Opinnäytetyö (AMK)

Tietojenkäsittely

Yrityksen tietojärjestelmät

2016

Jussi Maittila

KETTERIEN MENETELMIEN KÄYTTÖ TIETOVARASTOINNISSA

TURKU AMK
TURKU UNIVERSITY OF APPLIED SCIENCES

OPINNÄYTETYÖ (AMK) | TIIVISTELMÄ

TURUN AMMATTIKORKEAKOULU

Tietojenkäsittely | Yrityksen tietojärjestelmät

2016 | 38 sivua

Anne Jumppanen

Jussi Maittila

KETTERIEN MENETELMIEN KÄYTTÖ TIETOVARASTOINNISSA

Tämän opinnäytetyön tavoitteena on selvittää, miten hyvin toimeksiantajayrityksessä hyödynnetään ketteriä menetelmiä osana tietovarastointia. Toisena tavoitteena on löytää mahdollisia kehityskohteita toimeksiantajan käyttämissä menetelmissä. Opinnäytetyön toimeksiantajayritys on ottanut ketterät menetelmät käyttöön osana tietovarastointia kuusi kuukautta ennen opinnäytetyön aloittamista.

Työn teoriaosa käsittelee tietovarastointia ja keskittyy ketteristä menetelmistä Scrum-viitekehykseen ja tietovarastoinnin tuomiin haasteisiin menetelmien käytössä. Teoriaosuuden lähteinä on käytetty laajasti ketteriä menetelmiä ja tietovarastointia käsitteleviä teoksia, sekä erityisesti tietovarastoinnin kannalta ketteriin menetelmiin keskittyneitä teoksia.

Empiirinen osa koostui tietovarastointiprosessiin osallistumisesta kehittäjän roolissa. Tutkimuksessa toteutettiin ETL-prosessi käyttäen ketteriä menetelmiä ja kuvataan yrityksessä käytettyjä menetelmiä. Tutkimuksessa saatuja kokemuksia peilataan teoreettiseen viitekehykseen kehityskohteiden löytämiseksi.

Tutkimuksen aikana toteutettiin ETL-prosessi asetetuissa aikataulutavoitteissa käyttäen ketteriä menetelmiä. Tutkimuksen pohjalta saatiin kattavasti tuloksia ketterien menetelmien sopivuudesta tietovarastointiin. Tulosten pohjalta pystyttiin esittämään toimeksiantajayritykselle kehitysideoita ja asioita, joihin tulisi kiinnittää erityisesti huomiota.

ASIASANAT:

Ketterät menetelmät, tietovarastointi, Scrum, ETL

BACHELOR´S THESIS | ABSTRACT

TURKU UNIVERSITY OF APPLIED SCIENCES

Business Information Technology | Business Information Systems

2016 | 38 pages

Anne Jumppanen

Jussi Maittila

AGILE METHODS IN DATA WAREHOUSING

The aim of this bachelor's thesis is to find out how agile methods have been adapted to this thesis client's data warehousing process. The second aim is to find out possible ways to improve the used agile methods. The client of the thesis has started using agile methods in the autumn of 2015.

The theoretical part of the thesis introduces the concept of data warehousing and agile methods. In addition, the theoretical section of the thesis concentrates on the Scrum framework and introduces characteristics of agile data warehousing.

The empirical section of this thesis is based on taking part in the client's data warehousing process as a developer. The empirical section includes the description of the Extract – Transform – Load (ETL) process carried out with using agile methods and common description of client's agile methods. Findings made during study are used along with the theoretical framework to find out possible areas for improvement.

The ETL -process was completed successfully achieving the given goals. The case study provided good evidence about how agile methods are suitable for data warehousing projects. Several areas for improvement possibilities were came up in the client's agile methods.

KEYWORDS:

Agile methods, data warehousing, Scrum, ETL

SISÄLTÖ

KUVAT

KUVIOT

TAULUKOT

SANASTO

BI	Business Intelligence, tietojen käyttämistä hyväksi liiketoiminnassa ja siihen liittyviä työkaluja (Hovi ym. 2009, 188).
Dataset	Tietojen tallentamiseen IBM DataStage-ohjelmistossa käytettävä tiedostotyyppi (IBM 2016).
ETL	Extract – Transform – Load, prosessi tietojen tietovarastoon siirtämiseksi (Hovi ym. 2009, 48).
Iteraatio	Useaan kertaan suoritettava prosessin toistaminen (Barbee 2012, 13).
SQL	Structured Query Language on tietokantojen kanssa käytettävä kyselykieli (Microsoft 2016).
UML	Unified Modelin Language, standardoitu graafinen mallinnuskieli (Lucidchart 2015).

1 JOHDANTO

Ketterät menetelmät ovat yleistyneet ohjelmistotuotantoprojekteissa viimeisen vuosikymmen aikana ja niitä käytetään suurimmassa osassa ohjelmistotuotantoprojekteja jossain muodossa. Tietovarastointiprojekteissa ketterien menetelmien käyttö ei ole yleistynyt yhtä nopeasti ja suosio tulee kymmenen vuotta jäljessä perinteiseen ohjelmistokehitykseen verrattuna. (Hughes 2013, 3.)

Tämän opinnäytetyön tavoitteena on selvittää, miten hyvin toimeksiantajayrityksessä hyödynnetään ketteriä menetelmiä osana tietovarastointia. Toisena tavoitteena on löytää mahdollisia kehityskohteita toimeksiantajan tietovarastoinnissa käyttämissä ketterissä menetelmissä. Aihe on ajankohtainen, koska ketterät menetelmät on otettu käyttöön toimeksiantajayrityksen tietovarastoinnissa kuusi kuukautta ennen tämän työn aloittamista. Lisäksi ketteriä menetelmiä on tutkittu laajalti ohjelmistoprojektien näkökulmasta, mutta tietovarastoinnin näkökulmasta tehtyjä tutkimuksia on saatavilla vain vähän.

Opinnäytetyön toimeksiantajayritys on monikansallinen yhtiö, jonka toiminnassa tietovarastointi on tärkeässä asemassa. Yritys on johtavassa asemassa omalla alallaan, joka on tekemisissä yksityisasiakkaiden sekä yritysasiakkaiden kanssa. Tietovarastointi on yksi tekijöistä, jotka auttavat yritystä tarjoamaan kilpailukykyisiä tuotteita ja kehittämään asiakkailleen entistä sopivampia tuotteita. Ketterät menetelmät on otettu käyttöön yrityksen tietovarastoinnissa 2015 vuoden syksyllä ja toimeksiantaja halusi tutkittavan miten heidän käyttämät ketterät menetelmät toimivat tietovarastoinnissa, ja löytyykö niissä kehittämisen kohteita.

Työn teoriaosuudessa käsitellään tietovarastointia, ketteriä menetelmiä yleisesti sekä perehdytään tietovarastoinnin ketterille menetelmille asettamiin haasteisiin. Teoriaosuuden aihe keskittyy ketterien menetelmien kohdalta käsittelemään tarkemmin Scrum-viitekehystä, koska toimeksiantajan käyttämät ketterät menetelmät perustuvat siihen.

Työn empiirisessä osuudessa kuvataan tietovarastointiprosessia kehittäjän näkökulmasta ja peilataan siinä saatuja kokemuksia teoriatietoon. Empiirisen osuuden ohessa pyritään löytämään kehitysideoita yrityksen ketterien menetelmien käytössä.

Opinnäytetyö on tyypiltään toiminnallinen. Toiminnallinen opinnäytetyö on eräänlainen kehittämis- ja tutkimustyö. Työn tulee pohjautua ammattiteorialle ja sisältää toiminnalli-

nen osuus. (Vilkka & Airaksinen 2003, 9.) Opinnäytetyn lähestymistapana on tapaustutkimus (case study). Tutkimuskysymys on miten yrityksen käyttämät ketterät menetelmät sopivat tietovarastointiin ja löytyykö menetelmistä kehityskohteita. Tutkimuskysymykseen etsitään vastausta osallistumalla tietovarastointiin kehittäjän roolissa sekä kuvailemalla yksittäistä tapausta. Pääpaino on tutkimuskohteen kuvaamisessa ja tehtyjen havaintojen esiin tuomisessa, ei yleistettävien teorioiden luomisessa. (Saaranen-Kauppinen & Puusniekka 2006).

2 TIETOVARASTOT

2.1 Määritelmä

Yrityksillä on nykyaikana monenlaista pääomaa, jota kutsutaan myös resursseiksi. Niihin lasketaan esimerkiksi henkilöstö, tuotteet ja yrityksen tiedot. Tietoja kertyy jokapäiväisistä toiminnoista valtava määrä. Ne ovat arvokkaita resursseja ja yritysten pääomaa, joka pitäisi olla hyvin kuvattuna ja saatavilla palvellakseen liiketoiminnan tarpeita. Päätöksenteon tueksi organisaatioiden päättäjät ja johtajat tarvitsevat monenlaista tietoa. Suurin osa päätöksiin tarvittavista tiedoista on jo olemassa organisaation sisällä, eri järjestelmissä ja niiden tietokannoissa, mutta sen saamiseksi käyttöön vaaditaan suuria ponnistuksia. (Hovi ym. 2009, X, 4.)

Yritysten käyttämiä perusjärjestelmiä kutsutaan usein operatiivisiksi järjestelmiksi, ja ne on toteutettu tapahtumankäsittelyjärjestelmiksi (OnLine Transaction Processing System) (Hovi ym. 2009, 22). Niiden tehtävä on huolehtia päivittäisten toimintojen hallinnasta. Esimerkiksi verkkokaupasta tehdystä tilauksesta tallennetaan useita tapahtumia operatiiviseen järjestelmään, josta muodostuu yksittäinen kokonaisuus. Tapahtumankäsittelyjärjestelmät ovat suunniteltu erityisesti tiedon tallennukseen tietokantoihin ja ne pystyvät käsittelemään suuria määriä yhtäaikaisia ja toistuvia tapahtumia ongelmitta. (Ponniah 2010, 12.)

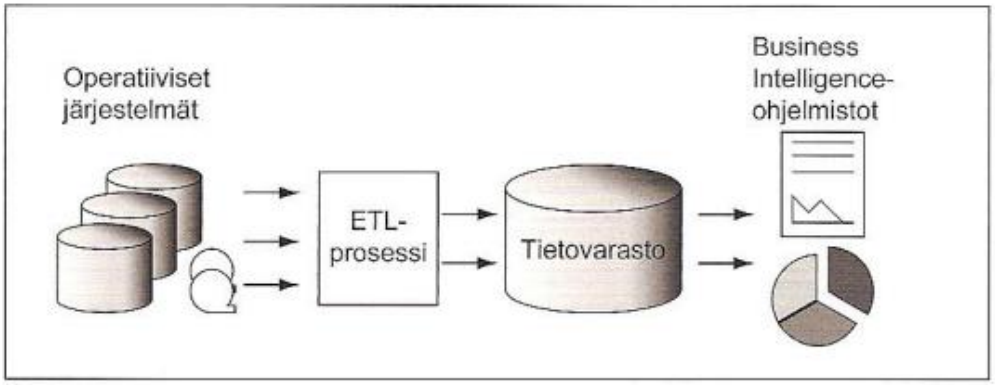
Jotta tietoja pystytään hyödyntämään tehokkaasti raportteina ja analyyseina, edellyttää se juuri tähän käyttötarkoitukseen räätälöityjä ja suunniteltuja tietokantoja eli tietovarastoja (Hovi ym. 2009, 14). Informatiiviset järjestelmät on suunniteltu tietojen lukuun eikä tallentamiseen. Muita eroavaisuuksia operatiivisiin järjestelmiin on pienempi käyttäjämäärä, käyttöaste ja käyttötapaukset. (Ponniah 2010, 13.)

Tietovaraston tärkeimmät tehtävät Ponniahin mukaan ovat

- Tuottaa integroidun ja yhtenäisen kuvan yrityksen tiedoista
- Tuo yrityksen nykyisen ja historiallisen tiedon helposti saataville päätöksenteon tueksi
- Mahdollistaa päätöksenteon tuen ilman operatiivisiin järjestelmiin koskemista
- Esittää strategiset tiedot
- Tuo yrityksen tiedot ristiriidattomasti yleisesti saataville (Ponniah 2010, 15.)

Tietovarastot toteutetaan nykypäivänä yleisemmin relaatiotietokantatekniikalla (RDBMS). Relaatiotietokannat esitteli ensimmäisen kerran 1970-luvun alussa Edgar Frank Codd. Mallia pidetään erittäin yksinkertaisena ja sen tietorakenteet ovat yhtenäiset. Relaatiomallissa tiedot yhdistetään toisiinsa useilla relaatioilla. (Vaze & Joshi 2010, 75–76.)

Kuviossa 1 esitetään perinteistä tietovarastointi prosessia. Tietovarastosta tiedot saadaan liiketoimintatiedon hallinnan käyttöön (Business Intelligence). BI auttaa yritysten liiketoimintaa tekemään harkitumpia päätöksiä ja ohjaamaan toimintaa oikeaan suuntaan informaatioon perustuen. (Hovi ym. 2009, 74.)



Kuvio 1. Esimerkki tietovarastointi prosessista (Hovi ym. 2009, 14).

2.2 ETL-prosessi

ETL-prosessissa (Extract – Transform – Load) perusjärjestelmien tiedot luetaan ja muokataan tietovarastoon sopivaan muotoon, mikä edellyttää yhdenmukaistamista ja yhdistämistä. ETL-vaiheen lopuksi tiedot ladataan tietovarastoon. (Hovi ym. 2009, 14.)

Prosessin ensimmäisessä vaiheessa tiedot poimitaan operatiivisista järjestelmistä tai muista ulkoisista lähteistä, kuten Excel-taulukoista. Poiminta on helpoimmillaan erittäin suoraviivaista, jos tiedot on tallennettu relaatiotietokantaan. Mutta usein operatiivisten järjestelmien monimuotoisuus vaikeuttaa poimintavaihetta ja on järkevintä, että poiminnan suorittavat järjestelmien ylläpitäjät, sillä he tuntevat parhaiten tietojen sijainnit ja järjestelmien rakenteet. (Hovi ym. 2009, 55.)

Tietojen muokkausvaihe on tärkeä osa ETL-prosessia. Kaikki poimittu tieto yhtenäistetään tietovarastoon sopivaan muotoon, jotta raportointia ja analysointia on mahdollista tehdä (Ponniah 2010, 38). Ensin varmistetaan tietojen oikeellisuus, puhdistamalla virheelliset tiedot ja poistamalla päällekkäisyydet, jos samaa tietoa tuodaan eri lähteistä. Muita esimerkkejä tietoihin tehtävistä muokkauksista ovat koodien varustaminen selitteillä, kuten postinumeron viereen kunnan nimen lisääminen tai koodien yhdenmukaistaminen esimerkiksi sukupuolien merkitsemisessä. Toisessa järjestelmässä ne voi olla merkitty M/F ja toisessa M/N, tällöin ne muutetaan vastaamaan yhtenäistä muotoa. (Hovi ym. 2009, 56.)

ETL-prosessin viimeinen vaihe on muokattujen tietojen lataaminen tietovarastoon. Tehtävät lataukset voidaan jakaa kahteen kategoriaan, ensimmäisellä kerralla tehtävään historialatauksiin ja päivityslatauksiin. Historialatauksessa tarkoituksena on tuoda suuri määrä tietoa kerralla tietyn aikajakson ajalta. Myöhemmin päivityslatauksissa tuodaan vain uusimmat ja päivittyneet tiedot halutuin väliajoin. (Ponniah 2010, 39.)

2.3 ETL-toteutus

Ennen ETL-prosessi suoritettiin manuaalisesti ohjelmoimalla. Nykyään erityisesti tiedon integrointiin kehitetyt ETL-työkalut ovat korvanneet pitkälti manuaalisen ohjelmoinnin. Monimutkaisen ohjelmointi- ja SQL-koodin kirjoittaminen on usein vaivalloista ja hidasta, sekä koodin huoltaminen ja muokkaaminen jälkikäteen aiheuttaa paljon ongelmia. (Prabhu 2007.) ETL-työkalut ovat suunniteltu juuri latausprosessien toteutukseen.

ETL-työkalujen tyypillisiä ominaisuuksia ovat

- Graafinen käyttöliittymä
- FTP-tuki
- Valmiit funktiot tiedon muokkausvaiheeseen
- SQL-käskyjen suoritusmahdollisuus
- Tietojen yhdistelymahdollisuudet useasta lähteestä
- Latausajojen loki tiedot ja tilastot
- Uudelleenkäynnistettävyys eri prosessin vaiheista (Hovi ym. 2009, 61.)

Nykyään markkinoilla on useita kymmeniä ETL-työkalujen toimittajia. Tunnetuimpia ETL-ohjelmistoja ovat

- Informatica PowerCenter
- IBM InfoSphere
- SAP BusinessObjects
- Oracle OWB ja ODI
- SAS Data Management (Randall, L. & Thoo, E. 2016.)

Kuviossa 2 esitetään analyysi markkinoilla olevista ETL-työkalujen toimittajista ja niiden ominaisuuksien suhteista. Kuvio on jaettu neljän ominaisuuden mukaan, haastajat, johtajat, kapea markkinasektori ja visionäärit. Kuvion mukaan erityisesti Informatica ja IBM ovat johtavassa asemassa ETL-työkalujen toimittajina (Randall, L. & Thoo, E. 2016).

Kuvio 2. ETL-työkalujen vertailu ominaisuuksien mukaan (Randall, L. & Thoo, E. 2016).

ETL-prosessit pystytään automatisoimaan helposti ja käynnistämään ne ajastetusti aikoina, jolloin muiden järjestelmien käyttö on vähäistä maksimoidakseen niiden suorituskyky. Automatisointi nopeuttaa ja helpottaa ajoja käsityönä tehtyihin ajoihin verrattuna ja vähentää virhemahdollisuuksia (Hovi ym. 2009. 16).

2.4 Tietovarastoinnin hyödyt

Jotta tietovarastoinnin suunnittelu, toteutus ja ylläpito olisivat kannattavaa, on tietovarastoinnin tuotettava selkeää hyötyä yritykselle. Yleinen ongelma suuremmissa yrityksissä on tiedon siiloutuminen eri yksiköiden tai osastojen välillä. Tämä on seuraus tehokkuuteen pyrkivästä työnjaosta, jossa kukin osasto ajaa omaa asiaansa ja hankkii itselle parhaiten sopivat järjestelmät ja ohjelmistot, eikä huomiota kiinnitetä tarpeeksi kokonaisuuteen. (Hovi ym. 2009, XI.) Tietovarastoinnissa eri lähteissä olevat tiedot yhdistetään eli integroidaan yhteiseen tietokantaan. Näin osastokohtaisesta tiedoista saadaan yritystason tietoja. (Hovi ym. 2009, 15.) Tietovarastoinnin avulla voidaan tärkeimpiä tunnuslukuja ja johdettuja tietoja laskea valmiiksi. Nämä muodostavat yhden oikean totuuden (single version of the truth), eli kaikilla yrityksen sisällä on saatavilla oikeat tiedot luotettavasti ja varmasti aina kun niitä tarvitaan. (Roberts 2007, 17.)

Tietovarastoissa säilytetään myös historiatietoja, joka avaa mahdollisuuden monipuolisempiin analyyseihin esimerkiksi kehitystä tarkastellessa. Historiatiedoista voidaan muodostaa organisaation muisti, eli sen tietopääomaa jota on hyödyllistä säilyttää tulevaisuutta varten. Operatiivisissa järjestelmissä saatavilla on vain tämänhetkinen tieto, esimerkiksi pankkitilin saldosta. Tietovarastosta voidaan etsiä tilin saldo haluttuna menneenä ajankohtana. (Machado ym. 2014, 171–172.)

Operatiivisten järjestelmien käyttö suoraan tietojen analysointiin tai raportointiin ei ole järkevää, koska niitä ei ole suunniteltu siihen. Tietovarastoa käytettäessä raportointiin, operatiivisten järjestelmien kuormitus laskee ja samalla suorituskyky paranee. Tietovaraston käyttö mahdollistaa myös operatiivisten järjestelmistä vanhojen tietojen poiston, joka parantaa järjestelmien suorituskykyä. (Hovi ym. 2009, 14–15.)

Tietovarastot eivät ole riippuvaisia liiketoiminnallisten järjestelmien muutoksista. Vaikka operatiivisia järjestelmiä vaihdettaisiin, pystytään vanhan tietovaraston käyttöä jatkamaan rakentamalla uudesta järjestelmästä samanlainen rajapinta ETL-prosesseihin kuin

vanhastakin. Tämä mahdollistaa tietojen käytön jatkumisen saumattomasti. (WatchWise 2011.)

3 KETTERÄT MENETELMÄT

3.1 Määritelmä

Ketterät menetelmät termillä tarkoitetaan viime vuosikymmeninä kehittynyttä ohjelmistokehityksen suuntausta, jossa tavoitteena parantaa laatua, joustavuutta sekä liiketaloudellista hyötyä. Näiden menetelmien tarkoituksena on välttää historian nojalla todetut ohjelmistokehityksen haasteet ja ongelmat, kuten budjetin ylittyminen, huonolaatuiset lopputuotteet ja myöhästyneet aikataulut. (Cooke 2012.)

Ketterien menetelmien lähtökohtana voidaan pitää ketterää julistusta, joka tehtiin vuonna 2001. Tällöin joukko eri ketterien menetelmien kehittäjiä kokoontui keskustelemaan ketterien menetelmien käyttäjien kanssa kehitysmenetelmien parantamisesta. Kokoontumisen pohjalta syntyi ensimmäinen dokumentoitu yhteenveto ketteristä menetelmistä. Julistuksen sisällön mukaan menetelmissä arvostetaan

- Yksilöitä ja kanssakäymisiä enemmän kuin menetelmiä ja työkaluja
- Toimivaa ohjelmistoa enemmän kuin kattavaa dokumentaatiota
- Asiakasyhteistyötä enemmän kuin sopimusneuvotteluja
- Vastaamista muutokset enemmän kuin pitäytymistä suunnitelmassa

Julistuksessa kuitenkin korostetaan, ettei jälkimmäisiä tule unohtaa, vaikka edellä mainittua asioita arvostetaan enemmän. (Agile Manifesto 2001.)

Historiallisesti ketterät menetelmät mielletään sopimaan parhaiten ohjelmistokehitykseen, mutta niitä voidaan käyttää monipuolisesti myös ohjelmistosta riippumattomissa kehityskohteissa. Projektikontekstissa ketterät menetelmät perustuvat nopeisiin iteraatioihin, kasvavaan ja kehittyvään kehitysrytmiin. Tarkoituksena on luoda toiminnoista toimivia palasia lyhyissä iteraatioissa. Tämä on yksi suurin eroavaisuus aikaisemmin käytetystä vesiputousmallista. Vesiputousmallia käsitellään tarkemmin myöhemmin. Ketterän kehityksen toisena perustana ovat löyhät vaatimusmäärittelyt ennen kehityksen aloittamista. Määrittelyitä tarkennetaan ja muokataan kehittämisen ohella ja tämän ansiosta ongelmiin ja puutteisiin pystytään vastaamaan joustavasti ilman suuria takapakkeja. (Davis & Radford 2014, 127–128.)

Ketterät menetelmät tähtäävät tuottamaan jatkuvasti liiketoiminnallista arvoa fokusoimalla työntekijöitä työskentelemään korkeimmin priorisoitujen toimintojen parissa ja tuottamaan täysin toimivaa, täysin testattua ja tuotantovalmista tuotetta. Ketterissä menetelmissä tulee painottaa kommunikaatiota liiketoiminnan ja projektiryhmän jäsenten välillä, koska sen on katsottu parantavan lopputuotteen käytettävyyttä, laatua ja lopputulokseen tyytyväisyyttä. (Cooke 2012, 29–30.)

3.2 Menetelmät

Eri ketteriä menetelmiä on nykyään kymmeniä. Ne kaikki sisältävät kuitenkin samat perustavoitteet. Näihin kuuluvat esimerkiksi etukäteissuunnittelusta siirtyminen jatkuvaan suunnitteluun. Jatkuvassa suunnittelussa uutta informaatiota käytetään hyväksi läpi projektin. Tarkoituksena ei ole luopua kokonaan etukäteissuunnittelusta vaan pyrkiä tekemään se laadukkaasti ja tämän jälkeen jatkuvasti varmistaa yhteneväisyys toteutuksen kanssa. Teknisten riskien huomioonottaminen tehdään mahdollisimman aikaisessa vaiheessa prosessia, jotta niiden aiheuttamat kustannukset ja aikataulun venytykset voidaan minimoida.

Ketteriä menetelmiä käytetään laaja-alaisesti sovelluskehityksestä sovellusten ylläpitostrategioihin. Tunnetuimpia ketteriä menetelmiä ovat Scrum, Extreme Programming (XP), Rational Unified Process (RUP) ja Lean Software Development (LSD). Taulukossa 1 on esitetty yleisempien ketterien menetelmien eroavaisuuksia.

	Scrum	XP	RUP	LSD
Tärkeimmät ominaisuudet	Tarkka seuranta, päivittäiset palaverit	Testivetoinen kehitys	Käyttötapaukset ja UML-mallit	Perustuu Lean-ajatteluun
Yleisyys	50 %	6 %	-	3 %
Iteraation pituus	1-4 viikkoa	1-2 viikkoa	2-6 viikkoa	Ei määritelty
Tiimin koko henkilöinä	< 10	7-12	3-100	5-6
Kehityssyklit	Monta lyhyttä iteraatiota	Monta lyhyttä iteraatiota	Monta lyhyttä tai muutamia pidempiä iteraatiota	Ei määritelty
Muodollisuudet	Kevyt dokumentointi, vähän vaiheita	Kevyt dokumentointi, vähän vaiheita	Kattava dokumentaatio	-

Taulukko 1. Yleisimpien ketterien menetelmien vertailu (Collaborative Consulting 2015).

Scrum ja XP ovat käytetyimmät ketterät menetelmät ja ne ovat ominaisuuksiltaan hyvin lähellä toisiaan. XP sisältää kuitenkin monia ominaisuuksia, jotka on kehitetty ohjelmistokehitystä silmälläpitäen. Yksi tärkeimmistä on testivetoinen kehitys, jossa ennen ohjelmakoodin kirjoittamista tehdään testitapaukset valmiiksi. Sen tavoite on tehdä ohjelmakoodista helposti testattavaa. (Leffingwell 2011, 178–179.)

3.3 Scrum

Scrum on ketterien menetelmien yleisin ja käytetyin viitekehys. Se oli alun perin suunnattu ohjelmistonkehitysprojektien hallintaan, mutta se toimii hyvin myös yleisesti projektinhallinnassa. Scrum juuret ovat peräisin 1980-luvun lopusta Japanista, missä ensimmäiset viittauksen Scrumiin tehtiin Hirotaka Takeuchin ja Ikujiro Nonakan toimesta. (Schwaber & Sutherland 2013.)

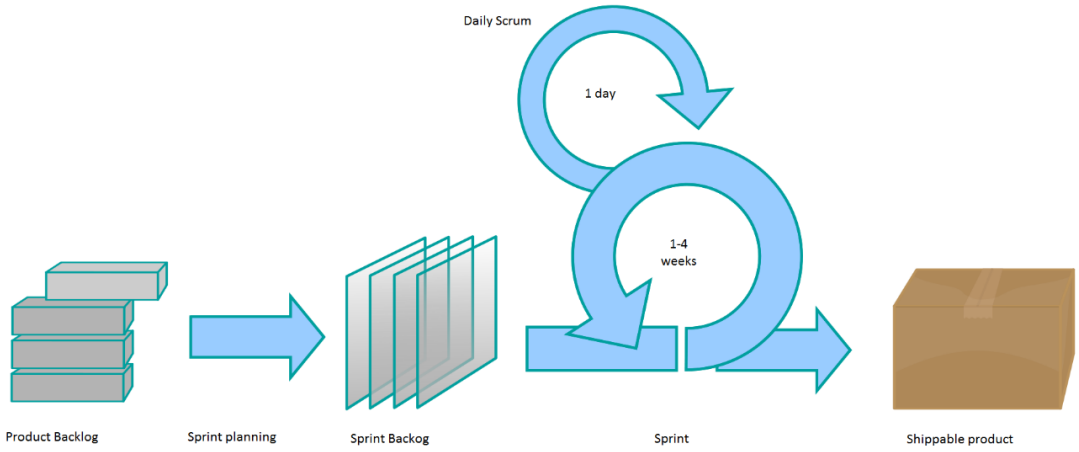
Scrum ei anna valmiita ohjeita, miten kehitysprojekteja tulisi hallita, vaan päätöksenteko jätetään kehitystiimeille. Scrumin yksi perusta on itseohjautuvat tiimit. Tiimeille annetaan mahdollisuus päättää itse miten työt hoidetaan ja roolit jaetaan, eikä tiimeissä ole varsinaista tiiminjohtajaa. Tämä kannustaa vastuullisempaan työskentelyyn ja sen on katsottu parantavan tiimin jäsenten motivaatiota ja luovuutta. (Agile Alliance 2016.)

Scrum suosii työskentelyä pienissä tiimeissä, suositeltu koko on 5-8 henkilöä. Tiimit koostuvat tuotteen omistajasta (Product Owner), kehittäjistä ja Scrum-mestarista (Scrum Master). Scrum-mestarin tehtävä on auttaa tiimiä pääsemään tavoitteisiinsa ja pääsemään mahdollisten ongelmien tai työtä hidastavien esteiden yli mahdollisimman nopeasti. Hänen tavoitteena on pyrkiä optimoimaan tiimin työteho ja vastata, että työskentely toteutuu Scrum-viitekehyksen mukaisesti. Scrum-mestarin velvollisuuksiin kuuluu myös päivittäisen Scrum-palaverin johtaminen. (Hughes 2013, 43–44.)

Scrum-mestarilla on iso rooli ketterien menetelmien onnistumiseksi tietovarastoinnissa. On tärkeää, että Scrum-mestari on ollut mukana tietovarastointiprojekteissa ja että hänellä on käsitys tietovarastoinnin monitasoisesta arkkitehtuurista sekä ymmärrys mahdollisesta suuresta tietomäärästä aiheutuvista ongelmista. Scrum-mestari pystyy usein työskentelemään kehittäjän tehtävässä, sillä hänen työtehtävät eivät vaadi kokopäiväistä panosta. (Hughes 2013, 20–21.)

Tuotteen omistaja vastaa tuotteen kehitysjonon (Product Backlog) sisällöstä ja päättää missä järjestyksessä työt tulisi tehdä. Tuotteen omistajan vastuulla on tuotteen ominaisuudet ja loppukädessä projektin onnistuminen. Hänen pitää olla perillä pyrähdyksen etenemisestä ja saada tietää mahdollisesta viivästymisistä. Tuotteen omistajan täytyy pystyä antamaan tiimille selkeä visio tuotteista ja antaa selkeät vastaukset kysymyksiin, mitä tehdään ja miksi. Asiakasvaatimusten kerääminen ja esittäminen kehitystiimille kuuluu hänen tehtäviinsä. Sen vuoksi hänen tulisi olla tiimin jäsenten tavoitettavissa aina kun he sitä tarvitsevat. (Viscardi 2013, 19.)

Scrumin iteraatiot ovat pituudeltaan normaalisti 1-4 viikkoa ja niitä kutsutaan pyrähdykseksi (Sprint). Kuvassa 2 esitetään perinteistä Scrum-prosessia. Pyrähdyksen ovat siinä keskeisessä osassa. Pyrähdyksen päätyttyä uusi alkaa välittömästi, jolloin kehitys on jatkuvaa. Tiimit saavat itse päättää pyrähdyksen sopivan pituuden, mutta on tärkeää ettei pituus ylitä 4 viikkoa.



Kuva 1. Esimerkki perinteisestä Scrum-prosessista (Leffingwell 2011, 15).

Jokainen pyrähdys alkaa suunnittelulla (Sprint Planning) jossa päätetään mitkä toiminnot on tarkoitus tehdä pyrähdyksen aikana. Kuten ketterissä menetelmissä yleisestikin, on tavoitteena pyrähdyksen päätteeksi olla valmiita julkaisukelpoisia toimintoja. Toiminnot valitaan tuotteen kehitysjonosta, johon tuotteen omistaja on ne valinnut ja priorisoinut. (Greene & Stellman 2014, 89–91.)

Pyrähdyksen suunnitteluun pituudesta Scrumissa pidetään ohjenuorana, 4-viikon pyrähdyksessä enintään 8 tuntia ja sitä lyhemmissä enintään 4 tuntia. Tiimien toimintatapojen

vaihtelun vuoksi suunnitteluiden pituudet saattavat vaihdella huomattavasti. Pyrähdyksen suunnittelussa tuotteen omistajan tehtävänä on kertoa tuoteluettelon tärkeimmät tavoitteet ja esittää niistä havainnollistavat suunnitelmat ja mahdolliset mallit. Tiimin tulee arvioida pyrähdyksen suunnittelun aikana, paljonko tavoitteista saadaan toteutettua seuraavan pyrähdyksen aikana. Samalla tiimin jäsenet sopivat keskenään tärkeimmäksi priorisoitujen töiden tekemisestä omien osaamisalueiden ja halujen pohjalta. (Viscardi 2013, 20.)

Päivittäisen palaverin (Daily Scrum) ideana on käydä läpi lyhyesti pyrähdyksen tilanne. Palaverin pituus tulee pitää lyhyenä, enintään 15 minuutissa. Palaverissa kukin tiimin jäsen vastaa seuraaviin kysymyksiin

- Mitä teit edellisen päivän aikana?
- Mitä aiot tehdä tämän päivän aikana?
- Mitkä tekijät estävät sinua saavuttamasta pyrähdyksen tavoitteita?

Palaveri järjestetään nimensä mukaisesti joka päivä ja mielellään aina samaan aikaan. Jokaisen tiimin jäsenen tulee osallistua palavereihin, myös tuotteen omistajan ja Scrummestarin. Päivittäisissä palavereissa puhevuorot pyritään pitämään lyhyinä ja niissä pyritään välttämään ongelmiin ratkaisujen löytämistä, vaan kyse on niiden esille tuomisesta. (Viscardi 2013, 20.) Jos ratkaisujen löytäminen vaatii lisää keskustelua, tulee asianomaisten sopia uusi palaveri mahdollisimman nopeasti ratkaistakseen ongelma. Näin muiden tiimin jäsenten aikaa ei kulu turhaan palavereissa. (Greene & Stellman 2014, 90.)

Pyrähdyksen päätteeksi järjestetään pyrähdyksen katselmointi (Sprint Review). Katselmoinnissa tiimi esittää pyrähdyksen tavoitteet ja valmiit toiminnot käyttäjille ja sidosryhmille. Vain täysin valmiit ja tuotteen omistajan hyväksymät toiminnot voidaan esitellä. Käyttäjät ja sidosryhmä voivat esittää kysymyksiä ja antaa palautetta esitetyistä toiminnoista. Mahdolliset muutospyynnöt tulee kirjata työluetteloon tuotteen omistajan toimesta, jotta ne voidaan sisällyttää uusiin pyrähdyksiin. (Greene & Stellman 2014, 90–91.) Koska katselmoinnissa esitetään vain valmiita tuotteita, ei katselmointiin valmistautumiseen kuuluisi mennä tiimeiltä ylimääräistä aikaa. Katselmoinnissa pyritään näyttämään tuote toiminnassa, ennemmin kuin kiillotettuja PowerPoint-esityksiä.

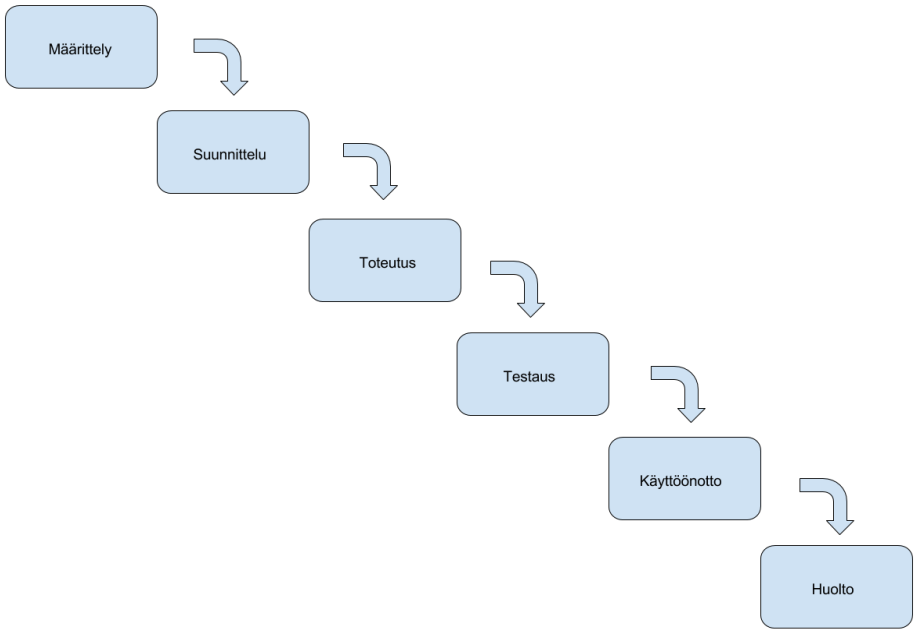
Viimeinen vaihe Scrum-prosessissa, ennen uuden pyrähdyksen aloittamista, on pyrähdyksen retrospektiivi (Sprint Retrospective). Retrospektiivissä tarkoituksena on kehittää

jatkuvasti Scrum-tiimin toimintaa. Koko tiimi osallistuu keskusteluun mikä toimii ja mikä ei toimi käytetyissä toimintatavoissa. Tarkoituksena on analysoida pyrähdyksen aikana ilmenneitä puutteita tai kehityskohteita. Ne voivat liittyä esimerkiksi kommunikaatioon, ympäristöön tai käytettyihin työkaluihin. Retrospektiivin lopuksi päätetään tehdäänkö toimintatapoihin muutoksia seuraavaan pyrähdykseen. (Kenneth 2012, 375–376.)

3.4 Vesiputousmalli

Ennen ketteriä menetelmiä yleisin projektin- ja ohjelmistonkehitysmalli oli vesiputousmalli (Waterfall). Alun perin rakennus- ja teollisuusprojekteja varten kehitetty menetelmä vakiintui 1950-luvulta alkaen myös ohjelmistoprojekteihin, sen yksinkertaisuuden ja loogisuuden takia (Davis & Radford 2014,145). Vesiputousmallissa kehitys on jaettu selkeisiin työvaiheisiin, joita on lähteistä riippuen viidestä seitsemään. Kuvassa 1 on esitetty kuusivaiheinen malli. Yleisimmät vaiheet ovat

- Määrittely
- Suunnittelu
- Toteutus
- Testaus
- Käyttöönotto
- Huolto



Kuva 2. Vesiputousmalli kuusivaiheisen mallin mukaan (Leffingwell 2011, 30).

Vesiputousmallissa oleellista on, että jokainen vaihe tulee olla suoritettu valmiiksi ennen kuin seuraavaan voidaan siirtyä. Se helpottaa kehityksen seurantaa ja hallintaa, mutta viivästyttää kehityksen aikataulua huomattavasti. Vesiputousmallin mukaisessa kehityksessä korostetaan dokumentaation tärkeyttä ja jokainen vaihe dokumentoidaan kattavasti. Jopa vesiputousmallin kehittäjän pidetty Winston Royce on todennut, ettei mallin käyttö sovi suuriin ohjelmistonkehitysprojekteihin. (Leffingwell 2011, 7–9.)

Mallissa on useita ongelmakohtia nykypäivän kehitystä ajatellen ja niitä tarkastellaan seuraavaksi tarkemmin.

3.5 Ketterien menetelmien edut

Ketterien menetelmät tuovat usein ratkaisun vesiputousmallin yleisimpiin ongelmiin. Vesiputousmallin mukaisessa toteutuksessa julkaisuilla on tapana kestää kauemmin ja yhä kauemmin. Ketterässä kehityksessä jokaisen iteraation päätteeksi lisätään uusia ominaisuuksia pala palalta. Kehittäminen voidaan lopettaa silloin kun katsotaan, että maksimaalinen hyöty on saavutettu. Tutkimusten mukaan on osoitettu että jopa puolet kehitettyjen ohjelmistojen ominaisuuksista on käytössä harvoin tai ei koskaan. (Schwaber &

Sutherland 2012, 3-4.) Tämän lisäksi vesiputousmallia käytettäessä aikatauluissa pysytään harvoin, kun taas ketterässä kehityksessä julkaisuajankohta ei voi myöhästyä kuin maksimissaan kuukauden, sillä se on yhden iteraation suositeltu maksimipituus. Tärkeimmät ominaisuudet kehitetään ensin, joka takaa sen että tärkeimmät toiminnot saadaan nopeasti käyttöön. (Kannan 2011.)

Vesiputousmallissa tuotteen stabilointi ja testaus kehityksen lopulla vievät yhä kauemmin aikaa. Ketterässä kehityksessä stabilointiin tarvittava aika ei kasva, sillä edelliseen iteraatioon tarvittava stabilointi ja testaus on tehty jo valmiiksi ja vain uudet ominaisuudet tulee testata.

Vesiputousmallissa suunnitteluun menee liikaa aikaa ja se ei pidä paikkaansa projektin edetessä. Ketterän kehityksessä suunniteluun käytetään alussa keskimäärin vain noin 20 prosenttia siitä ajasta mitä vesiputousmallissa. Määrittelyiden tekemistä jatketaan toteutusvaiheen ohessa, tärkeintä on päästä nopeasti aloittamaan itse toteutus. Ketterissä menetelmissä ei pidetä virheitä suurena paheena, kuten vesiputousmallissa, kunhan virheet huomataan aikaisessa vaiheessa ja ne korjataan nopeasti. Vesiputousmallissa on riskinä virheiden huomaaminen liian myöhään, mikä johtaa pahimmillaan kokonaan epäonnistuneeseen projektiin. (Hughes 2013, 16–17.)

3.6 Ketterät menetelmät tietovarastoinnissa

Vaikka ketterien menetelmien käyttö on yleistynyt valtavasti viimeisen kymmenen vuoden aikana perinteisissä ohjelmistoprojekteissa, tietovarastointi- ja business intelligence projekteissa käyttöönotto on ollut huomattavasti varovaisempaa. Yksi syy on tietovarastointiprosessin monivaiheisuus. Yhtä tietovarastointiprojektia voidaan verrata usean yksinkertaisen sovellusprojektin toimittamiseen kerralla. (Hughes 2013, 3.)

Tietovarastointiprojekteissa ei pystytä niiden luonteen takia useinkaan käyttämään suoraan ketteriä menetelmiä, koska niitä ei ole suunniteltu tietovarastoinnin asettamia haasteita silmälläpitäen. Tietovarastoinnissa vaaditaan laajaa sovellusmäärää ja tämän takia projektin jäsenten osaaminen on painottunut helposti johonkin sovelluksiin tai työtehtäviin, eikä kaikkia tehtäviä voida jakaa tasaisesti. Tietovarastoinnissa usein toistettavat tehtävät voivat pakottaa yhden tai useamman tiimin jäsenen toimettomaan tilaan kun toiset voivat olla ylikuormitettuja. (Hughes 2013, 251–252.)

Kuten tyypillisessä Scrum-tiimissä, ei tietovarastoinnissa voida jakaa tiiminjäseniä satunnaisesti. Ralph Hughesin mukaan ylimääräinen roolitus on välttämätöntä. Scrum-viitekehystä käyttäessä voidaan määrittää neljä ylimääräistä roolia helpottamaan työnjakoa. Ylimääräiset roolit ovat

- Projektiarkkitehti
- Data-arkkitehti
- Järjestelmäanalyytikko
- Järjestelmätestaaja

Projektiarkkitehdin tulee ymmärtää tarvittavat vaatimukset, liiketoiminnallinen tarkoitus työlle sekä hahmottaa niitä vastaava tekninen ratkaisu. Data-arkkitehtia tarvitaan muuntamaan luonnokset ratkaisuista loogisiksi- ja fyysisiksi malleiksi. Järjestelmäanalyytikon tehtävä on määrittää korkean- ja keskitason käsittelysäännöt sekä tietojen yhdistämisessä käytettäviin muunnoksiin tarvittavat lisätiedot. Järjestelmätestaajaa tarvitaan toteuttamaan jatkuvaa testausta ja kertomaan kehittäjille testauksessa havaitsemistaan virheistä. (Hughes 2013, 255–256.)

Ketterät menetelmät sopivat hyvin projekteihin jotka ovat arvaamattomia (Kannan 2011). Tietovarastointiprojektit sopivat tähän kuvaukseen hyvin. Arvaamattomuuden lisäksi tietovarastointiprojekteilla on tapana epäonnistua jopa 50 % todennäköisyydellä. (Collier 2011, 19–20.)

Seuraavat syyt kannustavat ketterien menetelmien käyttöön tietovarastoinnissa

- Tarvittavia resursseja voidaan lisätä projektin edetessä
- Liiketoiminnan edustajat hyötyvät nopeasta kehityksestä silloin kun vaatimusten määrittely on vielä kesken
- Tietovarastointiprojekteja pidetään usein jatkuvina
- Liiketoiminnan edustajat eivät ole täysin tietoisia tiedon luonteesta ennen kuin se on nähtävillä kohdejärjestelmässä (Kannan 2011.)

3.7 Ketterien menetelmien haasteet

Kuten kaikissa kehitysmenetelmissä, myös ketterien menetelmien käyttöönotossa tietovarastoinnissa on omat riskinsä. Ketterien menetelmien metodologiaa ei tule ajatella uskontona eikä yrittää seurata määrityksiä orjallisesti. Jokaisessa organisaatiossa tarvitsee

tehdä pientä räätälöintiä jopa tiimien välillä saadakseen kaikki mahdollinen hyöty irti ketteristä menetelmistä. Isoissa yrityksissä ei ole järkevintä lähteä käyttöönottamaan ketteriä menetelmiä ensimmäiseksi tietovarastoinnissa, vaan kokemuksia niistä kannattaa kerätä perinteisillä ohjelmistoprojekteilla, jos mahdollista. (Collaborative Consulting 2015.)

Ketterien menetelmien lyhyet iteraatiot, nopea suunnittelu ja määrittely voivat tuoda ongelmia erityisesti ETL-prosessin toteuttamisessa. Jos tietomallit tai tiedon käsittelysäännöt eivät ole täysin valmiita ennen toteutusvaiheen aloittamista, luo se suuren riskin turhan työn tekemiselle, joka johtaa pahimmillaan epäonnistuneeseen toteutukseen. Tämän takia on tärkeää saada tietomallit ja käsittelysäännöt tehtyä heti iteraation alussa tai jopa edellisen iteraation aikana, jotta ETL-toteutus päästään aloittamaan mahdollisimman nopeasti riittävillä määrityksillä. (Collaborative Consulting 2015.)

ETL-prosessin toteuttaminen yhden iteraation aikana voi tuottaa vaikeuksia monissa tapauksissa. ETL-prosessissa vaadittavia useita työvaiheita ja lopulta tiedon esittämistä vaatimusten mukaisesti ei ole aina mahdollista toteuttaa kokonaisuudessaan yhden iteraation aikana. Tämän vuoksi jos tietovarastoinnissa käsitellään suuria tietolähteitä, voidaan ETL-prosessit jakaa vaiheiden perusteella usean iteraation aikana tehtäväksi. Esimerkiksi ensimmäisessä iteraatiossa suoritetaan tiedon tuonti ja puhdistus ja seuraavassa muuntaminen ja latausvaihe. (SegueTech 2015.)

Ketteriä menetelmiä käytettäessä tiimin sisäinen kommunikaatio on erityisen tärkeässä asemassa. Tiimien sijoittumisella on suuri vaikutus ketterien menetelmien käytön onnistumiseen kommunikaation osalta. Tutkimuksen (Ambler 2008) mukaan samassa tilassa työskentelevien tiimien onnistumisprosentti on 80 %, mutta jos yksi tai useampi tiimin jäsen ei pysty liittymään muuhun tiimiin ilman matkustamista, laskee onnistumisprosentti 60 %:iin. Tämä tarkoittaa epäonnistumisen riskin tuplaantumista (Hughes 2013, 107). Pelkästään samoissa tiloissa työskentely ei kuitenkaan suoraan takaa onnistumista, vaan se vaatii panostusta yhdessä työskentelyyn tiimin sisältä. Parhaat kokemukset ketteristä menetelmistä saadaan, kun tiimit työskentelevät lähekkäin ja mielellään samassa työtilassa, jolloin jatkuva kommunikaatio on mahdollista. (Collier 2012, 44–45.)

Toimiakseen tehokkaasti, ketterissä menetelmissä tulee välttää usein toistuvia manuaalisia toimintoja. Mahdollisuuksien mukaan automatisointia tulisi lisätä niin kehitysvaiheessa, testauksessa ja käyttöönotossa. Manuaalisesti suoritettu työ ei ole pelkästään hidasta, vaan se suurentaa virheiden mahdollisuuksia. (Electric Cloud 2011.) Tietovarastoinnissa monivaiheisten toimintojen automatisointi ei ole helppoa. ETL-työkalut ovat

helpottaneet yhden vaiheen toteuttamista, mutta esimerkiksi testauksen automatisointiin tulisi kiinnittää erityistä huomiota. Tietovarastoinnissa testaus tulee suorittaa useilla kymmenillä testi-tiedostoilla, jotta lopputulos on analysointi- ja raportointitarkoituksiin riittävän luotettavaa. Manuaalisesti suoritettuna niin laajan aineiston testaaminen on hidasta. Iteraation lopussa suoritettava testausvaihe on altis aikataulun myöhästymiselle ja tästä syystä laajan testauksen tekeminen manuaalisesti ei aina onnistu. (Hughes 2013, 288–292.)

4 KETTERIEN MENETELMIEN HYÖDYNTÄMINEN TOIMEKSIANTAJAYRITYKSESSÄ

4.1 Toimeksiantajayrityksen ketterät menetelmät

Opinnäytetyön toimeksiantajayrityksessä tietovarastoinnilla on tärkeä rooli sen toiminnan kannalta. Ketteriä menetelmiä on käytetty yrityksen tietovarastoinnissa 2015 syksystä alkaen. Sitä ennen ketteriä menetelmiä on alettu käyttämään yrityksen muissa IT-yksiköissä 2013 vuoden lopulla. Toimeksiantajayrityksen tietovarastoinnin kehitystyössä käyttämät ketterät menetelmät pohjautuvat Scrum-viitekehykseen, mutta niitä on muokattu sopimaan paremmin yrityksen omiin tarpeisiin.

Tietovarastointiyksikössä työntekijät on jaettu viiteen eri Scrum-tiimiin. Jokaisella tiimillä on oma lähtökohtainen vastuualue liiketoiminnan yksiköiden puolelta ja tiimien tuotteen omistajat ovat lähtöisin kyseisistä yksiköistä. Tarvittaessa töitä voidaan jakaa eri tiimien välillä, vaikka ne olisivatkin lähtöisin eri liiketoiminnan yksiköstä. Töiden jakamisen lisäksi tiimien jäseniä voidaan jakaa yli tiimien rajojen. Esimerkiksi tietovarastoinnin suunnittelussa tarvittavia arkkitehteja ei ole jokaiselle tiimille omaa, vaan arkkitehdit toimivat jaetusti tiimien välillä.

Tietovarastoinnissa pyrähdysten pituudeksi on vakiintunut 2 viikkoa. Jos pyrähdyksen ajalle osuu esimerkiksi useita arkipyhiä, voidaan pyrähdyksen pituutta pidentää viikolla. Pyrähdysten lisäksi yrityksessä on käytössä julkaisut (release). Yhden julkaisun pituus on normaalisti 7 viikkoa. Julkaisu koostuu kolmesta pyrähdyksestä ja yhdestä suunnitteluviikosta. Suunnitteluviikon aikana suoritetaan tuoteluettelon priorisointi. Kehittäjät voivat käyttää suunnitteluviikot keskeneräisten töiden viimeistelyyn, havaittujen virheiden korjaamiseen tai mahdollisiin kouluttautumisiin. Julkaisujen käyttö mahdollistaa pidemmän tähtäimen suunnittelun ja mahdollisuuden suurempien kokonaisuuksien hallittuun julkaisuun.

Yrityksessä käytetään Atlassianin kehittämää JIRA-tehtävänhallintajärjestelmää Scrum-työskentelyn tukena suunnittelussa, tehtävien listaamisessa ja työn etenemisen seuraamisessa. Jokainen tuotelistalla oleva työpyyntö on lisätty JIRAan. Pyynnöistä kerrotaan

lyhyesti sisältö ja tärkeimmät ominaisuudet. Tuotteen omistaja pystyy seuraamaan JIRAsta pyrähdyksen etenemistä ja tavoitteiden täyttymistä. Jokaisesta työstä on tehty työmääräarvio helpottamaan töiden jakamista pyrähdyksen suunnittelussa.

Työpyyntöjä voi tulla keneltä tahansa yrityksen sisältä ja kaikki pyynnöt käsitellään yhtäläisesti. Kaikki työpyynnöt kulkevat tuotteen omistajan kautta kehitysjonoon. Tuotteen omistajien yläpuolella toimii Chief Product Owner (CPO), joka vastaa koko yksikön tavoitteiden saavuttamisesta. CPO vastaa lopullisesta tuotelistan priorisoinnista (Kenneth 2012, 184). Tietovarastoinnin yhtenäisestä suuntauksesta ja liiketaloudellisen strategian noudattamisesta vastaa ohjausryhmä (Steering Group).

4.2 Prosessin aloitus

Scrum-viitekehyksen perusteella prosessi alkoi pyrähdyksen suunnittelulla, jossa määrättiin tehtävät työt. Minulle valikoitui työlistasta datan integrointiin liittyvä toteutusvaihe ETL-prosessia käyttäen. Työhön vaadittava suunnittelutyö ja käsittelysäännöt oli tehty aiemmissa pyrähdyksissä, joten toteutusvaihe päästiin aloittamaan nopeasti. Pyrähdyksen suunnitteluun osallistui kaikki tiimin kehittäjät ja Scrum-mestari. Toteutusvaihe oli tarkoitus tehdä valmiiksi yhden pyrähdyksen aikana.

4.3 ETL-prosessin työkalut ja suunnittelu

Datan integrointi toteutettiin käyttämällä IBM InfoSphere DataStage ETL-työkalua, joka on yksi markkinoiden johtavimmista ETL-työkaluista (Randall, L. & Thoo, E. 2016). DataStage koostuu kolmesta eri sovelluksesta, joita ovat Administrator, Director ja Designer. Tässä työssä pääpaino oli Designer-sovelluksen käytössä, joka on tarkoitettu latausten suunnitteluun ja toteutukseen graafista käyttöliittymää käyttäen. Director on latausten valvomiseen ja ajastamiseen käytetty sovellus. Administratoria käytetään DataStage-ympäristön ylläpitoon. (Alur ym. 2008, 16.)

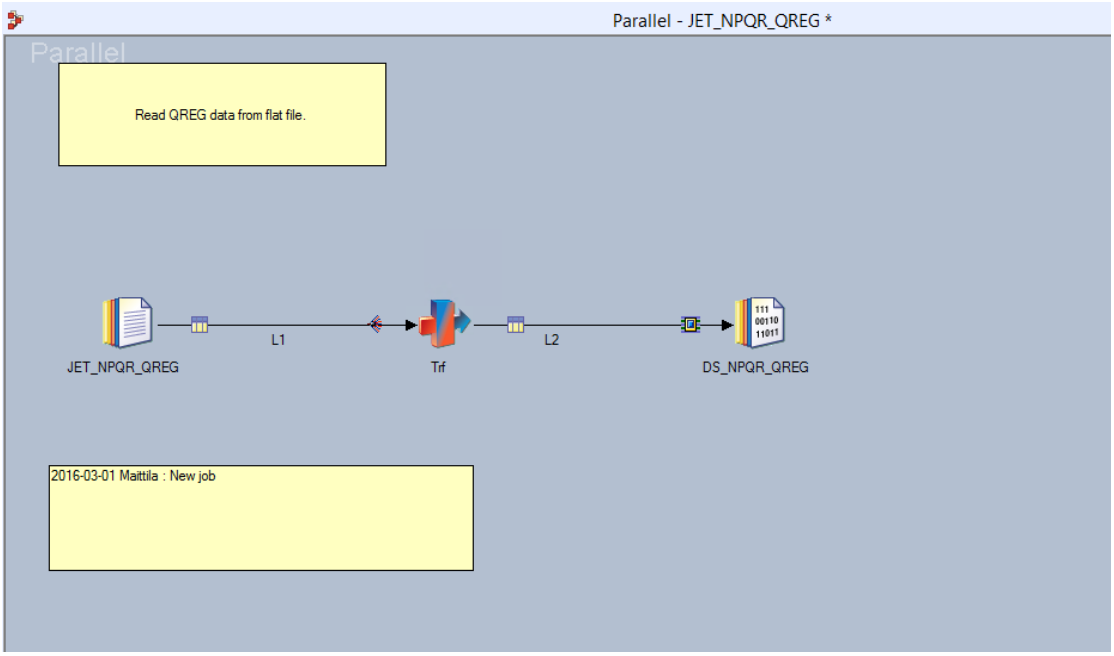
Toteutusvaiheessa tarkoituksena oli integroida operatiivisesta järjestelmästä saatavat peräkkäistiedostot (flat file) tietovarastoon. Peräkkäistiedostot tuodaan tekstimuodossa ja jokainen tietue on niissä omalla rivillään. Käsittelysäännöissä oli määrätty että tiedot muokataan tietovarastointikäyttöön sopivaan muotoon jakamalla tietue seitsemäksi kentäksi ja lisäämällä historiointia varten tarvittavat alku-ja loppupäivä sekä päivitysaika.

Yleensä operatiivisissa järjestelmissä on merkitty tieto rivin muuttujasta ja muutospäivästä, joka mahdollistaa vain muuttuneiden tietojen latauksen (Hovi ym. 2009, 57). Koska operatiivisessa järjestelmän tiedoissa ei ollut historiointia varten tarvittavaa päivitysaikaa, jouduttiin tietoja ladattaessa vertaamaan niitä aiemmin ladattuihin tietoihin ja tekemään tarvittavat versiointipäivitykset sen pohjalta.

ETL-prosessi suunniteltiin tehtäväksi yhden pyrähdyksen aikana kokonaisuudessaan, eli lopputuloksena oli tarkoitus olla toimiva ja testattu tietovarastointi lataus. Lataus oli tarkoitus ajaa automatisoidusti viisi kertaa viikossa yöaikaan. Latauksen ajoittamisessa täytyi ottaa huomioon oikea ajojärjestys. Operatiivisesta järjestelmästä tehtävä peräkkäistiedoston tuonti täytyi olla suoritettu, ennen kuin ETL-ajoa voidaan suorittaa. Väärä ajojärjestys johtaa virhetilanteisiin ja tietovaraston päivitys epäonnistuu.

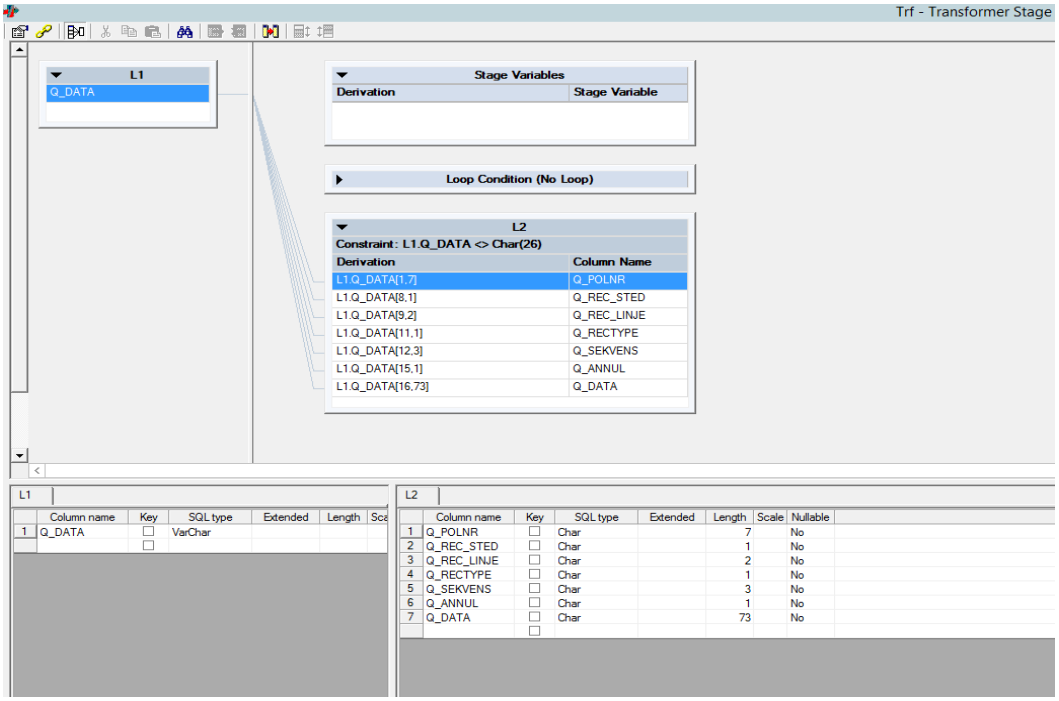
4.4 ETL-ajot

ETL-prosessin jokaisesta vaiheesta luotiin omat rinnakkaisajot (paraller job). Ajojen nimeämisessä käytettiin sovittua käytäntöä helpottaakseen muiden työskentelyä samojen ajojen kanssa. Kaikki rinnakkaisajot alkavat J-kirjaimella (job). Tietojen poiminta ja muunnos vaiheet on yhdistetty yhdeksi ajoksi, joka on nimetty JET_NPQR_QREG. Ajo koostuu kolmesta DataStagen-vaiheesta. Kuvassa 3 esitetään prosessin ensimmäinen vaihe DataStage-ohjelmistosta katsottuna. JET_NPQR_QREG-vaiheessa luetaan operatiivisesta järjestelmästä saatavaa tekstitiedostoa. Tiedot viedään Trf-nimiseen vaiheeseen jossa suoritetaan tarvittavat muunnokset. Peräkkäistiedostossa yksittäinen tietue on yhdellä omalla rivillä ja yhden rivin tiedot luetaan yhdeksi kentäksi.



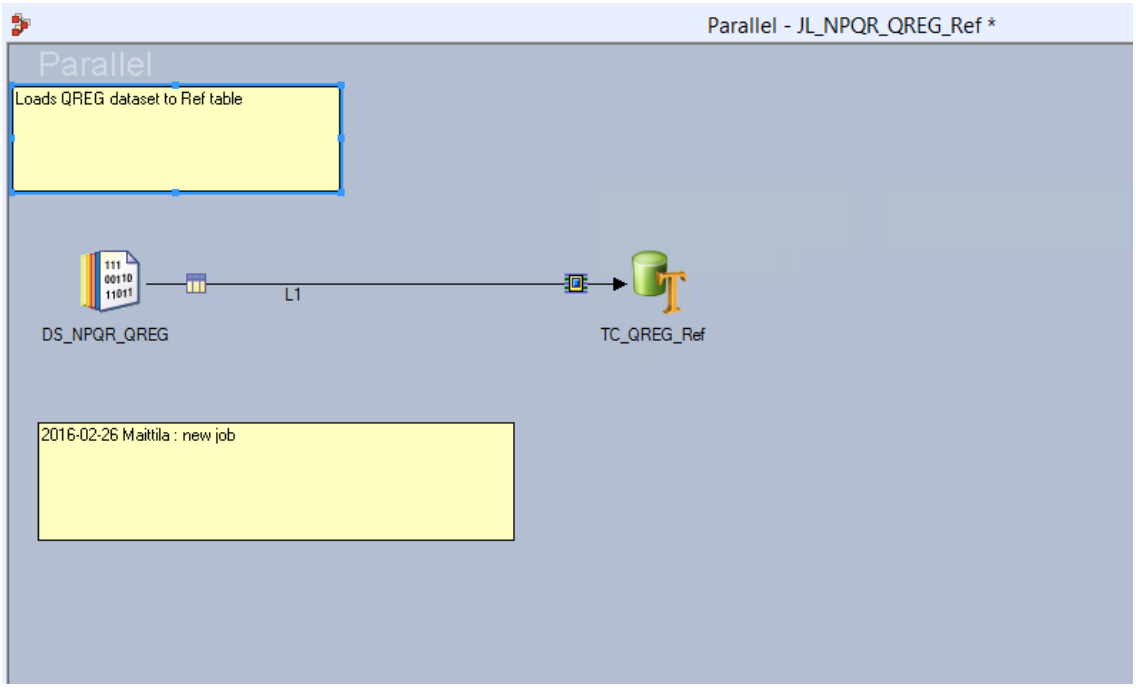
Kuva 3. JET_NPQR_QREG-rinnakkaisajo.

Kuvan 4 muunnosvaiheessa tietue jaetaan seitsemäksi määrämittaiseksi kentäksi. Koska kenttien arvot ovat aina yhtä pitkiä, ei ylimääräisiä käsittelyjä tai funktiota tarvitse käyttää. Muunnosvaiheen jälkeen tiedot siirretään datasettiin odottamaan jatkokäsittelyä.



Kuva 3. JET_NPQR_QREG-ajon muunnos-vaihe.

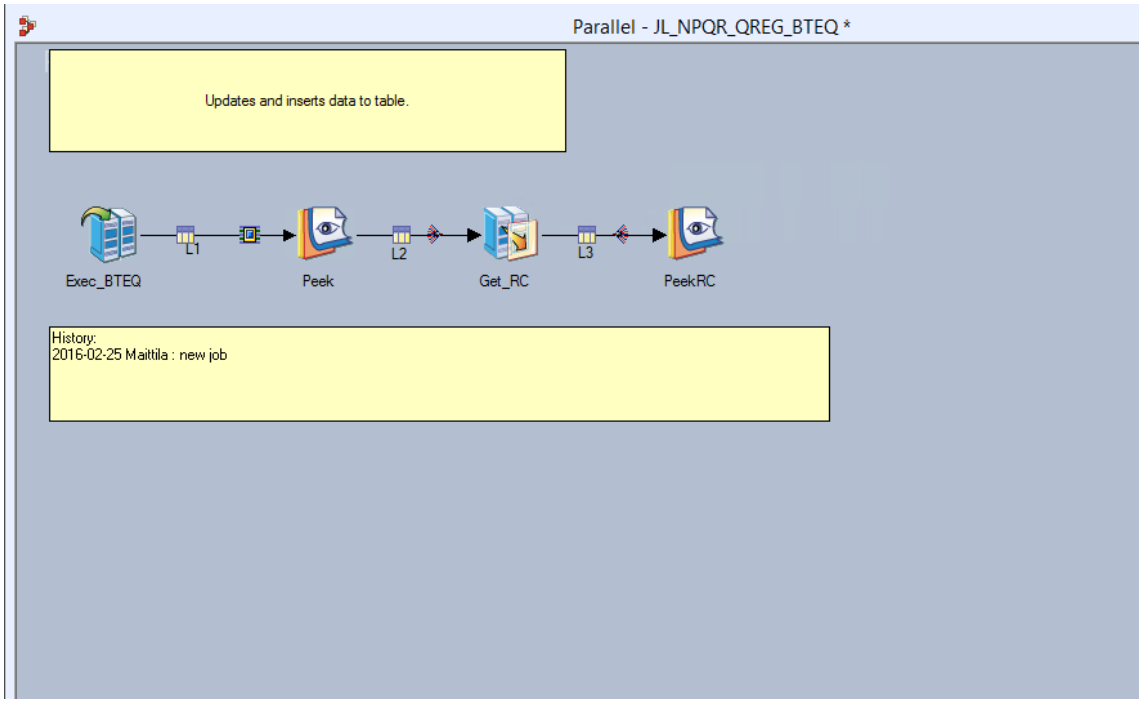
JL_NPQR_QREG_Ref-ajossa ladataan DS_NPQR_QREG-datasetti tietovaraston tietokannan vertailua varten luotuun tauluun. Kuvassa 5 näkyy tietokannan yhteyden luonti ja tietojen lataus tietokantaan ilman SQL-lauseen kirjoittamista.



Kuva 4. JL_NPQR_QREG_Ref-rinnakkaisajo.

Koska operatiivisen tietokannasta saatavissa tietueissa ei ole merkitty päivämäärätietoja, jouduttiin versiointi toteuttamaan manuaalisesti vertaamalla jokaista tietueen kenttää aiemmin ladattuihin. Aina kun tieto muuttuu, lisätään uusi rivi ja samalla vanha rivi päätetään. Jos tieto poistuu kokonaan lähdejärjestelmästä, päätetään rivi asettamalla voimassaoloaika sen hetkiseen aikaan. Vertailutaulu tyhjennetään aina ennen uusien tietojen lataamista, tällöin vain tuoreimmat tiedot ovat taulussa valmiina vertailua varten.

ETL-prosessin viimeisessä vaiheessa JL_NPQR_QREG_BTEQ kuvassa 6 ladataan tiedot tietovaraston lopulliseen kohdetauluun. Vaiheessa suoritetaan myös tiedon historiointia varten vertailu tietojen välillä. Ajossa suoritetaan manuaalisesti kirjoitettu SQL-koodi, joka tutkii löytyykö vertailutauluun ladattuja tietoja lopullisesta taulusta. Jos tietoja ei löydy, lisätään uudet kentät versiointia varten, Txn_Fm_Tms, Txn_To_Tms ja Insrt_Tms.



Kuva 5. JL_NPQR_QREG_BTEQ-rinnakkaisajo.

ETL-toteutuksen jälkeen valmista ajoa testattiin ajamalla manuaalisesti testidataa. Testauksessa verrattiin tietovaraston rivejä operatiivisen järjestelmän vastaaviin käyttämällä satunnaisotantaa. Versioinnin testausta varten jouduttiin odottamaan seuraavana päivänä saatua uutta testidataa, jotta pystyttiin toteamaan muuttuneiden rivien päättäminen ja uusien lisääminen.

4.5 Ketterien menetelmien käyttö ETL-toteutuksessa

Kokemukset ketteristä menetelmistä tietovarastoinnissa ovat lähtökohtaisesti positiivisia. Työskentely oli mahdollista aloittaa nopeasti, sillä tarvittavat käsittelysäännöt ja suunnitelmat oli tehty jo aiemmissa pyrähdyksissä siihen vaiheeseen, että toteutusvaihe pystyttiin aloittamaan heti. Ketteriin menetelmiin kuuluen, käsittelysääntöjä jouduttiin muokkaamaan kesken ETL-prosessin toteutusvaiheen, kun huomattiin että peräkkäistiedostojen rakenne ei vastaa oletettua (Hughes 2015, 90). Ketteryyden ansiosta tarvittavat muutokset pystyttiin toteuttamaan nopeasti ja työtä päästiin jatkamaan ilman suurempia viivästyksiä.

Ketterissä menetelmissä käytettävä kevyt dokumentointi mahdollisti tehokkaan ajankäytön itse toteutustyöhön ja se sopi hyvin ETL-vaiheen tekemiseen. Valmiista latauksesta

luotiin kevyt dokumentaatio Confluence-nimiseen organisaatiowikiohjelmistoon. Confluence helpottaa yhteisen dokumentaation ylläpitoa ja nopeuttaa työssä tarvittavien dokumentaation löytämistä (Atlassian 2016).

ETL-vaiheen toteutuksen aikana suoritettiin jatkuvaa testausta. Näin mahdolliset virhetilanteet löydettiin jo mahdollisimman aikaisessa vaiheessa. Testausvaiheet suoritettiin manuaalisesti ajamalla jokaista ETL-prosessin vaihetta läpi. Lopullinen hyväksymistestaus suoritettiin pyrähdyksen lopussa erillisen testaajan toimesta. Testausmenetelmiä ei ollut automatisoitu.

4.6 Kokemukset ketteristä menetelmistä yleisesti

Kokemukset yrityksessä käytetyistä ketteristä menetelmistä pohjautuvat toteutetun ETL-prosessin lisäksi kolmen kuukauden ajan osallistumisesta erinäisiin tietovarastoinnin tehtäviin.

Päivittäisiä Scrum-kokouksia ei pidetty, vaan kokoukset järjestettiin kolme kertaa viikossa, maanantaisin, keskiviikkoisin ja perjantaisin. Jokaisen pyrähdyksen alussa pidettiin pyrähdyksen suunnittelu. Suunnitteluun oli varattu aikaa tunti, eikä koko päivää kuten Scrum-viitekehyksessä on yleensä tapana. Suunnittelussa käytiin tiimin kesken läpi priorisoidut tehtävät ja jaettiin ne tiimin jäsenten kesken heidän omien halujen ja kykyjen mukaan.

Jokaisen julkaisun alussa käytiin yhdessä tietovarastointiyksikön kesken läpi julkaisun suunnitelma. Suunnitelmasta kävi ilmi mitkä ovat tärkeimmät tehtävät julkaisun sisällä ja mitä tehtäviä muut tiimit tekevät. Pyrähdysten aikana jokainen tiimi työskentelee tiiviisti omien töiden kanssa, eikä muiden tiimien tekemisistä tiedetä välttämättä mitään. Julkaisun suunnitelman läpikäynti auttoi ymmärtämään paremmin mitä muut tiimit tekevät. Läpinäkyvyys tiimien välillä vähentää mahdollisia ongelmatilanteita, jossa työskennellään tietämättä muiden tiimien töiden kanssa ristiin.

Koska Scrum-tiimin jäsenet työskentelivät useassa eri toimipisteessä ja maassa, oli välttämätöntä pitää kokoukset verkkopalavereina käyttäen Microsoft Skype-ohjelmistoa. Kokouksille oli sovittu kiinteät ajat. Samassa toimipisteessä työskentelevät henkilöt olivat sijoittuneet mahdollisuuksien mukaan suunnittelupalavereiden ajaksi samaan kokoustilaan. Kokouksien aikatauluista pidettiin hyvin kiinni ja ne päästiin aloittamaan aina ajallaan.

Scrum-tiimien jäsenten jakautuminen usean maan välillä loi omat haasteensa parhaan mahdollisen työtehon saavuttamiseksi. Vaikka yrityksessä oli panostettu laadukkaisiin kommunikaatiovälineisiin ja tietoliikenneyhteyksiin tehdäkseen yhteyksien pidon tiimin jäseniin helpoksi Skype-ohjelmistoa käyttäen, ei se pysty korvaamaan fyysisen tapaamisen tuomia etuja ja lisäarvoa (Hughes 2012, 108). Tiimin jäsenten välillä oli enintään tunnin aikavyöhyke-erotus, joten se ei aiheuttanut ylimääräisiä järjestelyä tai pitkiä viivästyksiä. Sähköpostin lähettäminen tai Skype-palaverin sopiminen vei kuitenkin enemmän aikaa kuin kasvokkain hoidettu keskustelu. Tämän takia jokaiselle kehittäjälle pyrittiin antamaan useita töitä pyrähdyksen ajaksi, jolloin odotusajan voi käyttää toisen työn tekemiseen.

Työskentelyn aikana tuotteen omistaja ei osallistunut päivittäisiin palavereihin ja vain satunnaisesti suunnitelupalavereihin. Koska tuotteen omistaja ei aina osallistunut suunnittelupalavereihin, ei tiimin jäsenille kerrottu pyrähdyksen tavoitteista eikä priorisoituja tuotelistaa esitelty tarkemmin. Tiimin jäsenet eivät voineet myöskään esittää mahdollisia kysymyksiä koskien töitä. Vasta vähän aikaa tuotteen omistajana toimineelle henkilölle on yleistä, ettei vuorovaikutus tuotteen omistajan ja tiimin välillä ole tarpeeksi suurta (Kenneth 2012, 170–171).

Scrum-viitekehykseen kuuluvat pyrähdysten lopuksi pidettävät retrospektiivit eivät kuuluneet toimeksiantajayrityksen toimintatapoihin. Tiimin jäsenet eivät päässeet esittämään pyrähdysten lopuksi mielipiteitään toimintavoista tai ideoita miten toimintaa voitaisiin kehittää. Scrumin yksi tavoitteista, jatkuva kehitys ja kommunikointi ei tässä tapauksessa toteutunut. Retrospektiiviä pidetään yhtenä Scrum-viitekehyksen aliarvostetuimmista, mutta tärkeimmistä toimintatavoista (Kenneth 2012, 375).

4.7 Kehitysideat

Osallistuminen toimeksiantajayrityksen tietovarastointiin kehittäjän roolissa ja tutustuminen ketterien menetelmien teoriaan nosti esille mahdollisia kehityskohteita joiden avulla toimintoja voitaisiin tehostaa.

Ensimmäinen kehitysidea on tuotteen omistajan panoksen nostaminen tiimin työskentelyssä. Tuotteen omistajalla on tärkeä rooli edesauttaa tiimin onnistumista ja ymmärrystä pyrähdysten tavoitteista. Tuotteen omistajan tulisi osallistua vähintään jokaiseen pyräh-

dyksen suunnitteluun ja mielellään myös päivittäisiin palavereihin. Tiimien yhteenkuuluvuuden tunne paranisi jos tuotteen omistaja osallistuisi aktiivisemmin päivittäiseen toimintaan. Nykyisellä toiminnalla tuotteen omistajan rooli jäi eteiseksi kehittäjätiimistä.

Toinen kehitysidea on retrospektiivien ottaminen käyttöön. Retrospektiivien pitäminen edesauttaisi jatkuvaa kehittymistä. Varsinkin vasta lyhyen aikaa ketteriä menetelmiä käyttäneiden tiimien kohdalla toimintatapojen kehittäminen ja epäkohtien esille tuominen on erityisen tärkeää. Pelkästään kehityskohteiden esilletuonti ei riitä, vaan muutoksia täytyy lähteä toteuttamaan. Muuten ongelmana on tiimin turhautuminen turhia retrospektiiviä kohtaan. (Kenneth 2012, 393–394.)

Testausvaiheen automatisointi on yksi kehityskohde, jota toimeksiantajayrityksen kannattaa miettiä. Automatisoitu testaaminen nopeuttaa testaamista ja tekee siitä luotettavampaa. Nopeutunut testaaminen mahdollistaisi useamman valmiin tuotteen toimittamisen pyrähdyksen aikana, eikä testausvaihetta välttämättä tarvitsisi siirtää seuraavaan pyrähdykseen. Ilman automatisoitua testaamista on mahdoton toimia täysin ketterästi. (Hughes 2012, 293–295).

5 LOPUKSI

Opinnäytteen tavoitteena oli selvittää miten toimeksiantajayrityksessä käytetään ketteriä menetelmiä osana tietovarastointia ja voidaanko käytettyjä menetelmiä parantaa. Työlle asetetut tavoitteet täytettiin ja työn ohessa luotiin ETL-lataus, jota käytetään osana toimeksiantajayrityksen tietovarastointia. Toiminnallisen tutkimuksen avulla saatiin käsitys miten yrityksessä käytetään ketteriä menetelmiä ja miten ne soveltuvat juuri tietovarastointiin.

Toimeksiantajayrityksessä työskentelyn aikana saatiin hyvä käsitys Scrumin käytöstä kehittäjän näkökulmasta. Tutkimuksen aikana kävi selväksi, että Scrum-viitekehys sopivat hyvin myös tietovarastoinnin toteuttamiseen. Yrityksessä oli otettu huomioon mahdollisia riskitekijöitä ja sen takia puhtaaseen Scrumin käyttöön ei edes pyritty.

Työn tuloksena löydettiin kehitysideoita toimeksiantajan käyttämistä ketteristä menetelmistä. Kehityskohteita olivat tuotteen omistajan aktiivisempi osallistuminen tiimin toimintaan, retrospektiivien käyttöönotto ja testauksen automatisointi.

Datan integroinnin toteuttamisesta minulle oli ehtinyt kertyä jonkin verran kokemusta ennen opinnäytetyön aloittamista, mutta työn teoriaosuutta tehdessäni kirjallisuuteen perehtyminen auttoi ymmärtämään toteutusvaihetta syvemmin. Ketterien menetelmien teoria ei ollut minulle entuudestaan kovin tuttua, vaan aiempi tietämys oli kehittynyt työskentelemällä osana Scrum-tiimiä. Ketterien menetelmien- ja Scrum-kirjallisuuteen perehtyminen auttoi ymmärtämään paremmin työskentelytapoja ja mahdollisti uusien näkökulmien esilletuomisen toimintatapoja arvioidessa.

Työn haasteeksi osoittautui raportointia tehdessäni aiheen laajuus. Aiheen rajaus olisi pitänyt tehdä tarkemmin, jolloin asioihin olisi voitu perehtyä paremmin. Nyt valittu aiheen rajaus johti osittain asioiden pintapuoliseen käsittelyyn. Jos opinnäytetyö aloitettaisiin nyt uudelleen, rajaisin aiheen käsittelemään tarkemmin ketteriä menetelmiä ja mahdollisesti vain jotain pienempää osa-aluetta niistä. Lisäksi tapaustutkimuksen sijasta opinnäytetyön empiirinen osuuden pohjana voisi pitää haastatteluja, jolloin saataisiin laajempi käsitys miten ketterien menetelmien käyttöönotto on onnistunut ja kehitysideoita saataisiin laajemmalta käyttäjäkunnalta.

Jatkokehityskohteena on kehitysideoiden pohjalta tehtävien muutosten suunnittelu ja mahdollinen toteutus. Olisi mielenkiintoista nähdä auttaisivatko kehitysideat parantamaan työskentelyä ja tehostamaan ketteryyttä entisestään toimeksiantajayrityksessä.

Työssä esitettyä teoriaosuutta voidaan sen yleispätevyyden ansiosta käyttää tietovarastoinnin ketterien menetelmien käyttöönoton suunnittelun ja käyttöönottoprojektien tukena. Koska ketterien menetelmien toimintatavat ovat yritys- ja tapauskohtaisia, ei tutkimusosassa tehtyjä päätelmiä voida pitää yleispätevinä.

LÄHTEET

Agile Alliance 2016. Self-organizing Scrum teams - Challenges and Strategies. Viitattu 1.5.2016 https://agilealliance.org/wp-content/uploads/2016/01/Self-organizing-Scrum-teams-Challenges-and-Strategies.pdf

Agile Manifesto 2001. Manifesto for Agile Software Development. Viitattu 26.4.2016 http://agile-manifesto.org

Alur, N.; Takashasi, C.; Toratani, S. & Vasconcelos, D. 2008. Viitattu 10.5.2016 http://www.red-books.ibm.com/redbooks/pdfs/sg247576.pdf

Ambysoft 2008. Agile Adoption Rate Survey Results: February 2008. Viitattu 10.5.2016 http://www.ambysoft.com/surveys/agileFebruary2008.html

Atlassian 2016. Confluence Features. Viitattu 10.5.2016 https://www.atlassian.com/software/con-fluence/features

Collaborative Consulting 2015. Best Practices for Applying Agile Techniques to Data Warehouses. Viitattu 24.4.2016 http://www.collaborative.com/wp-content/uploads/2015/06/best-practices-for-applying-agile-techniques-to-data-warehouses-WP.IM_.130.pdf

Collier, K. 2012. Agile Analytics – A value-driven approach to business intelligence and data warehousing. Boston: Addison-Wesley.

Cooke, J.L. 2012. Everything you want to know about Agile. Cambridge: IT Governance Publishing.

Davis, B. & Radford, D 2014. Going Beyond the Waterfall. Plantation: J. Ross Publishing.

Greene, J. & Stellman, A. 2014. Learning Agile. Sebastopol: O'Reilly Media.

Electronic Cloud 2011. Five Reasons why Agile Won't Scale Without Automation. Viitattu 10.5.2016 https://help.rallydev.com/sites/default/files/multimedia/EC-WP_5Reasons-Agile.pdf

Hovi, A. Hervonen, H. & Koistinen, H. 2009. Tietovarastot ja business intelligence. Jyväskylä: WSOY.

Hughes, R. 2013. Agile Data Warehousing Project Management Business Intelligence Systems Using Scrum. Waltham: Morgan Kaufmann.

IBM 2016. Data set stage. Viitattu 10.5.2016 https://www.ibm.com/support/knowledgecen-ter/SSZJPZ_11.5.0/com.ibm.swg.im.iis.ds.parjob.dev.doc/topics/c_deeref_Data_Set_Stage.html

Kannan, K. 2011. Agile Methodology for Data Warehouse and Data Integration Projects. Informatica Professional Services Viitattu 3.3.2016 https://www.informatica.com/down-loads/1749_Agile_Methodology_wp_web.pdf

Kennethm S.R. 2012. Essential Scrum – A practical guide to the most popular agile process. Boston: Addison-Wesley.

Lucidchart 2015. What is UML? Viitattu 10.5.2016 https://www.lucidchart.com/pages/what-is-UML-unified-modeling-language

Machado, C. & Davim, JP. 2014. Transfer and Management of Knowledge. Somerset: Wiley.

Microsoft 2016. Structured Query Language (SQL). Viitattu 1.5.2016 https://msdn.microsoft.com/en-gb/library/windows/desktop/ms714670(v=vs.85).aspx

Ponniah, P. 2010. Data Warehousing Fundamentals for IT Professionals 2nd edition. Hoboken: Wiley.

Prabhu, S. 2007. Data Mining and Warehousing. Daryaganj: New Age International.

Randall, L. & Thoo, E. 2016. Magic Quadrant for Data Integration Tools. Viitattu 19.4.2016 http://www.gartner.com/doc/reprints?id=1-2KDMO20&ct=150731&st=sb

Roberts, P. 2007. Guide to Project Management. London: Profile Books.

Saaranen-Kauppinen, A. & Puusniekka, A. 2006. Menetelmäopetuksen tietovaranto. Viitattu 6.6.2016 http://www.fsd.uta.fi/menetelmaopetus/kvali/L5_5.html

Schwaber, K. & Sutherland, J. 2012. Software in 30 Days. Hoboken: Wiley.

Schwaber, K. & Sutherland, J. 2013. The Scrum Guide. Viitattu 29.4.2016 http://www.scrumguides.org/docs/scrumguide/v1/Scrum-Guide-US.pdf

SegueTech 2015. Common Problems Experienced When Adopting Agile Development. Viitattu 6.5.2016 http://www.seguetech.com/blog/2015/08/05/Common-Problems-adopting-agile-development

Vaze, S. & Joshi, S. 2010. Computer Fundamentals and RDBMs. Mumbai: Himalaya Publishing House.

Vilkka, H. & Airaksinen, T. 2003. Toiminnallinen opinnäytetyö. Helsinki: Tammi.

Viscardi, S. 2013. Professional ScrumMaster's Handbook. Olton: Packt Publishing Ltd.

WatchWice 2011. Why Data Warehousing? Viitattu 12.5.2016 http://www.watchwise.net/data-warehousing.htm