Liiketoiminnan kehitysprosessien kehitys

Nikolas Kunnas

Haaga-Helia ammattikorkeakoulu

Amk-opinnäytetyö

2021

Tietojenkäsittelyn koulutusohjelma

Tiivistelmä

Tekijä
Nikolas Kunnas

Tutkinto
Tietojenkäsittelyn koulutusohjelma

Raportin/Opinnäytetyön nimi
Liiketoiminnan kehitysprosessien kehitys

Sivu- ja liitesivumäärä
43 + 7

Tämä opinnäytetyö toteutettiin toimeksiantona asiakasyritykselle ja sen tarkoituksena oli osana toimintatutkimusta tutkia ja parantaa yrityksen liiketoimintaosaston nykyistä kehitysprosessimallia ja puuttua siinä ilmeneviin ongelmakohtiin. Toimintatutkimuksessa yhdistyy teoria ja käytännönläheisyys ja siinä tutkija osallistuu aktiivisesti tutkimukseen tehden muutosinterventioita eli muutokseen tähtääviä väliintuloja.

Tutkittavan yrityksen IT-osaston henkilöstömuutos vuonna 2021 sai kehitysprosessin muutoksen tarpeen alulle. Koko IT-osasto vaihtui uusiin työntekijöihin ja tämä luonnollisesti aiheutti muutoksia osastossa tehtävään työhön, sillä aikaisempi useiden vuosien IT-osaaminen oli kadonnut ja sitä piti lähteä rakentamaan uusiksi. Tämän lisäksi oli jo entuudestaan havaittu ongelmakohtia nykyisessä mallissa. Nämä johtivat nykyisen kehitysprosessimallin tutkimukseen ja kehitykseen toimintatutkimuksen muodossa.

Teoreettinen viitekehys liittyy tämän työn osalta järjestelmäkehityksen elinkaareen (Systems Development Life Cycle tai SDLC) ja siihen liittyviin filosofioihin ja metodeihin. Kirjoitettu teoria luo pohjan kehitysprosessien ’ideaalimallille’, jota voidaan reflektoida liiketoimintaosaston nykyiseen kehitysprosessimalliin, joka toimii osaltaan yhtenä suunnannäyttäjänä parannusehdotuksille.

Opinnäytetyön tutkimusmenetelmät-osioon kuului laajat puolistrukturoidut teemahaastattelut neljän yrityksessä työskentelevän henkilön osalta. Vastausten avulla pystyttiin muodostamaan selkeä kokonaiskuva kehitysprosessin nykymallista ja siihen liittyvistä ongelmakohdista sekä mahdollisista parannusehdotuksista. Haastattelukysymysten tarkoitus oli saada selville tietoa, joka palvelisi kehitysprosessin käytänteiden kehittämisessä. Kysymysten avulla selvitettiin mikä kehitysprosesseissa yleisesti ottaen toimii ja mikä ei. Haastattelukysymyksistä selvisi myös muita tärkeitä seikkoja, jotka auttoivat ongelmakohtien selvittämiseen ja niihin puuttumiseen. Muita tutkimusmenetelmiä olivat osallistava havainnointi ja aineiston analyysi.

Haastattelukysymysten vastausten ja tutkijan tekemän osallistavan havainnoinnin perusteella kehitysprosessissa pelkästään ei ollut vikaa, vaan pullonkaulaksi ilmeni myös prosessia noudattavat ihmiset. Lopputuloksena tämän opinnäytetyön osalta syntyi parannusehdotelma, jossa nostetaan esille nykyisen kehitysprosessimallin ongelmakohdat ja niiden parannusehdotukset.

Asiasanat
Kehitysprosessi, SDLC, toimintatutkimus, järjestelmäkehitys

Sisällys

1 Johdanto

Sovellukseen tai järjestelmiin liittyvää kehitystyötä tehdessä noudatetaan usein jotakin tiettyä viitekehystä, joka ohjaa kehitystyön vaiheita, määrittelee kehitysprosessin ja pitää kokonaisuuden organisoituna. Kehitysprosessilla tarkoitetaan tämän opinnäytetyön osalta vaiheita, joiden mukaisesti järjestelmäkehitystyötä edistetään. Viitekehyksiä on lukuisia eikä ole olemassa yhtä ja oikeaa viitekehystä, joka sopisi mihin tahansa organisaatioon käytettäväksi, vaan käytettävä viitekehys tulisi arvioida yrityksen tarpeiden mukaisesti (Peterson, J. 2020).

Toimeksiantajayrityksen IT-osastolla, jossa itsekin työskentelen, tapahtui vuonna 2021 henkilöstömuutos, jonka takia aikaisemmin hankittu, yhteensä jopa 25 vuoden IT-alan työkokemus oli hävinnyt. Kehitysprosessit eivät nykyisellä IT-osaamisella toimineet yhtä hyvin kuin aikaisemmin. Osaston uusien ja pidemmän aikaa töissä olleiden työntekijöiden kesken oli ollut puhetta siitä, että kehitysprosessit kaipaisivat muutosta parempaan. Koko prosessista löytyy useita epäkohtia, jotka hankaloittavat koko prosessin tehokkuutta ja työntekijöiden tuottavuutta. Näistä mainitaan tarkemmin luvussa 6.1.

Kehitysprosesseista: Yrityksen politiikkaan on kuulunut omanlainen ideaprosessi, jossa kuka tahansa koko konsernista voi halutessaan esittää idean, joka voi mahdollisesti päästä suunnitteluvaiheesta aina toteutusvaiheeseen saakka, mikäli idea todetaan toimivaksi. Kun esitetyn idean toteutuskelpoisuutta aletaan miettimään, käynnistyy kehitysprosessi, jossa päätetään, hyväksytäänkö jonkin idean toteutus, onko uusi mahdollinen ominaisuus mahdollista toteuttaa, onko se realistinen tai edes tarpeellinen. Ideoiden lisäksi liikevaihtoa parantavat asiakastarpeesta lähtöiset kehitystyöt ovat sellaisia töitä, jotka käsitellään kehitysprosessien kautta. Kehitysprosessien sisältämillä vaiheilla saadaan karsittua mm. kannattamattomat kehitystyöt pois listalta ja liikevaihtoa tai toiminnallisuutta parantavat tuotokset eteenpäin kehityksessä.

1.1 Toiminnallisen opinnäytetyön tarkoitus

Tämän opinnäytetyön tarkoituksena on tutkia nykyistä liiketoimintaosaston kehitysprosessimallia, löytää sen pahimmat ongelmakohdat ja esittää niiden suhteen parempia toimintatapoja. Ongelmakohtien lisäksi etsitään myös vahvuuksia nykyisestä kehitysprosessimallista. Tavoitteena on parantaa kehitysprosessien toimivuutta ylipäätään ja poistaa mm. resursseja tuhlaavia ja tarpeettomia vaiheita koko prosessista. Lopputuloksena tämän toimintatutkimuksen osalta syntyy parannusehdotelma, jossa puututaan epäkohtiin ja esitetään niihin tutkimuksen avulla löydetty parempi toimintatapa.

Kun kehitysprosessia lähtee tutkimuksen ja teorian kautta kehittämään paremmaksi, voi se tuoda selkeitä parannuksia varsinkin päivittäiseen työhön: Parempi vuorovaikutus eri sidosryhmien välillä, läpinäkyvyys työn eri vaiheissa, työntekijöiden tehokkaampi sitouttaminen prosesseihin ja kehitystyön parempi jälki. Myös tarpeettomien, aikaa vievien tapahtumien poistaminen prosessista, jättää aikaa muuhun tärkeämpään tekemiseen.

1.2 Opinnäytetyön tavoitteet ja tutkimuskysymykset

Tämän opinnäytetyön tavoitteena on tutkia yrityksen eri osastojen kehitystyössä käytettäviä liiketoimintalähtöisiä kehitysprosesseja ja selvittää miten nimenomaan yhden osaston kehitysprosesseja voitaisiin jatkokehittää entistä paremmiksi. Näin ollen tutkimuskysymys on:

1. Miten nykyistä kehitysprosessimallia voidaan parantaa?

2 Toimeksiantajan kuvaus

Toimeksiantajana on kansainvälinen finanssialan organisaatio, joka keskittyy toiminnassaan talouspalveluiden tarjoamiseen asiakkaille. Toimeksiantajayrityksen nimeä en tässä työssä mainitse toimeksiantajan toiveesta. Tarjottaviin talouspalveluihin kuuluu mm. reskontrapalvelut, luottokelpoisuuden tarkastus, saatavien kotiutus ja arvonlisäveropalvelut. Tässä opinnäytetyössä keskitytään tämän organisaation Suomen yrityksen yhden osaston toimintaan kehitysprosessien suhteen. Tässä opinnäytetyössä tuota osastoa kutsutaan liiketoimintaosastoksi tai vain liiketoiminnaksi. Erikseen myös viitataan IT-osastoon, joka työskentelee liiketoimintaosaston alla keskittyen IT-toimintoihin, kuten järjestelmän ylläpitoon, virhetilanteiden korjaamiseen ja muihin vastaaviin.

Organisaatiossa tapahtuva järjestelmien ja ohjelmistojen kehitystyö liittyy asiakkaille tarjottavien palveluiden ja tuotteiden käyttöön. Samalla tehdään kehitystyötä, jonka avulla voidaan pitää omat järjestelmät ja IT-infrastruktuuri ajan tasalla ja kilpailukykyisenä. Järjestelmään ja ohjelmistoon liittyvä kehitystyö tehdään lähes kokonaan itse kehitetyssä ympäristössä, juurikaan hyödyntämättä valmiita alustoja.

2.1 Sidosryhmät

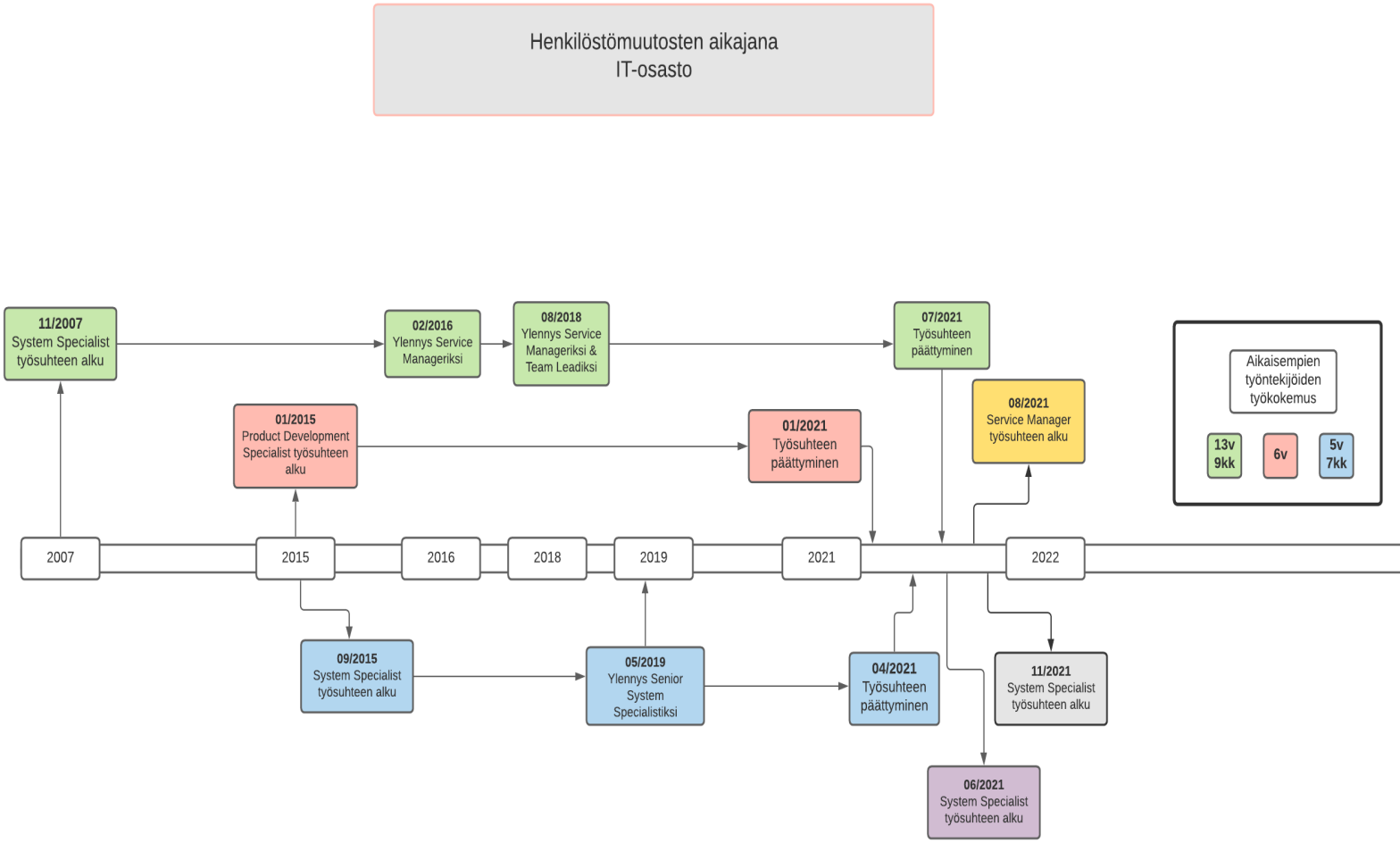
Liiketoimintaosasto tekee tiivistä yhteistyötä tärkeiden sidosryhmien kanssa. Sidosryhmillä tarkoitetaan organisaatiota tai henkilöryhmää, joiden kanssa yritys on tekemisissä. Sidosryhmä vaikuttaa yrityksen toimintaan, mutta samaan aikaan yritys vaikuttaa vastaavasti sidosryhmien toimintaan. Sidosryhmät voidaan jakaa sisäisiin ja ulkoisiin. Tyypillisesti yrityksen sisäisiin sidosryhmiin kuuluu mm. työntekijät ja omistajat. Ulkoisiin sidosryhmiin voi kuulua asiakkaat, yhteistyökumppanit, valtio ja palvelun toimittajat. (Freeman – Mouchnick 2013, 5–9)

Kuvassa liiketoimintaosaston alle lukeutuu FIN IT - ja DEV-osastot. FIN IT tarkoittaa IT-osastoa ja DEV ohjelmistokehitysosastoa. Molemmat näistä osastoista vastaavat liiketoimintaosastolle ja tekevät kehitystöitä näiden vaatimusten mukaisesti. FIN IT on enemmän keskittynyt järjestelmien ylläpitoon, hallintaan ja ylipäätään IT-infrastruktuuriin, kun taas DEV-osaston toiminnan pääpaino on sovelluskehityksessä.

Kuva 1 Sidosryhmäkaavio
Sidosryhmäkaaviossa FIN IT:llä tarkoitetaan organisaation Suomen yksikön liiketoiminta-osaston alle kuuluvaa IT-osastoa. DEV tarkoittaa ohjelmistokehityksen osastoa, joka on vastuussa kehitystöiden tekemisestä Liiketoiminnalle ja FIN IT:lle.

2.2 Henkilöstömuutokset

IT-osastolla on tapahtunut vuoden 2021 aikana merkittäviä henkilöstömuutoksia, sillä kaikki osaston työntekijät irtisanoutuivat saman vuoden aikana, päättäen useamman vuoden työuransa. Näiden henkilöstömuutoksen seurauksena aikaisempi vankka osaaminen on hävinnyt ja nyt työntekijöiden on täytynyt opetella työtehtävät, järjestelmät, toimintatavat ja hyvät käytännöt uudelleen.



Kuva 2 Henkilöstömuutosten aikajana

Kesäkuussa 2021 aloittanutta uutta työntekijää (violetti) ehdittiin perehdyttää ennen vihreällä väritetyn työntekijän työsuhteen päättymistä. Perehdytyskausi jäi kuitenkin kovin lyhkäiseksi, vain muutaman viikon mittaiseksi, eikä perehdytys ollut tämän takia riittävällä tasolla. Täysimittainen henkilöstömuutos onkin yksi merkittävä syy siihen, miksi toimintatapoja ja kehitysprosesseja mietitään uudelleen, sillä malli halutaan kehittää uudelle henkilöstölle mahdollisimman sopivaksi.

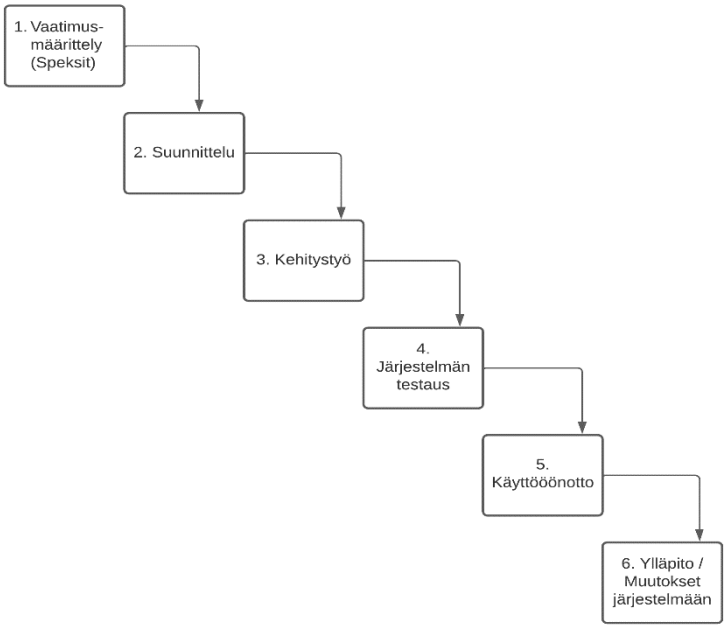
3 Viitekehykset

Kun puhutaan järjestelmiin liittyvästä kehitystyöstä, noudatetaan sen suhteen tiettyjä määriteltyjä toimintatapoja ja käytäntöjä, joiden mukaan kehitystyötä lähdetään edistämään järjestelmällisesti. Tässä opinnäytetyössä viitataan järjestelmäkehitykseen, jossa noudatetaan järjestelmäkehityksen elinkaaren alle lukeutuvia erilaisia viitekehyksiä ja metodeja. Näihin kuuluu mm. lineaarinen kehitysmalli, ketterät menetelmät sekä Scrum. Niistä kerrotaan tarkemmin tässä kappaleessa. Esille nostetaan myös jokaisen alaotsikon jälkeen kunkin mallin vahvuudet ja heikkoudet yhteenvetotaulukossa.

3.1 Lineaarinen kehitysmalli

Lineaarinen kehitysprosessimalli oli ensimmäinen esitetty Systems Development Life Cycle (SDLC) -malli. Tästä mallista käytetään myös termiä vesiputousmalli (Tutorialspoint). Tyypillistä vesiputousmallille on lineaarisuus, jossa yhden vaiheen päätyttyä siirrytään seuraavaan vaiheeseen ilman iteraatiota. Kun seuraavaan vaiheeseen on siirrytty, ei voida enää palata aikaisempaan vaiheeseen. (GeeksforGeeks, 2021)

Martin. M (2022) esittää blogikirjoituksessa, että vesiputousmallin vaiheet ovat 1. Vaatimusmäärittely, (Requirement Analysis), 2. Järjestelmän suunnittelu (System Design), 3. Implementointi (Implementation), 4. Järjestelmän Testaus (System Testing), 5. Järjestelmän Käyttöönotto (Deployment) ja 6. Järjestelmän Ylläpito (System Maintenance).



Kuva 3 - Vesiputousmallin vaiheet

Ensimmäisessä vaiheessa kerätään yksityiskohtaiset vaatimukset ohjelmistojärjestelmän kehitystyöstä. Toisessa vaiheessa suunnitellaan, millä ohjelmointikielellä työ tehdään tai vaikkapa mihin tietokantaan työ toteutetaan. On myös mahdollista, että suunnitellaan projektiin liittyviä muita teknisiä vaatimuksia. Kolmannessa vaiheessa tapahtuu ohjelmiston kehitystyö. Neljännessä vaiheessa tapahtuu ohjelmiston testaaminen, jotta voidaan varmistua siitä, että se noudattaa ensimmäisessä vaiheessa kerättyjä vaatimuksia. Toteutusvaiheessa tehdään implementaatio eli käyttöönotto kehitystyön suhteen. Toiseksi viimeisessä vaiheessa tehdään sovelluksen käyttöönotto valitussa ympäristössä (esim. tuotanto- tai testiympäristö). Viimeisessä vaiheessa keskitytään järjestelmän ylläpitoon ja mahdollisiin koodimuutoksiin. (M. Martin, 2022)

Sharma, L (2021) esittää vesiputousmallin vahvuuksia ja heikkouksia. Niistä on luotu yhteenvetona taulukko alle.

Vahvuudet	Heikkoudet
Soveltuu pienemmille projekteille, jossa vaatimukset ovat hyvin määritelty.	Ei optimaalinen malli monimutkaisille projekteille, jossa vaatimukset muuttuvat
Malli on helppo ymmärtää ja suoraviivainen. Yhden vaiheen päätyttyä seuraava vaihe alkaa.	Asiakkaalta saatavaa palautetta ei pysty sisällyttämään implementointivaiheeseen (3).
	Yhden vaiheen päätyttyä ei ole mahdollisuutta päästä takaisin aikaisempiin vaiheisiin.
	Testausvaihe tulee myöhäisessä vaiheessa kehitysprosessia.
	Virheet tai pienet muutokset, jotka nousevat esiin käyttöönotetussa ohjelmistossa voivat aiheuttaa paljon ongelmia.
	Esiintyvät virheet voidaan korjata vasta viimeisessä (6) vaiheessa.

Taulukko 1: Lineaarisen kehitysmallin vahvuudet ja heikkoudet

3.2 Ketterät menetelmät (Agile & Scrum)

Ketterä malli (Agile Model) keskittyy iteratiiviseen, lähestymistapaan, jonka pääpainona on tuottaa tuottamaan korkealaatuista ohjelmistoa niin nopeasti ja kustannustehokkaasti kuin mahdollista. Agile-malli luotiin vuonna 2001 pääosin sitä varten, että kehittäjät voivat luoda projektin, joka on mukautuvainen muuttuviin vaatimuksiin.

Ketterä malli pitää sisällään kuusi vaihetta ja ne ovat

1. Konsepti. Tässä vaiheessa tehdään projektiin liittyvä priorisointi ja työmäärän laajuuden arviointi. Eli päätetään mitä otetaan mukaan kehitysjonoon ja mitä lähdetään kehittämään.

2. Projektin käynnistäminen. Tässä vaiheessa tehdään yhteistyötä sidosryhmien kanssa ja kerätään vaatimukset kehitystyön osalta. Tässä kohtaa suunnitellaan ja demonstroidaan kehitettävän järjestelmän tai ohjelmiston toiminnallisuuksia.

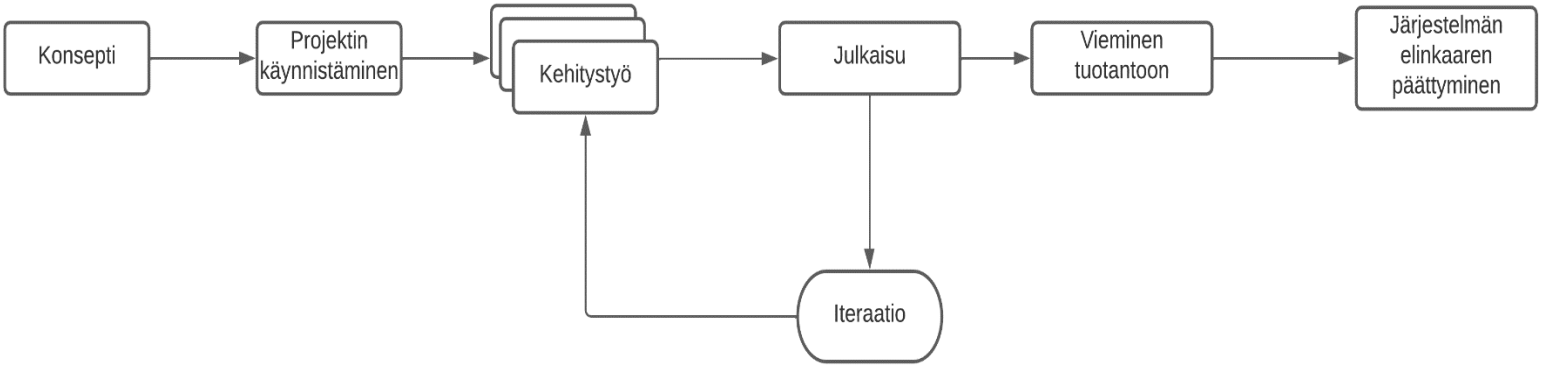
3. Kehitystyö. Vaatimusten keräämisen jälkeen varsinainen kehitystyö voi ensimmäisen iteraation osalta alkaa. Tavoitteena on saada toimiva tuote aikaiseksi, jonka voi ottaa käyttöön sprintin päätyttyä.

4. Julkaisu / iteraatio. Tähän vaiheeseen kuuluu laadun varmistus, järjestelmän testaaminen ja vikojen löytäminen. Mikäli puutteita todetaan tai tuote ei vastaa toisessa vaiheessa kerättyjä vaatimuksia, tapahtuu uusi iteraatiokierros, jossa ongelmakohdat korjataan.

5. Käyttöönotto / tuotantoon vieminen. Kun edellisen vaiheen lopullinen iteraatio on saatu valmiiksi, tapahtuu tuotteen käyttöönotto ja tuotantoympäristöön vieminen. Tähän vaiheeseen myös kuuluu jatkuva tuki ja parannustyöt. Tämä vaihe kuitenkin päättyy, kun tuki tuotteelle päättyy tai kunnes julkaistu tuote otetaan pois käytöstä ja siirrytään esimerkiksi käyttämään uudempaa korvaavaa tuotetta.

6. Järjestelmän elinkaaren päättyminen. Viimeisen vaiheen osalta, kun järjestelmä on tullut elinkaarensa päähän, poistetaan se tuotantoympäristöstä ja korvataan uudemmalla versiolla tai kokonaan toisella järjestelmällä.

Kuva 4 havainnollistaa Agilen vaiheita ja siinä korostuu iteraatio. Mikäli havaitaan, että kehitetystä ohjelmistosta puuttuu jotakin, voidaan iteratiivisesti siirtyä takaisin kehitysvaiheeseen ja korjata ongelmakohdat. Kun kaikki iteraatiot prosessissa ovat valmiita, lopullinen tulos näytetään asiakkaille ja muille sidosryhmille, jotka antavat viimeisen hyväksynnän tuotantoon viemiselle. (Lucidchart)



Kuva 4 - Agilen vaiheet

Agile-malli sai alkunsa vuonna 2001 luodusta Agile-manifestista, joka identifioi neljä ydinarvoa ja 12 periaatetta. Sovelluskehittäjien tulisi Manifestin luojien mukaan antaa näiden ydinarvojen ja periaatteiden ohjata heidän työtään. Agilen neljä ydinarvoa ovat:

- Yksilöt ja interaktiot prosessien ja työkalujen sijasta
- Toimiva ohjelmisto kokonaisvaltaisen dokumentaation sijasta
- Asiakkaan yhteistyö sopimuksen neuvottelun sijasta
- Muutokseen vastaaminen suunnitelman noudattamisen sijasta.
  ”Vaikka oikeilla esiintyvissä asioissa on arvoa, pidämme vasemmalla olevia asioita arvokkaampana”

Agile-manifestin 12 periaatetta ovat esitetty alla olevaan taulukkoon.

Korkein prioriteetti on asiakastyytyväisyys ohjelmiston toimituksessa sekä aikaisessa, että jatkuvassa vaiheessa.	Muuttuvien vaatimusten vastaanottaminen jopa myöhäisessä vaiheessa.	Toimivan ohjelmiston säännöllinen toimitus, mielellään lyhyen aikavälin sisällä.
Liiketoimintaihmisten ja kehittäjien täytyy olla päivittäisessä yhteistyössä läpi koko projektin.	Rakenna projekteja motivoituneiden yksilöiden ympärille. Tue heitä ja luota heihin.	Tehokkain tapa tiedon välittämiselle on kehityksen osalta henkilökohtaiset keskustelut.
Toimiva ohjelmisto on ensisijainen kehityksen mittari.	Agilen prosessit kannustavat kestävään kehitystyöhön. Kehittäjien ja muiden osallistujien tulisi jatkuvasti ylläpitää tasainen tahti	Jatkuva huomio, tekninen ylivertaisuus ja hyvä suunnittelu parantavat ketteryyttä.
Yksinkertaisuus – tehdyn työmäärän maksimointi minimaalisella työllä – on välttämätöntä.	Parhaat arkkitehtuurit, vaatimukset ja suunnitelmat kumpuavat itseorganisoiduista tiimeistä.	Tasaisin väliajoin tiimi reflektoi ja pohtii miten olla tehokkaampi ja säätelee toimintaansa sen mukaan.

Taulukko 2: Agile-manifestin 12 periaatetta. (Agilemanifesto, 2001)

Alle on esitetty ketterän mallin vahvuudet ja heikkoudet yhteenvetotaulukkona.

Vahvuudet:	Heikkoudet:
Korkeampi asiakastyytyväisyys, sillä product owner (PO) on jatkuvasti mukana kehitysprosessissa ja tekemässä tarvittavia parannuksia.	Jatkuva yhteistyö ja vuorovaikutus muiden sidosryhmien kanssa aikaa ja energiaa kaikilta osallisilta.
Projektin riskien minimoiminen, sillä Agile-malli on suunniteltu yllättäviä muutoksia varten.	Tarpeellisen dokumentaation puute. Tämä ilmenee jo Agilen yhdestä ydinperiaatteesta: ”*Toimiva ohjelmisto kokonaisvaltaisen dokumentaation sijasta”*
Lyhyiden sprinttien ansiosta nopeampi tuotto sijoittajille.	Vähemmän strukturoidun mallin takia Agile-mallia noudattavat projektit voivat muuttua suuremmiksi, kuin oli alun perin tarkoitus.
Ne asiat, mitkä ovat mainittu Agile-manifestin 12 periaatteessa.	

Taulukko 3: Agile-mallin vahvuudet ja heikkoudet. (University of Minnesota, 2022)

3.2.1 Scrum

Agile-mallin alle kuuluu lukuisia laajasti käytettyjä ja tunnettuja metodeja, jotka noudattavat ketterän mallin perusperiaatteita ja ydinasioita. Nämä kaikki ovat ikään kuin omanlaisia variaatioita ketterästä mallista. Ylivoimaisesti käytetyin metodi Agile-malliin pohjautuen on Scrum. Scrum-metodille tyypillistä on kehitystyön vaiheet, joita kutsutaan myös sprinteiksi. Metodia käytetään yleensä kehitysprojektien hallinnoimiseen ohjelmistotuotannon osalta, mutta se soveltuu myös liiketoimintalähtöiseen toimintaan. (Lamelas. A, 2018)

Tunnusomaista Scrumille on ydinroolit kehitystyössä, joita on yhteensä kolme: **Tuoteomistaja (Product Owner, PO),** joka on tärkeä linkki kehitystiimin ja tuotteen sidosryhmien osalta. **Scrum Mestari (SM)** vastaa siitä, että projekti pysyy aikataulussa ja ohjelmistokehitystiimi noudattaa Scrumin prosessia. Scrum Master kommunikoi PO:n kanssa, järjestää palavereja ja hoitaa projektissa ilmenevät esteet ja pullonkaulat sitä mukaa kun niitä ilmenee**.**
Ohjelmistokehitystiimi, joka ohjelmoijien lisäksi voi koostua myös suunnittelijoista ja testaajista, tekee tiivistä yhteistyötä keskenään. Heille ei ole määritetty tiettyjä rooleja, mutta

heille annetaan joukko tehtäviä, jotka heidän pitää suorittaa sprintin aikana. (Tateeda, 2020).

Scrum-tiimin tulisi koostua alle 9 ihmisestä. Suurissa projekteissa ideaali Scrum-tiimin koko on seitsemän henkilöä (PO, SM ja 5 sovelluskehittäjää), pienemmät projektit tyypillisesti koostuvat neljän hengen tiimistä, johon kuuluu PO, SM ja 2 sovelluskehittäjää. (Mendix, 2021)

Roolien lisäksi Scrumin koko prosessi on riippuvainen kolmesta asiasta:
1. **Tuotteen kehitysjono** (Product Backlog), 2. **Sprintin kehitysjono** (Sprint Backlog), 3. **Tuotteen inkrementti** (Product Increment).

1. Tuotteen kehitysjonolla tarkoitetaan suunniteltua prosessia, joka määrittää ja priorisoi tarvittavat Scrumin vaiheet, joiden avulla projekti saadaan onnistuneesti valmiiksi. Tuoteomistaja ja Scrum Master siistivät (Groom) yhdessä tuotteiden kehitysjonoa ennen sprintin suunnittelun alkamista. Tuotteen kehitysjono on ikään kuin ’elävä’ dokumentti, jota jatkuvasti hiotaan, uudelleenjärjestellään ja päivitetään sprintin aikana.

2. Sprintin kehitysjono on pienoismalli tuotteen kehitysjonosta ja se pitää sisällään listan niistä tehtävistä, jotka pitää suorittaa yhden sprintin aikana. Sprintin kehitysjonoon voidaan myös lisätä nk. käyttäjätarinoita, jotka kuvaavat tuotteen toivottua toiminnallisuutta.

3. Scrumissa tuotteen inkrementillä tarkoitetaan tuotekehitysjonossa valmiiksi saatujen kohtien yhteissummaa nykyisen tai aiemman sprintin aikana.

Kuvassa 5 on esitetty yhden sprintin vaiheet Scrumissa:

4. Sprintin ensimmäinen varsinainen vaihe on sprintin suunnittelu, joka saa alkunsa tuotteen kehitysjonossa olevista kehitystöistä.

5. Sprintin suunnitellussa päätetään mitä kehitystöitä tuotteen kehitysjonosta lisätään sprintin kehitysjonoon.

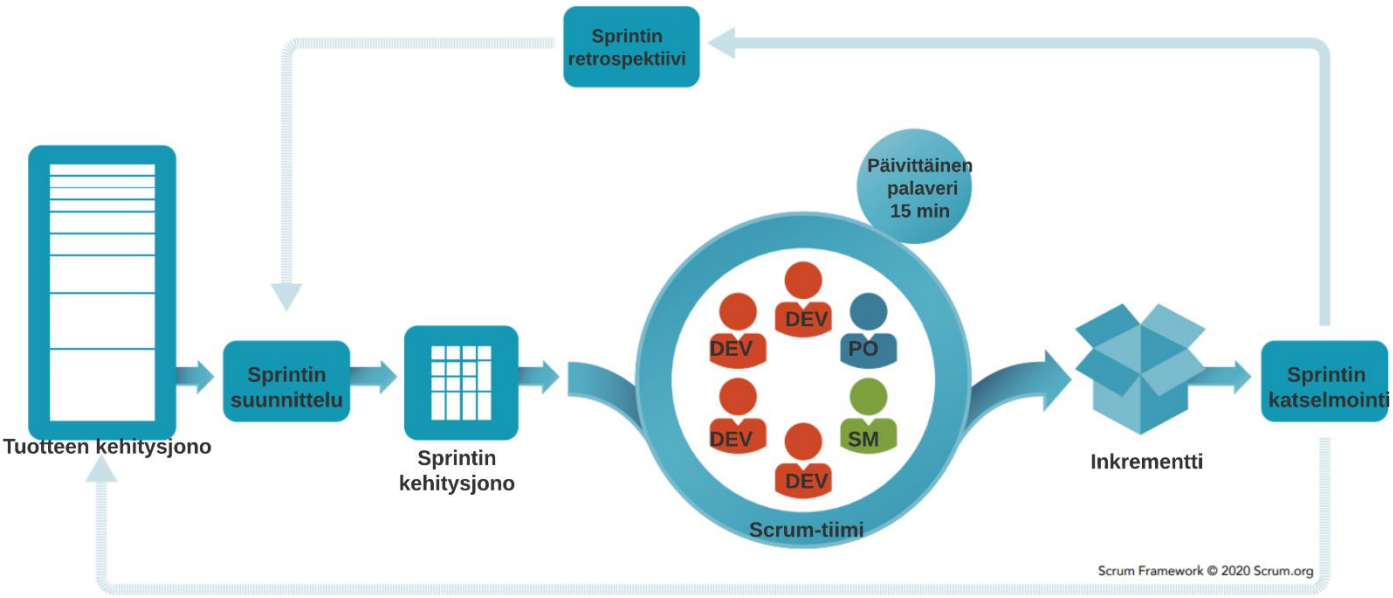
6. Työ alkaa sprintin kehitysjonossa olevien kehitystöiden osalta.

7. Jokaisen työpäivän aikana ennalta sovittuna ajankohtana on korkeintaan 15 minuuttia kestävä päivittäispalaveri, jossa suunnitellaan seuraavan 24 tunnin työt ja tarkastellaan työn edistymistä kohti sprintin tavoitetta.

8. Kun kehitystyö on tehty ja tuote on käyttövalmis sekä valmiina julkaistavaksi, on inkrementti valmis.

9. Sprintin katselmoinnissa käydään läpi kehitetty inkrementti. Tarkoitus on saada keskustelua ja palautetta aikaiseksi siitä missä onnistuttiin, missä olisi kaivannut parannusta. Asiakkaat, johto ja muut sidosryhmät voivat osallistua palaveriin nähdäkseen mitä on saatu aikaiseksi ja antaa samalla palautetta.

10. Viimeisenä vaiheena sprintissä on retrospektiivi, joka on sprintin viimeinen palaveri, johon osallistuu koko Scrum-tiimi. Palaverissa on tarkoitus tarkastella toimintaa ja keskustella mitä mahdollisia parannuksia voidaan tehdä jatkossa ja miten ne voidaan ottaa käyttöön tulevissa sprinteissä.
    (Tateeda, 2020)



Kuva 5 – Yhden sprintin kehitysprosessi Scrumissa. Lähde: Scrum.org
Lisätty erikseen tekstit lähteestä löytyneeseen tyhjään mallipohjaan.

Scrumin vahvuudet ja heikkoudet ovat esitetty alla olevaan yhteeenvetotaulukkoon.

Vahvuudet	Heikkoudet
Suuret kokonaisuudet saadaan pilkottua pienempiin, helposti hallinnoitaviin sprint-teihin.	Roolitus kehittäjien kesken ei välttämättä ole selkeästi määritelty, mikä voi tuoda epäjärjestystä tiimin jäsenten kesken.
Läpinäkyvyys päivittäisten palaverien ansiosta.	Scrum-metodin käyttäminen suuressa tiimissä on haastavaa.
Ketteryyden ansiosta tarvittavat muutostyöt saadaan tehtyä nopeasti.	Päivittäiset palaverit voivat joskus olla turhauttavia tiimin jäsenille.
Scrumin avulla saadaan käytettyä aika ja resurssit tehokkaasti.	Onnistuakseen, Scrum vaatii kokeneen tiimin.

Taulukko 4: Scrum-metodin vahvuudet ja heikkoudet. (Chandana, 2022)

Koska Agilessa ja Scrumissa molemmissa on iteratiivinen prosessi, kollaboratiivinen päätöksenteko ja jatkuva vuorovaikutus asiakkaan kanssa, voivat ne mennä sekaisin keskenään. Erot ovat kuitenkin selkeitä, nimittäin Agile on projektinhallinnan filosofia, joka hyödyntää sen sisällä olevia ydinarvoja tai perusperiaatteita, kun taas Scrum on Agilen yksi metodeista, joka helpottaa projektia. Agilessa iteraatioiden päätyttyä koko kehitystyö toimitetaan kokonaisuudessaan, kun taas Scrumissa työmäärä jaetaan helposti hallinnoitaviin sprintteihin ja pienempiin kokonaisuuksiin (Stobierski, T. 2021).

4 Tutkimusmenetelmät

Tämä toiminnallinen opinnäytetyö tehtiin toimeksiantona yritykselle, jossa olen töissä. Opinnäytetyön tyyppi on kvalitatiivinen eli laadullinen toimintatutkimus, jonka tarkoituksena on tutkia sosiaalista toimintaa liittyen kehitysprosessiin. Tämän työn osalta olemassa olevaa kehitysprosessia reflektoidaan eli peilataan kirjoitettuun teoriaan ja tutkimuksessa saatuun aineistoon ja tämän pohjalta pyritään löytämään tietoa, joka palvelisi kehitysprosessien käytänteiden kehittämistä. Tämän projektin osalta lopputuloksena syntyy parannusehdotelma kehitysprosessin ongelmakohtien suhteen.

Toimintatutkimukselle ei ole yhtä selkeää määritelmää, vaan sitä sovelletaan useilla tieteenaloilla. Eri koulukunnilla kirjallisuudessa on toimintatutkimukseen liittyen eroja menetelmissä, tavoitteissa ja taustaoletuksissa. Toimintatutkimuksen koulukunnille yhteistä on kuitenkin toiminnan havainnointi, reflektointi, toimintaan kiinnittyminen ja sen muuttaminen sekä teorian ja toiminnan liittäminen toisiinsa. (Juuti, P & Puusa, A. 2020, 256)

Yleensä kun puhutaan perinteisestä tutkimuksesta, johtaa sitä jokin teoreettinen intressi. Halutaan tietää millä tavalla asiat ovat toisin kuin toimintatutkimuksessa, jossa tavoitteena on löytää informaatiota, joka palvelee käytänteiden kehittämistä. Kyseisen tiedon etsimiseksi voidaan käyttää erilaisia tutkimusmenetelmiä. Tutkija ei ole ulkopuolinen, vaan osallistuu toimintatutkimuksessa aktiivisesti tutkimukseen tehden muutosinterventioita eli muutokseen tähtääviä väliintuloja. Toiminnalla tässä tapauksessa tarkoitetaan sosiaalista toimintaa, ihmisten toimintaa toisten ihmisten kanssa. (Valli, R. 2018, 182)

Juuti ja Puusa (2020, 256) mainitsevat, että toimintatutkimus yhdistetään enimmäkseen laadulliseen tutkimukseen. Toimintatutkimukselle tyypillistä on siinä yhdistyvä käytäntö ja teoria. Käytännönläheisyydestä huolimatta, toimintatutkimukselta vaaditaan käytettävän tieteen menetelmiä systemaattisesti.

Toimintatutkimuksen yhtenä tutkimusmenetelmänä käytettiin teemahaastatteluja. Valitsin teemahaastattelut siksi, että niiden avulla saa avointa keskustelua aikaiseksi haastateltavien osalta ja näin ollen haastateltavien näkemykset osastojen kehitysprosesseista nousevat paremmin esille. Teemahaastattelu myös mahdollisti haastateltavien omien kokemusten esiin tuomisen ja monipuolisemmat näkemykset aiheeseen liittyen.

Kaikkien haastattelujen litterointiin hyödynnettiin sisältöanalyysiä, jonka tarkoituksena on selvittää mistä aiheista, asioista ja teemoista analysoitava aineisto kertoo. Toisin sanoen tarkoituksena on sisällönanalyysin avulla karsia pois opinnäytetyöhön liittymättömät asiat ja pelkistää ilmaisuja kaikista haastatteluista.

Teemahaastattelun lisäksi osallistava havainnointi oli vahvasti mukana tässä toimintatutkimuksessa. Empiria tapahtui päivittäisessä työssä, palavereissa, kehitysprosessien seuraamisen osalta ja monessa muussa käytännön tilanteessa. Olen päässyt todistamaan käytännön tasolla useita kertoja kehitysprosessin etenemisen alusta loppuun saakka ja sen pullonkaulat, mutta myös vahvuudet. Tämän lisäksi olen tutkinut suurta määrää yrityksen sisäistä dokumentaatiota varsinkin kehitysprosessin osalta. Yrityksen sisäinen dokumentaatio onkin yksi kirjoitetun teorian lähteistäni.

4.1 Haastattelut

Osana tätä opinnäytetyötä haastateltiin neljää eri henkilöä kolmelta eri osastolta yhden sijaan, jotta otanta ja haastatteluista saatu aineisto olisi monipuolisempi. Haastattelussa kysyttiin työntekijöiltä heidän näkemyksiään osaston ja konsernin nykyisistä kehitysprosesseista. Haastattelut tapahtuivat Microsoft Teams -alustalla ja jokainen haastattelu nauhoitettiin litterointia varten. Haastattelut olivat kaikki tyypiltään puolistrukturoituja tiedonkeruumenetelmiä, joita voidaan kutsua myös teemahaastatteluiksi.

Puolistrukturoitu haastattelu sijoittuu teemahaastattelun ja täysin strukturoidun lomakehaastattelun välille. Puolistrukturoiduille haastatteluille on tyypillistä, että ne sisältävät tiettyjä vapauksia haastattelun osalta, kuten avoimia vastauksia ja kysymysten järjestyksen muuttamisen (Hirsjärvi & Hurme 2015, 47).

Haastattelukysymykset oli muotoiltu siten, että niihin vastataan avoimesti, joka mahdollisti laajemman aineiston keräämisen suhteessa haastateltavien määrään, verrattuna siihen mitä pelkästä lomakehaastattelusta, johon vastataan joko ”kyllä” tai ”ei”, olisi saanut. Kysymykset rajattiin kehitysprosessien ja niihin liittyvien viitekehysten ympärille. Haastattelujen avulla sain kattavan kuvan eri osastojen työskentelytavoista kehitysprosesseihin liittyen ja näin ollen laajemman kokonaiskuvan aiheesta.

Haastatteluja oli yhteensä neljä ja ne sijoittuivat marras-joulukuulle 2021. Kaikki haastattelut tapahtuivat etänä Microsoft Teams -verkkoalustalla, jossa ne myös nauhoitettiin. Lyhin haastattelu kesti 25 minuuttia ja pisin 65 minuuttia. Haastattelujen keskimääräinen kesto oli noin 45 minuuttia. Aluksi jokainen haastattelu litteroitiin eli kirjoitettiin puhtaaksi auki juuri sillä tavalla kuin haastattelutilanne oli oikeasti tapahtunut. Toisin sanoen kaikki mitä haastattelussa sanottiin, litteroitiin.

Haastateltaviksi valittiin pienestä otannasta huolimatta varsin edustava otos, sillä kaikki haastateltavat olivat keskeisessä roolissa kehitysprosessien suhteen. Haastateltavana olivat ammattinimikkeiltään projektipäällikkö, osastopäällikkö, kehityspäällikkö ja tuoteomistaja (Product Owner). Jokaisella oli vähintään kuuden vuoden kokemus ja pisin työkokemus, 20 vuotta, oli osastopäälliköllä. Työkokemuksen laskennassa on otettu huomioon koko yhtäjaksoinen työhistoria yrityksessä sisältäen myös aikaisemmat mahdolliset tehtävänimikkeet.

Taustatiedot haastateltavista:
Haastateltava 1: **Ammattinimike**: Kehitystyöhön erikoistunut luottokonsultti & projektipäällikkö (liiketoimintaosasto) **Työkokemus yrityksessä**: 6 vuotta, 4 kuukautta
Haastateltava 2: **Ammattinimike**: Osastopäällikkö (liiketoimintaosasto) **Työkokemus yrityksessä**: 20 vuotta
Haastateltava 3: **Ammattinimike**: Kehityspäällikkö (Osasto X) **Työkokemus yrityksessä**: 13 vuotta
Haastateltava 4: **Ammattinimike**: Product Owner (Osasto Y) **Työkokemus yrityksessä**: 6 vuotta

4.2 Aineiston analyysi

Kerätyn haastatteluaineiston analyysissä hyödynnettiin aineistolähtöistä sisällönanalyysiä. Aineistolähtöisessä sisällönanalyysissä on kolme vaihetta ja ne ovat aineiston redusointi eli pelkistys, aineiston klusterointi, siis ryhmittely ja abstrahointi, joka tarkoittaa teoreettisten käsitteiden luomista.

Haastatteluista saatu tietomäärä oli kovin laaja ja sen vuoksi aineistosta piti karsia tutkimuksen kannalta kaikki epäolennaiset ilmaisut ja asiat. Aineiston redusoinnissa karsitaan pelkistämisen lisäksi kaikki epäoleellinen tutkimuksesta pois ja saadaan näin ollen aineisto kompaktimpaan muotoon. Seuraavana vaiheena oli klusterointi, jossa redusointivaiheen aikana kerätyt alkuperäisilmaukset käydään tarkasti läpi ja etsitään joko samankaltaisia tai eroavia käsitteitä. Tämän jälkeen kyseiset käsitteet luokitellaan eri luokiksi, joista lopulta koostuu alaluokat. Abstrahoinnissa eli aineiston käsitteellistämisessä tarkoitus on erottaa tutkimukselle olennainen tieto ja seuraavaksi muodostaa siihen liittyvät teoreettiset käsitteet. Alkuperäisaineiston ilmauksista edetään teoreettisten käsitteiden kautta johtopäätöksiin. (Tuomi, Sarajärvi 2017; 93–95).

Alle olen esittänyt kolme kohtaa haastatteluissa esiintyvien alkuperäisten ilmaisujen pelkistämisestä. Taulukko on luotu pohjautuen Salmisen (2018, 22) opinnäytetyössä esitettyyn taulukkoon alkuperäisistä ilmaisuista ja niiden pelkistämisestä.

Alkuperäinen ilmaisu	Pelkistetty ilmaisu
”Scrum. Meillä oli niin paljon ongelmia ongelmia siinä vanhassa mallissa ja niiden kanssa taisteltiin jo pidemmän aikaa, niin sitten joku jossain vihdoin totesi et nyt on jotain isompaa muutosta saatava ja mä en sit tiiä tarkalleen mistä tää Scrum tuli onks se sit tällänen mihin toimeksiantajayritys on haluamassa ylipäätään mennä sitä kohti vai mikä, mutta se tuli sitte sitä vanhaa mallia sitte korjaamaan. Oli pakko tehä jotain isoja liikkeitä ja laittaa koko homma uusiks	Scrum. Ongelmia vanhassa mallissa. Muutos oli pakollinen uuden toimivamman mallin kehittämiseksi.
Sanotaanko et suurimmaks osaks se on tullu kyllä tästä työkokemuksen kautta ja matkalla oppien ku asiat on muuttunu n isit on taas opiskeltu et mitenkäs tässä uudessa mallissa täytyy toimia tän roolin kannalta, PO roolin kannalta oon käyny aikanaan koulutuksenkin ennen ku meillä oli edes tiedossa et tähän scrumiin sit siirrytään mut siitä oli vähän niinku alustavia ajatuksia, ni siihen oon käyny kurssin mut en oo hirveesti kyllä muita kursseja tähän käyny.	Työkokemuksen kautta hankittu osaaminen. Project Owner -roolin yhteydessä käyty Scrum-koulutus.
Ei ole, samalla tavalla toiminut koko prosessi vuodesta 2018.	Ei muutoksia vuodesta 2018.

Taulukko 1 – Esimerkki pelkistetyistä ilmauksista

Seuraava taulukko esittää muutaman esimerkin abstrahoinnista eli käsitteellistämisestä.

Pelkistetty ilmaisu	Alaluokka	Yläluokka	Pääluokka
Kehitysprosesseihin saatu osaaminen työkokemuksen kautta	Työkokemuksen myötä tullut osaaminen	Työkokemus	Tausta kehitysprosessien suhteen
Pitkä työkokemus kehitysprosesseihin osallistumisesta, mutta ei mukana kehittämässä niitä	Mukana kehitysprosessissa, ei sen kehityksessä		
Suorittanut Project Owner koulutuksen, enimmäkseen työkokemuksen kautta	Koulutuksen ja työkokemuksen kautta		

Taulukko 2 – Esimerkki ilmaisujen abstrahoinnista

5 Kehitysprosessien kuvaus

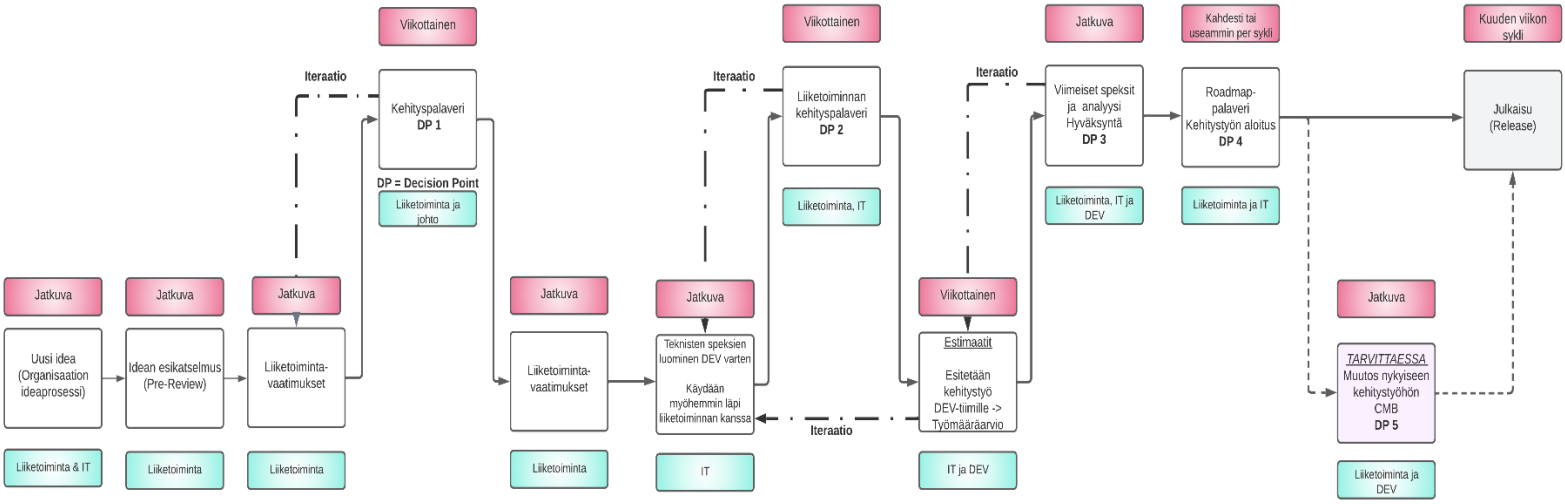
Kehitysprosesseja käytetään systemaattisesti mm. ohjelmisto- ja järjestelmäkehityksessä. Tämän opinnäytetyön osalta puhutaan järjestelmäkehityksen elinkaaresta (SDLC – Systems Development Life Cycle). Järjestelmäkehityksessä kehitystyö alkaa suunnittelusta ja selkeästi määritellystä prosessista, jota noudatetaan. Sekä järjestelmä-, että ohjelmistokehityksestä käytetään lyhennettä SDLC (Systems / Software Development Life Cycle) eli järjestelmä- tai ohjelmistokehityksen elinkaari (Techopedia, 2020) mutta tässä työssä viitataan SDLC:llä järjestelmäkehitykseen.

SDLC pitää sisällään metodologian, jossa on mukana selkeät, ennalta määritellyt prosessit laadukasta järjestelmäkehitystä varten (mondayblog, 2020). Suosittuja SDLC viitekehyksiä ja filosofioita ovat mm. Agile (ketterät menetelmät), Vesiputousmalli ja Iteratiivinen lähestymistapa (Zinios Edge, 2020.)

5.1 Nykymalli

Viimeisin versio liiketoimintaosaston kehitysprosessista on luotu tammikuussa 2020 eikä sitä näin ollen ole kahteen vuoteen päivitetty, vaikka tarve olisi tähän ollutkin. Koko kehitysprosessimalli sisältää 12 vaihetta kehitystyön edistämiseen liittyen. Vaiheissa korostuu myös kommunikaatio eri sidosryhmien kesken sekä ketterän kehityksen iteraatiot.

Seuraavan sivun Kuva 6 esittää ’ison kuvan’ kehitysprosessin kulusta. Prosessi käynnistyy organisaation omasta ideaprosessista ja noudattaa kuuden viikon syklejä. Termiä sykli käytetään sprintin sijaan siksi, että sovelluskehitysosastolla on omat erilliset kehityssprintit, joita ei tässä kuvassa esitetä, mutta ne etenevät syklin mukaisesti ja sisältyvät siihen. Punaisilla laatikoilla kuvan yläkulmassa on korostettu, onko kyseessä jatkuva, useasti vai pelkästään kerran esiintyvä vaihe. Joidenkin vaiheiden kohdalla voi tapahtua myös iteraatio, joka tuo ketteryyttä kehitysprosessiin. Iteraation kohdalla voidaan siis palata takaisin aikaisempaan vaiheeseen esimerkiksi hiomaan kehitystyön vaatimuksia paremmaksi ja sen jälkeen edetä taas seuraavaan vaiheeseen. Kuvassa esiintyy termi Decision Point eli DP, joka tarkoittaa tässä tapauksessa päätöspistettä. Tällöin tehdään päätöksiä siitä, voiko kehitystyö edetä seuraavaan vaiheeseen vai pitääkö esimerkiksi vaatimuksia vielä hioa.



Kuva 6 - Liiketoimintaosaston kehitysprosessi, johon osallistuu myös IT-osasto ja johto

Tärkeänä huomiona kehitysprosessista mainittakoon, että siinä esiintyvät vaiheet eivät tapahdu saman syklin aikana, toisiksi viimeistä vaihetta lukuun ottamatta (CMB). Kehitysprosessissa suunnitellaan seuraavan syklin kehitystöitä. Mikäli kehitystöitä lisättäisiin jatkuvasti saman syklin etenemissuunnitelmalle (Roadmap), niin aiheuttaisi se sekaannusta ja kiirettä kehitystöiden osalta. Kun kehitystyöt suunnitellaan seuraavalle syklille, pysyy kokonaisuus organisoituna ja järjestelmällisenä. Etenemissuunnitelmassa ennalta määritetään tietyn sprintin (tässä tapauksessa syklin) kehitystyöt ja se ohjaa päivittäistä työtä (Radigan, D).

Ensimmäisessä vaiheessa ideasta on jo tiketti luotuna tiketinhallintajärjestelmään, jonka IT- ja liiketoimintaosasto yhdessä käyvät läpi. IT-osaston on tarkoitus arvioida idean tekniset toteutusmahdollisuudet, kun taas liiketoimintaosasto esittää liiketoiminnalliset aspektit, mutta kummatkin osapuolet voivat ottaa molempiin aiheisiin kantaa. Mikäli IT-osaston mielestä idea ei ole teknisessä mielessä toteutuskelpoinen, hylätään kyseinen idea siihen paikkaan tai ehdotetaan vaihtoehtoisia tapoja idean toteutukselle.

Toisessa vaiheessa liiketoimintaosaston työntekijä toteuttaa idean esikatselmuksen ja kerää tietoa toteutettavasta ideasta. Kerättävä tieto voi olla statistiikkaa, jota voidaan

hyödyntää idean osalta. Toiseen vaiheeseen kuuluu myös ideoiden miettiminen ja tarvittaessa näkemysten esittäminen muille, jolloin he voivat ilmaista mielipiteensä idean osalta. Tämän jälkeen päätetään, siirretäänkö idea seuraavaan vaiheeseen vai ei.

Jos idea etenee, niin kolmas vaihe kehitysprosessissa on liiketoimintatarpeiden ja liiketoimintavaatimusten määrittely sekä hyötyjen puntarointi. Tämän jälkeen kirjoitetaan ideasta ”Story”, joka on oma kategoria tiketinhallintajärjestelmän kehitystöille. Story tarkoittaa liiketoimintalähtöistä, ennalta suunniteltua kehitystyötä. Tätä ei tule sekoittaa käyttäjätarinaan eli user storyyn, jossa kerrotaan käytännön tasolla millaisia ominaisuuksia tuotteella on ja miten ne toimivat käyttäjän näkökulmasta.

Neljäntenä vaiheena ideaprosessissa on kehityspalaveri **(DP 1)**, jossa liiketoimintaosasto ja johto tapaavat ja keskustelevat liiketoimintaosaston kehitystöiden nykytilanteesta, mitä kehitystöitä etenemissuunnitelmassa on seuraavaksi luvassa ja mitkä ovat viisi tärkeintä, työn alla olevaa kehitystyötä. Tässä vaiheessa päätetään, eteneekö idea seuraavaan vaiheeseen, tarvitaanko siitä lisätietoa vai hylätäänkö idea.

Tämä on myös kehitysprosessissa se vaihe, jolloin voi tapahtua ensimmäinen iteraatio. Mikäli idea hylätään, voidaan palata edelliseen vaiheeseen takaisin ja kirjoittaa idean liiketoimintavaatimukset uudelleen sekä puuttua mahdollisiin epäkohtiin, joita ideasta on kehityspalaverissa esitetty.

Viidennessä vaiheessa hiotaan liiketoimintavaatimuksia ja käyttäjätarinoita ja tarkistetaan onko vaatimukset yleisesti ottaen kunnossa. Tässä vaiheessa tyypillistä on liiketoimintavaatimusten täsmentäminen ja selkeyttäminen.

Kuudenteen vaiheeseen osallistuu pelkästään IT-osasto ja se koskee teknisten vaatimusten kirjoittamista kehitystöiden tiketteihin, jolloin sovelluskehittäjät tietävät mitä tehdä teknisestä näkökulmasta katsottuna. IT-osaston luomat tekniset vaatimukset kerrotaan eteenpäin liiketoiminnalle joko päivittäisissä tai viikoittaisissa palavereissa.

Seitsemäs vaihe **(DP 2)** pitää sisällään liiketoimintaosaston kehityspalaverin, johon osallistuu liiketoiminnan lisäksi IT-osasto. Palaverissa käydään läpi uusia käyttäjätarinoita ja tarkastellaan aikaisemmin luotuja vaatimuksia.

Tässä vaiheessa arvioidaan myös teknisten tehtävien toteutusta samalla ja mikäli ne hyväksytään, niin lisätään ne seuraavan syklin etenemissuunnitelmalle. Mikäli tehtäville ei anneta hyväksyntää, siirretään ne myöhemmäksi.

Teknisillä tehtävillä (Technical tasks) tarkoitetaan sellaisia tehtäviä, joita sovelluskehittäjät luovat tiketinhallintajärjestelmään. Nämä tehtävät liittyvät esimerkiksi järjestelmän päivitykseen tai olla osa jonkin kehitystyön kokonaisuutta. Tekniset tehtävät ovat joidenkin tikettien osalta välttämättömiä eikä ilman niitä kehitystyötä voida muuten edistää.

Kahdeksannessa vaiheessa IT-osasto esittelee kehitystyön sovelluskehittäjille ja selvittää karkean estimaatin eli aika-arvion kehitystyöhön ja testaukseen käytettävien työtuntien osalta. Mikäli estimaattia ei hyväksytä palautuu kehitystyö iteraation kautta takaisin kuudenteen vaiheeseen, vaatimusten määrityksiin, jossa niitä tarkastellaan uudelleen ja korjataan mahdolliset ongelmakohdat.

Yhdeksännessä vaiheessa **(DP 3)** karkeiden estimaattien jälkeen on aika viimeistellä ja analysoida kehitystyön vaatimukset IT-osaston toimesta. Sovelluskehitysosasto kirjoittaa ohjelmistokehittäjän näkökulmasta To-Do-listan tikettiin, joka ohjaa ohjelmistokehittäjien toimintaa kehitystyön osalta ja lisää läpinäkyvyyttä. Toisin sanoen kaikki sovelluskehittäjätiimin henkilöt pääsevät näkemään, mitä kehitystyöltä vaaditaan sovelluskehittäjän näkökulmasta.

Kymmenennessä vaiheessa **(DP 4)** pidetään Roadmap-palaveri (etenemissuunnitelmapalaveri), johon osallistuu IT- ja liiketoimintaosasto. Tässä vaiheessa päätetään seuraavan julkaisuun liittyvät aikataulut. Kapasiteettiluvut eli sovelluskehittäjien kehitystyöhön käytettävät työtunnit kirjataan ylös, jotta pysytään kartalla siitä, miten paljon resursseja voidaan mihinkin kehitystyöhön allokoida.

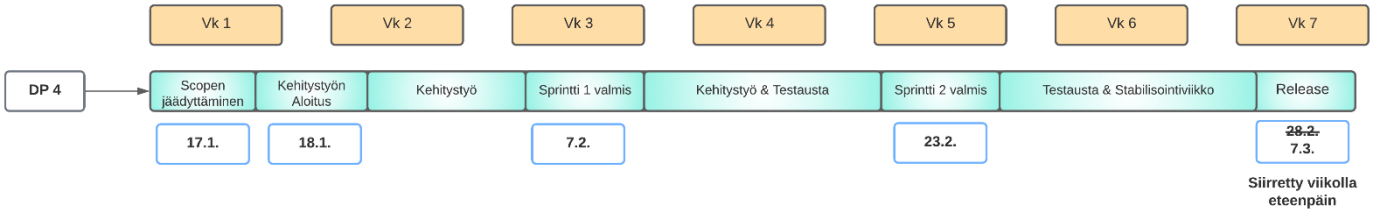
Samalla päätetään myös, miten laaja koko sykli on ja milloin tapahtuu ’scope freeze’ eli syklin laajuuden jäädyttäminen, jonka jälkeen ei siis oteta enää uusia kehitystöitä kyseisen syklin etenemissuunnitelmaan mukaan. Syklin julkaisupäivä päätetään myös tuossa palaverissa. Näiden lisäksi tehdään myös päätös siitä, minä päivänä varsinainen kehitystyö alkaa. Viimeisimmässä syklissä kehitystyö on alkanut päivä scope freezen jälkeen.

Viimeinen vaihe ennen julkaisua on CMB eli Change Management Board, joka on ainoa vaihe, jossa voidaan tehdä tarvittaessa, painavasta syystä, muutoksia käynnissä olevan syklin kehitystöihin. Tämä vaihe ei ole siis pakollinen, vaan se voidaan jättää prosessista

kokonaan pois, jos siihen ei ole tarvetta. CMB:hen osallistuu liiketoiminta- ja sovelluskehitysosasto. Esimerkkitilanne, jossa CMB:tä tarvitaan voisi olla kesken syklin muuttuneet asiakkaan vaatimukset tuotteesta. Tämä on siis vaiheena valinnainen, jonka vuoksi se on merkitty katkoviivoilla.

Viimeisenä kuvassa näkyykin Release eli julkaisu, johon päättyy kuuden viikon sykli. Seuraava sykli alkaa seuraavalla viikolla syklin päättymisestä.

Äskeisestä kuvasta ei ilmennyt sovelluskehittäjien sprinttejä ollenkaan, vaan ne avataan alla olevassa aikajanassa tarkemmin käytännön case-esimerkkinä. Toisin sanoen, kyseinen aikataulu kuvastaa oikeaa kehitystyötä toimeksiantajayrityksen osastolla. Aikajana on joiltakin osin suuntaa antava eikä välttämättä kuvasta koko totuutta sprintin suhteen. Joitakin asioita on yksinkertaistettu ja pelkistetty, jotta kokonaisuus on voitu esittää helposti ymmärrettävällä, selkeällä aikajanalla.



Kuva 7 – Case-esimerkki. Aikajana sovelluskehitysosaston kehitystyöstä. Myös IT-osasto osallistuu kehitystyöhön mm. testauksen myötä (käyttäjä- ja regressiotestit)

Sykli alkaa laajuuden jäädyttämisestä, josta on päätetty DP 4:ssä eli Roadmap-palaverissa. Laajuuden jäädyttämisellä tarkoitetaan sitä, että Roadmapille ei enää hyväksytä uusia kehitystöitä, vaan aloitetaan kehitystyöt niiden tikettien osalta, mitkä on entuudestaan Roadmapille suunniteltu. Varsinainen kehitystyö alkaa 18.1. Kuvassa on otettu myös huomioon IT-osastolle kuuluvat tehtävät kehitystyön edetessä.

Syklin aikana julkaistaan useampi sprintti, jonka jälkeen sovelluskehittäjien kehitystyö jatkuu. IT-osaston vastuulle jää julkaistujen sprinttien tuotteiden käyttäjätestaus, jossa testataan toimintoja tuotteen tulevan käyttäjän näkökulmasta. Viikkoa ennen julkaisua (Release) tapahtuu nk. stabilisointiviikko, jossa keskitytään pelkästään käyttäjä – ja regressiotesteihin. Regressiotesteissä testataan, toimiiko aikaisemmin toimineet ominaisuudet uudistetussakin järjestelmässä.

6 Tutkimustulokset

Tässä toimintatutkimuksessa oli tarkoituksena selvittää, miten yhden osaston kehitysprosessia voitaisiin kehittää paremmaksi toimeksiantajayrityksessä. Tähän kysymykseen vastaamista lähdettiin toteuttamaan puolistrukturoitujen haastattelujen ja osallistavan havainnoinnin kautta tarkastellen ongelmakohtia ja parannusehdotuksia nykyiseen kehitysprosessimalliin. Tämän lisäksi nykyistä mallia reflektoitiin kirjoitettuun teoriaan ja hyviin käytäntöihin aiheesta, jotka voidaan ajatella ikään kuin ’ideaalimallina’.

Ongelmakohtien tarkastelun lisäksi tarkastellaan myös nykyisen mallin vahvuuksia, jotta tiedetään mitä ainakaan ei tulisi muuttaa ja mihin tulisi jatkossakin keskittyä uudessa mallissa. Olen koonnut yhteenvetona kappaleiden 6.1 ja 6.2 loppuun erilliset taulukot, josta kiteytyy kappaleissa käsiteltävät ongelmakohdat, niiden parannusehdotukset ja kehitysprosessin vahvuudet.

6.1 Ongelmakohdat ja parannusehdotukset

Tämän kappaleen alla olevat alaotsikot kuvaavat konkreettisesti löytyneitä ongelmakohtia kehitysprosessista. Suurimmat ongelmakohdat koko kehitysprosessissa liittyivät päivittäisiin palavereihin, kehitysjonoon ja julkaisuun. Ongelmakohtien lisäksi yhteenvetotaulukossa esitetään parannusehdotus jokaiselle ongelmakohdalle. Yhteenvetotaulukossa tosin vain tiivistetään parannusehdotukset. Liitteessä 2 on esitetty parannusehdotukset tarkemmin.

6.1.1 Päivittäiset palaverit

Yrityksen IT-osastolla työskennellyt **Product Development Specialist** (PDS) oli ollut aikaisemmin samassa yrityksessä töissä liiketoimintapuolen tehtävissä ja auttoi liiketoimintavaatimusten tekemisessä. Tämän lisäksi hän toimi **Product Ownerin** (tuoteomistajan) roolissa ja oli vastuussa tuotekehitysjonon hallinnasta ja tikettien priorisoinneista, jotka ovat erittäin tärkeitä tehtäviä kehitysprosessin kokonaisuuden hallinnassa. Kun PDS päätti työsuhteensa 01/2021 (ks. s.6, Kuva 2 – henkilöstömuutoskaavio), aiheutti tämä luonnollisesti tyhjiön osaamisalueeseen, jonka takia toimintatapoja piti miettiä uusiksi, jotta toimintaan saataisiin jonkinlainen jatkumo. Seuraavaksi liiketoimintaosasto otti tuotemistajan roolin harteilleen, mutta kyseisen roolin tehtävät aiheuttivat resurssiongelmia, sillä

liiketoimintaosaston ja PO-roolin tehtäviin oli hankala löytää aikaa tasapuolisesti. Eräs liiketoimintaosaston työntekijä mainitsi, että Product Owner -rooli on kokopäivätyö ja siihen pitäisi nimetä erikseen oma henkilö.

Liiketoimintavaatimusten tekemisessä oli myös mukana **Service Manager** (SM). Pitkäaikaisen työkokemuksen ansiosta SM pystyi kommunikoimaan ja vaikuttamaan suoraan liiketoimintavaatimuksiin suhteen ilman ’ylimääräisiä’ välikäsiä tai palavereja. Näin ollen prosessi oli varsin sujuva. Hänen työsuhteensa päättyessä hävisi siis toinenkin liiketoimintavaatimusten osaava tekijä.

Liiketoimintaosasto seuraavaksi mietti uutta läpinäkyvää tapaa käydä läpi päivittäistä työtä ja uusia tikettejä, jotka IT-osastolle kuuluvat. Keskiössä oli ajatus siitä, että uudet, ennalta suunnittelemattomat tiketit saadaan työn alle eivätkä ne jää huomiotta. Tämä johti siihen, että alettiin pitämään päivittäisiä, 30 minuuttia kestäviä palavereja liiketoimintaosaston ja IT-osaston välillä, jonka tarkoituksena on ollut käydä läpi nykytilannetta yleisellä tasolla, saman vuorokauden aikana tapahtuvia työtehtäviä sekä uusia tikettejä.

Päivittäinen 30 minuutin palaveri ei ole ollut kuitenkaan ongelmaton, sillä se on välillä tuhlannut työntekijöiden aikaa ja tämän lisäksi palaverin aikana nostetut asiat tai kehitystyöt ovat ohittaneet prioriteetiltaan tärkeämmät työlistalla olevat asiat. Tämä on osaltaan sotkenut etenemissuunnitelmalla (Roadmap) olevia kehitystöitä, jotka eivät etene sovitun mukaisesti.

Työntekijöiden aikaa on kulunut siihen, kun päivittäisessä palaverissa ei ole joka kerralla varsinaista agendaa läpikäytävistä asioista, vaan usein mietitään useitakin minuutteja, että ”Olikos meillä tänään mitään asiaa?” ”Mistäs tänään puhutaan?”. Usein ongelmana on myös, että palaverissa nostetut asiat eivät välttämättä koske tai ole oleellisia kaikille palaveriin osallistujille. Tämä on ongelmallista, sillä tässä kuluu jokaisen työntekijän aikaa. Jos ajatellaan, että palaverissa on paikalla kuusi työntekijää ja kuluu 10 minuuttia pelkästään agendan pohtimiseen, niin tässä tapauksessa on jo kulunut 60 minuuttia työaikaa kaikilta yhteensä. Mikäli tämä toistuu vaikkapa kolme kertaa viikossa vuoden ajan (3x60x52), on tuhlattu **9360 minuuttia**, joka tekee **156 tuntia**, ja vastaa yhden kokoaikaisen työntekijän työtunteja (160 h) kuukauden ajalta.

6.1.2 Kehitysjono (Backlog)

Yleisesti ottaen kehitysjonon hallinnoiminen on paikoin osoittautunut haastavaksi ideaprosessin takia. Ideaprosessin tarkoitus on saada sellaisia potentiaalisia ideoita eteenpäin,

joilla on positiivinen vaikutus esimerkiksi palveluun, tuotteen ominaisuuteen tai liikevaihtoon. Monesti kuitenkin on ajateltu, että ”Tämä ideahan on ihan kiva, jätetään se Backlogille”, joka on osaltaan paisuttanut kehitysjonoa liian suureksi ja tehnyt sen hallinnasta haastavaa. Tämä voi myös osaltaan sotkea prioriteetteja kehitystöiden suhteen – voi olla hankalaa hahmottaa mitkä kehitystyöt ovat tärkeämpiä kuin toiset, kun kehitystöiden määrä kehitysjonossa kasvaa.

6.1.3 Julkaisu (Release)

Tavoitteena on nykyisen kehitysprosessin mukaisesti, että jokaista kuuden viikon sykliä kohden julkaistaan uusi julkaisu. Usein aikataulu julkaisun suhteen on pysynyt ennallaan, mutta joskus julkaisu on venynyt seitsemään, jopa kahdeksaan viikkoon, mutta tätä en nostaisi ongelmakohdaksi, sillä ainahan on riski, että asiat viivästyvät esimerkiksi odottamattomien muutosten takia. Tässä tapauksessa julkaisun aikataulun venyminen on muutenkin osaltaan johtunut uusista IT-osaston työntekijöistä, joilla ei ole ollut riittävästi osaamista vielä Releaseen liittyvien tehtävien hallinnoinnissa. Tilanne siis parantunee itsestään tämän osalta, kun työkokemusta ja osaamista kertyy.

Ongelmia sen sijaan on esiintynyt julkaisun jälkeisenä ajanjaksona. Kun sykli on valmistunut ja Release on saatu tuotantoon, niin on ollut välillä epäselvyyttä siitä, että mitä seuraavaksi pitäisi tehdä ja missä ajassa.

Julkaisun jälkeen on ollut pohdintaa siitä, onko sovelluskehitysosastolla tarpeeksi töitä vai odottavatko he liiketoiminnalta ja IT-osastolta kehitystöitä uuteen sykliin kuuluvalle sprintille. Mikäli sovelluskehitysosastolla ei ole tarpeeksi töitä, on tämä pois tehokkuudesta ja tuottavuudesta. Toimettomuuden sijasta sovelluskehittäjät voisivat edistää tuotekehitysjonossa esiintyviä töitä.

Uuden julkaisun tuotantoon viemisen yhteydessä pitäisi olla jo selvillä seuraavan julkaisun päivämäärät, joihin sisältyy mm. kehitystyön alkaminen ja tuotantoon vieminen. Mikäli näitä päivämääriä ei ole ei se luonnollisestikaan ohjaa tekemistä kenenkään osalta ja voi aiheuttaa hämmennystä, kun ei ole selvillä mikä on aikataulullisena tavoitteena. Kun aikataulu uuden julkaisun osalta on selvillä, niin on paljon helpompi mukauttaa kehitystöitä ja niiden priorisointeja siihen. Tämä mahdollistaa myös päätöksen siitä, että milloin tehdään ’Scope freeze’ eli jäädytetään syklin laajuus, eikä oteta enää uusia suunniteltuja kehitystöitä uuden syklin etenemissuunnitelmalle tietyn ajan jälkeen.

6.1.4 Ongelmakohtien ja parannusehdotusten yhteenveto

Nykyisen kehitysprosessimallin ongelmakohdat liittyvät päivittäisiin palavereihin, kehitysjonon hallintaan, kehitysprosessin noudattamiseen, syklin lopussa tehtävään julkaisuun ja rooleihin. Alle olen esittänyt yhteenvetotaulukkoon tiivistettynä aiheen, siihen liittyvät ongelmakohdat ja parannusehdotukset. Parannusehdotuksia avataan paremmin liitteessä 2, mutta alla olevassa yhteenvetotaulukosta näkee tämän tiedon tiivistettynä.

Aihe	**Ongelma**	**Parannusehdotus**
Päivittäiset palaverit	Kaikissa palavereissa ei agendaa, johtaa strukturoimattomaan kokoukseen. Käsiteltävät asiat eivät koske kaikkia osallistujia Edellä mainitut seikat tuhlaavat työntekijöiden resursseja.	Ennalta suunniteltu agenda palavereja varten. Vain osa henkilöistä osallistuisi kokoukseen kolmesti viikossa. Täysi kokoonpano vain kaksi kertaa viikossa. 15min palaverin kesto jatkossa (Scrum).
Kehitysjono (Backlog)	Liikaa ideoita kehitysjonossa ideaprosessin myötä, vaikeuttaa kehitysjonon hallintaa ja kehitystöiden priorisointia Osa ideoista jää roikkumaan kehitysjonoon pidemmäksi aikaa.	Ideoiden tarkempi suunnittelu. Kehitysjonoon pidemmäksi aikaa jääneiden tikettien katselmointi tai siistiminen kerran syklin aikana.
Kehitysprosessi	Kehitysprosessia ei aina noudateta ihmisten toimesta.	Sitoutuminen noudattamaan kehitysprosessia. Kommunikoinnin lisääminen liittyen siihen, miksi ei ole noudatettu prosessia.

	Kehitystöiden lisääminen etenemissuunnitelmalle vastoin kehitysprosessin käytäntöä johtaa kiireeseen, riittämättömiin vaatimuksiin ja huonoon lopputulokseen	Jokaisesta kehitystyöstä tulisi tehdä tiketti, jotta kehitysprosessi etenee normaalin käytännön mukaisesti.
Julkaisu (Release)	Sovelluskehittäjien resurssien tehokas hyödyntäminen Puutteelliset aikataulut julkaisun suhteen. Retrospektiivin puuttuminen (Scrum). Ei päästä hyödyntämään oman toiminnan kehitysmahdollisuutta julkaisun yhteydessä.	Pitkän tähtäimen suunnittelu ja resurssien allokointi kehitystöiden osalta. Etukäteen pitää olla suunniteltuna seuraavan kahden syklin aikataulut. Retrospektiivien pitäminen jatkossa, kommunikointi ja tavoitteiden asettaminen seuraavaa julkaisua varten.
Roolit	Kokoaikaisen tuoteomistajan (PO) puuttuminen. Liikaa kehitystyötä liiketoimintaosaston vastuulla.	Yhden IT-osaston työntekijän nimittäminen tuoteomistajan tehtävään.

Taulukko 5: Ongelmakohtien ja niihin liittyvien parannusehdotusten yhteenveto.

6.2 Nykyisen mallin vahvuudet

Vaikka nykyisessä kehitysprosessimallissa on ongelmakohtia, jotka vaativat muutosta, että kehitysprosessi toimisi optimaalisesti, on nykyisessä mallissa vahvuuksiakin. Tähän opinnäytetyöhön liittyvissä haastatteluissa selvitettiin kahdelta liiketoimintaosastolla työskennelleeltä henkilöltä (ks. s.19, haastateltavien taustatiedot -taulukko), mitkä ovat nykyisen kehitysprosessin vahvuudet ja selvisi, että kehitysprosessissa ei ole suoranaisesti vikaa – se on selkeä, hyvin määritelty ja sillä saa kehitystyötä edistettyä tehokkaasti.

Vahvuutena mainittiin myös tiimin (Liiketoiminta-, IT- ja sovelluskehitysosasto) pienen kokoonpanon tuoma ketteryys ja helppous prosessin läpiviennissä. Kommunikointi on helppoa ja asioihin voi vaikuttaa tehokkaasti.

Tämän lisäksi tiedon löytäminen ja läpinäkyvyys on yleisesti ottaen hyvällä mallilla. Välillä tieto voi toki olla vanhentunutta tai sen löytämisessä voi kestää, mutta yksi haastateltavista pohti, että etsittävä tieto löytynee helpommin pienessä tiimissä, kuin isommista tiimeistä etsittäessä.

Vahvuuksien yhteenvetotaulukko:

Aihe	Vahvuus
Kehitysprosessi	Selkeä ja helposti ymmärrettävä prosessi. Kehitysprosessia pääsääntöisesti noudatetaan ja kunnioitetaan hyvin. Päätöksentekopisteet (DP) luovat struktuuria ja tuovat selkeyttä kehitysprosessiin. Iteratiivinen lähestymistapa pitää kehitysprosessin ketteränä ja valmiina muutoksille.
Kokoonpano (Liiketoiminta, FIN IT ja DEV)	Palautteen antaminen toimii matalalla kynnyksellä, joka osaltaan ohjaa yksilön kehitystä toiminnassaan.

	Kommunikointi ja läpinäkyvyys toimivat hyvin. Motivoituneet, osaavat työntekijät.
Tiedon löytäminen ja läpinäkyvyys	Helpot ja selkeät alustat tiedon jakamiselle. Vanhentunutta tietoa pääsee kuka tahansa käyttöoikeuden saanut työntekijä päivittämään eli matala byrokratia sen osalta. Päätökset kehitystöistä läpinäkyviä. Firman sisäisen dokumentaation käyttöoikeudet hyvällä mallilla. Tarvittavat osapuolet pääsevät käsiksi tarvittaviin dokumentteihin.

Taulukko 6: Nykyisen kehitysprosessin vahvuudet

7 Johtopäätökset

Tämän toimintatutkimusopinnäytetyön viimeisessä kappaleessa keskitytään tutkijan eli tässä tapauksessa kirjoittajan pohdintaan ja lopullisiin päätelmiin tutkimustuloksista sekä miten niitä voidaan hyödyntää lopputuloksen kannalta. Tavoitteena tällä opinnäytetyöllä oli vastata tutkimuskysymykseen: ”Miten nykyistä kehitysprosessia voidaan parantaa?”. Lopputuloksena syntyi parannusehdotelma, jossa nostetaan esille liiketoimintaosaston nykyisessä kehitysprosessimallissa ilmenneet ongelmakohdat ja niiden parannusehdotukset.

Tutkimustulokset -kappaleessa esitettiin nykyisen kehitysprosessimallin ongelmakohdat ja vahvuudet. Ongelmakohtia nousi esille huomattavasti enemmän kuin vahvuuksia, joten selvää on, että nykyisessä mallissa on parannettavaa. Johtopäätös siitä, mikä voisi olla toimiva malli jatkossa, kiteytyy empiiristen havaintojen, haastattelujen, aineiston analyysin, ja kirjoitetun teorian ympärille.

Haastatteluaineiston avulla pystyttiin selvittämään suuri osa liiketoimintaosaston kehitysprosessin ongelmakohdista, vahvuuksista ja parannusehdotuksista. Aineisto toi myös selkeyttä liiketoimintaosaston kehitysprosessin vaiheisiin ja auttoi sen visuaalisessa esittämisessä (ks. Kuva 6 & Kuva 7). Vaikka kaksi haastateltavista työskentelevät liiketoimintaosastolla ja kaksi muuta eri osastoilla, oli kaikista haastatteluista silti hyötyä, sillä niiden avulla pystyttiin selvittämään toistuvia teemoja kehitysprosessin onnistumisesta, epäonnistumisesta ja toivotuista parannusehdotuksista. Samaten saatiin kehitysprosessin kokonaiskuvasta lisätietoa.

Lopputuloksena on luotu ehdotelma parannetuista toimintatavoista, joka on lisätty tämän opinnäytetyön toiseksi liitteeksi. Parannusehdotelmassa kerrotaan esille nousseiden ongelmakohtien taustoista, mikä on niiden vaikutus tällä hetkellä ja samalla esitetään parannusehdotukset niihin. Ongelmakohtiin voi löytyä useita vaihtoehtoja parannusehdotusten suhteen, mutta nämä vaihtoehdot eivät ole kuitenkaan toisiaan poissulkevia, vaan on eriytetty toisistaan selkeyden vuoksi.

Toimintatutkimusta aloittaessani, minulla oli ongelmia aiheen rajauksen suhteen. Aluksi aihe oli liian laaja enkä ollut täysin varma, mitä tutkimuksesta tulisi jättää pois. Aihe kuitenkin tarkentui toimintatutkimuksen edetessä ja loppua kohden alkoi olemaan selkeämpää, mihin asioihin tämän opinnäytetyön kannalta tulisi keskittyä. Opinnäytetyötä tehdessä pääsin perehtymään kattavasti kehitysprosessien teoreettisiin viitekehyksiin ja vaikuttamaan muutosinterventioilla kehitysprosessin nykymalliin.

Jos mietitään lopputulosta tämän opinnäytetyön osalta, voin sanoa olevani varsin tyytyväinen siihen. Tiedonkerääminen onnistui hyvin osallistavan havainnoinnin, haastattelujen, toimeksiantajayrityksen sisäisen dokumentaation ja muiden valmiiden kirjoitettujen lähteiden osalta. Haaga-Helian opinnäytetöiden raportointiohje loi pohjan tämän opinnäytetyön rakenteelle ja samalla antoi eväät tutkimusmenetelmien valinnalle. Sain tutkimuksen avulla kuvattua liiketoimintaosaston kehitysprosessin nykymallin ja siinä ilmenneet ongelmakohdat selkeästi. Tämän lisäksi onnistuin esittämään ongelmakohdille parannusehdotukset, jotka tekevät kehitysprosessimallista paremman.

Lähteet

Agile Manifesto. Luettavissa: https://agilemanifesto.org/. Luettu 18.2.2022.

Business Technology Standard. Johdanto – Bisnesteknologiamalli. Luettavissa: https://btmalli.fi/book/introduction/introduction-to-business-technology-standard/. Luettu 12.12.2021

Chandana. 2022. Scrum Project Management: Advantages and Disadvantages. Luettavissa: https://www.simplilearn.com/scrum-project-management-article. Luettu: 18.2.2022.

Cprime, What is Agile? What is Scrum? Luettavissa: https://www.cprime.com/resources/what-is-agile-what-is-scrum/. Luettu: 15.1.2022

GeeksForGeeks, Classical Waterfall Model. Luettavissa: https://www.geeksforgeeks.org/software-engineering-classical-waterfall-model/. Luettu 26.2.2022

GeeksForGeeks, Differences between Verification and Validation. Luettavissa: https://www.geeksforgeeks.org/differences-between-verification-and-validation/?ref=leftbar-rightbar. Luettu: 26.2.2022

Hirsjärvi, S. & Hurme, H. 2015. Tutkimushaastattelu. Gaudeamus Helsinki University Press Oy Yliopistokustannus, HYY yhtymä

J. MacKay, 2019 Software Development Process: How to Pick The Process That's Right For You. Luettavissa: https://plan.io/blog/software-development-process/#the-5-best-software-development-processes-and-how-to-pick-the-right-one-for-you Luettu: 14.1.2022.

Juuti, P & Puusa, A. 2020. Laadullisen tutkimuksen näkökulmat ja menetelmät. Gaudeamus Oy.

Kajaanin AMK, Toimintatutkimus. Luettavissa: https://www.kamk.fi/fi/opari/Opinnaytetyopakki/Teoreettinen-materiaali/Tukimateriaali/Toimintatutkimus Luettu: 20.12.2021.

Lamelas, A. 2018. Top 5 main Agile methodologies: advantages and disadvantages. Luettavissa: https://www.xpand-it.com/blog/top-5-agile-methodologies/. Luettu 18.2.2022.

Lucidchart. The Stages of the Agile Software Development Life Cycle. Luettavissa: https://www.lucidchart.com/blog/agile-software-development-life-cycle. Luettu: 18.2.2022.

M. Martinsuo and M. Blomqvist, 2010. Process modeling for improved performance.

Mondayblog, 2020. What is Agile SDLC and how should you use it in 2021? Luettavissa: https://monday.com/blog/rnd/agile-sdlc/ Luettu: 14.1.2022.

Stobierski, T. 2021. Agile vs. Scrum: What's the Difference. Luettavissa: https://www.northeastern.edu/graduate/blog/agile-vs-scrum/. Luettu 18.2.2022.

Pries H & Quigley M, Jon M. 2010. Scrum Project Management. CRC Press.

Oulun AMK 2018. Toiminnallisen opinnäytetyön oppimiskokemukset. Luettavissa: https://www.theseus.fi/bitstream/handle/10024/152055/ePooki%2045_2018.pdf?sequence=1. Luettu: 13.11.2021.

Peterson, J. 2020. Software Development Life Cycle: Finding a Model That Works. Luettavissa: https://www.whitesourcesoftware.com/resources/blog/sdlc-software-development-life-cycle/. Luettu: 7.2.2022

R. Edward Freeman, Alexander Moutchnik, 2013. Stakeholder management and CSR: questions and answers. In: Umwelt Wirtschafts Forum, Springer Verlag, Vol. 21, Nr. 1

Radigan. D. Agile Roadmaps: build, share, use, evolve. Luettavissa: https://www.atlassian.com/agile/product-management/roadmaps. Luettu: 18.3.2022.

Reutersen, A. 2013. The First Ninety Years. Intrum Justitia.

Salminen, E. 2018. Palvelumuotoilun hyödyntäminen yrityksen toiminnan kehittämisessä. Opinnäytetyö. Laurea amk. Luettavissa: https://urn.fi/URN:NBN:fi:amk-2018052410130. Luettu: 10.12.2021

Scrum. What is scrum? Luettavissa: https://www.scrum.org/resources/what-is-scrum. Luettu: 18.2.2022.

Sharma, L 2021. WaterFall Model. Luettavissa: https://www.toolsqa.com/software-testing/waterfall-model/. Luettu 9.1.2022

Tateeda. 2021. The Scrum Cycle in Agile Software Development. Luettavissa: https://tateeda.com/blog/the-scrum-cycle-in-agile-software-development. Luettu: 18.2.2022.

Techopedia, 2020. Software development life cycle. Luettavissa: https://www.techopedia.com/definition/22193/software-development-life-cycle-sdlc. Luettu 15.1.2022.

Tutorialspoint. SDLC – Waterfall Model. Luettavissa: https://www.tutorialspoint.com/sdlc/sdlc_waterfall_model.htm. Luettu: 15.1.21

Valli, S. 2018. Ikkunoita tutkimusmetodeihin 1. Metodin valinta ja aineistonkeruu: virikkeitä aloittelevalle tutkijalle. PS-Kustannus

VAMK verkkojulkaisu. Saari, A-K 2021. Unohtakaa vesiputousmalli – ketterä kehitys on tullut jäädäkseen!
Luettavissa: http://urn.fi/URN:NBN:fi-fe2021101451078. Luettu 9.1.2022

Van Der Hoek, J. 2021. How to Structure an Agile Development Scrum Team. Luettavissa: https://www.mendix.com/blog/the-road-to-adopting-scrum-team-composition/. Luettu: 18.2.2022.

W3Schools. SDLC Agile Model. Luettavissa: https://www.w3schools.in/sdlc-tutorial/agile-model/. Luettu 18.2.2022.

Zinios Edge, 2020. Top 5 Software Development Life Cycle (SDLC) Methodologies. Luettavissa: https://ziniosedge.com/top-5-software-development-life-cycle-sdlc-methodologies/. Luettu: 15.1.2022.

W3schools. Agile model. Luettavissa: https://www.w3schools.in/sdlc-tutorial/agile-model/. Luettu: 18.2.2022.

Liitteet

Liite 1. Puolistrukturoitu haastattelu liittyen kehitysprosesseihin

1. Millainen on kehitysprosessi osastolla, jolla työskentelet?
2. Mikä on oma tausta kehitysprosessien suunnittelussa? (Koulutus, pätevyydet, kokemus)
3. Mitä viitekehystä noudatetaan kehitysprosessien taustalla? Miksi?
4. Ketkä ovat mukana kehitysprosessin suunnittelussa?
    a. Miltä osastoilta kehitysprosesseihin osallistuvat henkilöt tulevat?
5. Millainen on kehitysprosessien hierarkia?
    a. Onko kehitysprosesseihin osallistuvien henkilöiden osalta tapahtunut muutosta? Jos on, niin milloin, miksi?
6. Ovatko nykyiset kehitysprosessit toimineet halutulla tavalla?
    a. Mitä kehitettävää niissä voisi olla?
    b. Entä mitkä ovat niiden vahvuudet?
7. Millaisia toivoisit kehitysprosessien jatkossa olevan?
8. Tuleeko mieleen muuta haastatteluun relevanttia asiaa, joita kysymyksissä ei vielä käyty läpi?

Liite 2. Parannusehdotukset kehitysprosessiin

Johdanto

Kappaleessa 6.1 esitettyjen ongelmakohtien korjaamisella voi olla merkittävä vaikutus kehitysprosessin tehokkuuteen ja yleisesti prosessin toimivuuteen. Nykyinen kehitysprosessimalli aiheuttaa resurssien haaskausta joka viikko. Tästä kertyy pidemmän aikavälin sisällä suuri määrä hukkaan valuneita resursseja, jotka olisi voinut käyttää johonkin järkevämpään tekemiseen. Seuraavaksi esitän kehitysprosessin nykymallin ongelmakohdat ja niiden parannusehdotukset omiksi otsikoiksi jaoteltuna.

Päivittäispalaverin kesto

Nykyisellään päivittäiset palaverit kestävät 30 minuuttia ja niissä ei ole varsinaista asialistaa, jonka mukaan mennään, vaan usein asiat nousevat esille spontaanisti. Harvemmin koko palaverin keston saa hyödynnettyä järkevästi. Välillä palaverit menevät yliajalle, välillä ne lopetetaan hieman aikaisemmin. Selvää kuitenkin on, että strukturoimattoman rakenteen takia aikaa menee hukkaan.

Ehdotettu ratkaisu 1.

Scrum-metodiin kuuluvat päivittäiset palaverit ovat kestoltaan korkeintaan 15 minuuttia ja niissä käydään läpi nykytilanne ja seuraavan 24 tunnin aikana tapahtuva päivittäinen työ. Tämä agenda antaa itsessään jo struktuuria sekä agendaa koko palaverille ja pitää palaverin sidosryhmät ajan tasalla nykytilanteesta. Vaikka palaverissa ei olisi paljoa asiaa läpikäytävänä, ei aikaa menisi yhtä paljon hukkaan kuin 30 minuuttia kestävissä palavereissa. Lyhyempi aikataulu mielestäni vähentää ’turhaa’ asiaa ja ohjaa osallistujien tekemistä. Esitetyt asiat todennäköisesti mietitään huolellisemmin, että ne ehditään esittämään ja palaverin struktuuri pysyy kasassa.

Ehdotettu ratkaisu 2.

Koska päivittäisen palaverin aiheet eivät aina koske kaikkia osallistujia, niin järkevää olisi, että kaikki eivät olisi paikalla joka kerta. Tällä pystyttäisiin välttämään tehokkaan työajan haaskaus. Etukäteen voisi sopia esimerkiksi, että kaikki osallistuisivat päivittäiseen palaveriin kaksi kertaa viikossa ja vain osa henkilöistä osallistuisi kolmena päivänä päivittäispalavereihin. Mikäli palavereissa, johon osallistuu vain osa henkilöistä, sattuu nousemaan esille sellaiset asiat, jotka koskettavat koko tiimiä, niin ne henkilöt, jotka ovat palavereihin

osallistuneet voivat välittää kaikkia koskettavan tiedon eteenpäin. Tällä mallilla voidaan varmistaa tehokas resurssien käyttö.

Nämä ratkaisut eivät ole missään nimessä toisiaan pois sulkevia, vaan kummatkin vaihtoehdot voidaan yhdistää yhdeksi kokonaisuudeksi tai molemmista ratkaisusta voidaan yhdistää asioita, muodostaen toimivan kokonaisuuden. Ratkaisut olivat helpompia esittää erillisinä vaihtoehtoina, sillä ensimmäinen kuvastaa Scrum-metodin mukaista lähestymistapaa, toinen omaa, tutkimustulosten avulla esiin ilmennyttä ratkaisua.

Päivittäisissä palavereissa esille nostettujen asioiden priorisointi

Päivittäisissä palavereissa esille nostetut, asiat ovat välillä nousseet prioriteettien osalta korkeammalle kuin tärkeämmät, ennalta suunnitellut kehitystyöt. On syynsä sille, miksi asioita ei hyvän käytännön mukaisesti kuuluisi nostaa spontaanisti muiden kehitystöiden edelle – se sekoittaa prioriteetit, käytettävät kehitystyötunnit ja koko kehitysprosessin toimivuuden. Tämä voi osaltaan johtaa huonoihin, riittämättömiin spekseihin. Päivittäisissä palavereissa nostetut asiat ovat tärkeitä ja lisäävät läpinäkyvyyttä työntekijöiden välillä, mutta kehitysprosessin mukaista käytäntöä tulee silti noudattaa.

Kun esimerkiksi nykyisten ongelmien korjaukseen liittyviä kehitystöitä nostetaan esille palavereissa, tärkeää on, etteivät ne saa korkeinta mahdollista prioriteettia, vaan niistä tulisi luoda tiketti ja tikettiin tulisi arvioida asian prioriteetti. Tiketin kirjoittaja ehtii näin ollen ajatella asiaa kahdesti ja arvioida koko asian uudelleen.

Kun asioista luodaan tiketti, joissa kirjoitetaan asian kuvaus ja myöhemmin mahdollinen ratkaisu, jää siitä dokumentaatio myöhempien tilanteiden varalle. Mikäli asioita tehdään ilman virallista prosessia ja tikettejä, ei aiheesta synny dokumentaatiota tai se on huomattavasti hankalampaa jatkossa löytää. Etenemissuunnitelman prioriteetit pysyvät kehitysprosessin mukaista käytäntö noudatettaessa ennallaan ja kehitysprosessi toimii.

Tuotekehitysjonon hallinta

Yksi ongelma tuotekehitysjonon hallinnassa on se, että uudet kehitysideat jäävät liian herkästi tuotekehitysjonoon tikettinä ’roikkumaan’ ja tämä osaltaan paisuttaa tuotekehitysjonoa liian suureksi. Kun liian monta tikettiä löytyy tuotekehitysjonosta, tekee se kokonaisuuden hallinnan haastavaksi ja paisuttaa kehitysjonoa liian suureksi.

Toinen ongelma liittyy rooleihin tai tarkemmin yhden roolin puutteellisuuteen. Tällä hetkellä tuoteomistajan (Product Owner) roolia hoitaa liiketoimintaosasto. Nykyisellään roolin tekee haastavaksi se, että siihen liittyvien tehtävien hoitamiseen ei jää tarpeeksi aikaa, sillä liiketoimintaosaston päivittäiset tehtävät vievät suurimman osan resursseista ja vain vähän aikaa jää lopulta tuoteomistajan työhön.

Ratkaisuehdotus:
Tuotekehitysjonon hallinnan osalta ehdotan, että ideaprosessin synnyttämien kehitystyötikettien suunnittelua vietäisiin pidemmälle ja tehtäisiin matalammalla kynnyksiä päätöksiä siitä, viedäänkö jokin idea eteenpäin vai ei. Tärkeää on myös ideaa eteenpäin vietäessä päättää, että mille etenemissuunnitelman syklille kehitystöitä viedään.

Mikäli huomataan jokin sellainen idea, joka on jäänyt ’roikkumaan’ kehitysjonoon pidemmäksi aikaa, niin tulisi tällaisten ideoiden osalta pitää katselmuksia vaikkapa kerran syklin aikana, jossa idea poistetaan kokonaan tuotekehitysjonosta tai suunnitellaan niiden seuraavia vaiheita aikataulullisesti. Katselmuksissa voisi olla koko IT- ja liiketoimintaosasto paikalla.

Tuoteomistajan roolin suhteen ehdotan, että yksi henkilö IT-osastolta nimitetään tuoteomistajan rooliin, joka vähentäisi liiketoimintaosaston työmäärää tämän suhteen. Liiketoimintaosasto suorittaisi tuoteomistajan tehtäviä jatkossakin, mutta työmäärä jakaantuisi tasaisemmin liiketoiminta- ja IT-osaston välillä.

Julkaisu

Ensimmäinen ongelma julkaisun osalta on se, että syklin loppuvaiheessa on ounasteltu, riittääkö sovelluskehittäjillä töitä, kun olemassa olevat kehitystyöt julkaisun osalta on saatu tehtyä. Missään nimessä ei ole optimaalista, että saatavilla olevia resursseja ei käytetä hyödyllisesti. Toki sovelluskehittäjät ovat itseohjautuvia ja keksivät tekemistä, mikäli heillä näyttää olevan kehitystyöt syklin osalta tehty. Tämä on silti ongelma, sillä sovelluskehittäjät eivät itseohjautuvasti keskity tällöin niihin töihin, jotka vaativat liiketoimintaosaston näkökulmasta eniten huomiota. Tässä tapauksessa vastuu sovelluskehittäjien tehokkaasta resurssien allokoinnista jää liiketoiminta- ja it-osaston harteille

Toinen ongelma on, että aina ei ole ollut tiedossa seuraavan julkaisun aikatauluja, joka on johtanut epätietoisuuteen ja hämmennykseen tehtävien töiden osalta. Epäselväksi on jäänyt milloin seuraavan syklin kehitystyö pitää aloittaa, milloin jäädytetään julkaisun laajuus (’scope freeze’) ja milloin tuleva julkaisu pitäisi viedä tuotantoon. Aikataulut on lopulta kyllä ilmoitettu, mutta reilusti myöhässä.

Kolmas ongelma on syklin jälkeisen retrospektiivipalaverin puuttuminen, jossa käydään läpi syklin aikana tehdyt työt, onnistumiset ja epäonnistumiset. Tarkoitus on reflektoida omaa ja kehittää toimintaansa seuraavia syklejä varten.

Ehdotettu ratkaisu: Hyvällä suunnittelulla saadaan ratkaistua kaikki kolme ongelmaa. Syklin päättyessä tulisi olla jo suunniteltuna seuraavan etenemissuunnitelman kehityssyklin mukaiset kehitystyöt, jolloin sovelluskehittäjät siirtyvät tekemään niitä, silloin kun aikaisemman syklin tuotekehitysjonossa olleet kehitystyöt ja sprintit ovat saatu valmiiksi.

Mielestäni pitäisi olla aina tiedossa vähintään kahden julkaisun aikataulut – nykyisen ja tulevan. Näin pystytään varmistumaan siitä, että aina on olemassa aikataulu, joka ohjaa niin sovelluskehittäjien, IT-osaston ja liiketoimintaosaston toimintaa, joka puolestaan pitää kokonaisuuden toimivana.

Retrospektiivejä tulisi jatkossa pitää jokaisen syklin päättyessä IT-osaston ja sovelluskehittäjien kesken. IT-osasto voisi raportoida retrospektiivin aikana nostetuista asioista liiketoimintaosastolle erikseen. Retrospektiiveissä tarkastellaan miten sprintti (tässä tapauksessa sykli) sujui, missä onnistuttiin ja missä voitaisiin parantaa. Keskustelua käydään myös siitä, että mitä sitoudutaan parantamaan seuraavan syklin aikana. Retrospektiivin

avulla saadaan hyödyllistä keskustelua aikaiseksi ja kehitettyä tiimiin kuuluvien yksilöiden toimintaa paremmaksi seuraavia sprinttejä varten.

Kehitysprosessin noudattaminen

Kappaleen 5.1 kehitysprosessikaaviossa (Kuva 6) on esitetty liiketoimintaosaston nykyisen kehitysprosessin ’iso kuva’. Siihen liittyy useita vaiheita ja palavereja eri sidosryhmien kanssa. Viime aikoina olen kuitenkin huomannut, että esimerkiksi Roadmap-palaveria (etenemissuunnitelmapalaveri) ei ole pidetty ajallaan, vaan se on pidetty vasta syklin päätyttyä, myöhässä, vaikka todellisuudessa se on ollut tarkoitus pitää syklin lopussa.

Syklin lopussa pidettävässä Roadmap-palaverissa mietitään seuraavalle syklille lisättäviä kehitystöitä tuotekehitysjonosta. Mikäli kyseinen palaveri pidetään vasta seuraavan syklin alettua, voi se aiheuttaa aikataulu- ja resurssihaasteita. Pahimmillaan se johtaa kehitystyön myöhäiseen aloitukseen, joka osaltaan voi johtaa kiireeseen, prioriteettien sekoittamiseen, riittämättömiin spekseihin ja lopulta huonoon, ei-toivottuun lopputulokseen kehitystyön osalta.

Liite 3. Kuva liiketoimintaosaston kehitysprosessista