Toimintamalli asiakkuudenhallintajärjestelmän kehittämiselle vuorovaikutuksessa loppukäyttäjien kanssa

Perttu Haavisto

Haaga-Helia ammattikorkeakoulu
Tradenomi (YAMK)
Liiketoiminnan uudistamisen ja johtamisen koulutus
Digitaalisen liiketoiminnan mahdollisuudet
Opinnäytetyö
2023

Tiivistelmä

Tekijä
Perttu Haavisto

Tutkinto
Tradenomi (YAMK)

Raportin/Opinnäytetyön nimi
Toimintamalli asiakkuudenhallintajärjestelmän kehittämiselle vuorovaikutuksessa loppukäyttäjien kanssa

Sivu- ja liitesivumäärä
72

Tässä opinnäytetyössä esitellään toimintamalli asiakkuudenhallintajärjestelmän kehittämistyön hallinnointiin vuorovaikutuksessa loppukäyttäjien kanssa. Kohdeorganisaatiossa oli otettu uusi asiakkuudenhallintarjärjestelmä käyttöön, kun opinnäytetyön teko aloitettiin. Opinnäytetyön tavoitteena oli selvittää, miten uuden asiakkuudenhallintajärjestelmän kehitystyötä hallinnoidaan, miten siitä viestitään, ja miten loppukäyttäjiä osallistetaan kehitystyöhön.

Lähtötilanne taltioitiin kyselytutkimuksella, jossa selvitettiin loppukäyttäjien tyytyväisyyttä uuden asiakkuudenhallintajärjestelmän käyttöönoton aikaiseen pääkäyttäjäryhmän toimintaan ja prosesseihin. Tämän jälkeen parempia toimintatapoja etsitään ketterän kehityksen malleista: Scrum, Kanban ja XP sekä palvelumuotoilun ja viestinnän kirjallisuudesta.

Toteutus jakautuu kahteen pääteemaan, joista toinen palvelee itse kehitystyötä hallinnoivaa pääkäyttäjäryhmää ja keskittyy ketterän kehityksen menetelmiin ja kehitystyön hallinnointiin. Ketterän kehityksen mallien pohjalta esitellään kohdeorganisaation skaalaan, työkaluihin ja rooleihin soveltuva kehitystyön malli. Toteutuksena esitetty malli ei ole suoraan kopio mistään viitekehyksen malleista, vaan mukautettu malli, joka hyödyntää osia kaikista viitekehyksen malleista, eli Scrumista, Kanbanista ja XP:stä.

Toinen osio liittyy loppukäyttäjien osallistamiseen. Tämä osio sisältää palautteen antamisen, vuorovaikutuksen, viestinnän ja viestintäkanavat. Loppukäyttäjien osallistamisen mallin pohjana toimi edellä mainitut ketterän kehityksen menetelmät, palvelumuotoilu, ja viestinnän teoria.

Työn alkuvaiheilla tehty kyselytutkimus toistettiin, kun opinnäytetyössä esitellyt mallit olivat olleet käytössä hieman yli kolme kuukautta, tarkoituksena arvioida työn hyödyllisyyttä.

Asiasanat
Kehitystyö, asiakkudenhallintajärjestelmä, ketterä ohjelmistokehitys, palvelumuotoilu, backlog

Sisällys

1 Johdanto ... 1
1.1 Työn taustaa ... 1
1.2 Työn tavoite ja rajaukset ... 2
1.3 Käsitteet ... 3
1.4 Tietoa yrityksestä ... 4
2 Lähtötilanne ... 5
2.1 Lähtötilanteen kuvaus ... 5
2.2 Kyselytutkimukset ... 6
3 Prosessinhallinta ja ketterä kehitys ... 10
3.1 Ketterä kehitys ... 11
3.1.1 Ketterän kehityksen julistus ... 12
3.1.2 Ketterän kehityksen konsensus ... 13
3.1.3 Vesiputousmalli, ketterän kehityksen edeltäjä ... 14
3.1.4 Ketterän kehityksen ideaputki ... 15
3.2 Extreme Programming (XP) ... 17
3.2.1 Lähtöhaasteet ja ratkaisut ... 17
3.2.2 Muutoksen hinta ... 18
3.2.3 Arvot ... 20
3.2.4 Periaatteet ... 21
3.2.5 Kehitystyön runko ... 22
3.2.6 Tapahtumat ... 23
3.2.7 Tekniikoita käytäntöön ... 23
3.2.8 Kehitystyön strategia ... 25
3.2.9 Roolit ... 25
3.3 Kanban ... 27
3.3.1 Visualisointi ... 27
3.3.2 Jonot ... 29
3.3.3 Tehtäväkortit ... 29
3.3.4 Työstettävät tehtävät ... 31
3.3.5 Työstettävien tehtävien rajoittaminen ... 34
3.3.6 Työn virtaus ... 34
3.3.7 Työn virtauksen kehittäminen ... 35
3.4 Scrum ... 37
3.4.1 Läpinäkyvyys, tarkastelu ja adaptoituminen ... 37
3.4.2 Arvot ... 37

3.4.3 Roolit ... 38
3.4.4 Sprintit ja tapahtumat ... 39
3.4.5 Tavoitteet ... 40
3.4.6 Tuotteen Backlog ... 40
3.4.7 Sprintin Backlog ... 41
3.4.8 Inkrementti ja valmiin määritelmä ... 41
3.5 Vertailu XP, Scrum, Kanban ... 42
3.5.1 Roolit ... 43
3.5.2 Tapahtumat ... 43
3.5.3 Backlog ... 43
3.5.4 Keskeiset mittarit ... 44
3.5.5 Palaute ... 45
4 Palvelumuotoilu, viestintä, ja loppukäyttäjien osallistaminen ... 46
4.1 Palvelumuotoilu ... 46
4.1.1 Iteratiivinen ja inkrementaalinen ... 47
4.1.2 Divergenssi ja konvergenssi ... 47
4.1.3 Visualisointi ... 48
4.1.4 Palvelumuotoiluprosessi ... 48
4.1.5 Määrälliset ja laadulliset tutkimukset ... 49
4.2 Palvelumuotoilun menetelmiä ... 49
4.2.1 Tiedonkeruun menetelmiä ... 50
4.2.2 Prototyypit ja pilotointi ... 51
4.3 Viestintä ... 52
4.3.1 Tiedottaminen ... 52
4.3.2 Palaute ... 53
4.3.3 Oikeat viestintäkanavat ... 53
5 Mallien esittely ... 54
5.1 Projektinhallinta malli kehitystiimille ... 54
5.1.1 Työkalut ... 54
5.1.2 Roolit ... 55
5.1.3 Backlog ... 56
5.1.4 Tapahtumat ... 58
5.1.5 Backlog tehtäväkorttien käsittely ... 60
5.2 Loppukäyttäjien osallistaminen ... 61
5.2.1 Tiedottaminen ... 62
5.2.2 Palaute ja ideointi ... 62

5.2.3 Palautteen laadun mahdollistaminen 63
6 Pohdinta 65
6.1 Kyselytutkimus 2 65
6.2 Oma analyysi 69
6.3 Jatkokehitys 70
Lähteet 71

1 Johdanto

Tässä opinnäytetyössä esitellään toimintamalli asiakkuudenhallintajärjestelmän jatkuvaan kehittämiseen ja loppukäyttäjien osallistamiseen.

Asiakkuudenhallintajärjestelmä on ohjelmisto, jonka tarkoitus on tarjota työntekijöille käyttäjäystävällinen työkalu yrityksen datan tarkasteluun, käsittelyyn, hyödyntämiseen ja rikastamiseen. Asiakkuudenhallintajärjestelmän on tarkoitus säilöä dataa yrityksen työntekijöiden tehtävistä, tarvittavista yhteystiedoista, asiakastiedoista ja vuorovaikutuksesta asiakkaisiin ja kumppaneihin. Nämä datat esitetään järjestelmässä visuaalisesti, tarjoten selkeän näkymän asiakkaan tiedoista ja työntekijöiden aktiviteeteista ja tehtävistä. Datan on hyvä kertyä mahdollisimman strukturoidusti, tarjoten kattavat mahdollisuudet analytiikan hyödyntämiseen. (Specchia 2022, luku 1.2.)

Loppukäyttäjiä asiakkuudenhallintajärjestelmän osalta ovat pääsääntöisesti yrityksen työntekijät. Asiakkuudehallintajärjestelmä on ensisijaisesti työntekijöiden työkalu laadukkaan asiakastyön mahdollistamiseksi ja siitä syystä asiakkuudenhallintajärjestelmän kehitystyö tulee tehdä vuorovaikutuksessa loppukäyttäjien kanssa.

1.1 Työn taustaa

Kohdeorganisaatiossa on otettu käyttöön uusi asiakkuudenhallintajärjestelmä, eli CRM-järjestelmä. Asiakkuudenhallintajärjestelmä on räätälöity kohdeorganisaation käyttötarpeisiin sisäisen administraattoriryhmän sekä avuksi palkatun konsulttiyrityksen toimesta. Uusi asiakkuudenhallintajärjestelmä on merkittävästi aikaisempaa ratkaisua kattavampi ja kokonaisvaltaisempi järjestelmäkokonaisuus. Se korvasi vanhan asiakkuudenhallintajärjestelmän lisäksi monia muita käytössä olleita toteutuksia ja palveluita sekä tarjoaa täysin uusia mahdollisuuksia.

Ensimmäistä tuotantoversiota rakentaessa käytettiin konsulttiyrityksen toimintamalleja ja projektinhallinta toteutettiin heidän ohjaamana. Tuotantoon siirtymisen jälkeen ylläpitovastuu siirtyi suurilta osin pääkäyttäjäryhmälle, vaikka konsulttiyritys jatkoi vielä pääkäyttäjäryhmän apuna kehitystyön tekemistä. Tuotantokäytön vakiinnuttua pääkäyttäjäryhmän on tarkoitus ottaa enemmän vastuuta myös kehitystyöstä.

Aikaisemman asiakkuudenhallintajärjestelmän käyttöä ei aktiivisesti kehitetty ja järjestelmän käyttö ei ollut kovin laaja-alaista. Vanha järjestelmä ei ollut yhtä helposti räätälöitävissä ja näin ollen haasteellisempia liiketoimintatarpeita ratkaistiin järjestelmän ulkopuolella, esimerkiksi rikastamalla raportointia Google Sheets – taulukoilla tai ratkaisemalla raportointitarpeita vapailla tekstikentillä tai käyttämällä kenttiä luovasti oikeasta käyttötarkoituksesta poiketen. Tällaisten ratkaisujen ylläpitäminen on haasteellista ja työlästä, eivätkä ratkaisut usein skaalaudu tulevaisuuden tarpeisiin.

Uusi asiakkuudenhallintajärjestelmä on hyvin laaja, skaalautuva ja räätälöitävä, joten aikaisemmasta poiketen jatkuvalle ja aktiiviselle kehittämiselle on mahdollisuus. Koska aikaisemmin asiakkuudenhallintajärjestelmän kehitystyötä ei aktiivisesti tehty, ei asiakkuudenhallintajärjestelmän ylläpitämisestä ja kehittämisestä vastaavalla pääkäyttäjäryhmällä ole entuudestaan työkaluja tai prosesseja kehitystyön hallinnointiin tai siitä viestimiseen.

Kattavien toiminnallisuuksien ja kokonaisvaltaisuutensa vuoksi uusi asiakkuudenhallintajärjestelmä vaatii myös käyttäjiltään enemmän aikaa ja korkeampaa taitotasoa. Vanhassa asiakkuudenhallintajärjestelmässä loppukäyttäjälle riitti, kun oppi perehdytyksessä perusasiat. Uusi asiakkuudenhallintajärjestelmä tulee kehittymään jatkuvasti ja vaatii laajemman perehdytyksen lisäksi jatkuvaa uuden oppimista.

1.2 Työn tavoite ja rajaukset

Työn tavoitteena on luoda toimintamalli asiakkuudenhallintajärjestelmän kehittämiseen vuorovaikutuksessa loppukäyttäjien kanssa. Tutkimuskysymyksiä on kolme:

- Miten hallinnoida asiakkuudenhallintajärjestelmän jatkuvaa kehitystä?
- Miten viestiä kehitystyöstä loppukäyttäjille?
- Miten osallistaa loppukäyttäjät kehitystyöhön?

Teorian osalta työ jakautuu kahteen kokonaisuuteen: kehitystyön prosessihallintaan ja loppukäyttäjien osallistamiseen. Työn tarkoitus on, että nämä kaksi osa-aluetta nivoutuisivat mahdollisimman tiiviisti osaksi samaa prosessia, sen sijaan, että kehitystyö tehtäisiin ilman loppukäyttäjien osallistumista ja viestintä koskisi vasta valmiita tuotoksia.

Prosessinhallinnan osalta tavoitteena on luoda malli siihen, miten kehittämistehtäviä kerätään, jaotellaan, priorisoidaan, delegoidaan, toteutetaan ja visualisoidaan. Pääkäyttäjäryhmän työnjakoa ja rooleja tarkastellaan ja tarvittaessa muutetaan.

Loppukäyttäjien osallistamiseen on tavoite luoda yhtenäiset, selkeät, ja käyttäjäystävälliset toimintamallit ja viestintäkanavat. Tavoitteena on selvittää, miten loppukäyttäjille viestitään ja miten heiltä kerätään palautetta, kehitysehdotuksia, ja ideoita, kuormittamatta heitä liikaa.

Lisäksi mallin tulee olla läpinäkyvä, jotta sidosryhmillä on näkyvyys tehtävään kehitystyöhön, tulevaisuuden suunnitelmiin ja antamiensa palautteiden ja ideoiden käsittelyyn.

Asiakkuudenhallintajärjestelmän käytön perehdyttämiseen tai kouluttamiseen sidosryhmille ei oteta kantaa tässä työssä. Lisäksi kehitystyön julkaisuun liittyvä versionhallinta ja julkaisujen tekniset vaatimukset rajataan pois.

1.3 Käsitteet

Tässä kappaleessa esitellään tässä opinnäytetyössä esiintyviä käsitteitä.

Agile, eli ketterä kehitys	Ohjelmistokehityksessä suosittu ideologia, joka toimii monien mallien perustana. Ketterä kehitys perustuu käyttäjälähtöisyyteen, yhteistyöhön, oppimiseen ja reaktiivisuuteen.
Asiakkuudenhallintajärjestelmä	Loppukäyttäjälähtöinen järjestelmä asiakastiedon tarkasteluun, hallinnointiin, ylläpitoon ja käsittelyyn.
Backlog	Työ- ja tehtäväjono. Fyysinen tai digitaalinen paikka, johon kehitystiimin työt kerätään tehtäväkorteiksi.
Inkrementaalinen	Vähitellen kasvava. Inkrementaalinen kehitys tarkoittaa sitä, että pilkotaan isoja tavoitteita pienempiin osiin, ja toteutetaan yksi osa kerrallaan.
Iteratiivinen	Toistuva. Iteratiivinen kehitystyö viittaa lyhyen aikavälin toistoihin, eli ei luoda yhtä suurta julkaisua vain toistuvasti pienempiä.
Low-code ja No-code ohjelmistot	Ohjelmistoja, jotka tarjoavat perinteisten ohjelmistokielen sijaan käyttäjäystävällisempiä visuaalisia työkaluja koodin luontiin.
Roadmap	Pitkän aikavälin kehityssuunnitelma. Suurpiirteisempi kuin työjono (backlog).
Sandbox	Hiekkalaatikkoversio ohjelmistosta, eli tässä kontekstissa versiokopio asiakkuudenhallintajärjestelmän tuotantoympäristöstä. Kehitystyö tehdään aina Sandboxin puolella, jotta ei vahingossa häiritä tuotantoympäristön toimintaa.

Scrum	Scrum määritelmän mukaisesti, Scrum on kevyt viitekehys kehitystyön prosessinhallintaan.
Slack	Yrityskäyttöön suunniteltu pikaviestintäsovellus sisältäen sekä kanavat, että yksityisviestit.
Sprint	Yksittäinen kehitystyön iteraatio, usein 1–4 viikkoa pitkä.
Trello	Verkkopohjainen luetteloinnisovellus, jolla voi luoda muun muassa Kanban-mallisia työjonoja
User story	Käyttäjätarina, eli tapa strukturoida kehitystoive loppukäyttäjälähtöisestä näkökulmasta
Vesiputousmalli	1990- ja 2000-luvuilla suosittu ohjelmistokehityksen malli, joka perustuu siihen, että ohjelmisto rakennetaan valmiiksi tarkan tarvemäärittelyn pohjalta, ja toimitetaan valmiina kokonaisuutena.

1.4 Tietoa yrityksestä

Kohdeorganisaatio on ohjelmistotalo, joka työllistää noin 330 henkilöä ja liikevaihto on hieman yli 100 miljoonaa euroa.

Yrityksen tarjoamat ohjelmistot ovat nykyaikaisia pilvipalveluita, ja organisaatiokulttuuri on moderni ja nuorekas. Myynnin funktio, johon tämän työn tuotos pääasiallisesti vaikuttaa, työllistää noin 50 henkilöä.

2 Lähtötilanne

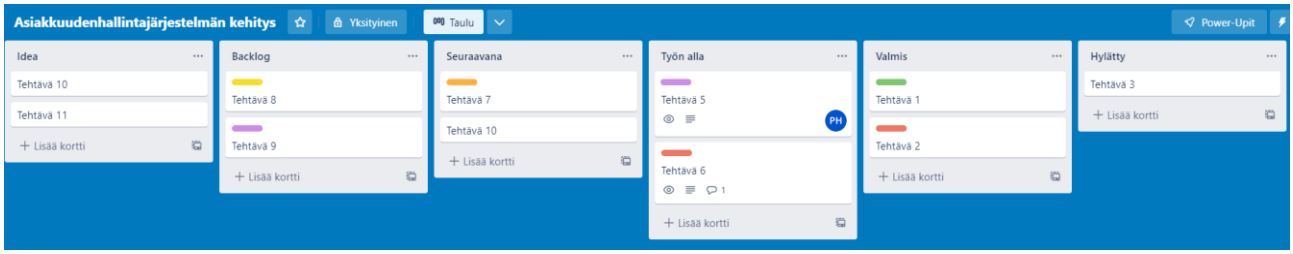
Tässä kappaleessa käsitellään tilannetta missä kohdeorganisaatiossa on oltu, kun tämä työ on aloitettu. Ensimmäisessä kappaleessa kuvataan lähtötilannetta ja käytössä olleita prosesseja. Toisessa kappaleessa tarkastellaan aloitushetken loppukäyttäjäkokemusta kyselytutkimuksen tulosten pohjalta. Kappaleessa kuusi käsitellään käyttöönoton jälkeistä tilannetta ja käydään uudelleen toteutetun - mutta saman sisältöisen - kyselytutkimuksen tuloksia läpi siltä ajankohdalta.

2.1 Lähtötilanteen kuvaus

Asiakkuudenhallintajärjestelmän käyttöönoton yhteydessä kaikille käyttäjille perustettiin yhteinen Slack-kanava käyttötukea ja toiveiden esittämistä varten. Tämän lisäksi loppukäyttäjät ovat hyödyntäneet omien tiimiensä kanavia sekä laittaneet yksityisviestejä pääkäyttäjille, niin toivomusten, kuin käyttötuen suhteen. Ideoiden ja toiveiden esittämiseen ei ole prosessia ja näin ollen palaute kertyy useisiin kanaviin, mistä pääkäyttäjät pyrkivät niitä keräämään.

Backlogia hallinnoidaan Trellossa, joka on joustava, kollaboratiivinen ja visuaalinen työnhallintatyökalu, kuten kuvasta 1 voi havaita (kuva 1). Kaikki järjestelmäpääkäyttäjät, joita on viisi, ylläpitävät yhdessä Trello-kanavaa. Pääkäyttäjät lisäävät uusia tehtäviä kanavalle sitä myötä, kun niitä ilmenee, joko loppukäyttäjien toiveista tai pääkäyttäjien omien havaintojen pohjalta. Tehtävien kirjaamisessa ei ole määriteltyä rakennetta ja niiden kuvaukset kirjataan vaihtelevalla tarkkuustasolla. Toisinaan tehtävälle on annettu yksityiskohtainen kuvaus ja sidosryhmät on mainittu, kun taas toisinaan tehtävä voi olla pelkkä otsikko.

Trellon työjonossa (kuva 1), on nähtävissä vaiheet ideat, backlog, työn alla, valmis ja hylätty. Nämä kuvastavat tehtävien vaiheistusta, jossa uudet ideat lisätään ensimmäiseen kolumniin, ja ne etenevät sieltä vaiheistuksen mukaan vasemmalta oikealle, kunnes ne ovat valmiita.



Kuva 1. Trello backlog

Trellon työjonoa on tarkoitus päivittää reaaliaikaisesti. Tehtävien valinta, priorisointi ja tekijöiden määrittäminen tehdään viikoittain yhteisessä palaverissa. Samaisessa palaverissa käydään myös läpi, onko viime viikolla sovitut tehtävät saatu tehtyä vai jäikö jotain seuraavalle viikolle. Sidosryhmillä, eli järjestelmän muilla käyttäjillä, ei ole näkyvyyttä tähän työjonoon.

Tähän asti muutokset on tehty suurilta osin suoraan asiakkuudenhallintajärjestelmän tuotantoympäristöön, joten muutos on ollut jatkuvaa. Tiedottaminen muutoksista ei ole ollut systemaattista, vaan satunnaista ja struktuuriltaan epäsäännönmukaista.

Pitkän aikavälin tavoitteisiin, eli roadmapin suunnitteluun on alustavasti sovittu kuukausittain toistuvat palaverit pääkäyttäjäryhmän kanssa. Roadmappia ei ole vielä jaettu sidosryhmille.

2.2 Kyselytutkimukset

Laadullista palautetta pääkäyttäjäryhmä saa arjessa kattavasti, ja sitä käsitellään aktiivisesti. Lähtö- ja lopputilanteiden vertailun helpottamiseksi, käytetään loppukäyttäjille kohdistetussa kyselytutkimuksessa puolestaan määrällisiä menetelmiä. Vaihtoehtona oli laadulliset haastattelut tai etnografiset tutkimukset (Tuulaniemi 2011,62–63.). Asiakkuudenhallintajärjestelmän käyttöönottoaika oli työllistävää ja kiireistä aikaa, niin pääkäyttäjille, kuin loppukäyttäjille. Loppukäyttäjät joutuivat käyttämään työaikaansa uuden opettelemiseen, joten kyselytutkimuksesta haluttiin tehdä mahdollisimman kevyt ja kustannustehokas, samalla kuitenkin pyrkien laajaan kokonaiskuvaan yksittäisten mielipiteiden sijaan. Tästä syystä kyselytutkimus valikoitui lähtötilanteen kartoittamiseen sopivimmaksi lähestymistavaksi.

Ensimmäisellä kyselytutkimuksella kartoitettiin loppukäyttäjien tyytyväisyyttä lähtötilanteeseen, eli asiakkuudenhallintajärjestelmän käyttöönoton aikaiseen toimintaan, prosesseihin ja vuorovaikutukseen. Kartoituksessa teemoja on kaksi. Ensimmäisenä se, miten hyvä kuva loppukäyttäjillä oli siitä, mitä järjestelmäkehityksessä tehtiin, eli kehitystyön läpinäkyvyys. Toisena teemana oli loppukäyttäjien kokemus siitä, kuinka hyvin heidät ja heidän palautteensa huomioitiin.

Sama kysely tehtiin uudelleen, kun tässä opinnäytetyössä luodut toimintamallit olivat otettu käyttöön. Tarkoituksena oli arvioida opinnäytetyön onnistumista ja hyötyä, ja tunnistaa mahdollisia jatkokehityskohteita. Toisen kyselytutkimuksen tulokset esitellään opinnäytetyön lopussa kappaleessa kuusi.

Ensimmäisen kyselytutkimuksen tulokset:

Asteikon kuvaus: Vastaukset asteikolla 1-5
1 = täysin eri mieltä
2 = eri mieltä
3 = ei samaa, eikä eri mieltä
4 = samaa mieltä
5 = täysin samaa mieltä

Ensimmäisen kyselyn tulokset:

- Minulla on hyvä käsitys siitä, mitä järjestelmässä kehitetään parhaillaan: 3.7

Kysymyksen tarkoitus oli selvittää, kuinka hyvin olimme onnistuneet viestimään kehityskohteista, niiden ollessa työn alla. Meillä ei ollut tähän strukturoitua esitystapaa, vaan olimme viestineet kehityskohteista ja niiden aikataulusta satunnaisesti käyttäjille muun vuorovaikutuksen yhteydessä, kuten koulutuksissa tai loppukäyttäjien kysellessä kaipaamiensa toiminnallisuuksien perään.

- Tiedän mitä muutoksia järjestelmään on lähiaikoina tehty: 3.8

Uusista toiminnallisuuksista pyrittiin viestimään Slack-kanavalla, jota käytettiin myös loppukäyttäjätuen kanavana. Alun perin kanava luotiin loppukäyttäjien kysymyksiä varten ja näin ollen sitä ei aluksi mielletty tiedottamisen kanavaksi. Tämän kysymyksen tarkoituksena oli selvittää, kuinka hyvin viestintämme tehdyistä muutoksista oli jalkautunut loppukäyttäjille.

- Tiedän mitä järjestelmässä ollaan kehittämässä seuraavan kvartaalin aikana 3.1

Tulevaisuuden visioista ja roadmapista oli ollut puhetta loppukäyttäjien kanssa koulutuksissa ja toimistolla käydyissä keskusteluissa. Oli myös keskusteltu uuden asiakkuudenhallintajärjestelmän tulevaisuuden potentiaalista, ja loppukäyttäjät esittivät toiveita uusien toiminnallisuuksien suhteen. Näitä tulevaisuuden suunnitelmia ei kuitenkaan esitetty loppukäyttäjille keskitetysti tai missään strukturoidussa muodossa. Ensimmäisellä kierroksella kysymys auttoi tarkastelemaan arvosteluasteikkoa, sillä kyselyhetkellä edes pääkäyttäjäryhmällä ei ollut tarkkaa tietoa seuraavan kvartaalin tavoitteista, ja silti loppukäyttäjien antama arvosana oli keskiarvoltaan 3.1. Kyselytutkimuksen toisella kierroksella kysymyksen tarkoitus oli tarkastella roadmapin selkeyttä ja onnistumistamme sen esittämisessä.

- Tiedän, millaisia tavoitteita meillä on järjestelmän kehityksen suhteen pitkällä aikavälillä: 3.5

Tämä kysymys poikkeaa aikaisemmasta siinä, että aikasidonnaisuus puuttuu. Loppukäyttäjille on useampaan otteeseen kerrottu koulutusten ja muun vuorovaikutuksen yhteydessä, mitä toivottuja toiminnallisuuksia ollaan tuomassa järjestelmään, mutta aikatauluun ei olla otettu kantaa. Näin ollen ensimmäisellä kierroksella kysymys toimii jälleen arvosteluasteikon tarkastelussa, sillä arvosanan tähän kysymykseen pitäisi olla alhainen, mutta hieman edeltävää kysymystä korkeammalla tasolla, kuten se onkin.

- Koen, että minun tarpeitani on huomioitu järjestelmän kehityksessä: 4.5

Pääkäyttäjäryhmään ja järjestelmän kehitykseen on toivottu mahdollisimman paljon loppukäyttäjien näkemyksiä. Toiset loppukäyttäjät ovat olleet toiveidensa esittämisessä ja osallistumisessa aktiivisempia kuin toiset. Tällä kysymyksellä pyritään kartoittamaan yksilötason tyytyväisyyttä ja hajontaa vastauksissa.

- Koen, että tiimini tarpeita on huomioitu järjestelmän kehityksessä: 4.5

Kaikki yksilöt eivät ole osallistuneet kehitystyöhön, suunnitteluun tai ideointiin, mutta jokaisesta tiimistä on pyritty saamaan vähintään yksi henkilö osallistumaan aktiivisesti. Tällä kysymyksellä selvitetään, onko tiimien tarpeet onnistuttu huomioimaan tällä menetelmällä loppukäyttäjien näkemyksen mukaan.

- Koen, että voimme vaikuttaa siihen, mihin suuntaan järjestelmää kehitetään: 4.6

Pääkäyttäjäryhmän tavoite on ollut viestiä loppukäyttäjille, että kehityksen suunta perustuu heidän toiveisiinsa ja tarpeisiin. Kysymyksen tarkoituksena on tarkastella, kuinka hyvin kyseinen viesti on välittynyt.

- Palautteen antamiseen on selkeät kanavat: 4.8

Kyselyhetkellä palautteen antamiseen on käytetty Slack-kanavaa, ja vastausten pohjalta on tarkoitus tehdä päätös siitä, vaihdetaanko kanavaa.

- Palautteen antamiseen on kannustettu: 4.8

Vain suhteellisen pieni osa loppukäyttäjistä on aktiivisesti antanut palautetta. Kysymyksellä pyritään varmistamaan, että pääkäyttäjäryhmän palautepositiivinen asenne on onnistuttu viestimään loppukäyttäjille, eikä palautteen puute johdu huonosta viestinnästä.

- Koen, että palaute huomioidaan: 4.8

Oleellisesti palautteen jättämisen motivaatioon vaikuttava tekijä on tunne siitä, että palautteet huomioidaan. Nykyisellään palautteeseen erinäisissä kanavissa on pyritty vastaamaan suoraan, ja niitä on pyritty kirjaamaan ylös.

Kyselytutkimuksen tulokset ovat korkeammalla tasolla kuin olisi ollut odotettavissa. Vastaukset kysymyksiin 3 ja 4 viittaisivat siihen, että loppukäyttäjät ovat suhteuttaneet vastauksensa tilanteeseen, eivätkä ole halunneet antaa alhaisia arvosanoja, koska ymmärtävät käyttöönoton haasteet ja alkuvaiheen puutteet. Loppukäyttäjien kohteliaisuus vastauksissa vaikeuttaa tulosten tulkitsemista, joskin selkeitä eroja on havaittavissa ja tyytymättömyyden kohteet tunnistettavissa.

Vastauksista voi tulkita, että loppukäyttäjien luottamus pääkäyttäjäryhmään ja siihen, että järjestelmää tehdään loppukäyttäjiä varten, on vahva. Kehitystyön viestinnässä ja työn läpinäkyvyydessä on puolestaan puutteita. Suhteutettuna toisiinsa vastaukset ovat oletetun mukaisia, mutta keskiarvoltaan merkittävästi odotettua korkeammalla.

3 Prosessinhallinta ja ketterä kehitys

Teoriapohjaa valitessa ehdolla oli useita teorioita ja malleja, joista vain osa valikoitui syvempään tarkasteluun. Kehitystyötä ja prosesseja läpikäydessä erilaisia prosessirunkoja, teorioita ja malleja tuli vastaan ja peruskuvausten pohjalta syvempään tarkasteluun valikoituivat seuraavat: Lean, Scrum, Agile, Kanban, DevOps, XP, Knowledge management, Palvelumuotoilu, ja UX. Hakukriteereinä olivat kevyet prosessikehityksenmallit, moderni ohjelmistokehitys ja loppukäyttäjälähtöisyys.

Lean ja DevOps mallit valikoituivat mukaan, sillä tavoitteena oli luoda kevyt malli ja näille malleille yhteinen tavoite hukan poistamisesta kuulosti mielekkäältä. Nämä mallit sisälsivät paljon hyviä metodeja prosesseihin, joissa vaiheistus ja tehtävien siirrot henkilöltä tai funktiolta toiselle aiheuttavat paljon hukkaa. Kohdeorganisaatiossa kehitystyö toteutetaan pääsääntöisesti saman funktion sisällä ja yksittäinen henkilö toteuttaa usein tehtävän alusta loppuun. Näin ollen valtaosa hukan minimoimiseen esitellyistä tekniikoista eivät olleet sovellettavissa kohdeorganisaatioon ja nämä mallit jätettiin tästä syystä pois.

Viittauksia lean-ajatteluun ilmentyy valituissa malleissa, mutta lähtökohtaisesti pelkään leaniin keskittyvä kirjallisuus tai esitellyt lean-mallit eivät tukeneet loppukäyttäjälähtöistä ja ketterää kehitysrunkoa, jota tässä työssä tavoitellaan. Näin ollen leanille ei omistettu omaa kappaletta, vaikka siihen viitataan usein ketterän kehityksen kirjallisuudessa.

DevOps perustuu kehitys- ja operaatiot-funktioiden yhdistämiseen kehitysprosessissa. Kohdeorganisaatiossa asiakkuudenhallintajärjestelmän kehitystyö tehdään kokonaisuudessaan operaatiot-funktion toimesta. Tästä syystä DevOps teoria ei palvellut riittävän kattavasti tämän työn tavoitteita.

Knowledge management, eli tietämyksenhallinta, herätti kiinnostuksen tiedonjaon ja dokumentaation tueksi. Ylätasolla käytännönläheiset mallit oikeaoppiseen tiedonjakoon vaikuttivat hyödyllisiltä, mutta knowledge management olikin suunnattu byrokraattisempiin, suuriin organisaatioihin, eivätkä teoriat ja mallit olleet luontevasti sovellettavissa kohdeorganisaation kaltaisessa ketterässä ja verrattaen pienessä yrityksessä.

”User Experience Design”, eli UXD, tarjosi hyvin mielenkiintoisia malleja loppukäyttäjäkokemuksen parantamiseen, mutta tämä työ keskittyy kehitysprosessiin ja siitä viestimiseen. Näin ollen UXD jäi rajauksen ulkopuolelle.

Lopulta empiiriseen osuuteen valikoitui perusperiaatteena ja ideologiana toimiva Agile – eli ketterä kehitys. Ketterän kehityksen periaatteet soveltuivat hyvin tämän opinnäytetyön tavoitteisiin. Ketterä kehitys tukee kohdeorganisaation tarvetta luoda kevyt, nopeaan arvontuottoon perustuva loppukäyttäjälähtöinen prosessi. Lisäksi soveltuvia malleja vertaillessa oli helposti havaittavissa, että

lähes kaikki sopivat mallit käyttivät ketterän kehityksen periaatteita perustanaan. Ketterän kehityksen ideologian tueksi läpikäydään tarkemmin näitä käytännönläheisiä malleja: Kanban, Scrum ja XP, jotka noudattavat hyväksi todettuja ketterän kehityksen periaatteita, mutta tarjoavat hieman enemmän konkretiaa.

Kanban-mallissa keskitytään paljon loppukäyttäjäystävällisiin visualisointeihin ja selkeyteen, jotka palvelevat tämän työn tavoitetta loppukäyttäjien osallistamisen osalta. Lisäksi Kanban mahdollistaa kevyet ja säännöstelemättömät prosessit ilman tarkkoja roolituksia tai tapahtumia, mahdollistaen mallin helpon käyttöönoton olemassa olevien ratkaisujen rinnalle.

Scrum puolestaan tarjoaa selkeää prosessirunkoa kehitystyön hallinnointiin, tarjoten tuoreelle pääkäyttäjäryhmälle rungon ja viitekehyksen yhteisen työn hallinnointiin strukturoidusti.

XP-malli keskittyy ohjelmistokehityksen perusperiaatteisiin, tavoitellen yksinkertaisuutta ja laatua. Koska pääkäyttäjäryhmä on kohtalaisen kokematon, on arvokasta huomioida ohjelmistokehityksen fundamentaaliset perusteet. XP-malli on luotu Kent Beckin toimesta vuonna 1999, haastaen silloin suositun vesiputousmallin. Vaikka XP-malli on suhteellisen vanha, löytyy siitä hyviä perustavanlaatuisia ohjelmistokehityksen periaatteita ja käytännön malleja, jotka ovat relevantteja edelleen.

Loppukäyttäjien osallistaminen huomioidaan kattavasti jo edellä mainituissa malleissa, mutta siihen syvennytään vielä palvelumuotoilun ja viestinnän teorioiden kautta.

3.1 Ketterä kehitys

Agile, eli ketterä kehitys, on ideologia ja ajattelutapa. Ketterä kehitys ei itsessään tarjoa prosessirunkoa tai työkaluja, vaan koostuu arvoista ja periaatteista. (Shore 2007, luku 1.1.)

> ”Valikoima arvoja ja periaatteita, jotka vaalivat muutosta, yhteistyötä, loppukäyttäjälähtöisyyttä, ja jatkuvaa innovaatiota, tuottavuutta tavoitellen” (Gullo 2016, luku 5.1).

Ketterän kehityksen konkretian puute on myös luonut kysynnän käytännönläheisille ketterän kehityksen oppaille. Ketterän kehityksen ideologian toteuttamiseen on luotu monia itsenäisiä, käytännönläheisiä malleja ja runkoja, kuten Scrum, Lean, XP ja Kanban. (Gullo 2016, luku 5.1.) Yhteistä näille kaikille malleille on tavoite tarjota käytännönläheisiä tapoja toteuttaa ketterää kehitystä (Moreira 2017, luku 2.1).

3.1.1 Ketterän kehityksen julistus

Ketterän kehityksen periaatteita on hyödynnetty erinäisissä käyttötarkoituksissa ja asiayhteyksissä jo vuosisatoja, mutta ohjelmistokehityksen kontekstissa ”Agile” nimen ja selkeän muodon nämä periaatteet saivat vuonna 2001, kun 17 alan asiantuntijaa kirjoittivat ketterän kehityksen julistuksen:

> Löydämme parempia tapoja tehdä ohjelmistokehitystä, kun teemme sitä itse ja autamme muita siinä. Kokemuksemme perusteella arvostamme:
>
> Yksilöitä ja kanssakäymistä enemmän kuin menetelmiä ja työkaluja
> Toimivaa ohjelmistoa enemmän kuin kattavaa dokumentaatiota
> Asiakasyhteistyötä enemmän kuin sopimusneuvotteluja
> Vastaamista muutokseen enemmän kuin pitäytymistä suunnitelmassa
>
> Jälkimmäisilläkin asioilla on arvoa, mutta
> arvostamme ensiksi mainittuja enemmän.
> (Ketterän ohjelmistokehityksen julistus.)

Julistus sisältää näiden arvojen lisäksi listauksen julistuksen takana olevista periaatteista:

> Tärkein tavoitteemme on tyydyttää asiakas toimittamalla tämän tarpeet täyttäviä versioita ohjelmistosta aikaisessa vaiheessa ja säännöllisesti.
>
> Otamme vastaan muuttuvat vaatimukset myös kehityksen myöhäisessä vaiheessa. Ketterät menetelmät hyödyntävät muutosta asiakkaan kilpailukyvyn edistämiseksi.
>
> Toimitamme versioita toimivasta ohjelmistosta säännöllisesti, parin viikon tai kuukauden välein, ja suosimme lyhyempää aikaväliä.
>
> Liiketoiminnan edustajien ja ohjelmistokehittäjien tulee työskennellä yhdessä päivittäin koko projektin ajan.
>
> Rakennamme projektit motivoituneiden yksilöiden ympärille. Annamme heille puitteet ja tuen, jonka he tarvitsevat ja luotamme siihen, että he saavat työn tehtyä.
>
> Tehokkain ja toimivin tapa tiedon välittämiseksi kehitystiimille ja tiimin jäsenten kesken on kasvokkain käytävä keskustelu.
>
> Toimiva ohjelmisto on edistymisen ensisijainen mittari.
>
> Ketterät menetelmät kannustavat kestävään toimintatapaan. Hankkeen omistajien, kehittäjien ja ohjelmiston käyttäjien tulisi pystyä ylläpitämään työtahtinsa hamaan tulevaisuuteen.
>
> Teknisen laadun ja ohjelmiston hyvän rakenteen jatkuva huomiointi edesauttaa ketteryyttä.
>
> Yksinkertaisuus - tekemättä jätettävän työn maksimointi - on oleellista.
>
> Parhaat arkkitehtuurit, vaatimukset ja suunnitelmat syntyvät itseorganisoituvissa tiimeissä.
>
> Tiimi tarkastelee säännöllisesti, kuinka parantaa tehokkuuttaan,

ja mukauttaa toimintaansa sen mukaisesti.
(Ketterän ohjelmistokehityksen julistus.)

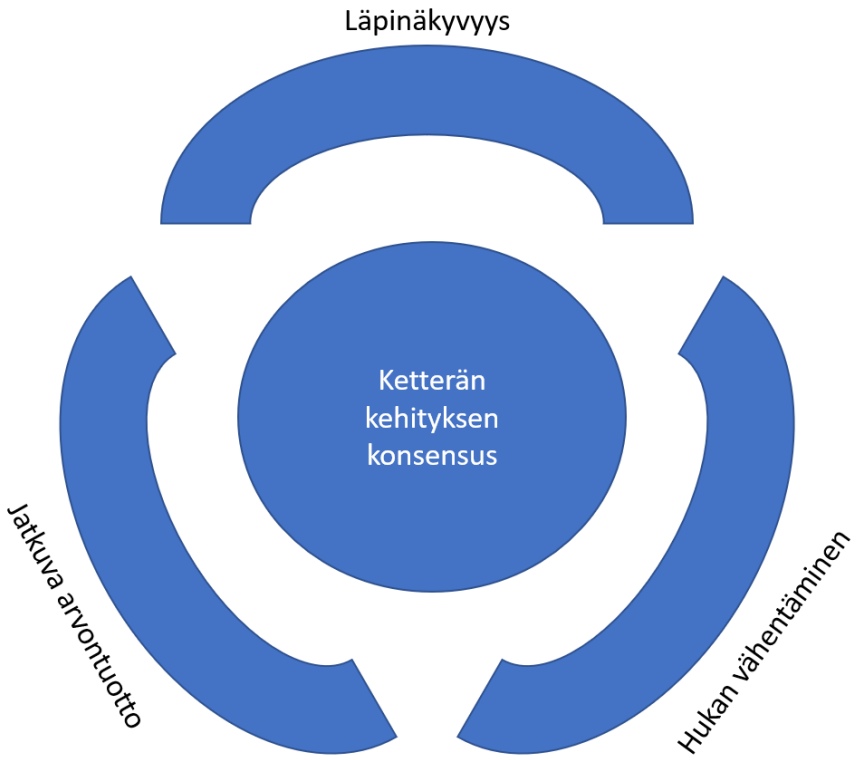
3.1.2 Ketterän kehityksen konsensus

Kuten aikaisemminkin mainittu, ketterän kehityksen julistus on innoittanut monia ohjelmistokehityksen prosessimalleja. Nämä mallit sisältävät sekä eroavaisuuksia, että päällekkäisyyksiä, mutta Guckenheimer & Loje ovat kiteyttäneet näiden mallien yhdessä luoman paradigman ketterän kehityksen konsensukseksi. (Guckenheimer & Loje 2012, luku 1.)

Ketterän kehityksen konsensus koostuu kolmesta perusperiaatteesta (kuva 2), jotka tukevat toinen toisiaan:

1. Jatkuva arvontuotto, jossa arvon määrittelee loppukäyttäjä
2. Hävikin poistaminen, jotta jatkuvalle arvontuotolle ei ole esteitä
3. Läpinäkyvyys

(Guckenheimer & Loje 2012, luku 1.)



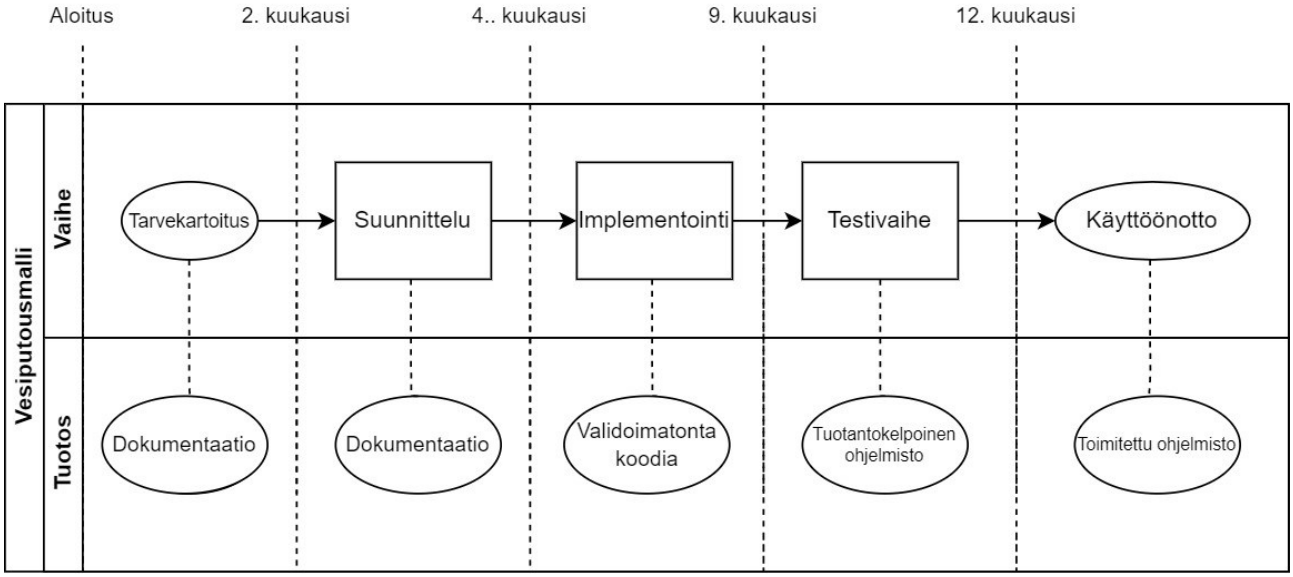
Kuva 2 – Ketterän kehityksen konsensus (Guckenheimer & Loje 2012, luku 1).

3.1.3 Vesiputousmalli, ketterän kehityksen edeltäjä

Ennen ketterää kehitystä ja sen periaatteisiin nojaavia malleja, suosittiin ohjelmistokehityksessä vesiputousmallia. Vesiputousmallin ymmärtämäinen auttaa hahmottamaan niitä ongelmia, mihin ketterän kehityksen kautta lähdettiin hakemaan vastauksia.

Vesiputousmallilla tarkoitetaan ohjelmistokehityksessä käytettyä projektirunkoa, joka perustuu vaiheistettuun kehitystyöhön, jonka lopullisena tavoitteena on julkaista valmis ohjelmisto (kuva 3). Vaiheet ovat toisistaan irrallisia ja vaiheesta toiseen siirrytään määritellyn luovutusprosessin kautta (Gullo 2016, luku 1.1).

Vaiheistuksessa ei edetä taaksepäin, vaan siirtyminen seuraavaan vaiheeseen tarkoittaa aina, että edeltävä vaihe on valmis. Ennen julkaisua, avataan pala palalta rakennettu ohjelmistokokonaisuus testaamista varten, ja testausvaiheen jälkeen julkaistaan valmis kokonaisuus. (Gullo 2016, luku 1.1.)



Kuvio 3. Vesiputousmalli (mukaillen Gullo 2016, luku 1.1.)

Merkittävänä haasteena vesiputousmallissa on läpinäkyvyyden puute. Epävarmuus siitä, palveleeko kehitettävä ohjelmisto loppukäyttäjien tarpeita, on läsnä aina testaamisvaiheeseen asti. Mahdolliset väärinkäsitykset tai puutteet voidaan huomata vasta tässä vaiheessa, jolloin korjausliikkeet ovat huomattavasti suurempia, kuin ne olisivat olleet varhaisemmassa vaiheessa. Toisaalta, vaikka ohjelmistohanke menisi hyvin, vesiputousmallilla tuotettu ohjelmisto alkaa tuottamaan arvoa vasta täysin valmiina, ja on näin ollen väkisinkin hidas prosessi arvontuottamisen kannalta. (Gullo 2016, luku 1.1.)

Kuten kuviosta X huomaa, ei ennen yhdeksättä kuukautta tuoteta muuta kuin validoimatonta koodia, jolla viitataan siis loppukäyttäjien validoinnin puutteeseen. Vaikka toteutus olisi teknisestä näkökulmasta laadukas, voi se olla loppukäyttäjille hyödytön, mikäli tarve on ymmärretty väärin. (Gullo 2016, luku 1.1).

3.1.4 Ketterän kehityksen ideaputki

Ketterän kehityksen julistus ei sisällä käytännön malleja tai menetelmiä, mutta julistus on toiminut pohjana monille käytännönläheisille malleille ja kirjallisuudelle. Valtaosa näistä on saanut oman nimensä, kuten Scrum ja Kanban, mutta osa kirjallisuudesta kantaa suoraan ketterän kehityksen nimeä, kuten Moreiran ”The Agile Enterprise: Building and Running Agile Organizations”. Tässä kappaleessa esitellään Moreiran ketterän kehityksen ideaputki.

Backlog täydentyy useista eri lähteistä, mutta pääasiallisena lähteenä tulisi olla käyttäjien ideat ja palautteet. Ideointia varten tulisi olla oma keskitetty kanava, joka tarjoaa keskitetyn näkymän tuotettuihin ideoihin, helpottaa niiden backlogiin vientiä ja auttaa nostamaan tärkeät ideat korkeammalle priorisoinnissa. Lisäksi ideaputki tarjoaa organisaatiotason läpinäkyvyyttä tehtävään kehitystyöhön. (Moreira 2017, luku 11.)

Ideaputki vaatii ketterän kehittämisen kulttuurin, jossa uudet ideat otetaan nopeasti käsittelyyn ja niitä muokataan ja priorisoidaan palautteen pohjalta. Mikäli organisaatio on pieni tai järjestelmiä on vain yksi, voi ideaputki olla osa tuotteen backlogia. Myös ideaputken ollessa osa backlogia tulee sidosryhmillä olla mahdollisuus tuottaa ja priorisoida ideoita. (Moreira 2017, luku 11.)

Ideaputken tulisi olla kaikille nähtävillä (Moreira 2017, luku 7).

Ideoiden keräämisen ensimmäinen vaihe on idean dokumentointi. Tämän dokumentaation tulisi olla näkyvillä kaikille organisaatiossa, jotta korjauksia voidaan tehdä missä vain prosessin vaiheessa. Idea voi tulla keneltä tahansa. (Moreira 2017, luku 11.)

Dokumentaation tulisi sisältää kuvaus, persoona, segmentti, arvo, riskit ja oletukset. Idean kirjaamisesta tulisi kuitenkin tehdä mahdollisimman kevyttä, sillä ideaa ei olla vielä hyväksytty ja sitä voi rikastaa prosessin myöhemmissä vaiheissa. (Moreira 2017, luku 11.)

Seuraavaksi siirrytään idean esittelyyn. Esittelyvaiheessa idea tulee yhteiseen ideaputkeen kaikkien nähtäville arvon tai prioriteetin mukaisessa järjestyksessä. Ylimmäksi nousee korkeimman arvon tai prioriteetin omaavat ideat, ja sidosryhmien tulisi säännöllisesti keskustella yläpään ideoista ja niiden viemisestä eteenpäin. Tässä vaiheessa on myös tärkeää haastaa idean tuottamaa arvoa

tai toteutuskelpoisuutta. Kun ideasta on keskusteltu ja mahdolliset korjaukset sisältöön tai priorisointiin on tehty, voi idean siirtää eteenpäin. (Moreira 2017, luku 11.)

Hyväksymisen jälkeen idean toteuttajien tulee ymmärtää idea ja jäsennellä se toteuttamista varten pienemmiksi tehtäviksi. Tässä kohtaa kaikkien järjestelmän kehittämiseen ja toteuttamiseen liittyvien sidosryhmien on hyvä olla mukana. Idea tulisi saada muunnettua backlog tehtäväksi ja mikäli tämä vaatii kollaboraatiota sidosryhmien kanssa, tulisi se tehdä ennen etenemistä toteutusvaiheeseen. (Moreira 2017, luku 11.)

Toteutusvaiheen tulisi sisältää kehittäjien suosima prosessi, jossa suunnitellaan, rakennetaan, tarkastellaan, ja muokataan toteutusta iteratiivisesti, kunnes se on toimituskelpoinen. On oleellista, että sidosryhmät osallistetaan jo toteutusvaiheen aikana, jotta lopputulos tuottaisi aidosti arvoa. (Moreira 2017, luku 11.)

Kun toteutus on valmis, on idea tai sen osa valmiina julkaistavaksi. Julkaisusta tulee tiedottaa sidosryhmiä ja ideaputkea tulee päivittää, jotta kehitystyö on läpinäkyvää organisaatiolle ja kuva nykytilanteesta säilyy reaaliaikaisena. (Moreira 2017, luku 11.)

3.2 Extreme Programming (XP)

Extreme Programming – eli XP – on Kent Beckin luoma ohjelmistokehityksen malli. XP tavoittelee laadukasta työn jälkeä tarjoamalla tekniikoita koodaamiseen, kommunikointiin ja yhteistyöhön. Alun perin XP:n kuvattiin soveltuvan erityisesti pienille ja keskikokoisille tiimeille ohjelmistokehitykseen nopeasti muuttuvassa ympäristössä, jossa tarvemäärittely on suurpiirteistä. Beckin mukaan käytäntö on kuitenkin osoittanut, että malli soveltuu myös suuremmille organisaatioille, mikäli rakenne on mahdollista säilyttää kevyenä. (Beck &Anders 2004, luku 1.1.)

3.2.1 Lähtöhaasteet ja ratkaisut

Kent Beck kehitti ja julkaisi kirjan ”Extreme Programming Explained” vuonna 1999. Extreme programming on ohjelmistokehityksen malli, jonka kehittämisen perustaksi luotiin lista ohjelmistokehityksen ongelmista ja ne pyrittiin ratkaisemaan pala kerrallaan. (Beck 1999, luku 1.1.)

Ensimmäinen tunnistettu ongelma oli aikataulutuksen haasteet ja liika optimismi. XP:n esittämä ratkaisu tähän on iteratiivinen kehitysprosessi, jossa työ jaetaan lyhyisiin, yhdestä neljään viikkoa kestäviin sykleihin. Iteraatio pilkotaan vielä pienempiin tehtäviin, jotka priorisoidaan, ja joita työstetään tärkeysjärjestyksessä. Näin saavutetaan parempi kontrolli iteraation edistymisestä ja mahdollisesti rästiin jäävät tehtävät eivät ole ainakaan tärkeimmästä päästä. (Beck 1999, luku 1.1.)

Toinen yleinen haaste oli projektien peruuntuminen ennen julkaisua. Tätä haastetta käsiteltiinkin jo edellisessä kappaleessa vesiputousmallin yhteydessä, jonka ongelmakohtiin XP monin kohdin puuttuu. Ratkaisu tähän on nopea tuotantokäyttöönotto. Ensimmäiseen julkaisuun tulisi valita mahdollisimman pieni, mutta arvoa tuottava kokonaisuus. Näin päästään tuotantoon ja tuottamaan arvoa nopeasti ja matalla riskillä. (Beck 1999, luku 1.1.)

Kolmas esitetty ongelma on ohjelmiston happaneminen. Ohjelmisto ei onnistuneenkaan julkaisun jälkeen skaalaudu, kehity tai ylläpidä itseään automaattisesti ja pahimmillaan ohjelmistoa voi joutua vaihtamaan jo lähivuosina käyttöönotosta. Tähän Beck esittää ratkaisuksi XP:n kattavaa sarjaa testejä, joilla ohjelmiston laatua valvotaan jatkuvasti. (Beck 1999, luku 1.1.)

Neljäntenä listalla on ohjelmiston viallisuus. Monipuoliset ohjelmistot sisältävät usein myös monipuolisesti vikoja, ja mikäli näitä on liikaa voi muutosvastarinta ja käyttöaste nousta ongelmiksi. Vikoja voidaan minimoida toiminnallisuuksien kattavalla testaamisella kehitysvaiheessa, niin kehittäjien kuin loppukäyttäjien näkökulmasta. (Beck 1999, luku 1.1.)

Viides haaste on väärinkäsitykset. Ohjelmistot luodaan lähtökohtaisesti ratkaisemaan liiketoiminnan tarpeita, mutta toisinaan tämän tarpeen viestimisessä epäonnistutaan. Tällöin ohjelmisto ei

ratkaise esitettyä ongelmaa, eikä palvele liiketoimintaa. Sidosryhmät tulisi osallistaa kehitystyöhön koko prosessin ajan ja tarvekuvausta tulisi tarkentaa ja päivittää projektin edetessä, kun molemminpuolinen ymmärrys kehittyy. (Beck 1999, luku 1.1.)

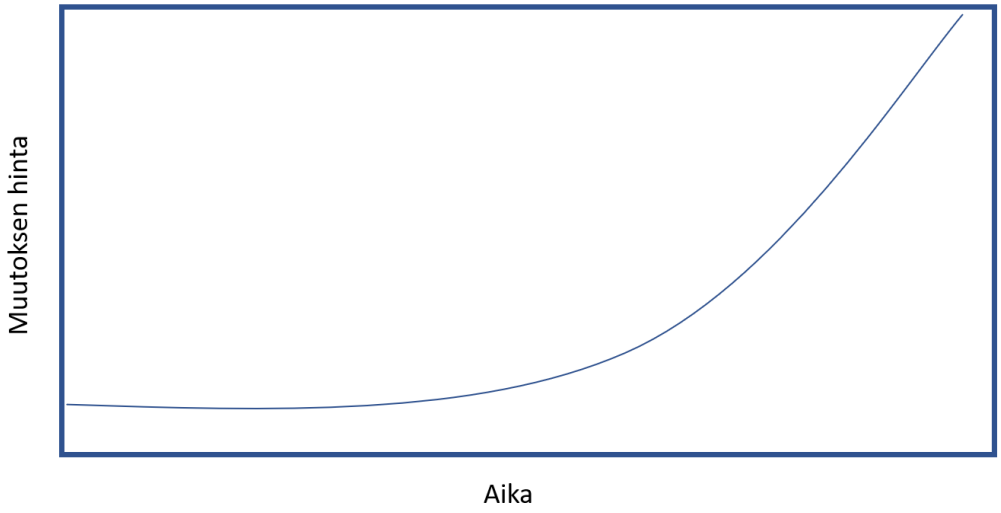
Kuudentena haasteena Beck esittelee liiketoiminnan muutokset. Liiketoiminta tapaa kehittyä ja muuttua, eikä ohjelmiston kehitysvaiheessa annettu tarvekuvaus välttämättä päde enää myöhemmin. XP mallin mukainen iteratiivinen prosessi kehittää toiminnallisuuksia nopealla rytmillä ja tuoreilla tarvekuvauksilla. Jatkuva liiketoimintatarpeiden tarkentaminen ja päivittäminen mahdollistaa sen, että ohjelmisto pysyy liiketoiminnan kehityksessä mukana. (Beck 1999, luku 1.1.)

Seitsemäs esimerkki on liialliset toiminnallisuudet. Toiminnallisuustarpeita kerätessä, kehittäessä ja ideoidessa voi helposti innostua liikaa. Uusia toiminnallisuuksia voidaan luoda vauhdilla, mutta tämä kasvattaa ohjelmiston monimutkaisuutta ja ei välttämättä palvele liiketoimintatarpeita. Tästä syystä tehtäviä tulisi priorisoida, ja niistä pitäisi toteuttaa vain tarvittavat. (Beck 1999, luku 1.1.)

Kahdeksas ja viimeinen ongelmakohta on henkilöstöriippuvuus. Ohjelmistokehitys on hyvin henkilöriippuvaista ja poistuman riksi tulee huomioida. XP pyrkii huomioimaan tämän tarjoten mallin, joka on kehittäjäystävällinen ja näin lähtökohtaisesti pienentää poistumariskiä. Mikäli henkilöstömuutoksia kuitenkin tulee, sisältää XP mallin tehtävien siirtoon. (Beck 1999, luku 1.1.)

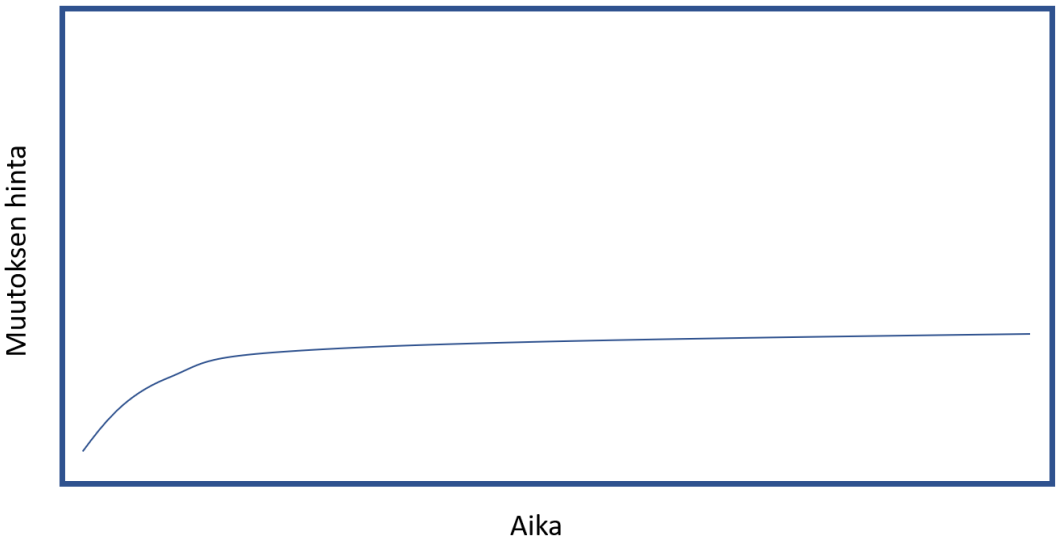
3.2.2 Muutoksen hinta

Ohjelmistokehityksessä muutoksen hinnan on nähty kasvavan eksponentiaalisesti sen mukaan, kuinka pitkällä kehityksessä ollaan (kuva 4.) Ideoita tarkastellessa on vielä helppoa ja edullista todeta, ettei idea ole hyvä tai se kaipaa muutosta. Mikäli idea puolestaan on hyväksytty, toteutettu ja julkaistu tuotantokäyttöön, voisi sen muuttaminen siinä kohtaa nähdä kalliiksi ja raskaaksi. (Beck 1999, luku 1.5.)



Kuva 4, muutoksen hinta ennen (mukaillen Beck 1999, luku 1.5.)

Extreme programming perustuu siihen oletukseen, ettei näin ole. Tekninen lähtökohta XP:n toteuttamiselle on se, että ohjelmisto on kevyesti räätälöitävissä, ja muutoksen hintakäyrä saadaan tasattua. Helposti räätälöitävät ohjelmistot ja testiympäristöt, joista kehitystyöt on helppo siirtää tuotantoon mahdollistavat tämän. Kun muutoksen hinta säilyy alhaisena (kuva 5), voidaan kehitystyötä tehdä interaktiivisemmin loppukäyttäjien kanssa, julkaisten puolivalmiita toteutuksia tuotantoon ja räätälöiden niitä palautteen pohjalta. (Beck 1999, luku 1.5.)



Kuva 5, muutoksen hinta nykyään (Beck 1999, luku 1.5.)

3.2.3 Arvot

Extreme programming mallin taustalla elää neljä arvoa: kommunikaatio, yksinkertaisuus, palaute ja rohkeus (Beck 1999, luku 1.7).

Kommunikaatio on arvoista ensimmäinen. Useimmat ongelmat ohjelmistokehityksessä juontaa juurensa siihen, että kommunikaatio on katkennut tai puuttunut jossain prosessin vaiheessa. Syitä kommunikaatiopuutoksille voi olla useita. Syynä voi olla huono kulttuuri, jossa ikävistä asioista tai virheistä ei haluta puhua tai epäselvät prosessit, jolloin ei tiedetä milloin, tai kenelle tulisi viestiä. Kommunikaatiohaasteita voi olla yhtä hyvin sidosryhmien välisessä viestinnässä, kuin sisäisessä tiedonkulussa. (Beck 1999, luku 1.7.)

XP-mallissa kommunikaatiohaasteita pyritään ehkäisemään tuomalla kehitysprosessiin sellaisia vaiheita, jotka vaativat kommunikaatiota. Näitä ovat testaaminen, parityöskentely ja tehtävien arvioiminen. (Beck 1999, luku 1.7.)

Yksinkertaisuus arvona tarkoittaa sitä, että ratkaisutoteutuksia suunnitellessa pyritään aina miettimään, mikä on yksinkertaisin toteutus, jolla tarpeeseen saadaan vastattua. Tässä otetaan tietoinen riski siitä, että yksinkertainen ratkaisu voi vaatia työstämistä jälkeenpäin. Samalla luotetaan siihen, että pitkässä juoksussa jälkityöt vievät vähemmän aikaa kuin se, että aina pyritään tuottamaan loppuun asti ajateltuja ja hiottuja ratkaisuja. Tämä myös vauhdittaa toimituksia ja mahdollistaa nopeamman arvon tuottamisen. Lisäksi yksinkertaisuus helpottaa kommunikointia, kun taas hyvä kommunikointi edistää yhteistä ymmärrystä ja tekee toteuttamisesta yksinkertaisempaa – ja näin arvot ruokkivat toisiaan. (Beck 1999, luku 1.7.)

Neljäs arvo on palaute. Kehittäjät sortuvat usein liialliseen optimismiin, mutta palaute toimii siihen hyvänä vastalääkkeenä. (Beck 1999, luku 1.7). Viitaten jälleen vesiputousmalliin, ennen ohjelmistokehityksen julkaisussa pidettiin tavoitteena, että ohjelmisto olisi julkaisuvaiheessa valmis. XP-mallissa esitetään, että kehitysvaiheessa olevasta toteutuksesta on äärimmäisen vaikeaa saada palautetta ja siitä syystä tuotantoon pitäisi pyrkiä mahdollisimman nopeasti. Tuotantoympäristössä kattava palautteen kerääminen on mahdollista ja luontevaa. Palaute toimii myös kommunikaatiokanavana ja edistää vuorovaikutusta sidosryhmien välillä. (Beck 1999, luku 1.7.)

Viimeinen arvo on rohkeus. Ohjelmistokehityksessä tulee vastaan tilanteita, jossa uuden innovaation toteuttaminen voi vaatia ohjelmistoon rakenteellista muutosta, joka vie ohjelmistoa taaksepäin päiviä, viikkoja tai jopa kuukausia. Vaihtoehtoisesti ohjelmisto vaatii korjauksen, joka pysäyttää kaiken muun kehityksen useamman iteraation ajaksi. Pienemmässä mittakaavassa yksittäinen kehittäjä voi todeta, että hänen edeltävänä päivänä vaivalla tehty toteutus ei olekaan toimiva, ja tulisi

tuhota kokonaan. Tällaiset hetket vaativat kehittäjiltä rohkeutta tehdä päätöksiä. (Beck 1999, luku 1.7.)

3.2.4 Periaatteet

Arvot antavat XP-mallille suunnan, mutta eivät yksinään ole riittävän konkreettisia tukeakseen käytännön tekemistä. Askeleen lähemmäs arjen käytäntöjä päästään, kun arvot jaetaan käytännönläheisiksi periaatteiksi. (Beck 1999, luku 1.8.)

Ensimmäinen periaate on nopea palaute. Tämä on kaksisuuntainen tie, jossa kehittäjät toivovat nopeaa palautetta toteutuksistaan, mutta vastavuoroisesti reagoinnin tulee olla nopeaa. Palautteenkäsittelyn reaaliaikaisuus parantaa dialogia ja edistää molempien osapuolten oppimista. (Beck 1999, luku 1.8.)

Toisena periaatteena on se, että tulisi tavoitella yksinkertaisuutta. Jokainen ongelma, tarve, ja toive tulisi kohdata sillä oletuksella, että sen ratkaisu on erittäin yksinkertainen ja pyri ratkaisemaan se niin yksinkertaisesti kuin mahdollista. Tämän periaatteen avulla toteutetaan XP-mallin yhtä pääarvoista – yksinkertaisuutta. Logiikka tässä on luottaa siihen, että kun jättää yliajattelun pois ja toteuttaa nopeita yksinkertaisia ratkaisuja, säästää enemmän aikaa, kuin jälkikorjaukset ja monimutkaisuuden lisääminen myöhemmin vievät. (Beck 1999, luku 1.8.)

Kolmas periaate on inkrementaalinen muutos. Pienet muutokset ja korjausliikkeet toimivat XP-mallin mukaan paremmin, kuin suuret ja raskaat muutokset. Tämä pätee ohjelmistokehityksen lisäksi kehittäjätiimin toimintatapojen muutoksiin, henkilöstömuutoksiin, kehityspolun suunnitteluun ja muutosten jalkauttamiseen. (Beck 1999, luku 1.8.)

Muutoksen vaaliminen on neljäs periaate. Tässä ei kuitenkaan tarkoiteta pelkästään muutosmyönteistä asennetta. Periaatteen ydin on se, että kehitysratkaisuista tulisi suosia niitä, jotka jättävät tilaa myöhemmille muutoksille. (Beck 1999, luku 1.8.)

Viides – ja viimeinen pääarvo – on laadukas työ. Laadusta ei saisi missään kohtaa tinkiä. Se vaikuttaa kehittäjien työn mielekkyyteen, loppukäyttäjien yleiskokemukseen ja tyytyväisyyteen sekä tietenkin projektiin kokonaisvaltaisesti. Laatu tulisi priorisoida ajan, resurssien ja laajuuden ohi, eli enemmin käytetään enemmän aikaa ja resursseja, tai jätetään matalamman prioriteetin tarpeita huomioimatta. (Beck 1999, luku 1.8.)

3.2.5 Kehitystyön runko

Arvojen ja periaatteiden määrittelyn jälkeen XP-malli käsittelee itse tekemistä. Kehitystyö on jaettavissa neljään tekemisen osa-alueeseen: koodaukseen, kokeiluun, kuunteluun ja muotoiluun. (Beck 1999, luku 1.9.)

Oli kyseessä aito koodaustyö tai jonkin low-code tai no-code ohjelmiston käyttö, voidaan tekemistä käsitellä koodaamisena. Ohjelmiston toiminnallisuuksien luominen ja kehittäminen vaatii ideoiden strukturointia loogiseen muotoon ja on sen myötä hyvin opettavaista. Koodi ei kaunistele asioita, vaan toteuttaa sen mitä sille on määritelty. Mikäli idea, toteutus tai logiikka on ollut virheellinen, ei koodi toimi. Luova tekeminen vaatii ohjelmistoon uppoutumista ja se on kehittäjälle paras oppimismenetelmä. (Beck 1999, luku 1.8.)

Uusien toiminnallisuuksien toimivuudesta tulee varmistua testaamisen kautta, ennenkö ne julkaistaan tuotantokäyttöön. Testejä luodaan ensin yksinkertaisilla lähtökohdilla, mukaillen optimaalista käyttötapaa. Tämän jälkeen kehittäjä on kuitenkin velvollinen pohtimaan oudompia ja harvinaisempia käyttötapauksia ja testaamaan toiminnallisuutta myös näiden poikkeuksellisten tapausten kulmalta. Vasta kun toiminnallisuus läpäisee kaikki testit, on se teknisestä näkökulmasta julkaisukelpoinen. Hyvät testit kasvattavat myös kehittäjän itsevarmuutta ja ymmärrystä järjestelmästä. (Beck 1999, luku 1.9.)

Kuuntelu on osa kehitystyön kulmakiviä. Kehittäjät harvoin ymmärtävät liiketoiminnan tarpeita tai arkea riittävän hyvin, joten he joutuvat tukeutumaan sidosryhmien toiveisiin ja palautteisiin. Aktiivinen vuorovaikutus on yksinkertaisuudestaan huolimatta käytännön tasolla vaikea toteuttaa ja paras tapa edistää sitä on selkeät viestintämallit ja -kanavat. (Beck 1999, luku 1.9.)

Muotoilulla tarkoitetaan pitkäkatseista kehitystyötä. Toiminnallisuuksia on helppo ja nopea luoda lyhytnäköisesti, mutta ilman hyvää muotoilua törmätään nopeasti siihen tilanteeseen, että toiminnallisuudet risteävät ja ainoa tapa saada uusi toiminnallisuus käyttökelpoiseksi on muuttaa tai poistaa vanha. Hyvän muotoilun tavoitteena on välttää näitä tilanteita (Beck 1999, luku 1.8.)

Hyvään muotoiluun kuuluu, että toiminnallisuudet ovat mahdollisimman yksinkertaisia. Ne toimivat datan näkökulmasta mahdollisimman lähellä lähdedataa ja vaikuttavat vain siellä missä tarvitsee. Ne ovat laajennettavissa ilman, että rakennetta tarvitsee muuttaa. (Beck 1999, luku 1.9.)

Huono muotoilu puolestaan on monimutkaista ja toiminnallisuuksilla on monia riippuvaisuuksia. Lisäksi logiikka on monistettu moneen paikkaan, eivätkä toteutukset ole skaalattavissa ilman rakenteellisia muutoksia. (Beck 1999, luku 1.9.)

3.2.6 Tapahtumat

XP-mallissa esitellään kahdenlaisia tapahtumia: viikko- ja kvartaalipalaverit. (Beck & Andres 2004, luku 1.7.)

Viikkopalaverit järjestetään työviikon alussa, ja niillä on kolme tavoitetta. Ensimmäisenä käydään läpi edistymistä, eli kuinka pitkällä ollaan viime viikon tehtävien suhteen. Seuraavaksi valikoidaan seuraavan iteraation tehtävät asiakkaan toiveiden mukaisesti. Kolmantena puretaan tehtävät tarvittaessa pienempiin osiin, täsmennetään niiden kuvauksia ja nimetään niille omistajuudet. (Beck & Andres 2004, luku 1.7.)

Kvartaalipalavereissa on tarkoitus keskittyä tekemiseen yksittäisiä tehtäviä laaja-alaisemmin. Tavoitteena on tunnistaa pullonkauloja ja ongelmakohtia, suunnitella korjausliikkeitä, valita teemat seuraavalle kvartaalille, poimia tehtäviä teemojen tueksi ja keskustella projektikokonaisuudesta yleisellä tasolla. Kvartaali on lisäksi hyvä tarkasteluväli tiimin reflektointiin ja mahdollisten kokeilujen tai pilotointien tarkastelujaksoksi. (Beck & Andres 2004, luku 1.8.)

3.2.7 Tekniikoita käytäntöön

XP-mallissa esitellään useita erilaisia käytännön tekniikoita ja malleja. Nämä auttavat jalkauttamaan XP-mallin periaatteita konkreettisemmin osaksi kehitystyötä ja arkea. Tässä kappaleessa esitellään tekniikat: suunnittelupeli, pienet julkaisut, yksinkertainen muotoilu, parikoodaaminen, yhteinen omistajuus ja läsnä oleva asiakas.

Suunnittelupeli, eli dialogi kahden liiketoiminnan ja kehityksen välillä. Ohjelmistosuunnittelu koostuu Beckin mukaan aina kahdesta näkökulmasta: liiketoiminnan tarpeista ja teknisistä mahdollisuuksista. Liiketoiminnan näkökulmasta tulisi määritellä seuraavat asiat:

- Laajuus - kuinka suuri osa halutuista toiminnallisuuksista tarvitaan, jotta julkaisu tuotantokäyttöön on hyödyllistä?
- Prioriteetti – jos tarvelistauksesta pitää valita vain osa, mitkä ovat tärkeimpiä ja missä järjestyksessä ne halutaan?
- Julkaisun kokoaminen – kuinka laaja julkaisun tulisi olla, että siitä on eniten hyötyä?
- Laajuus - kuinka suuri osa halutuista toiminnallisuuksista tarvitaan, jotta julkaisu tuotantokäyttöön on hyödyllistä?
- Tärkeät päivämäärät – onko jotain liiketoimintakriittisiä aikatauluja

Kehittäjien tulisi teknisestä näkökulmasta puolestaan tarjota vastaukset seuraaviin:

- Aika-arviot – kuinka kauan minkäkin toiminnallisuuden toteuttaminen vaatii

- Seuraamukset – mitä teknisiä seuraamuksia jollain toiminnallisuudella on? Kärsiikö esimerkiksi datan laatu tai vaatiiko toteutus paljon ylläpitoa?
- Prosessi – kuinka ja kenen toimesta kehitystyö toteutetaan?
- Aikatauluttaminen – mitä kehityskohteita otetaan työn alle seuraavaan iteraatioon ja missä järjestyksessä ne tehdään.

Suunnittelutyö on jatkuvaa dialogia näiden kahden näkökulman välillä, joiden tulisi olla keskustelussa mahdollisimman tasavertaisia. (Beck 1999, luku 1.10.)

Pienet julkaisut. Ketterässä ohjelmistokehityksessä tavoitellaan tiivistä julkaisuväliä. On kuitenkin tärkeää, että julkaisut sisältävät aina toimivia ja järkeviä kokonaisuuksia, eivätkä koostu puolikkaista tai keskeneräisistä toiminallisuuksista. (Beck 1999, luku 1.10). Pienistä julkaisuista saadaan paras arvo, kun suunnittelutyö on tehty hyvin ja julkaisuihin saadaan aina jotain korkean prioriteetin tuotoksia mukaan. Julkaisuja puolestaan helpottaa, kun toiminnallisuuksien testaamiseen ja tuotantoon siirtoon on hyvät mallit ja työkalut. Lisäksi tätä mallia helpottaa, kun muistaa XP-mallin periaatteen ”oleta yksinkertaisuutta”, ja julkaisee sillä hetkellä riittäviä, yksinkertaisia toteutuksia. (Beck 1999, luku 1.11.)

Yksinkertainen muotoilu. Uusien toiminnallisuuksien tulisi aina täyttää tietyt kriteerit.

1. Niiden tulee läpäistä kattavat testit, jotka kehittäjät luovat toimivuuden varmistamiseksi.
2. Duplikaattilogiikkaa ei saisi luoda, kuten esimerkiksi rinnakkaisia hierarkiamalleja.
3. Toteuttaa sen mitä kehittäjä haluaa
4. Sisältää mahdollisimman vähän kategorioita ja metodeja

Niin pitkään, kun vaiheet 1–3 toteutuvat, voi toiminnallisuutta riisua ja keventää. Koska XP-mallissa uskotaan, että muutos on halpaa, ei monimutkaisuutta luoda turhaan tai ennakkoon, vaan pyritään tekemään mahdollisimman yksinkertaisia ratkaisuja, joita laajennetaan vain tarvittaessa. (Beck 1999, luku 1.10.)

Parikoodaaminen tai parikehittäminen tarkoittaa yhdessä työskentelyä niin, että toinen tekee ja toinen seuraa. Ajatus on, että seuraava osapuoli arvioi tekemisen oikeellisuutta, pohtii mikä mahdollisesti menee väärin ja voisiko tarpeen ratkaista jollain yksinkertaisemmalla tavalla. Tällä mallilla työskentely ei ole niin virhealtista, osaaminen jakaantuu tekijöiden välillä ja työn laatu paranee. (Beck 1999, luku 1.10.)

Yhteinen omistajuus. Kukin kehittäjä voi tehdä muutoksia toisen kehittäjän tuottamiin toiminnallisuuksiin tai koodiin, mikäli näkee sille tarvetta ja osaa tehdä parannukset. Kun yksittäisistä toiminnallisuuksista tai osa-alueista ei jaeta omistajuutta, vaan omistajuus on yhteistä, saavat kehittäjät paremman kokonaiskuvat järjestelmästä. Tällä mallilla saadaan myös minimoitua odotusaikoja

pienten korjausten tekemisessä, kun kaikkea ei tarvitse kierrättää omistajuuksien kautta. (Beck 1999, luku 1.10.)

Läsnä oleva asiakas. Loppukäyttäjillä tulee olla edustaja tai edustajia, jotka ovat läsnä kehittäjien arjessa. Loppukäyttäjien edustajat vastaavat kysymyksiin, ratkovat kiistelyitä toteutusvaihtoehdoista, asettavat prioriteetteja ja jakavat mielipiteitä. On oleellista, että tämä loppukäyttäjien edustaja on aito loppukäyttäjä, eli käyttää aktiivisesti järjestelmää arjessaan. Haasteena tässä on se, paljonko työaikaa loppukäyttäjän kannattaa käyttää järjestelmäkehitykseen osallistumiseen. (Beck 1999, luku 1.10.)

3.2.8 Kehitystyön strategia

Strategia, eli kehitystyön suunnitelma, on tuottaa palvelua mahdollisimman yksinkertaisena. XP-mallin neljästä pääarvosta yksinkertaisuus toistuu myös strategian keskiössä, mutta samalla tämä strategia tukee muitakin mallin arvoista. (Beck 1999, luku 2.17.)

Ensimmäinen arvo on kommunikaatio. Hyvää kommunikaatiota tukee yksinkertaiset tavoitteet, toteutukset ja strategia, jolloin kommunikaatio on merkittävästi helpompaa. (Beck 1999, luku 2.17.)

Yksinkertaiset toteutukset mahdollistavat nopean ja suoraviivaisen toteuttamisen, joka puolestaan madaltaa kynnystä ja helpottaa palautteen antamista. Näin strategia tukee myös XP-mallin kolmatta arvoa, palautetta. (Beck 1999, luku 2.17.)

Yksinkertainen kehittäminen tarkoittaa usein käytännössä sitä, että kehittäjän pitää hyväksyä se, ettei toteutusta ole ajateltu loppuun asti tai kaikilta kulmilta. Esitetty ongelma ratkaistaan nopeasti ja suoraviivaisesti, samalla luotetaan siihen, että mikäli jokin ajattelematon kulma aiheuttaa myöhemmin ongelmia, on toteutus muokattavissa. Tässä apuna toimii mallin neljäs arvo, rohkeus. (Beck 1999, luku 2.17.)

3.2.9 Roolit

XP-mallin esittelemä roolitus sisältää seuraavat tehtävät: kehittäjä, asiakas, testaaja, tarkkailija, valmentaja, konsultti, esimies. Saatesanoina Beck painottaa, ettei roolitus sovi kaikille ja mallia käyttöönottaessa tulisi ennemmin muokata rooleja kuin henkilöitä, mikäli roolit eivät sovi luontevasti. (Beck 1999, luku 3.22.)

Kehittäjä on XP-mallissa se taho, joka tuottaa ja kehittää ohjelmistoa. Luonnollisesti kehittäjä keskittyy ohjelmiston rakentamiseen, mutta toinen keskeinen velvollisuus kehitystyön rinnalle on kommunikointi. Kommunikointi on tarpeen, kun tehdystä kehityksestä pitää viestiä eteenpäin, jotta se

jalkautuu käytäntöön, ja kommunikointi on tarpeen, kun kehittäjä saa kehityspyyntöjä, joita pitää täsmentää, jäsennellä ja pelkistää mahdollisimman yksinkertaiseen ja teknisesti kestävään muotoon. (Beck 1999, luku 3.22.)

Asiakas – eli loppukäyttäjä – on toinen keskeinen osa XP-mallin roolitusta. Kehittäjä tietää miten ohjelmistoa kehitetään, mutta asiakkaan tulisi kertoa mitä kehitetään. Hyvä asiakas osaa tehdä päätöksiä, priorisoida toiveitaan, kuvata tarpeitaan selkeästi ja joustaa tarvittaessa. Taito erottaa tarpeet ja vanhat tottumukset toisistaan on hyvin keskeinen taito, jotta ei kehitetä turhia, tarpeettomia tai huonoja toteutuksia vain siitä syystä, että niin on totuttu tekemään. (Beck 1999, luku 3.22.)

Testaaja koettelee tehdyn kehitystyön laatua ja koeajaa säännöllisesti toiminnallisuuksia, jotta voidaan varmistua ohjelmiston toimivuudesta ja laadusta. Testaaja ei välttämättä ole nimetty henkilö, vaan se voi olla jaettu velvollisuus. (Beck 1999, luku 3.22.)

Tarkkailija pitää silmällä kehitystyön etenemistä ja arvioita. Tarkkailijalla tulisi olla käsitys siitä, kuinka hyvin iteraatio edistyy ja onko iteraatiolla valitut tehtävämäärät realistisia. Samalla tarkkailijalla tulisi olla ajantasainen ja realistinen käsitys isommasta kuvasta, kuten roadmap suunnitelman etenemisestä. Tarkkailija on myös vastuussa kehitystiimin mittareista ja niiden seuraamisesta. (Beck 1999, luku 3.22.)

Valmentaja huolehtii prosessikokonaisuudesta. Hän huolehtii, että kaikki pitäytyvät rooleissaan ja prosesseissa, jotta kokonaisuus toimii. Valmentaja tarkkailee prosessin kulkua ja tehdyn työn laatua, mutta pyrkii välttämään holhoamista ja puuttuu ongelmiin ennemmin välillisesti ohjaillen. Tiimin ei tulisi olla riippuvainen valmentajasta, vaan toimia itsenäisesti ja sallia valmentajan satunnaiset korjaustoimenpiteet silloin kun niille on tarvetta. (Beck 1999, luku 3.22.)

Konsultti on hyvä olla käytettävissä, mikäli tarvitaan syvällisempää teknistä osaamista. XP-mallissa pyritään yksinkertaisiin toteutuksiin ja kehittäjille ei jaeta omia osa-alueita, jolloin kukin kehittäjä osaa hieman kaikkea. Tämä johtaa usein siihen, että kun tarvitaan syväosaamista jostain tietystä osa-alueesta, ei tiimin kehittäjiltä löydy tarvittavia taitoja, ja tarvitaan konsultin apua. Konsulttia ei tulisi vapauttaa tekemään toteutuksia itsenäisesti, vaan kehittäjien tulisi osallistua konsultin tekemään kehitystyöhön ja oppia siitä. (Beck 1999, luku 3.22.)

Esimies tarjoaa tukea ja pyrkii pysymään ajan tasalla. Hän pitää säännöllisiä tarkastuspalavereja ja ylläpitää avointa dialogia. Esimies tarjoaa turvaa, rohkeutta ja itsevarmuutta tiimille. Hän tarkkailee toimintaa ja hänelle selitetään työn etenemisestä, joka on kehittäjille hyödyllistä, sillä tekemänsä kehitystyön kuvaileminen vähemmän teknisesti voi olla hyvä hetki tunnistaa ongelmakohtia. (Beck 1999, luku 3.22.)

3.3 Kanban

Kanban on Leanin ja ketterän kehityksen ideologioihin perustuva toimintamalli kehitystyön hallinnointiin. Kanban koostuu kolmesta pääperiaatteesta: visualisoi, minimoi kerrallaan työstettävien tehtävien määrä ja hallinnoi työn virtausta. (Hammarberg & Sunden 2014, luku 2.)

3.3.1 Visualisointi

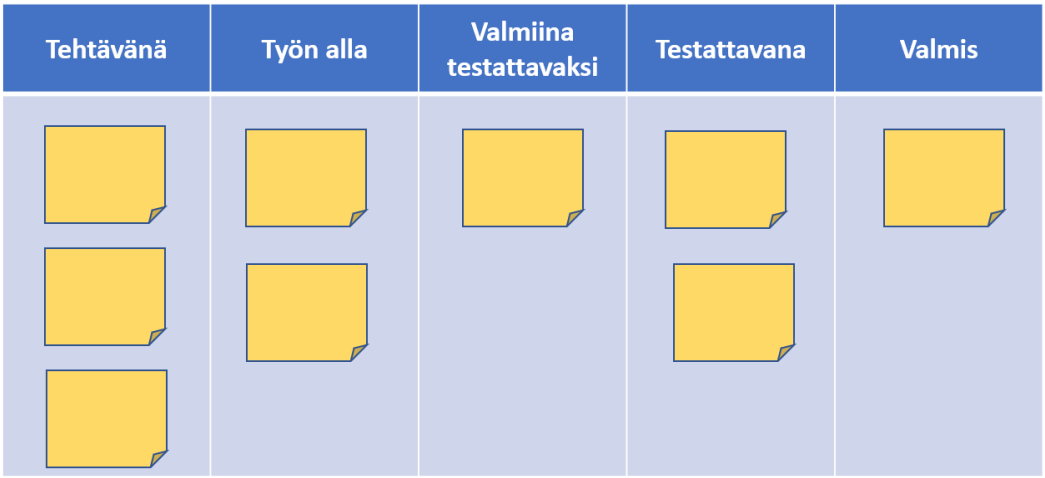
Visualisoinnin keskeisin tavoite on parantaa läpinäkyvyyttä ja jakaa informaatiota selkeällä ja tehokkaalla tavalla. Toimiva visualisointi selkeyttää prosesseja ja vähentää epäselvyyttä, kun vastuut on jaettu ja käytänteet on vaiheistettu ja jaoteltu mahdollisimman yksiselitteisellä tavalla. (Hammarberg & Sunden 2014, luku 3.)

Projektin visualisoinnin tulisi olla projektitiimin nähtävillä jatkuvasti, mutta myös sidosryhmillä tulisi olla siihen pääsy tai näkyvyys. Jotta visualisointia olisi hyödyllistä seurata, tulisi sen olla mahdollisimman ajantasainen, ja jotta se pysyy ajantasaisena, tulee sen päivittämisen olla erittäin helppoa. Lisäksi Kanban-mallissa suositaan uuden kokeilua ja jatkuvaa kehitystä. Näistä syistä visualisointityökalun valinnassa tulee huomioida, että se on mahdollista jakaa sidosryhmille, sitä on helppo ylläpitää, ja sitä on mahdollista muokata rakenteellisesti tarvittaessa. (Hammarberg & Sunden 2014, luku 3.1.)

Etenkin digitaalisissa visualisointityökaluissa on riski, että hyvinkin ylläpidetty informaatio säilötään niin sanotusti piiloon. Piiloon säilömisellä tarkoitetaan, että järjestelmään täytettyä informaatiota pitää osata etsiä ja käyttäjän tulee tietää missä tieto sijaitsee. Erityisesti sidosryhmille tiedon pitäisi olla esillä visuaalisessa muodossa niin, ettei sitä tarvitse erikseen etsiä. Tähän tarvitaan visuaalinen tietotaulu, joka voi olla esimerkiksi mittari, taulukko, Kanban-taulu tai koontinäkymä tärkeimmistä luvuista. (Hammarberg & Sunden 2014, luku 3.1.)

Digitaalisia työkaluja käyttäessä tietotaulu on mahdollista jalkauttaa laajemmin, ja sen täydentämiseen voi käyttää automatiikkaa. Tiedon tulisi olla riittävän selkää, jotta kuka tahansa voi ymmärtää sisällön ilman syvempää perehtymistä. Lisäksi tiedon tulisi olla supistetussa muodossa niin, että taulu on informatiivinen jo nopealla vilkaisulla. On myös tärkeää nostaa tietotaululle vain sidosryhmiä kiinnostavia asioita, ja jättää kehittäjiä kiinnostava taustatieto suosiolla pois tukkimasta näkymää. (Hammarberg & Sunden 2014, luku 3.1.)

Kanban-mallin ydin on Kanban-taulu (kuva 6). Edellä mainittuja tiedon visualisointeja ja tietotauluja on monenlaisia, mutta Kanban-mallissa runkona suositaan itse Kanban-taulua, joka voi toimia myös suoraan sidosryhmille esitettävänä tietotauluna. (Hammarberg & Sunden 2014, luku 3.2.)



Kuva 6. Kanban taulu (mukaillen Hammarberg & Sunden 2014, luku 3.2)

Taulu koostuu rakenteellisesti seuraavista elementeistä: tehtäväkortit, kolumnit ja käyttäjät. Kortit sisältävät kuvauksen tehtävästä työstä, vaihtelevalla tarkkuustasolla tiimistä ja toteutustavasta riippuen. Kortit nimetään kehittäjille, jonka jälkeen ne otetaan työn alle. Kolumnit kuvaavat työn vaiheita ja kortteja siirrellään työn edetessä kolumnista toiseen, yleensä vasemmalta oikealle. Ensimmäinen kolumni vasemmalta tapaa olla listaus tehtävistä töistä ja viimeinen kolumni oikealla puolestaan kuvaa valmiita tehtäviä. Kolumnit välissä kuvaavat kyseiselle tiimille sopivaa tehtävien etenemisen vaiheistusta. Tämä kokonaisuus tarjoaa helposti tiedon siitä, kuka on tekemässä mitäkin ja milloin. (Hammarberg & Sunden 2014, luku 3.2.)

Kanban-taulun kolumnien tulee kuvata tiimin aitoa työn virtausta ja vaiheistusta. Taulua luodessa on hyvä käydä tiimin kesken avoin keskustelu siitä, miten kukakin näkee työn vaiheistuksen, sillä siinä voi olla näkemyksellisiä eroja. Vaiheistuksesta on tärkeää löytää yhteinen näkemys, joka sitten mallinnetaan Kanban-taulun kolumneihin, selkeyttäen samalla prosessia kaikilla osapuolille. Tehtävät työt ovat harvoin niin yhdenmukaisia, että sama prosessi kuvaisi ja tukisi niitä kaikkia. Taululle mallinnetun vaiheistuksen tulisi kuvata valtaosaa tehtävistä ja vaiheiden nimistä kannattaa tehdä mahdollisimman yleispäteviä, niin että ne palvelevat useanlaisia tehtäviä. (Hammarberg & Sunden 2014, luku 3.2.2.)

Kanban-taulu voi olla niin fyysinen, kuin digitaalinenkin. Molemmilla toteutustavoilla on puolensa ja toisinaan voi olla hyödyllistä ylläpitää molempia malleja rinnakkain. Kanban-taulu on kehittäjätiimille luonnollinen kohtaamispaikka, jonka ympärillä on hyvä käydä keskustelua tehdyistä, työn alla olevista, ja tulevista töistä. Kohtaamiset toimistoympäristössä voivat tapahtua luonnostaan, mutta vaihtoehtoisesti Kanban-taulua voi pitää runkona säännöllisille palavereille, niin etänä, kuin paikan päällä. Kanban-mallissa suositaan päivittäisiä pikapalavereita kehittäjätiimin kesken, mielellään

päivän aloituksena. Näiden palaverien olisi hyvä olla enintään varttitunnin mittaisia ja keskittyä agendaltaan Kanban-taulun tapahtumiin yleisellä tasolla. Mikäli yksittäiset aiheet vaativat enemmän keskustelua, on tämä sopiva hetki sopia jatkopalavereista tarvittavalla kokoonpanolla. (Hammarberg & Sunden 2014, luku 3.2.1)

3.3.2 Jonot

Jono-kolumnit auttavat työn virtauksen ylläpitämisessä. Ne helpottavat tehtävien siirtoa ja selkiyttävät vaiheistusta. Jono-vaihetta voi hyödyntää, kun tehtävä läpäisee yhden vaiheen, mutta jää odottamaan seuraavaan siirtymistä. Esimerkiksi tehtävä voidaan analysoida ja valmistella kehitykseen, mutta kehittäjillä on työlistalla vielä tehtäviä, joiden pitää valmistua ennenkö he ottavat uusia. Muita esimerkkejä voi olla työlistalle hyväksytyt tehtävät, kehitystyöt, jotka odottavat testaamista, ja valmiit tehtävät, jotka odottavat tuotantojulkaisua. (Hammarberg & Sunden 2014, luku 3.3.)

Jonoja voidaan hyödyntää, kun tehtävä siirtyy henkilöltä toiselle, mikäli kehitystiimissä useampi tekijä osallistuu tehtävän tekemiseen sen eri vaiheissa. Vaihtoehtoisesti voidaan jono toimia kehittäjälle oman työn jäsentelyssä ja edistymisen viestinnässä muille käyttäjille ja sidosryhmille. (Hammarberg & Sunden 2014, luku 3.3.)

Mikäli kolumnien määrittely jätetään otsikointitasolle, on väärinkäsitysten ja erilaisten näkemysten riski suuri. Vaiheistuksen selkiyttämiseksi on kullekin vaiheelle hyvä luoda kriteeristö, joka tehtäväkorttien tulee täyttää siirtyäkseen vaiheesta toiseen. Kriteeristön voi halutessaan listata esiin suoraan Kanban-taulun kolumneille. Tätä kriteeristöä, niin kuin muitakin taulun osia, tulee kehittää, mikäli käytäntö osoittaa puutteita tai kehityskohtia alkuperäisessä toteutuksessa. (Hammarberg & Sunden 2014, luku 3.3.)

3.3.3 Tehtäväkortit

Tehtäväkorttien on tarkoitus olla kompakteja, visuaalisia ja informatiivisia samaan aikaan. Niiden avulla jokainen tietää mitä kukakin tekee parhaillaan ja mitä on tulossa. Informatiivisuuden suhteen korttien olisi hyvä sisältää tieto kyseisen tehtävän prioriteetti- tai riskitasosta. (Hammarberg & Sunden 2014, luku 4.1.)

Tehtävien priorisointi voi myös periytyä luokittelusta, mikäli kehitystyössä on osa-alueita, jotka ovat selkeästi tärkeämpiä kuin toiset. Tällöin korttien kategorisointiin tai luokitteluun tulee luoda selkeät mallit ja säännöt. (Hammarberg & Sunden 2014, luku 4.1.)

Tehtäväkorttien rakennetta suunnitellessa on tärkeää pitää mielessä, että korttien tulisi olla mahdollisimman yksinkertaisia. Ne ovat osa visuaalista näkymää, jossa suuret informaatiomäärät

rikkovat Kanban-taulun informatiivisuutta ja tieto hukkuu omaan paljouteensa. (Hammarberg & Sunden 2014, luku 4.1.)

Korteilla tulisi olla otsikko ja sen lisäksi kuvaus, jolle on määritelty omat vaatimuksensa. Otsikkotasolla kirjoitettu tehtävä voi olla selkeä luontihetkellä, mutta mikäli selkeä kuvaus uupuu ja asiayhteys tai tekijä vaihtuu välissä, voi kortin tarkoitusperää olla haasteellista ymmärtää. Kuvauksen pitäisi toimia sillä tasolla, että kuka tahansa tiimissä ymmärtää tehtävän tavoitteen. (Hammarberg & Sunden 2014, luku 4.2.)

Tehtävän kuvaukseen on olemassa muutama suosittu malli. Käyttäjätarina on suosittu tapa muotoilla tehtäväkorttien kuvausta. Tässä mallissa käytetään seuraavanlaista lauserakennepohjaa:
”[Roolissa] tahdon, että [toiminnallisuus], jotta [hyödyt]”
esim. ”[Myyjänä] tahdon, että [voin luoda omia raportteja], jotta [voin arvioida omaa tekemistäni]”
(Hammarberg & Sunden 2014, luku 4.2.)

Käyttäjätarina on selkeä ja informatiivinen pohja, mutta ei aina sovi rakenteeltaan tehtävien kuvaukseen. Tällöin vaihtoehtona voi toimia esimerkiksi ”Se missä…”-malli. Tässä tekniikassa kuvaus aloitetaan aina sanoilla ”Se missä…”, jolloin kirjotusasu säilyy kuvauksissa samanlaisena. (Hammarberg & Sunden 2014, luku 4.1.)

Tehtävillä on aina tekijänsä ja tämä on hyvä tuoda ilmi tehtäväkortilla. Kun tekijä on selvillä, on kortille hyvä nostaa tieto tästä, mieluiten visuaalisessa muodossa. Kehitystiimin jäsenten visualisointi hahmoiksi jollain tapaa parantaa Kanban-näkymän luettavuutta merkittävästi ja vähentää tekstin määrää tehtäväkorteilla. (Hammarberg & Sunden 2014, luku 4.2.)

Tehtävillä tapaa olla määräaikoja, eli jokin takaraja milloin tehtävä pitää olla tehtynä. Tämä voi olla hyvin tärkeä tieto ja se tulee tarpeen vaatiessa nostaa osaksi tehtäväkorttia, mielellään sellaisella tavalla, että se erottuu selkeästi, esimerkiksi eri värillä kuin muu teksti. (Hammarberg & Sunden 2014, luku 4.2.)

Kehittäjät päätyvät usein tilanteisiin, että työlistalla olevaa tehtävää ei voi edistää, ennenkö jokin toinen tehtävä on tehty valmiiksi. Näitä riippuvaisuuksia eri tehtävien välillä on hyvä tuoda osaksi tehtäväkortteja, etenkin silloin kun se vaikuttaa suunniteltuun työjärjestykseen. Tarkoituksena on, että esteenä toimivat tehtävät voidaan priorisoida ja nostaa tehtävälistan kärkeen, jotta työ voi jatkua suunnitelman mukaisesti. Esteiden visualisointiin on useampia tapoja, kuten korttien pinoaminen päällekkäin, oma työjono estetyille korteille, tai estettyjen ja esteenä olevien tehtävien merkkaaminen jollain merkillä, kuten värikoodilla. (Hammarberg & Sunden 2014, luku 4.2.)

Erilaiset tyyppiluokat selkiyttävät tehtävienhallintaa. Luokittelua voi hyödyntää tehtävien jakamisessa, priorisoinnissa, aikataulutuksessa ja raportoinnissa. Tehtävien luokittelun voi helposti visualisoida esimerkiksi niin, että kunkin luokan korteilla on omat värinsä. Luokkia voi esimerkiksi olla virhekorjaukset, huolto- ja ylläpitotehtävät, ja uudet toiminnallisuudet. Tarkastelemalla avoimien tai tehtyjen tehtävien määriä luokittain, voidaan saada arvokasta tietoa järjestelmän tilasta. (Hammarberg & Sunden 2014, luku 4.3.)

Kanban-taulun kolumnit kuvaavat työn vaiheistusta selkeästi ja niiden avulla saa hyvän kuvan siitä, mitä on parhaillaan työn alla. Nämä kolumnit eivät kuitenkaan kerro, kuinka pitkään tehtäväkortti on ollut työn alla, kuinka pitkälle sen kanssa on edetty, tai kuinka suuri osa siitä on vielä tekemättä. Tätä varten tehtäväkorteille voi tuoda etenemismittarin, jolla voi viestiä työn kestosta ja etenemisestä. (Hammarberg & Sunden 2014, luku 4.4.)

Kehitystyössä tehtäväkorttien työmäärien arvioiminen tarkalla tasolla ennakkoon voi olla hyvin vaikeaa. Tästä syystä Kanban-mallissa tehtäväkorttien kokoluokittelussa pyritään vertailemaan kortteja keskenään ja käyttämään suurpiirteisempiä aikamääreitä, kuten T-paitojen kokoluokittelua S, M, ja L. (Hammarberg & Sunden 2014, luku 4.5.)

Lyhyesti tiivistettynä, tehtäväkorteilla olisi hyvä olla

- kuvaus, josta saa hyvin selvillä mistä on kyse
- tekijätieto, mielellään visuaalisessa muodossa
- tärkeät päivämäärät, kuten määräajat ja takarajat
- esteiden ja estettyjen tehtävien hallinta
- tehtäväluokat selkeästi tunnistettavassa muodossa
- etenemismittarit, joiden avulla tiedetään kuinka pitkällä mikäkin tehtävä on
- kokoluokittelu, jonka avulla hahmotetaan tehtävien laajuutta ja saadaan kattavampi kokonaiskuva
- dataa strukturoituna niin, että tiedonkeruu on mahdollista kehitystyön analysoimiseksi

3.3.4 Työstettävät tehtävät

Työn alle otetuista tehtävistä käytetään nimitystä työstettävät tehtävät. Työstettävät tehtävät vaiheella viitataan tehtäviin, jotka ovat joko työn alla, tehtynä ja odottaa julkaisua tai tulossa työn alle saman iteraation aikana. (Hammarberg & Sunden 2014, luku 5.)

Kanban-mallissa pyritään minimoimaan kerrallaan työstettävien tehtävien määrää, jotta läpimenoajat pysyvät lyhyenä. Tämän havainnollistamiseen voidaan soveltaa ”Littlen lakia”, joka on John C.V. Littlen kehittämä matemaattinen kaava (kuva 7.). (Hammarberg & Sunden 2014, luku 5.1.)

$$\text{Läpimenoaika} = \frac{\text{Työstettävät tehtävät}}{\text{Suoritusteho}}$$

Kuva 7, Littlen laki (mukaillen Hammarberg & Sunden 2014, luku 5.1.)

Kaava kuvaa sitä, kuinka läpimenoaika korreloi suoraan samanaikaisesti työstettävien tehtävien määrän mukaan. Nopea läpimenoaika mahdollistaa arvontuoton lyhyissä iteraatioissa ja sitä kautta nopean palautteen saamisen ja hyödyntämisen – molemmat ketterän kehityksen periaatteiden mukaisesti. (Hammarberg & Sunden 2014, luku 5.1.)

Useiden tehtävien työstäminen samassa iteraatiossa voi vaatia kehittäjältä poukkoilua aiheiden välillä, eli kontekstin vaihtoa. Kehitystyö tapaa vaatia ajatustyötä, jonka käynnistäminen ja sitä myötä vauhtiin pääseminen vie aina aikaa. Mikäli aihetta joutuu vaihtamaan kesken tehtävän työstämisen, joutuu kehittäjä virittämään taas ajatuksensa uuteen kontekstiin, menettäen aikaa tekemiseltä. Kuva 7 havainnollistaa tämän vaikutusta työn tehokkuuteen. Vasen kolumni kuvaa selkeää työjärjestystä, jossa kehittäjä saa tehdä tehtävän kerrallaan, kun taas oikea kolumni kuvaa hektisempää ympäristöä, jossa työstettävien tehtävien jono on ruuhkautunut ja kehittäjä hyppii tehtävien välillä. (Hammarberg & Sunden 2014, luku 5.2.)

Kuva 7, kontekstin vaihto (mukaillen Hammarberg & Sunden 2014, luku 5.2.)

Pienet tehtävämäärät ja tiiviit julkaisut mahdollistavat toiminnallisuuksien nopean testaamisen ja välittömän palautteen. Mitä aikaisemmin kehittäjät saavat palautetta tekemästään työstä, sitä helpompaa on asiaan reagoida. Kehittäjän on esimerkiksi helppo korjata virhe edeltävällä viikolla tehdystä toiminnallisuudesta yksi päivä sen julkaisun jälkeen, sillä toteutus on vielä tuoreessa muistissa ja virhe tuskin on ehtinyt levitä kovin laajalle. Mikäli kehittäjä puolestaan saa palautteen, että kaksi kuukautta sitten rakennetussa, ja kuukausi sitten julkaistussa toiminnallisuudessa on virhe, on edessä huomattavasti mittavampi työmäärä. Näin isot työstettävien tehtävien määrät aiheuttavat ylimääräistä työtä palautekierron pidentyessä ja prosessi ketteryys kärsii. (Hammarberg & Sunden 2014, luku 5.2)

Kaiken kaikkiaan liialliset tehtävämäärät heikentävät kehittäjien työn laatua, kun läpimenoajat pitenee, kehittäjien keskittyminen hajautuu, palautteen saaminen venyy ja yleinen tyytyväisyys laskee. Näin ollen on selvää, miksi työstettävien tehtävien määrä tulisi pitää minimaalisena ja selkeänä. (Hammarberg & Sunden 2014, luku 5.3.)

3.3.5 Työstettävien tehtävien rajoittaminen

Samanaikaisesti työn alle otettavien tehtävien määrä tulisi rajata selkeästi. Oikean rajauksen löytäminen voi kuitenkin olla vaikeaa ja hyvin tapauskohtaista. Kanban-malli tarjoaa ohjenuoria sopivan rajauksen löytämiseen, tai ainakin etsimiseen.

”Vähempi parempi”. Mitä pienempi rajaus on, sitä lyhyemmät läpimenoajat tehtävillä on ja sitä nopeampaa arvontuotto loppukäyttäjille on. Liian pienessä tehtävämäärässä on kuitenkin riskinsä. Tehtävien suorittaminen vaatii usein kommunikaatiota sidosryhmien kanssa ja tehtävillä on riippuvaisuuksia toisiinsa. Näistä syistä tehtävät jäävät välillä odottamaan jonkun vastausta tai toisen tehtävän valmistumista. Mikäli tehtävämäärät ovat hyvin pieniä, on riski, että kehittäjän kaikki työn alla olevat tehtävät odottavat jotakin, ja kehittäjä joutuu joko odottamaan tai rikkomaan prosessia. (Hammarberg & Sunden 2014, luku 6.1.1.)

”Tehtävät seisovat tai ihmiset seisovat”. Mikäli rajaus on liian korkea, eivät kehittäjät ehdit tehdä kaikkia työn alla olevia tehtäviään ja tehtäväkortit voivat lojua pitkiä aikoja työstettävät tehtävät -vaiheessa. Tällöin työstettävät tehtävät -vaihe ei enää aidosti kuvasta sitä, mitä ollaan parhaillaan tekemässä, eikä siitä voi päätellä, onko tehtävä valmistumassa vai ei. Mikäli rajaus on puolestaan liian matala, päädytään edellä esitettyyn tilanteeseen, jossa kehittäjän kaikki tehtävät odottavat jotakin, eikä kehittäjällä ole työstettävää. (Hammarberg & Sunden 2014, luku 6.1.2.)

Rajauksen tekemiseen on olemassa useita tyylejä. Tehtävien rajauksen voi tehdä yksilö-, tiimi-, tai taulutasolla. Kun rajauksia tehdään Kanban-taulun kolumneihin, on hyvä aloittaa pullonkaulana toimivasta kolumnista. Kun pullonkaulan ylikuormittuminen estetään tarkalla rajauksella, saadaan koko työn virta sovitettua aidon toteuttamisen tahtiin. (Hammarberg & Sunden 2014, luku 6.2.)

3.3.6 Työn virtaus

Jatkuva työn virtaus saavutetaan, kun kehitysprosessi on yhtenäinen vaiheistettu ketju, jossa mahdolliset ongelmat katkaisevat jatkumon ja on käsiteltävä heti alta pois. Tämä ideologia on Toyotan tehtailla alkunsa saaneen Lean mallin perusta, josta se on adaptoitu myös Kanban-malleihin. (Hammarberg & Sunden 2014, luku 7.1.)

Hyvä vaiheistus auttaa tunnistamaan ongelmakohtia, pullonkauloja ja hukkaa. Lean-ajattelun voi kiteyttää kysymyksiin ”mikä estää työtä virtaamasta?” ja ”miten minimoida hukka?”. Prosessin tavoitteena on aina tuottaa arvoa, ja kaikki ne vaiheet, jotka eivät tuota arvoa ovat hukkaa. Hukan tunnistaminen ja kategorisointi Toyotan tehtaalla on erilaista, kuin ohjelmistokehityksessä, mutta Lean-ajattelua on myöhemmin sovitettu myös ohjelmistokehityksen tarpeisiin. (Hammarberg & Sunden 2014, luku 7.1.)

Hukkaa ohjelmistokehityksessä voi jakaa seuraaviin osa-alueisiin (Hammarberg & Sunden 2014, luku 7.1):

- Keskeneräinen työ, eli työ, joka on aloitettu, mutta pysähtynyt jossain kohtaa ennen tuotantoon pääsyä.
- Ylimääräiset toiminnallisuudet. Ohjelmistokehityksessä on hyvin tavanomaista, että suurta osaa ohjelmiston toiminnallisuuksista ei koskaan käytetä. Tämän eliminoimiseksi tarvitaan riittävää ymmärrystä loppukäyttäjien tarpeista, jotta osataan kehittää oikeita asioita.
- Uudelleen oppiminen. Ketterä ohjelmistokehitys perustuu jatkuvaan oppimiseen ja sen kautta toiminnan kehittymiseen, mutta mikäli oppia ei syystä tai toisesta omaksuta prosessissa, voidaan päätyä opettelemaan uudelleen samoja asioita, tai toistamana uudelleen samoja virheitä.
- Työn luovuttaminen, eli tekijän vaihtaminen. Kun tekijä vaihtuu kesken kehityksen, menee tiedonvaihtoon runsaasti resursseja ja siitä huolimatta informaatiota menetetään lähes aina.
- Viivästymiset. Toimitusten viivästyminen pitkittää palautekiertoa ja sitä tuottaa ylimääräistä työtä.
- Kontekstien vaihto, kuten kappaleessa 3.3.4 käytiin läpi, tehtävien välillä vaihtaminen vaatii kehittäjältä aina lyhyen siirtymäajan ja toistuva tehtävien vaihto aiheuttaa näin ollen hukkaa.
- Viat. Huolimattomuus ja huono työn laatu aiheuttavat vikoja, jotka usein tulee korjata nopeasti. Näitä on vaikea ennakoida ja sovittaa prosessiin, ja aiheuttavat siksi haasteellista hukkaa.

3.3.7 Työn virtauksen kehittäminen

Työn virtausta voi parantaa useilla tavoilla. Aikaisemmassa kappaleessa käytiin läpi samanaikaisesti työstettävien tehtävien rajoittamista, joka on keskeinen tapa tehostaa läpimenoaikoja ja taata jatkuva työn virtaus. Lyhyet läpimenoajat mahdollistavat jatkuvan työn virtauksen, ja lyhyiden läpimenojen mahdollistamiseksi tulisi varmistaa, etteivät tehtävät seiso paikallaan työjonoissa. Yksi tapa on edellä mainittu tehtävämäärien rajaaminen, jolloin kehittäjillä pysyy paremmin kontrolli käsiteltävistä tehtävistään. Toinen tapa on visualisoida tehtävät, jotka ovat pysähtyneet syystä tai toisesta, esimerkiksi odottamaan toista tehtävää. (Hammarberg & Sunden 2014, luku 7.2.)

Tehtävien etenemistä ja etenkin niitä tilanteita, joissa tekijä vaihtuu, voi sujuvoittaa hyvillä vaatimusmääritelmillä. Kun tehtäväkortteille on määritelty vaatimukset vaiheistuksen mukaisesti, vältytään epäselvyyksiltä ja voidaan olla varmoja, että tehtävä on valmiina etenemään seuraavaan vaiheeseen. (Hammarberg & Sunden 2014, luku 7.2.)

Työn virtausta seurataan usein Kanban-taululla tehtäväkorttien etenemisen mukaan ja erikokoiset tehtäväkortit voivat vaikeuttaa tai vääristää tätä seurantaa. Tästä syystä tehtäviä tulisi pilkkoa pienempiin osiin, jolloin tehtäväkortit vastaavat suurin piirtein samaa työmäärää. Näin taulun

visuaalisuus, seurattavuus ja ennustettavuus säilyy hyvänä ja työn virtaa on helpompi ylläpitää. (Hammarberg & Sunden 2014, luku 7.2.)

Työn virtauksen suurimpia esteitä on tehtäväkorttien estyminen, eli se, että ne jäävät odottamaan jotakin toista tehtävää. Kehittäjien tulisi pyrkiä ennakoimaan näitä tilanteita, jotta niihin päädyttäisiin mahdollisimman harvoin. Ohjelmistoympäristön ja kehittäjätiimien toimintatapojen tulisi myös tukea tätä ajatusta, eli pyritään välttämään riippuvaisuuksia. Mikäli kehittäjä päätyy kuitenkin estettyyn tilaan, tulisi este pyrkiä purkamaan tai löytämään tapa ohittaa se, sen sijaan, että ottaisi uutta tehtävää työn alle. (Hammarberg & Sunden 2014, luku 7.2.)

Hetkiin kun työn alla olevat tehtävät jäävät jumiin, voi säästää tehtäviä, jotka eivät ole aikakriittisiä, mutta ovat silti tärkeitä. Sillä tavoin kehittäjä voi pysyä aktiivisena ja tuottavana ilman, että tukkii työjonoaan tai poikkeaa prosesseista. (Hammarberg & Sunden 2014, luku 7.2.)

3.4 Scrum

Scrum on projektinhallinnan viitekehys, jonka mallin luojat Ken Schwaber ja Jeff Sutherland ovat tiivistäneet kompaktiin kolmetoista sivuiseen oppaaseen nimeltä ”The Scrum Guide”, eli suomeksi Scrum opas. He kehittivät Scrumin jo 90-luvulla, mutta ensimmäinen versio Scrum oppaasta julkaistiin vuonna 2010, jolloin se saavutti suurempaa suosiota. He ovat päivittäneet opasta pienillä muutoksilla ja viimeisin versio on julkaistu 2020. (Schwaber & Sutherland 2020, 1.)

Scrum oppaan määritelmän mukaisesti, Scrum on kevyt viitekehys, joka auttaa yksilöitä, tiimejä ja organisaatioita tuottamaan arvoa monimutkaisiin ongelmiin adaptiivisten ratkaisujen avulla. Scrum ei ole yksityiskohtainen projektinhallinnan tai kehittämisen malli, vaan viitekehys, joka tarjoaa raamit Scrum teorian mukaiselle kehitystyölle. (Schwaber & Sutherland 2020, 3.)

Scrum perustuu pohjimmiltaan empirismiin ja ”Lean” ajatteluun. Empirismin, eli kokemusajattelun mukaan, tieto pohjautuu kokemukseen. Lean ajattelu painottaa ylimääräisten vaiheiden poistamista ja keskittymistä oleelliseen. (Schwaber & Sutherland 2020, 3.)

Koska Scrum on viitekehys, sovelletaan sitä aina kohdeorganisaation käyttötarpeisiin ja rakenteisiin, muodostaen aina organisaatiotason yksilöllisen Scrum mallin (Rubin 2012, luku 2.1).

3.4.1 Läpinäkyvyys, tarkastelu ja adaptoituminen

Scrum oppaassa painotetaan läpinäkyvyyttä, tarkastelua ja adaptointia. Kehitystyön prosessin tulee olla selkeästi näkyvillä kehittäjien lisäksi myös loppukäyttäjille. Mikäli kehitystyö ja sen tavoitteet eivät ole loppukäyttäjille näkyvillä, voidaan tehdä päätöksiä, jotka ovat riskialttiita tai vähentävät arvontuottoa. (Schwaber & Sutherland 2020, 3.)

Läpinäkyvyys tarjoaa kaikille mahdollisuuden kehitystyön ja prosessien tarkasteluun. Tarkastelua tulee tehdä usein ja perusteellisesti, peilaten tehtyä työtä ja backlogia työn tavoitteisiin. Scrum tapahtumat tarjoavat tälle tarkastelulle luontevat tilaisuudet ja keskustelufoorumit. (Schwaber & Sutherland 2020, 4.)

Ketterälle kehitykselle ominaiseen tapaan, nopea adaptoituminen on välttämätöntä, mikäli tarkastelussa havaitaan sille tarvetta. Mikäli prosessia ei noudateta tai tuotteen kehitys etenee väärään suuntaan, tulee korjausliikkeitä tehdä mahdollisimman nopeasti. (Schwaber & Sutherland 2020, 4.)

3.4.2 Arvot

Aidosti toimiakseen, täytyy Scrum tiimin noudattaa teorian mukaan seuraavia arvoja: sitoutuminen, rohkeus, keskittyminen, avoimuus, ja kunnioitus. (Schwaber & Sutherland 2020, 4.)

Sitoutunut tiimi luo kykenevän työilmapiirin, jossa voi luottaa siihen, että arvoa tuotetaan yhdessä jokaisessa iteraatiossa. Tiimin tulee sitoutua toimimaan prosessin mukaan ja tavoittelemaan yhdessä sovittuja päämääriä tukien toisiaan. (Ripley & Miller 2020, luku 3.1).

Rohkeutta tarvitaan, jotta pystytään tekemään vaikeita päätöksiä, keskustelemaan avoimesti ongelmista ja sitoutumaan prosessiin häiriötekijöistä huolimatta. Rohkeus on myös tarpeen uusien ideoiden avoimessa vastaanottamisessa, varsinkin jos vanhoja ratkaisuja ja toimintatapoja tulee korvata tai hylätä näiden myötä. (Ripley & Miller 2020, luku 3.1).

Keskittyminen on luovan työn edellytyksiä. Kehittäjille tulisi tarjota tilaa keskittyä sprintin sisältöön ja omiin tehtäviin ilman liiallisia häiriötekijöitä. Tehokkuus ja työn laatu paranee, kun yksilöt saavat uppoutua tekemiseen. (Ripley & Miller 2020, luku 3.1).

Avoimuus tarjoaa läpinäkyvyyttä, joka on yksi niistä syistä miksi Scrum toimii. Mikäli prosessista puutuu läpinäkyvyys, ei tiimi saa aitoa palautetta eikä se pysty kehittymään, adaptoitumaan tilanteeseen tai toimimaan yhdessä. (Ripley & Miller 2020, luku 3.1). Avoimuutta vaaditaan niin sidosryhmiltä tarpeiden suhteen, kuin kehitystiimiltä ongelmien ja haasteiden osalta. (Schwaber & Sutherland 2020, 4).

Kunnioitus mahdollistaa turvallisen ympäristön, jossa kukin voi olla avoin ajatuksistaan ja pyytää apua tarvittaessa. Hyvin toimivassa Scrum tiimissä jäsenet kunnioittavat toinen toisiaan, luoden perustan rehelliselle keskustelulle ja turvalliselle ilmapiirille, jossa haasteet ovat miellyttäviä kohdata yhdessä. (Ripley & Miller 2020, luku 3.1).

3.4.3 Roolit

Scrum tiimi koostuu kolmesta eri roolista: tuoteomistaja, scrummaster ja kehittäjät (Schwaber & Sutherland 2020, 5–6).

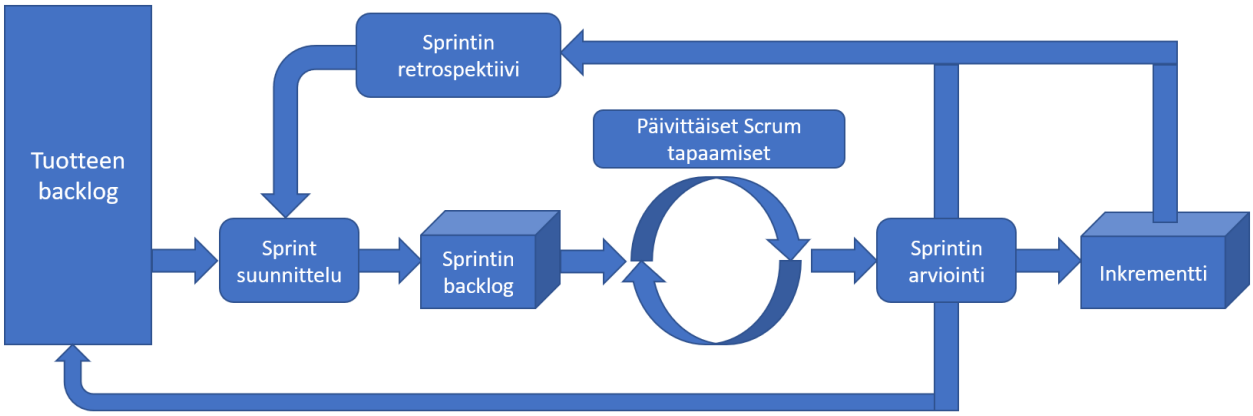
Tuoteomistaja on vastuussa tuotteen maksimaalisen arvon tuottamisesta. Hän vastaa backlogin ylläpitämisestä, tuotteen tavoitteista ja näiden selkeästä viestimisestä. (Schwaber & Sutherland 2020, 5–6). Tuoteomistaja huolehtii myös avoimesta keskustelusta sidosryhmien kanssa ja muokkaa tavoitteita palautteen ja markkinan pohjalta. (Ripley & Miller 2020, luku 4.1).

Scrummaster vastaa siitä, että tiimi ymmärtää ja toteuttaa prosesseja Scrum periaatteiden mukaisesti. Hän perehdyttää ja valmentaa jäseniä toimimaan itsenäisesti, auttaa keskittymään sprintin tehtävien valmistumiseen ja huolehtii, että Scrum tapahtumat järjestetään ja ne pysyvät tuottavina ja positiivisina. (Schwaber & Sutherland 2020, 6). Tiimin hallitessa prosessit, jää scrummasterille hyvin vähän tehtäviä. (Guckenheimer & Loje 2012, luku 4).

Kehittäjät fokusoivat tuottamaan arvoa tehtävien kautta. He valitsevat tehtävät sprinttiin, vastaavat niiden laadukkaasta toteutuksesta ja päivittävät kehityksen sovitusti. (Schwaber & Sutherland 2020, 5.)4.7

3.4.4 Sprintit ja tapahtumat

Sprintit ovat Scrumin ydin. Kaikki prosessin vaiheet nivotaan sprintteihin, jotka ovat yhdestä neljään viikkoa pitkiä ja sisältävät kaikki Scrum tapahtumat (kuva 8). Uusi sprintti alkaa aina edeltävän päätyttyä. Sprintti itsessään nähdään yhtenä tapahtumana, joka pitää sisällään neljä pienempää tapahtumaa: sprintin suunnittelu, päivittäiset scrum tapaamiset, sprintin arviointi ja sprintin retrospektiivi. (Schwaber & Sutherland 2020, 8.)



Kuva 8, Scrum viitekehys (mukaillen Scrum.org).

Ensimmäinen sprintin alainen tapahtuma on sprintin suunnittelu. Sprintin suunnittelussa arvioidaan, miten tuleva sprintti tuottaa arvoa, mitä kaikkea tulevassa sprintissä saadaan tehtyä, ja miten tehtävät saadaan tehtyä. (Schwaber & Sutherland 2020, 8.)

Arvontuottoa sidosryhmille arvioidaan tiimin toimesta ja niitä konkretisoidaan luomalla sprintille tavoite. (Schwaber & Sutherland 2020, 8.)

Sprinttiin valitaan tehtävät tuotteen backlogista ja niitä muotoillaan vielä tarpeen mukaan yhdessä. Näin muodostetaan sprintin backlog. Tässä vaiheessa on haastavaa arvioida, kuinka monta tehtävää ehtii tehdä ennen sprintin päättymistä, mutta ennustaminen helpottuu kokemuksen myötä. (Schwaber & Sutherland 2020, 8.)

Viimeiseksi kehittäjät voivat arvioida miten tehtävät saadaan tehtyä. Tässä tehdään backlog tehtävien jakoa, pilkkomista, aikataulutusta ja muulla tarkempaa suunnittelua. (Schwaber & Sutherland 2020, 8.)

Toinen tapahtuma on päivittäiset scrum tapaamiset. Nämä tapaamiset ovat lyhyitä, noin 15 minuutin mittaisia Scrum tiimin tapaamisia, joissa tarkastellaan, miten Sprint etenee ja tehdään tarvittaessa muutoksia. (Schwaber & Sutherland 2020, 9.) Päivittäisen Scrum tapaamisen ei ole tarkoitus olla tilannekatsaus tehtyihin töihin, vaan katsaus seuraavaksi tehtäviin töihin. Tehdyt työt on jo viestitty Scrum backlogin reaaliaikaisella päivittämisellä ja näin ollen voidaan keskittyä tulevaan (Guckenheimer & Loje 2012, luku 4.) Mikäli tulevien tehtävien osalta vuorovaikutusta vaaditaan enemmän, voivat jäsenet sopia erillisiä palavereja tarkemmalla agendalla. (Schwaber & Sutherland 2020, 9.)

Sprintin arvioinnissa Scrum tiimi esittelee tärkeimmille sidosryhmille sprintin tuloksia, ja niitä arvioidaan sprintin ja tuotteen tavoitteisiin peilaten. Tämän jälkeen voidaan keskustella seuraavista vaiheista. (Schwaber & Sutherland 2020, 9.)

Scrum retrospektiivissä Scrum tiimi kokoontuu vielä arvioimaan tapoja kehittää tulevia sprinttejä niin laadun, kuin tuottavuuden suhteen. Juuri toteutunutta sprinttiä arvioidaan ja sieltä pyritään löytämään onnistumisia, haasteita, ja kehityskohtia. (Schwaber & Sutherland 2020, 10.)

3.4.5 Tavoitteet

Scrumissa tulisi määritellä eri osille tavoitteet, jotka ovat läpinäkyvyyden edistämiseen ja jatkuvaan arviointiin luotu selkeyttämisen työkalu. Backlogin suhteen tavoitteina toimii tuotteen tavoite, eli perimmäinen tarkoitus tuotteen olemassaololle.

Sprintin backlogin suhteen tavoitteena toimii sprintin tavoite, joka tulisi kyetä viestimään selkeästi. Sprintin tavoitteen tulisi kuvata miksi kyseistä sprinttiä ollaan tekemässä.

Yksittäisen inkrementin suhteen tavoite on vastata valmiin määritelmää.

3.4.6 Tuotteen Backlog

Backlog on paikka mihin kaikki Scrum tiimin tehtävät on kerätty yhteen ja kuvattu mahdollisimman tarkasti. Sieltä Scrum tiimi ottaa tehtäviä työn alle ja päivittää tietoa niiden kehityksestä. Tehtävien kuvausta, tilaa ja kokoa päivitetään tilanteen muuttuessa suoraan backlogiin. (Schwaber & Sutherland 2020, 10.)

Backlogin kaikkien tehtävien tulisi tukea tuotteen tavoitetta. Tämä tulee olla määriteltynä, ja tehtäviä kirjatessa on hyvä arvioida, palvelevatko ne tätä tarkoitusta. (Schwaber & Sutherland 2020, 10–11.)

Tehtävät on hyvä jaotella tehtävätyyppeihin kehitystyön tarpeiden mukaan. Rubin (2012, luku 6.1) esittää tehtävien jaotteluun seuraavaa jakoa:

- Toiminnallisuus, eli täysin uusi ominaisuus, jollaista ohjelmistossa ei vielä ole.
- Muutos, eli olemassa olevan ominaisuuden tai toiminnallisuuden päivittäminen paremmaksi.
- Viankorjaus, eli jonkin toimimattoman tai väärin toimivan ominaisuuden korjaaminen.
- Tekninen kehitys, kuten ohjelmiston versiopäivitykset tai muu teknisen toteutuksen laadun parantamiseen tai ylläpitämiseen liittyvä työ.
- Tiedonkeruu, eli uuden oppimista tai vaihtoehtoisiin ratkaisuihin syventymistä, vertailua ja tiedonkeruuta.

3.4.7 Sprintin Backlog

Tuotteen backlogilla olevat tehtävät voivat olla suuria, eivätkä välttämättä kovin yksityiskohtaisesti kuvattuja. Tästä syystä sprinttiä varten suuria tehtäviä tulisi pilkkoa pienempiin osiin ja täsmentää kuvauksia, jotta sprintille saadaan realistisen kokoisia ja täsmällisesti kuvattuja tehtäviä, jotka voi saada sprintin aikana tehtyä. (Rubin 2012, luku 6.2.)

Kun sprintti on suunniteltu, siirretään valitut tehtävät sprintin tehtäviin. Sprint backlogin tulisi vastata siihen, mitä arvoa sprintti on tuottamassa, mistä tehtävistä se koostuu ja miten nämä toteutetaan. Sprintin backlogin on tarkoitus olla visuaalinen ja reaaliaikainen työkalu, joka auttaa kehittäjiä keskittymään sprintin tavoitteeseen. (Schwaber & Sutherland 2020, 11.)

3.4.8 Inkrementti ja valmiin määritelmä

Inkrementti on arvoa tuottava pienempi kokonaisuus ja konkreettinen askel kohti tuotteen tavoitetta. Sprintti voi sisältää useamman inkrementin ja niitä voidaan julkaista jo sprintin aikana, jotta arvoa saadaan tuotettua mahdollisimman nopeasti. (Schwaber & Sutherland 2020, 12.)

Jotta tehtävä muuttuu arvoa tuottavaksi inkrementiksi, tulee sen vastata valmiin määritelmää. Valmiin määritelmä on Scrum tiimin muodostama kriteeristö sille, mitä pitää olla tehtynä, jotta tehtävää voidaan kutsua valmiiksi. Määritelmä voi olla hyvin tiukka, jolloin toteutukselta voidaan vaatia esimerkiksi kattavia testejä ja dokumentaatiota, tai se voi olla hyvin löyhä, jolloin testikäyttöön julkaistavaa voi kutsua valmiiksi. (Rubin 2012, luku 4.)

3.5 Vertailu XP, Scrum, Kanban

Tässä osiossa kerätään yhteen kehitystyön prosessihallinnan keskeisiä teemoja ja vertaillaan XP-, Scrum- ja Kanban-malleja näiden teemojen valossa. Vaikka kaikki ovat ketterän kehityksen periaatteisiin ja arvoihin pohjautuvia malleja, on mallien väleillä paikoin merkittäviäkin eroja.

Taulukko 1. XP, Scrum, ja Kanban vertailutaulukko

	XP	Scrum	Kanban
Määritellyt roolit	X	X	
Aikaikkunaan sidotut iteraatiot	X	X	
Visualisoinnit		X	X
Määritellyt tapaamiset	X	X	
Keskittyy lyhyeen läpimenoon			X
Keskittyy arvontuottoon ja ongelmanratkaisuun		X	
Keskittyy laatuun ja yksinkertaisuuteen	X		
Sisältää käytännön tekniikoita	X		X
Tehtävät pisteytetään	X	X	X
Tehtävien määrää rajataan			X
Tuotetaan toimintoja vain nykyiseen tarpeeseen	X		X
Loppukäyttäjät ja palaute keskiössä	X	X	X
Hyvin skaalautuva	X	X	
Ottaa kantaa siihen, miten työ tehdään	X		
Tehtävät priorisoidaan systemaattisesti	X		

3.5.1 Roolit

Scrum tiimiin kuuluisi kolme eri roolia: tuoteomistaja, scrummaster ja kehittäjät (Schwaber & Sutherland 2020, 5–6). Nämä ovat tarkemmin avattuna kappaleessa 3.5.3.

XP-mallissa esitellään rooleina kehittäjä, asiakas, testaaja, tarkkailija, valmentaja, konsultti ja esimies. XP:n lähestyminen ottaa siis laajemmin kantaa kokonaisuuteen roolittamalla myös henkilöesimiehet, mahdolliset ulkoiset konsultit ja loppukäyttäjät osaksi kokonaisuutta. (Beck 1999, luku 3.22.)

Kanban on XP- ja Scrum-malleja suurpiirteisempi, eikä sisällä yhtä tarkkoja määrityksiä. Kanban on tarkoitettu käyttöönotettavaksi olemassa olevien roolitusten ja prosessien päälle tai rinnalle, muuttamatta esimerkiksi yksilöiden rooleja. (Hammarberg & Sunden 2014, luku 2.1.) Näin ollen Kanban malli ei sisällä roolimäärityksiä, joskin suosittelee käyttämään tiimille luontaista roolitusta (Brechner 2015, luku 5.1).

3.5.2 Tapahtumat

Scrum mallissa esitellään viisi tapahtumaa, joista ylätasolla toimii kahdesta yhdestä neljään viikkoa pitkä sprintti, joka pitää sisällään neljä pienempää tapahtumaa: sprintin suunnittelu, päivittäiset scrum tapaamiset, sprintin arviointi ja scrum retrospektiivi (Schwaber & Sutherland 2020, 8–10).

XP malli puolestaan esittelee viikoittain ja kvartaaleittain toistuvat tapaamiset, joista viikkotaso keskittyy tehtäviin ja iteraatioihin, kun taas kvartaalipalaverit keskittyvät isompaan kuvaan, ongelmiin ja teemoihin. (Beck & Andres 2004, luku 1.7.)

Kuten jo mainittu, Kanban-malli ei sisällä kovin tarkkoja määrityksiä siitä miten ja milloin asioita tulisi tehdä, eikä näin ollen sisällä määriteltyjä tapaamisia. Edes iteraatiot eivät ole Kanban-malliin määriteltyjä, vaan Kanbanin voi nähdä jatkuvana työn virtauksena. (Hammarberg & Sunden 2014, luku 12.3). Malliin suositellaan kuitenkin adaptoimaan aikamääritteitä, kuten sprintin kaltaisia iteraatioita ja säännöllisiä tapaamisia, kuten päivittäiset scrum tapaamiset (Hammarberg & Sunden 2014, luku 7.3).

3.5.3 Backlog

XP-mallissa ei puhuta backlogista. Siinä esitellään lyhyesti myös muissa ketterän kehityksen malleissa suosittuja käyttäjätarinoita ja ohjeita niiden muotoiluun. Käyttäjätarinoiden kirjoittaminen on kirjattu asiakkaan vastuulle ja niille on määritelty tietty rakenne. Käyttäjätarinat kirjataan korteille, joista valikoidaan sopiva määrä seuraavaan iteraatioon tehtäväksi ja loput säilötään seuraavaa iteraatiota varten. (Beck & Fowler 2000, luku 10.1).

Kanban mallin keskiössä on Kanban-taulu, joka on hyvin tunnettu ja suosittu tapa visualisoida ja jäsennellä backlogia (Hammarberg & Sunden 2014, luku 3.2). Aikaisemmin mainittu luetteloinnisovellus Trello mahdollistaa Kanban-mallin mukaisen taulun luonnin ja kevyen ylläpidon paikkariippumattomasti, joka sopii kohdeorganisaation kulttuuriin, jossa etätyöskentely on arkea.

Scrum oppaassa esitellään tuotteen backlogia teorian tasolla, ja painotetaan, että sen tulisi olla hyvin näkyvillä. Backlogia hyödynnetään tapaamisten keskustelun perustana ja sprinttien suunnittelu tehdään poimimalla backlogilta tehtäviä. (Schwaber & Sutherland 2020, 8.)

3.5.4 Keskeiset mittarit

Kuten XP, Scrum, ja Kanban vertailutaulukosta (taulukko 1) näkee, useista samankaltaisuuksista ja päällekkäisyyksistä huolimatta mallit eroavat toisistaan täysin tarkasteltavien mittarien osalta.

XP-mallin keskiössä on laadukas työnjälki ja nopean palautteen mahdollistava ketterä iteratiivinen kehitystyö. Näin ollen keskeisten mittareiden olisi hyvä keskittyä laadun varmistamiseen ja ideoiden läpimenon mittaamiseen, esimerkiksi mittaamalla virheiden määrää julkaisuissa ja tehtävien läpimenoaikaa ideasta tuotantoon. (Beck & Andres 2004, luku 1.10).

Scrum oppaassa ei mainita keskeisiä mittareita työn mittaamiseen, joten scrum tiimeissä käytetään erinäisiä tarpeeseen soveltuvia mittareita, jotka usein esiintyvät muissakin ketterän kehityksen malleissa. Koska scrum keskittyy arvontuottoon, pyritään tuotettua arvoa mittaamaan, jos vain mahdollista. Tuotetun arvon lisäksi on hyvä mitata kehitystyön nopeutta, jotta voidaan tarkastella työn tehokkuutta ja ennustaa sen pohjalta tulevaa kehitystä. (Sutherland & Coplien 2019, luku 3.50.)

Scrumissa sprintit ja tehtävät suunnitellaan siltä pohjalta, että ne palvelevat tuotteen tavoitetta. Tehdyn kehitystyön suhdetta suurpiirteisempään tuotteen tavoitteeseen tulisi pystyä mittamaan ja esittämään. Tämä auttaa keskittymään kestävään kehitystyöhön, joka palvelee pitkän aikavälin tavoitetta ja tuo työlle tarkoitusta. (Ripley & Miller 2020, luku 4.6).

Kanban-mallissa mitataan läpimenoaikoja, kuten kappaleessa 3.4.4 käydään läpi. Tavoitteena on mahdollisimman lyhyet läpimenoajat, ja tätä pyritään tehostamaan minimoimalla kerrallaan työstettävien tehtävien määrää. (Hammarberg & Sunden 2014, luku 5.1.) Läpimenoajan lisäksi Kanban-mallissa mitataan kehitysaikaa, joka vastaa muutoin läpimenoaikaa, mutta laskenta alkaa vasta kun tehtävä otetaan työn alle. Tässä mittarissa poissuljetaan siis tehtävän jonotusaika, ja keskitytään vain siihen kauanko aito toteuttaminen kestää, tarkoituksena tunnistaa johtuuko pitkät läpimenoajat hitaasta kehitystyöstä vai pitkistä tehtäväjonoista. (Hammarberg & Sunden 2014, luku 11.1.)

3.5.5 Palaute

Ketterä kehitys, jonka periaatteita kaikki esitellyt mallit noudattavat, on ennen kaikkea palaute- ja loppukäyttäjälähtöinen ideologia. Jatkuvan tai tiiviin aikavälin arvontuoton päätarkoitus on lyhentää palautesykliä, jotta reagointi palautteeseen on nopeaa ja toimintaa peilataan aina nopeasti saatuun palautteeseen. (Martin 2019, luku 1.4.)

Yksi XP-mallin viidestä pääarvosta on palaute. Koska kehitysvaiheessa olevasta toteutuksesta on vaikea saada palautetta, suositellaan XP-mallissa lyhyitä iteraatioita ja nopeaa tuotantoon siirtymistä. Koska muutosten jalkauttaminen on nykytyökaluilla helppoa, voi toteutuksesta julkaista tuotantoon viimeistelemättömän version, ja päivittää viimeistelyt palautteen pohjalta korjattuina seuraavissa iteraatioissa. (Beck 1999, luku 7.)

Kanban-mallissa vastaavasti puhutaan lyhyen läpimenon hyödyistä, koska se mahdollistaa nopean tuotantotestaamisen ja sen myötä nopean palautteen, lyhentäen palautesykliä. Mitä aikaisemmin palaute saadaan, sitä helpompaa siihen on reagoida ja tehdä tarvittavat korjaukset. Nopean palautesyklin mahdollistamiseksi Kanban-mallissa painotetaan työstettävien tehtävien määrää, kuten kappaleessa 3.4.4 on tarkemmin kuvattu. (Hammarberg & Sunden 2014, luku 5.1.)

Scrum oppaan mukaan prosessin täytyy olla nähtävillä tekijöiden lisäksi sidosryhmille, joihin tuotos vaikuttaa. Mutten osalliset eivät voi aidosti tarkastella tehtävää työtä, ja tarkastelu ilman läpinäkyvyyttä johtaa hukkaan ja väärinkäsityksiin. Scrum-mallissa palautetta pyritään saamaan järjestämällä säännöllisiä palavereja sidosryhmien kanssa ja huolehtimalla kehitysprosessin läpinäkyvyydestä. (Schwaber & Sutherland 2020, 3.)

4 Palvelumuotoilu, viestintä, ja loppukäyttäjien osallistaminen

Tämän opinnäytetyön tavoitteena oli luoda malli asiakkuudenhallintajärjestelmän jatkuvaan kehittämiseen loppukäyttäjälähtöisesti. Ketterän kehityksen ja prosessinhallintamallien osuus tarjoaa hyvän loppukäyttäjälähtöisen pohjan kehitystyöhön.

Tässä kappaleessa haetaan vielä syvempää ymmärrystä loppukäyttäjien osallistamisesta kehitystyöhön, käsittelemällä palvelumuotoilun ja viestinnän teoriaa.

Palvelumuotoilussa käsitellään sitä, miten kehitystyöstä saadaan läpinäkyvää ja ymmärrettävää loppukäyttäjille, miten pää- ja loppukäyttäjät voivat yhdessä ohjata kehitystyön suuntaa ja miten pääkäyttäjät voivat syventää ymmärrystään loppukäyttäjien kokemuksista.

Viestinnän osalta käsitellään viestinnän tyylejä, palautetta ja viestintäkanavia.

4.1 Palvelumuotoilu

Service Design, eli palvelumuotoilu on loppukäyttäjälähtöisyyteen ja kevyisiin prosesseihin keskittyvä jatkuvan kehittämisen malli (Lawrence, Stickdorn, Hormess, Schneider 2019, luku 0). Palvelumuotoilu on jonkin palvelun muotoilua asiakasymmärrystä ja -näkökulmaa painottaen, tavoitteena parantaa käyttäjäkokemusta ja sitä kautta edesauttaa palvelun tai tuotteen tavoitteita (Reason, Løvlie, Flu 2015, luku 0).

Jotta kompleksista kokonaisuutta pystytään hallinnoimaan ja käsittelemään, on se pilkottava pienempiin osakokonaisuuksiin, jotka käsitellään itsenäisinä osina (Tuulaniemi 2011,30). Palvelumuotoilun keskeinen ajatus on, että nämä osakokonaisuudet käsitellään yhteiskehittämisen menetelmillä osakokonaisuuteen liittyvien sidosryhmien kanssa (Tuulaniemi 2011,53.)

Käyttäjien mielipiteeseen ei vaikuta pelkästään palvelun toiminnallisuudet ja niiden käytettävyys. Loppukäyttäjäkokemukseen vaikuttaa perustoiminnallisuuksien lisäksi vahvasti informaation kulku, empaattisuus, osallistaminen päätöksentekoon ja tunne siitä, että käyttäjän tarpeista huolehditaan (Lawrance ym. 2019, luku 1.1.)

Käyttäjätyytyväisyyden seuraaminen on merkittävä osa käyttäjälähtöistä kehitystyötä ja hyvin suunniteltuna auttavat tunnistamaan ongelmakohdat nopeasti. (Lawrance ym. 2019, luku 1.1.)

Palvelumuotoilu nähdään usein työpajana tai projektina, mutta oikein toteutettuna palvelumuotoilu on jatkuvan kehittämisen malli, kokoelma työkaluja ja prosessin perusta. (Lawrance ym. 2019, luku 2.1.)

Loppukäyttäjien tarpeiden ja motiivien ymmärtäminen on yksi palvelumuotoilun tavoitteista. Loppukäyttäjien osallistaminen ja yhteiskehittämisen menetelmät edistävät tätä ymmärrystä ja parantavat palvelun arvontuottoa. Palvelumuotoilun sitouttaa kaikkia osapuolia palvelun kehittämiseen ja arvon luontiin. Vaikka tavoitteena on mahdollistaa osallistuminen kaikille sidosryhmille, ei tämä tarkoita, että kaikki osapuolet osallistuisivat päätöksentekoon. Palvelumuotoilun yhteiskehittämisen tavoitteena on saavuttaa mahdollisimman kattava ja laaja-alainen ymmärrys kehitettävästä asiasta, jonka pohjalta kehitystyötä voidaan ohjata. (Tuulaniemi 2011,53.)

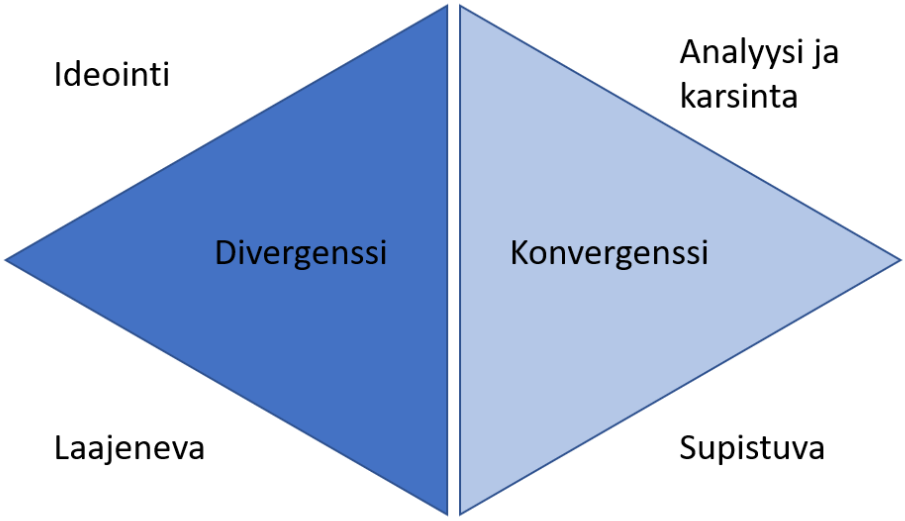
4.1.1 Iteratiivinen ja inkrementaalinen

Palvelumuotoilun kehitysmenetelmät ovat inkrementaalisia ja iteratiivisia. Inkrementaalinen tarkoittaa vähittäin kasvavaa ja kehittämismenetelmän näkökulmasta inkrementaalisella tavalla tarkoitetaan sitä, että suurempi asiakokonaisuus pilkotaan pienempiin osiin, joiden kautta kehitystyötä tehdään osa osalta.
Iteratiivisuus puolestaan tarkoittaa toistoa. Iteratiivinen kehitysmenetelmä tarkoittaa sitä, ettei pyritä kerralla julkaisemaan täydellistä toteutusta, vaan ensimmäinen versio pyritään tekemään mahdollisimman nopeasti, ja sitä kehitetään iteroinnin, eli toistuvan kehityksen kautta.

Iteratiivinen ja inkrementaalinen kehitysmenetelmä tarkoittaa käytännössä asiakokonaisuuden pilkkomista pienempiin osakokonaisuuksiin, joita kehitetään iteraation kautta. (Tuulaniemi 2011,50.)

4.1.2 Divergenssi ja konvergenssi

Luova ongelmanratkaisu voidaan vaiheistaa ja jakaa kahteen osioon: divergenssiin ja konvergenssiin (kuva 9). Divergenssi – eli hajaantuminen tai laajeneminen – palvelumuotoilussa tarkoittaa luovaa ideointia, vapaata asioiden yhdistelyä ja ratkaisujen tavoittelua. Konvergenssi – eli supistaminen – toimii vastapainona divergenssille. Konvergenssi tarkoittaa analyyttistä päättelyä ja karsintaa. Divergenssin kautta luotuja ideoita validoidaan konvergenssin kautta ja näistä muodostuu iteratiivinen validoinnin ja kehittämisen prosessi. (Tuulaniemi 2011,50.)



Kuva 9 – Divergenssi ja konvergenssi (mukaillen Tuulaniemi 2011,50.)

4.1.3 Visualisointi

Asioiden ja käsitteiden muuntaminen näkemällä havaittavaan muotoon kutsutaan visualisoimiseksi. Palvelukehityksessä visualisoinnin on tärkeää tukea ihmisten ajatusmallien muodostumista ja edistää sitä kautta ymmärrystä. Visualisointi toimii myös yhteiskehittämisen työkaluna, sillä se auttaa suunnittelu- ja sidosryhmiä hahmottamaan käsitteet ja prosessit ja toimii näin keskustelun tukena. (Tuulaniemi 2011,50–51.)

4.1.4 Palvelumuotoiluprosessi

Palvelumuotoilua voidaan käsitellä viisivaiheisena prosessina (kuva 10). Vaiheet ovat määrittely, tutkimus, suunnittelu, tuotanto ja arviointi. (Tuulaniemi 2011,55.)

Kuva 10. Palvelumuotoiluprosessi (Tuulaniemi 2011,53.)

Prosessi on toistuva loogisesti etenevien toimintojen sarja, joka selkeyttää työtä, vapauttaa aikaa ja säästää resursseja. (Tuulaniemi 2011,53.)

Ihanteellista olisi, että kaikki sidosryhmät ovat mukana palvelumuotoiluprosessin kaikissa vaiheissa. Tämän mahdollistamiseksi olisi prosessissa hyvä olla mukana palvelumuotoilija, joka mahdollistaa osallistumisen ja huolehtii vuorovaikutuksen ylläpitämisestä. (Tuulaniemi 2011,56.)

4.1.5 Määrälliset ja laadulliset tutkimukset

Määrälliset ja laadulliset tutkimusmenetelmät täydentävät toisiaan, mutta palvelukehityksessä kaivattua asiakasymmärrystä saa kasvatettua paremmin laadullisilla menetelmillä. (Tuulaniemi 2011,61.)

Markkinatutkimuksen menetelmillä, kuten kvantitatiivisilla tyytyväisyyskyselyillä, voidaan helposti mitata suurpiiteistä informaatiota esimerkiksi tyytyväisyydestä, kehityskohteista tai käyttäjätottumuksista. Käyttäjäpaneeleilla saadaan puolestaan kerättyä syvällisempiä mielipiteitä, ideoita ja toiveita. Konseptitestauksella voidaan tarkastella käyttäjien reaktioita uusiin konsepteihin, ideoihin tai toimintamalleihin. (Tuulaniemi 2011,62–63.)

Tiedonkeruumenetelmiä asiakasymmärrykseen:

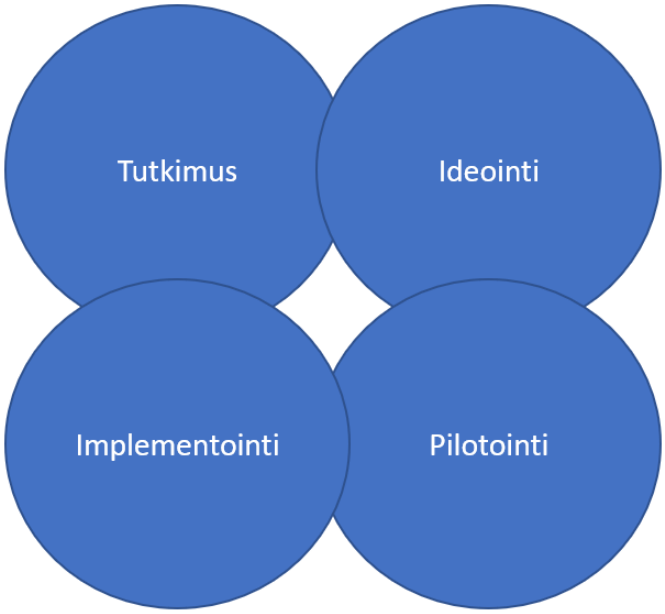
- Olemassa olevat tausta-aineistot ja lähteet
- Haastattelut ja kyselytutkimukset
- Havainnointi, joko passiivisesti tai osallistumalla kohderyhmän arkeen
- Kohderyhmän osallistaminen suunnittelutyöhön
- Itsedokumentoiti menetelmät
- Etnografiset tutkimukset

Etnografisten tutkimusmenetelmien tarkoitus on tutkia käyttäjiä luonnollisissa olosuhteissa. Laadullista tutkimusta tehdessä tutkijan tulee kyetä lähestymään asiaa empaattisesti. Syvällinen ja aito kiinnostus loppukäyttäjiä kohtaan on keskeistä, kun tutkitaan ihmisten arvoja, tarpeita ja toiveita. (Tuulaniemi 2011,62–63.)

4.2 Palvelumuotoilun menetelmiä

Palvelumuotoiluun kuuluu runsaasti erilaisia käytännön malleja ja työkaluja muun muassa tiedonkeruuseen, ideointiin, pilotointiin, implementointiin ja fasilitointiin (Lawrence ym. 2019, luku 2.3.)

Menetelmät ovat jaettavissa neljään ydinosaan (kuva 11): tutkimukseen, ideointiin, pilotointiin ja implementointiin.



Kuva 11 – Palvelumuotoilun menetelmät (Lawrence ym. 2019, luku 2.3.)

Palvelumuotoilun tarkoituksena on tarjota loppukäyttäjille ja sidosryhmille helppokäyttöiset välineet ja menetelmät yhteiskehittämiseen. Sidosryhmillä tulee olla mahdollisuudet ongelmanratkaisuun, vuorovaikutukseen, oppimiseen ja itseilmaisuun. ((Tuulaniemi 2011,53.)

4.2.1 Tiedonkeruun menetelmiä

Palvelumuotoilun tutkimus mahdollistaa kehittäjille syvemmän ymmärryksen loppukäyttäjistä ja auttaa kehittämään palvelua loppukäyttäjälähtöisesti (Lawrance ym. 2019, luku 2.3.)

Tiedonkeruun tarkoitus on auttaa ymmärtämään loppukäyttäjiä. Tietoa voi kerätä monella tapaa, mutta tulkintojen tekeminen kerätystä tiedosta, etenkin subjektiivisissa asioissa, tulee suhtautua varoen. (Stickdorn, Hormess, Lawrence, Schneider 2018, luku 3.1.)

Tiedonkeruu on palvelumuotoilussa kriittistä, sillä sen avulla kehittäjät voivat varmistaa, etteivät toimi pelkkien olettamusten pohjalta. Tiedonkeruun voi jakaa määrällisiin ja laadullisiin menetelmiin. Määrällisillä menetelmillä on hyvä etsiä vastauksia kysymyksiin ”mitä” ja ”miten”, kun taas laadulliset menetelmät auttavat selvittämään ”miksi”. Tiedonkeruuta voi järjestelmäkehityksessä hyödyntää tunnistamaan loppukäyttäjien välisiä taitoeroja ja ongelmakohtia, erilaisten toiminnallisuuksien toimivuutta ja käyttöastetta, ja tunnistamaan kehityskohtia ja -ideoita. (Stickdorn ym. 2018, 5.1.)

Tiedonkeruuta loppukäyttäjiltä voi tehdä joko tarkastelemalla heidän järjestelmäkäyttöänsä, tekemällä kyselytutkimuksia, haastattelemalla, tai järjestämällä työpajoja. Kvantitatiiviset tiedonkeruumenetelmät, kuten tapahtumamäärien seuraaminen järjestelmästä tai tyytyväisyyskyselyt numeroasteikolla, auttavat löytämään ja tunnistamaan ongelmakohtia. Työpajat tai avoimilla kysymyksillä varustetut kyselytutkimukset tarjoavat puolestaan oivaltavampia näkemyksiä ja uusia ideoita, mutta vaativat enemmän aikaa ja valmistelua. (Stickdorn ym. 2018, 5.2.)

Kvantitatiivisten tutkimusten tulisi olla iteratiivisia, jotta niistä saa parhaiten hyödyn irti. Muutosta on helppo tarkastella, ja tutkimuksen laajuutta tulisi nopeasti supistaa, kun oleellisimmat kohdat on tunnistettu. Tutkimukselle tulee luoda toistuva kierto, jotta muutosta on helppo mitata, mahdollistaen nopean reagoinnin. (Stickdorn ym. 2018, 5.2.)

Palvelumuotoilussa käytetään monia erilaisia tiedonkeruun menetelmiä, usein suosien kvalitatiivista dataa. Kaikissa tutkimusmuodoissa on kuitenkin niin hyviä, kuin huonoja puolia. Tästä syystä on hyvä käyttää useampaa menetelmää tiedonkeruussa. Nyrkkisääntönä olisi hyvä löytää yksi tutkimusmetodi kaikista seuraavista kategorioista: työpöytätutkimus, etnografinen tutkimus, osallistujatutkimus, osallistujaton tutkimus, yhteiskehittämis työpajat. (Stickdorn ym. 2018, 5.2.)

Metoditriangulaatio tarkoittaa tiedonkeruumenetelmien monipuolista käyttöä datan laadun varmistamiseksi. Ajatus perustuu navigointitekniikkaan, jossa useamman maamerkin tunnistamalla pystyy laskemaan todellisen sijaintinsa. Mitä useampi maamerkki, sitä varmempi voi olla sijainnistaan. Tiedonkeruun kontekstissa tämä tarkoittaa siis, että totuus löytyy, kun vertailee eri tiedonkeruumenetelmien tuottamia tuloksia keskenään, eikä luota vain yhden menetelmän tuottamaan dataan. (Stickdorn ym. 2018, 5.2.)

4.2.2 Prototyypit ja pilotointi

Prototyyppien tekeminen auttaa sekä kehittäjiä, että loppukäyttäjiä tutustumaan ja arvioimaan järjestelmää ja sen toiminnallisuuksia. Prototyypit auttavat viestimään mahdollisuuksista järjestelmän tulevaisuudessa. Prototyyppien avulla voi nopeasti tunnistaa tärkeitä huomioita uuden konseptin käytännöllisyydestä ja toteutuskelpoisuudesta. Lisäksi se auttaa luomaan yhteistä ymmärrystä ratkaisukokonaisuudesta. (Stickdorn ym. 2018, 7.1.)

Kokeilemalla ideoiden pohjalta luotuja prototyyppejä käytännössä, oikeiden sidosryhmien kanssa ja realistisissa asiayhteyksissä, päästään eroon kehittäjän ennakkoasenteista ja virheellisistä näkemyksistä. (Stickdorn ym. 2018, 7.1.)

4.3 Viestintä

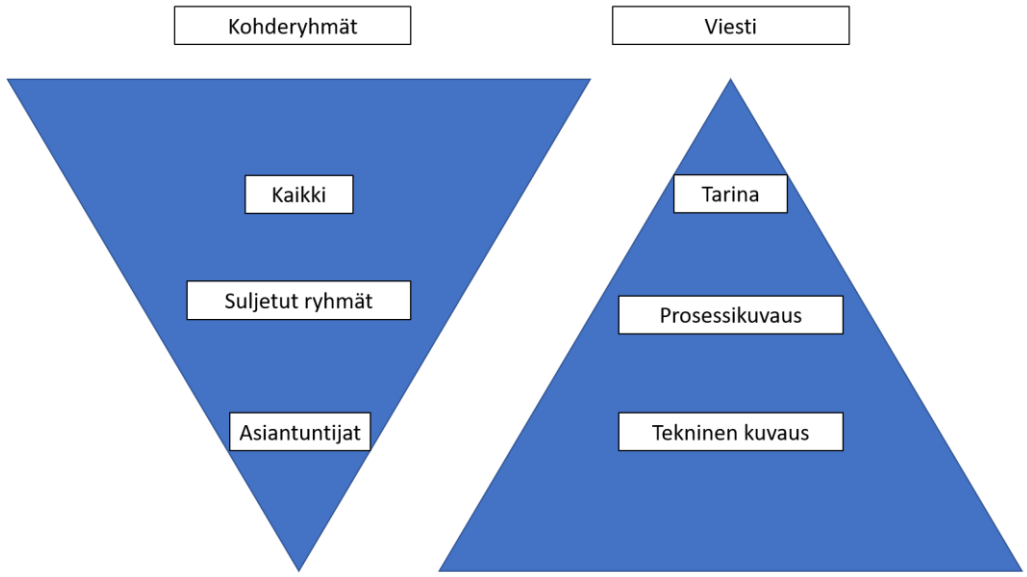
Loppukäyttäjien osallistaminen mainitaan useaan otteeseen kehitystyön ja palvelumuotoilun malleissa. Osallistaminen vaatii aina viestintää, ja tässä osiossa käsitellään lyhyesti aihepiiriin liittyviä viestinnän periaatteita.

On helppoa mainita, että pyritään avoimeen viestintään ja että kuka tahansa saa kommentoida tai osallistua, mutta käytännön toteutus on vaativampaa. Halutulle viestinnälle pitää luoda sellaiset puitteet, että se on luontevaa, informatiivista ja helppoa kaikille sidosryhmille ilman, että viestintään osallistuminen tuntuu työläältä, kiusalliselta tai pelottavalta. Tähän vaaditaan oikeat viestintäkanavat. (Juholin 2009, 155–157.)

4.3.1 Tiedottaminen

Tiedottaminen on yksisuuntaista viestiä, jossa tavoitteena on välittää tietoa vastaanottavalle osapuolelle. Tiedottaminen voi synnyttää vuoropuhelua, mutta se ei ole sen perimmäinen tarkoitus, vaan päätavoite on jakaa tietoa. Tiedotteen on hyvä olla tiivis, hyvin otsikoitu ja visuaalisesti jäsennelty. Tiedotteen sisältö tulee lajitella tärkeysjärjestykseen. (Lohtaja-Ahonen & Kaihovirta-Rapo 2012, 45–46).

Viestintästrategiaa suunnitellessa on hyvä miettiä, mille tasoille tiedottaminen kannattaa jakaa ja minkälaista viestinnän olisi hyvä olla. Tähän voi hyödyntää kohdistamisen kolmioita (kuva 12), jonka mukaan viestissä kannattaa painottaa tarinankerronnallista näkökulmaa, kun viestitään suurelle yleisölle, prosessikeskeistä viestintää, kun viestitään tietylle ryhmälle, ja teknistä kuvausta kun, viestitään asiantuntijoille. (Juholin 2009, 261).



Kuva 12. Kohdistamisen kolmiot (mukaillen Juholin 2009, 261)

4.3.2 Palaute

Hyvä palaute on käyttäjän ilmaisema havainto, joka kuvaa menneisyyttä tai nykytilaa ja sen tavoitteena on vaikuttaa tulevaan. Tarkoitus voi olla vahvistaa tai korjata palautteen saajan toimintaa. Palaute ei moiti tai tuomitse, vaan on neutraali tosiasia, eikä sitä tulisi mieltää myönteisenä tai kielteisenä. (Lohtaja-Ahonen & Kaihovirta-Rapo 2012, 26).

Palaute on hyvä tapa kehittää toimintaa. Vahvistava palaute kannustaa jatkamaan ja kehittämään oikeita asioita, kun taas korjaava palaute antaa mahdollisuuden harkita toiminnan muuttamista. (Lohtaja-Ahonen & Kaihovirta-Rapo 2012, 26).

4.3.3 Oikeat viestintäkanavat

Tietotulva on yleinen ongelma, jossa työntekijät kokevat, että tietoa tulvii monesta eri kanavasta eikä kaikki tieto ole heille relevanttia. Tällöin oikean ja kiinnostavan tiedon löytäminen voi olla vaikeaa, ja siksi on tärkeää valita oikea viestintäkanava, jolla tavoittaa oikeat kohderyhmät ja saa viestinsä näkyviin. (Lohtaja-Ahonen & Kaihovirta-Rapo 2012, 50).

Kanavaa valitessa on hyvä huomioida yrityksen kulttuuri ja jos mahdollista, hyödyntää jotain jo käytössä olevaa ja kulttuurinmukaista viestintäkanavaa. Viestinnässä voi hyödyntää useaa esitystapaa, kuten tekstiä, kuvaa ja ääntä. On eduksi, jos viestintäkanava tarjoaa joustavan mahdollisuuden käyttää näitä esitystapoja monipuolisesti. (Lohtaja-Ahonen & Kaihovirta-Rapo 2012, 51).

Kanava tulee valita niin kohderyhmän, tilanteen, kuin viestin mukaan. Näin ollen jopa saman teeman ympärille voi tarvita useamman viestintäkanavan, riippuen siitä haluaako kerätä palautetta vai tiedottaa, tai viestiikö suurelle yleisölle vai jollekin tietylle kohderyhmälle. (Lohtaja-Ahonen & Kaihovirta-Rapo 2012, 51–52).

5 Mallien esittely

Tässä kappaleessa esitellään opinnäytetyön tuotos, eli toimintamallit asiakkuudenhallintajärjestelmän kehitystyön projektinhallintaan ja loppukäyttäjien osallistamiseen. Lopullinen toteutus hyödyntää aikaisempiin kappaleisiin koottua relevanttia teoriaa ketterästä ohjelmistokehityksestä, työpaikkaviestinnästä ja palvelumuotoilusta. Toteutus on yhdistelmä teoriaosuudessa esiteltyjä malleja, sovitettuna kohdeorganisaation ja pääkäyttäjäryhmän tarpeisiin.

Ensimmäisenä esitellään projektihallinnan malli, joka koostuu työkaluista, rooleista, tapahtumista ja backlogista. Ketterän kehityksen periaatteita ja Scrum-, Kanban-, ja XP-mallin käytänteitä yhdistellen kehitettiin roolitusta, backlogia ja sen hallintaa, tapahtumien sisältöä ja prosessin läpinäkyvyyttä.

Toinen osio on loppukäyttäjien osallistaminen. Loppukäyttäjälähtöisyys on ketterän kehityksen perusideologiaa ja esiintyy vahvasti jatkuvassa kehitystyössä ja työvaiheissa. Osallistamisen malli koostuu kolmesta pääkohdasta, jotka ovat tiedottaminen, palaute ja ideointi, sekä palautteen laadun mahdollistaminen. Nämä on rakennettu hyödyntäen pääsääntöisesti teoriaosuudessa läpikäytyjä palvelumuotoilun ja viestinnän osuuksia, mutta myös poimintoja ketterän kehityksen malleista.

5.1 Projektinhallinta malli kehitystiimille

Projektinhallinnan malliksi ei valikoitunut mitään yhtä tiettyä teorian osa-aluetta, tai valmista projektimallia, vaan kohdeorganisaation tarpeisiin räätälöity kokonaisuus hyödyntäen ketterän kehityksen periaatteita ja sopivimpia osa-alueita ja oppeja eri malleista. Tämä johtuu siitä, ettei yksikään läpikäydyistä malleista soveltunut sellaisenaan kohdeorganisaation käyttöön tai olisi ollut puutteellinen. Kaikissa malleissa oli kuitenkin hyviä piirteitä, ideologioita ja tekniikoita, joista sai koottua kohdeorganisaation tarpeisiin räätälöidyn mallin.

Projektinhallinnan malli koostuu oikeista työkaluista, pääkäyttäjien roolituksesta, säännöllisistä tapahtumista ja erityisesti backlogista ja sen hallinnasta.

5.1.1 Työkalut

Projektinhallinnan keskeisin työkalu on luettelointisovellus Trello, joka otettiin käyttöön jo asiakkuudenhallintajärjestelmän käyttöönottoprojektin alkupuolella, ennen tämän opinnäytetyön tekoa. Trellossa ylläpidetään Kanban-mallin mukaisesti työjonoja ja niiden vaiheistusta. Trello on kevyt, visuaalinen ja räätälöitävä pilvisovellus, jota on helppo ylläpitää osana arjen tekemistä. Pääkäyttäjäryhmällä on muokkausoikeudet Trelloon, joilla voi luoda uusia tehtäväkortteja ja päivittää niiden

sisältöä. Loppukäyttäjillä on puolestaan katseluoikeudet samaiselle taululle, joten he näkevät läpinäkyvästi pääkäyttäjäryhmän aktiviteetteja ja suunnitelmia.

Versionhallinnassa käytetään asiakkuudenhallintajärjestelmän omia työkaluja. Sandboxit tarjoavat turvallisen kehittämisympäristön, jossa voi vapaasti kokeilla järjestelmän toiminnallisuuksia, automaatioita, päivittää visuaalista ilmettä, tai luoda uutta. Asiakkuudenhallintajärjestelmässä on oma työkalu, joilla kätevästi siirtää uusia toteutuksia ympäristöstä toiseen, lopulta tuotantoon asti.

Viestinnän työkaluna toimii kohdeorganisaatiossa muutoinkin laajassa käytössä oleva Slack pikaviestintäsovellus. Slack mahdollistaa eri teemaisten kanavien luonnin. Kohdeorganisaatiossa näitä luotiin kaksi, yksi yleistä tiedottamista ja asiakkuudenhallintajärjestelmää käyttötukea varten ja tiedotuksia ja ideointia varten toinen kanava.

5.1.2 Roolit

Kohdeorganisaation kehitystiimissä on viisi pääkäyttäjää. Täysipäiväisesti asiakkuudenhallintajärjestelmän parissa työskentelee vain yksi henkilö, jonka rooli onkin asiakkuudenhallintajärjestelmän administraattorina toimiminen. Loput tekevät pääkäyttäjätehtäviä muiden työtehtävien ohessa.

Asiakkuudenhallintajärjestelmän loput pääkäyttäjät edustavat kaikki omaa aikaisemman tai pääsääntöisen työnkuvansa funktiota, muodostaen roolitukseksi myynnin, markkinoinnin, kumppanihoidon ja datatiimin pääkäyttäjät. Näistä funktioista myynti ja kumppanihoito toimivat samassa myynnin järjestelmäympäristössä, markkinointi omassa markkinointiympäristössään ja datatiimi puolestaan tukee molempia järjestelmiä pääsääntöisesti rajapintojen kautta.

Täysipäiväinen asiakkuudenhallintajärjestelmän administraattori toimii luontevasti Scrumin mukaisen tuoteomistajan tapaisessa roolissa, vastaten tuotteen arvon tuottamisesta, backlogin ja roadmapin ylläpitämisestä, tuotteen tavoitteista ja tavoitteiden viestimisestä (Schwaber & Sutherland 2020, 5–6). Hänen vastuullaan on myös ylläpitää avointa ja aktiivista keskustelua sidosryhmien kanssa säilyttääkseen kokonaiskäsityksen markkinasta ja varmistuakseen, että tuotteen tavoite palvelee sidosryhmien tavoitteita. (Ripley & Miller 2020, luku 4.1).

Uuden projektinhallintamallin jalkauttamisesta ja seuraamisesta tulee vastaamaan tämän opinnäytetyön tekijä, eli myynnin pääkäyttäjä. Myynnin pääkäyttäjä on lisäksi aktiivisesti mukana itse kehitystyössä, toteuttaen myynnin tarvitsemia uusia toiminnallisuuksia.

Muilta osin roolituksessa sovelletaan Kanban-mallia, jossa ei esitellä uusia roolituksia projektinhallinnan mallin käyttöönoton yhteydessä, vaan malli rakennetaan nykyisten roolien päälle, muokkaamatta työnkuvia.

XP:n roolitusnäkökulma kuvaa kohtalaisen hyvin kokonaisuutta sisältäen:

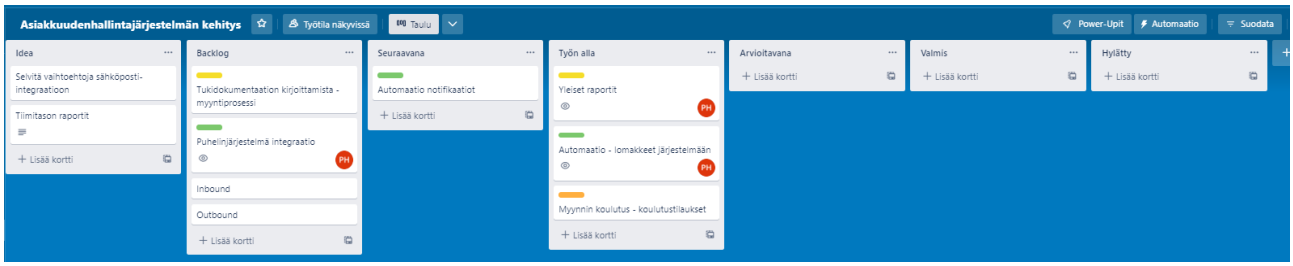
- kehittäjät, jotka kohdeorganisaatiossa ovat järjestelmän pääkäyttäjäryhmä
- asiakkaat, eli järjestelmän loppukäyttäjät
- testaajan roolin jaettuna vastuuna loppukäyttäjien ja kehittäjien välille
- tarkkailijan roolin jatkossa asiakkuudenhallintajärjestelmän administraattorin tehtävänä
- valmentajan roolin myynnin pääkäyttäjällä, joka vastaa prosessin käytöstä
- konsultin, joka on ulkoisesta konsulttiyrityksestä tarpeen tullen käytettävissä
- esihenkilöitä, joita on useampi johtuen pääkäyttäjäryhmän edustamista eri funktioista

5.1.3 Backlog

Kohdeorganisaation käyttötarve on kevyt, visuaalinen, selkeä ja helposti ylläpidettävä luettelointiratkaisu. Loppukäyttäjillä ei ole resursseja tai riittävää kiinnostusta backlogin syvempään tarkasteluun, joten läpinäkyvyys vaatii hyvin pelkistetyn ja visuaalisen ilmeen. Kanban-taulu teoreettisena mallinna ja Trello työkaluna mahdollistavat tämän yhdistelmän, tarjoten helposti ja reaaliaikaisesti ylläpidettävän visuaalisen ratkaisun, joka on informatiivinen loppukäyttäjille samalla kun se palvelee pääkäyttäjäryhmän tarpeita riittävällä syvyydellä.

Trellon lisensointi ja käyttäjienhallinta mahdollistaa muokkausoikeuksilla toimivien pääkäyttäjien lisäksi loppukäyttäjien lisäämisen Trello-taululle pelkillä katseluoikeuksilla. Tämä läpinäkyvä Kanban-taulu tukee myös Scrum-mallissa painotettua läpinäkyvyyttä, jonka mukaan on ehdotonta, että kehitystyön prosessi on nähtävillä myös sidosryhmille (Schwaber & Sutherland 2020, 3).

Trelloon rakennettu backlog (kuva 13) koostuu viidestä kronologisesta vaiheesta: idea, backlog, seuraavana, työn alla, arvioitavana, ja valmis. Lisäksi jatkumolta irrallisena vaiheena hylätty. Näihin vaiheisiin lisätään tehtäväkortteja, jotka kuvaavat pääkäyttäjäryhmän työlistalla olevia tehtäviä.



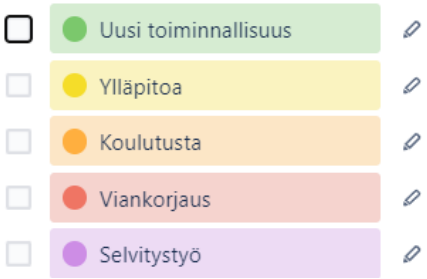
Kuva 13. Trello backlog

Vaiheistus kertoo siitä, missä vaiheessa elinkaartaan tehtäväkortti on. Vaiheistuksen lisäksi tehtäväkortit sisältävät otsikon, joka näkyy Kanban-taulun koontinäkymässä ja tarkemman kuvauksen, jonka näkee, kun kortin avaa (kuva 14). Näiden tietojen lisäksi tehtäväkortit kategorisoidaan.

Kuva 14. Trello tehtäväkortti

Tehtäväkortit luokitellaan seuraaviin kategorioihin: uusi toiminnallisuus, ylläpito, koulutus, viankorjaus, selvitystyö ja pikatyö. Näin ollen backlog ei tue pelkästään kehitystyötä, vaan kaikkea asiakkuudenhallintajärjestelmän ylläpitämiseen liittyvää tekemistä pääkäyttäjien toimesta, tarjoten kokonaisvaltaista kuvaa myös heidän ajankäytöstänsä ja sen kehityksestä. Tähän hyödynnetään Trellon toiminnallisuutta ”nimilaput” (kuva 15). Kategorioiden merkitykset:

- Uusi toiminnallisuus tarkoittaa jotain sellaista toiminnallisuutta, automaatiota tai työkalua, mitä asiakkuudenhallintajärjestelmässä ei ole ennen ollut
- Ylläpito kattaa kaiken asiakkuudenhallintajärjestelmän ylläpitoon ja nykytoiminnallisuuksilla tehtävän pääkäyttäjätyön, kuten esimerkiksi soittolistojen teon, raportoinnin, sivupäivitykset ja käyttäjäprofiilien päivittämisen
- Koulutukset tarkoittavat loppukäyttäjätukeen liittyviä palavereja, niin suunniteltuja kuin suunnittelemattomia
- Viankorjaukset tarkoittavat nykyisissä ja uusissa toteutuksissa olevien vikojen korjaamista, tavoitteena tarkkailla julkaisujen ja järjestelmän yleistä laatua
- Selvitystyöllä tarkoitetaan asiakkuudenhallintajärjestelmän ominaisuuksien, integraatioiden, asetuksien, toiminnallisuuksien, uudistusten tai mahdollisuuksien opettelua tai niihin tutustumista. Tämä on pääkäyttäjätyössä välttämättömyys ja sitä on hyvä seurata siinä missä muutakin tekemistä
- Pikatyö kategorialla tarkoitetaan tehtävää, joka ilmenee vasta kesken iteraation ja otetaan aikakriittisyytensä vuoksi työn alle. Pikatyö kategoria lisätään normaalin kategorian rinnalle, eli näillä tehtäväkorteilla on poikkeuksellisesti kaksi kategoriaa



Kuva 15. Trellon tehtäväkorttien kategoriat.

Näiden kategorioiden on tarkoitus helpottaa tehtävien jakoa, priorisointia, aikataulutusta ja raportointia. Trellon raportointimahdollisuudet tarjoavat hyvän näkyvyyden näihin kategorioihin, jolloin pääkäyttäjäryhmän kehitystä on mahdollista arvioida ja ohjata.

Otsikon, kuvauksen ja kategorian lisäksi Trello tarjoaa kollaboraatioon ja työnjakoon tarvittavat työkalut. Näiden tietojen lisäksi tehtäväkortille määritellään yksi tai useampi vastuuhenkilö, ja tehtäväkortti sisältää kommentointiosion, keskustelua ja huomioita varten (kuva X).

Manuaalisesti kirjattujen tietojen lisäksi Trelloon on luotu automaatio, joka kommentoi tehtäväkortille aina lokitiedon, kun tehtävä siirretään työlistalta toiseen, jolloin tehtäväkortin elinkaaresta kertyy automaattisesti helposti hahmotettava kuva.

5.1.4 Tapahtumat

Scrum mallissa esitellään viisi tapahtumaa, joista ylätasolla toimii yhdestä neljään viikkoa pitkä iteraatio – eli sprintti –, joka pitää sisällään neljä pienempää tapahtumaa: sprintin suunnittelu, päivittäiset scrum tapaamiset, sprintin arviointi ja scrum retrospektiivi (Schwaber & Sutherland 2020, 8–10). Kohdeorganisaatiossa tyydytään kuitenkin kahteen viikoittaiseen palaveriin ja lisäksi yhteen kuukausittaiseen palaveriin.

Sprintin kaltaiset iteraatiot ovat luonteva tapa tehdä kehitystyötä ja tulevat olemaan osa kehitystyön rytmitystä kohdeorganisaatiossa. Kohdeorganisaation iteraatioiden kesto tulee olemaan yksi viikko. Iteraation suunnittelu tehdään pääkäyttäjäryhmän viikoittaisessa tapaamisessa, jonka agenda on seuraava:

- Käydään läpi edeltävän iteraation tehtäväkortit ja hyväksytään tuotantoon siirto
- Arvioidaan ja muotoillaan uudet ideat tehtäväkorteiksi backlogille
- Valikoidaan seuraavaan iteraatioon tehtäväkortit backlogilta, ja määritellään lykkäysaika lopuille backlog tehtäväkorteille

- Siirretään tehtäväkortit seuraavana-vaiheeseen
- Aikataulutetaan valittujen tehtäväkorttien tekemiseen tarvittavat palaverit tarkemmilla teemoilla ja osallistujatiedoilla, mikäli tarpeen

Tehtäväkorttien käsittelyä käydään tarkemmin läpi seuraavassa kappaleessa. Scrum näkökulmasta pääkäyttäjien viikkopalaveri nivoo siis yhteen sprintin suunnittelun, arvioinnin ja retrospektiivin. Toisaalta XP-mallin näkökulmasta tämä suunniteltu viikkopalaveri vastaa sekä ajoitukseltaan, että sisällöltään siinä esiteltyä viikkopalaverimallia, joskin sillä poikkeuksella, että loppukäyttäjät eivät ole läsnä valitsemassa seuraavan iteraation tehtäväkortteja. Lisäksi pääkäyttäjäryhmä on pitänyt entuudestaan jo viikkopalavereja, mutta ilman tarkkaan määriteltyä agendaa. Tämän opinnäytetyön esittelemä viikkopalaveriformaatti on siis yhdistelmä kohdeorganisaation tottumuksia, Scrumia ja XP-mallia.

Tämän viikkopalaverin lisäksi pidetään tekninen palaveri, jossa tehdään tuotantojulkaisu. Tämä tehdään asiakkuudenhallintajärjestelmän pääkäyttäjän valvomana ja palaveriin osallistuvat kaikki, joilla on sandboxista tuotantoon siirtyviä monimutkaisia toteutuksia. Pienet, hyvin valmistellut tuotantomuutokset eivät edellytä läsnäoloa, vaan pääkäyttäjä voi tehdä ne, mikäli näin on sovittu viikkopalaverissa. Kyseessä on siis suoraviivainen, lyhyt, ja operatiivinen palaveri.

Koska kohdeorganisaation pääkäyttäjäryhmä ei täysipäiväisesti työskentele asiakkuudenhallintajärjestelmän parissa, on päivittäisten tapaamisten järjestäminen haasteellista ja mahdollisesti tehotonta, vaikka sitä suositeltaisiin niin Scrum-, kuin Kanban-malleissa.

Pääkäyttäjäien viikkopalaverit keskittyvät tiiviisti iteraatioiden sisältöön ja tehtäväkorttien käsittelyyn, eikä pidemmän aikavälin tavoitteita käydä niiden yhteydessä erikseen läpi. Tätä varten pääkäyttäjillä on viikkopalaverien lisäksi kuukausittainen roadmap palaveri, jossa käsitellään roadmap-dokumenttia. Tämä dokumentti sisältää funktioittain jaotellut pidemmän aikavälin tavoitteet. Palaverin agenda on seuraava:

- Tarkastellaan yhdessä roadmap-dokumenttia ja aikaisempaa suunnitelmavedosta.
- Kukin pääkäyttäjä esittelee edistymistä ja muutostarpeita oman funktionsa roadmap-tavoitteiden osalta.
- Merkitään mitä on tehty ja keskustellaan edistymisestä.
- Päivitetään suunnitelmaa tarpeen mukaan.
- Tunnistetaan pullonkauloja ja ongelmakohtia, niin järjestelmässä kuin prosesseissa.
- Tarkastellaan Trellon tuottamien raporttien pohjalta pääkäyttäjien toimintaa
- Punnitaan teemoja seuraavalle kvartaalille.
- Lisätään uusia pääkohtia, mikäli sellaisia on ilmennyt.

Kuukausipalaveri vastaa viikkopalaverin tapaan vahvasti XP-mallissa esitettyä kvartaalipalaveria. XP:n lähestymistavasta poiketen aikaväli on tiiviimpi ja pääpaino on enemmän tulevan suunnittelussa kuin tehdyn reflektoinnissa, vaikka molemmat ovat osana agendaa.

Edeltävän kuukauden reflektointiin voidaan hyödyntää Trellon raportteja, jotka kuvaavat tehtäväkorttien jakautumista tekijöiden, vaiheiden ja kategorioiden välillä.

Ennen kuukausipalaveria pääkäyttäjien tulee käydä oman funktionsa osuus roadmap-dokumentista läpi, ja valmistautua kertomaan edistymisestä ja muutostarpeista sen suhteen. Tämä sujuvoittaa kuukausipalaverin kulkua, kun yleinen tilannekuva saadaan läpikäytyä tehokkaasti.

5.1.5 Backlog tehtäväkorttien käsittely

Kuten edeltävässä kappaleessa mainittiin, pääkäyttäjien viikkopalaverissa käsitellään ensin valmiit tehtävät, jonka jälkeen suunnitellaan seuraava iteraatio. Tämä kappale on rakenneltu tehtäväkortin elinkaaren mukaan, eli aloitamme tehtäväkortin luonnista.

Kanban ideologiaa hyödyntäen uudet tehtävät lisätään backlogin ensimmäiseen sarakkeeseen, eli ’idea’-vaiheeseen. Tähän vaiheeseen tehtäväkortteja voi lisätä kuka tahansa pääkäyttäjistä, koska tahansa, eikä tehtäväkorteilla tarvitse vielä olla tarkkaan määriteltyä rakennetta tai erityisen kattavaa kuvausta. Kuvauksessa on kuitenkin hyvä olla sen verran sisältöä, että sen osaa jäsennellä myöhemmin pääkäyttäjien viikkopalaverissa.

Pääkäyttäjien viikkopalaverissa uudet ideat strukturoidaan paremmin ja siirretään backlog vaiheeseen, mikäli ne hyväksytään työjonoon. Idea-vaiheen tehtäväkortteja voidaan myös siirtää hylätty vaiheeseen, mikäli idea ei jostain syystä ole toteutettavissa tai sitä ei haluta toteuttaa.

Seuraavaksi backlogilta valikoidaan tehtävät työjonoon seuraavana-vaiheeseen. Backlog-vaiheesta siirrettäessä tehtäväkortti saa vastuuhenkilön, joka ottaa vastuun tehtäväkortin edistämisestä. Tämäkin vaihe tapahtuu pääkäyttäjien viikkopalaverin yhteydessä. Tarkoitus on kerätä seuraavana-vaiheeseen sellainen määrä tehtäviä, jonka pääkäyttäjät saavat toteutettua seuraavan iteraation aikana.

Backlog tehtäviä läpikäydessä tehtävät poimitaan joko seuraavana-vaiheeseen tai lykätään myöhemmälle. Trello tarjoaa toiminnallisuuden, jossa lykätylle tehtäväkortille määritellään lykkäysaika, jonka ajaksi se katoaa Kanban-näkymästä. Näin backlog saadaan pidettyä lyhyenä ja helposti läpikäytävänä, ilman pelkoa siitä, että jokin tehtävä unohtuisi, sillä tehtävä palaa lykkäysajan jälkeen takaisin näkyviin.

Tässä vaiheessa päättyy pääkäyttäjien viikkopalaverin osuus, ja seuraavana-vaiheesta eteenpäin tehtäväkortit etenevät tekijöiden toimesta iteraation aikana.

Sitä myötä, kun tehtäväkortteja otetaan työstettäväksi, siirretään niitä työn alla - vaiheeseen. Tehtäväkorttien kommentit osioon tulee kommentoida työn edistymisestä, tai mikäli tehtäväkortin työstämisessä tulee vastaan jotakin muuta huomionarvoista. Työn alla – vaiheesta tehtäväkortti etenee joko arvioitavana-vaiheeseen odottamaan iteraation päättymistä tai tuotantoon vientiä, tai tarpeen mukaan seuraavana-, backlog-, tai ’hylätty’-vaiheeseen.

Lopulta iteraatio päättyy ja koittaa uusi pääkäyttäjien viikkopalaveri. Ensin tarkastellaan arvioitavana-vaiheessa olevat kortit ja tehtäväkorttien vastuuhenkilöt kertovat lyhyesti tehtävästä ja sen toteutuksesta. Esittelyn jälkeen tehtäväkortti siirretään ’valmis’-vaiheeseen, mikäli kaikki ovat asiasta yhtä mieltä. Mikäli tehtävästä syntyi luonnollinen jatkotehtävä, niin kuin usein käy, on se hyvä kirjata otsikkotasolla ideat-vaiheeseen heti.

Valmis-vaiheeseen kertyy koonti iteraation aikana tehdyistä tehtävistä, ja asiakkuudenhallintajärjestelmän pääkäyttäjä tekee näiden pohjalta viikkokoonnin palaverin jälkeen. Valmis-vaiheessa tehtäväkortit ovat näkyvillä aina seuraavan iteraation ajan, jonka jälkeen ne arkistoituvat automaattisesti. Tämän jälkeen ne eivät ole enää näkyvillä backlogilla, mutta löytyvät tarvittaessa arkistosta.

Tässä kohtaa pääkäyttäjien viikkopalaverin ensimmäinen vaihe on käyty läpi ja siirrytään takaisin tämän kappaleen alkuun, eli ’idea’-vaiheen tehtävien käsittelyyn.

5.2 Loppukäyttäjien osallistaminen

Kehitystyössä pyritään aina loppukäyttäjälähtöiseen ajatteluun, mutta sen varmistamiseksi ja vahvistamiseksi on loppukäyttäjiä hyvä osallistaa. Kehitystyöhön osallistamiseen on kolme oleellista osa-aluetta: tiedottaminen, ideointi, ja palaute.

Tämän osion toteutusehdotukset on luotu hyödyntäen teoriaisuuksia etenkin viestinnän ja palvelumuotoilun osalta, mutta myös ketterän kehityksen malleista.

On helppoa mainita, että pyritään avoimeen viestintään ja että kuka tahansa saa kommentoida tai osallistua, mutta käytännön toteutus on vaativampaa. Halutulle viestinnälle pitää luoda sellaiset puitteet, että se on luontevaa, informatiivista ja helppoa kaikille sidosryhmille ilman, että viestintään osallistuminen tuntuu työläältä, kiusalliselta tai pelottavalta. (Juholin 2009, 155–157.)

Hyvä palaute on käyttäjän ilmaisema havainto, joka kuvaa menneisyyttä tai nykytilaa ja sen tavoitteena on vaikuttaa tulevaan. Tarkoitus voi olla vahvistaa tai korjata palautteen saajan toimintaa. Palaute ei moiti tai tuomitse, vaan on neutraali tosiasia, eikä sitä tulisi mieltää myönteisenä tai kielteisenä. (Lohtaja-Ahonen & Kaihovirta-Rapo 2012, 26).

5.2.1 Tiedottaminen

Viestinnän tulee ennen kaikkea olla selkeää, ja tämän selkeyden ylläpitämiseksi on viestinnälle hyvä olla omat prosessinsa. Avoin dialogi loppukäyttäjien kanssa alkaa yhteisestä ymmärryksestä. Hyvällä tiedottamisella ja viestinnällä saadaan luotua pohja loppukäyttäjälähtöiselle kehitystyölle.

Kaikille loppukäyttäjille tärkeitä asioita ovat tiedotteet ja käyttäjätuki, joten nämä tuodaan yhteen asiakkuudenhallintajärjestelmän käyttäjien yhteiselle Slack-kanavalle. Tämän kanavan tavoitteena on varmistaa loppukäyttäjien kyvykkyys käyttää asiakkuudenhallintajärjestelmää riittävällä tasolla ja pysyä ajan tasalla muutoksista.

Tiedottaminen rakentuu pääsääntöisesti viikoittaisista päivityskoosteista, jotka julkaistaan pääkäyttäjäryhmän toimesta sprintin päätteeksi Slack-kanavalle. Tämä viestintä on kohdistettu kaikille, ja on näin ollen tiivis ja selkeästi jäsennelty, eikä kovin yksityiskohtainen. Slack valikoitui viestintäkanavaksi tiedottamiseen, koska se on jo osa kohdeorganisaation viestintäkulttuuria ja tarjoaa näin hyvän tavoittavuuden. Lisäksi se mahdollistaa tekstin visualisoinnin ja jäsentelyn toiminnallisuudet, ja lisäksi mahdollisuuden rikastaa tekstiä linkeillä, äänitteillä ja videoilla.

Linkkien upottaminen päivityskoosteisiin mahdollistaa tarkempien prosessikuvauksien ja -ohjeiden linkittämisen suoraan koosteisiin, jolloin aiheesta kiinnostuneiden käyttäjien on helppo porautua tarkempaan informaatioon suoraan päivityskoosteesta.

Mikäli nämä tiedottamisen tasot eivät riitä riittävän varmaan tavoitettavuuteen, voidaan viestiä vielä tehostaa kohderyhmälle luontevammalla foorumilla, tai jos tiedotteen sisältö on liian monimutkainen sisäistettäväksi, voidaan aiheesta pitää erillinen koulutus asiaankuuluvalla kokoonpanolla.

5.2.2 Palaute ja ideointi

Edellisessä kappaleessa käsitelty tiedottaminen on pääsääntöisesti yksisuuntaista, eikä kovin osallistavaa. Se luo kuitenkin hyvin pohjan yhteiselle ymmärrykselle, joka on perustavanlaatuinen edellytys yhteiskehittämiselle. Yhteinen kehitystyö koostuu kahdesta osasta, palautteesta ja ideoinnista.

Palautteen antamisen kynnys on aina suuri ja sitä tulee madaltaa mahdollisuuksien mukaan. Tästä syystä palautekanavana toimii jo kohdeorganisaatiolle tuttu Slack. Se on käyttäjille helppo ja

luonteva kanava, joka mahdollistaa avoimen dialogin ja hyvän näkyvyyden. On myös luontevaa, että asiakkuudenhallintajärjestelmään liittyvät viestintäkanavat ovat samassa paikassa, ja näin ollen tiedottamiseen ja käyttäjätukeen käytetään Slack kanavaa.

Palautteen anto ja ideointi perustuu vapaaehtoisuuteen, kiinnostukseen ja aktiivisuuteen. Koska loppukäyttäjille asiakkuudenhallintajärjestelmän kehitystyö ei ole korkean prioriteetin tehtävä ja vie aina aikaa heidän omilta työtehtäviltänsä, tulee osallistumisen kynnys tehdä mahdollisimman matalaksi.

Palvelumuotoilussa esitetään ideoinnin jaottelu divergenssiin ja konvergenssiin, jossa divergenssi on luovaa ja vapaata ideointia, kun taas konvergenssi analyyttisempaa ja kriittisempää lähestymistä. Slack-kanavan on tarkoitus luoda luonteva ympäristö divergenssin harjoittamiseen loppukäyttäjille.

Loppukäyttäjien divergenssin kautta luotuja luovia ideoita supistetaan pääkäyttäjäryhmän toimesta lähestymällä ideoita konvergenssin kautta. Tämä kahtiajako tulee näkymään konkreettisesti siinä, että loppukäyttäjien ideointi ja siihen liittyvä dialogi tehdään vapaamuotoisella Slack foorumilla, josta pääkäyttäjät koostavat ideat Trellon backlogille selkeällä struktuurilla. Trellon backlog on myös nähtävillä ideoiden edistymien seurantaa varten.

5.2.3 Palautteen laadun mahdollistaminen

Kuten kaikki esitellyt ketterän kehityksen mallit painottavat, loppukäyttäjälähtöisyys ja palautteen huomioiminen prosessissa ovat menestyksen kulmakiviä. Kaikilla esitellyillä ketterän kehityksen malleilla on yhtäläisyyksistä huolimatta erilainen kulma siihen, miten laadukkaan palautteen saanti mahdollistetaan:

- XP-mallissa painotettaan kehitystyön nopeita tuotantojulkaisuja, jotta kehitystyö ja loppukäyttäjäpalaute tapahtuvat lähekkäin (Beck 1999, luku 7).
- Kanban-mallissa tavoitellaan lyhyttä palautesykliä rajoittamalla kerrallaan työstettävien tehtäväkorttien määrää ja näin takaamalla lyhyet läpimenoajat (Hammarberg & Sunden 2014, luku 5.1).
- Scrum-mallissa palautteen perustana nähdään läpinäkyvät kehitystyön prosessit ja säännölliset tapaamiset sidosryhmien kanssa (Schwaber & Sutherland 2020, 3.)

Kuten aikaisemmin mainittu, kohdeorganisaatiossa tullaan tavoittelemaan nopeita ja säännöllisiä tuotantojulkaisuja ja lyhyitä iteraatioita. Näin ollen kehitystyön ja loppukäyttäjäkokemuksen ja palautteen saannin välille jää lyhyt väli, mahdollistaen ajankohtaisen palautteen ja helpon reagoinnin.

Kanban rajoitukset mahdollistavat realistisen näkymän työstettäviin tehtäviin, sen sijaan, että se olisi jatkuvasti kasautuva työlista backlogin rinnalle. Rajoituksilla on tarkoitus varmistaa, ettei pääkäyttäjä aloita useita tehtäviä, viimeistelemättä vanhoja tehtäväkortteja. Tarkka rajaus pakottaa viimeistelemään ja saattamaan tuotantokäyttöön vanhat tehtävät, ennenkö voi ottaa uusia tehtäväkortteja työn alle. Näin varmistetaan, että tehty kehitystyö jalkautuu nopeasti tuotantokäyttöön ja altistuu loppukäyttäjäpalautteelle.

Scrum-mallissa esitetty vaatimus läpinäkyvyydelle toteutuu kaikille vapaiden Slack-kanavien ja Trello backlogin myötä. Säännölliset tapaamiset sidosryhmien kanssa järjestetään kohdeorganisaatiossa tiimitasolla kvartaaleittain. Agendana elää ajankohtaisten asioiden pohjalta, tavoitteena käsitellä palautteita, kouluttaa uusia toiminnallisuuksia, kasvattaa käyttöastetta ja lisätä molemminpuolista ymmärrystä.

6 Pohdinta

Tässä kappaleessa käydään läpi opinnäytetyön tuloksia. Ensin puretaan toinen kyselytutkimus, joka oli identtinen ensimmäisen kyselytutkimuksen kanssa. Toinen kyselytutkimus toteutettiin kolme kuukautta opinnäytetyössä esiteltyjen mallien jalkauttamisen jälkeen, kun taas ensimmäinen kyselytutkimus tehtiin opinnäytetyön alkuvaiheilla, samassa kuussa, kun uusi asiakkuudenhallinta-järjestelmä otettiin käyttöön.

Lisäksi pohdintaan miten opinnäytetyön tavoitteet saavutettiin, miten yleishyödyllinen tuotos on, missä erityisesti onnistuttiin ja missä puolestaan oli haasteita.

Lopuksi käydään läpi jatkokehittämisideoita ja –tarpeita.

6.1 Kyselytutkimus 2

Toinen kyselytutkimus toteutettiin noin 10 kuukautta ensimmäisen jälkeen. Ensimmäinen tehtiin samassa kuussa, kuin uusi asiakkuudenhallintajärjestelmä otettiin käyttöön.

Esitetyt kysymykset olivat identtisiä ja käytössä oli sama 1–5 asteikko.

- Minulla on hyvä käsitys siitä, mitä järjestelmässä kehitetään parhaillaan

Ensimmäisen kyselytutkimuksen keskiarvo: 3.7
Toisen kyselytutkimuksen keskiarvo: 3.8
Muutos: +0.1

Kysymyksen tarkoitus oli selvittää, kuinka hyvin olimme onnistuneet viestimään työstettävistä tehtävistä, niiden ollessa työn alla. Vaikka läpinäkyvyyttä työstettäviin tehtäviin parannettiin tarjoamalla kaikille katseluoikeudet Trelloon, ei loppukäyttäjillä ole riittävää mielenkiintoa sen seuraamiseen ja näin ollen vastaus ei ole erityisemmin muuttunut tehdystä kehityksestä huolimatta.

- Tiedän mitä muutoksia järjestelmään on lähiaikoina tehty

Ensimmäisen kyselytutkimuksen keskiarvo: 3.8
Toisen kyselytutkimuksen keskiarvo: 4.4
Muutos: +0.6

Tehdystä kehityksestä viestitään viikoittain selkeällä ja kompaktilla formaatilla kaikkien loppukäyttäjien Slack-kanavalle. Tämä on ollut loppukäyttäjille mieluisa ja vaivaton tapa pysyä ajan tasalla muutoksista ja se näkyy myös kyselytutkimuksen vastauksissa.

- Tiedän mitä järjestelmässä ollaan kehittämässä seuraavan kvartaalin aikana

Ensimmäisen kyselytutkimuksen keskiarvo: 3.1
Toisen kyselytutkimuksen keskiarvo: 3.1
Muutos: 0

Näkyvyys seuraaviin kuukausiin on osin ennustettavissa Trello backlogilta, mutta suuri osa kehitystarpeista ilmenee yllättäen ja aikakriittisenä, johtaen huonoon ennustettavuuteen. Matala arvosana tähän kysymykseen johtuu osin kehitystyön vaikeasta ennustettavuudesta ja osin loppukäyttäjien matalasta kiinnostuksesta tulevia kehityskohteita kohtaan.

- Tiedän, millaisia tavoitteita meillä on järjestelmän kehityksen suhteen pitkällä aikavälillä

Ensimmäisen kyselytutkimuksen keskiarvo: 3.5
Toisen kyselytutkimuksen keskiarvo: 3.5
Muutos: 0

Pääkäyttäjäryhmä on ylläpitänyt roadmap-dokumenttia kuukausittaisten palaverien kautta, mutta pitkän aikavälin tavoitteiden ennustettavuus on ollut heikkoa ja roadmapista on luistettu. Tästä syystä pitkän aikavälin tavoitteista ei ole viestitty aktiivisesti loppukäyttäjille, jotta pääkäyttäjät eivät esitä lupauksia, jotka jäävät lunastamatta. Näin ollen kehitystä tämän kysymyksen suhteen ei ollut odotettavissakaan.

- Koen, että minun tarpeitani on huomioitu järjestelmän kehityksessä

Ensimmäisen kyselytutkimuksen keskiarvo: 4.5
Toisen kyselytutkimuksen keskiarvo: 4.5
Muutos: 0

Yleisistä kanavista ja prosesseista huolimatta pääkäyttäjät käyvät paljon kahdenkeskeisiä keskusteluja ja palavereita loppukäyttäjien kanssa kehityksen, prosessien, koulutuksen ja ideoinnin

parissa. Tämä johtaa eri tasoisiin yksilökokemuksiin ja vaikka keskiarvo on korkealla, oli vastausten seassa myös tyytymättömiä.

- Koen, että tiimini tarpeita on huomioitu järjestelmän kehityksessä

Ensimmäisen kyselytutkimuksen keskiarvo: 4.5
Toisen kyselytutkimuksen keskiarvo: 4.3
Muutos: -0.2

Vastauksista on havaittavissa, että kehitystyö ei ole palvellut kaikkia tiimejä tasapuolisesti. Tyytyväisten tiimien vastaukset ovat säilyneet samana, mutta tyytymättömien osalta arvosanat ovat laskeneet entisestään.

- Koen, että voimme vaikuttaa siihen, mihin suuntaan järjestelmää kehitetään

Ensimmäisen kyselytutkimuksen keskiarvo: 4.6
Toisen kyselytutkimuksen keskiarvo: 4.5
Muutos: -0.1

Kehitystyötä on pyritty tekemään pääsääntöisesti käyttäjien toiveiden pohjalta ja näin ollen arvosanakin on korkea.

- Palautteen antamiseen on selkeät kanavat: 4.8

Ensimmäisen kyselytutkimuksen keskiarvo: 4.8
Toisen kyselytutkimuksen keskiarvo: 4.8
Muutos: 0

Slack pidettiin pääsääntöisenä viestintäkanavana, mutta viestinnän struktuuria ja laajuutta parannettiin entisestään. Slack oli entuudestaan käyttäjille tuttu ja on näin ollen alusta asti tuntunut heille luontevalta paikalta viestinnälle.

- Palautteen antamiseen on kannustettu

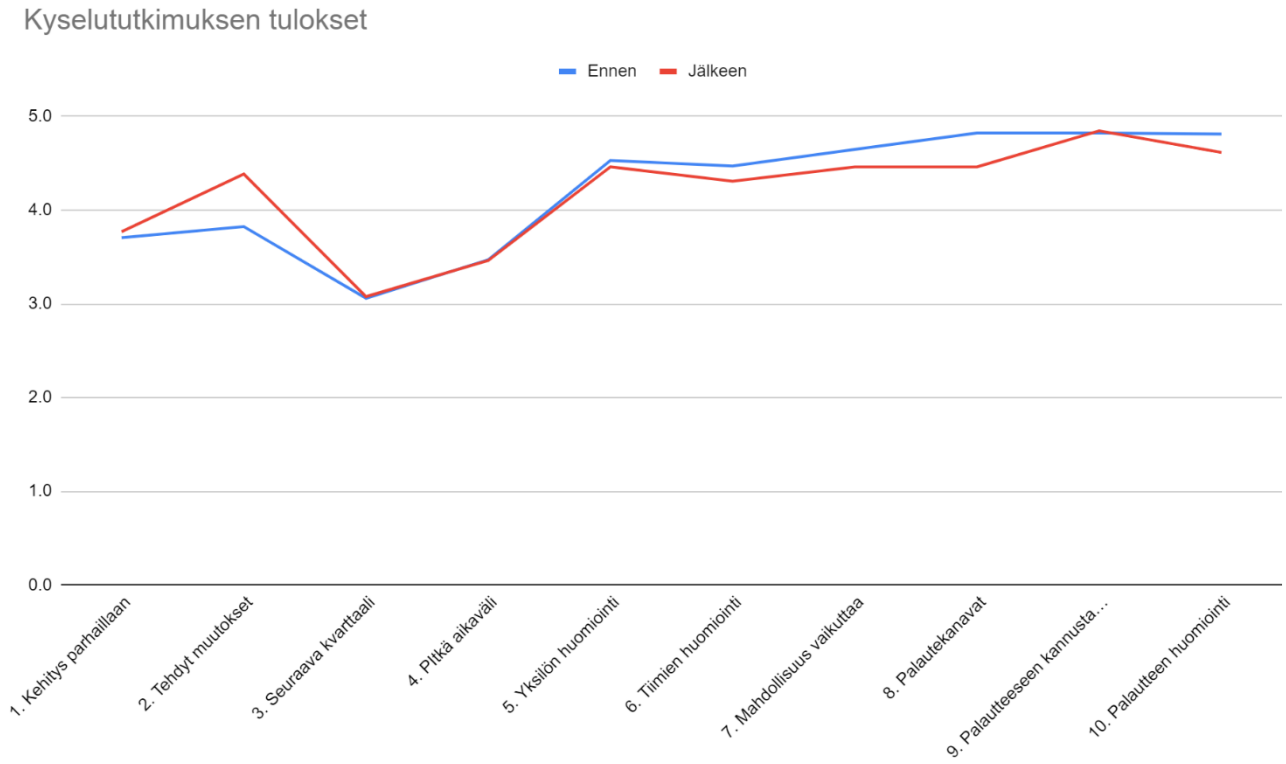
Ensimmäisen kyselytutkimuksen keskiarvo: 4.8
Toisen kyselytutkimuksen keskiarvo: 4.8
Muutos: 0

Lähes jokainen kehitystyöhön liittyvä viesti tai puheenvuoro pääkäyttäjäryhmän toimesta on sisältänyt loppukaneetin, joka on kehottanut loppukäyttäjiä antamaan palautetta tai ottamaan yhteyttä pääkäyttäjiin. Vastausten arvosanasta voi päätellä, että tämä on huomioitu.

- Koen, että palaute huomioidaan

Ensimmäisen kyselytutkimuksen keskiarvo: 4.8
Toisen kyselytutkimuksen keskiarvo: 4.6
Muutos: -0.2

Kaikki palaute on pyritty huomioimaan, mutta se ei tarkoita, että kaiken palautteen pohjalta olisi toimittu tai, että kaikkia toiveita olisi toteutettu. Tästä huolimatta on tyytyväisyys ja käsitys palautteen huomioinnista pysynyt hyvällä tasolla.



Kuva 16. Kyselytutkimusten tulokset.

Ensimmäinen kyselytutkimus tehtiin hyvin varhain asiakkuudenhallintajärjestelmän käyttöönoton jälkeen ja vastausten positiivisuudesta heräsi huoli, että muutoksen aiheuttamat haasteet eivät olleet vielä ehtineet näkymään loppukäyttäjille ja vastaukset olivat turhan optimistisia. Tästä huolimatta kyselytutkimusten tulokset olivat yllättävän samankaltaiset (kuva 16). Näin ollen se, että 10 kuukautta käyttöönoton jälkeen on saatu ylläpidettyä vastauksissa kokonaiskeskiarvona 4.2 asteikolla 1–5, voidaan nähdä onnistumisena.

6.2 Oma analyysi

Työn tavoite oli luoda malli kehitystyön hallintaan ja loppukäyttäjien osallistamiseen. Tämä opinnäytetyö palveli aitoa tarvetta, sillä pääkäyttäjäryhmästä ei löytynyt osaamista tällaisen mallin rakentamiseen. Näiltä osin työn tavoitteet saavutettiin ja tuotoksena syntynyt malli jalkautettiin käytäntöön.

Opinnäytetyön tuotoksena syntynyt kehittämistyön malli on sovellettavissa sisäisen kehitystyön tehtäviin pienille kehitystiimeille tai –ryhmille, jotka tavoittelevat joustavaa mallia, jonka keskiössä on ketterä kehitys ja loppukäyttäjälähtöisyys.

Tämä opinnäytetyö olisi toki ollut hyvä tehdä valmiiksi ennen uuden asiakkuudenhallintajärjestelmän käyttöönottoa, jotta malli olisi voitu ottaa käyttöön heti alusta alkaen, mutta tämä ei valitettavasti ollut mahdollista. Nyt malli rakentui opinnäytetyötä tehdessä hiljalleen käyttöönoton rinnalla, mikä toisaalta mahdollisti korjausliikkeet matkan varrella.

Kyselytutkimuksesta selvisi, että loppukäyttäjien tyytyväisyydessä oli tiimikohtaisia eroja ja tämän pohjalta voidaan todeta, että tasa-arvoisempi kehitystyö ja kaikkien huomioiminen olisi voinut onnistua vielä paremmin.

Pääkäyttäjien osalta on selvää, että Trelloon rakennettu prosessi tuottaa lisäarvoa, lisää näkyvyyttä ja parantaa työn laatua, tehokkuutta ja selkeyttä. Loppukäyttäjien osalta suurin kehityskohta oli läpinäkyvä ja selkeä viestintä kehitystyöstä ja loppukäyttäjien hyvät vaikuttamismahdollisuudet. Läpinäkyvyys työstettäviin ja tuleviin tehtäviin herätti puolestaan yllättävän vähäistä mielenkiintoa.

Opinnäytetyön tekeminen tuotti myös itsessään lisäarvoa. Raportin kirjoittajalla ei ollut lainkaan kokemusta prosessinhallinan malleista, kuten Kanbanista, Scrumista tai XP:stä, eikä ketterä kehitys ollut tuttu käsite. Myös backlogin hallintaan valikoitunut Trello oli tekijälle alkujaan vieras työkalu.

Ketterän kehityksen malleihin perehtyminen ja jatkuva käytäntöön sovittaminen tukivat opinnäytetyön ammatillista kasvua merkittävästi. Uuden asiakkuudenhallintajärjestelmän

käyttöönottoprojektin jalkauttaminen oli jo itsessään ammatillisesti kasvattava kokemus, mutta opinnäytetyön edellyttämä kattava perehtyminen teoriaan, asioiden pohtiminen, strukturointi ja kirjallinen ilmaisu vauhdittivat ja kasvattivat oppikokemusta merkittävästi.

6.3 Jatkokehitys

Kyselytutkimukset ja kenttäkokemus ovat tuoneet hyvin ilmi, mitkä osa-alueet tämän työn tuotoksessa ovat tuottaneet tulosta ja missä tarvitaan vielä tehostamista. Viestintä tehdyistä ja työn alla olevista tehtävistä on ollut onnistunutta. Toisaalta viestintä tulevasta kehitystyöstä, tavoitteista ja pitkän aikavälin suunnitelmista on laahannut. Näiden asioiden esille tuominen vaatii vielä selvästi kehitystyötä.

Kehitystyötä on tehty pitkälti loppukäyttäjien toiveiden pohjalta. Toiset käyttäjäryhmät ja tiimit ovat olleet aktiivisempia kuin toiset ja tämä näkyi kyselytutkimuksen vastauksissa pienenä tyytyväisyyden laskuna tiimien ja yksilöiden huomioimista sekä vaikutusmahdollisuuksia koskevissa kysymyksissä.

Priorisoinnissa tulisi näin ollen huomioida tasa-arvoinen kehitystyö, jotta kaikki loppukäyttäjät voisivat kokea järjestelmän omakseen ja tuntea omaavansa mahdollisuuden vaikuttaa sen kehittämiseen.

Pääkäyttäjien osalta backlogiin olisi hyvä sisällyttää jollain tavalla myös pitkän aikavälin tavoitteet. Kun roadmap tavoitteet kirjataan omaan suunnitelmaansa, mutta arki pyörii backlog tehtävien parissa, pääsee pitkän aikavälin tavoitteet usein unohtumaan ja tehtäväjono voi ohjata tekemistä suunnitellun suunnan ulkopuolelle. Tämä johtaa siihen, että kuukausittaisessa pääkäyttäjäryhmän roadmap palaverissa havaitaan, että on tehty täysin muita asioita kuin mistä viime kuussa sovittiin.

Roadmapista tulisi myös muotoilla kompaktiin ja visuaaliseen muotoon, jonka avulla voisi viestiä loppukäyttäjille paremmin tulevista kehityssuunnista.

Lähteet

Ketterän ohjelmistokehityksen julistus. Luettavissa: https://agilemanifesto.org/. Luettu: 16.11.2022

Beck, K. & Andres, C. Extreme Programming Explained: Embrace Change, Second Edition. Addison-Wesley Professional. Indiana. E-kirja. Luettu: 10.8.2022

Beck, K. & Fowler, M. 2000. Planning Extreme Programming. Addison-Wesley Professional. Upper Saddle River. E-kirja. Luettu: 5.7.2022

Brechner, E. 2015. Agile Project Management with Kanban. Microsoft Press. Washington. E-kirja. Luettu: 7.9.2022

Guckenheimer, & Loje, N. 2012. Visual Studio Team Foundation Server 2012: Adopting Agile Software Practices: From Backlog to Continuous Feedback, Third Edition. Addison-Wesley Professional. Indiana. E-kirja. Luettu: 10.7.2022

Gullo, J. 2016. Real World Agility: Practical Guidance for Agile Practitioners. Addison-Wesley Professional. Indiana. E-kirja. Luettu: 2.7.2022

Hammarberg, M. & Sunden, J. 2014. Kanban in Action. Manning Publications. Shelter Island. E-kirja. Luettu: 8.8.2022

Juholin, E. 2009. communicare!. 5. uudistettu painos. WS Bookwell Oy. Porvoo. 9.10.2022

Lohtaja-Ahonen, S. & Kaihovirta-Rapo, M. 2012. Tehoa työelämän viestintään. Alma Talent. Hämeenlinna. E-kirja. Luettu: 18.5.2023.

Martin, R. 2019. Clean Agile: Back to Basics. Pearson. Addison-Wesley Professional. Indiana. E-kirja. Luettu: 9.10.2022

Moreira, M. 2017. The Agile Enterprise: Building and Running Agile Organizations. Apress. Winchester. E-kirja. Luettu: 10.7.2022

Reason, B., Løvlie, L., Flu, M. 2015. Service Design for Business. John Wiley & Sons, Inc. New Jersey. E-kirja. Luettu: 12.9.2022

Ripley, R. & Miller, T. 2020. Fixing your Scrum. Pragmatic Bookshelf. E-kirja. Luettu: 5.7.2022

Rubin, K. 2012. Essential Scrum: A Practical Guide to the Most Popular Agile Process. Addison-Wesley Professional. Indiana. E-kirja. Luettu: 01.01.2023

Schwaber, K. & Sutherland, J. The Scrum Guide. The Definitive Guide to Scrum: The Rules of the Game. Luettavissa: https://scrumguides.org/docs/scrumguide/v2020/2020-Scrum-Guide-US.pdf. Luettu 18.5.2023.

Scrum.org. Luettavissa: https://www.scrum.org/resources/what-is-scrum. Luettu 14.1.2023
Shore, J. 2007. The Art of Agile Development. O'Reilly Media, Inc. Sebastopol. E-kirja. Luettu: 2.7.2022

Specchia, A. 2022. Customer Relationship Management (CRM) for Medium and Small Enterprises. Deanta Global Publishing Services. Chennai. E-kirja. Luettu: 18.5.2023.

Stickdorn, M., Hormess, M., Lawrence, A. & Schneider, J. 2018. This Is Service Design Doing. O'Reilly Media, Inc. Sebastopol. E-kirja. Luettu: 09.09.2022

Stickdorn, M., Hormess, M., Lawrence, A. & Schneider, J. 2018. This Is Service Design Methods. O'Reilly Media, Inc. Sebastopol. E-kirja. Luettu: 09.09.2022

Sutherland, J. & Coplien, J. 2019. A Scrum Book. Pragmatic Bookshelf. E-kirja. Luettu: 5.7.2022

Tuulaniemi, J. 2012. Palvelumuotoilu. Alma Talent. Hämeenlinna. E-kirja. Luettu: 12.9.2022