Jussi Kasala

OHJELMISTOTUOTANNON PROSESSIT JA MENETELMÄT

Tietojenkäsittelyn koulutusohjelma

2016

samk
Satakunnan ammattikorkeakoulu
Satakunta University of Applied Sciences

OHJELMISTOTUOTANNNON PROSESSIT JA MENETELMÄT

Kasala, Jussi
Satakunnan ammattikorkeakoulu
Tietojenkäsittelyn koulutusohjelma
Maaliskuu 2016
Ohjaaja: Nuutinen, Petri
Sivumäärä: 38
Liitteitä: 1

Asiasanat: ohjelmistotuotanto, ohjelmistokehitys, prosessit, menetelmä

Tämän opinnäytetyön tarkoituksena oli tutkia ohjelmistokehitykseen liittyviä prosesseja ja menetelmiä. Ohjelmistotuotanto on projektityötä, joten siihen liittyy väistämättä erilaisia vaiheita. Työn tavoitteeksi asetettiin saada yleispätevä kuva ohjelmistotuotannon prosesseista ja menetelmistä.

Työn teoriaosuudessa esiteltiin aihepiiriä lähteisiin perustuen. Ohjelmistotuotannon prosessit muodostuvat esitutkimuksesta, vaatimusmäärittelystä, suunnittelusta, toteutuksesta, testauksesta, käyttöönotosta ja ylläpidosta. Prosesseja hyödynnytetään erilaisin ketterin ja perinteisin menetelmin. Menetelmien avulla ohjelmistotyötä voidaan jaksottaa ja painottaa tarpeen mukaan.

Tutkimusosa käsitteli laadullisen tutkimuksen keinoin ohjelmistoyritysten tapaa hyödyntää teoriaosuudessa esiteltyjä prosesseja ja menetelmiä. Tutkimusosa suoritettiin teemahaastatteluna, johon osallistui ohjelmistoyrityksiä Satakunnan alueelta.

Tutkimuksen perusteella yritysten tapa toimia vaikuttaa, siihen miten teoreettisiltakin tuntuvia prosesseja sovelletaan. Prosesseista vaatimusmäärittely kuvattiin tärkeimmäksi prosessiksi projektin onnistumisen kannalta. Suosituimmiksi menetelmiksi osoittautuivat ketterät menetelmät ja Scrum-menetelmän muunnokset. Testaukseen ja loppukäyttäjän kouluttamiseen yritykset haluavat panostaa entistä enemmän.

SOFTWARE ENGINEERING PROCESSES AND METHODS

Kasala, Jussi
Satakunnan ammattikorkeakoulu, Satakunta University of Applied Sciences
Degree Programme in Business Information Technology
March 2016
Supervisor: Nuutinen, Petri
Number of pages: 38
Appendices: 1

Keywords: software engineering, software development, processes, methods

---

The purpose of this thesis was to investigate software engineering processes, methods and models. Software engineering is a project work, so there are various development phases. The goal was to receive a good overview of the software engineering.

The theoretical part introduces the topic and it is based on literary sources. The software engineering processes consists of preliminary studying, requirement specification, design, implementation, testing, deployment and maintenance. The development processes are used for traditional models and agile methods. The project team will be able to space out work processes with these methods and models.

The aim of the functional part was to find out how companies uses processes and methods for their projects. The research was a qualitative and it was theme interview for software engineering companies in the Satakunta area.

According to the theme interviews the way how companies operate affects to processes and methods. The main result was that the requirement specification is the most important process with agile methods. The companies want to invest more to testing and end-user training.

.

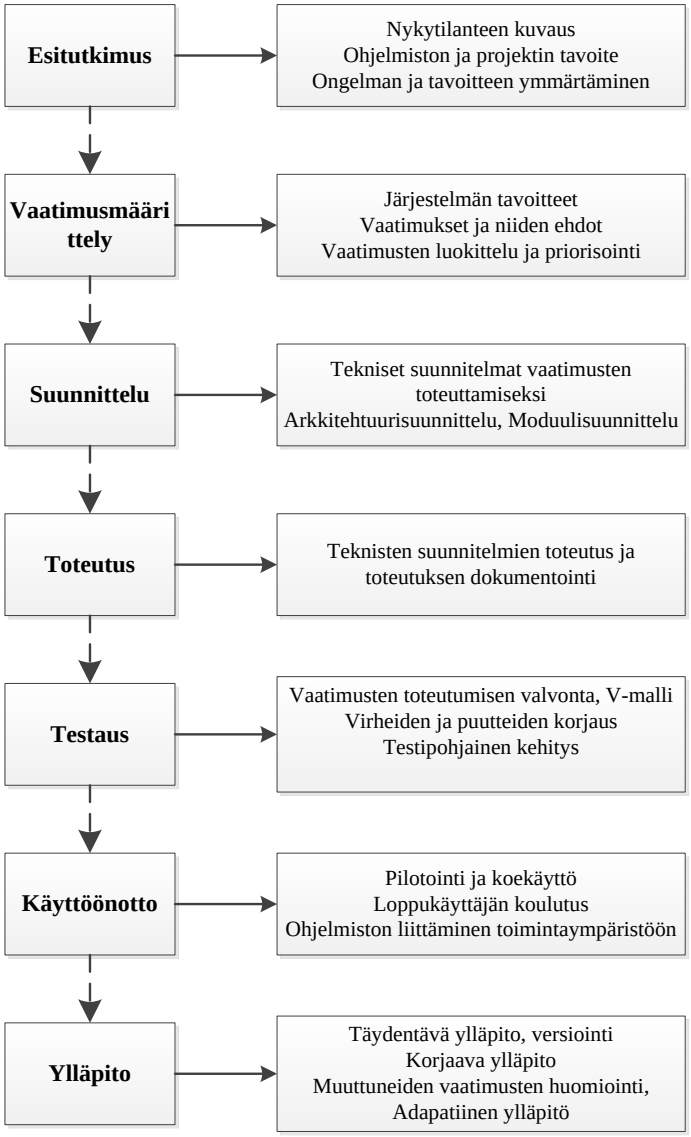
SISÄLLYS

1 JOHDANTO

Tässä työssä tarkastellaan ohjelmistokehitykseen liittyviä prosesseja ja menetelmiä. Teoriaosuudessa käsitellään prosesseja ja niihin liittyviä menetelmiä lähteisiin perustuen. Toiminnanlinen osa muodostuu tutkimuksellisesta osiosta, jossa tarkoituksena on selvittää haastattelulla Satakunnassa sijaitsevien ohjelmistoyritysten käyttämiä prosesseja ja menetelmiä. Aiheen rajaamiseksi työssä ei keskitytä ohjelmistoprojektien hallintaan tai tukitoimintoihin, vaan painopisteinä ovat tuotantoprosessit ja menetelmät.

Ohjelmistotuotanto on perusluonteeltaan projektityötä. Ohjelmistoprojekti on laaja kokonaisuus, joten siihen kuuluu väistämättä useita hahmotettavissa olevia prosesseja ja menetelmiä.

Parhaan lopputuloksen saamiseksi ohjelmistoprojekti jaetaan prosesseihin, joita hyödynnetään erilaisin menetelmin. Prosessit muodostavat ohjelmistotuotteen sekä ohjelmistoprojektin elinkaaren (Kuva 1). Menetelmät jaetaan perinteisiin ja ketteriin menetelmiin ja niillä voidaan jaksottaa kehitystyötä havaitun tarpeen mukaan. Prosessit ja menetelmät ovat saaneet vaikutteensa tuottavasta teollisuudesta, mutta ovat kuitenkin vakiinnuttaneet paikkansa nykyaikaisissa ohjelmistoprojekteissa.



Kuva 1. Ohjelmistotuotannon prosessit kuvattuna elinkaarimallin mukaisesti. (Haikala & Mikkonen 2011, mukailtuna.)

Ohjelmistoprojektin lähtökohtana on kehittää uutta tai ylläpitää vanhaa järjestelmää. Useassa ohjelmistoprojekteja koskevassa kirjallisuuslähteessä projekti kuvataan onnistuneeksi, kun se täyttää sille asetetut vaatimukset.

Tilaajan eli asiakkaan kannalta ohjelmistoprojektilla haetaan tuottavuusetuja liiketoimintaan. Ohjelmistoprojekti toimii osana suurempaa hanketta. Projektiorganisaation muodostavat tilaaja ja toimittaja.

2 OHJELMISTOTUOTANNON PROSESSIT

2.1 Esitutkimus

Esitutkimus on ohjelmistokehityksen ensimmäinen vaihe, mutta se ei aina johda projektin aloittamiseen. Tarpeen mukaan tutkimus voidaan jakaa kahteen osaan; vaatimusten ja tietojen keräämiseen tai toiminnan kartoittamiseen. (Vesterholm & Kyppö 2006, 52-53.)

Esitutkimuksen tarkoituksena on tuoda esille yleiset vaatimukset ohjelmistolle ja sen järjestelmätasolle. Esitutkimukset ovat usein luonteeltaan asiakasvaatimuksia kokoavia, jossa asiakas kuvaa järjestelmän tarpeen. Huomioitavaa on, ettei esitutkimuksen tarkoituksena ole ottaa kantaa, millainen järjestelmän tulisi olla. (Haikala & Märijärvi 2004, 37.)

Esitutkimus sisältää asiakkaan tietojärjestelmien nykytilanteen kuvauksen, jos se liittyy olennaisesti kehitettävään järjestelmään. Lisäksi tuodaan esille, mihin ongelmaan tulevaa järjestelmää tullaan käyttämään. Tutkimuksen lopuksi laaditaan alustava suunnitelma projektin etenemisestä. Tutkimustuloksena syntyy hahmotus siitä, että onko ohjelmistoprojekti kokonaisuudessaan kannattava ja luodaan lähtökohdat projektin aloittamiselle. (Pohjonen 2002, 27-28.)

Tilaaja on sitoutettava projektiin jo esitutkimusvaiheessa ja projektitiimin on perehdyttävä tutkimustuloksiin huolellisesti. Tutkimustulosten perusteella projektia räätälöidään tilaajan haluamaan suuntaan. (Kalliala 2010.)

Pienissä ohjelmistoprojekteissa esitutkimus on osana määrittelyvaihetta, koska asiakkaan tarpeiden analysointi jatkuu koko projektin ajan. Laajoissa projekteissa tutkimus toimii yhtenä itsenäisenä osana. (Haikala & Märijärvi 2004, 37.)

2.2 Vaatimusmäärittely

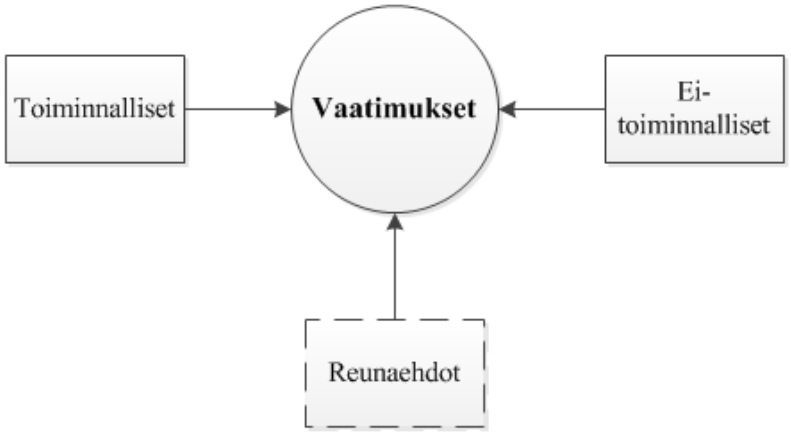
Ohjelmistokehityksessä on tyypillistä, että yritysasiakas tilaa käyttöönsä sovelluksen osan tai kokonaisen tietojärjestelmän. Tilaussopimuksella solmitaan yhteistyö siitä, mitä ollaan toteuttamassa. Tähän liittyvää vaatimusten kokonaisuutta kutsutaan vaatimusmäärittelyksi. (Wiio 2004, 67.)

Vaatimusmäärittelyssä kuvataan järjestelmälle asetetut tavoitteet. Tätä vaihetta voidaan pitää ohjelmistoprojektin onnistumisen kannalta merkityksellisimmistä vaiheista. Tärkein kysymys onkin, mitä ohjelman pitäisi tehdä. Vaatimusten tavoitteena on tuoda tietoa ohjelmiston rakenteesta sekä toiminnoista. Asiakkaan roolia tässä projektivaiheessa ei voi aliarvioida. Vaihe ei ota kantaa, miten esille tulleet vaatimukset pitäisi toteuttaa. (Vesterholm & Kyppö 2006, 53-54.)

Vaiheeseen osallistuvat projektipäällikkö, tilaajan edustajat, prosessien ja tietohallinnon asiantuntijat. Tarvittaessa vaiheeseen osallistuu myös ulkopuolisia asiantuntijoita, jos sopivia henkilöitä ei löydy projektitiimistä. Määrittelyn päätteeksi tilaaja tarkastaa ja hyväksyy vaatimukset. (Murch 2002, 87; Julkisen hallinnon suositukset, 2012.)

2.2.1 Vaatimusten luokittelu

Hyvä vaatimus on tarkka, ymmärrettävä, testattava ja jäljitettävissä oleva. Vaatimuksen toteutumista voidaan mitata, kun on mahdollista selvittää vaatimuksen alkuperä ja kohde (Haikala & Mikkonen 2011, 64). Vaatimukset voidaan luokitella kolmeen pääluokkaan toiminnallisiin, ei-toiminnallisiin ja reunaehtoihin (Kuva 2).



Kuva 2. Vaatimukset muodostuvat toiminnallisista, ei-toiminnallisista vaatimuksista ja reunaehdoista. (Haikala & Märijärvi 2004, mukailtuna.)

Toiminnalliset vaatimukset ovat tyypiltään asiakasvaatimuksia tai teknisiä ohjelmistovaatimuksia. Tavoitteena on kuvata se, mitä ohjelmiston tulisi tehdä ongelman ratkaisemiseksi ja mitä valmiilta tuotteelta odotetaan. Toiminnallisten vaatimusten yhteydessä kuvataan tiedot ominaisuuksista, käyttöliittymästä ja rajapinnoista (Haikala & Märijärvi 2004, 40.)

Ei-toiminnalliset vaatimukset ovat usein järjestelmävaatimuksia eli toimintaympäristön luomia vaatimuksia, joita toiminnallisten vaatimusten on noudatettava. Ei-toiminnallisia vaatimuksia ovat esimerkiksi suuren käyttäjämäärän huomioiminen web-sovellusta kehitettäessä tai muistikapasiteetin rajoitukset sulautetuissa järjestelmissä. (Haikala & Märijärvi 2004, 40; Julkisen hallinnon suositukset, 2012.)

Reunaehdot muodostavat kolmannen kategorian, johon vaatimuksia luokitellaan. Reunaehdoissa esitetään muita havaittuja ehtoja, kuten järjestelmään liittyvät lakinormit ja tarkemmat tekniset vaatimukset. Ehdot muodostuvat usein tietojärjestelmään ja tietojen käsittelyyn liittyvistä seikoista. (Haikala & Märijärvi 2004, 40; Julkisen hallinnon suositukset, 2012.)

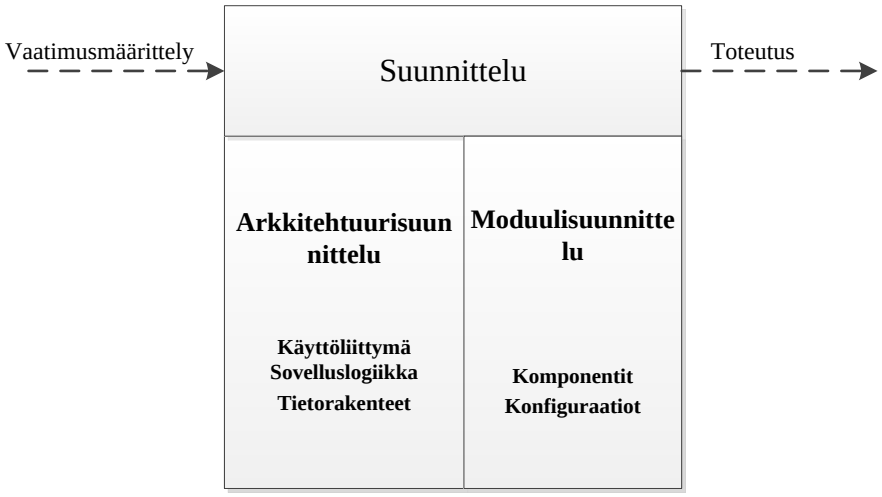
2.2.2 Vaatimusmäärittelyn vaiheet

Vaatimusten kokoaminen muodostuu valmistautumisesta, tuottamisesta sekä hyväksymisestä. Prosessin taustalla on esitutkimuksen perusteella saadut kokonaisuudet. Valmistautumisessa tavoitteet tarkistetaan ja projektin kulku suunnitellaan. Tuottamisessa järjestelmän tarpeet kuvataan ja ne luokitellaan. Lopuksi molemmat tilaaja ja toimittaja hyväksyvät määrittelyn tuloksen. Tavoitteena on varmistaa vaatimusten laatu ja oikeellisuus. (Julkisen hallinnon suositukset, 2012.)

Jokaisessa vaiheessa tiedon kerääminen on tärkeässä roolissa ja vaatii hyviä kommunikaatiotaitoja. Tietoa kerätään haastattelemalla, havainnoimalla sekä tutkimalla aikaisempia dokumentteja. Kerätty tieto mallinnetaan ja jalostetaan seuraavaan prosessiin sopivaksi. (Kaskela 2005.)

2.3 Suunnittelu

Suunnitteluprosessissa määritellään vaatimusten tekniset suunnitelmat eli miten ohjelman tulisi toimia (Haikala & Märijärvi 2004, 40). Vaatimusmäärittely ja suunnittelu voivat olla myös osittain päällekkäisiä eli toisiaan tukevia, mutta suunnittelu nähdään usein itsenäisenä kokonaisuutena (Kuva 3). Kun vaatimusten kerääminen on toteutettu huolellisesti, voidaan siirtyä suunnitteluvaiheeseen.



Kuva 3. Suunnitteluprosessiin sisältyvät suunnitelmat arkkitehtuurista ja moduuleista. (Haikala & Märijärvi 2004, 82 mukailtuna.)

2.3.1 Arkkitehtuurisuunnittelu

Arkkitehtuurisuunnittelun tarkoituksena on jakaa kehitystyö osa-alueisiin. Kerrosajattelun mukaisesti toteutettava sovellukseen kuuluu käyttöliittymä-, sovelluslogiikka sekä tietojensäilytyskerros.

Käyttöliittymän suunnittelulla pyritään ohjelman helppokäyttöiseen ja tarkoituksenmukaiseen toimintalogiikkaan. Toimintalogiikka kuvataan suunnitteluvaiheessa näyttötauluina ja taulujen siirtymäketjuina. Käyttöliittymän toiminnalla ja sen huolellisella suunnittelulla on merkitystä moneen osa-alueeseen, kuten järjestelmän virheettömään toimintaan, työtehokkuuteen ja käyttömukavuuteen. (Virkki & Somermeri 1999, 79; Vesterholm & Kyppö 2006, 188.)

Käyttöliittymän visuaalinen ilme on ensimmäinen konkreettinen asia, jonka asiakas kehitettävässä järjestelmässä kohtaa. Tästä voi syntyä asiakkaalle mielikuva, että ohjelmisto on valmis, vaikka projekti on vasta suunnitteluvaiheessa. Epäyhtenäinen toimintalogiikka ja visuaalisesti sekava näyttö tuo asiakkaalle negatiivisen mielikuvan ohjelmiston laadusta. Käyttöliittymän suunnitteluvaiheessa tärkeintä on kuvata jokin osa tai kokonaisuus tulevasta järjestämästä. Tulokset arvioidaan ja testataan. (Wiio 2004, 217.)

Sovelluskerroksessa suunnitellaan ohjelmiston toimintalogiikka ja sovellukseen liittyvät ominaisuudet. Vaatimusmäärittelyn toiminnalliset vaatimukset saavat tekniset toteutussuunnitelmat, joilla haluttu toiminta toteutetaan. Määrittelydokumentissa on kuvattu se, mitä sovelluksen halutaan tekevän ja suunnitteluvaiheessa miten toiminta toteutetaan. (Haikala & Märijärvi 2004, 39-40; Haikala & Mikkonen 2011, 188.)

Tietojensäilytyskerroksessa eli tietokantasuunnittelussa kuvataan tietokannan taulut ja relaatiot. Tavoitteena on suunnitella tehokkaasti toimiva tietokanta, jossa ei ole esimerkiksi tarpeetonta tiedon toistoa. Suunnittelun tavoitteena on saada dokumentit tietokantaratkaisusta. Suunnittelussa käytetään apuna ER-kaavioita tietorakenteista. Vaikka yksinkertainen sovellus ei tarvitsisi omaa tietokantaa, on liityntä muihin tietokantoihin suunniteltava. (Haikala & Märijärvi 2004, 117-118; Vesterholm & Kyppö 2006, 64-65.)

2.3.2 Moduulisuunnittelu

Moduulisuunnittelu seuraa arkkitehtuurisuunnittelun jälkeen ja sen tarkoituksena on ohjelmiston yksittäisen rakenteen jakaminen pienempiin noin 1000 koodirivin lohkoihin. Lohkot muodostavat ohjelmiston itsenäisesti toimivat komponentit ja konfiguraation. Jokainen komponentti suunnitellaan vastaamaan vaatimuksia ja arkkitehtuuriratkaisua. (Haikala & Märijärvi 2004, 81-83.)

Moduuleihin jaon tavoitteena on jakaa toteutustyö hallittaviin kokonaisuuksiin, jotka toteutetaan erillään muista osista. Jos muutoksia joudutaan tekemään, tehdään se vain yhteen moduuliin. Projektitiimi tarkastaa suunnitelmat ja viimeistelee määrittelyn toteutettavaksi. (Murch 2002, 100; Haikala & Märijärvi 2004, 81.)

2.4 Toteutus

Vaatimusmäärittelyn ja suunnittelun merkitys korostuvat toteutusvaiheessa, koska heikosti suunnittelun ohjelmiston vaatimukset ovat haastavia toteuttaa. Toteutuksessa ohjelmoidaan määritetyllä ohjelmointikielellä ja sovelluskehittimellä. Vaihe voidaan toteuttaa moduuli kerrallaan ja lopuksi koota yhdeksi kokonaisuudeksi. Toteutusvaihe on suoraviivaista ohjelmistotyötä, kun voidaan ohjelmoida suoraan suunnitelmadokumenttien mukaan. Esimerkiksi olio-ohjelmoinnissa luokkakaavion pohjalta muodostetaan olioluokat. (Pohjonen 2002, 34–35; Vesterholm & Kyppö 2006, 77.)

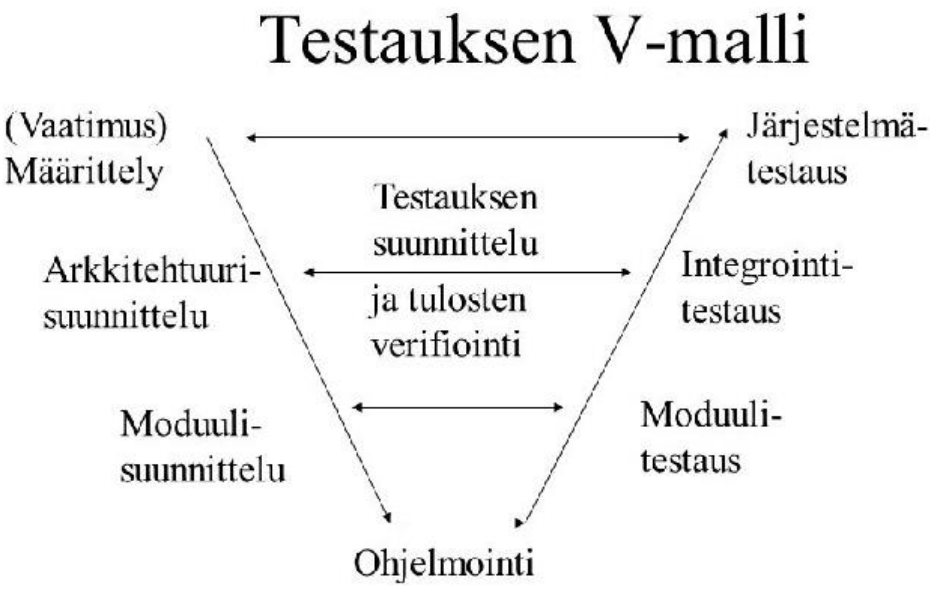
Päämääränä on valmistaa suunnitteluvaiheen mukainen järjestelmä. Dokumentaatio on tärkeä osa toteutusvaihetta, jotta projektitiimi tietää, mitä on tehty ja miksi ratkaisu on tehty. Päävastuu on ohjelmoijalla, mutta dokumentointiin voi osallistua myös muita projektitiimin jäseniä. Hyvin dokumentoitu ohjelmointivaihe helpottaa virheiden korjaamista, mikäli niitä havaitaan. Yksinkertaistettu toteuttamisvaihe on puhtaasti mekaanista työtä suunnitelman mukaan. Mekaanista työtä voidaan automatisoida skripteillä, mutta pääosin toteutusvaihe on ohjelmoijan työtä. (Ruuska 1999, 23; Virkki & Somermeri 1999, 90-91; Vesterholm & Kyppö 2006, 77.)

2.5 Testaus

Testauksen päätarkoituksena on paikantaa virheet, joita ohjelmasta löytyy. Yksinkertaisin testausmuoto on yksikkötestaus, jossa koodaaja testaa oman tuotoksensa virheiden varalta. Testattavana on 100-1000 koodiriviä kerrallaan ja kohteena voi olla esimerkiksi metodin toiminta. Yksikkötestauksessa tulosta verrataan suunnitteludokumenttiin. Monimutkaisempi muoto tästä on järjestelmätestaus, jossa riippumaton testaaja suorittaa testauksen. Vaiheessa tarkistetaan ohjelmiston virheettömyys järjestelmän, halutun toiminnan ja teknisten vaatimusten valossa. Integrointitestauksessa painopisteen muodostavat järjestelmän rajapintojen toimivuus ja yhteneväisyys arkkitehtuurisuunnittelun kanssa. (Ruuska 1999, 23; McConnell 2002, 72-73.)

2.5.1 Testauksen V-malli

Tuotantoprosessien ja testauksen välistä yhteyttä kuvataan V-mallilla (Kuva 4). Malli kuvaa visuaalisesti testauksen ja prosessien tasot V-kirjaimen muodossa. Testaus tapahtuu sitä vastaavan työvaiheen kanssa erottelemalla testaustasot. Yksikkötestauksessa testauskohteena on yksittäinen luokka, metodi tai luokkien muodostama kokonaisuus. Luokkatoimintaa verrataan tekniseen suunnitteludokumenttiin. (Haikala & Mikkonen 2011, 206-207.)



Kuva 4. Testauksen V-mallin vaiheet (Haikala & Märijärvi 2004, 289.)

2.5.2 Testivetoinen suunnittelu

Testivetoisen suunnittelun (Test-Driven Development, TDD) perusajatuksena on toteuttaa testitapaukset eli testipatteristo ennen kuin ohjelmiston ominaisuutta toteutetaan. Perinteisesti V-mallin mukaan ohjelmisto testataan projektin loppuvaiheessa, mutta TDD keskittyy testaukseen kaikissa prosessivaiheissa. Testivetoisen suunnittelun hyötynä on jatkuva testaus, jolloin ohjelmakoodi toimii saumattomasti muiden ohjelmistomoduulien ja järjestelmien kanssa. (Haikala & Mikkonen 2011, 214-216.)

Testivetoisen suunnittelun alkuvaiheessa uutta ominaisuutta vastaavat testitapaukset lisätään patteristoon. Seuraavaksi suoritetaan testipatteristo ja tarkistetaan onnistuuko ohjelmakoodi läpäisemään testit. Toteutuskoodin ei tarvitse alkuvaiheessa olla virheetön, koska muokkausta tehdään koko prosessin ajan. Kuitenkin uuden muokatun koodin on läpäistävä myös aikaisemmat testit. Testipatteriston avulla ohjelmistokoodien kokonaisuus pysyy määrittelyn mukaisena. (Haikala & Mikkonen 2011, 214-216.)

Testivetoisen kehityksen etuna on varmuus oikeasta ja virheettömästä toiminnasta kohdistuen muihin ohjelmistomoduuleihin, järjestelmiin tai laitteistoihin. Prosessi voidaan myös automatisoida järjestelmän hoidettavaksi, koska samat testit toistuvat koodin muutoksissa. Testivetoista suunnittelua käytetään etenkin sulautettujen järjestelmien kehityksessä. (Lehtonen 2014, 63-64.)

2.6 Käyttöönotto

Käyttöönotolla tarkoitetaan kokonaisuutta, jossa kehitetty ohjelmistotuote toimitetaan tai julkaistaan. Käyttöönoton tehtävänä on käynnistää järjestelmän toiminta tilaajan ympäristössä. Vaiheen päätteeksi tilaaja hyväksyy toimituksen. Tässä vaiheessa myös sovitaan ylläpidollisista seikoista, jollei niistä ole aikaisemmin sovittu. Valmis ohjelmisto voidaan myös julkistaa suurelle asiakasryhmälle, jos tuotteella ei ole määriteltyä tilaajaa. (Ruuska 1999, 23; Kettunen & Simons 2001, 27.)

Käyttöönotto voidaan toteuttaa monella tapaa, mutta siihen kuuluvat koekäyttö eli pilotointi, loppukäyttäjän koulutus ja varsinainen käyttöönoton osuus (Kuva 5).

Kuva 5. Tyypillisen käyttöönottoprosessin tehtävät.

Pilotoinnin tavoitteena on saada kokemusta todellisesta käytöstä ja sopivuudesta käyttötarkoitukseen. Huomiota kiinnitetään käyttöliittymän toimivuuteen, toiminnan tehokkuuteen ja käyttökokemukseen. Pilottijaksolta kerätään tietoa tilaajalle ja toimittajalle, jotta hallittu käyttöönotto olisi mahdollista suorittaa. (Lahden ammattikorkeakoulun julkaisu, 2013.)

Tilaajan koulutus on yksi käyttöönoton tehtäväkokonaisuuksista. On selvää, että tuntematonta ohjelmaa käyttävältä kuluu jonkin tehtävän suorittamiseen enemmän aikaa kuin kokeneelta ammattilaiselta. Koulutus tulee suunnitella oppimisprosessia tukevaksi, koska ilman tehokasta ja kattavaa koulutusta tilaaja ei pysty toimimaan tehokkaalla ohjelmiston mahdollistavalla tavalla. Asiakas haluaa ohjelmalta yritykseensä tuottavuusetuja tai ratkaisun kohdattuun ongelmaan. Julkistettuun ohjelmistoon tarjotaan kaupallista koulutusta. (Virkki & Somermeri 1999, 94; McConnell 2002, 358-359.)

Toimivan ja tehokkaan ohjelmiston käyttöönoton edellytyksenä on osaava loppukäyttäjä. Koulutuksen tarkoituksena on perehdyttää koko asiakkaan henkilöstö tai osa henkilöstöstä ohjelman käyttöön. Jo toteutusvaiheessa voidaan antaa perehdytystä esimerkiksi ohjelmiston pääkäyttäjille. Koulutus ja toimintatavan kehitys perustuu organisaation ja työntekijöiden oppimiseen, jossa tekninen toteutus on osa kokonaisuutta. (Virkki & Somermeri 1999, 94; Kettunen & Simons 2001, 20-21.)

Tarkoituksena on myös huolehtia, että kaikki ohjelmistoon liittyvät laitteistot, tietoverkot ovat uuden systeemin vaatimassa kunnossa. Tavoite on sovittaa järjestelmä osaksi muuta tuotantoympäristöä. Ympäristö muodostuu kokonaisuudesta, jossa uuden tuotteen on toimittava. (Virkki & Somermeri 1999, 98–99.)

Käyttöönotto seuraa hyväksytyn testausvaiheen jälkeen. Epäonnistuneen ohjelmistoprojektin tunnusmerkkeihin kuuluvat ohjelman heikko käyttöönotto. Tavoitteet eivät aina ole selvillä, joten vaihe tulee suunnitella ja dokumentoida. Prosessi täyttää tavoitteensa silloin, kun käyttöönottovaihe sujuu joustavasti asiakkaan sekä tuotantotiimin kannalta. Kuitenkaan vaihe ei käytännössä etene suoraviivaisesti, joten ongelmat ja puutokset tulee huomioida. Mittavissa projekteissa käyttöönotto kestää useita vuosia ja siihen osallistuu myös kolmansia osapuolia. Vaihe voi epäonnistua, jos asiakas ei käytätätkään ohjelmaa tai prosessia ei dokumentoida esimerkiksi asiakasyrityksen tietohallinnon käyttöön. (Vesterholm & Kyppö 2006, 76-77; Julkisen hallinnon suositukset 2012.)

2.7 Ylläpito

Ylläpitoon kuuluvat asiakkaan havaitsemien ongelmien ratkominen sekä havaittujen virheiden korjaaminen. Myös osaltaan asiakkaan vaatimusten muuttumisen myötä tehtävät korjaukset ovat ylläpitoa. (Haikala & Märijärvi 2004, 41.)

Ylläpito jaetaan kolmeen osa-alueeseen: täydentävään, korjaavaan tai adaptiiviseen alueeseen. Laajassa ohjelmistoprojektissa kohtaa väistämättä kaikki muodot. Täydentävässä ylläpidossa tavoitteena on ohjelmiston eliniän nostaminen, joko muuttamalla kokonaisuudessaan ohjelmistoa esimerkiksi tuottamalla uusi versio tuotteesta tai sitten lisäämällä valmiiseen tuotteeseen lisätoiminnallisuutta. Korjaavassa ylläpidossa paikataan todellisessa käytössä havaittuja virheitä. Jos taustalla on muuttuneet vaatimukset tai käyttöympäristö, ohjelmaan odotettavista muutoksista käytetään nimitystä adaptiivinen ylläpito. (Haikala & Märijärvi 2004, 41-42.)

Ylläpidolla on suuri merkitys ohjelmiston elinkaaressa. Jos koko ohjelmistoprojektin aikana ylläpidolliset seikat on huomioitu, voi ylläpito olla pitkäkestoinen ja ohjelmistoyritykselle tuottava prosessi. Suppealla suunnittelulla ja dokumentaatiolla ylläpito voi olla resursseja tuhlaavaa sekä ohjelmistoprojektia pitkittävä vaikutus. Markkinointikeinoja käytetään ylläpidon houkuttimena, kun järjestelmä on asiakkaalle toimitettu. (Vesterholm & Kyppö, 2006, 77.)

Muutamia ohjelmistoratkaisuita tarjoavat ohjelmistotalot voivat keskittyä ylläpidossa tuotteiden jatkokehitykseen. Tarkoituksena on tällöin tuottaa vanhenevasta järjestelmästä uusi päivitetty versio, jossa voi olla esimerkiksi uusia ominaisuuksia.

3 MENETELMÄT JA MALLIT

3.1 Koodaa ja korjaa

Koodaa ja korjaa -malli on silloin käytössä, kun mitään muuta mallia tai menetelmää ei tietoisesti käytetä. Menetelmä perustuu näkemykseen siitä, millainen tulevan ohjelmiston tulisi olla. Projektia viedään kohti haluttua lopputulosta. (McConnell 2002, 140-141.)

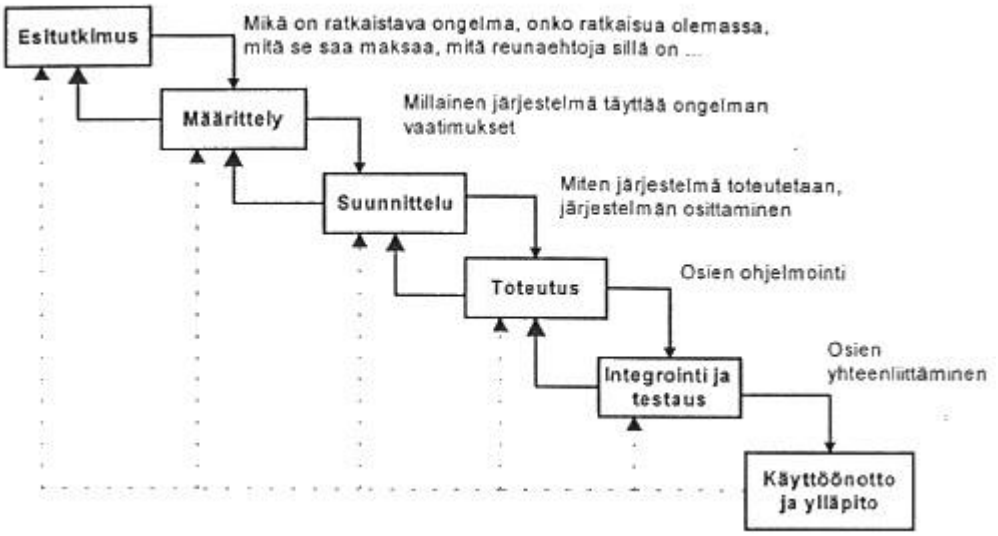
Etuna mallissa on se, että vältetään aikaa vievät työvaiheet, kuten suunnittelu, dokumentointi ja laadunvarmistuksen prosessit. Tällöin projektiin saadaan tuotettua nopealla aikataululla puhdasta ohjelmakoodia. Menetelmä soveltuu parhaiten tukiprojektiksi, jossa projektitiimi voi testata ohjelmiston ominaisuuksia. (McConnell 2002, 140-141.)

Haasteellista mallissa on ohjelmakoodin muuttaminen, koska sitä ei ole suunnitelmallisesti jäsennelty tai dokumentoitu. Virheiden ja toiminnan korjaamiseen saattaa lopulta kulua lopulta enemmän työtä, kuin varsinaisen ohjelmistokoodin kirjoittamiseen. Menetelmää ei sovelleta muodollisissa projekteissa, mutta sillä voidaan tuottaa nopeasti tutkittavia ominaisuuksia demoprojektin muodossa. (McConnell 2002, 140-141.)

3.2 Vesiputousmalli

Vesiputousmallin vaiheisiin kuuluvat vaatimusmäärittely, suunnittelu, toteutus, testaus, asennus ja ylläpito. Mallista on kehitetty useita versioita, mutta kaikista versioista on hahmotettavissa määrittely-, suunnittelu ja toteutusvaiheet (Kuva 6). Eteneminen menetelmissä tapahtuu järjestyksessä ja jokainen prosessi suoritetaan loppuun ennen

seuraavaan vaiheeseen siirtymistä. (Royce 1970; Haikala & Märijärvi 2004, 37.) Alkuperäisestä vesiputousmallista on useita versioita riippuen projektitiimin koosta ja projektin luonteesta.



Kuva 6. Vesiputousmallin vaiheistus ohjelmistoprojektin prosessien avulla. (Haikala & Märijärvi 2004, 36.)

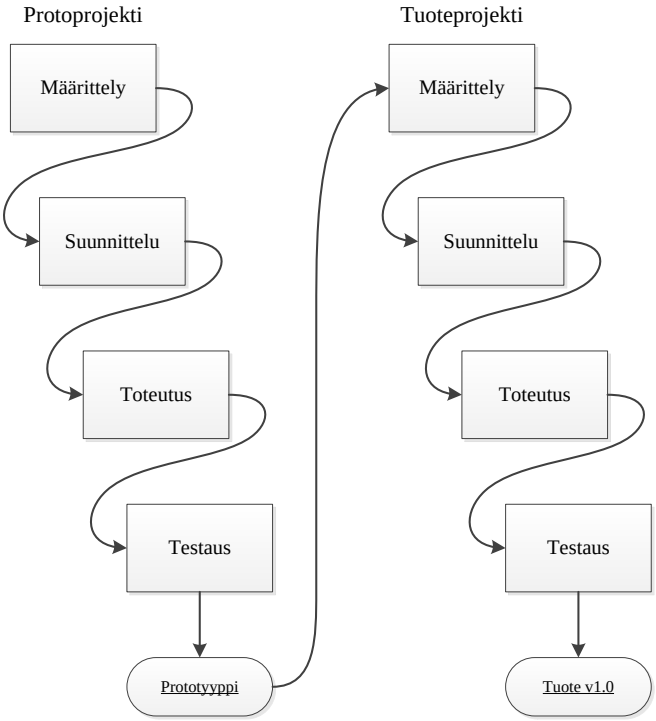
Kaikki muut menetelmät ovat ottaneet vaikutteita vesiputousmallista. Vesiputousmalli onkin yksi ensimmäisistä ja tunnetuimmista kehityksen malleista. Kehitysprojekti kulkee läpi kaikki ohjelmistotuotannon prosessit. Lisäksi vesiputousmalliin sisältyy laadunvarmistuksen tehtävät kuten tarkastukset, katselmukset ja testaukset. Käytännössä täysin puhdasta vesiputousmallia on haasteellista toteuttaa. (McConnell 2002, 137.)

Dokumentointi kuuluu vahvasti vesiputousmallin toteuttamiseen. Jokaisen menetelmän esimerkiksi testauksesta tai määrittelystä luodaan hyväksytty dokumentti. Laaja ja raskas dokumentointi voi kuitenkin tuhlata turhaan resursseja pientä ohjelmistoprojektia kohden.

3.3 Prototyyppimalli

Prototyyppimallinnuksella tarkoitetaan työskentelytapaa, jolla saadaan vaatimukset ja ongelmat ratkaistua prototuotteella. Proton toteuttamisessa käytetään ohjelmointikieltä ja kehittämismenetelmiä. Valmiin proton hyväksyttyjä ominaisuuksia hyödynnytetään tulevassa tuotteessa (Kuva 7). Huonoiksi todetut ratkaisut jätetään pois ja syyt

poistamiseen perustellaan. Protoilu liittyy myös suunnitteluvaiheeseen esimerkiksi käyttöliittymäsuunnitteluun, jossa ohjelmiston käyttöliittymää mallinnetaan kohti haluttua lopputulosta. (McConnell 2002, 569; Haikala, & Mikkonen 2011, 39.)



Kuva 7. Prototyypin kehitys valmiiseen tuoteversioon asti. (Haikala & Märijärvi 2004, 42.)

Prototyyppi voi olla myös poisheitettävä, jolla kuvataan tuotteen ominaisuuksia. Poisheitettävä prototyyppi ei tule olemaan osa valmista järjestelmää. Tämän tyyppinen protoilu toimii esimerkiksi ohjelmiston vaatimuksia hahmoteltaessa. (McConnell 2002, 569; Haikala & Mikkonen 2011, 38-39.)

Sovelluskehittimet tarjoavat luontevan tavan protomallinnukseen, koska esimerkiksi lomakejärjestelmän tietokantoineen saa mallinnettua vaivattomasti. Ongelmana protoprojekteissa on liiallinen koodin kierrättäminen ja valmiiden tuntemattomien ratkaisujen käyttö. Lopulliseen tuoteversioon voi tällöin jäädä tahattomia virheitä, joiden ratkaiseminen on haastavaa dokumentaation puuttumisen vuoksi. (Wiio 2004, 64-65.)

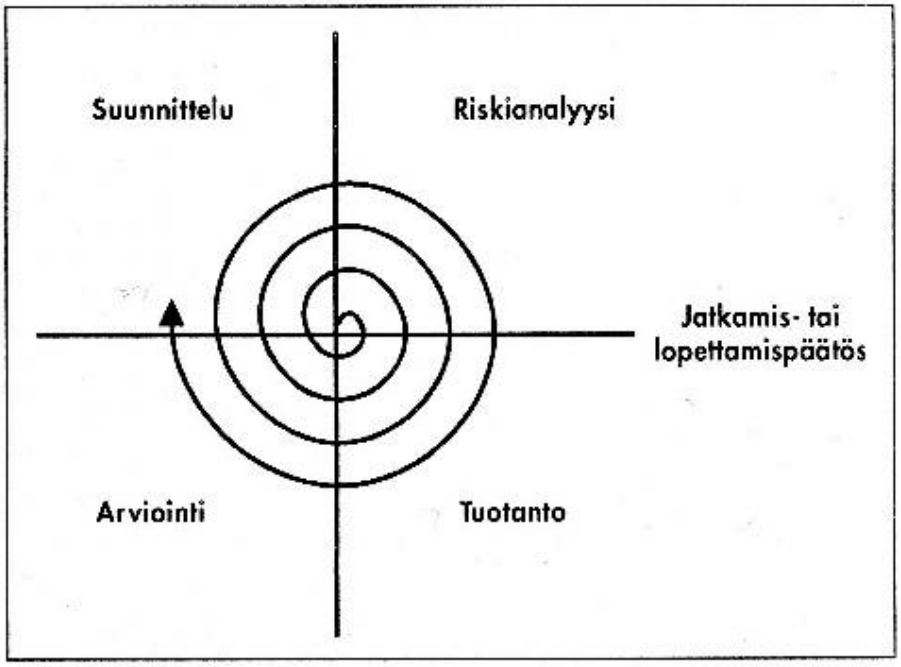
Tilaajan kannalta keskeneräisen prototuotteen perustelu on hankalaa, koska asiakas voi prototuotteen nähtyään ajatella järjestelmän olevan käyttövalmis. Perustelemalla

se, että työprosessi on kesken ja näyttämällä vielä puutokset voidaan tältä välttyä. (Haikala & Mikkonen 2011, 39.)

3.4 Spiraalimalli

Spiraalimalli on kehysviite, jossa suunnittelu ja varsinainen kehitysprosessi vuorottelevat. Jokaisella spiraalin kierroksella ohjelmisto laajenee ja yhteen kierrokseen kuuluvat kaikki ohjelmistotuotannon prosessit suunnittelusta ylläpitoon. (Manninen 2007.)

Spiraalimallin tarkoituksena on joka kierroksella selvittää mahdolliset etenemistä haittaavat riskit analyysin avulla (Kuva 8). Kun päätös jatkamisesta on tehty, voidaan edetä projektissa eteenpäin. Ohjelmistoprojekti kulkee spiraalin keskipisteestä edeten ulkoreunaa kohti. Ohjelmistoprojekti on valmis, kun se saavuttaa spiraalin ulkokehällä olevan päätepisteen. (McConnell 2002, 141.)



Kuva 8. Yksinkertaisen spiraalimallin vaiheet. (Pohjonen 2002, 42.)

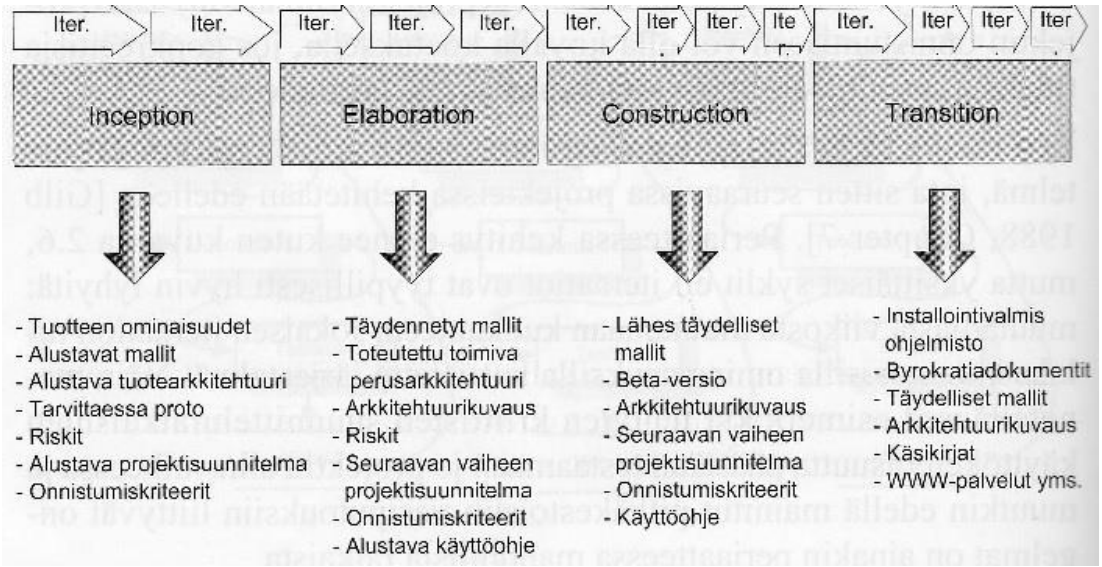
Malli jakaa projektin pienempiin osaprojekteihin, jotka seuraavat toisiaan. Kehitys aloitetaan spiraalin keskustasta edeten kohti ulkoreunaa. Etuina ja perusideana kyseisessä mallissa on riskienhallinta, koska jokaisella kierroksella tulkitaan tehtyyn työhön

liittyvät ongelmat. Malli soveltuu projekteihin, jossa halutaan keskittyä tarkasti riskien arviointiin. Projektissa voidaan painottaa osa-alueita havaitun tarpeen mukaan. (McConnell 2002, 141-142.)

3.5 Rational Unified Process (RUP)

Spiraalimallin kehityksenä muodostui RUP-kehysviite. RUP on jaettu neljään osaan aloittamiseen, tarkentamiseen, rakentamiseen ja siirtymiseen (Kuva 9). Malli perustuu johdonmukaisiin ja peräkkäisiin iteraatioihin eli useaan kertaan tapahtuvaan toistoon. (Haikala & Mikkonen 2011, 42-43.)

Aloitusvaiheessa suunnitellaan tuotteen erilaisia ominaisuuksia ja vaihtoehtoisia malleja. Tarkentamisessa suunnitellut mallit täydentyvät ja vaiheen lopputuloksena syntyy perusarkkitehtuuri. Rakentamisvaiheessa järjestelmästä luodaan jokaisella iteraatiolla uusi versio. Siirtymisvaiheen tuloksena on asennusvalmis järjestelmä käyttöohjeineen ja ylläpitopalveluineen. (Haikala & Märijärvi 2004, 47.)



Kuva 9. RUP-mallin eteneminen vaiheittain (Haikala & Märijärvi 2004, 46.)

Aloitusvaiheessa projektin ydintavoite määritellään ja vaiheeseen sisältyy myös vaatimusten kerääminen korkealla tasolla. Vaiheessa laaditaan yleiset vaatimukset tuotteelle ja sen ominaisuuksille. Lopuksi projektista on hahmotettu vaatimukset, aikataulu sekä riskit. (IBM 2001.)

Tarkentamisessa analysoidaan aloitusvaiheessa havaitut ongelmat, jotka korjataan. Tarkentamisessa vaatimuksia kerätään lisää ja ne arvioidaan. Vaiheeseen sisältyy arkkitehtuurin määritykset, kuten toiminnalliset ja ei-toiminnalliset vaatimukset. Tarkentamisvaihe on projektin kannalta tärkeimmässä roolissa, koska se sisältää vaatimusten arvioinnin. (IBM 2001.)

Rakentamisvaiheessa kaikki komponentit ja ominaisuudet toteutetaan ja testataan. Vaiheen lopuksi komponentit integroidaan keskitetysti lopulliseen tuotteeseen. Ohjeistus ja siihen liittyvä materiaali laaditaan ennen seuraavaan vaiheeseen siirtymistä. Rakennusvaiheessa voidaan toteuttaa ylläpitoon liittyvä toimintoja ja tehtäviä, kuten testikäyttöä ja pilotointia. (IBM 2001.)

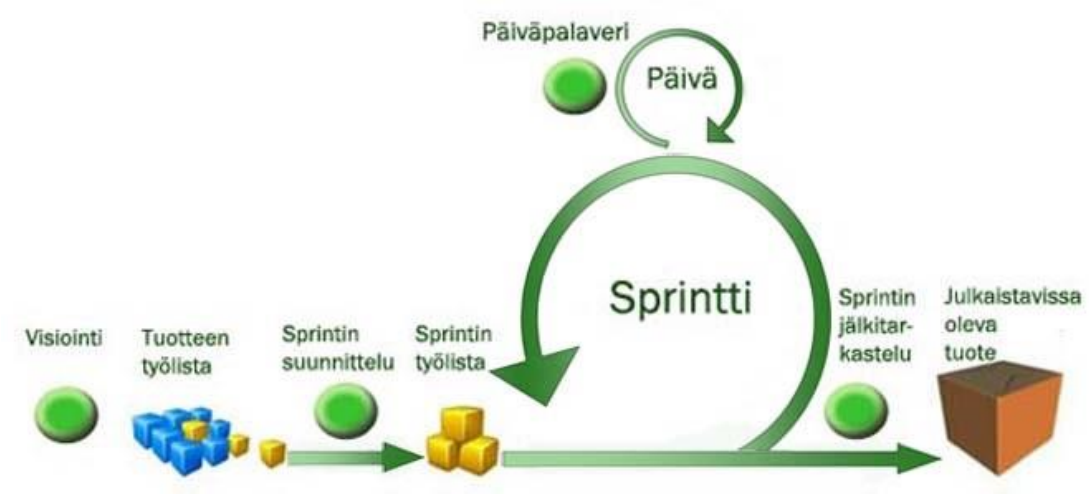
Siirtymävaihe siirtää tuotetun järjestelmän tilaajalle. Projektin tuloksena on asennusvalmis järjestelmä. Toimituksen ja asennuksen jälkeen voidaan keskittyä ylläpitoon. Siirtymävaiheessa kerätään tietoja kokemuksista ja ongelmatilanteista loppukäyttäjältä. (IBM 2001; Haikala & Märijärvi 2004, 47.)

3.6 SCRUM

Scrum on ketterä projektimalli, joka on ottanut käyttöönsä vesiputousmallin ajatuksen. Menetelmä sisältää usean vesiputouksen sarjan, jotka ovat seurausta edellisestä putouksesta. Toistettavuus ja lisäävä toimintapa muodostavat projektimallin rungon. (Haikala & Mikkonen 2011, 37.)

Scrum on nykyisin yleisin käytössä oleva ketterä tuotantomenetelmä, jonka tarkoituksena on monimutkaistenkin tuotteiden kehittäminen osissa. Alkuperäiseen ajattelumalliin harva tiimi kuitenkaan yltää. Scrum tuo ohjelmistoprojektiin ihmislähtöisen kokonaiskuvan, koska ongelmaa lähestytään roolijaon avulla. Lisäksi Scrum tehostaa projektitiimin kommunikaatiota palaverien avulla. Sprintti on yksi keskeisimmistä käsitteistä ja se kuvaa ajanjaksoa, jonka aikana palaverissa päätetyt tehtävät suoritetaan.

Esimerkiksi sprintti voi olla 1-4 viikon mittainen jakso, jonka aikana työlista suoritetaan (Kuva 10). Jokainen sprintti sisältää sisällään myös päiväpalavereja, joissa käydään läpi tärkeimmät tehtävät. (Schwaber & Sutherland 2013.)



Kuva 10. Scrum-mallin eteneminen. (Suntuubin www-sivut 2009.)

Perinteiseen Scrum–mallin rinnalle on noussut Scrumbut-versio. Versio on projektitiimin sovellettu malli alkuperäisestä menetelmästä. Poikkeamia voivat olla esimerkiksi palaverien pitäminen viikoittain, kun alkuperäisessä mallissa on päivittäinen menettely. Samoin sprinttien ajanjaksoa voidaan vaihtaa. (Haikala & Mikkonen 2011, 52-53.)

Scrum muodostuu asiantuntijarooleista, joissa mukana ovat master, kehitystiimi sekä asiakkaan tuoteomistaja. Master on vastuussa siitä, että Scrum-periaatteita noudatetaan ja toimii tiimin sekä tuoteomistajan tukena. Keskeisimpiin masterin tehtäviin kuuluvat vastuu sprintin tuloksesta ja palavereissa suunniteltujen vaatimusten toteutumisesta. (Haikala & Mikkonen 2011, 49; Schwaber & Sutherland 2013.)

Tuoteomistaja on vastuussa projektin taloudellisesta kehityksestä. Omistaja toimii yhdessä muiden kanssa ja kerää vaatimukset työlistaksi. Vaatimukset ovat tuotteen ominaisuuksia, käyttötapauksia ja käyttökokemuksia. Työlistan päivittäminen sisältyy tuoteomistajan vastuualueeseen. (Haikala & Mikkonen 2011, 48-49.)

Kehitystiimi vastaa projektin tehtävien suorittamisesta. Scrum-tiimi koostuu eri ohjelmistoprojektin menetelmien asiantuntijoista, kuten tietokantasuunnittelijasta ohjelmoijaan ja testauksen ammattilaiseen. Koska Scrum periaatteisiin kuuluu itseohjautuvuus, tiimi saa itse päättää tai äänestää asiantuntijaroolinsa. Kokemuksen myötä roolijakoa ohjaa osaaminen ja työtehokkuus. Tehokkuuden näkökulmasta optimaalisessa tiimissä on korkeintaan seitsemän henkilöä. Tiimissä on asiantuntijoita eri sovelluskehityksen osa-alueista, jotka tulevat ohjaamaan roolien jakoa tulevissa projekteissa. Alkuperäiseen ajattelutapaan kuitenkin kuuluu roolien kierrättäminen tiimin sisällä. (McConnell 2002, 283; Haikala & Mikkonen 2011, 48-49.)

3.7 Extreme Programming (XP)

Extreme Programming (XP) perustuu jatkuvaan testaamiseen ja pariohjelmointiin. Pariohjelmoinnissa ohjelmistokehitys tapahtuu pareittain, joissa ohjelmoija kirjoittaa koodia ja toinen kommentoi tuotoksen välittömästi. (Haikala & Märijärvi 2006, 47.)

XP:llä saadaan tehokkaasti onnistuneita tuloksia, koska kehityksessä voidaan ottaa huomioon muuttuvat vaatimukset, jopa projektin elinkaaren lopussa. Extreme Programming perustuu ohjemalleihin, joita ohjaavat tiimityötä tukevat arvot. Iteratiiviset ohjemallit kuvataan palapelin paloina, joista projekti koostuu. Yksikään pala ei saa puuttua, jotta projektin kokonaisuus säilyy. (Haikala & Mikkonen, 2011, 43; Extremeprogramming www-sivut 2013.)

Ohjemallien mukaan projekti tulee suunnitella jaettaviksi iteraatioihin, jotka suunnitellaan jokaisen vaiheen jälkeen. Projektiin haetaan hallittavuutta avoimella työympäristöllä sekä asettamalla projektille mitattavissa olevan päämäärän. Määrittelyn tulee perustua yksinkertaisuuden arvoon, jotta vältetään turhien resurssien käyttö. XP:ssä kaikki tuotettu koodi on pariohjelmoitu sekä testivetoisesti suunniteltu. (Extremeprogramming www-sivut 2013.)

XP:ssä ohjelmoija kommunikoi jatkuvasti yhdessä muun tiimin ja asiakkaan kanssa. Työtehokkuutta ohjaa yksinkertaisuus ja välitön palaute. Testausta suoritetaan projektin alusta alkaen, jotta tarvittavia muutoksia voidaan toteuttaa nopealla aikataululla. (Extremeprogramming www-sivut 2013.)

Extreme Programming ohjelmistokehityksen arvot ovat:

1. Yksinkertaisuus (KISS-periaate, Keep It Simple, Stupid)
2. Kommunikaatio
3. Palaute
4. Kunnioitus
5. Rohkeus

3.8 Lean & Kanban

Lean perustuu toiminnan kehittämiseen, joka ei ota kantaa yksittäisiin prosesseihin, vaan keskittyy työn tehostamiseen. Käytännössä kehitys on Lean-ajattelun soveltamista ohjelmistotuotantoon. Kanban-menetelmän tarkoituksena on muuntaa kehitys tehokkaaksi Lean-ajattelumalliin sopivaksi. (Lehtonen 2014, 8.)

Lean-malli on yksi ketteristä malleista, joka siirtynyt ohjelmistotuotantoon Japanin autoteollisuudesta. Mallin ydinidea on tehostaa ohjelmistotyön toimintaa ja välttää hukkatyötä. Malli ei ota kantaa siihen mitä tehdään, vaan perustuu ajattelutapaan työn arvoketjuista. (Lean.org www-sivut.)

Lean korostaa asiakkaalle tuotettavaa arvoa ja minimoi tuottamattomat hukkatoiminnot. Arvon lisäämisellä ja hukkatyön poistolla on suoravaikutus asiakastyytyväisyyteen, kustannuksiin, laatuun ja läpimenoaikaan. Läpimenoaika kuvaa aikaa, joka kuluu ohjelmiston tilaamisesta sen toimittamiselle. (Lean.org www-sivut.)

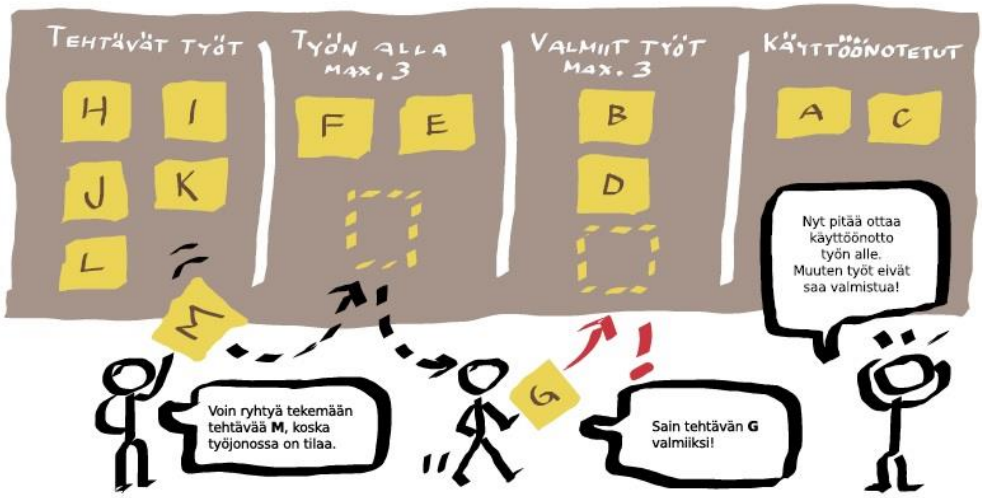
Malli koostuu viidestä perusperiaatteesta, joilla saadaan aikaa Lean-ajattelun mukainen ohjelmistoprojekti (Kuva 11). Periaatteet kuvataan helpoiksi ymmärtää, mutta haastavaksi toteuttaa käytännön projektityössä. (Lean.org www-sivut.)

Kuva 11. Lean menetelmän viisi periaatetta prosessikaaviona. (Lean.org www-sivut.)

Lean prosessi alkaa, kun projektista tai tuotteesta tunnistetaan kaikki asiakkaalle saatava arvo (kohta 1.) Seuraavaksi kartoitetaan arvovirta ja poistetaan hukkatekijät (kohta 2). Hahmotelmia voidaan tehdä kaksi, joissa kuvataan nykytilanne ja toisessa tavoite johon pyritään. Seuraavaksi arvovirrasta muodostetaan arvoketju tilauksesta toimitukseen (kohta 3). Prosessin neljännessä vaiheessa (kohta 4) projektia ohjaa imuohjaus, jossa projektia ohjataan asiakasvetoisesti. Tehdään vain se mitä asiakas tilaa, jotta voidaan tavoitella täydellisyyttä (kohta 5). Lean prosessia toistetaan niin kauan, kuin täydelliseen arvoon päästään ilman yhtään hukkatekijää. (Lean.org www-sivut.)

Kanban sana merkitsee taulua tai mainoskylttiä. Menetelmän historia ulottuu monien muiden mallien tavoin teollisuuteen, mutta on sittemmin siirtynyt ohjelmistotuotantoon. Tavoitteena on prosessien optimointi tarvevetoisuuden (JIT, Just-In-Time), visualisoinnin sekä samanaikaisen työn rajoittamisen avulla. JIT-strategian avulla varmistetaan, että työtä tehdään vasta, kun tiedetään mitä, miten ja milloin tehdään. Prosesseja voidaan tehostaa, kun ymmärretään miten ne toimivat. Ohjelmistotuotannon prosessit tuodaan visuaalisesti esille, jolloin siihen liittyvät ongelmat on helpompi havaita ja ratkaista. (Hämäläinen 2014; Lehtonen 2014, 15; Tuominen 2010, 182.)

Prosessit visualisoidaan Kanban-taulun avulla. Perinteinen Kanban-taulu sisältää sarakkeet, joissa on kuvattu tuotantoprosessit. Taulu sisältää sarakkeet tehtäville, meneillään oleville, valmiille ja julkistetuille töille (Kuva 12). Taulun avulla koko projektin prosessit on hahmotettavissa. (Lehtonen 2014, 15.)



Kuva 12. Työprosessien visualisointi Kanban-taulun avulla. (Lehtonen 2014, 16.)

Työtä ohjaa päällekkäisten tehtävien rajoitus, jonka tarkoituksena on estää resursseja tuhlaava työ. Taulussa työprosessi ei siirry seuraavaan vaiheeseen ennen kuin seuraavassa sarakkeessa on tilaa. Visuaalisuuden myötä projektista on helppo hahmottaa pullonkaulat ja ratkaista ne. (Lehtonen 2014, 15.)

Kanban menetelmän käyttö vaatii projektitiimiltä kokemusta, jotta taulun hyödyntäminen ei jää pelkäksi hahmotelmaksi, jossa työprosesseja siirrellään sarakkeesta toiseen. Kokemuksen myötä taulusta muodostuu projektitiimille prosesseja tehostava työkalu. (Hämäläinen 2014.)

4 HAASTATTELUTUTKIMUS

4.1 Tutkimusmenetelmät

Kvalitatiivinen eli laadullinen tutkimus vastaa ensisijaisesti kysymyksiin miksi, millainen ja miten. Laadullinen tutkimus auttaa ymmärtämään ilmiöitä yksityiskohtaisesti. Laadullisessa tutkimuksen tehtävänä on käsitteellinen ymmärtäminen, jossa tarvitaan teorian käsitteistö ja kokemusperäinen vastine. Laadullisen tutkimuksen mene-

telmiä ovat ryhmäkeskustelut, haastattelut, kyselyt, havainnointi ja kohderyhmän itsetuottama aineisto. Kvalitatiiviset menetelmät kohdistuvat usein valittuihin yksilöihin, jotka edustavat yksilöinä suurempaa joukkoa. (Tilastokeskus 2007; Insiprans 2014.)

Kvantitatiivinen eli määrällinen tutkimus vastaa tutkimusongelmaan numeerisesti. Määrällisen tutkimuksen tavoitteena on ilmiön ymmärtäminen tilastoinnin ja matemaattisten menetelmien avulla. Tarkoituksena on selittää, kuvata, kartoittaa tai vertailla. (Vilkka 2014, 13-14.)

Tutkimuksessa voidaan käyttää molempia sekä laadullista, että määrällistä menetelmää. Sanallisia vastauksia voi olla tehokkaampi analysoida laadullisilla menetelmillä, mutta tuloksia voidaan tiivistää käyttäen määrällisiä menetelmiä. Vastaavasti numeerisia tulee käsitellä määrällisin keinoin. (Vehkalahti 2014, 11.)

Laadullisen ja määrällisen tutkimushaastattelun erot muodostuvat mm. kohderyhmästä ja haastattelun perusmuodosta, joka voi olla avoin, puolistrukturoitu tai strukturoitu. Laadullinen haastattelu pyrkii aineiston tulkintaan ja määrällinen aineiston todenmukaisuuteen perustuvaan tilastoon. Laadullisen ja määrällisen haastattelun erot on kuvattu taulukossa 1. (Tilastokeskus 2007.)

Määrällinen tutkimus	**Laadullinen tutkimus**
Haastatellaan satunnaisotosta	Haastatellaan valittuja yksilöitä
Strukturoitu kysymys tai haastattelu	Avoin kysymys tai teema
Numeerinen havaintomatriisi aineistosta	Tulkinta aineiston pohjalta
Oletuksena aineiston todenmukaisuus, josta johtopäätökset	Aineiston tulkinnan perusteella johtopäätökset

Taulukko 1. Määrällisen ja laadullisen haastattelututkimuksen erot. (Tilastokeskus 2007.)

Tutkimusmenetelmäksi tässä työssä valittiin kvalitatiivinen tutkimusmenetelmä, jossa käytettiin puolistrukturoitua teemahaastattelua. Tutkimusmenetelmään päädyttiin, koska olettamuksena on se, että ohjelmistotaloissa sovelletaan monin eri tavoin ohjelmistotuotannon prosesseja ja käytänteitä. Puolistrukturoitu teemahaastattelu antaa pa-

remmat mahdollisuudet perustelulle sekä tarkennukselle ja tuo oletetusti yrityksen näkökulmaa esille, johon ei esimerkiksi pelkällä kyselylomakkeella päästä. Teemahaastattelussa käytetyt runkokysymykset löytyvät liitteestä 1. Runkokysymyksillä varmistetaan se, että haastattelutilanne pysyy teeman asettamissa rajoissa

4.2 Tarkoitus ja tavoitteet

Tutkimuksen tarkoituksena on selvittää miten ohjelmistotalot soveltavat teoreettisilta vaikuttavia prosesseja ja menetelmiä. Tutkimuskohteet rajataan ohjelmistotaloihin, joilla on toimipiste Satakunnan alueella. Tutkimuskohteet edustavat kvalitatiivisen tutkimuksen mukaan suuremmasta joukosta valittuja yksilöitä.

Tavoitteena on saada yleiskäsitys siitä, millaisia prosesseja ja menetelmiä tutkimuskohteena olevat yritykset käyttävät ohjelmistoprojekteissa. Miksi prosessit ja mallit on valittu ja miten mallit ohjaavat käytännön työtä. Lisäksi tavoitteena on tuoda esille se, mikä prosessi on projektin kannalta tärkeimmässä roolissa. Tutkittavaksi kohteiksi on valittu erilaisia ohjelmistoratkaisuja tarjoavia yrityksiä, niin sulautettujen järjestelmien kuin web-palveluiden projekteista.

4.3 Tutkimuskohteet ja aineisto

Tutkimuskohteet edustavat kvalitatiivisen tutkimusperiaatteen mukaan suuremmasta joukosta valittuja yksilöitä. Kohteiksi on pyritty valitsemaan laaja-alaisesti eri ohjelmistoratkaisuja tarjoavista yrityksistä. Kohdeyrityksissä haastateltavaksi valittiin henkilö tai henkilöt, joilla on laaja-alaisin kokemus ohjelmistotuotannon projektityöstä. Haastattelut toteutettiin kokonaisuudessaan kuukauden aikana. Haastateltavaksi valittiin kuusi yritystä, joista neljä osallistui lopulliseen tutkimukseen. Haastattelulle varattiin aikaa 10–15 minuuttia haastattelua kohden.

4.3.1 Teollisuusohjelmistoihin keskittyvä yritys A

Ohjelmistoprojektin alussa yritys luo tuotantoprosesseille selkeät rajat ja vaatimukset. Jokainen prosessi tuottaa halutun lopputuloksen, joka dokumentoidaan. Vaatimuksista on tärkeä saada mahdollisimman kokonaisvaltainen kuva jo projektin alussa, jotta tiedetään mistä lähdetään liikkeelle.

Vaatimusmäärittely kuvataan tärkeimmäksi prosessiksi. Projektille annetut työmäärät ja kustannukset ylittyvät herkästi, jos vaatimuksia ei ole pystytty ymmärtämään tai ne muuttuvat projektin edetessä. Prosesseilla luodaan joustavuutta ja pelivaraa tilaajan ja toteuttajan näkökulmasta. Tarvittaessa prosesseja voidaan muokata projektiin sopivammaksi.

Testaus suoritetaan yksikkötestauksena ja integraatiotestauksena koko projektin ajan. Testauksella varmistetaan toteutetun järjestelmän toiminta kohdeympäristössä. Suunnitelmallisuus kuuluu testaukseen, joten myös asiakas osallistuu testaamiseen ja testisuunnitteluun.

Käyttöönottoon kuluu toteutetun ohjelmiston viimeistely osaksi asiakkaan tietojärjestelmää. Koulutus on tärkeä osa käyttöönottoa, johon yritys panostaa yhä enemmän jatkossa. Ylläpitoon sisältyvät versiointi asiakas- sekä ohjelmistokohtaisesti.

Yritys käyttää ketterää Scrum-menetelmää, jossa käytössä on 2-3 viikon sprintit. Kehitystehtävät jaetaan sprinttisuunnitelmassa osiin ja jokainen sprintti suunnitellaan vastaamaan työlistaa. Haastavaksi Scrum-menetelmässä koetaan päivittäiset palaverit, koska aikatauluttaminen on haasteellista. Tuoteomistaja, master ja tiimi toimivat tiiviisti yhteistyössä ja ideointia tapahtuu koko projektin ajan.

4.3.2 Teollisuusohjelmistoihin keskittyvä yritys B

Yritys käyttää ohjelmistoprojekteissaan testiohjautuvaa suunnittelua. Projekti aloittamiseen kuuluu alkumäärittely, jonka päämääränä on kartoittaa projektin tavoitteet ja se minkä ongelman ohjelmisto ratkaisee. Projekti voi olla myös osa toista laajempaa

kokonaisuutta. Alkumäärittelyn tehtävänä on suunnitella projektin vaiheet ja prosessit. Määriteltäviä kohteita voivat olla käytettävät ohjelmistokehittimet, kehitystiimit, arkkitehtuurit ja halutut lopputulokset.

Ohjelmistotuotannon prosesseiksi yritys kokee suunnittelun, toteutuksen ja testauksen. Testiohjautuvassa kehityksessä testipohjaisuus on vahvasti läsnä. Tärkeimmäksi prosessiksi yritys kuvaa alkuvaiheen suunnittelun ja määrittelyn, koska päämäärä on oltava selvillä alusta alkaen. Projektipäällikön ja tiimin on sitouduttava projektiin koko sen elinkaaren ajan. Ylläpidon tarkoituksena on päivittää ohjelmistoa uudella kokonaisuudella, versiolla tai ominaisuudessa.

Menetelmissä yritys korostaa ketteriä menetelmiä, joissa parhaimmiksi puoliksi luetellaan lisäävä ja kasvattava kehitysote. Menetelmä muistuttaa Extreme Programming–mallia, mutta painotus on testisuunnittelussa, eikä pariohjelmoinnissa. Scrummallista käytössä ovat palaverit sekä tehtävien työlistaukset. Käytännössä projekteissa ohjelmistokoodilla ratkaistaan samankaltaisia ongelmia, joten yritys käyttää toteutusvaiheessa valmiita automatisoituja skriptejä. Uusi tuotettu koodi käy lävitse yksikkötestauksen, jonka suorittaa koodin ohjelmoinut henkilö, jonka jälkeen ohjelmakoodi siirtyy testipatteriston testattavaksi.

4.3.3 Taloushallinnon ohjelmistopalveluita tuottava yritys

Esitutkimuksessa yritys valmistelee projektisuunnitelman, johon on kirjattu projektin kannalta oleelliset päätökset ja seikat. Esitutkimuksen myötä yritys luokittaa projektinsa, joko haastavaksi, neutraaliksi tai suoraksi projektiksi.

Suunnittelu ja määrittelyvaiheessa projektitiimi jakaa projektin selkeisiin osa-alueisiin. Suunnittelu ja määrittely kuvataan projektikokonaisuuden tärkeimmäksi vaiheeksi, joten vaatimusmäärittelyyn halutaan panostaa. Kuitenkaan resursseja ei ole käytettävissä loputtomasti.

Toteutuksessa tehtäviin kuuluvat suunnitelmien toteuttaminen, valvonta ja testaus. Käyttöönotto ja testaustulokset hyväksytetään myös asiakkaalla. Ylläpitoprosessiin sisältyvät räätälöinti ja koulutus havaitun tarpeen mukaan. Tekninen tuki ja ohjaus ovat myös ylläpidon tehtäväkokonaisuuksia.

Yritys käyttää projekteissaan lähtökohtaisesi RUP sekä Scrum-mallin yhdistelmää. Rakennus- eli toteutusvaiheessa käytetään versiointia, jolloin ohjelmistoa kehitetään lisäävästi versioittain. Käytettävät menetelmät ovat kehittyneet projektien aikana ja niitä yritys pyrkii soveltamaan parhaansa mukaan. Esitutkimuksessa määritelty projektiluokitus vaikuttaa siihen miten menetelmiä ja prosesseja hyödynnetään. Projektit voivat olla laajuudeltaan ja sisällöltään hyvinkin erilaisia.

4.3.4 Markkinointitoimisto

Yrityksen ohjelmistoprojekti lähtee liikkeelle informaation keräämisestä toimittajan ja tilaajan puolelta. Molemmat varaavat projektin alkuvaiheessa tarvittavan määrän resursseja ja ilmoittavat sitoutuvansa ohjelmistoprojektiin. Onnistuneen ohjelmistoprojektin edellytyksenä ovat hyvät kommunikaatiotaidot tilaajan ja toteuttajan kannalta.

Prosesseihin kohteena oleva yritys kuvaa kuuluvan määrittelyn, suunnittelun ja toteutuksen. Toteutus eli rakennus vaiheeseen kuuluvat myös testaus, käyttöönotto sekä mahdollinen ylläpito. Markkinointitoimistossa suunnittelu ja määrittely korostuvat koko projektin ajan. Käyttöliittymällä ja sen suunnittelulla on vaikutusta siihen, miten asiakas tuotteen kokee. Hyvin suunniteltu on puoliksi tehty, jolloin pääpaino on projektin alkuvaiheen prosesseissa.

Testauksessa yritys käyttää yksikkö ja moduulitestausta, joissa varmistetaan haluttu toiminta. Testaukseen osallistuvat koodin tuottanut henkilö sekä toinen ohjelmistoalan ammattilainen. Ratkaisu on koettu tärkeäksi, koska omasta tuotoksesta on haastava havaita virheitä tai puutteita.

Yritys käyttää ketteriä menetelmiä, mutta projektin alussa käytetään vesiputousmallia, jotta tarvittavat vaatimukset saadaan koottua mahdollisimman tarkasti. Suunnittelu ja

määrittely prosessit suoritetaan loppuun, jolloin voidaan ketterästi ja lisäävästi jatkaa projektia. Määrittelyn tulokset korostuvat projektin edetessä.

Kysyttäessä markkinointitoimisto ei käytä tarkkaan mallin mukaista menetelmää, vaan käytössä ovat vesiputousmalli projektin alussa ja Scrum-menetelmä lopussa. Sprintit ovat viikon mittaisia, jonka aikana haluttu kokonaisuus suunnitellaan ja toteutetaan hyväksytetysti. Hyväksymiseen osallistuu myös asiakkaan projektiryhmän edustajat. Projekti jaetaan työmäärältään viikon mittaisiin osiin, jotka sisältävät suunnitellut työvaiheet.

5 JOHTOPÄÄTÖKSET JA POHDINTA

Prosesseilla tutkimuskohteena olevat ohjelmistoalan yritykset pyrkivät jaksottamaan kehitystyötä pienempiin kokonaisuuksiin, jotta saadaan haluttu ja helposti hallitavissa oleva lopputulos.

Osassa yrityksistä esitutkimus oli osa vaatimusmäärittelyä ja sekä suunnittelu ja määrittely kuvattiin samaksi prosessiksi. Toteutusprosessiin kuvattiin kuuluvan testaus ja käyttöönotto, joten prosessien väliset rajat vaihtelivat. Syy tähän löytyy projektitiimin ja projektin koosta, sekä tuotettavasta ohjelmistotuotteesta. Kehitettävä ohjelmisto voi olla suunniteltu suurelle joukolle asiakkaita tai sitten yritys keskittyy asiakaskohtaiseen räätälöintiin.

Tutkimuskohteena olleista yrityksistä lähes jokainen kuvasti vaatimusmäärittelyn tärkeimmäksi prosessiksi (Taulukko 2). Syyt tähän löytyvät projektille asetetuista aika- ja kustannusrajoitteista. Kun tiedetään mitä tehdään, on vaivattomampi viedä ohjelmistoprojektia eteenpäin. Tilaajan on voitava esittää tarpeeksi tietoa haluamastaan järjestelmästä ja oltava läsnä projektin alusta alkaen. Teollisuuden ohjelmistotuotteisiin keskittyneet yritykset painottivat testausta ja sen merkitystä.

	Teollisuuteen ohjel. tuottava yritys A	Teollisuuteen ohjel, tuottava yritys B	Taloushallinnon yritys	Markkinointitoimisto
Tärkein prosessi	Vaatimusmäärittely ja testaus	Testaus	Vaatimusmäärittely	Vaatimusmäärittely ja suunnittelu
Parannettavaa	Käyttöönotto	Käyttöönotto ja testaus	Vaihtelee, Vaatimusmäärittely	Vaihtelee
Menetelmä	Scrum	Scrum ja XP	Scrum ja RUP	Scrum + Vesiputous
Scrumbut	Kyllä	Kyllä	Kyllä	Kyllä
Sprintin kesto	2-3 viikkoa	Vaihtelee	Vaihtelee	1 viikko
Ketterä ja lisäävä kehitystyö	Kyllä	Kyllä	Kyllä	Kyllä

Taulukko 2. Tulostaulukko tutkimushaastattelun havainnoista.

Varsinkin teollisuuden ohjelmistotuotteita suunnittelevissa yrityksissä on huomattavissa testauksen rooli. Jos valmiiseen tuotokseen jää virheitä, voivat ne aiheuttaa myöhemmässä vaiheessa ongelmia esimerkiksi tuotantoon. Käyttöönotossa yritykset haluavat panostaa loppukäyttäjän kouluttamiseen, jotta ohjelmistosta olisi saatavissa suurin hyöty. Koulutustarjonta ja käyttäjätuki mahdollistavat uudenlaista liiketoimintaa ohjelmistoyrityksille.

Ketterät menetelmät osoittautuivat käytetyimmäksi menetelmäksi, joka perustuu siihen, että ohjelmistoalan yritykset haluavat tuottaa nopeassa syklissä konkreettisesti hahmotettavissa olevaa. Lisäävä ja ketterä kehitys mahdollistavat myös keinon vaikuttaa muuttuviin vaatimuksiin. Tilaaja ei aina projektin alussa osaa kertoa, mitä ohjelmiston tulisi todella tehdä. Kun vaatimuksia ja suunnittelua pystytään jaksottamaan projektin ajalle, varmistutaan oikeasta kehityssuunnasta. Haastattelujen perusteella resursseja ei ole aina tarpeeksi käytettävissä, kuin syväluotaava vaatimusten kerääminen vaatisi.

Menetelmät ovat haastatteluiden perusteella mukautuneet yrityskulttuuriin, eikä menetelmiä noudateta aivan niin täsmällisesti, kuin teoriaosuudessa käsiteltiin. Käytännössä menetelmät ovat muovautuneet yritysten omiin toimintatapoihin, joita sovelletaan painotetusti projektien välillä. Laajoissa projekteissa voidaan jopa käyttää useitakin menetelmiä. Myös käytössä olevilla resursseilla sekä asiakkaan sitoutuneisuudella on vaikutusta menetelmien soveltamiseen.

6 YHTEENVETO

Tämän opinnäytetyön tarkoituksena oli selvittää, mitä prosesseja ja menetelmiä liittyy ohjelmistotuotantoon. Aihepiiriä käsiteltiin lähteisiin perustuvan teorian ja yritysmaailmaan sijoittuvan haastattelututkimuksen perusteella. Työn tavoitteena oli saada kattava yleiskuva aiheesta.

Prosesseihin kuuluvat kaikki ohjelmistotuotannon elinkaaren mukaiset vaiheet. Esitutkimuksessa tavoitteena on luoda puitteet projektin aloittamiselle ja kiinnittää ohjelmistoyrityksen ja asiakasyrityksen tiimit. Vaatimusmäärittelyssä luodaan ne vaatimukset, jotka ohjelmiston ja projektin on täytettävä. Suunnitteluvaiheessa määritellään ratkaisut, joilla vaatimukset saadaan toteutettua. Toteutuksessa rakennetaan suunnittelun mukainen ohjelmisto, joka testataan virheiden varalta. Tilaajalle toimitetaan testattu tuote käyttöönotossa ja ylläpidossa varmistetaan ohjelmiston pitkä elinkaari.

Ohjelmistotuotannon prosesseja sovelletaan menetelmien ja mallien avulla. Koska ohjelmistotuotanto on nuori tieteenala, ovat menetelmät peräisin teollisuudesta. Menetelmien avulla varmistutaan siitä, että projekti pysyy hallittavissa ja oikeissa puitteissa. Menetelmät luokitellaan perinteisiin ja ketteriin menetelmiin. Perinteisissä menetelmissä erottuu työn suoraviivaisuus ja prosessit nähdään usein itsenäisinä kokonaisuuksina. Perinteiset menetelmät soveltuvat projekteihin, joissa vaatimukset ovat hyvin tarkkaan tiedossa jo projektin alkumetreillä. Ketterissä menetelmissä kehitystyötä voidaan jaksottaa ja prosesseja voidaan suorittaa rinnakkain. Ketterissä menetelmissä on

myös lisäävä kehityssuunta, jolloin prosesseja voidaan toistaa, jolloin lopullinen ohjelmisto laajenee vähitellen. Ketterät menetelmät soveltuvat projekteihin, joissa vaatimukset voivat muuttua projektin edetessä.

Haastattelututkimuksen perusteella yrityksen tapa toimia vaikuttaa siihen, miten prosesseja sovelletaan. Prosessien rajat vaihtelivat yrityksestä riippuen, johon vaikuttavat projektien laajuus ja käytettävissä olevat resurssit. Testaukseen ja loppukäyttäjän kouluttamiseen yritykset haluavat panostaa entistä enemmän. Haastattelujen perusteella vaatimusmäärittely on kriittisin vaihe ohjelmistoprojektissa ja se koettiin suurimmaksi syyksi projektin epäonnistumiseen resurssipulan ohella. Vaatimusmäärittelyn merkitys kasvaa projektin edetessä. Lisäävällä ja ketterällä ohjelmistokehityksellä voidaan vaikuttaa muutoksiin, jotka voivat olla riippuvaisia asiakkaalta saatavista vaatimuksista, ympäristöstä tai projektille asetetuista resursseista.

Yleisesti opinnäytetyö eteni ensin teoriataustaan tutustuen, jonka jälkeen suunniteltiin tutkimusosaa. Tutkimusmenetelmät ja tavoitteet asetettiin ennen kohdeyrityksien valintaa. Aiheen rajaamiseksi työssä keskityttiin vain ohjelmistotuotantoon liittyviin toimintoihin, eikä projektinhallintaan liittyviin tukiprosesseihin, kuten dokumentointiin tai riskienhallintaan. Tutkimusosan edetessä myös teoriaosuus täydentyi tarpeellisin osin. Kysymysrunko olisi saanut olla tarkemmin rajattu, mutta käytetyt tutkimusmenetelmät soveltuivat tutkimukseen, koska yritykset painottavat eri asioita.

LÄHTEET

Extreme Programming www-sivut. Viitattu 22.10.2015. Saatavissa: http://www.extremeprogramming.org/

Haikala, I & Märijärvi, J. 2006. Ohjelmistotuotanto. Helsinki: Talentum

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

Hämäläinen, H. 2014. Tehokas Kanban vai kasa lappuja seinällä. Viitattu 3.11.2015. Saatavissa: http://contribyte.fi/tehokaskanban

IBM. 2001. Rational Unified Process. Viitattu 11.2.2016. Saatavissa: https://www.ibm.com/developerworks/rational/library/content/03July/1000/1251/1251_bestpractices_TP026B.pdf

Inspirans www-sivut. 2014. Kvalitatiivinen tutkimus. Viitattu 13.10.2015. Saatavissa: http://www.inspirans.fi/kvalitatiivinen-tutkimus/

Julkisen hallinnon suositukset. 2012. ICT-palvelujen kehitäminen.Viitattu 28.1.2016. Saatavissa: http://docs.jhs-suositukset.fi/jhs-suositukset/JHS173/JHS173.html

Kalliala, E. 2010. Ketteryys ja hoikkuus pilvessä. Viitattu 19.2.2015. Saatavissa: http://eijakalliala.fi/2010/10/ketteryys-ja-hoikkuus-pilvessa/

Kaskela, L. 2005. Tietotekniikka hankinnat. Viitattu 28.1.2016. Saatavissa: http://www.tieke.fi/pages/viewpage.action?pageId=3441263

Kettunen, J & Simons, M. 2001. Toiminnanohjausjärjestelmän käyttöönotto PK-yrityksessä. Viitattu 2.2.2016. Saatavissa: http://www.vtt.fi/inf/pdf/julkaisut/2001/J854.pdf

Lahden ammattikorkeakoulun julkaisu. 2013. Yhteistoiminnallinen Ohjelmistohankinta. Viitattu 1.2.2016. Saatavissa: http://www.lpt.fi/tykes/instructions_docs/Yhteistoiminnallinen_hankinta_-COSA.pdf

Lean.org www-sivut 2015. Viitattu 27.10.2015. Saatavissa: http://www.lean.org/

Lehtonen, T. 2014. Ketterä käsikirja. Turku: TEKES

Manninen, T. 2007. Pelisuunnittelun sanasto Viitattu 16.2.2015. Saatavissa: http://www.ludocraft.com/pelisuunnittelija/sanasto.html

McConnell, S. 2002. Ohjelmistotuotannon hallinta. Helsinki: Edita

Murch, R. 2002. IT-Projektinhallinta. Helsinki: Edita

Pohjonen, R. 2002. Tietojärjestelmien kehittäminen. Jyväskylä: Docendo Findland Oy

Royce, W. 1970. Managing the Development of Large Software Systems. Viitattu 22.2.2015. Saatavissa: http://www.cs.umd.edu/class/spring2003/cmsc838p/Process/waterfall.pdf

Ruuska, K. 1999. Projekti hallintaan. Jyväskylä: Suomen atk-kustannus Oy

Schwaber, K. & Sutherland, J. Viitattu 16.2.2015. Saatavissa: https://www.scrum.org/resources/what-is-scrum/

Suntuubi www-sivut. 2009. Viitattu 4.3.2015. Saatavissa: http://hybridimenetelma.suntuubi.com/?cat=16

Tilastokeskus www-sivut. 2007. Viitattu 13.10.2015. Saatavissa: www.tilastokeskus.fi/virsta/tkeruu/01/07

Tuominen, K. 2010. Lean käytännössä Yritysesimerkkejä tehokkaista lean-periaatteista ja käytännöistä. Helsinki: Readme.fi

Vehkalahti, K. 2014. Kyselytutkimuksen mittarit ja menetelmät. Helsinki: Finn Lectura

Vesterholm, M & Kyppö, J. 2006. Java-ohjelmointi. Helsinki: Talentum

Vilkka, H. 2014. Tutki ja mittaa. määrällisen tutkimuksen perusteet. Helsinki: Tammi

Virkki, P & Somermeri, A. 1999. Systeemityö tutuksi. Vantaa: Provano Oy

Wiio, A. 2004. Käyttäjäystävällisen sovelluksen suunnittelu. Helsinki: Edita

LIITE 1

Teemahaastattelun runkokysymykset:

OHJELMISTOPROJEKTI

1. Millainen on hyvä projekti?
2. Mikä on yleisin syy siihen, että ohjelmistoprojekti epäonnistuu?
3. Mikä rooli asiakkaalla (tilaajalla) on projektissa?

PROSESSIT

1. Mistä prosesseista ohjelmistoprojektinne koostuu?
2. Mitä kuuluu ohjelmistoprojektin aloittamiseen ja päättämiseen?
3. Mikä merkitys on vaatimusmäärittelyllä?
4. Millaisia negatiivisia ja positiivisia kokemuksia teillä on suunnittelusta ja määrittelystä?
5. Mitä testaukseen ja käyttöönottoon sisältyy?
6. Mitä ylläpidossa tulee huomioida?
7. Kuinka prosessit ohjaavat käytännön projektityötä? Miten prosessit koetaan?

MENETELMÄT JA MALLIT

1. Mitä menetelmiä teillä on käytössä (ketterät vai perinteiset)?
2. Miten menetelmää sovelletaan? Miten menetelmä etenee tyypillisessä projektissa?
3. Vaikuttavatko projektin laajuus tai haastavuus menetelmien käyttöön?
4. Miksi käytössä on kyseinen menetelmä? Mikä vaikuttaa kyseiseen valintaan?