Scrum-viitekehyksen hyvien käytäntöjen selvittäminen

Sebastian Hämäläinen

2020 Laurea

Laurea-ammattikorkeakoulu

Scrum-viitekehyksen hyvien käytäntöjen selvittäminen

Sebastian Hämäläinen

Tietojenkäsittely

Opinnäytetyö

Huhtikuu, 2020

Laurea-ammattikorkeakoulu **Tiivistelmä**
Tietojenkäsittely
Tradenomi (AMK)

Sebastian Hämäläinen
Scrum-viitekehyksen hyvien käytäntöjen selvittäminen
Vuosi 2020 Sivumäärä 38

Opinnäytetyössä tutkittiin, kuinka valitut asiakasprojektit käyttävät Scrum-viitekehystä työarjessaan. Tavoitteena oli selvittää, kuinka tarkasti kyseiset projektit noudattavat Scrum-viitekehystä. Tavoitteena oli myös selvittää kuinka paljon projektien käytänteet eroavat toisistaan ja löytää mahdolliset syyt.

Opinnäytetyö toteutettiin laadullisena tutkimuksena. Tutkimuksessa hyödynnettiin menetelmänä haastattelemista ja havainnointia, jonka avulla kerättiin tutkimusdataa. Tietoperusta koostui varsinaisesti virallisesta Scrum-oppaasta. Tutkimuksessa selvitettiin, kuinka tarkasti asiakasprojektit noudattavat Scrum-viitekehystä, eroavat toisistaan sekä mitkä ovat Scrumin parhaimpia käytäntöjä.

Tutkimuksen tutkittavista projekteista selvisi, että kokonaisuuden koolla sekä monimutkaisuudella voidaan havaita ja päätellä, kuinka tarkasti Scrumia hyödynnetään. Tutkimuksen tuloksista ilmeni, että mitä monimutkaisempi kokonaisuus on, sitä enemmän Scrum-viitekehyksen sääntöjä noudatetaan. Haastatteluiden avulla selvitettiin, mitkä ovat Scrumin parhaimpia käytäntöjä. Scrumin ehdottomiin hyviin käytäntöihin kuuluvat tämän joustavuus eri ongelmatilanteissa, Scrum prosessin tuoma ryhti ja läpinäkyvyys projektille sekä tiivis yhteistyö osapuolien kesken.

Tutkimuksesta selvisi, että tutkittavat asiakasprojektit noudattivat Scrum-viitekehystä hyvin. Kooltaan pienemmät projektit sovelsivat hieman Scrum-viitekehystä omien tarpeiden mukaan, kuten yhdistämällä Scrum tapahtumia. Laajemmat projektit sen sijaan noudattivat Scrum-viitekehystä erittäin tarkasti.

Asiasanat: scrum, projektinhallinta, ketterä ohjelmistokehitys

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

Sebastian Hämäläinen
Identifying Good Practices from the Scrum Framework
Year 2020 Pages 38

This thesis examined how selected customer projects use the Scrum framework in their daily work. The aim of this Bachelor's thesis was to examine how closely these projects follow the Scrum framework. The intention was also to find out how much the project practices differ from each other and to find possible reasons.

This research applied a qualitative method. The research utilized interviews and observations as methods to collect research data. The theoretical framework mainly consists of the official Scrum guide. This thesis examined how closely customer projects follow the Scrum framework, differ from each other, and what are the best practices of the Scrum framework.

The results show that the size of the project and its objectives can be used to determine how closely the Scrum framework is followed. The more complex the project is, the closer the Scrum framework will be followed. Interviews were used to find out what Scrum's best practices and features are. It was revealed that these were the flexibility of the framework in different problem situations, the structure arising towards processes from Scrum events, and the overall transparency of the project.

The main conclusion is that the examined customer projects followed the Scrum framework well. Smaller projects adapted the Scrum framework slightly to their own needs, such as combining Scrum events. Larger projects, on the other hand, followed the Scrum framework very closely.

Keywords: scrum, project management, agile software development

Sisällys

1 Johdanto

Tässä opinnäytetyössä tutkittiin, kuinka valitut asiakasprojektit käyttävät Scrum-viitekehystä työarjessaan. Tutkimusta varten valikoituivat kolme erillistä asiakasprojektia toimeksiantajan kanssa. Projekteilla oli yhteistä se, että nämä hyödyntävät Scrum-viitekehystä arjessaan. Tutkimuksessa tavoite oli tutkia projektien käytäntöjä ja selvittää, kuinka tarkasti nämä noudattavat Scrum-viitekehystä. Samassa yhteydessä tutkittavia projekteja verrattiin toisiinsa, jotta saataisiin selville mahdollisien erojen syyt sekä Scrum-viitekehyksen vahvuudet.

Tutkimuksessa ensimmäiseksi tutustuttiin lyhyesti tutkittaviin asiakasprojekteihin ja siihen, minkälaisia toteutuksia nämä olivat luonteeltaan. Tämän jälkeen perehdyttiin ohjelmistotuotantoon ja kuinka tämä jakautuu kahteen eri osaan: ohjelmistotuotantoprosessi sekä ohjelmistotuotantomenetelmät. Ohjelmistotuotantomenetelmissä perehdyttiin perinteisiin menetelmiin sekä ketteriin menetelmiin. Menetelmistä kuitenkin perehdyttiin vain vesiputousmalliin sekä Scrum-viitekehykseen, sillä nämä olivat tutkimuksen kannalta oleellisimmat.

Tutkimus toteutettiin laadullisena tutkimuksena. Tietoperustana käytettiin virallista Scrum-opasta sekä tutkimusmenetelmien avulla saatuja tietoja. Tutkimuksessa hyödynnettiin erilaisia tutkimusmenetelmiä, jotka olivat tässä tapauksessa haastatteleminen sekä havainnointi. Haastatteluja suoritettiin projektissa toimiville henkilöille, joko avoimena haastatteluna tai puolistrukturoidulla sähköpostihaastattelulla. Havainnointi toteutettiin kehitystiimin jäsenenä projekteissa työn arjessa sekä haastatteluiden aikana. Tutkimusmenetelmien avulla saatujen tietojen kanssa toteutettiin projektien vertailuanalyysi, jonka avulla saatiin tutkimuksen tulokset.

2 Työn lähtökohdat

Tämän työn lähtökohtana toimii se, että Scrum-viitekehys asettaa tarkat käytännöt ja säännöt, joita pitää noudattaa saadakseen maksimaalisen hyödyn irti Scrum-viitekehyksestä. Nykyisin yhä useammat asiakasprojektit vaihtavat perinteisistä projektinhallinta menetelmistä ketteriin malleihin. Syynä on, että yhä useampi projekti sisältää tavalla tai toisella ohjelmistokehitystä, johon yleisesti kuuluu enemmän tuntemattomia kuin tunnettuja muuttujia. Perinteiset menetelmät tunnetaan näiden epäsoveltuvuudesta tuntemattomien muuttujien kanssa.

Otettuaan käyttöön erilaisia ketterän mallin menetelmiä asiakasprojektit eivät kuitenkaan noudata mallien sääntöjä ja käytäntöjä niin kuin niiden on määritetty. Useimmissa

tapauksissa projektit muuntelevat määritettyjä sääntöjä ja käytäntöjä sopeuttaakseen ne asiakasprojektin organisaation omiin menetelmiin ja käytäntöihin. Projektiin osallistuvat henkilöt eivät useimmissa tapauksissa tee tätä itse, vaan pakote muutoksiin tulee organisaation ylemmiltä tahoilta.

Esimerkkinä käytetään Scrum-viitekehystä, joka on erittäin kovassa suosiossa ympäri maailmaa ja kuuluu myös Suomessa puhutuimpiin ja käytetyimpiin ketterän kehityksen menetelmiin eri organisaatioissa. Scrum-viitekehys, niin kuin muutkin ketterän mallin menetelmät, asettaa omat säännöt ja käytännöt, joita eri organisaatiot ovat menneet muuntelemaan omin päin.

Ensimmäisenä esimerkkinä organisaatio muuttaa Scrum-viitekehyksen päivittäispalaverin olemaan joka toinen päivä säästääkseen laadukasta ja tehokasta työskentelyaikaa. Toisena esimerkkinä asiakasprojektiin allokoitu kehitystiimi ei työskentele päätoimisesti kyseisessä asiakasprojektissa. Yhden projektin sijasta kehitystiimi tai yksittäinen jäsen kehitystiimistä työskentelee monessa eri projektissa samanaikaisesti. Kolmantena ja ehkä tutuimpana esimerkkinä organisaatio on vain valinnut parhaimmin tämän toimintaan sopivat Scrum-viitekehyksen prosessin osat hylätäkseen loput käytännöt, säännöt ja osat. Näin Scrum-viitekehystä käyttävä organisaatio voi sanoa olevansa ketterä ja käyttävänsä mallin parhaita käytäntöjä, vaikka tämä ei ehkä pidäkään täysin paikkaansa. Tämän takia asiakasprojekteihin syntyy merkittäviä eroja, vaikka käyttävät samaa ketterän kehityksen mallia. Tätä teorian suhdetta arkikäytäntöihin olisi erittäin kiinnostava tutkia tarkemmin.

2.1 Työn tarkoitus ja tavoitteet

Opinnäytetyössä tutkittiin toimeksiantajan kolmea eri asiakasprojektia, jotka käyttävät aktiivisesti Scrum-viitekehystä työarjessaan. Tarkoituksena on soveltaa eri tutkimusmenetelmiä tutkiakseen ja selvittääkseen miten tarkasti kyseiset asiakasprojektit noudattavat Scrum-viitekehyksen asettamia sääntöjä ja hyviä käytäntöjä työn arjessa. Jos jokin asiakasprojekti erosi erityisen paljon Scrum-viitekehyksen määrittämistä säännöistä ja käytännöistä, pyrittiin myös selvittämään miksi projekti ei noudattanut niitä tarkasti. Kun Scrum-viitekehystä käyttävä asiakasprojekti ei noudata viitekehyksen sääntöjä tarkasti, projekti suurella todennäköisyydellä on soveltanut ja luonut rinnalle omia käytäntöjä. Näitä omia käytäntöjä ja sääntöjä tutkittiin ja selvitettiin, kuinka hyvin ne soveltuivat Scrum-viitekehyksen rinnalle.

Työn toisena tarkoituksena oli havainnoida ja vertailla asiakasprojekteja. Vertailukohteena olivat projektin eroavaisuudet Scrum-viitekehyksen pohjaan ja asiakasprojektien keskeiset erot ja yhtäläisyydet. Asiakasprojekteja tutkittiin seuraavista näkökulmista: eroavaisuudet ja yhtäläisyydet toimintatavoissa, käyttämissä projektinhallintaohjelmistoissa, käytännöissä sekä säännöissä.

Työn päätavoitteena oli selvittää asiakasprojektien eroihin ja yhtäläisyyksiin vaikuttavat tekijät ja syyt. Tavoitteena oli myös selvittää miten ja kuinka paljon projektit poikkeavat Scrum-viitekehyksen määrittämistä säännöksistä ja käytännöistä. Toisena tavoitteena oli luoda mahdollisia parannusehdotuksia löydettyihin eroavaisuuksiin, mutta vain jos näistä oli ilmennyt silmäänpistäviä heikkouksia.

2.2 Tutkimuskysymykset ja työn rajaus

Tämän opinnäytetyön tutkimuskysymykset ovat:

- Kuinka tarkasti asiakasprojektit noudattavat Scrum-viitekehystä?
- Eroavatko Scrum-viitekehystä käyttävät projektit toisistaan, jos niin kuinka paljon?
- Mitkä ovat Scrumin hyvät käytännöt?

Tutkimuksen aihealue rajoittuu varsinaisesti Scrum-viitekehykseen ja tämän määritettyihin käytäntöihin sekä sääntöihin ja tutkittavien asiakasprojektien tuloksiin. Työssä kuitenkin myös käsitellään lyhyesti perinteisten menetelmien vesiputousmallia sekä ketterien menetelmien lähestymistapoja.

2.3 Työn toimeksiantaja

Tämän työn toimeksiantajana toimi Druid Oy, jatkossa Druid, joka toimii ohjelmistojen suunnittelun ja valmistuksen toimialalla, eli on luonteeltaan aito ohjelmistotalo. Druid on perustettu vuonna 2012 viiden ohjelmistokehittäjän toimesta luodakseen parhaan mahdollisen työpaikan, jossa työskennellään rehellisesti, tinkimättömästi ja ketterästi. Nykyisin Druid työllistää hiukan alle 30 henkilöä, eikä rekrytoimiselle näy loppua. (Druid Oy 2020.)

Druid on alusta pitäen erikoistunut ketterään palvelukehitykseen ja se on vakiinnuttanut selvän aseman suurten palvelukokonaisuuksien toteuttajana ja monipuolisena digialan kumppanina niin julkishallinnon kuin yksityisen sektorin projekteissa. Druid pääasiallisesti rakentaa ja toteuttaa eri tyyppisiä verkkopalveluita, pienistä kotisivuista, laajoihin kokonaisuuksiin, jotka saattavat sisältää integraatioita asiakkaan ulkoisiin järjestelmiin. (Druid Oy 2020.)

Ensisijaisesti Druidilla on käytössä Drupal 8 -sisällönhallintajärjestelmä, jota voidaan laajentaa hyödyntäen erilaisia teknologioita mm. suosituilla JavaScript ohjelmistoviitekehyksillä, joihin lukeutuu esim. ReactJS. Drupal 8 on myös mahdollista laajentaa verkkokauppa-alustaksi hyödyntämällä ilmaista Drupal Commerce -laajennusta. Druidin laajempiin toteutuksiin kuuluvat mm. räätälöidyt verkkokaupat sekä verkkopalvelukokonaisuudet, joissa on monia eri tyyppisiä integraatioita useisiin erilaisiin ulkoisiin järjestelmiin. (Druid Oy 2020.)

2.3.1 Toimeksiantajan tarjoamat palvelut

Druid tarjoaa erilaisia digitaalisia palveluita, joista sen asiakkaat löytävät kaiken tarpeellisen digitaalisien palveluidensa rakentamiseen tai täydentämiseen. Druid auttaa asiakkaitaan muodostamaan idean, konseptin, suunnitelman ja toteutuksen, yhteistyössä asiakkaan tarpeiden mukaan. (Druid Oy 2020.)

Näihin sisältyvät muun muassa verkkopalvelun kipu- ja kehittämiskohteiden kartoittaminen Druidin kehittämällä löytöretki -formaatilla, ketterä verkkopalvelukokonaisuuden suunnitteluprosessi, viimeisimpiä verkkoteknologioita hyödyntävät ohjelmistokehitystoteutukset sekä luotettava asiakasprojektin jatkokehitys varsinaisen julkaisuvaiheen jälkeen, eli ylläpito (Druid Oy 2020.)

2.3.2 Asiakasprojektien toteutustapa

Asiakasprojekteissa hyödynnetään hyvin usein ketteriä kehitysmenetelmiä, joiden ansiosta asiakasprojektit saadaan hyvin suurella prosentilla maaliin. Joukosta löytyy tietenkin myös joitakin epäonnistumisia. Näistä epäonnistumiset eivät ole niin suuria ketterän kehitysmenetelmän ansiosta. Kehitystyö tehdään läheisessä yhteistyössä asiakkaan kanssa, ja projektin ajautuminen väärään suuntaan huomataan lähes välittömästi. Näin siihen pystytään reagoimaan nopeasti eikä vasta loppuvaiheessa.

Tästä syystä Druidin kaikki asiakasprojektit toteutetaan ketterillä menetelmillä. Pääsääntöisenä ketteränä menetelmänä toimii Scrum-viitekehys, jota hyödynnetään laajempien asiakasprojektien yhteydessä. Lyhytkestoisissa (1-2 viikkoa) asiakasprojekteissa ei päästä hyödyntämään Scrum-viitekehyksen tuomia etuja, koska näin lyhyellä ajanjaksolla ei ehdi suorittamaan useita Scrumin käyttämiä sprinttejä eli iteraarioita. Iteraatiot nimenomaan muodostavat ketterien kehitysmenetelmien edut suhteessa perinteisiin menetelmiin. Tämän vuoksi Druid käyttää tämän tyyppisissä projekteissa Kanban-viitekehystä, joka sopii loistavasti lyhyisiinkin asiakasprojekteihin. (Druid Oy 2020.)

3 Tutkittavat asiakasprojektit

Tutkittavat asiakasprojektit valittiin yhdessä toimeksiantajan kanssa. Projektit valikoituivat kahdella eri perusteella: ne ovat jokseenkin erikokoisia kokonaisuuksia ja niiden projekteissa toimivat toimeksiantajan asettamat eri Scrummasterit.

3.1 TC Vehicle Oy

TC Vehicle Oy on vuonna 1982 perustettu osakeyhtiö, joka tarjoaa asiakkaillensa matkailuautojen vuokrauspalvelua. Yritys on vuosien varrella noussut segmentissään johtajaksi ympäri Euroopan franchising-ketjullaan nimeltä Touring Cars. Päämaja sijaitsee Suomessa, mutta Touring Cars -konttoreita löytyy ympäri Eurooppaa 12:sta eri maasta. (TouringCars 2020.)

TC Vehicle Oy:n Touring Cars projekti alkoi toimeksiantajan kanssa vuoden 2018 lopulla. Tarkoitus oli korjata vanhaa verkkopalvelua, mutta todettiin kuitenkin olevan järkevämpää rakentaa kokonaan uusi verkkopalvelu. Lopullinen verkkopalvelu, joka toteutettiin, oli matkailuautojen varausjärjestelmä yhdistettynä verkkosivustoon. Alkuperäisen toteutuksen jälkeen oli myös toteutettu pienempiä ominaisuuksia vuoden mittaan. Tutkimuksessa tutkittiin vuoden 2020 keväällä toteutettua lisäominaisuusprojektia, joka toteutettiin Scrum-viitekehyksen avulla. (Druid Oy 2020.)

3.2 Tamro Oyj

Tamro Oyj on vuonna 1895 perustettu julkinen osakeyhtiö. Tamro toimii lääkkeiden ja terveystuotteiden jakelijana sekä terveysalan palveluntarjoajana. Sen varsinaisiin asiakasryhmiin kuuluvat eri lääkeyritykset, apteekit, sairaalat, yksityinen terveyden huolto sekä yksityishenkilöille tarkoitetut vähittäiskaupat. Yrityksen toimiala on lääketukkukauppa, ja sen päämaja sijaitsee Vantaalla Tamrotalossa. (Tamro 2020.)

Tutkimuksessa tutkittiin Tamron ja toimeksiantajan kanssa toteutettua projektia, joka aloitettiin vuoden 2020 keväällä. Projektin tarkoituksena oli rakentaa Tamrolle uusi verkkosivusto vanhan tilalle.

3.3 Helsingin yliopisto

Helsingin yliopisto tunnetaan Suomen suurimpana sekä vanhimpana tiedekorkeakouluna. Kyseisessä koulussa on yhteensä noin 40 tuhatta opiskelijaa ja työntekijää. (Helsingin yliopisto 2020.)

Helsingin yliopiston projekti on ollut jo vuodesta 2013 asti aktiivisessa kehityksessä. Projektin aikana uudistettiin Helsingin yliopiston konsernisivut, koulutushaku sekä mukautettu, yleiskäyttöinen verkkopalvelualusta. Toimeksiantaja on ollut alusta pitäen mukana projektin toteuttajana. Tutkimuksessa tutkittiin vuoden 2019 lopulla käynnistettyä Scrum-projektia, jonka tarkoituksena oli päivittää järjestelmän versiota uudempaan. (Druid Oy 2020.)

4 Ohjelmistotuotantoprosessi

Ohjelmistotuotanto muodostuu kahdesta eri kokonaisuudesta, jotka ovat ohjelmistotuotantoprosessi ja ohjelmistotuotantomenetelmät. Ennen kuin ohjelmistotuotantomenetelmiä pystyy sisäistämään kunnolla, täytyy ymmärtää, kuinka ohjelmistotuotantoprosessi toimii. Seuraavassa kappaleessa selvitetään, kuinka ohjelmistotuotantoprosessi etenee.

Ohjelmistotuotantoprosesseja on monia erilaisia kuten esimerkiksi web-kehitys, videopeliohjelmointi, mobiiliohjelmointi, työpöytäohjelmistojen kehitys ja data-analytiikka (Hyperiondev 2017). Tässä opinnäytetyössä pureudutaan aiheeseen web-kehityksen näkökulmasta.

Ohjelmistotuotantoprosessin lähtöpisteessä, ennen kuin mitään konkreettista on vielä olemassa, aloitetaan suorittamalla määritysprosessi, jossa määritetään tulevan ohjelmiston vaatimukset ja tavoitteet. Vaatimuksiin sisältyy esimerkiksi kohderyhmä, jonka pohjalta kartoitetaan potentiaalista laitekantaa, joille palvelu tulee toimia moitteitta. Määrittelyvaihetta seuraa suunnitteluprosessi, jossa luodaan ohjelmiston arkkitehtuuria ja mahdollisesti infra-arkkitehtuuria määritysvaiheessa luotujen vaatimusten mukaisesti. Ohjelmistoarkkitehtuuri sisältää muun muassa seuraavia valintoja: ohjelmistossa käytettävät teknologiat (esim. ohjelmointikielet), mahdolliset käytettävät ohjelmistoviitekehykset (esim. Javascript ohjelmointiviitekehys ReactJS ja PHP ohjelmointiviitekehys Symfony) ja muut valmiit ostettavat ulkoiset palvelut, johon kuuluu esimerkiksi verkkoisäntäpalvelut eri palveluntarjoajilta. (Shinde 2019.)

Suunnittelun jälkeen alkaa varsinainen ohjelmistokehitys, jossa luodaan vaatimusten mukainen ohjelmisto suunnitellun arkkitehtuurin mukaisesti. Ohjelmistokehitysvaiheessa luodaan myös tekninen dokumentaatio tulevaa kehitystä varten, joka toimii apuna seuraaville kehittäjille. Ohjelmistokehityksen aikana kaikki yksittäiset toteutetut ominaisuudet käyvät koodinkatselmoinnissa, jonka yleisesti suorittaa asiakasprojektissa oleva toinen kehittäjä tai ulkopuolinen kehittäjä samasta organisaatiosta. Koodikatselmoinnin jälkeen ominaisuus viedään testiympäristölle, jonka jälkeen alkaa varsinainen testausvaihe, ja asiakas pääsee testaamaan kyseistä ominaisuutta.

Testausvaiheessa asiakas testaa testiympäristössä sinne julkaistua ominaisuutta ja tarkastaa vastaako toteutettu ominaisuus sille määriteltyjä vaatimuksia. Jos asiakas mielestänsä havaitsee jotain puutteita, hän ilmoittaa siitä kehittäjälle. Kehittäjä tarkistaa asiakkaan ilmoittamat mahdolliset puutteet ja raportoi oman näkemyksen toteutuksen tilasta. Kehittäjä ilmoittaa asiakkaalle, onko kyseessä todellinen puute vai mahdollinen väärinymmärrys kommunikoinnissa vaatimusten suhteen. Jos kyseessä oli oikea puute, kehittäjä ottaa ominaisuuden nopeasti taas työn alle ja korjaa nämä löydetyt puutteet ja/tai virheet. Korjauksien jälkeen kehittäjä vie muutokset taas testiympäristöön ja ilmoittaa asiakkaalle ominaisuuden olevan valmis uudelleen testattavaksi. Asiakas testaa uudelleen ominaisuutta ja tällä kertaa toteaa

tämän vastaavan määrityksiä ja hyväksyy ominaisuuden. Ominaisuus voidaan nyt merkitä olevan tehty ja valmis.

Ohjelmistotuotantoprosessin viimeisenä varsinaisena vaiheena on ohjelmiston tuotantoon vienti eli kokonaisuuden käyttöönotto. Tässä vaiheessa kaikki kokonaisuutta koskevat ominaisuudet ovat valmiina, käyneet koodikatselmoinnissa sekä ovat läpäisseet asiakkaan tarkastuksen. Kun kaikki on kunnossa, sovitaan päivä, jolloin muutokset viedään asiakkaan tuotantoalustalle. Julkaisun jälkeen alkaa virallisesti ylläpitovaihe, jonka aikana koodia ylläpidetään, korjataan ja mahdollisesti lisätään uusia ominaisuuksia, jolloin ohjelmistotuotantoprosessi käynnistyy uudelleen (kuvio 1). Ylläpito-vaihetta varten yleisesti luodaan uudet allekirjoitetut sopimukset, jossa määritetään miten projektin kehitys ja/tai ylläpito tulee jatkua tulevaisuudessa.

Kuvio 1: Ohjelmistotuotantoprosessi mallikuviona

5 Ohjelmistotuotantomenetelmät

Edellisissä kappaleissa tutustuttiin ohjelmistotuotantoprosessin eri vaiheisiin. Varsinaisena tavoitteena näillä vaiheilla on toimittaa korkealaatuinen tuote, joka vastaa asiakkaan toiveita

ja vaatimuksia. Tämä voi olla osittain haastavaa, koska joissakin tapauksissa toiveet ja vaatimukset voivat olla epäselviä ja/tai osoittautua laajemmiksi, kuin etukäteen osattiin arvioida. Ohjelmistotuotantoprosessin helpottamiseksi on kehitetty erityyppisiä ohjelmistotuotantomenetelmiä, joissa kehitystyö on jaettu osiin. Näitä menetelmiä voidaan kutsua vaihejakomalleiksi.

Suurimpana erona ohjelmistotuotantomenetelmien välillä on niiden lähestyminen eri ohjelmistotuotantoprosessin vaiheita kohtaan, mutta kaikissa menetelmissä kuitenkin säilyvät samat perusvaiheet lähes muuttamattomina. Menetelmien lähestymistapojen eroavaisuuksien vuoksi menetelmät ryhmitellään kahteen eri luokkaan: perinteiset menetelmät ja ketterät menetelmät (Tervonen 2017). Perinteisillä sekä ketterillä menetelmillä on selviä eroja, mutta myös yllättävän paljon yhteneväisyyksiä. Eroavaisuuksiin pureudutaan tarkemmin seuraavissa kappaleissa.

5.1 Perinteiset menetelmät

Vanhimpia eli alkuperäisiä malleja ei ole alun perin tarkoitettu ohjelmistotuotantoprosessin hallintaan vaan ne on sovellettu ohjelmistotuotantoprojekteihin perinteisten tehtaiden toimintamalleista. Perinteisissä menetelmissä on erittäin tyypillistä, että kaikki asiakasprojektin käytännön toiminta suunnitellaan tarkasti ennen aloittamista (Tervonen 2017). Nämä lähestymistavat sopivat erittäin hyvin asiakasprojekteihin, joissa vaatimukset ymmärretään hyvin, esimerkiksi teollisuuden kaltaisilla aloilla, joilla kaikki ymmärtävät lopputuotteen selvästi (KPI Partners 2018).

Perinteiset mallit seuraavat yksisuuntaista eli ns. lineaarista lähestymistapaa. Käytännössä asiakasprojekti alkaa projektin määrittelyvaiheesta, sitten seuraavaksi etenee suunnitteluvaiheeseen, suunnittelun jälkeen ohjelmiston kehitysvaiheeseen, kehityksen jälkeen asiakastestaukseen ja viimeisenä ylläpitoon. Perinteisten menetelmien kaikissa työvaiheissa luodaan erittäin paljon dokumentaatiota. Dokumentaatiolla varmistetaan, että tuotettu ominaisuus on kehitetty vaatimuksen mukaiseksi (Tapio 2010). Kun on saavuttu ylläpitovaiheeseen, asiakasprojektin varsinainen aktiivinen kehitysvaihe on saatettu päätökseen.

Perinteisten mallien suurimpana ongelmana on niiden heti alussa lukkoon lyödyt tarkat vaatimukset ja kokonaisuuden asiakastestaukset vasta loppuvaiheena. Tämä lähestymistapa ei anna mahdollisuutta joustaa tarpeiden mukaan projektin edetessä. Kun muutos tarpeita havaitaan, pitää aloittaa uusi projekti. Perinteiset menetelmät eivät siis sovellu erittäin hyvin ohjelmistotuotantoprojekteihin näiden muuttuvien tekijöiden vuoksi.

Perinteisiä menetelmiä ovat esimerkiksi vesiputousmalli, prototyyppimalli ja spiraalimalli. Prototyyppimallissa toteutetaan nopeasti osa ohjelmistosta ja muokataan tarpeen mukaan, jotta asiakas pääsee näkemään ohjelmistoa mahdollisimman aikaisin. Spiraalimalli sen sijaan

on iteratiivinen malli, joka yhdistää vesiputousmallin, prototyyppimallin ja riskianalyysin (Laine & Paakki 2003). Seuraavassa kappaleessa kuvataan tarkemmin perinteisistä malleista eniten käytettyä eli vesiputousmallia, joka on edelleen käytössä valitettavan useissa projekteissa etenkin julkispuolen hankinnoissa.

Yleisesti, kun puhutaan perinteisistä menetelmistä, viitataan vesiputousmalliin. Tästä syystä tässä tutkimuksessa perehdytään vain kyseiseen perinteiseen menetelmään.

Vesiputousmalli ohjelmistotuotannossa on saanut alkunsa vuonna 1970 kun Walter Royce julkaisi artikkelin suurten ohjelmistojärjestelmien kehittämisestä. Walter Royce oli lisännyt artikkeliin kuvan esittääkseen ohjelmistojärjestelmän kehittämisen ja toimittamisen eri vaiheet, jotka muistuttivat erittäin paljon vesiputousmallin vaiheita. Vesiputousmalli on kuitenkin saanut nimensä myöhemmin vuonna 1976, kun Bell ja Thayer käyttivät sitä julkaisemassa artikkelissaan. (Baca, Petersen & Wohlin 2009; Bell & Thayer 1976; Ismagulov 2017.)

Vesiputousmallin käyttämä rakenne kuitenkin syntyi alun perin teollisuudesta, kun näiden projektit vaativat hyvin strukturoituja sekä vaiheistettuja prosesseja. Vesiputousmalli tuotiin ohjelmistotuotantoon pienin muutoksin ja sitä käytetään paljon vielä tänäkin päivänä. (Tapio 2010.)

Vesiputousmalli käyttää samaa yksisuuntaista lähestymistapaa, kuin muutkin perinteiset menetelmät. Yksisuuntaisessa lähestymistavassa mallin kehitys jaetaan eri osiin eli vaiheisiin. Nämä eri vaiheet aina seuraavat toista ja ovat riippuvaisia toisistaan. Vesiputousmallilla on 7 varsinaista vaihetta, jotka ovat järjestelmävaatimukset, ohjelmistovaatimukset, ohjelmiston määrittely, ohjelmiston suunnittelu, ohjelmiston kehitys, asiakastestaus ja viimeisenä ohjelmiston käyttöönotto (kuvio 2). Vesiputousmallissa korostetaan sitä, että sen looginen eteneminen on toteutettava koko ohjelmistotuotantoprosessin ajan, ylhäältä alas, niin kuin vesiputous. Tästä etenemistyylistä malli on juuri saanut nimensä. (Tapio 2010; Tervonen 2017.)

Kuvio 2: Vesiputousmallin vaiheet mallikuviona

Vesiputousmallin toiminnan ensimmäinen vaihe on järjestelmävaatimuksien selvittäminen. Tätä vaihetta voidaan myös kutsua nimellä esitutkimus, mutta varsinaisesti tutkitaan järjestelmävaatimuksia. Tarkoituksena on kartoittaa erilaisia ratkaisuja ohjelmiston järjestelmälle asiakkaan esiintuoduista ongelmista, tarpeista sekä toivomuksista. Kun järjestelmän vaatimukset on selvitetty, siirrytään seuraavaan vaiheeseen eli ohjelmistovaatimuksiin. Tämän vaiheen tarkoitus on lähinnä sama kuin edellisenkin, mutta järjestelmän sijaan selvitetään ja kartoitetaan ohjelmistolle ratkaisuja. Kaikki mahdolliset vaatimukset dokumentoidaan tässä vaiheessa vaatimusmäärityksasiakirjaan, jotta niihin voidaan referoida seuraavissa vaiheissa. (Free Management eBooks 2015; Tapio 2010; Velez 2016.)

Ohjelmiston määrittelyvaihe käynnistyy, kunhan asiakkaalta on saatu kaikki mahdolliset ongelmat, tarpeet ja toiveet. Tässä vaiheessa referoidaan äskettäin luotuun vaatimusmäärityksasiakirjaan, jota analysoidaan tarkasti. Sen kautta muodostetaan toteutettavan ohjelmiston arkkitehtuuri, jota lopulta aloitetaan työstämään. (Free Management eBooks 2015; Tapio 2010; Velez 2016.)

Kun järjestelmän ja ohjelmiston vaatimukset on määritetty, alkaa varsinainen suunnitteluvaihe. Suunnitteluvaiheessa suunnitellaan toteutettavan ohjelmiston perusarkkitehtuuri tutkimalla edellisten vaiheiden määrittämiä vaatimuksia vaatimusmäärityksasiakirjasta. Suunnitteluvaiheen lopputuloksena syntyy yksi tai useampi tekninen dokumentti toteutettavasta ohjelmistosta, jota varsinaisesti ryhdytään toteuttamaan. (Free Management eBooks 2015; Tapio 2010; Velez 2016.)

Ohjelmiston toteutusvaiheessa kehittäjät ryhtyvät toteuttamaan määritettyjen vaatimuksien mukaista ohjelmistoa. Ohjelmisto toteutetaan ensin pienissä osissa, jonka jälkeen osat eli ominaisuudet, integroidaan yhteen kokonaisuudeksi. Tämän vaiheen lopussa pitäisi olla toimiva ohjelmisto, joka tekee sen, mitä ohjelmiston määrittely vaiheessa hahmoteltiin. Tämän jälkeen siirrytään testausvaiheeseen. (Free Management eBooks 2015; Tapio 2010; Velez 2016.)

Testausvaiheessa asiakas alkaa tutkimaan toteutettua ohjelmistoa. Tässä vaiheessa asiakas varmistaa ohjelmiston toimivan oikein ja olevan vaatimuksien mukainen. Ohjelmiston koosta riippuen testausvaihe voi olla melko pitkä. Asiakkaan löydettyä mahdolliset viat ja virheet ohjelmistossa, ne dokumentoidaan testausraporttiin ja ilmoitetaan toteuttajalle. Virheet täytyy korjata ennen varsinaista ohjelmiston käyttöönottovaihetta. Kun testausvaihe on suoritettu, toteutettu ohjelmisto voidaan ottaa käyttöön asiakkaan järjestelmässä. (Free Management eBooks 2015; Tapio 2010; Velez 2016.)

Vesiputousmallissa löytyy hyviä sekä huonoja puolia riippuen millä alalla mallia käyttää. Ehdoton hyvä puoli mallissa on se, että sitä on erittäin helppo ymmärtää. Mallia käytettäessä ei tule ikinä epäselvyyttä siitä, mikä vaihe on juuri nyt käynnissä. Mallin vaiheissa on erittäin tarkasti määritetty, mitä yksittäisessä vaiheessa tehdään. Yksittäisen vaiheen aikana ei myöskään saa tehdä minkään muun vaiheen työtä, vaan seuraavaan vaiheeseen edetään vasta, kun edellinen on suoritettu. Ennen kuin edetään seuraavaan vaiheeseen, luodaan kuitenkin erittäin tarkka dokumentaatio vaiheessa toteutetusta työstä. Näin voidaan todistaa, että määritetyt vaatimukset täyttyvät. (Ismagulov 2017; Tapio 2010; Tervonen 2017.)

Vesiputousmallin säännöt ovat erittäin tiukat, joka tekee projektinhallinnasta periaatteellisella tasolla erittäin mukavaa ja helppoa. Kun kaikki asiakasprojektiin liittyvät vaatimukset suunnitellaan ja määritellään jo mallin ensimmäisissä vaiheissa, kehittäjät voivat vain alkaa toteuttaa sitä, mikä heille annetaan ohjelmiston kehitysvaiheessa. Malli myös mahdollistaa ohjelmiston vaatimuksien määrityksien muuttamista ja/tai täsmentämistä varhaisissa vaiheissa kehittäjien ja asiakkaiden kesken. Mahdolliset halutut muutokset on mahdollista toteuttaa heti tai pienellä vaivalla, kun ohjelmiston kehitys- tai ohjelmiston käyttöönottovaihetta ei vielä työstetä. (Ismagulov 2017; Powell-Morse 2016; Tapio 2010; Tervonen 2017.)

Vesiputousmalli ei ole valitettavasti tunnettu sen hyvistä puolista, vaan enemmänkin huonoista puolistaan. Mallia on kritisoitu vuodesta 1970 lähtien alkaen Walter Roycen julkaisemasta artikkelista, jossa hän kuvaa mallin suurimpia ongelmia ohjelmistotuotannossa ja kuinka ne olisi mahdollista korjata. Myös Bell ja Thayerin julkaisemassa artikkelissa vuonna 1976 he ilmaisivat vesiputousmallin tarvitsevan parannustarpeita, sillä vaatimusten keräämistä ei koskaan suoriteta oikein projekteissa. (Bell & Thayer 1976; Ismagulov 2017.)

Mallin suurimpia ongelmapuolia on se, että mallin jokainen vaihe on suoritettava loppuun, ennen kuin seuraava vaihe voi alkaa. Näissä vaiheissa ei myöskään voi olla päällekkäisyyksiä. Tämän kaltainen lähestymistapa soveltuu projekteihin hyvin, joissa pystyy ennakoimaan mahdollisia muuttujia projektin aikana tai projektin kulku on erittäin selkeää heti ensimmäisestä vaiheesta lähtien. (Ismagulov 2017; Tapio 2010; Tervonen 2017.)

Mallin kehityksen haittana on, että se ei salli paljoakaan pohdintaa tai tarkistamista ensimmäisten vaiheiden jälkeen. Kun projekti on asiakastestausvaiheessa, on erittäin vaikea palata takaisin ja muuttaa jotain, jota ei ole dokumentoitu hyvin määrittelyvaiheessa. Testausvaihe on myös viivästetty viimeisimpiin vaiheisiin, joka tuottaa ongelmia, kun ongelmat löydetään vasta projektin loppuvaiheilla. Tästä syystä ohjelmistotuotannossa tämän kaltainen lähestymistapa ei toimi. Kehittäessä ohjelmistoa ei voida ikinä varma siitä, mitä muuttujia tulee projektin aikana. Projektin alussa ei voi myöskään määrittää ohjelmiston kokonaisuutta valmiiksi, sillä vaatimukset tarkentuvat projektin aikana. (Ismagulov 2017; Tapio 2010; Tervonen 2017; Velez 2016.)

Walter Royce kuvaa julkaisemassaan artikkelissa vesiputousmallin suurimpia ongelmia ohjelmistotuotannossa ja sitä, kuinka ne olisi mahdollista korjata. Näihin korjauksiin kuuluvat varhainen prototyyppien suunnittelu ja iteratiivinen kehityksentoimitus asiakkaalle. Projektissa pitää myös olla tiivis yhteistyö asiakkaan kanssa (Ismagulov 2017). Näitä Walter Roycen ehdottamia parannuksia on suurimmaksi osaksi käytössä nykyajan ketterissä menetelmissä, joihin pureudutaan seuraavassa kappaleessa.

5.2 Ketterät menetelmät

Ohjelmistokehittäjät huomasivat ja olivat tietoisia perinteisten menetelmien epätoimivuudesta, ongelmista ja puutteista ohjelmistotuotannossa. Kehittäjät olivat työskennelleet erilaisissa laajoissa verkkopalvelukokonaisuus-asiakasprojekteissa käyttäen erilaisia malleja, pääsääntöisesti vesiputousmallia. Reaktiona raskaisiin perinteisiin menetelmiin kehittyi useita kevyitä ohjelmistotuotantomenetelmiä 1990-luvun aikana, joiden tarkoitus oli korjata perinteisten menetelmien ongelmat ohjelmistotuotannossa. Nämä uudet kevyet ohjelmistotuotantomenetelmät tunnetaan nykyään ketterinä menetelminä. (Agile Alliance 2001; Ismagulov 2017; Tapio 2010.)

Nämä kevyet ohjelmistotuotantomenetelmät omasivat samantapaisia piirteitä ja lähestymistapoja toistensa kanssa. Kaikki korostivat tiivistä yhteistyötä kehitysryhmän ja liiketoiminnan sidosryhmien välillä, usein toimitettavaa liikearvoa, itseohjautuvia ryhmiä sekä älykkäitä tapoja käsitellä, vahvistaa ja toimittaa koodia.

Käytettyään näitä kevyitä ohjelmistotuotantomenetelmiä luojat huomasivat ja kokivat menetelmien hyödyt käytännössä. He myös tajusivat, että muutkin saattaisivat olla kiinnostuneita

käyttämään näitä. Menetelmien luojat alkoivat kehittämään puitteita menetelmiä varten ja levittämään näitä eteenpäin eri organisaatioille. Jälkeenpäin alkoi ilmestyä erilaisia viitekehyksiä, joita muun muassa olivat Scrum, Extreme Programming, Feature-Driven Development (FDD) ja Dynamic Systems Development Method (DSDM). (Agile Alliance 2001; Ismagulov 2017.)

Menetelmien luojat huomasivat, että näillä eri menetelmillä ei ollut mitään johdonmukaista tapaa kuvata tapojansa kehittää ohjelmistoja, joten he päättivät vuonna 2001 kokoontua yhteen Yhdysvalloissa. Yhteensä 17 ohjelmistoalan huippuasiantuntijaa kokoontui yhteen, ja he alkoivat pohtia näiden menetelmien todellisia mahdollisuuksia ja tehokkaita hyödyntämistapoja. He pyrkivät löytämään yhtäläisyyksiä sekä eroavaisuuksia näiden menetelmien välillä. Kokoontumisen lopputuloksena julkaistiin ketterän ohjelmistotuotannon manifesti, joka määrittää kaikelle ketterälle ohjelmistotuotannolle yleiset arvot ja periaatteet. (Agile Alliance 2001; Ismagulov 2017; Tapio 2010.)

Ketterän ohjelmistotuotannon manifesti koostuu neljästä perusarvosta ja 12 periaatteesta, jotka johtavat ketterään lähestymistapaan ohjelmistojen kehittämiseen. (Eby 2016; Tapio 2010.)

Ketterien menetelmien perusarvot:

”Löydämme parempia tapoja tehdä ohjelmistokehitystä, kun teemme sitä itse ja autamme muita siinä. Kokemuksemme perusteella arvostamme:” (AgileManifesto 2001.)

1. **Yksilöitä ja kanssakäymistä** enemmän kuin menetelmiä ja työkaluja
2. **Toimivaa ohjelmistoa** enemmän kuin kattavaa dokumentaatiota
3. **Asiakasyhteistyötä** enemmän kuin sopimusneuvotteluja
4. **Vastaamista muutokseen** enemmän kuin pitäytymistä suunnitelmassa

”Jälkimmäisilläkin asioilla on arvoa, mutta arvostamme ensiksi mainittuja enemmän.” (AgileManifesto 2001.)

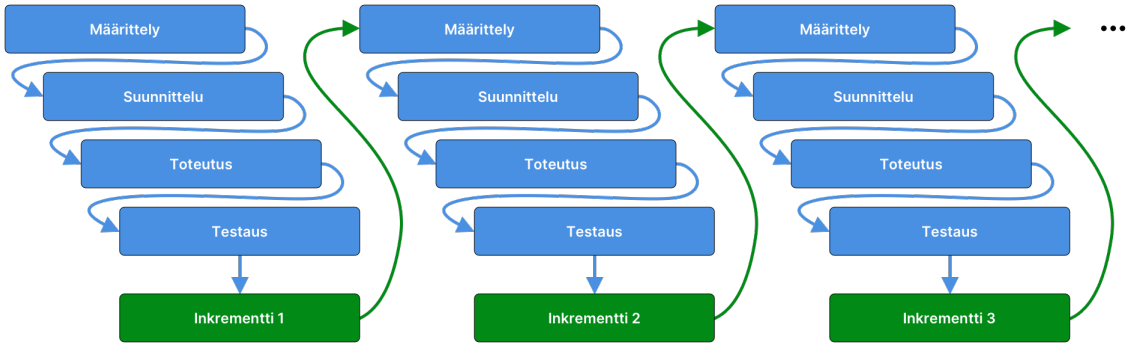
Ketterien menetelmien periaatteet:

1. Tärkein tavoitteemme on tyydyttää asiakas toimittamalla tämän tarpeet täyttäviä versioita ohjelmistosta aikaisessa vaiheessa ja säännöllisesti.
2. Otamme vastaan muuttuvat vaatimukset myös kehityksen myöhäisessä vaiheessa. Ketterät menetelmät hyödyntävät muutosta asiakkaan kilpailukyvyn edistämiseksi.
3. Toimitamme versioita toimivasta ohjelmistosta säännöllisesti, parin viikon tai kuukauden välein, ja suosimme lyhyempää aikaväliä.
4. Liiketoiminnan edustajien ja ohjelmistokehittäjien tulee työskennellä yhdessä päivittäin koko projektin ajan.

5. Rakennamme projektit motivoituneiden yksilöiden ympärille. Annamme heille puitteet ja tuen, jonka he tarvitsevat ja luotamme siihen, että he saavat työn tehtyä.
6. Tehokkain ja toimivin tapa tiedon välittämiseksi kehitystiimille ja tiimin jäsenten kesken on kasvokkain käytävä keskustelu.
7. Toimiva ohjelmisto on edistymisen ensisijainen mittari.
8. Ketterät menetelmät kannustavat kestävään toimintatapaan. Hankkeen omistajien, kehittäjien ja ohjelmiston käyttäjien tulisi pystyä ylläpitämään työtahtinsa hamaan tulevaisuuteen.
9. Teknisen laadun ja ohjelmiston hyvän rakenteen jatkuva huomiointi edesauttaa ketteryyttä.
10. Yksinkertaisuus - tekemättä jätettävän työn maksimointi - on oleellista.
11. Parhaat arkkitehtuurit, vaatimukset ja suunnitelmat syntyvät itseorganisoituvissa tiimeissä.
12. Tiimi tarkastelee säännöllisesti, kuinka parantaa tehokkuuttaan, ja mukauttaa toimintaansa sen mukaisesti. (AgileManifesto 2001.)

Näitä perusarvoja sekä periaatteita noudattavat ja soveltavat kaikki ketterät menetelmät, mutta jokainen soveltaa neljää perusarvoa eri tavoin. Kaikki menetelmät kuitenkin luottavat arvojen ohjaavan laadukkaiden, toimivien ohjelmistojen kehittämistä ja toimittamista. (Eby 2016; Tapio 2010.)

Ketterät menetelmät seuraavat pohjimmiltaan inkrementaalista mallia. Inkrementaalisen mallin päätarkoituksena on kehittää projektin kehitettävää ohjelmistoa vaiheittain. Näitä inkrementaalisen mallin vaiheita kutsutaan inkrementtikierrokseksi (kuvio 3).



Kuvio 3: Inkrementaalisen mallin vaiheet mallikuviona

Inkrementaalinen prosessi on iteratiivinen, joka pyrkii tuottamaan jokaisella inkrementtikierroksella toimivan tuotteen, eli inkrementin. Malliin liittyy myös erittäin tiivistä yhteistyötä kehitettävän ohjelmiston asiakkaan kanssa, jotta inkrementtikierroksilla kehitettävä ohjelmisto kehittyy inkrementaalisesti asiakaspalautteen myötä kohti haluttua lopputulosta.

Ensimmäisellä inkrementtikierroksella keskitytään ohjelmiston perusasioihin ja pyritään toimittamaan toimiva versio ohjelmistosta, jättäen isäominaisuudet huomioimatta. Asiakaspalautteen perusteella kehitetään suunnitelmia seuraaville inkrementtikierroksille, joiden aikana toteutetaan uusia tuotoksia sekä parannetaan edellisten kierroksien tuotoksia. Tätä prosessia toistetaan niin kauan, kunnes saavutetaan haluttu lopputulos. (Fagerström & Laakso 2011; Tapio 2010.)

Ketterät menetelmät eroavat jonkin verran toisistaan, mutta kuitenkin tärkeimpänä tavoitteena menetelmillä on toteuttaa toimiva ohjelmisto. Menetelmien erot ilmenevät ainoastaan näiden tavoista toteuttaa toimivaa ohjelmistoa. Esim. missä järjestyksessä ohjelmiston osia halutaan toteuttaa tai mikä on menetelmien kannalta tärkeintä (Tapio 2010). Seuraavassa kappaleessa pureudutaan kattavammin Scrum-viitekehykseen ja sen tapoihin toteuttaa toimivaa ohjelmistoa.

5.3 Scrum-viitekehys

Scrum on ketterä prosessikehys, joka on tarkoitettu monimutkaisten tuotteiden kehittämiseen, toimitukseen ja ylläpitoon. Scrum alun perin kehitettiin tuotehallintaan ja tuotteiden kehittämiseen vuonna 1986, mutta 1990-luvun alussa se laajensi ohjelmistotuotantoon kevyenä ohjelmistotuotantomenetelmänä Ken Schwaberin ja Jeff Sutherlandin toimesta. He kumpikin kehittivät omat menetelmänsä, jotka käyttivät samanlaista lähestymistapaa asiakasprojekteissa. Huomattuaan tämän he aloittivat yhteistyön vuonna 1995 integroidakseen menetelmiensä ideat yhteen kehykseen. Seurauksena he kehittivät yhdessä nykyään erittäin tunnetun sekä käytetyn Scrum-viitekehyksen muodollisena prosessina. (Lynch 2019; Schwaber & Sutherland 2017.)

Scrum-viitekehys perustuu empiiriseen prosessinhallintateoriaan eli jatkuvaan oppimiseen ja sopeutumiseen vaihteleviin tekijöihin. Empiiriseen prosessinhallintaan kuuluu kolme tukijalkaa: läpinäkyvyys, tarkastelu ja sopeuttaminen. Tukijalat vaativat sen, että merkittävien tekijöiden pitää olla helposti havaittavissa lopputuloksesta vastaaville, ja Scrumin tuotoksia pitää säännöllisesti tarkastella havaitakseen haitalliset poikkeamat halutusta lopputuloksesta. Löydettyään hyväksyttävien raja-arvojen ulkopuolella olevia poikkeamia tulee prosessia säätää mahdollisimman nopeasti, jotta minimoitaisiin haitat ja myöhemmät poikkeamat. (Drumond 2018; Schwaber & Sutherland 2017.)

Scrum myöntää, että projektin alussa ei tiedetä kaikkea ja että se kehittyy kokemuksen kautta. Scrum-viitekehys on rakennettu niin, että se auttaa työryhmiä luonnollisesti sopeutumaan muuttuviin olosuhteisiin ja käyttäjien vaatimuksiin hyödyntämällä iteratiivis-inkrementaalista lähestymistapaa ennustettavuuden optimoimiseen ja riskien kontrolloimiseen. Scrum-viitekehys on rakenteellinen, mutta se ei ole täysin jäykkä ja sitä on mahdollista räätälöidä minkä tahansa organisaation tarpeisiin. (Drumond 2018; Schwaber & Sutherland 2017.)

5.3.1 Scrum-tiimi

Yksi Scrum-tiimin jäsenistä on tuoteomistaja (Product Owner). Tuoteomistaja on vain yksi henkilö, joka edustaa asiakkaan toiveita sekä tarpeita. Tuoteomistaja voi olla itse asiakas tai mahdollisesti asiakkaan nimeämä ulkopuolinen henkilö. Tällöin asiakkaan on varmistettava tuotteenomistajan ymmärtävän asiakkaan tarpeita ja toiveita. Tuotteenomistajan tulee myös nähdä Scrum-tiimin tuoma arvo asiakkaalle omalla näkemyksellään. (Medlock 2017; Schwaber & Sutherland 2017; West 2018.)

Tuoteomistajan vastuulla on tuotteen kehitysjonon ajan tasalla ylläpitäminen, toteutettavien tehtävien priorisointi sekä toteutetun tuotteen arvon maksimointi. Tuoteomistajan on tiedettävä mitä tuotteen kehitysjonossa on sekä varmistettava kehitystiimin ymmärtävän toteutettavan tuotteen riittävän tarkasti. Scrumin aikana tuoteomistajan ja kehitystiimin kommunikaatio on erittäin tärkeää maksimoidakseen tuotteen arvon. (Medlock 2017; Schwaber & Sutherland 2017; West 2018.)

Tuoteomistajan luoman kehitysjonon lopulliseksi valmiiksi tuotteeksi toimittamisesta on vastuussa kehitystiimi (Development team). Kehitystiimi koostuu pääsääntöisesti, mutta ei aina, ohjelmistokehittäjistä, jotka tekevät asiakasprojektin varsinaisen työn eli ohjelmiston kokonaisuuden toteuttamisen. Kehitystiimin jäseniä ei lokeroida varsinaisilla titteleillä toteuttamaan vain yhden tyyppisiä tehtäviä, vaan kehitystiimi vastaa kokonaisuudesta yhdessä. Tämä tarkoittaa, että jokainen kehitystiimin jäsen voi työskennellä minkä tahansa kehitysjonon tehtävän parissa. Useimmiten päädytään kuitenkin siihen, että kehitystiimin jäsenet jakavat tehtävät kunkin jäsenen vahvuusalueisiin parhaimmin sopiviksi. Optimoidakseen kehitystiimin tehokkuutta tämän koko ei saa olla liian pieni mahdollisen osaamispulan vuoksi, eikä liian suuri. Se tuo kompleksisuutta liiallisen koordinoinnin vuoksi. Scrumissa suositellaan kehitystiimin koon olevan vähintään kolme henkilöä ja enintään yhdeksän henkilöä. (Medlock 2017; Schwaber & Sutherland 2017; West 2018.)

Viimeisenä Scrum-tiimin roolina on Scrummaster (Scrum Master). Scrummasterin pääasiallisena tehtävänä on palvella Scrum-tiimiä eräänlaisena tiimi- tai projektipäällikkönä ilman minkäänlaista määräysvaltaa. Scrummaster huolehtii päivittäin siitä, että Scrum-tiimi pystyy tekemään työtään optimaalisella tavalla. Tämä tapahtuu esimerkiksi poistamalla mahdolliset esteet niiden ilmetessä ja valmentamalla Scrum-tiimiä käyttämään Scrumin runkoa mahdollisimman tehokkaasti. (Medlock 2017; Schwaber & Sutherland 2017; West 2018.)

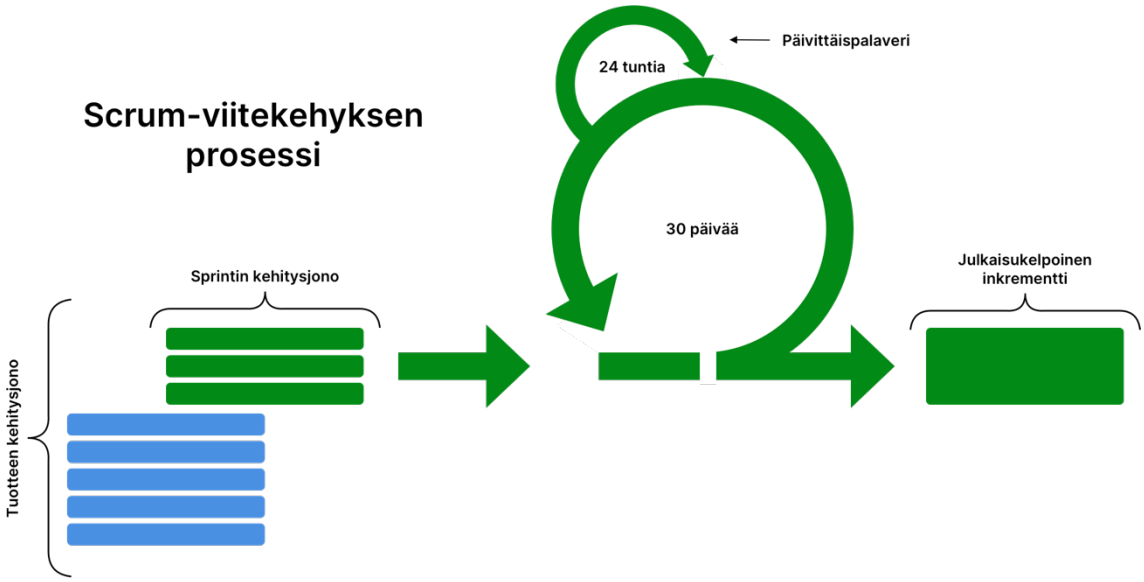
Scrummaster palvelee tuoteomistajaa seuraavasti: kuinka tuotteen kehitysjono kannattaisi järjestää arvon maksimoimiseksi, ehdottamalla tehokkaita kehitysjonon hallintatekniikoita, fasilitoimalla erilaisia Scrumin tapahtumia pyydettäessä tai tarpeen mukaan, sekä varmistaa Scrum-tiimin ymmärtävän mahdollisimman hyvin tavoitteet, laajuuden sekä tuotealueen. Scrummaster palvelee kehitystiimiä samoin keinoin, mutta pyrkii myös valmentamaan

itseohjautuvuuteen ja moniosaamiseen. Näin Scrummaster varmistaa Scrum-tiimin suoriutuvan työtehtävistään mahdollisimman tehokkaalla tavalla. (Juselius 2012; Medlock 2017; Schwaber & Sutherland 2017; West 2018.)

5.3.2 Scrumin rakenne ja toimintatapa

Scrum-viitekehyksestä yleisesti sanotaan, että sitä on yksinkertaista ymmärtää, mutta vaikea taitaa. Scrumia vaikeuttavat tämän eri käytännöt ja prosessit, jotka ovat oleellisia osia Scrumin onnistumiseen. Scrum-viitekehys koostuu kolmesta eri osasta: Scrum-tiimi, Scrum tapahtumat ja Scrumin tuotokset. Edellisessä kappaleessa käytiin läpi Scrum-tiimin eri roolit ja näiden roolien omat vastuut Scrum projektin aikana. Seuraavassa kappaleessa käydään läpi mitä Scrumin tuotokset ja tapahtumat oikein ovat ja miksi nämä ovat erittäin tärkeitä Scrumin onnistumisen kannalta. (Schwaber & Sutherland 2017.)

Scrumin työnkulku voidaan käytännössä jakaa kolmeen eri vaiheeseen (kuvio 4), näiden vaiheiden korrekti termi kuitenkin on Scrumin tuotokset (Scrum Artifacts). Scrumin tuotoksissa ensimmäisenä on tuotteen kehitysjono (Product Backlog). Tuotteen kehitysjono on tuotteenomistajan järjestämä lista tuotteen tarpeista ja vaatimuksista. Tämä lista ei ole ikinä valmis ja se kehittyy jatkuvasti projektin aikana mahdollisten muutoksien ja uusien tarpeiden ilmestymisien vuoksi. Tätä listaa voidaan myös karkeasti pitää projektisuunnitelmana. (Tapio 2010; Schwaber & Sutherland 2017.)



Kuvio 4. Scrum-viitekehyksen prosessi mallikuviona

Tuotteen kehitysjonossa on tuotteen ominaisuuksia, toimintoja, vaatimuksia, parannuksia sekä korjauksia. Näitä kutsutaan yleisesti yksittäisiksi tehtäviksi, eli storyiksi (story). Yksittäisiin tehtäviin tuotteenomistaja luo tarkat kuvaukset, tai niin tarkat kuin vain mahdollista sinä

ajankohtana. Kuvauksien lisäksi tuotteenomistaja luo ns. testikuvauksen, jonka perusteella voidaan toteutuksen jälkeen määrittää tehtävän olevan ”valmis”. (Dalmijn 2017; Schwaber & Sutherland 2017; Tapio 2010.)

Yksittäisille tehtäville tehdään kuvauksien perusteella työmääräarvio ja tehtävä pisteytetään. Pisteyttäminen on projektikohtaista, mutta yleisesti käytetään tarinapisteitä (Story Points). Nämä pisteet edustavat tehtävän haastavuutta, esim. 1 tarinapiste voi tarkoittaa 4-12 tuntia, 2 tarinapistettä 10-20 tuntia ja niin edelleen. Tehtävän pisteyttää vain ja ainoastaan kehitystiimi, joka tehtävän toteuttaa, sillä Scrumin säännöt määrittävät näin. Tuoteomistaja voi vaikuttaa työmääräarvioon selventämällä tehtävää tai tekemällä kompromisseja, mutta lopullinen päätös on kehitystiimillä. Tämän jälkeen edetään seuraavaan vaiheeseen eli ns. toteutusvaiheeseen. (Dalmijn 2017; Schwaber & Sutherland 2017; Tapio 2010.)

Scrumin toteutusvaihe koostuu Scrum tapahtumista (Scrum Events) ja Scrumin seuraavasta tuotoksesta eli sprintin kehitysjonosta (Sprint backlog). Scrumin toteutusvaiheessa työnkulku etenee pyrähdyksissä, jotka ovat enintään kuukauden pituisia iteraatioita. Näitä iteraatioita kutsutaan sprinteiksi (Sprint), jotka koostuvat suunnittelupalaverista (Sprint Planning), päivittäispalaverista (Daily Scrum), kehitystyöstä, sprintin katselmoinnista (Sprint Review) ja sprintin retrospektiivistä (Sprint Retrospective). Nämä ovat Scrumin keskeiset tapahtumat. (Schwaber & Sutherland 2017.)

Scrumin sprintti aloitetaan suunnittelupalaverilla. Alkuun tuotteenomistaja määrittää sprintin tavoitteen (Sprint Goal). Tämän mukaan Scrum-tiimi suunnittelee ja valitsee tuotteen kehitysjonosta tehtäviä, jotka pyritään toteuttamaan sprintin aikana, jotta sprintin tavoite toteutuisi. Sprintin aikana toteutettava tuote on nimeltään inkrementti, joka on Scrumin kolmas ja viimeinen tuotos. Yleisesti sprintin kehitysjonoon otetaan tuotteen kehitysjonosta eniten arvoa tuovat tehtävät, jotka ovat listan kärjessä. Kun sprintti on suunniteltu, varsinainen työvaihe alkaa ja jatkuu määritettyyn päivämäärään saakka. Sprintin pituus on projektikohtainen, mutta yleisesti se on kaksi viikkoa pitkä. (Medlock 2017; Schwaber & Sutherland 2017.)

Sprintin aikana kehitystiimi pitää päivittäin päivittäispalaverin, joka on aikarajattu 15 minuuttiin. Kehitystiimi päättää päivittäispalaverin ohjelmasta, mutta yleinen ohjelma koostuu kolmesta kysymyksestä, joihin kehitystiimin jäsenet vastaavat vuorollaan: mitä tein eilen, mitä aion tehdä tänään sekä onko mitään esteitä? Näillä kysymyksillä kartoitetaan kehitystiimin työtilannetta ja sprintin etenemistä. Kehitystiimi voi myös sopia miten he etenevät ilmestyneiden esteiden kanssa yhdessä. Scrummasterin ei tarvitse olla henkilökohtaisesti päivittäispalaverissa paikalla, mutta hänen pitää varmistaa kehitystiimin käyvän palaverin. Läsnäollessa hän huolehtii, etteivät muut mahdolliset läsnäolijat häiritse palaveria. (Medlock 2017; Schwaber & Sutherland 2017.)

Kun sprintti lähenee loppua, pidetään sprintin katselmointipalaveri, johon osallistuvat asiakasedustajat, Scrum-tiimi sekä tuoteomistajan mahdollisesti kutsumat eri sidosryhmät. Palaverissa esitellään ja näytellään sprintin aikana kehitettyä inkrementtiä asiakasedustajille sekä kerrotaan mahdollisista eteen tulleista ongelmista ja onnistumisista. Palaverin tavoitteena on saada suoraa asiakaspalautetta tuotteen tilasta, keskustella tuotteeseen liittyvistä asioista sekä mahdollisista muutoksista tai uusista ominaisuuksista. Näiden avulla pohditaan seuraavia toimenpiteitä ja myös tuotteen markkinatilaa. Palaverin päätyttyä tuloksena on päivitetty ja tarkistettu tuotteen kehitysjono seuraavia sprinttejä varten. (Medlock 2017; Schwaber & Sutherland 2017.)

Sprintin katselmoinnin jälkeen ja ennen seuraavan sprintin suunnittelupalaveria, pidetään vielä yksi palaveri, sprintin retrospektiivi eli sprintin jälkitarkastelu. Se on vain ja ainoastaan Scrum-tiimille tarkoitettu palaveri, jossa tarkastellaan ja arvioidaan edellisen sprintin toiminnan sujuvuutta etenkin Scrum-tiimin dynamiikan, prosessien ja työkalujen ympärillä. Retrospektiivin tavoitteena on tunnistaa mahdollisia parannuksia ja implementoida parannukset seuraavassa sprintissä. Parannuksia voi milloin tahansa sprintin aikana implementoida, mutta retrospektiivi tarjoaa muodollisen tilaisuuden sitä varten. Palaverin jälkeen on kyseinen sprintti suoritettu, ja Scrum-tiimi on toteuttanut julkaisukelpoisen inkrementin, jonka pohjalta seuraava sprintti käynnistyy. Seuraava sprintti käynnistyy heti edellisen sprintin loputtua. (Medlock 2017; Schwaber & Sutherland 2017.)

6 Tutkimusmenetelmät

Tässä luvussa käsitellään tutkimusmenetelmiä, joita hyödynnettiin tämän tutkimuksen tiedonkeruussa. Tutkimusmenetelminä käytettiin haastattelemista, havainnointia ja vertailemista, joilla saatiin vastauksia tutkimuskysymyksiin.

6.1 Haastatteleminen ja havainnointi

Tutkimusmenetelmien perusvaihtoehtoihin kuuluu haastatteleminen. Se on erittäin joustava ja vahva tiedonkeruumenetelmä, jonka vuoksi se sopii erittäin moniin erilaisiin tutkimustarkoituksiin. Tutkimukseen tarvittavaa tietoa voidaan kerätä erilaisilla haastattelumuodoilla, esimerkkejä näistä muodoista ovat yksilöhaastattelut, sähköpostihaastattelut, puhelinhaastattelut ja ryhmähaastattelut. Käyttötarkoituksen ja -tilanteen mukaan haastattelut jaetaan kolmeen eri lajityyppiin: strukturoituun, puolistrukturoituun ja avoimeen haastatteluun. (Accomazzo 2019; KAMK; Näpärä 2017; Saukkonen 2003.)

Haastattelutyyppien ydintekeminen on kaikilla sama: keskustellaan ja kysytään haastateltavalta kysymyksiä. Lajityypit eroavat toisistaan kuitenkin haastattelun toteutustavalla. Strukturoidussa haastattelussa käytetään lomaketyylistä toteutusta. Haastattelija on etukäteen

valmistanut kysymyksiä, joihin kaikki haastateltavat vastaavat samassa järjestyksessä. Näiden kysymyksien vastaukset on myös rajattu vastausvaihtoehdoilla, jonka vuoksi haastateltava joutuu valitsemaan omaa mielipidettä lähinnä olevan vaihtoehdon. Strukturoidut haastattelut sopivat silloin, kun on monta haastateltavaa ja he edustavat yhtenäistä ryhmää. (KAMK; Näpärä 2017.)

Puolistrukturoidussa haastattelussa kysymykset on strukturoidun haastattelun tapaisesti etukäteen mietitty. Erona strukturoituun haastatteluun on se, että kysymyksille ei ole luotu valmiita vastausvaihtoehtoja, joten haastateltavan annetaan vastata kysymyksiin vapaasti. Kysymykset kohdennetaan tiettyyn aiheeseen, yleisesti aiheeseen, jota ei ole vielä tutkittu paljon. (KAMK; Näpärä 2017.)

Avoin haastattelu on strukturoidun ja puolistrukturoidun vastakohta toteutustavaltaan. Avoin haastattelu toteutetaan vapaamuotoisena haastatteluna, jossa keskustellaan tietystä aiheesta. Se muistuttaakin enemmän perinteistä keskustelua kuin haastattelua. Avoimet haastattelut ovat yleisesti luontevia ja vapautuneita, jolloin haastateltava vastailee kysymyksiin suhteellisen vapaasti käyttämällä omia termejään ja ilmaisutapojaan. Keskustelun aikana aihe voi myös muuttua tai tuoda esiin jotain muuta aiheesta, sillä haastattelu perustuu henkilöiden keskeiseen vuorovaikutukseen. Haastattelun aikana voidaan myös kysyä seurantakysymyksiä, tarkkailla haastateltavan kehonkieltä tai havannoida muita vihjeitä. Haastattelutuloksilla tutkija pystyy hyödyntämään kerättyjä tietoja ja ymmärtämään tutkittavaa aihetta tai ongelmaa paremmin. (Accomazzo 2019; KAMK; Näpärä 2017; Saukkonen 2003.)

Havainnointi on myös yksi tutkimusmenetelmien perusvaihtoehdoista. Havainnoinnin avulla kerätään tietoa tutkittavasta kohteesta seuraamalla erilaisia tilanteita, tapahtumia ja käyttäytymisiä. Samalla selviää, toimiiko tutkittava kohde niin kuin se sanotaan toimivan. Havainnointi on erittäin tehokas menetelmä, jonka avulla saadaan välitöntä tietoa tutkittavasta kohteesta. (Autio & Kakko 2013; KAMK; Saukkonen 2003; Välisalo 2015.)

Havainnointi jaetaan kahteen eri muotoon: ei-osallistuvaan havainnointiin ja osallistuvaan havainnointiin. Ei-osallistuvassa havainnoinnissa tutkija havainnoi tutkittavaa kohdetta tarkasti rajatuissa tiloissa tai luonnollisissa tilanteissa. Ei-osallistuva havainnointi voidaan myös toteuttaa niin, että tutkittavan kohteen havaittavat eivät tiedä tutkijan läsnäolosta. Osallistuvassa havainnoinnissa on erittäin tyypillistä, että tutkija on yksi tutkimustilanteessa mukana olevista henkilöistä. Havainnoinnin tuloksilla tutkija pyrkii rakentamaan jonkinlaisen loogisen kokonaisuuden havainnoista sekä vertailemaan ja tulkitsemaan näitä omiin kokemuksiinsa. (Autio & Kakko 2013; KAMK; Saukkonen 2003; Välisalo 2015.)

6.2 Benchmarking

Benchmarking eli vertailuanalyysi on systemaattinen menetelmä. Menetelmällä tähdätään tiedon keruuseen, vertailuun ja tutkittavan kohteen oman toiminnan parantamiseen. Menetelmällä verrataan tutkittavaa organisaatiota toiseen organisaatioon. Yleensä tutkittavan organisaation tuotteita, toimintaa sekä prosesseja verrataan toisiin organisaatioihin, jotka ovat jossakin suhteessa parempia. Vertailukohteena voi myös olla organisaation käyttämät menetelmät, joita verrataan menetelmien omiin puitteisiin sekä määrityksiin. (Onatsu & Saari 2014.)

Vertailuanalyysin tarkoituksena on pystyä tunnistamaan heikkouksia tutkittavan kohteen omassa toiminnassa. Löydettyjen heikkouksien avulla voidaan laatia kehitysideoita toiminnan kehittämiseen. Tutkimalla organisaatioita, joilla on huippuluokan suorituskyky toiminnassaan, voi oppia erittäin paljon. Organisaatioiden toiminnasta pyritään erittelemään se, mikä tekee näistä niin ylivoimaisia. Löydetyillä tiedoilla pystytään mahdollisesti implementoimaan muutoksia omaan organisaatioon, tuottamalla merkittäviä parannuksia. (Onatsu & Saari 2014.)

6.3 Reliabiliteetti ja validiteetti

Tutkimukset asettavat tietyt arvot ja säännöt, joihin tutkimuksien tulisi pyrkiä. Sen vuoksi tutkimuksen luotettavuudella ja pätevyydellä on erittäin tärkeä rooli. Tutkimuksien luotettavuutta ja pätevyyttä arvioidaan, mitataan ja tarkastellaan mittausmenetelmillä nimeltään reliabiliteetti ja validiteetti. (Kyvyt 2019; Puusniekka & Saaranen-Kauppinen 2009.)

Reliabiliteetilla tarkastellaan tutkimuksen luotettavuutta mittaamalla, miten luotettavasti ja toistettavasti saadaan ei-sattumanvaraisia tuloksia. Tutkimuksen luotettavuutta voidaan mitata esim. toistomittauksilla. Käytännössä tämä tarkoittaa sitä, että sama tutkittava kohde mitataan useampaan kertaan, jonka jälkeen mittaustuloksia vertaillaan. Kun mittaustulokset täsmäävät ja ovat samat useamman mittauksen jälkeen, tutkittava kohde on reliaabeli. Poiketessa tuloksien samaisuudessa, kertoo tämä tutkittavan kohteen reliabiliteetista. (Kyvyt 2019; Puusniekka & Saaranen-Kauppinen 2009; Tilastokeskus.)

Validiteetilla tarkastellaan sen sijaan tutkimuksen pätevyyttä. Pätevyydellä tarkoitetaan tutkimuksen ”oikeutta” saatujen tuloksien ja tehtyjen päätelmien kanssa. Pätevyydellä myös tarkoitetaan tutkimuksen perusteellista toteuttamista. Saavuttaakseen korkean validiteetin, tutkimuksessa pitää tarkasti kuvailla tehty tutkimusprosessi ja perustella tehdyt valinnat suhteutettuna teoreettiseen viitekehykseen. Luettuaan tutkimuksen, lukijalla pitäisi olla mahdollisuus toteuttaa tutkimus uudestaan lukemansa perusteella. Virheiden ilmestyminen on myös mahdollista tutkimuksessa, kun esimerkiksi tutkija ymmärtää tutkittavaa aihetta väärin tai kyselee vääriä kysymyksiä tutkimusta varten. (Kyvyt 2019; Puusniekka & Saaranen-Kauppinen 2009.)

7 Tutkimuksen toteutus

Tutkimuksen lähtökohtana oli selvittää, kuinka tarkasti toimeksiantajan kanssa valitut asiakasprojektit noudattavat Scrum-viitekehyksen asettamia käytäntöjä ja sääntöjä. Tavoite oli selvittää poikkeavatko projektit Scrum-viitekehyksestä soveltamalla tämän käytäntöjä tai jättämällä sääntöjä noudattamatta. Samalla tavoitteena oli vertailla asiakasprojekteja toisiinsa selvittääkseen näiden keskeisiä eroja. Poiketessa käytännöistä ja säännöistä tarkoitus oli selvittää, onko projektin toiminta heikompaa soveltamilla tavoillaan. Tutkimuksen lopputuloksilla oli tarkoitus kehittää jonkinlaisia kehitysehdotuksia projekteille näiden ollessa tarpeen.

Tutkimuksen tiedonkeruutta varten hyödynnettiin haastattelu- ja havainnointitutkimusmenetelmiä. Menetelmillä pyrittiin saamaan mahdollisimman paljon tietoa projektien toimintatavoista, käytännöistä sekä mahdollisista muista asioista, jotka ilmenisivät tiedonkeruun aikana. Näitä tietoja hyödynnettiin myöhemmin vertailuanalyysissa, kun asiakasprojekteja verrattiin sekä toisiinsa että Scrum-viitekehykseen.

Tutkittavana kohteena oli kolme asiakasprojektia ja tutkimuksen toteutus tapahtui vuoden 2020 keväällä maalis- ja huhtikuun välillä, jonka ajalta tämän tutkimuksen kaikki tulokset ovat peräisin. Tutkittavia kohteita ei tutkittu samanaikaisesti, vaan aina toisen jälkeen.

Tutkimus aloitettiin maaliskuussa käynnissä olevalla TC Vehicle Oy:n projektilla. Seuraavana tutkittavana projektina toimi käynnissä oleva Tamro Oyj:n projekti, jota aloitettiin tutkimaan huhtikuun alussa heti edellisen projektin jälkeen. Viimeisenä tutkittavana projektina toimi käynnissä oleva Helsingin yliopiston projekti, jonka tutkiminen aloitettiin huhtikuun lopulla. Kaikissa projekteissa paitsi Helsingin yliopiston projektissa, tutkimuksen tekijä oli toiminut ohjelmistokehittäjänä kyseisissä projekteissa.

7.1 Asiakasprojektien osapuolien haastattelut

Saadakseen mahdollisimman hyvää ja vapaamuotoista tietoa tutkittavista asiakasprojekteista hyödynnettiin tiedonkeruumenetelmänä haastattelemista. Tutkimusta varten haastateltavat henkilöt valikoituivat haastateltavaksi heidän ollessa Scrum-tiimin jäseniä tutkittavissa asiakasprojekteissa. Haastattelut olivat haastattelutyypiltään avoimia sekä puolistrukturoituja, jotta voitaisiin kerätä hiukan erityyppisiä haastattelutuloksia. Avoimet haastattelut toteutettiin verkossa Google Meets -palvelun välityksellä ilman etukäteen mietittyjä kysymyksiä, mutta Scrum-viitekehysaihe oli osapuolilla tiedossa. Puolistrukturoidut haastattelut sen sijaan toteutettiin sähköpostiviestien välityksellä etukäteen mietityillä kysymyksillä.

Ensimmäiset haastattelut toteutettiin TC Vehicle Oy:n projektille. Haastateltavaksi valiutui projektin Scrum-tiimin Scrummaster sekä tuotteen omistaja. Scrummasterin haastattelu toteutettiin lyhyenä, 15 minuutin pituisena avoimena haastatteluna, jonka yhteydessä

keskusteltiin vapaasti projektin Scrum-viitekehyksen käytöstä ja yleisesti Scrumista. Haastattelun aikana keskusteltiin seuraavista aiheista: kuinka tarkasti projekti seuraa Scrum-viitekehystä, sovelletaanko sitä omin keinoin laisinkaan ja minkälaiseen projektiin Scrum-viitekehys sopii parhaiten.

Scrum-tiimin tuoteomistajaa haastateltiin avoimen haastattelun sijaan puolistrukturoidulla sähköpostihaastattelulla. Puolistrukturoidulla haastattelulla pyrittiin saamaan hiukan muodollisempia haastattelutuloksia Scrum-viitekehyksen käytöstä tutkimusta varten. Haastattelukysymyksinä olivat:

1. Oliko Scrum-viitekehys tuttu entuudestaan ennen nykyistä projektia?
2. Seurasiko projekti tarkasti Scrum-viitekehyksen asettamia käytäntöjä ja sääntöjä?
3. Sovellettiinko Scrum-viitekehyksen käytäntöjä (tapahtumia) projektin aikana? Jos näin oli, miten?
4. Mikä oli Scrum-viitekehyksen ehdoton vahvuus projektin aikana?
5. Mikä oli Scrum-viitekehyksen heikkous projektin aikana?

Seuraavat haastattelut toteutettiin Tamro Oyj:n projektille. Projektista haastateltiin Scrum-tiimin Scrummaster sekä tuotteen omistaja niin kuin edellisessä projektissa tehtiin. Puolistrukturoidun ja avoimen haastattelun sijaan, kummallekin haastateltavalle suoritettiin lyhyt 15 minuutin pituinen avoin haastattelu. Kummassakin haastattelussa keskusteltiin yleisesti siitä, miten Scrumia hyödynnetään projektissa, mikä oli Scrum-viitekehyksen ehdoton vahvuus sekä heikkous projektin aikana ja miten tutkimuksen toimeksiantaja pyrkii toteuttamaan Scrumia projekteissa.

Viimeisenä toteutettiin haastattelut Helsingin yliopiston projektille. Tähän projektiin valiutuivat haastateltavaksi Scrum-tiimin Scrummaster, tuoteomistaja ja yksi kehitystiimin jäsenistä. Scrummasterille toteutettiin avoin haastattelu, jonka aikana keskusteltiin Scrumista yleisesti, miten hyvin Scrum toimii Helsingin yliopiston projektin kokoisessa projektissa, mahdollisista heikkouksista ja ehdottomista vahvuuksista. Scrummaster myös jakoi ruutunsa haastattelun aikana näyttääkseen miten kyseistä projektia hallitaan käyttämällä tehtävienhallintaohjelmistoa. Projektin tuotteenomistajalle sekä kehitystiimin kehittäjälle suoritettiin puolistrukturoitu haastattelu. Haastattelua varten käytettiin samaa viiden kysymyksen pohjaa, kuin ensimmäisessä puolistrukturoidussa haastattelussa, keräämällä tietoa Scrumin käytöstä yleisesti projektista.

7.2 Asiakasprojektien toimintatapojen havainnoinnit

Havainnoiminen valiutui toiseksi tiedonkeruumenetelmäksi, koska menetelmällä saadaan välitöntä tietoa tutkivasta kohteesta. Tutkimuksessa käytettiin kumpaakin havainnoinnin olomuotoa, eli ei-osallistuvaa sekä osallistuvaa havainnointia riippuen siitä, mikä asiakasprojekti oli

kyseessä. Havainnoinnin avulla pyrittiin havainnoimaan, kuinka projektit ovat implementoineet Scrum-viitekehyksen käytännössä, mitä projektinhallintaohjelmistoja projektit käyttävät pääsääntöisesti, mitkä ovat parhaimpia käytäntöjä Scrum-viitekehyksen parissa, sekä miten henkilöt vuorovaikuttavat toistensa kanssa projektin aikana.

Havainnoinnit projekteille TC Vehicle Oy ja Tamro Oyj suoritettiin osallistuvalla havainnoinnilla maalis- ja huhtikuun aikana, kun tutkimuksen laatija toimi kehitystiimin jäsenenä kyseisissä projekteissa. Havainnoinnit kuitenkin aloitettiin TC Vehicle Oy projektin kanssa. Tamro Oyj:n projektin kanssa vasta huhtikuun alusta eteenpäin. Havainnoidessa tutkijan varsinaisena kohteena olivat projektin Scrum tapahtumat ja Scrum-tiimin vuorovaikutus toistensa kanssa tapahtumien aikana. Näiden avulla saatiin parempi kokonaiskuva Scrumin käytöstä, johon pystyi verrata muiden projektien toimintatapoja.

Helsingin yliopiston projektia havainnoitiin ei-osallistuvana havainnoinnilla huhtikuun lopulla. Helsingin yliopiston projekti on suurimmaksi osaksi eristetty ei-halutuilta henkilöiltä, joten havainnointi toteutettiin kyseisen projektin Scrummasterin avulla haastattelun yhteydessä ruudunjaolla. Haastattelun ruudunjaon aikana havainnoitiin, miten kyseistä projektia hallitaan projektinhallintaohjelmistolla, sekä miten projektin Scrum-tiimi vuorovaikuttavat muiden sisäisten tiimien kanssa. Haastattelun avulla saatiin jonkinnäköinen kokonaiskuvan laajasta operaatiosta.

7.3 Asiakasprojektien toimintatapojen erojen vertailu

Tämän tutkimuksen viimeisenä osana toteutettiin vertailuja haastattelun ja havainnoin avulla saatujen tietojen kanssa. Vertailu kohdistui tutkimuksen valittuihin asiakasprojekteihin. Tutkimuksessa toteutettiin kaksi eri vertailua: kuinka projektit noudattivat Scrum-viitekehyksen asettamia käytäntöjä ja sääntöjä ja millä tavalla projekteissa noudatettiin käytäntöjä verrattuna muihin tutkimuksessa olleisiin projekteihin.

Ensimmäisessä vaiheessa verrattiin, kuinka projektien arkikäytännöt seurasivat virallisessa oppaassa määritettyä Scrum-viitekehyksestä. Eli vertailussa verrattiin projektien todellisia käytäntöjä suhteessa Scrum oppaan virallisiin määritettyihin käytäntöihin. Vertailussa seurattiin vuorovaikutusmetodien käyttöä Scrum tapahtumien aikana, Scrumin määrittelemien tapahtumien toteutumista projektin arjessa sekä kuinka Scrum ilmeni projektin edistymisen kulussa. Tutkimuksen seuraava vaihe toteutetaan hyödyntäen tässä vaiheessa syntyneiden tuloksien sekä kerättyjen havaintojen pohjalta.

Tutkimuksen toisessa vaiheessa vertailtiin, kuinka projektit noudattavat Scrum-viitekehystä suhteessa toisiinsa. Vertailukohteena olivat projektien yhteneväisyydet ja erot Scrum käytäntöjen seuraamisessa. Vertailu toteutettiin hyödyntäen tuloksia silmämääräistä havainnointia sekä haastatteluista saatuja havaintoja. Vertailun varsinaisena tarkoituksena oli selvittää

projektien keskeiset erot ja yhteneväisyydet Scrum-viitekehyksen noudattamisessa. Tuloksissa otettiin myös huomioon projektien kokoerot, eli erot projektien tavoitteissa sekä kehitystiimin kehittäjien määrässä. Seuraavissa luvuissa on avattu tämän tutkimuksen löydöksiä sekä loppuyhteenveto löydöksien pohjalta.

8 Tulokset

Tutkimuksessa tutkittiin, kuinka tarkasti asiakasprojektit noudattivat Scrum-viitekehystä arjessaan sekä, kuinka paljon projektit eroavat toisistaan Scrum käytännöissään. Haastattelujen myötä saatiin hyvä kokonaiskuva siitä, kuinka projektit hyödyntävät Scrumia ja kuinka Scrum-tiimin dynamiikka toimii. Haastateltaville pääsääntöisesti esiteltiin samantapaisia kysymyksiä liittyen Scrum-viitekehykseen. Saadut vastaukset eivät eronneet toisistaan paljoa riippumatta henkilön roolista projektissa. Yleisimmät vastaukset Scrumin vahvuuteen olivat tämän tuovan ryhtiä sekä kokonaisläpinäkyvyyttä projektille. Haastattelujen ja havainnoimisen avulla saatiin selville, että projektit noudattavat Scrumia merkittävän hyvin, mutta hiukan soveltamista löytyy joissakin tapauksissa.

8.1 Asiakasprojektien eroavaisuudet Scrum-viitekehyksen käytäntöihin

Yleisin tapa miten Scrumia sovelletaan, on Scrumin tapahtumien sekä Scrum-tiimin yhteydessä. Jotta Scrum-projekti toimisi oikeaoppisesti, pidetään sprintin aikana neljä erillistä pakollista palaveria. Oleellisimmat palaverit kuitenkin näistä ovat suunnittelupalaveri ja sprintin katselmointi eli ns. sprintin demotilaisuus. Päivittäispalavereita ja sprintin retrospektiiviä sovelletaan usein jättämällä ne välistä tai niitä pidetään väljemmin aikavälein.

Tutkimusta tehdessä havaittiin, että kaikki projektit eivät toteuta Scrum tapahtumia samalla tavalla sprintin aikana. TC Vehicle Oy ja Tamro Oyj:n projektien aikana toteutettiin kaikki Scrum tapahtumat paitsi sprintin retrospektiivi. Sprintin retrospektiiville ei nähty varsinaista tarvetta, mutta se kuitenkin järjestettiin, kun tarve sitä vaati. Kun sprintin retrospektiivi toteutettiin, sitä ei kuitenkaan pidetty pelkästään omana palaverinaan niin kuin pitäisi, vaan se yhdistettiin sprintin katselmointipalaveriin. Helsingin yliopiston projektissa sen sijaan toteutettiin kaikki Scrum tapahtumat, näiden lisäksi myös viikoittain tuotteen kehitysjonon jalostamispalaveri, jota ei toisissa projekteissa järjestetty.

Tamron projektissa sovellettiin myös hieman päivittäispalaveria. Scrum-oppaan mukaan se on tarkoitettu vaan kehitystiimille tarkastaakseen työtilannetta eikä ulkopuolisille henkilöille laisinkaan. Projektissa kuitenkin hyödynnettiin tuoteomistajaa päivittäispalavereissa raportoimalla tuotteen kehitysjonon tilannetta, joka edisti Scrum-tiimin yhteistyötä huomattavasti.

Syyt eroihin ovat melko selviä: projektin koko sekä sen monimutkaisuus. TC Vehicle ja Tamron projektit ovat kooltaan pieniä kokonaisuuksia verrattuna Helsingin yliopiston projektiin, joten projekteissa pystytään hieman joustamaan Scrumin käytännöistä. Soveltaminen ei ole kuitenkaan suositeltavaa Scrum oppaan mukaan, mutta kuitenkaan Scrum tapahtumien soveltamisella ei ilmennyt olevan mitään haittoja projektien aikana, enemmänkin hyötyjä. Helsingin yliopiston projektin tapauksessa oli erittäin paljon muuttujia, jonka vuoksi Scrum-viitekehystä halutaan seurata mahdollisimman tarkkaan.

Scrumia sovelletaan myös toisella tavalla: Scrum-tiimin kanssa. Käyttäen Scrumia oikeaoppisesti Scrum-tiimin pitäisi vain ja ainoastaan keskittyä käynnissä olevaan Scrum-projektiin. Scrum-projektien aikana on monta asiaa tehtävänä, mutta vain tietyn verran aikaa ennen seuraavan sprintin alkua. Kun Scrum-tiimi tai vain yksikin tiimin jäsenistä työskentelee aktiivisesti toisessa projektissa samaan aikaan, menettävät kummatkin projektit maksimaalisesta tuottavuudestaan osan. Syy tuottavuuden menetykseen on se, että projektit vaativat kaikilta Scrum-tiimin jäseniltä maksimaalista keskittymistä projektiin.

Tutkittavista asiakasprojekteista tehtiin seuraavia havaintoja Scrum-tiimien käytöstä. TC Vehicle ja Tamron projekteissa Scrum-tiimejä sovellettiin lähes samalla tavalla. Projektin Scrummaster sekä kehitystiimien jäsenet toimivat samanaikaisesti muissa projekteissa aktiivisesti. Näin ei tulisi toimia, sillä tämä on Scrum-viitekehykselle vastaista. Tästä ei ollut varsinaisesti haittaa projektien aikana, sillä kaikki kehitystiimin jäsenet olivat myös lisensoituja Scrummastereita, joten projektin valmentajasta ei ollut pulaa.

Syy, miksi näin tehtiin, on kuitenkin ymmärrettävä: toimeksiantajalla ei yksinkertaisesti riittänyt henkilöstöä allokoimaan projektia varten sekä omaa kehitystiimiä että Scrummasteria. Tämän vuoksi projektit menettivät hieman maksimaalisesta tuottavuudestaan, mutta projektit kuitenkin kykenivät tuottamaan joka iteraatiossa julkaisukelpoisen inkrementin, joka vastasi sprintin tavoitetta.

Helsingin yliopiston projektissa sen sijaan seurataan tarkasti Scrum-viitekehystä. Kyseisen projektin Scrum-tiimi (Scrummaster sekä kehitystiimi) on toimeksiantajan puolesta allokoitu vain sille projektille, joten heille ei tule häiritseviä tekijöitä. Projektin Scrummaster toimi myös yhtenä kehitystiimin jäsenenä, joka on Scrumissa sallittua.

8.2 Asiakasprojektien keskeiset yhtäläisyydet ja erot

Kun projektit verrataan toisiinsa, voidaan havaita suhteellisen selvästi, miten projektit eroavat toimintatavoiltaan. Pienemmän kokoisissa projekteissa joustetaan projektien Scrum-viitekehyksen tarkkuudesta, sillä nämä eivät ole aina niin monimutkaisia kokonaisuuksia. Projekteja pystytään hallitsemaan hyvin ja mahdolliset poikkeavuudet voidaan havaita helposti ja

nopeasti. Scrum-viitekehyksen täsmällisyydelle ei ole erityistä tarvetta tai pakotetta, mutta Scrumin ydinarvot ovat kuitenkin ennallaan.

Laajemmissa kokonaisuuksissa, jotka ovat lisäksi monimutkaisia projekteja, pitää hyödyntää Scrum-viitekehystä tarkemmin, kuin ei niin monimutkaisissa. Syy tähän on projektien mahdollisien muuttujien määrä, joka on merkittävästi suurempi. Kun Scrumia ei hyödynnetä oikein ja tarkkaan, se vaikeuttaa ennalta ehkäisemään tulevia ongelmia. Ongelmat saattavat tulla vasta erittäin myöhään esiin, ja silloin voi olla jo liian myöhäistä. Tulevia ongelmia pystytään ehkäisemään mahdollisimman hyvin toteuttamalla Scrum tapahtumia niin kuin ne on määritetty.

Projektien toimintatavat olivat myös yhteneväisiä monin tavoin. Projektit käyttivät samaa projektienhallintaohjelmistoa, joka oli tässä tapauksessa Atlassianin julkaisema Jira. Jiran avulla projektien tuotteenomistajat pystyivät hallita tuotteen kehitysjonoa sekä kehitystiimi sprintin kehitysjonoa helposti ja kätevästi. Scrummasterit toimivat projekteissa myös oikein valmentamalla Scrum-tiimiä ja harjoittamalla Scrumia kaikille osapuolille. Projektien kehitystiimit toimivat myös läheisessä yhteistyössä tuotteenomistajien kanssa.

9 Yhteenveto ja johtopäätökset

Tutkimuksen päätavoitteena oli saada selkeyttä siihen, kuinka paljon Scrumia käyttävät asiakasprojektit eroavat toisistaan arkikäytännöistään sekä kuinka hyvin nämä noudattavat Scrum-viitekehyksen käytäntöjä. Tutkimuksen aikana saatujen tietojen perusteella saatiin onnistuneesti selkeyttä tutkimuskysymyksiin.

Tutkimuksen tutkittavista projekteista selvisi, että kokonaisuuden koosta sekä monimutkaisuudesta voidaan havaita ja päätellä, kuinka tarkasti Scrumia hyödynnetään. Projekteista ilmeni, että mitä monimutkaisempi kokonaisuus on, sitä enemmän Scrum-viitekehyksen sääntöjä noudatetaan ja päinvastoin. Vaikka Scrum-viitekehystä sovellettiin omilla käytännöillä, näiden takia ei ilmennyt mitään radikaaleja haittoja. Tämä oli projektien suurimpia eroavaisuuksia. Tärkeää kuitenkin on, että Scrumin ydinarvot säilyvät, jolloin marginaalinen soveltaminen tms. voi olla hyvä.

Tässä opinnäytetyössä käytettiin tutkimusmenetelmistä haastattelemista ja havainnoimista. Tutkimus täyttää reliabiliteetin määrityksen sillä, että menetelmät ovat toistettavissa ja toistaessa menetelmiä päästään samoihin lopputuloksiin. Haastatteluista ja havainnoinnista kerätyt tiedot hyödynnettiin vertailuanalyysin yhteydessä myöhemmin, kun tutkittavia kohteita verrattiin toisiinsa.

Tutkimuksessa selvitettiin myös, mitkä ovat Scrumin parhaimpia käytäntöjä, mihin saatiin hyviä vastauksia haastattelujen avulla eri Scrum-tiimien jäseniltä. Scrumin ehdottomiin hyviin käytäntöihin kuuluvat tämän joustavuus eri ongelmatilanteissa, Scrum prosessin tuoma ryhti ja läpinäkyvyys projektille sekä tiivis yhteistyö osapuolien kesken. Tutkimuksessa selvisi, että tutkimuksen asiakasprojektit seurasivat Scrumia hyvin, eivätkä nämä tarvinneet mitään ehdottomia kehitystarpeita arkikäytäntöihinsä.

Lähteet

Sähköiset

Accomazzo, A. 2019. Conducting qualitative research. Viitattu 17.4.2020. Saatavissa: https://www.surveymonkey.com/mp/conducting-qualitative-research/

Agile Alliance. 2001. Agile Manifesto. Viitattu 11.4.2020. Saatavissa: https://www.agilealliance.org/

AgileManifesto. 2001. Agile Manifesto. Viitattu 11.4.2020. Saatavissa: https://agilemanifesto.org/iso/fi/manifesto.html

Autio, M. ja Kakko, K. 2013. Potilaan tunnistaminen leikkausosastolla sairaanhoitajien toteuttamana – havainnointi tutkimus. AMK-opinnäytetyö. Oulun seudun ammattikorkeakoulu, hoitotyön koulutusohjelma. Viitattu 17.4.2020. Saatavissa: http://urn.fi/URN:NBN:fi:amk-2013101016006

Baca, D., Petersen, K. & Wohlin, C. 2009. The Waterfall Model in Large-ScaleDevelopment. Viitattu 4.4.2020. Saatavissa: https://www.researchgate.net/publication/30498645_The_Waterfall_Model_in_Large-Scale_Development

Bell, T. ja Thayer, T. 1976. SOFTWARE REQUIREMENTS: ARE THEY REALLY A PROBLEM? Viitattu 4.4.2020. Saatavissa: https://pdfs.semanticscholar.org/a50d/8f564d063e526e94114875220440f64d8666.pdf

Dalmijn, M. 2017. 12 common mistakes made when using Story Points. Viitattu 15.4.2020. Saatavissa: https://medium.com/serious-Scrum/12-common-mistakes-made-when-using-story-points-f0bb9212d2f7

Druid Oy. 2020. Helsingin yliopisto. Viitattu 28.4.2020. Saatavissa: https://www.druid.fi/reference/helsingin-yliopisto

Druid Oy. 2020. Onko sinussa druidiainesta? Viitattu 27.3.2020. Saatavissa: https://rekry.druid.fi/

Druid Oy. 2020. Touring Cars. Viitattu 28.4.2020. Saatavissa: https://www.druid.fi/referenssit/touring-cars

Druid Oy. 2020. Täyden palvelun ketterä ohjelmistotalo. Viitattu 27.3.2020. Saatavissa: https://www.druid.fi/palvelut

Drumond, C. 2018. What is Scrum? Viitattu 13.4.2020. Saatavissa: https://www.atlassian.com/agile/Scrum

Eby, K. 2016. Comprehensive Guide to the Agile Manifesto. Viitattu 12.4.2020. Saatavissa: https://www.smartsheet.com/comprehensive-guide-values-principles-agile-manifesto

Fagerström, P. ja Laakso, T. 2011. Asiakastietojärjestelmä Pyhtään kuntokeskukselle. AMK-opinnäytetyö. Turun ammattikorkeakoulu, tietojenkäsittelyn koulutusohjelma. Viitattu 12.4.2020. Saatavissa: http://urn.fi/URN:NBN:fi:amk-2011060110604

Free Management eBooks. 2015. The Waterfall Model. Viitattu 5.4.2020. Saatavissa: http://www.free-management-ebooks.com/news/waterfall-model/

Helsingin yliopisto. 2020. Helsingin yliopisto. Viitattu 28.4.2020. Saatavissa: https://www.helsinki.fi/fi/yliopisto

Hyperiondev. 2017. 10 Different Types of Software Development. Viitattu 1.4.2020. Saatavissa: https://blog.hyperiondev.com/index.php/2017/09/26/types-of-software-development/

Ismagulov, A. 2017. The reasons and the purposes of Waterfall and Agile. Viitattu 4.4.2020. Saatavissa: https://medium.com/@forkandpie/the-reasons-and-the-meaning-of-waterfall-and-agile-841adf708891

Juselius, T. 2012. Scrum-projektinhallintamenetelmä. AMK-opinnäytetyö. Laurea-ammattikorkeakoulu, tietojenkäsittelyn koulutusohjelma. Viitattu 14.4.2020. Saatavissa: http://urn.fi/URN:NBN:fi:amk-201201161359

KAMK. Haastattelu. Viitattu 17.4.2020. Saatavissa: https://www.kamk.fi/fi/opari/Opinnaytetyopakki/Teoreettinen-materiaali/Tukimateriaali/Aineiston-keruumenetelmat/Haastattelu

KAMK. Havainnointi. Viitattu 17.4.2020. Saatavissa: https://www.kamk.fi/fi/opari/Opinnaytetyopakki/Teoreettinen-materiaali/Tukimateriaali/Aineiston-keruumenetelmat/Havannointi

KPI Partners. 2018. Traditional vs. Agile Software Development Methodologies. Viitattu 3.4.2020. Saatavissa: http://www.kpipartners.com/blog/traditional-vs-agile-software-development-methodologies

Kyvyt. 2019. Luotettavuus. Viitattu 18.4.2020. Saatavissa: https://kyvyt.fi/view/artefact.php?artefact=304009&view=72174

Laine, H. ja Paakki, J. 2003. Prototyypin merkitys. Viitattu 3.4.2020. Saatavissa: https://www.cs.helsinki.fi/u/paakki/ohtuk03-luento2-bw.pdf

Lynch, W. 2019. The Brief of History of Scrum. Viitattu 13.4.2020. Saatavissa: https://medium.com/@warren2lynch/the-brief-of-history-of-Scrum-15efb73b4701

Medlock, J. 2017. A Short Introduction to the Scrum Methology. Viitattu 13.4.2020. Saatavissa: https://medium.com/chingu/a-short-introduction-to-the-Scrum-methodology-7a23431b9f17

Näpärä, L. 2017. Haastattelun lajityypit. Viitattu 17.4.2020. Saatavissa: https://spoken.fi/2180/

Onatsu, M. ja Saari, R. 2014. Benchmarking-matka ruotsiin. AMK-opinnäytetyö. Kajaanin ammattikorkeakoulu, matkailun koulutusohjelma. Viitattu 18.4.2020. Saatavissa: http://urn.fi/URN:NBN:fi:amk-2014121219532

Powell-Morse, A. 2016. Waterfall Model: What Is It and When Should You Use It? Viitattu 5.4.2020. Saatavissa: https://airbrake.io/blog/sdlc/waterfall-model

Puusniekka, A. ja Saaranen-Kauppinen, A. 2009. Menetelmäopetuksen tietovaranto Kvali-MOTV. Viitattu 18.4.2020. Saatavissa: https://www.fsd.tuni.fi/fi/tietoarkisto/julkaisut/kvali-motv.pdf

Saukkonen, P. 2003. Tutkimusmenetelmät. Viitattu 17.4.2020. Saatavissa: https://www.mv.helsinki.fi/home/psaukkon/tutkielma/Tutkimusmenetelmat.html

Schwaber, K. ja Sutherland, J. 2017. Scrum-opas. Viitattu 26.3.2020. Saatavissa: https://www.Scrumguides.org/docs/Scrumguide/v2017/2017-Scrum-Guide-Finnish.pdf

Shinde, V. 2019. SDLC (Software Development Life Cycle) Phases, Methodologies, Process, And Models. Viitattu 1.4.2020. Saatavissa: https://www.softwaretestinghelp.com/software-development-life-cycle-sdlc/

Tamro. 2020. Tamro. Viitattu 28.4.2020. Saatavissa: https://www.tamro.fi/fi/Tamro/Sivut/default.aspx

Tapio, T. 2010. Riskienhallinta perinteisessä ja ketterässä ohjelmistokehityksessä. Pro gradu -tutkielma. Tampereen yliopisto, tietojenkäsittelyoppi. Viitattu 3.4.2020. Saatavissa: https://trepo.tuni.fi/bitstream/handle/10024/81663/gradu04362.pdf?sequence=1

Tervonen, A. 2017. Ohjelmistokehitysmenetelmän valinta tuotekehitysprojektissa. Ylempi AMK-opinnäytetyö. Kajaanin ammattikorkeakoulu, teknologiaosaamisen johtamisen koulutusohjelma (Ylempi AMK). Viitattu 3.4.2020. Saatavissa: http://urn.fi/URN:NBN:fi:amk-201705249973

Tilastokeskus. 2019. Reliabiliteetti. Viitattu 18.4.2020. Saatavissa: https://www.stat.fi/meta/kas/reliabiliteetti.html

TouringCars. 2020. About Touring Cars motorhome rental. Viitattu 28.4.2020. Saatavissa: https://www.touringcars.eu/about-touring-cars-motorhome-rental

Velez, M. 2016. What Is Waterfall Methodology? Viitattu 5.4.2020. Saatavissa: https://castellansystems.com/kb/Waterfall.cshtml

Välisalo, T. 2015. Havainnointi eli observointi. Viitattu 17.4.2020. Saatavissa: https://koppa.jyu.fi/avoimet/hum/menetelmapolkuja/menetelmapolku/aineistonhankinta-menetelmat/havainnointi-eli-observointi-osallistuminen-ja-kenttaetyoe

West, D. 2018. Scrum roles and the truth about job titles in Scrum. Viitattu 13.4.2020. Saatavissa: https://www.atlassian.com/agile/scrum/roles

Kuviot