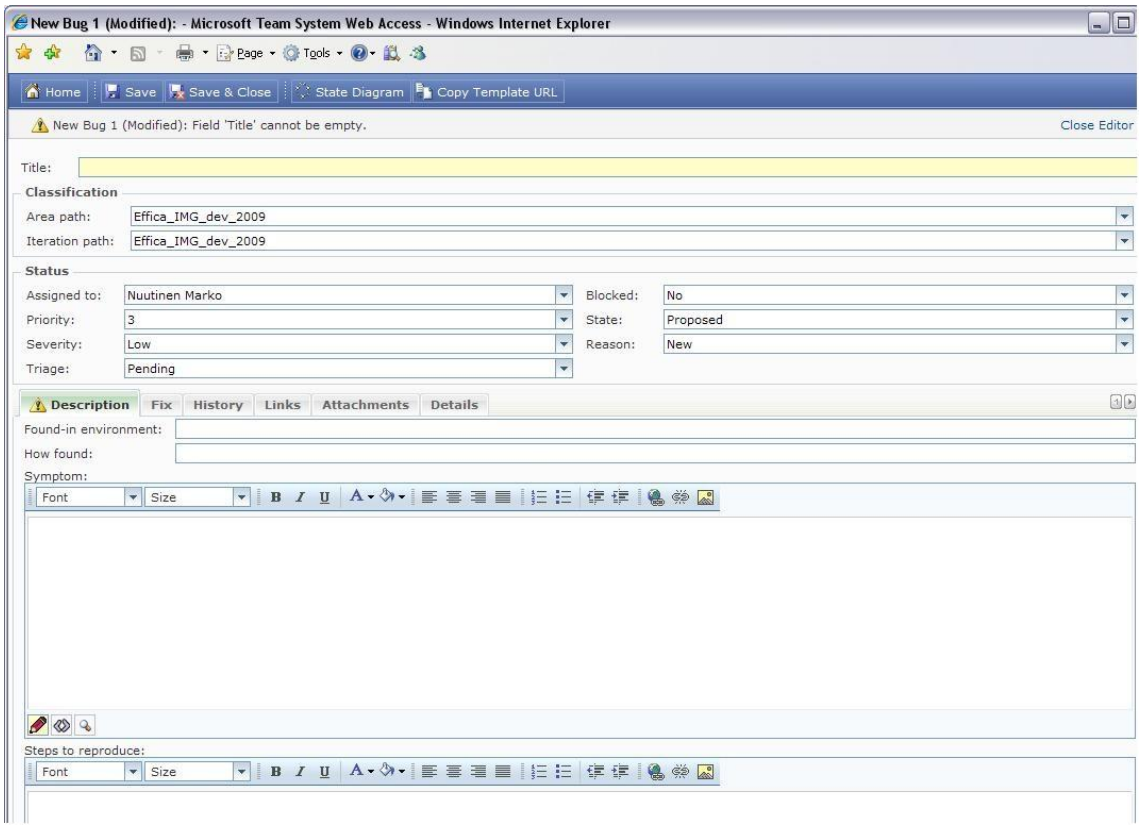
Haastattelujen perusteella VSTS-järjestelmän käyttö hallitaan erityisesti tuotekehityksessä varsin hyvin. Asiakaspalvelulle VSTS ei ole pääasiallinen työkalu ja monimutkaisempien hakukyselyjen tekemiseen kaivattiin lisäkoulutusta. Lisäksi koettiin, että valmiit raporttipohjat olisivat monessa tilanteessa helpompi ja nopeampi tapa tarkastella asioita, kuin erillisen kyselyn tekeminen, vaikka samat asiat osattaisiinkin hakea järjestelmästä kyselyn avulla.

Noin puolet kaikista haastateltavista totesi, että VSTS-järjestelmä on nykymallissaan melko hyvin toimiva, mutta siihen luotetaan joissain tapauksissa jo liikaa. Haastateltavat täsmensivät, että joissain tilanteissa luotetaan liian sokeasti ja kyseenalaistamatta järjestelmään kirjattuihin asioihin, jotka saattavat olla kuitenkin epätäydellisiä tai jopa virheellisiä. Järjestelmän rinnalle kaivattiin muutakin kommunikointia, kuin työkorttien lähettelyä henkilöiden välillä, joka miellettiin hitaaksi, kankeaksi ja epäinformatiiviseksi tavaksi verrattuna esimerkiksi puhelinsoittoon.

4.2 Työkortit

Tutkimuksen toisena tiedonkeruutapana käytettiin VSTS-järjestelmän sisältämien työkorttien tietojen selvittämistä. VSTS-järjestelmään on alettu kirjata muutospyyntöjä vuoden 2008 syksyllä. Alusta lähtien järjestelmään kirjatut muutospyynnöt ovat tulleet osittain asiakkaiden käyttämästä raportointijärjestelmästä ja osittain organisaation omien testien tuloksena. Aluksi siirrot asiakkaiden käyttämästä virheiden raportointijärjestelmästä tapahtuivat osittain manuaalisesti, mutta vuoden 2009 aikana siirrot järjestelmien välillä on kokonaan automatisoitu. Keväällä 2010 VSTS-järjestelmään kirjattiin useita kymmeniä muutospyyntöjä viikossa, kun otettaan huomioon kaikki uudet työkortit, ei pelkästään Monthly Fix -prosessin käsiteltäväksi meneviä. Muutospyynnöistä neljä viidesosaa kirjattiin organisaation omien testausten perusteella ja yksi viidesosa syntyi asiakkaiden kirjausten perusteella.

Kuten muutkin muutospyynnöt, myös Monthly Fix -prosessin aikana käsiteltävät ovat osittain peräisin organisaation omista testeistä ja osittain asiakkaiden kirjaamista virheilmoituksista ja kehitystoiveista. Käytetyt työkortit ovat Bug-tyyppisiä, mutta niiden sisältämiä tietokenttiä on muokattu ja uusia tietokenttiä on lisätty todella paljon verrattuna VSTS-järjestelmän vakiomuotoiseen Bug-työkorttiin (Kuva 4.1). Muokatusta työkortista käytetään järjestelmässä nimeä Main Bug. Main Bugeilla käsitellään järjestelmässä olevat virheet; muiden muutospyyntöjen, kuten kehitystoiveiden käsittelyyn käytetään erityyppisiä työkortteja. Kehitystoiveista luodaan Product Backlog Item -tyyppisiä työkortteja ja selvityspyynnöistä Task-tyyppisiä työkortteja. Raportoinnin kannalta olennaisimpia on kuitenkin Main Bug työkortit, joita järjestelmään muodostettavista työkorteista on selkeästi suurin osa.



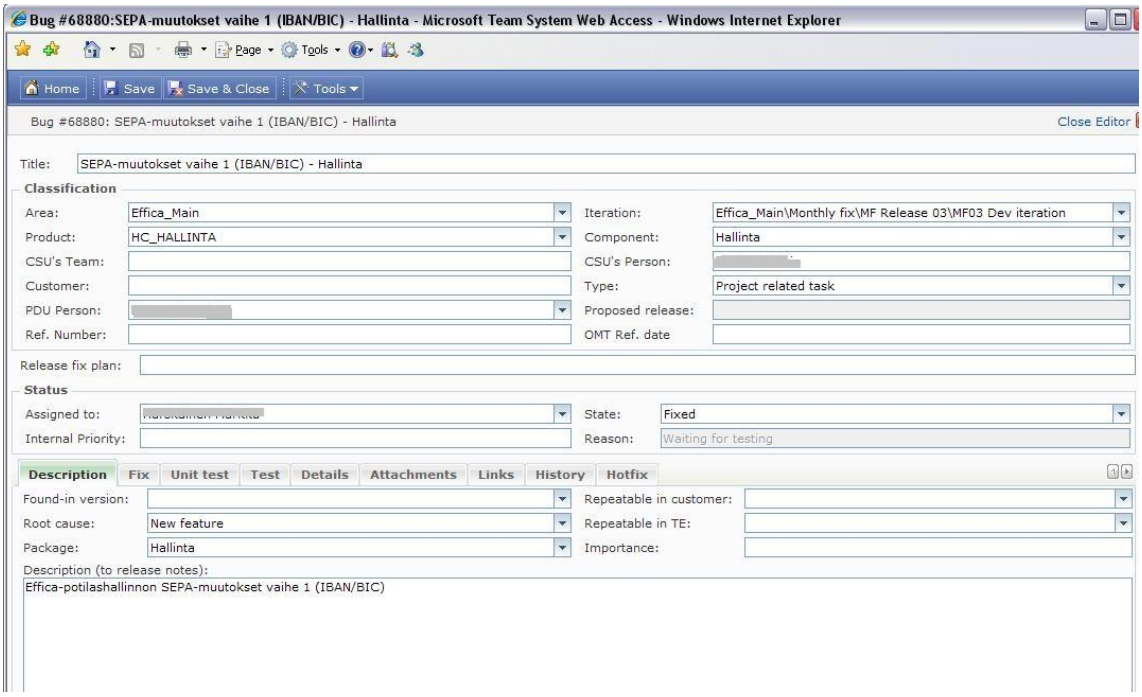
KUVA 4.1. Vakiomuotoinen Bug-tyypinen työkortti.

Main Bugit sisältävät kaiken tiedon yksittäisestä muutospyynnöstä. Jos Main Bug on luotu organisaation omissa testeissä löydetyn virheen perusteella, sen tietokenttien sisältämät tiedot kirjaa virheen löytänyt testaaja. Mikäli muutospyyntö tulee asiakkaalta, joka on kirjannut sen virheiden raportointijärjestelmään, luodaan Main Bug automaattisesti, kun muutospyyntö siirretään asiakkaan käyttämästä järjestelmästä VSTS-järjestelmään. Tällöin Main Bugin tietokenttiin siirtyy suoraan asiakkaan kirjaamat tiedot.

Main Bugin rakenteen voi jakaa kahteen osaan. Työkortin yläosassa ovat esimerkiksi perustiedot tuotteesta, asiakkaasta, asiaa käsittelevistä henkilöistä, muutospyynnön tila sekä prosessin kannalta olennaiset Area- ja Iteraatio-tiedot. Area-kentällä muutospyynnöt luokitellaan tuotteittain ja Iteraatio-kentällä määritellään, missä Monthly Fix -julkaisussa kyseinen muutos toteutetaan, lisäksi määritellään Monthly Fix – Iteraatio (Kuva 4.2).

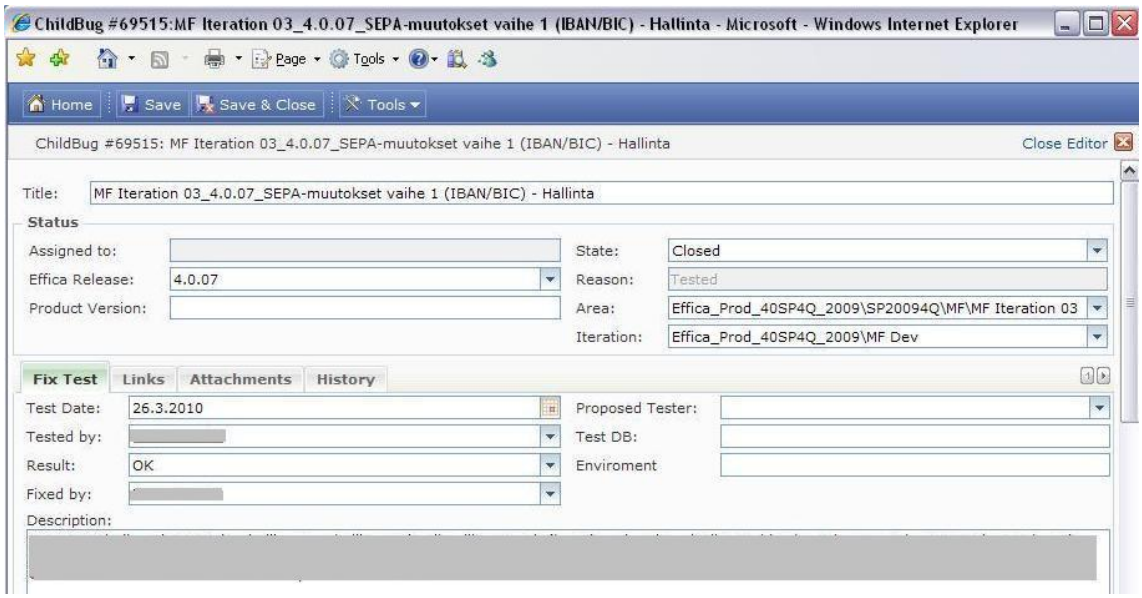
Työkortin alaosa on jaettu useaan välilehteen, joista jokainen sisältää esimerkiksi tarkempia tietoja muutospyynnön sisällöstä, kuvauksen tehdystä muutoksesta, ohjeet

muutoksen testauksesta sekä linkkejä muille työkorteille ja mahdollisia liitteitä kuvankaappauksista tai aiheeseen liittyviä dokumentteja.



KUVA 4.2. Potilastietojärjetelmän ylläpitoon muokattu Bug-tyyppinen työkortti.

Main Bug työkorttien lisäksi käytössä on Child Bug –työkortit (Kuva 4.3). Child Bugit ovat eräänlaisia versiokohtaisia muistilappuja, joilla varmistetaan, että jokainen Main Bugilla kuvattu muutospyyntö toteutetaan ja testataan kaikissa potilastietojärjestelmän versioissa, joihin se on päätetty tehdä. Child Bug näkyy linkkinä Main Bugilla, ja se sisältää huomattavasti vähemmän tietoa kuin Main Bug. Olennaisimmat tiedot Child Bugilla kertovat tarkan version muutospyynnölle, Monthly Fix -iteraation ja sen, missä tilassa kyseinen muutospyyntö juuri siinä versiossa on. Mahdollisia Child Bugin tiloja ovat Active, Fixed, Investigation, Closed ja Deleted. Mikäli huomataan, että kyseistä muutosta ei voida tehdä tiettyyn versioon tai muusta syystä muutos kyseiseen versioon jätetään toteuttamatta, Child Bug suljetaan kokonaan tai siirretään käsiteltäväksi myöhemmin. Prosessin näkökulmasta Main Bug on osoitettuna tilanteen mukaan tuotevastaavalle, tuotepäällikölle tai Monthly Fix -prosessin koordinaattorille. Child Bug puolestaan on osoitettuna tuotekehittäjälle tai testaajalle, sen mukaan missä vaiheessa elinkaartaan Child Bug on.

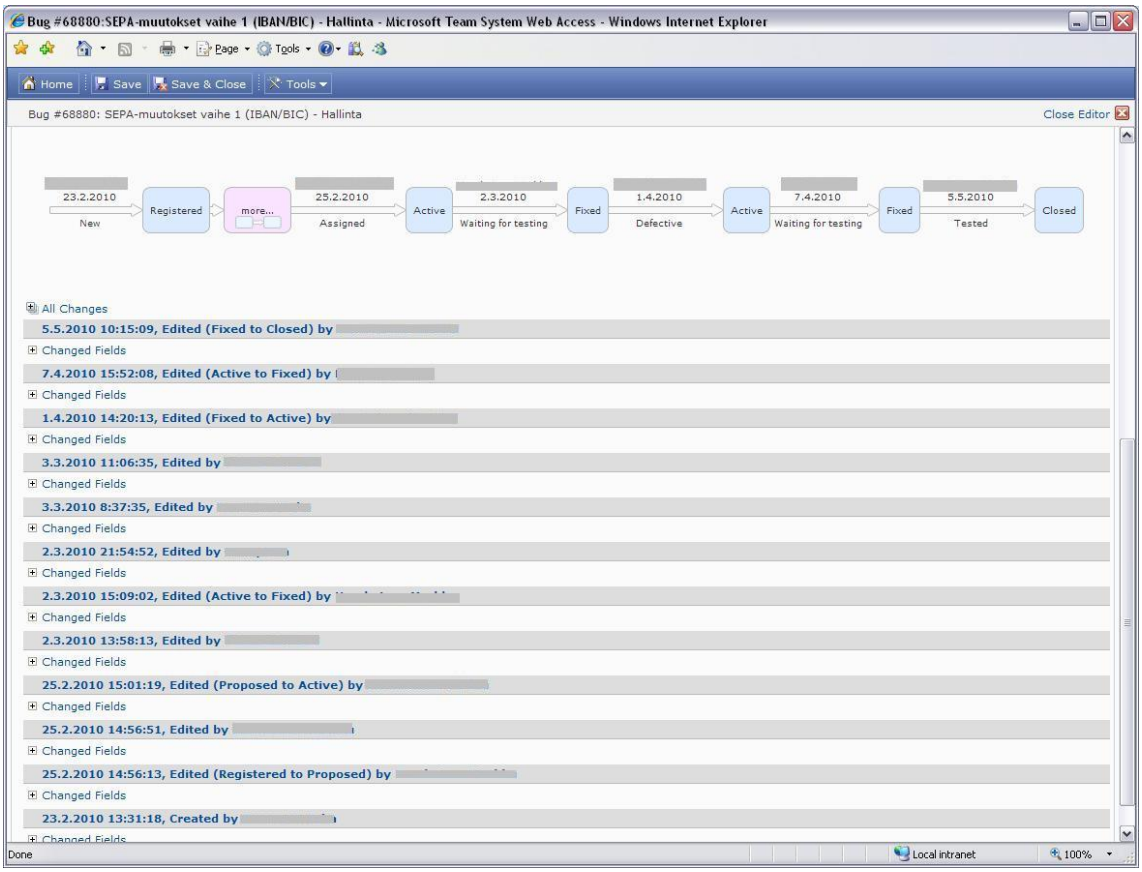


KUVA 4.3. Versiokohtainen muistilappu, Child Bug.

Child Bugeilla on tieto Monthly Fix -julkaisusta, jolla kyseinen muutos toteutetaan. Yhdistämällä kaikkien yksittäisten Monthly Fix -julkaisun Child Bugien tilanteet, voidaan seurata Monthly Fix -julkaisun kokonaistilannetta.

Sekä Main Bugin, että Child Bugin historiatieto-välilehdelle (Kuva 4.4) tallentuu tiedot kaikista muutoksista, joita työkorteille tehdään. Historiasta näkee muutoksen tekijän, tarkan päivämäärätiedon ja kellonajan sekä muutettujen tietokenttien sisällöt ennen ja jälkeen muutoksen. Työkorttien tarkkaan historiatietoon voidaan kohdistaa hakuja ja raportteja. Historiasta voidaan hakea tietoja esimerkiksi rajaamalla, mitkä työkortit ovat käyneet jossain vaiheessa suljettuina, tai työkortteja joiden tilaa ei ole muutettu kuukauteen.

Vakiomuotoiseen työkorttiin verrattuna muokatuilla työkorteilla on paljon tietokenttiä, joihin hakuja voi kohdistaa. Osa lisätyistä tietokentistä on ongelmallisia, koska nimet, joilla uudet tietokentät näkyvät työkortilla, eivät aina näy vastaavalla tavalla kyselyn tekemiseen käytettävässä ohjelmakomponentissa. Tietokenttien nimien yhdenmukaistaminen vastaamaan työkortilla näkyvää tietokentän nimeä helpottaisi hakujen tekemistä ja olisi sen vuoksi järkevää toteuttaa.



KUVA 4.4. Main Bugin historiatietokenttä.

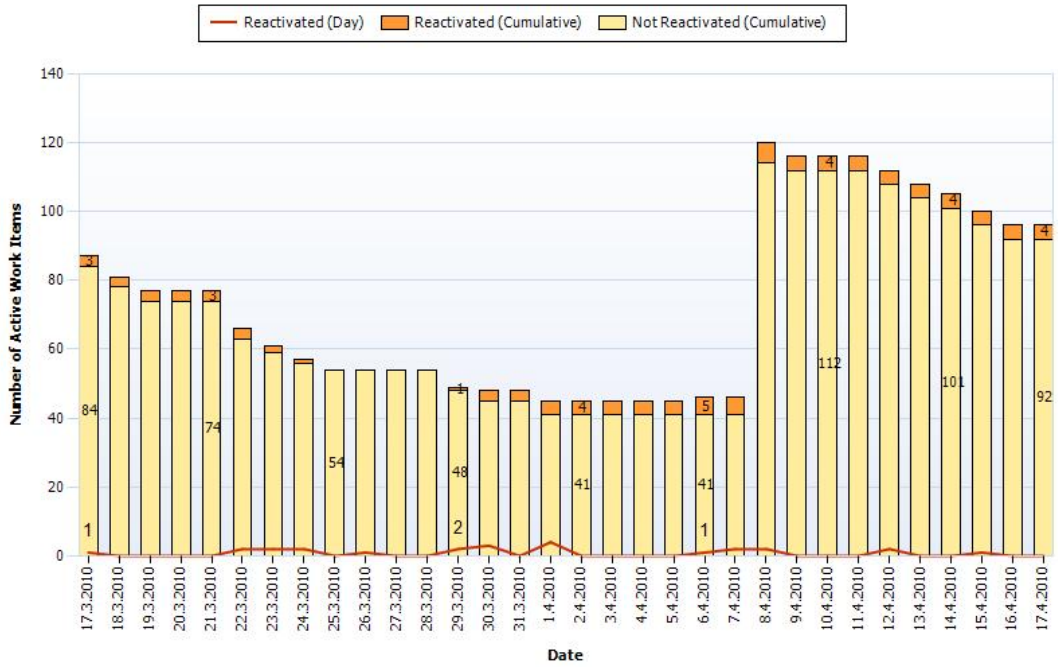
4.3 VSTS-vakioraportit

Kolmantena tiedonkeruutapana oli VSTS-järjestelmän vakiomuotoisiin raportteihin tutustuminen. Kuten aikaisemmin esiteltiin, jokainen Team Project sisältää tietyt vakioraportit sen mukaan, mikä prosessimalli on valittu Team Projectin pohjaksi sitä luotaessa. Vakioraporteissa on varsin hyviä raporttipohjia, mutta suurin osa vakioraporteista toimii vain vakiomuotoisten työkorttien kanssa. Tämän vuoksi niitä ei voi käyttää muokattujen työkorttien tietojen raportoimiseen, koska työkorttien tietokentät eivät vastaa vakioraportin tietokenttiä.

VSTS-järjestelmä kattaa kokonaisuutena monta eri osa-aluetta, ja jokaiseen tarkoitukseen on tehty muutamia vakioraporttimalleja. Monthly Fix -prosessissa ei kuitenkaan käytetä kaikkia VSTS:n osa-alueita, joten osa vakioraporteista on täysin turhia. Monthly Fix -prosessin kannalta raporteilla voidaan seurata lähinnä muutospyyntöjen edistymistä ja niiden tiloja sekä luoda konfiguraatiolistaa toimitettavasta kokonaisuudesta. Automaattisiin testeihin ja Team Buildiin liittyvät raportit ovat toistaiseksi turhia, koska kyseisiä ominaisuuksia ei VSTS-järjestelmän kautta tässä vaiheessa käytetä.

Järjestelmässä on olemassa muutamia toimivia vakioraportteja, mutta niiden käytettävyys on kuitenkin hankalaa, eikä raporteilta saadut tiedot ole kovinkaan täydellisiä.

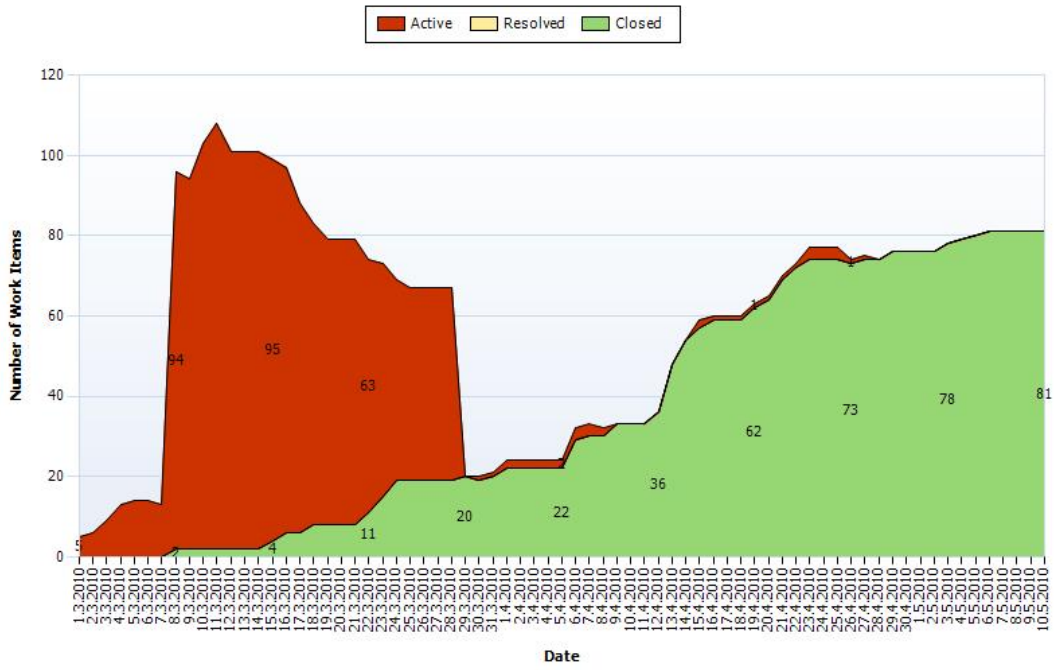
Reactivated-vakioraporttipohja sopii tilanteeseen, jossa tarvitsee tarkastella vain työkorttien lukumääriä. Sillä saa ajettua raportin muutospyynnöistä, jotka ovat käyneet suljettuina mutta avattu uudelleen, koska muutos ei olekaan ollut valmis tai toimiva. (Meier ym. 2007, 453.) Reactivated-raportin (Kuvio 4.1) ongelma on se, että pelkkä palautuneiden virheiden lukumäärä ilman tarkempaa kuvausta palautumisen syystä tai linkkiä työkortille jättää lopputuloksen vähän vajaaksi. Useimmiten olisi myös tarpeen nähdä, mistä tuotteesta virheet ovat palautuneet ja muita tarkempia tietoja suoraan työkortilta.



KUVIO 4.1. Reactivated-vakioraportti.

Toinen vakioraportti, jolle voisi olla käyttöä, on Remaining work –raportti (Kuvio 4.2). Sen avulla nähdään, paljonko työkortteja on käsittelemättä, paljonko niitä on valmiina mutta ei vielä suljettuna ja paljonko työkortteja on jo suljettu. Raportille voi valita halutut Iteraatiot ja Areat sekä rajata raportin koskemaan haluttua ajanjaksoa. Remaining work -raportin trendin avulla on mahdollista ennustaa, koska kyseisen työkohteen kaikki työkortit on suljettuna ja työkohde olisi valmis. (Meier ym. 2007, 453.)

Myös tämän vakioraportin ongelma on sen pelkistetty sisältö; raportilta nähdään ainoastaan työkorttien lukumäärät, mutta ei mitään yksityiskohtaista tietoa esimerkiksi siitä, mitkä työkortit on tekemättä ja mitkä on tehty.



KUVIO 4.2. Remaining Work -vakioraportti.

Yhteistä kaikille vakioraporteille on käyttämisen hankaluus, koska tietokentillä olevia areoita ja iteraatioita on käytännössä liian paljon ja valittavien yhdistelmien lukumäärä on todella suuri. Tästä syystä valittaessa vakioraporteille päätyviä tietoja täytyy tietää todella tarkkaan, missä Team Projektissa ja millä areoilla ja iteraatioilla raportille halutut työkortit sijaitsevat. Tämä puolestaan vaati jo niin pitkällistä perehtymistä työkorttien rakenteisiin, että kaikilla raportteja haluavilla ei sitä ole, eikä ole tarkoituksenmukaistakaan olla.

Vakioraporttien suurin ongelma onkin työkorttien tietojen hajanaisuus ja raporttien liian teknistä näkökulmaa vaativa toteutus. Yksinkertaista tapaa saada esille esimerkiksi kokonaisuus yhden Monthly Fixin muutospyyntöjen tilasta ei vakioraporteilla saa toteutettua.

4.4 Tutkimuksen yhteenveto

Tutkimuksen perusteella voidaan todeta, että VSTS-järjestelmään oltiin varsin tyytyväisiä käyttäjien keskuudessa, ja järjestelmän käyttöönoton myötä koko ylläpito- ja kehitystyö on mennyt selkeästi eteenpäin. Järjestelmän ansiosta ylläpitoprosesseja on saatu parannettua ja muutosten jäljitettävyys on paljon paremmalla tasolla kuin ennen järjestelmän käyttöönottoa. Raportoinnin kehittämiselle on kuitenkin selkeä tarve, koska nykymuodossaan järjestelmässä olevien tietojen tutkiminen on monissa tilanteissa liian hankalaa tai hidasta. Toisaalta VSTS-järjestelmän kaikkien hakuominaisuuksien käyttö ei haastattelujen perusteella ole kaikille täysin selvää ja osa raportointitarpeista olisi ratkaistavissa hakutoimintojen käytön tehostamisella.

Työkorttien muokkaaminen vastaamaan kehitettävän ohjelmistokokonaisuuden monimuotoisuutta on ollut pakollista, vakiomuotoinen Bug-työkortti on liian pelkistetty kattamaan suuren joukon osasovelluksia, asiakkaita ja versioita. Työkorttien muokkaaminen on kuitenkin johtanut tilanteeseen jossa tietokenttien valitseminen kyselyjen kohteeksi on hankaloitunut tietokenttien määrän sekä tietokenttien nimeämiskäytännön vuoksi.

Valmiista vakioraporteista on nähtävissä, että ne eivät sellaisenaan ole varsinaisesti kovinkaan hyödyllisiä Monthly Fix -prosessissa. Raporteilta saa kyllä selvitettyä kokonaistilannetta työkorttien lukumäärän muodossa, mutta sama asia selviää helpommin käyttämällä kyselyjä. Kyselyjen avulla näkee myös samalla mitä keskeneräiset työt ovat ja mihin sovelluksiin ne kohdistuvat. Tarvetta olisikin päästä tarkastelemaan asioita tuotteittain tai asiakaskohtaisesti raporttien kautta, jolloin kyselyjen tekemisen ja muokkaamisen voisi jättää pois.

5 UUDET RAPORTTIMALLIT

Tässä kappaleessa esitellään tutkimuksen perusteella Monthly Fix -prosessin tarpeisiin luotuja uusia VSTS-raporttimalleja ja niiden käyttötarkoitus Monthly Fix -prosessissa.

Tutkimuksen perusteella tehtiin yhteenveto haastatteluissa esille tulleista asioista ja valittiin kehitettäväksi raporttimalleja joista olisi hyötyä mahdollisimman useissa rooleissa toimiville. Työkorttien tietoja ja vakioraportteja tutkimalla rajattiin kehitettäväksi sellaisia raportteja, jotka ovat teknisesti toteuttamiskelpoisia. Käytännössä raporttien toteutus VSTS-järjestelmän Reporting Services -osaan tehdään tämän opinnäytetyön ulkopuolella. Seuraavissa kappaleissa esitellään kehitettävien raporttien mallit ja niiden kehitykseen tarvittavia tietoja.

Kehitettävät raporttimallit:

1. Avoimeksi jääneet vanhat työkortit
2. Trendi avatuista ja suljetuista muutospyynnöistä
3. Asiakasraportti

5.1 Avoimeksi jääneet vanhat työkortit

Ensimmäisenä päätettiin toteuttaa raportti, jolla saadaan tietoa VSTS-järjestelmään pitkäksi aikaa unohtuneista työkorteista. Tutkimuksessa selvisi, että järjestelmän käyttöönotosta lähtien osa työkorteista on jäänyt eri syistä johtuen avoimiksi pitkäksi aikaa. Asia tuli esille useissa haastatteluissa ja unohtuneet työkortit koettiin monessa tilanteessa hankalaksi. Lisäksi turhaan avoimena roikkuvat työkortit vääristävät tilastoja avointen virheiden määrästä. Osa pitkään avoinna olevilla työkorteilla kuvatuista muutospyynnöistä on toteutettu kokonaan, mutta työkorttia ei ole vaan muistettu sulkea. Osa muutospyynnöistä on toteutettu osittain; tehty yhteen versioon, mutta jostain syystä jätetty toteuttamatta johonkin toiseen versioon. Jotkin muutospyynnöt ovat prioriteetiltaan niin alhaalla, että niitä ei tulla välttämättä koskaan toteuttamaan. Järjestelmässä on myös paljon työkortteja, joilla kuvatuista muutospyynnöistä on pyydetty lisäselvitystä ja tarkennusta, mutta sitä ei ole koskaan saatu.

Raportin avulla pyritään vähentämään turhia avoimia työkortteja, jotka roikkuvat ihmisten työjonoissa tehtävinä töinä, jotka eivät kuitenkaan oikeasti ole keskeneräisiksi töiksi laskettavia.

Raportti on lähinnä tarkoitettu tuotepäälliköiden, tuotevastaavien ja asiakaspalvelun asiakasryhmäpäälliköiden käyttöön. Eniten hyötyä raportilla saavutettaneen, kun pitkään avoinna olleita työkortteja haettaisiin raportin avulla ennen Monthly Fixin aloitusta pidettävää Product Council -palaveria.

Raportilla täytyy pystyä hakemaan työkortteja viimeisimmän muutospäivämäärän mukaan antamalla päivämäärä, joita vanhempia työkortteja raportille otetaan mukaan. Lisäksi raportilta tulisi pystyä valitsemaan pikavalintatyylisesti esimerkiksi kaikki yli kolme kuukautta muuttamattomina olleet, mutta edelleen aktiivisessa tilassa olevat työkortit. Raportin sisältöä täytyy pystyä rajaamaan myös työkorttien tilan ja prioriteetin mukaan. Hakuehtoja on kuvattu tarkemmin taulukossa 5.1.

TAULUKKO 5.1. Avoimena olevien työkorttien raportin hakuehdot.

Työkortin viimeisin muutospäivä	Valittavissa raportilta, esimerkiksi: - Muutettu viimeksi yli 1, 2 tai 3 kk sitten. - Päivämäärärajaus
Työkortin tila	Valittavissa halutut, esimerkiksi - Active - Investigation
Prioriteetti	Mahdollisuus rajata, esimerkiksi - Critical - High

Raportilta (Kuva 5.1) tulee nähdä työkortin otsikko, kenelle työkortti on osoitettu, päivämäärätiedot koska muutoksia on edellisen kerran tapahtunut, työkortin yksilöllinen järjestysnumero, sisäinen prioriteetti sekä työkortin tila.

Yli 3kk muuttumattomina olleet avoimet työkortit

ID	Title	Changed Date	State	Assigned To	Internal Priority
26852	Bug/Feature 1	22.12.2009 10:05:15	Investigation	Tuotepäällikkö 1	High
31949	Bug/Feature 2	30.11.2009 11:14:26	Investigation	Tuotepäällikkö 2	High
32513	Bug/Feature 3	8.1.2010 16:19:20	Investigation	Tuotepäällikkö 1	High
33585	Bug/Feature 4	13.10.2009 13:27:04	Investigation	Tiimipäällikkö 1	High
33935	Bug/Feature 5	12.10.2009 13:09:49	Investigation	Tuotepäällikkö 1	High
34141	Bug/Feature 6	12.10.2009 13:11:31	Investigation	Tuotepäällikkö 1	Critical
36543	Bug/Feature 7	19.1.2010 10:08:43	Active	Kehittäjä 1	Critical
37612	Bug/Feature 8	22.12.2009 9:44:33	Investigation	Tuotepäällikkö 1	High
38784	Bug/Feature 9	19.10.2009 14:48:18	Investigation	Tuotepäällikkö 1	High
38788	Bug/Feature 10	22.12.2009 10:22:33	Investigation	Tuotepäällikkö 1	High
39301	Bug/Feature 11	14.10.2009 15:16:56	Investigation	Tuotepäällikkö 3	High
39499	Bug/Feature 12	20.10.2009 9:41:43	Investigation	Tuotepäällikkö 2	High
40222	Bug/Feature 13	20.10.2009 10:05:37	Investigation	Tuotepäällikkö 1	High
40227	Bug/Feature 14	15.10.2009 9:21:31	Active	Kehittäjä 2	High
40230	Bug/Feature 15	11.11.2009 8:37:16	Investigation	Tuotevastaava 1	High

KUVA 5.1. Malliraportti avoimeksi jääneistä työkorteista.

5.2 Trendi avatuista ja suljetuista työkorteista

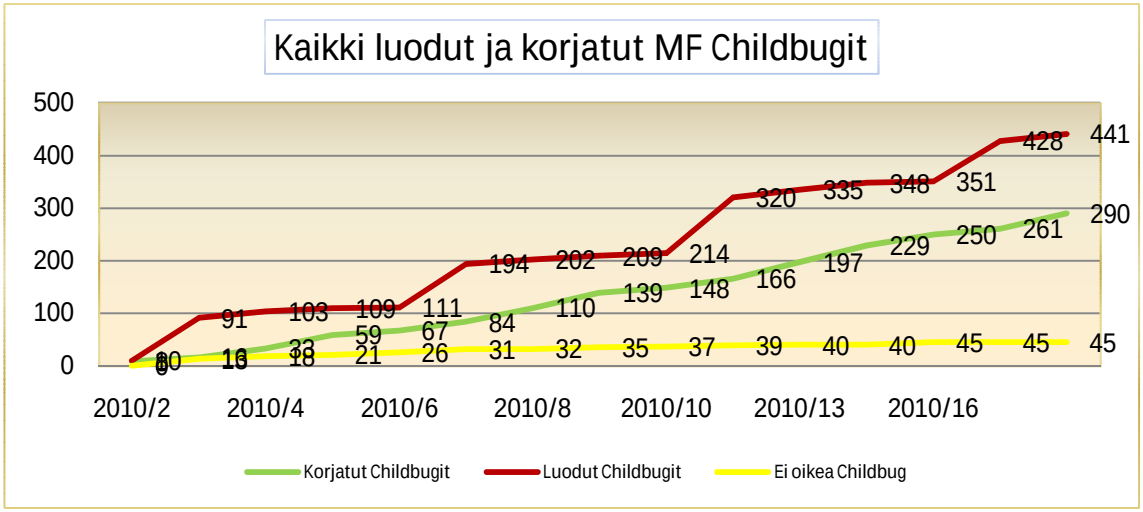
Toinen tärkeäksi havaittu raportti auttaa hallitsemaan Monthly Fixin kokonaistilannetta. Raportilla kuvataan kuinka paljon viikoittain avataan uusia työkortteja ja toisaalta paljonko niitä suljetaan.

Raportin hakuehtoina (Taulukko 5.2) täytyy olla useita valittavissa olevia tietokenttiä, joilla pystytään rajaamaan raportti kohdistumaan halutulle ajanjaksolle ja Monhly Fix -kierrokselle. Raportti on tarkoitettu ensisijaisesti tuotekehityksen tarpeisiin, mahdollistamaan sovelluskohtaisen ja koko järjestelmän pidempiaikaiseen laadun seurannan.

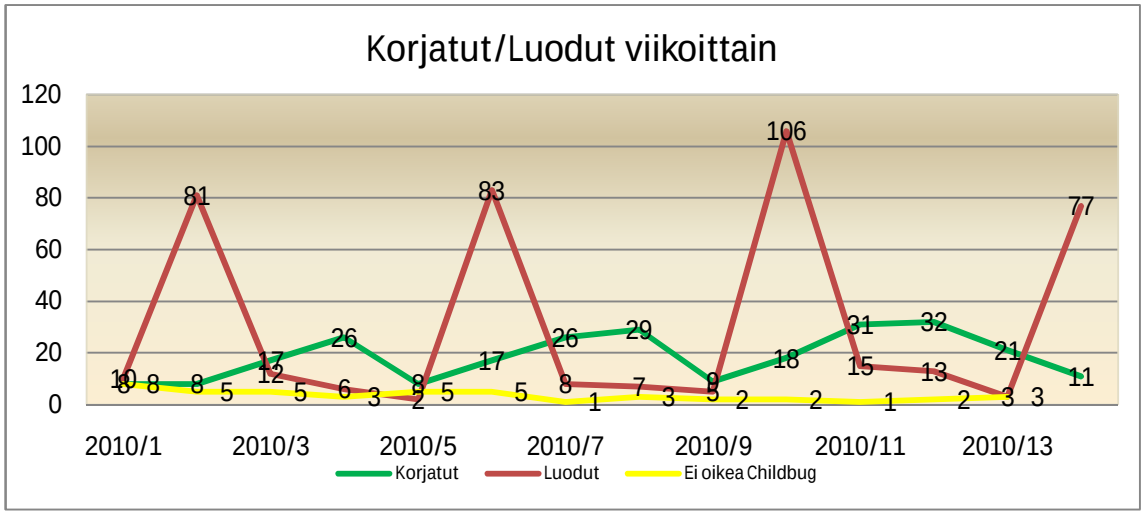
TAULUKKO 5.2. Avattujen ja suljettujen työkorttien hakemiseen tarvittavat hakuehdot.

Työkortin avaamispäivämäärä	Valittavissa raportilta: - Päivämäärärajaus, kalenterikomponentilta
Työkortin sulkemispäivämäärä	Valittavissa raportilta: - Päivämäärärajaus, kalenterikomponentilta
Työkortin tila	Valittavissa halutut, esimerkiksi - Active - Investigation
Area	Valittavissa halutut sovellukset, esimerkiksi: - Ajanvaraus - Osastonhallinta
Iteraatio	Mahdollisuus rajata Monthly Fix -kohtaisesti

Trendiä kuvaavalta raportilta tulee nähdä viikoittaiset uusien työkorttien määrät sekä suljettujen työkorttien määrät. Raporttia tulisi voida tarkastella kumulatiivisena pidemmän aikavälin kuvaajana (Kuvio 5.1) sekä viikkokohtaisena raporttina (Kuvio 5.2).



KUVIO 5.1. Malliraportti, jossa kumulatiivinen tilanne viikoittain kuvattuna



KUVIO 5.2. Malliraportti, jossa viikoittain luodut ja korjatut työkortit

5.3 Asiakaskohtaiset työkortit

Kolmanneksi kehityskohteeksi valittiin lähinnä asiakaspalvelua hyödyttävä asiakaskohtainen raportti. Haastatteluissa tuli esille, että asiakaspalvelun työssä tulee tilanteita, joissa asiakas tiedustelee esimerkiksi puhelinkeskustelun aikana ilmoittamiensa muutospyyntöjen tilannetta. Tämän raportin tarkoitus onkin helpottaa asiakaskohtaisen tilanteen hahmottamista esimerkiksi kuvatun kaltaisissa nopeissa puhelinkeskusteluissa sekä muissa asiakaspalavereissa.

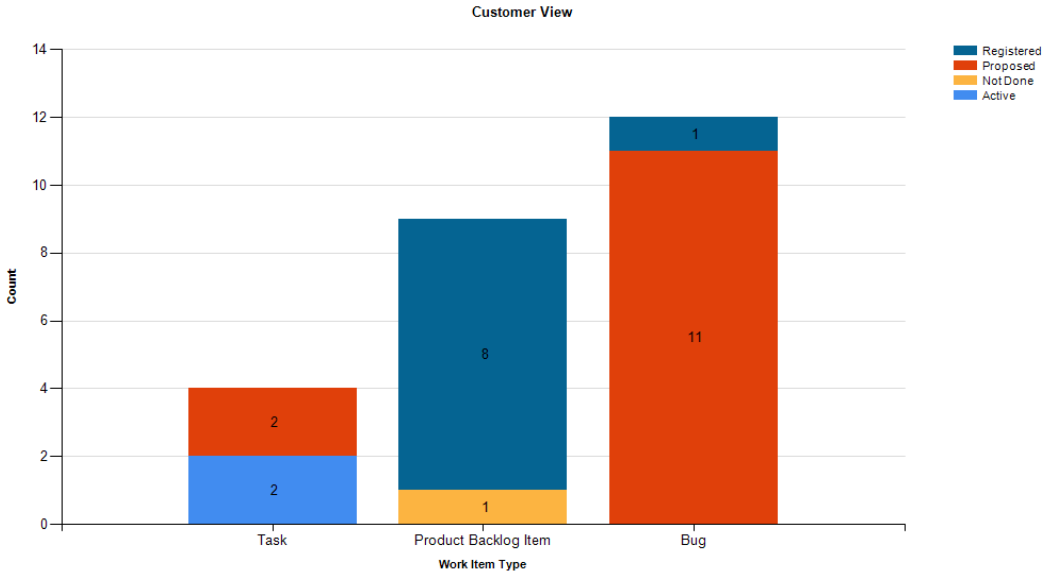
Asiakasraportilta tulee selvitä tietyn asiakkaan ilmoittamat muutospyynnöt ja niiden tilat. Valittavissa täytyisi olla siis asiakas, raportille haettavien työkorttien tyyppi ja tila sekä mahdollisuus rajata hakua koskemaan tiettyä ajanjaksoa (Taulukko 5.3).

Raportin toteuttamisen haasteena ovat työkorteilla olevat vapaat tekstikentät ja erilaiset kirjaamiskäytännöt. Saman asiakkaan nimi voi olla lyhennetty hyvin monella tavalla ja osa muutospyynnöistä voi jäädä pois raportilta. Raportin täytyisikin tukea kirjavaa merkintätapaa ja selkiyttää asiakaspalvelun näkemää tilannetta.

TAULUKKO 5.3. Asiakaskohtaisen raportin luomiseen tarvittavat hakuehdot

Asiakas	Valittavissa raportilta: - Yhden tai useamman asiakkaan mukaan
Työkorttien tyypit	Valittavissa raportilta: - Virheet, muutospyynnöt ja selvityspyynnöt
Työkortin tila	Valittavissa halutut, esimerkiksi: - Active - NotDone

Asiakasraportilta tulee nähdä valitun asiakkaan kaikkien erityyppisten muutospyyntöjen tilat ja niiden määrät selkeästi jaoteltuina esimerkiksi kuvion 5.3 tavoin.



KUVIO 5.3. Malliraportti yhden asiakkaan työkorttien tiloista eriteltynä tyypin mukaan

6 YHTEENVETO

Tässä opinnäytetyössä on tutkittu raportoinnin kehittämistä ketterää menetelmää hyödyntävässä ohjelmiston ylläpitoprosessissa.

Tärkeimmäksi tutkimuskohteeksi raportoinnin kehittämiselle valittiin prosessissa päivittäin eri rooleissa toimivat ihmiset. Tavoitteena oli saada käytännönläheistä tietoa jokapäiväisessä työssä vastaan tulevista tilanteista ja ongelmista, joita olisi mahdollista ratkaista kehittämällä raportointia. Lisäksi tutkittiin VSTS-järjestelmään kirjattuja työkortteja sekä tutustuttiin järjestelmän vakiotoimintoihin.

Haastattelujen valinta keskeisimmäksi tutkimusmenetelmäksi oli ehdottomasti oikea ratkaisu. Haastattelemalla monen eri käyttäjäryhmän henkilöitä saatiin kattava kuva siitä, millaisia raportointitarpeita eri tehtävissä toimivilla henkilöillä oikeasti on. Samalla selvisi eri käyttäjäryhmien tapoja käyttää VSTS-järjestelmää omassa työssään. Haastattelujen järjestäminen sujui ennalta oletettua helpommin. Käytössä oli hyvä tekninen apuväline Microsoft Live Meeting, jonka avulla haastatteluista pystyi tallentamaan keskustelut sekä jaetulla työpöydällä käsitellyt asiat. Live Meetingin avulla haastattelut oli helppo purkaa jälkeenpäin, jolloin ei tarvinnut tuhlata haastateltavien aikaa vastauksien kirjaamiseen ja haastattelu eteni jouhevasti ilman ylimääräisiä katkoja. Haastateltavat henkilöt suhtautuivat haastatteluihin positiivisesti ja useimmat olivat perehtyneet aiheeseen ennalta toimitettujen kysymysten ansiosta hyvin. Monet haastateltavista pystyivät myös esittämään hyvin tarkkoja kehitysehdotuksia järjestelmän kehittämiseksi.

Tutkimuksen perusteella saatiin selville useita konkreettisia asioita, joihin kaivattiin parannuksia, ja pystyttiin luomaan raporttimallit tärkeimmiksi katsotuista kohteista. Kokonaisuutena kehitystyö onnistui annettuihin tavoitteisiin nähden hyvin.

Suurin hankaluus koko VSTS-järjestelmän käytön kehittämiselle prosessin osana on resursointi. Tutkimuksen perusteella on selvää, että raportoinnin kehittäminen helpottaisi monien käyttäjäryhmien jokapäiväistä työtä. Kehitystyön läpivieminen on kuitenkin hidasta, koska kehittämiselle ei ole henkilöresursseja, joiden työkuvaan ei kuuluisi sisäisten kehitystehtävien edelle meneviä asiakasprojekteihin liittyviä tehtäviä. Yrityksen sisäisessä käytössä oleva järjestelmä on yritykselle kuitenkin periaatteessa vain kuluerä, joten sen kehittämisellä täytyisi pystyä saavuttamaan säästöjä jossain toi-

sessa vaiheessa prosessia. Jatkossa kehitystyö olisi mielestäni järkevintä projektisoida ja valita projektille sellaiset resurssit, jotka eivät ole sidoksissa muihin tehtäviin.

Oman haasteensa VSTS-järjestelmälle tuo ylläpidettävän potilastietojärjestelmän monimuotoisuus. Mielestäni VSTS-järjestelmä on parhaimmillaan, kun sitä käytetään yksittäisen projektin kehitysympäristönä ja voidaan hyödyntää sen kaikkia ominaisuuksia, siten kuin valmistaja on tarkoittanut. Ylläpidettäviä potilastietojärjestelmän rinnakkaisia versioita sen sijaan on useita, ja kokonaisuus muodostuu kymmenistä osasovelluksista ja projekteista. Sen vuoksi VSTS-järjestelmää on täytynyt muokata monin eri tavoin, eikä muokkauksen myötä ongelmiltakaan ole täysin voitu välttyä. Esimerkiksi vakiomuodosta muokatut työkortit vaativat toisinaan ylläpitoa ja muokkausten vuoksi uuden VSTS-version käyttöönotto hankaloituu. Saman järjestelmän on taivuttava ylläpitoprosessin lisäksi myös lukuisien projektien kehitysympäristöksi.

Koska kyseessä on varsinaisen työn tekemiseen käytettävä työkalu, sen tulisi toimia mahdollisimman sujuvasti ja helposti, ikään kuin huomaamatta. Toistaiseksi ollaan kuitenkin tilanteessa, jossa työkalun käyttö ei ole vielä kaikilta osin niin helppoa, että voisi puhua pelkästään taustalla käytettävästä työkalusta, vaan sen käytössä on edelleen selkeitä ongelmakohtia ja kehitettäviä asioita. Osa ongelmista voi ratketa uuden VSTS 2010 -version uusien ominaisuuksien avulla, mutta esimerkiksi potilastietojärjestelmän rinnakkain ylläpidettävien versioiden määrä teettää jatkossakin moninkertaista työtä osassa prosessin vaiheista ja edellyttänee myös jatkossa VSTS-järjestelmän muokkaamista.

Kehittämistehtävän tekeminen oman jokapäiväisen työn ohessa oli haasteellista. Vaikka kehitystehtävän aiheena olevan järjestelmän parissa toimii päivittäin, on omien tehtävien lisäksi tulevat lisätyöt hankala aikatauluttaa sopimaan muihin töihin.

Kehittämistehtävän haastattelut olivat antoisia jokapäiväisen työn kannalta. Haastattelujen ansioista omaa työympäristöään ja oman työn vaikutuksia pääsi tarkastelemaan laajemmin kuin tavallisesti ja prosessin kokonaiskuva tarkentui. Samalla havaitsi, miten prosessin eri osa-alueet muuttavat merkitystä, kun niitä tarkastelee organisaation eri puolilta. Jotkin kehitysprosessin asioista ovat hyvin tärkeitä tuotekehityksen näkökulmasta, mutta eivät lainkaan merkityksellisiä asiakaspalvelun työn kannalta ja päinvastoin.

Seuraavassa vaiheessa tämän työn tuloksien perusteella esitetyt raportit kehitetään lopulliseen muotoonsa. Jatkokehityksenä tulisi tutkia, kuinka tuleva VSTS 2010 -versio muuttaa koko toimintaympäristöä. Uuden version tuomat muutokset on joka tapauksessa otettava huomioon jo ensimmäisiä raportteja kehitettäessä ja järjestelmän muuttuessa koko ylläpitoprosessia voi joutua tarkentamaan joiltakin osin.

LÄHTEET

Agile Manifesti 2001. *Manifesto for Agile Software Development* [verkkosivu]. [viitattu 16.3.2010]. Saatavissa: http://agilemanifesto.org.

Agile Manifestin Periaatteet 2001. *Principles behind the Agile Manifesto* [verkkosivu]. [viitattu 2.7.2010]. Saatavissa: http://www.agilemanifesto.org/principles.html.

Codeplex 2008. *VSTS Scrum Process Template* [verkkosivu]. [viitattu 12.6.2010]. Saatavissa: http://vstsscrum.codeplex.com/.

Cottrell, W. 2004. *Standards, compliance, and Rational Unified Process, Part I: Integrating RUP and the PMBOK* [verkkosivu]. 26.5.2004 [viitattu 15.8.2010]. Saatavissa: http://www.ibm.com/developerworks/rational/library/4763.html.

Deemer, P., Benefield, G., Larman, C., & Vodde, B. 2010. *The Scrum Primer* [verkkojulkaisu]. [viitattu 15.1.2010]. Saatavissa: http://scrumtraininginstitute.com/home/stream_download/scrumprimer.

Eclipse Foundation 2008. *OpenUP wiki* [verkkosivu].16.11.2008 [viitattu 15.05.2010]. Saatavissa: http://epf.eclipse.org/wikis/openup/.

Eclipse Foundation 2007. *Introduction to OpenUP paper* [verkkojulkaisu]. [viitattu 20.2.2010]. Saatavissa: http://eclipse.org/epf/general/OpenUP.pdf.

Gustafsson, B. 2010. *OpenUP - the best of two worlds* [verkkojulkaisu]. [viitattu 15.3.2010] Saatavissa: http://www.projectkoach.com/papers/OpenUP_2_worlds.pdf.

Haikala, I., & Märijärvi, J. 2004. *Ohjelmistotuotanto.* Hämeenlinna: Talentum Media Oy.

Hirsjärvi, S., & Hurme, H. 2000. *Tutkimushaastattelu.* Helsinki: Yliopistopaino.

Hirsjärvi, S., Remes, P. & Sajavaara, P. 2005. *Tutki ja kirjoita.* Jyväskylä: Gummerus Kirjapaino Oy.

Kroll, P. 2007. *OpenUP In a Nutshell* [verkkosivu]. 15.9.2007 [viitattu 20.3.2010]. Saatavissa: http://www.ibm.com/developerworks/rational/library/sep07/kroll/#N100D5.

Levinson, J. & Nelson, D. 2006. *Pro Visual Studio 2005 Team System.* Berkeley: Apress.

McConnell, S. 1998. *Ohjelmistoprojektit - selviytymisopas.* Kääntänyt Marko Juoperi. Jyväskylä: Suomen Atk-kustannus Oy.

Meier, J.D, Taylor, J., Mackman, A., Bansode, P. & Jones, K. 2007. *TFSGuide.2007-08-02.2* [verkkojulkaisu]. [viitattu 15.3.2010]. Saatavissa: http://tfsguide.codeplex.com.

Microsoft Corporation 2008. *Visual Studio 2008 -versiot - Microsoft Visual Studio* [verkkojulkaisu]. [viitattu 10.4.2010]. Saatavissa: http://download.microsoft.com/download/1/b/f/1bf59803-9bd0-4475-94c1-565d1708ec50/6_Visual%20Studio%202008%20Yleiskatsaus.pdf.

Microsoft Corporation 2010. *Team Foundation Team Projects* [vekkosivu]. [viitattu 12.5.2010]. Saatavissa: http://msdn.microsoft.com/en-us/library/ms181234.aspx.

Pagels, R. 2010. *Team System Café.* [verkkosivu]. 21.10.2008 [viitattu 13.6.2010]. Saatavissa: http://www.teamsystemcafe.net/VSTS.aspx.

Stott, W. & Newkirk, J. 2007. *Visual Studio Team System : Better Software Development for Agile teams.* Boston: Addison-Wesley.

www.savonia.fi