jamk.fi

Ohjelmistosuunnittelun tuotekehitys-prosessin kehittäminen

Juho Riekkinen

Opinnäytetyö
Kesäkuu 2017
Tekniikan ja liikenteen ala
Insinööri (YAMK), automaatioteknologian tutkinto-ohjelma

Jyväskylän ammattikorkeakoulu
JAMK University of Applied Sciences

jamk.fi

Kuvailulehti

Tekijä(t)	Julkaisun laji	Päivämäärä
Riekkinen, Juho	Opinnäytetyö, ylempi AMK	04.06.2017
	Sivumäärä 94	Julkaisun kieli Suomi
		Verkkojulkaisulupa myönnetty: x

Työn nimi
Ohjelmistosuunnittelun tuotekehitysprosessin kehittäminen

Tutkinto-ohjelma
Automaatioteknologian tutkinto-ohjelma

Työn ohjaaja(t)
Rantapuska, Seppo ja Hukari, Sirpa

Toimeksiantaja(t)
Vision Development Oy

Tiivistelmä

Vision Development Oy tarjoaa monipuolisia ohjelmistosuunnittelupalveluita eri teollisuudenaloille ja on vahvasti mukana asiakkaidensa tuotekehityksessä. Pieniä ja keskisuuria asiakkuuksia voi olla samaan aikaan aktiivisena useita, jolloin projektien ja yksittäisten tehtävien hallinta monimutkaistuu.

Näiden projektien hallintaan tuli perustaa tiimi, jonka toimintatavat ja roolitukset tehtävienhallinnassa määriteltiin prosessikuvauksessa. Tavoitteena oli kehittää projektien ennustettavuutta, työn tehokkuutta, työntekijöiden kommunikaatiota ja toimintatapoja ja sitä kautta parantaa asiakastyytyväisyyttä. Prosessikuvaus käyttöönotettiin toteuttamalla puolen vuoden ajan kehitysperiodeja määritelmän mukaisesti. Lisäksi tehtävienhallintaa tehostamaan käyttöönotettiin ja konfiguroitiin tehtävienhallintatyökalu.

Prosessikuvaus ja työkalu määriteltiin toimeksiantajan vaatimusten mukaisesti ja toimintaa kehitettiin jatkuvasti käyttöönoton aikana. Kehitysperiodien tehtävienhallintaa seurattiin työkalusta saatavilla raporteilla. Tuloksiksi kerättiin käyttäjäpalautetta prosessiin ja työkaluun liittyvästä dokumentaatiosta, työkalun konfiguroinnista ja prosessin käyttöönotosta, jolla varmennettiin vaatimustenvastaisuutta.

Tuloksien perusteella todettiin, että tehtävienhallintatyökalu ja dokumentaatio vastasivat vaatimuksia ja kehittivät tehtävienhallintaa. Prosessin käyttöönoton tuloksia ei täysin pystytty varmentamaan tiimiläisten resurssitilanteen vuoksi. Tuloksista todettiin, että prosessi oli vaatimusmäärittelyn mukainen, sen käyttöönotossa havaittiin paljon huomioitavia asioita ja se tulee toimimaan hyvänä perustana jatkokehittää tehtävienhallintaa.

Avainsanat (asiasanat)
Ohjelmistosuunnittelu, ohjelmistokehitys, tuotekehitys, prosessit

Muut tiedot

jamk.fi

Description

Author(s)	Type of publication	Date
Riekkinen, Juho	Master's thesis	04.06.2017
		Language of publication: Finnish
	Number of pages 94	Permission for web publication: x

Title of publication
Development of product development process for software design

Degree programme
Degree programme of automation technology

Supervisor(s)
Rantapuska, Seppo and Hukari, Sirpa

Assigned by
Vision Development Oy

Abstract

Vision Development Oy is offering versatile software designing services for various fields of industry and is strongly involved in their customer's product development. Multiple small and medium sized customer ships may be active at the same time when the management of single projects and tasks becomes complicated.

A team had to be invented for these projects and the team's ways of acting and roling in task management were specified in a process description. Objective was to develop the predictability of the projects, effectiveness of working, communication and working habits of employees and that way increasing the customer satisfaction. Process description was deployed by implementing process specified development periods during six months. In addition a task management tool was commissioned and configured to enhance the task management.

Process description and task management tool was specified according to requirements by the mandator and the operations were developed continuously during the commissioning. Task management of development periods were tracked by the reports supplied by the tool. User feedback concerning the process and tool related documentation, tool configuration and process commissioning was collected as results to ensure the compliance against requests.

By the results it was noted that the task management tool and documentation were according to requests and were developing the task management. Because of the resource situation of employees the results of the process commissioning could not be completely confirmed. It could have been noted that the process was in compliance against request specification, during the commissioning a lot of notable issues were noticed and the process will be a good base for further development of the task management.

Keywords/tags (subjects)
Software design, software development, product development, processes

Miscellaneous

Sisältö

Kuviot

1 Johdanto tehtävienhallintaprosessin kehittämiseen

Ohjelmistot ovat nykyaikaisissa tuotteissa enemmän käytäntö kuin poikkeus. Nykyaikainen auto voi sisältää 100, yhteisöpalvelu 60 ja lentokone tai matkapuhelimen käyttöjärjestelmä yli 10 miljoonaa riviä ohjelmaa. Mitä enemmän tuotteet ja palvelut sisältävät ohjelmistoja, sitä merkittävämmäksi kasvaa ohjelmistotuotannon prosessien ja työkalujen merkitys.

Ohjelmistotuotannon prosesseilla pyritään tekemään ohjelmistokehityksestä järjestelmällisempää ja sitä myötä parantamaan ohjelmistojen laatua ja kehittäjien tuottavuutta. Prosessit voivat olla yksiselitteisiä kuvauksia työtavoista ja käytänteistä, joita jokainen ohjelmistokehitykseen osallistuva projektijohdosta kehittäjiin ja asiakkaisiin sitoutuu noudattamaan.

Prosessin käytännön toteutusta helpottamaan on kehitetty työkaluja. Työkalut vaihtelevat yksinkertaisista paperilapuista satojatuhansia euroja maksaviin ohjelmistoihin. Työkalut ja niiden käyttäminen tulee valita ja toteuttaa yrityksen käyttötarkoituksen ja tarpeiden mukaisesti.

Opinnäytetyön tarkoituksena on kehittää Vision Development Oy:n tuotekehitysprosessia ja siihen liittyviä työkaluja. Opinnäytetyön tuloksena yritykselle on kehitetty tuotekehitysprosessi, josta on dokumentoituna kuvaus ja ohjeistus, sekä käyttöönotetut työkalut ohjeistuksineen.

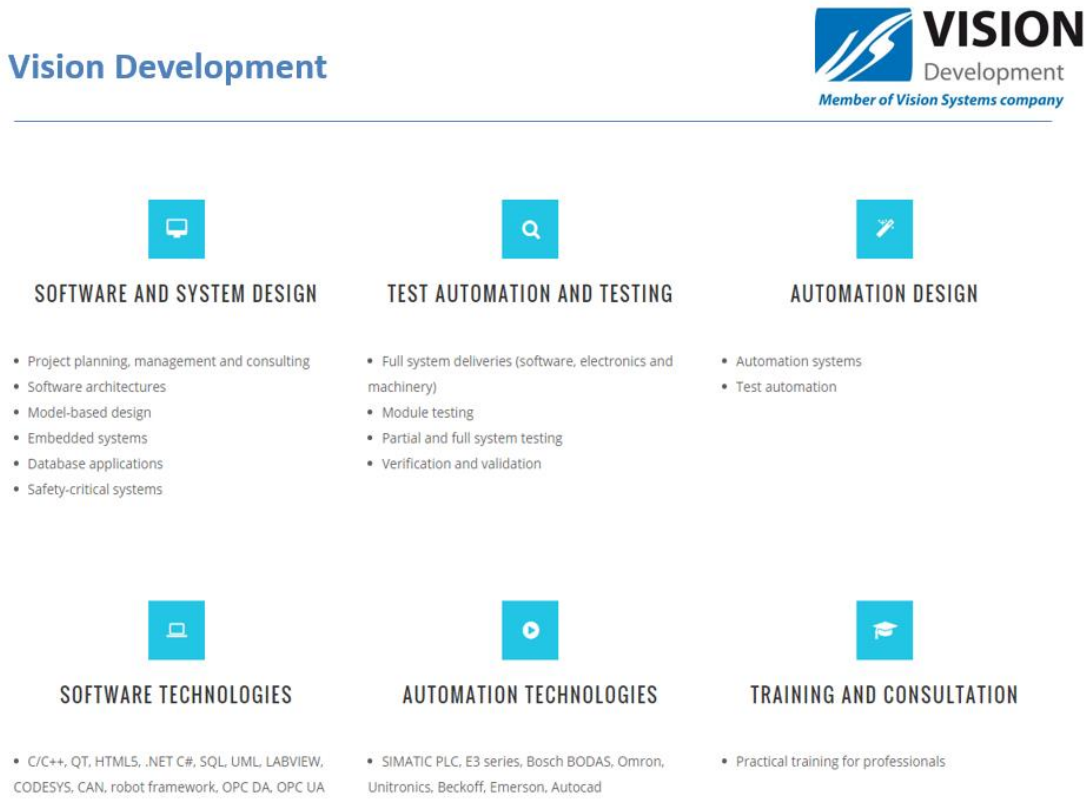
1.1 Toimeksiantajan esittely

Vision Development Oy (VD) irtaantui Vision Systems Oy:n (VS) tytäryhtiöksi vuonna 2014. Tätä ennen VS oli ehtinyt tarjoamaan ohjelmistokehitystä palveluna jo parin vuoden ajan, mutta liiketoiminnan pääpaino oli konenäkösovelluksissa.

VS on vuonna 1985 perustettu Jyväskyläläinen konenäön asiantuntijayritys, joka on tarjonnut konenäköratkaisuja vaativiin olosuhteisiin eri teollisuuden aloille. Yritys on kehittänyt omia ja integroinut valmiita järjestelmiä sovelluksen vaatimusten mukaisesti mm. rengas-, puu-, paperi- ja terästeollisuuteen. Yleisesti sovellukset ovat keskittyneet vaativimpiin kohteisiin, joihin valmiit tuotteet eivät suoraan ole pystyneet,

vaan joihin on täytynyt kehittää optimoitu ratkaisu. Yrityksellä on laaja-alaista kokemusta niin rauta- kuin ohjelmistopuolen ratkaisuista.

Vaikka VS oli jo pitkään tehnyt omien tuotteiden tuotekehitystään, vuosien 2012 ja 2013 aikana se alkoi suuntautumaan myös tuotekehityksen palveluntarjoajaksi. Aluksi palveluliiketoiminta työllisti vain muutaman henkilön, mutta kasvoi nopeasti suurimmaksi liiketoiminnan osa-alueeksi. Vuonna 2014 perustettiin oma yritys (VD) tuotekehityksen palveluliiketoiminnalle, joka tuolloin työllisti noin 10 henkilöä. Tällä hetkellä VD työllistää noin 35 henkilöä neljällä eri paikkakunnalla. Se tarjoaa kattavaa ohjelmisto- ja järjestelmäkehitystä (ks. Kuvio 1.) vaativiin sovelluksiin mm. liikkuviin koneisiin, koneteollisuuteen, testaukseen ja tietojärjestelmiin.



Kuvio 1. Vision Development Oy:n palveluliiketoiminta (Vision Development, 2016.)

VD:n projektit vaihtelevat vuosia kestävistä tuotekehitysprojekteista muutaman viikon mittaisiin kehitys- ja konsultointiprojekteihin. Riippuen projektien luonteesta

henkilöstö osallistuu joko yhteen tai useampaan projektiin kerrallaan. Lisäksi VD tekee tuotekehitys- ja toimitusprojekteja emoyhtiölleen VS:lle.

1.2 Tuotekehitysprosessin nykytilanne ja kehitystarpeet

VD on pyrkinyt toteuttamaan tiimityöskentelymallia tuotekehitysprojekteissa. Suurimmilla asiakkuuksilla on omat tiiminsä, jotka toteuttavat tuotekehitystä asiakkaan prosessien mukaisesti ja asiakkaan työkalujen avulla. Pienemmille projekteille on pyritty perustamaan tiimejä, jotka voivat toteuttaa useampia projekteja samanaikaisesti. Nämä tiimit ovat käyttäneet yrityksen sisäisiä menetelmiä ja työkaluja projektien läpiviemiseen.

Ohjelmistokehitykseen on pyritty soveltamaan V-mallia (ks. luku 2.2), mutta varsinaista yksiselitteistä prosessikuvausta tehtävienhallinnan toteutukselle ei ole ollut. Tehtävienhallinnan työkalut ovat myös olleet puutteellisia; projektien tuntitoteumaa ja suurempien kokonaisuuksien tilaa on seurattu ERP-työkalun ja yksityiskohtaisempien tehtävien tilaa Excel-taulukon pohjalle tehdyn työkalun avulla.

Prosessin määritelmän ja ohjeistuksen, sekä tarkoituksenmukaisten työkalujen puuttuminen johtavat mm. sekalaisiin käytänteisiin projektien läpiviemisessä ja aiheuttavat sen, ettei tuotteen vaatimuksia ole jäljitettävissä tehtyihin tehtäviin ja toiminnallisuuksiin.

Yhdenmukaiset käytänteet (vrt. prosessi) tehostavat työskentelyä, parantavat tuottavuutta, laatua ja tulevien projektien suunnittelua ja aikataulujen ennustettavuutta. Näiden lisäksi käytänteet vaikuttavat epäsuorasti mm. työn mielekkyyteen, työntekijöiden jaksamiseen ja yrityksen liiketoiminnan kannattavuuteen.

Jäljitettävyys tuotteen vaatimusten ja tehdyn toteutuksen välillä voi puolestaan olla jopa vaatimus esimerkiksi turvakriittisten järjestelmien suunnitteluprojekteissa. Yleisesti jäljitettävyys parantaa tuotteen ylläpitoa ja vaatimustenvastaavuutta, koska tehdylle toteutukselle on mahdollista perustella syy eli linkittää tuotteen vaatimus.

Pienempien projektien ongelmana lisäksi on ollut se, että niiden määrä vaihtelee huomattavasti ja näin ollen vaikuttaa pienprojektitiimin kokoon ja ko. projekteihin si-

toutettujen henkilöiden määrään. Jotta kehitysprosesseista saataisiin mahdollisimman suuri hyöty, tulisi prosessiin kuuluvat olla sitoutuneita noudattamaan sitä ja kehitystiimin pysyä henkilöstömäärältään soveltuvana.

Opinnäytetyössä esitellään teoriaa erilaisten tuotekehitysprosessien takana, tutustutaan tehtävienhallintatyökaluihin, esitellään yrityksen sisäiset ja pohditaan ulkoisia vaatimuksia tehtävienhallintaprosessiin ja –työkaluun, esitellään toteutettu kehityshanke ja tulokset, sekä pohditaan toteutuneen työn vaikutusta yrityksen toimintaan, omaan kehittymiseen ja tulevaisuuden kehittämishankkeisiin.

1.3 Tutkimusasetelma ja tutkimusmenetelmät

Opinnäytetyön tulisi vastata kysymyksiin, kuinka toimeksiantajan tehtävienhallintaprosessia ja työkaluja voitaisiin kehittää ja kuinka kehitysprojektin tulokset vastaavat tutkimusteoriaa ja toimeksiantajan vaatimuksia. Tutkimuksen aineistona käytetään kehitysprojektin perusteella saatua käyttäjäpalautetta tuotetusta prosessimääritelmästä, dokumentaatiosta, työkaluista ja prosessimallista. Aineisto kerätään kyselylomakkeella ja haastatteluilla. Koska kyselyaineisto on projektissa mukana olevien henkilöiden pienen määrän vuoksi suppea, pyritään haastattelujen ja prosessikäyttöönotossa havaitun materiaalin avulla saamaan tarkempaa tietoa tutkimusongelman ratkaisusta. Prosessikäyttöönoton ajalta kerätään tietoa (mm. toteutuneita työmääriä, meneillään olevien projektien määriä, henkilömääriä) toteutuneista kehitysperiodeista, joita voidaan verrata saatuun käyttäjäpalauteaineistoon.

Laadullinen ja määrällinen tutkimus kuvataan perinteisesti vastakkainasettelun kautta. On kuitenkin syytä tiedostaa, ovatko vastakkainasettelun osapuolet toisensa poissulkevia vai toisiaan täydentäviä. Perinteisesti laadullinen tutkimus kuvataan perustumaan mm. teoriaan ja havainnointiin, kun taas määrällinen tutkimus esimerkiksi tilastotieteisiin ja empirismiin. (Tuomi & Sarajärvi, 2009, 65 – 68.)

Laadullisen tutkimuksen yleisimpiä aineistonkeruumenetelmiä ovat haastattelu, kysely, havainnointi ja erilaisiin dokumentteihin perustuva tieto. Toisaalta samoja menetelmiä voidaan käyttää myös määrällisessä tutkimuksessa, mutta yleisesti aineiston määrä ja sen analyysi erottavat tutkimusmenetelmät toisistaan. (Tuomi & Sarajärvi, 2009, 71.)

Tutkimusmenetelmäksi valittiin laadullinen eli kvalitatiivinen tutkimus, jossa pyritään ymmärtämään tutkimusteorian ja tutkimusaineiston (tulosten) avulla tarkemmin ohjelmistokehitysprosessia ja analysoimaan toteutetun kehitysprojektin soveltuvuutta toimeksiantajan tuotekehitystoimintaan. Määrälliseen eli kvantitatiiviseen tutkimukseen aineistoa ei ole riittävän kattavasti, joka myös tukee kvalitatiivisen tutkimusmenetelmän käyttöä.

Tutkimusteoria kootaan ohjelmistokehityksen ja ohjelmistotuotannon kirjallisuudesta ja internetartikkeleista. Tutkimusteoria tukee kehitysprojektissa käsiteltävien käsitteiden ja toteutettujen menetelmien ymmärtämistä, sekä toimii osittain vertailukohtana tutkimustulosten käsittelyssä.

2 Tehtävienhallintaprosessit tuotekehityksessä

Puhuttaessa tuotekehityksestä voidaan tarkoittaa monen alan määrittely- ja suunnittelutyötä, kuten muotoilua, mekaniikkasuunnittelua, dokumentaatiota, käyttäjäkokemusta tai ohjelmistoa. Tämän kappaleen ei tulisi rajata suunnittelumalleja ja prosesseja ainoastaan ohjelmistokehitykseen, vaikka käsittelyssä onkin yleisesti juuri ohjelmistokehityksessä käytettävät menetelmät.

2.1 Ohjelmistoprojektit

Ennen tutustumista ohjelmistokehityksen malleihin ja prosesseihin, on hyvä käydä läpi keskeisiä asioita ohjelmistoprojekteissa ja niiden suunnittelussa.

Lehtimäen, 2006, mukaan projektien johtaminen kattaa yksinkertaisuudessaan kolme pääkohtaa; listaa tehtävät, tehkää tehtävät, tarkista, että tehtävät tulivat tehdyksi. Tehtävien listaamisella hän viittaa projektisuunnitteluun ja mainitsee, että myös projektisuunnittelua tulisi suunnitella.

Hyvä projektisuunnittelumalli pakottaa keskittymään tärkeimpiin asioihin, mutta nekin voi joko ohittaa kokonaan tai tehdä huolimattomasti. Lehtimäki, 2006, listaa neljä kohtaa, joihin projektisuunnittelussa tulisi keskittyä:

1. Mitkä ovat projektin tehtävät
2. Kuka tehtävät tekee
3. Milloin tehtävät tehdään

4. Mitä tehtävistä syntyy

Tehtävistä puhuessaan hän tarkoittaa suurempia kokonaisuuksia (enintään 40 tuntia), joiden työmääräarvioiden perusteella pystytään mm. arvioimaan projektin kustannukset ja seuraamaan projektin edistymistä. Myöhemmin käsiteltäessä tehtävienhallintaa, viitataan huomattavasti pienempiin kokonaisuuksiin, jotka ovat johdettu näistä suuremmista ”epiceistä”.

Se, kuka tehtävät tekee, vaikuttaa tehtävän työmääräarvioon. Näin ollen työmääräarvio tulisi tehdä tehtävän tekijä. Projektipäällikön tehtävä on kartoittaa tehtävien työmääräarviot ja osaamisvaatimukset, sekä tekijöiden (esim. tiimin) osaamistaustan ja halun, joiden perusteella se voi jakaa tehtävät tekijöille. (Lehtimäki, 2006, 17.)

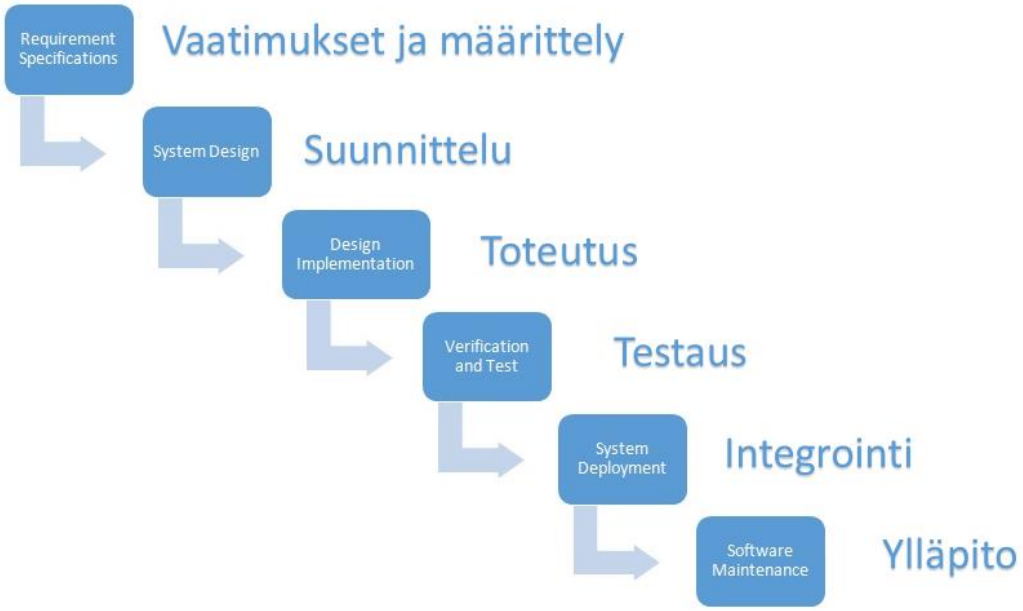
Aikataulutettaessa projektia tulee ottaa huomioon, ettei täysipäiväinen resurssi (henkilö) yleensä ole täysipäiväinen. Projektin tehtäville voidaan varata 80–90%, jolloin täydestä 100 prosentista loppu aika kuluu lomiin, sairastamiseen, koulutuksiin ja projektin ulkopuolisiin tehtäviin. Ns. tehtävien deadline tulisi myös kommunikoida arvioitua valmistumisaikaa myöhemmäksi, jolloin tiimillä on oma sisäinen tavoiteaika, joka sen on mahdollista saavuttaa pienellä ponnistelulla ja puolestaan projektilla realistinen valmistumisaika. (Lehtimäki, 2006, 17.)

Projektin tärkein tehtävä on tuottaa lopputuloksia. Jo projektin suunnittelusta alkaen on selvitettävä tarkasti kaikille osapuolille, mitkä ovat projektin lopputulokset eli tuotokset ja milloin. Etenkin ohjelmistojen osalta on tärkeä määritellä tarkasti, mitä syntyy ja minne. Mikäli tuotokset eivät ole kommunikoitu tarkasti, voi siitä toteutus- tai julkaisuvaiheessa tilaajalla ja toimittajalla olla eri käsitys, eivätkä projektin tulokset ole haluttuja. (Lehtimäki, 2006, 23.)

Ohjelmistoista on tullut niin suuria ja monimutkaisia, etteivät yhdet aivot enää pysty jäsentämään ja ottamaan huomioon kaikkia yksityiskohtia. Tämän vuoksi ohjelmistokehitykseen vaaditaan työryhmiä eli tiimejä. Ihmismäärän kasvu lisää myös kommunikaatiopolkuja eksponentiaalisesti, mikä puolestaan lisää kompleksisuutta. Kompleksisuuden vastapainoksi tarvitaan säännöllisyyttä ja järjestystä, joita tuodaan yrityksiin struktuurilla eli esimerkiksi määrittämällä hierarkioita, rooleja ja prosesseja. (Auer, ym. 2013, 10.)

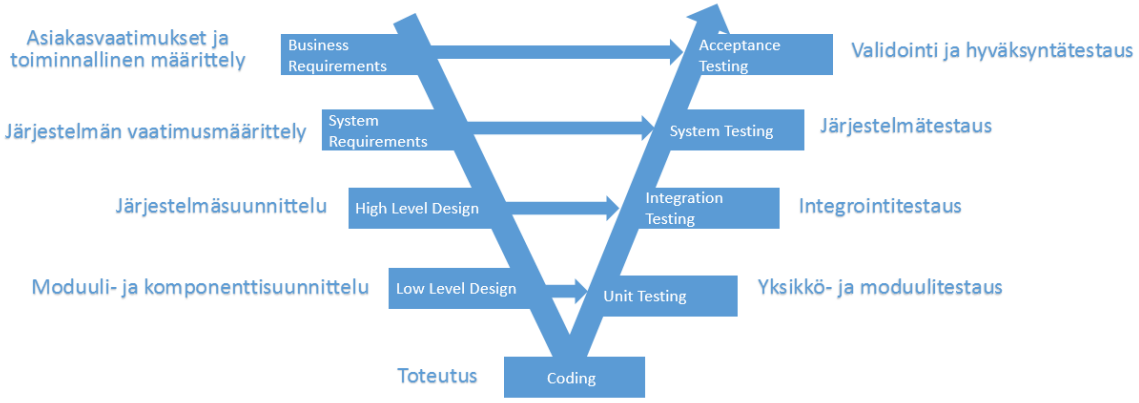
2.2 Ohjelmistokehityksen mallit ja prosessit

Ohjelmistokehitys kattaa koko ohjelmiston elinkaaren sen kehittämisen aloittamisesta sen poistamiseen käytöstä. Elinkaari voidaan jakaa vaiheisiin erilaisilla malleilla, joista yksi perinteisimmistä on vesiputousmalli (waterfall model, ks. Kuvio 2). Kukin vaihe sisältää tietyn määrän tarkastuksia, testausta ja katselmointeja, joiden avulla mahdolliset virheet pystytään havaitsemaan varhaisessa vaiheessa ja siirtymään seuraavaan vaiheeseen vasta kun määritellyt tavoitteet on saavutettu. (Haikala & Märijärvi 2004, 36 - 37.)



Kuvio 2. Vesiputousmalli

Vaihtoehtoinen, testauspainotteinen malli kuvata ohjelmistokehityksen elinkaarta on uudempi ja myös yleisesti käytössä oleva V-malli (V-model, ks. Kuvio 3). V-malli tulee sanoista validointi- ja verifiointimalli ja, samoin kuin vesiputousmalli, sisältää sarjan ohjelmiston elinkaaren vaiheita, jotka käydään läpi vaihe kerrallaan. Poikkeavuutena vesiputousmalliin, V-mallissa testaus esitetään rinnakkaisena toimenpiteenä muihin vaiheisiin.



Kuvio 3. V-malli

Koska tietyt turvakriittiset sovellukset edellyttävät ohjelmiston testausta, validointia, toteutuksen jäljitettävyyttä vaatimuksiin ja näiden dokumentointia, suosittelee standardi EN ISO 13849 V-mallin käyttöä ko. sovellusten toteutuksessa. (Hedberg, ym. 2011, 55.)

Toiminnallinen määrittely aloittaa ohjelmiston elinkaaren, jonka jälkeen asiakasvaatimuksista johdetun järjestelmän vaatimusmäärittelyn (SyRS – System Requirements Specification) ja/tai käyttötapausten perusteella tehdään järjestelmätason testaussuunnitelma. Järjestelmätason testauksella pyritään todentamaan vaatimusmäärittelyn vaatimukset. Käyttötapauksella (use case) kuvataan palvelun, tuotteen, käyttöliittymän ja käyttäjän välinen vuorovaikutus.

Järjestelmäsuunnittelussa keskitytään järjestelmän arkkitehtuurin suunnitteluun ja tuotetaan suunnitelma mm. käytettävästä alustasta, järjestelmästä ja linkityksistä eri ohjelmakomponenttien ja toimintojen, kuten sähköjärjestelmän välillä. Järjestelmäsuunnitteluvaiheessa tuotetaan integrointitestaussuunnitelma, jonka tarkoituksena on varmistaa toteutuksen yhteensopivuus muun järjestelmän kanssa.

Moduuli- ja komponenttisuunnittelussa määritellään nimensä mukaisesti ohjelmistomoduulit ja ohjelmistokomponentit. Tässä vaiheessa ohjelman varsinainen toimintalogiikka määritellään ja tuloksena voi olla esimerkiksi tila- tai luokkakaavio ja tarkempi kuvaus ohjelman osien toiminnasta (toimintokuvaus, specification). Näiden

tuotosten perusteella voidaan määritellä komponentti- ja moduulitestit. Ohjelmistokomponentilla tarkoitetaan spesifisen ohjelmistokokonaisuuden osaa, kun taas ohjelmistomoduuli on yleiskäyttöinen ohjelmiston osa, jota voidaan käyttää useassa sovelluksessa.

Alin taso V-mallissa on itse ohjelmointi eli ohjelmistokomponenttien ja moduulien toteutus sovelluskehittäjien toimesta. Sovelluskehittäjä suorittaa toteutetuille ohjelmakomponenteille- ja moduuleille yksikkötestausta. Toteutusvaiheen jälkeen V-mallissa siirrytään yläoikealle ja suoritetaan testausvaiheet alhaalta ylös.

V-mallin hyötyjä ovat mm. helppokäyttöisyys, toteutusta edeltävä suunnitelmallisuus niin ohjelmistokomponenttien kuin testauksen osalta, sekä aikainen virheiden ja epäkohtien havaitseminen. V-mallin huonoin puoli on sen joustamattomuus; toteutusvaiheen muutokset vaatimuksiin pakottavat myös muutokset testausdokumentaatioon, eikä ohjelmiston aikaisen vaiheen prototyyppejä ole mahdollista saada, vaan toteutus tehdään vasta huolellisen suunnittelun jälkeen. V-malli sopeutuu parhaiten pieniin ja keskisuuriin projekteihin, joissa vaatimukset ovat selkeitä ja viimeisteltyjä (ISTQB Certification Exam, 2017.).

Hyvönen, 2003, mainitsee kirjassaan, että tuotekehitysympäristö voi olla välillä niin dynaaminen ja turbulenttinen, että hyvin yksityiskohtaisesti määriteltyjen kehitysprosessien noudattaminen voi olla tuotteen käyttöympäristöön kohdistuvien muutosten vuoksi hankalaa. Tällöin usein toimivin ratkaisu on useiden iteraatioiden, demojen ja prototyyppien valmistaminen.

2.2.1 Ketterä kehitys ja tiimityöskentely

Ketterää kehitystä (Agile development) voidaan pitää edellä mainittujen ohjelmiston elinkaarimallien rinnakkaisprosessina, eivätkä nämä ole toisiaan poissulkevia. Ketterän kehityksen menetelmät ovat ennemminkin työkalu ohjelmiston elinkaaren toteuttamiseksi.

1990-luvun alkupuolella keksittiin uudenlainen lähestymistapa ohjelmistokehitykseen nimeltä Rapid Application Development (RAD). Se yhdisteli raskaita kaavamaisia ohjelmistokehitystekniikoita maanläheisiin toimintatapoihin, kuten prototyyppikäy-

täntöihin, jatkuvaan ohjelmiston toimitukseen ja intensiiviseen yhteistyöhön asiakkaan kanssa. Tämän myötä alkoi kehittymään ensimmäiset kevyet ohjelmistokehitysmenetelmät. Myöhemmin 2000-luvun alussa sana ”kevyt” vaihtui sanaan ”ketterä” ja alettiin puhumaan ketteristä eli Agile-kehitysmenetelmistä. Tuolloin esiteltiin myös Agile manifesti. (Appelo, 2011, 20.)

Cohen, 2010, esittelee Agile manifestin kirjassaan seuraavasti:

We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

- *Individuals and interactions over processes and tools*
- *Working software over comprehensive documentation*
- *Customer collaboration over contract negotiation*
- *Responding to change over following a plan*

That is, while there is value in the items on the right, we value the items on the left more.

Ketterä kehitys on monimuotoinen käsite, joka kattaa niin arkkitehtuuria, työkaluja, ihmisten roolituksia, sekä työn ja ihmisten organisoitumistakin. (Auer, ym. 2013, 14.)

Yleisesti ohjelmisto halutaan ja voidaan toteuttaa ja julkaista toiminnallisuus kerrallaan, jolloin on luonnollista muodostaa tiimejä jotka voivat toteuttaa toiminnallisuuksia yksi kerrallaan. Tiimit ovat itseohjautuvia ja niiden sisäinen kommunikaatio ja ohjelmiston toiminnallisuuksien välisen integraation sopiminen toteutuvat sujuvasti. (Auer, ym. 2013, 11.)

Tiimin koko voi vaihdella yleensä 2 – 20 tiimijäsenen välillä. Tiimin pienellä ja suurella koolla on havaittu positiivisia ja negatiivisia ominaisuuksia. Suuren tiimin etuna voi olla, että se sisältää kattavammilla taidoilla, kokemuksilla ja näkemyksillä varustettuja jäseniä, eikä jäsenen menettäminen ole niin suuri riski kuin pienessä tiimissä. Pienempien tiimien eduksi voidaan puolestaan laskea esimerkiksi pienempi todennäköisyys sille että jäsenet ajattelevat, että joku muu hoitaa meneillään olevat tehtävät, rakentavampi ja tiiviimpi kommunikointi tiiminjäsenten välillä, vähäinen koordinointiin vaadittava aika ja pientiimin toimintatapojen mielekkyys. (Cohn, 2009, 178 – 179.)

Kuten muissakin malleissa, ketterässä kehityksessä kehitettävä ohjelmisto tulee jakaa toiminnallisuuksiin, joiden kehittämis- ja julkaisujärjestys tulee suunnitella. Suunnittelua voidaan verrata priorisointiin ja yleisesti tämän tuotoksena on järjestelty toiminnallisten vaatimusten kehitysjono eli backlog, jonka perusteella tiimit kehitystyötään toteuttavat. Tuotteen kehitysjonon omistaa ja priorisoi usein lopulta liiketoiminnan omistaja. Priorisoinnissa ja kehitysjonon suunnittelussa tulee kuitenkin huomioida tekniset aspektit ja toiminnallisuuksien väliset riippuvuudet, jotka voivat olla tiedossa esimerkiksi vain kehitystiimillä.

Kehitysjonon alkiot voivat olla monimuotoisia, pienistä nopeasti toteutettavista toiminnoista suuriin ”tarinoihin” (epic, story). Tarinat ovat ohjelmiston toiminnallisuuksien kuvauksia luonnollisella kielellä, esimerkiksi "As a <role>, I want <goal/desire> so that <benefit>". Kehitysjonon alkiot voivat olla myös esimerkiksi käyttötapauksen skenaarioita, käyttöliittymän ruutuja tai ei-toiminnallisia vaatimuksia. Käytännössä alkiot voivat olla mitä tahansa tarpeeseen sopivia tehtäviä asioita. (Auer, ym. 2013, 70.)

Tärkeintä on, että kehitystiimillä on ymmärrys toteutettavasta ominaisuudesta ja sen tuomasta lisäarvosta. Kehitysjonoa tulisi työstää säännöllisesti ja käyttää siihen vähintään 5 % kulloisenkin kehitysperiodin työajasta. Kun yhteinen ymmärrys kehitysjonon alkion toteuttamisesta on saavutettu, se voidaan jakaa pienempiin tarinoihin kehitystiimin avustuksella. Tarinoista tulisi käydä ilmi em. esimerkin mukaisesti rooli, toiminnallisuus tai määränpää ja siitä saatava lisäarvo. Tarinalle tulee olla määriteltynä hyväksyntäkriteerit (DoD – Definition of Done), joiden perusteella tiimi tietää milloin tarina on valmis. Näistä hyväksyntäkriteereistä voidaan suoraan johtaa testitapaukset, jolloin testausdokumentaatio tai testausautomaatio voivat olla valmiina ennen varsinaista toteutusta. Näiden testien suorittaminen hyväksytysti varmentaa tarinan olevan valmis. (Auer, ym. 2013, 74.)

Braude & Bernstein, 2011, listaavat kirjassaan yhteenvetona kolme kohtaa, joista jokaisen käyttäjätarinan tulisi koostua; kirjoitettu kuvaus, asiakkaan kanssa käyty keskustelu tarinan tarkoituksen ja sisällön ymmärtämiseksi, sekä vaadittavat testit, joilla tarina validoidaan.

Kehitysjonon alkioilla voi olla riippuvuuksia toisiinsa (tai muihin aspekteihin), jotka tulee tiedostaa ja ottaa huomioon ennen tarinoiden estimointia eli toteutusarviota varten. Esimerkiksi toisen alkion toteutuksen puuttuminen voi estää toisen alkion toteutuksen, jolloin vaaditaan myös estävän alkion toteutus. Tällöin tarinoita voidaan joutua pilkkomaan pienempiin kokonaisuuksiin. Kun tarinoiden riippuvuudet on todettu ja kirjattu, voidaan tarinat estimoida työmäärän ja kompleksisuuden mukaan. Yleinen tapa on pisteyttää tarina numeroin (esimerkiksi 0, ½, 1, 2, 3, 5, 8, 13, 20, 40 ja 100). Pistemäärät voivat tarkoittaa eri tiimeillä eri asioita, joka johtuu tiimin suorituskyvystä. Pisteytys toteutetaan kehitystiimin jäsenten toimesta ja jokaisen tulisi pystyä perustelemaan antamansa pistemäärä. Mikäli ristiriitoja syntyy, niistä keskustellaan ja tarinaa mahdollisesti uudelleenarvioidaan, kunnes saavutetaan yhteisymmärrys tarinan koosta. Tarinoiden estimoinnin jälkeen kehitysperiodissa toteutettavan työn suunnittelu on mahdollista. (Auer, ym. 2013, 75.)

Muutostenhallinta on tärkeä osa ketterää kehitystä ja se mielletään hyvinkin erilaiseksi kuin perinteinen tapa, jonka mukaan muutospyyntöjä voidaan toteuttaa sitä mukaa, kun niitä tulee. On tärkeää tarkastella asioita yksittäisen kehitysperiodin ulkopuolelta arvon näkökulmasta eikä vain toteuttaa mitä tilataan. Ensisijaisesti tulisi panostaa työn alle otettavaan ominaisuuteen riittävästi, jotta se voitaisiin suunnitella ja toteuttaa yhdellä kertaa halutun laiseksi, eikä samoja ominaisuuksia tarvitsisi muuttaa toistuvasti. (Auer, ym. 2013, 76.)

Edellä mainitun vuoksi asiakas onkin avainasemassa ketterän kehityksen ohjelmistokehitysprosesseissa. Asiakas tulisi ennemmin mieltää tiiminjäsenenä kuin itsenäisenä tahona, jonka kanssa ainoastaan sovitaan mitä kehitetään milloinkin ja mitä jätetään kehittämättä. Monissa tapauksissa asiakas on myös ryhmä, joka evaluoi kehitetyn sovelluksen, jonka vuoksi asiakkaan läsnäolo ohjelmistokehityksessä on tärkeää. (Hazzan & Dubinsky, 2008, 45.)

Yksi ketterän kehityksen haasteita on ns. kehitysvelka eli tekninen velka, joka aiheutuu kun tuotetta kehitetään nopeasti ominaisuus kerrallaan, jolloin kehitys voi laadullisesti rappeutua ja jatkokehittäminen on yhä haastavampaa. Tätä ja muita haasteita varten tiimin tulisikin pystyä kehittämään toimintaansa jatkuvasti ja käydä näitä asioita läpi esimerkiksi retrospektiivi-palavereissa. Toinen haaste on ns. roikkuvat hännät eli tarinat ja tehtävät, jotka jatkuvat kehitysperiodista toiseen. Mikäli tehtävä jää

kesken tai toteuttamatta kehitysperiodin aikana, se tulisi siirtää seuraavaan kehitysperiodiin, mikäli edelleen nähdään tehtävän tuottavan lisäarvoa. Tämä kuitenkin voi sumentaa tiimin todellista kehitysnopeutta ja vaikeuttaa seuraavien kehitysperiodien suunnittelua, joista seuraa ennustettavuuden heikentyminen. Mikäli kehitystiimiltä jää jatkuvasti roikkuvia häntiä esimerkiksi estimoinnin epäonnistumisen tai osaamisen vuoksi, tulisi sen toimintaa jotenkin korjata. Korjaavia toimenpiteitä voivat olla esimerkiksi kehitysperiodien kokonaistyömäärän pienentäminen, parityöskentely erityisosaajien kanssa ja katselmointikäytänteiden parantaminen. (Auer, ym. 2013, 77 – 80.)

2.3 Menetelmiä ketterään kehitykseen

2.3.1 Scrum

Scrum on 1990-luvun alusta asti käytetty viitekehys, jonka avulla ihmiset voivat käsitellä monimutkaisia mukautuvia ongelmia samalla, kun tuottavat tehokkaasti ja luovasti korkeimman painoarvon tuotteita.

Scrumin osakehittäjien Schwaberin ja Sutherlandin (2013) mukaan scrum on kevytrakenteinen, helppo ymmärtää, mutta hankala hallita. Scrum sisältää tiimejä ja heihin liittyviä rooleja, tapahtumia, mittareita (artefakteja) ja sääntöjä. Jokaisella osalla on tärkeä merkitys ja ovat tarvittavia Scrumin käytön ja menestymisen kannalta.

Scrum-prosessin kolme kivijalkaa ovat:

Läpinäkyvyys

Prosessin merkittävät näkökulmat tulee olla näkyvissä tuotoksesta vastaaville. Tämä vaatii, että nuo näkökulmat tulee olla selkeästi määriteltynä. Näitä voivat olla esimerkiksi prosessissa käytettävä kieli ja valmiin tuotoksen määritelmä (Definition of Done, DoD).

Tarkastelu

Käyttäjien tulee jatkuvasti tarkastella Scrumin mittareita (artefakteja) ja tuotosten valmistumisasteita havaitakseen ei toivottuja muutoksia. Tarkastelu ei voi kuitenkaan olla niin tarkkaa, että se estää itse työn suorittamisen.

Sopeutuminen

Mikäli tarkastelussa havaitaan, että yksi tai useampi prosessin näkökulma poikkeaa liikaa ja tuotos ei tule olemaan hyväksyttävissä rajoissa, tulee prosessia tai työstettävää tuotetta muuttaa hyvissä ajoin. Näin pyritään minimoimaan jatkossa tapahtuvat poikkeavuudet.

Menetelmä määrittelee Scrum-tiimin, johon kuuluvat tuotteenomistaja (Product Owner, PO), kehitystiimi (Development Team) ja scrummaster (Scrum Master). Scrum-tiimi on itseohjautuva ja monitaitoinen, joka pystyy itsenäisesti päättämään kuinka parhaiten saavuttaa määränpäänsä ja sen jäsenet pystyvät itsenäisesti toimimaan riippumatta muista tiimin jäsenistä. Tiimimallin on tarkoitus olla optimaalisen joustava, luova ja tuottelias.

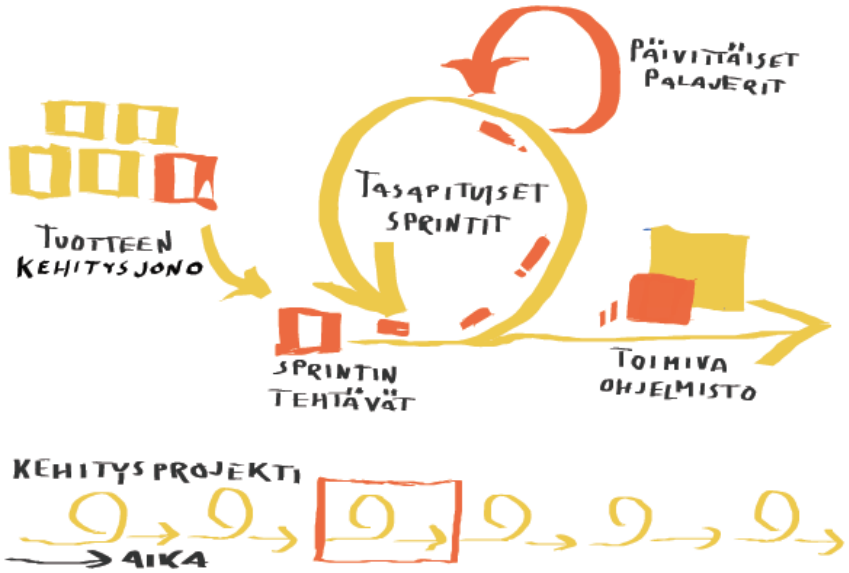
Tuotteenomistaja on yksi henkilö ja hänen rooli on maksimoida kehitettävän tuotteen ja kehitystiimin tekemän työn arvo. Tähän työkaluna hän käyttää tuotteen kehitysjonon järjestelyä. Kehitysjonon tulee olla järjestelty niin, että parhaiten saavutetaan halutut määränpäät, kehitystiimin työskentely on optimoitua ja että kehitysjono on läpinäkyvä, selkeä ja helposti kehitystiimin ymmärrettävissä.

Kehitystiimi suorittaa nimensä mukaisesti kehitystyön eli tuottaa jokaisen kehitysjonon alkion eli tuotteen inkrementin. Kehitystiimillä on osaaminen tuottaa jokainen inkrementti ja se päättää itsenäisesti kuinka nuo määränpäät parhaiten saavutetaan. Muut Scrum-tiimin jäsenet eivät voi vaikuttaa kehitystiimin toteutukseen. Kehitystiimin koko tulee olla tarpeeksi pieni ollakseen ketterä toiminnoissaan ja tarpeeksi suuri pystyäkseen suorittamaan jokaiseen inkrementtiin suunniteltu työ.

Scrummaster on vastuussa, että Scrumin käytänteet on ymmärretty ja niitä noudatetaan. Scrummaster takaa Scrum-tiimille työrauhan, pyrkii poistamaan työn esteet ja kommunikoi ympäristölle, kuinka nämä saavutetaan. Scrummaster auttaa tuotteenomistajaa kehitysjonon tehokkaassa ja selkeässä käytössä ja kehitystiimiä puolestaan valmentamalla sitä tehokkaaseen työskentelyyn, poistamalla työn esteitä ja järjestämällä Scrumiin liittyviä tapahtumia. (Schwaber & Sutherland 2013, 3 - 7)

Scrumissa tuotteen inkrementtejä kehitetään vähitellen noin 1–4 viikon mittaisten iteraatioiden eli sprinttien aikana (ks. kuvio 4). Inkrementin työtehtäviä hallinnoidaan

iteraation eli sprintin kehitysjonolla. Sprintin aikana kehitystiimi (ja mahdollisesti scrummaster) pitää päivittäisiä tilannepalavereita (päivittäinen Scrum, Daily Scrum), joiden tarkoituksena on päivittää kehitystiimin jäsenten työn eteneminen ja poistaa siinä mahdollisesti esiintyvät ongelmat ja esteet. Päivittäisen scrumin tarkoitus ei ole paneutua teknisiin yksityiskohtiin. Jokaisen sprintin alussa järjestetään suunnittelupalaveri, jossa Scrum-tiimi yhteisesti valitsee kehitysjonon alkiot, jotka se sitoutuu sprintin aikana toteuttamaan. Sprintin tuotokset katselmoidaan sprintin lopuksi katselmointipalaverissa (sprintin katselmointi, Sprint Review), jossa esitellään toimiva tuote sprintin aikana valmistuneiden ominaisuuksien eli inkrementin kautta. Lisäksi Scrum-tiimi kehittää omaa työtään jokaisen sprintin lopuksi pidettävällä retrospektiivillä (Sprint Retrospective), jossa Scrum-tiimi käy läpi edellistä toteutettua sprinttiä ja miettii menetelmiä, kuinka tiimin työskentelytapoja voisi kehittää entistä paremmiksi. (Lehtonen, ym. 2014, 4)



Kuvio 4. Scrum-prosessi (Lehtonen, ym. 2014, 5)

2.3.2 Kanban

Kanban on peräisin autovalmistaja Toyotan käyttämästä termistä, joka visualisoi koko Lean-ajatteluun pohjautuvan tuotantoketjun (ks. kuvio 5.). Kanban tarkoittaakin japaninkielellä visuaalista korttia. Kanbanin ohjelmistokehitykseen toi David Anderson vuonna 2004. (Kanban, 2017)

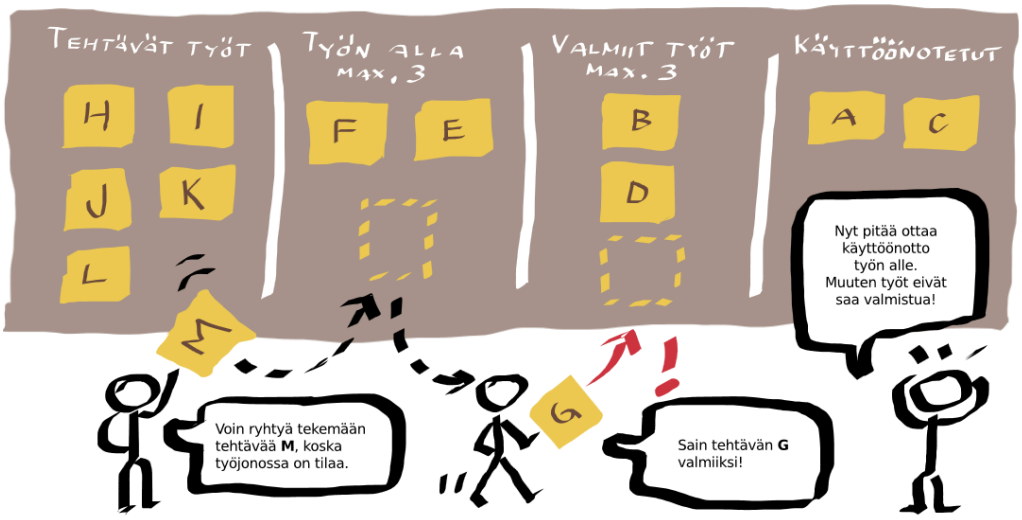
Kanban määrittelee ainoastaan kolme sääntöä:

1. Visualisoi työ
    a. Pilko työ sopivankokoisiin tehtäviin
    b. Kirjaa tehtävät
    c. Visualisoi tehtävien työvaihe

2. Rajoita tehtävien määrä (WIP, Work In Progress)
    a. Tehtävien määrä tulee rajata ko. työvaiheessa siten, ettei se hidasta muita työvaiheita

3. Mittaa työn läpimenoaika (Lead Time)
    a. Läpimenoaika mittaa yhden tehtävän valmistumiseen kuluvaa aikaa
    b. Mittarin avulla optimoidaan WIP-rajoitus läpimenoajan lyhentämiseksi ja ennustettavuuden parantamiseksi

Kanbanin määritelmä on Scrumia huomattavasti suppeampi, eikä se käytännössä määrittele kuin em. kolme sääntöä. Kanban ei määrittele kehitysjaksoja tai palavereita, vaan prosessi on jatkuva ja palavereita ja ohjelmistojulkaisuja toteutetaan aina kun se on tarpeellista ja mahdollista. Kanban-menetelmä sopii parhaiten projekteihin, joissa työn sisältö muuttuu usein tai muutoin työn muutoksia on vaikea ennustaa. (Lekman, 2009)



Kuvio 5. Kanban-prosessi (Lehtonen, ym. 2014, 9)

2.3.3 Muut menetelmät

Muita menetelmiä ja näiden yhdistelmiä ketterään kehitykseen on useita ja seuraavassa on esitelty lyhyesti muutama.

Extreme Programming (XP)

XP perustuu viiteen keskeiseen ominaisuuteen: viestintä, yksinkertaisuus, palaute, kunnioitus ja rohkeus. Ohjelmistokehittäjien ja asiakkaan välillä on jatkuva kommunikaatio, joka parantaa viestinnän toimivuutta. Ohjelmakoodin rakenne pidetään mahdollisimman yksinkertaisena ja vaikeiden kokonaisuuksien kuvaamiseen käytetään metaforia. Ohjelmaa testaan alusta alkaen ja toimitetaan asiakkaalle toimiva versio mahdollisimman aikaisessa vaiheessa, jolloin saadaan toteutuksesta palaute nopeasti ja pystytään siihen myös nopeasti reagoimaan. Jokaisen onnistuneesti suoritetun iteraation jälkeen kehittäjien kunnioitus ja luottamus toisiinsa lisääntyy ja täten pystyvät rohkeammin reagoimaan tuleviin muutoksiin. (Astels, ym. 2002, 4-8)

XP määrittelee myös lähteestä riippuen 10–15 yksinkertaista sääntöä suunnitteluun, johtamiseen, kehitykseen, ohjelmointiin ja testaukseen liittyen, joiden pohjalta menetelmää tulee toteuttaa. Stepanek, 2005, listaa seuraavat:

1. Suunnittelupeli
2. Testaus

3. Pariohjelmointi
4. Refaktorointi
5. Yksinkertainen suunnittelu
6. Yhteinen ohjelmakoodin omistajuus
7. Jatkuva integrointi
8. Läsnä oleva asiakas
9. Pienet julkaisut
10. 40-tuntinen viikko
11. Ohjelmointistandardit
12. Järjestelmän kielikuvien käyttö

Crystal-metodi

Crystal on Alistair Cockburnin luoma mukautuvaksi suunniteltu kehitysmenetelmä, jossa otetaan huomioon tiimien ja henkilöiden vaihtelevuus. Moni tiimi saavuttaa tavoitteensa, joskin eri tavalla. Crystal pyrkii olemaan yksinkertainen ja mukautuva säännöstö, jotka turvaavat projektin onnistumisen. Crystal onkin menetelmistä samalla eniten ja vähiten kuvaileva. Sen sijaan, että kehittäjät tekisivät etukäteen valmiiksi suunniteltuja tehtäviä toisensa perään, pyritään heitä kannustamaan käyttämään heidän aloitteellisuutta ja joustavuutta mukautumaan aina muuttuvaan tilanteeseen. (Stepanek 2005, 67 – 68.)

Crystal määrittelee kehitystiimeille värin tiimin koon mukaan; 2 – 8: kristallinkirkas, 9 – 20: kristallinkeltainen, 21 – 50: kristallinoranssi jne.. Koko määritellään, koska se vaikuttaa tiimien ongelmanratkaisutapoihin. Kaikille väreille yleisiä ominaisuuksia ovat seuraavat:

1. Tiheä toimitus – esimerkiksi kehitysperiodeihin pohjautuva
2. Heijastava kehitys – tiimin käytänteiden ja toimintatapojen jatkuva kehittäminen
3. Läheinen tai osmoottinen kommunikaatio – pienet tiimit ovat samassa tilassa ja suuremmat mahdollisimman lähekkäin
4. Henkilökohtainen turvallisuus – psykologinen turvallisuus takaamaan kehittäjien työrauhan
5. Keskittyminen – niin yksittäiseen tehtävään kuin suuremman määränpään saavuttamiseen
6. Helppo pääsy asiantuntijoihin – kehittäjät saavat näin palautetta ja tukea avonaisiin kysymyksiin
7. Tekninen ympäristö, ml. automaattinen testaus, konfiguraatiohallinta ja jatkuva integrointi (Stepanek 2005, 68 – 73.)

Lean Software Development (LSD)

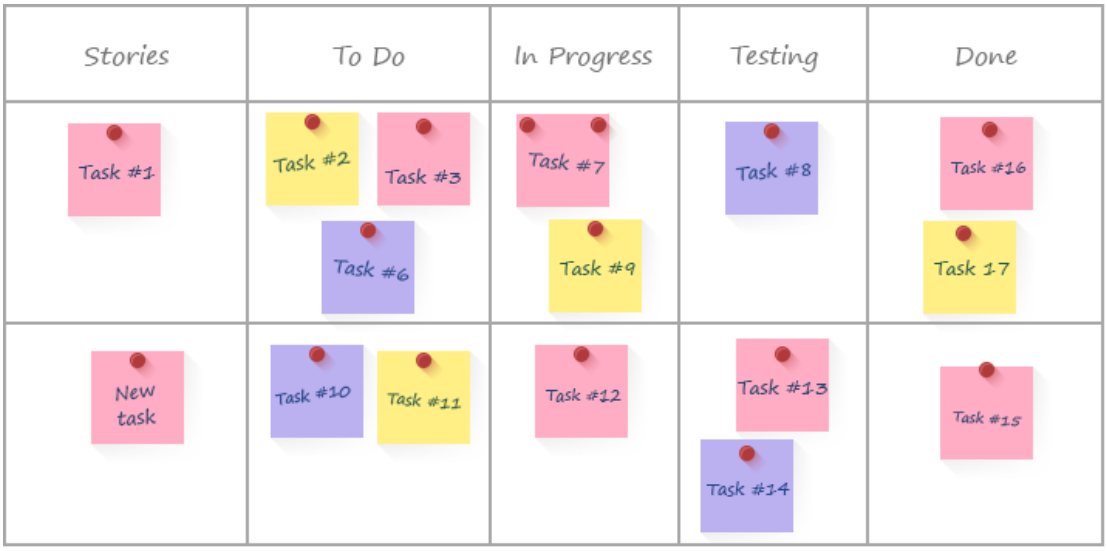
Lean-ohjelmistokehitysmenetelmä pohjautuu Japanin autoteollisuuteen keksittyyn Lean-menetelmään. Lean ei ole varsinaisesti menetelmä, vaan ennemminkin ajattelutapa, jota käytetään apuna prosessia kehittäessä. Ajattelutavan peruspilareita ovat mm. eliminoi hukka (kaikki, mikä ei tuo asiakkaalle lisäarvoa) ja ihmiskeskeisyys (keskittyminen ihmisiin, jotka tuovat lisäarvoa esimerkiksi arvostuksen, tehokkaan kommunikoinnin, hyvän työilmapiirin, koulutuksen ja toiminnan tehostamisen avulla). (Haikala & Mikkonen 2011, 54 – 55.)

2.4 Työkalut tehtävienhallinnassa

Ketterän kehityksen menetelmät, kuten Scrum ja Kanban voidaan mieltää työkaluina, mutta tässä yhteydessä puhuttaessa työkaluista, tarkoitetaan konkreettisia apuvälineitä, joiden avulla em. menetelmiä ja etenkin tehtävienhallintaa voidaan toteuttaa.

Tehtävienhallintatyökaluja on tarjolla ja niitä käytetään lukuisiin tarkoituksiin, kuten tehtävien ja tukipyyntöjen hallintaan (issue tracking), virheiden eli bugien hallintaan (bug tracking) ja ylemmillä tasoilla projektien- ja tuotannonhallintaan (ERP-järjestelmät).

Tehtävienhallintaan liittyy keskeisinä osina kehitysjonon hallinta ja esittäminen, työvaiheiden esittäminen, sekä tehtävien suoritukseen liittyvien mittareiden esittäminen. Työkalut voivat olla yksinkertaisimmillaan paperilappuja, jotka sisältävät tehtäviä ja niitä siirretään työvaiheiden välillä (ks. kuvio 6). Laajimmillaan työkalu voi olla suuren organisaation usealla toimipisteellä käytössä oleva ohjelmisto, joka integroi koko tuotteen tiedot projektisuunnitelmista vaatimuksiin, vaatimuksista tehtäviin ja tehtävistä toteutukseen.



Kuvio 6. Esimerkki yksinkertaisesta tehtävienhallintatyökalusta

Tehtävienhallintaan kehitetyt ohjelmistot yleensä sisältävät tai mahdollistavat myös muita ohjelmistokehityksen hallintaan ja menetelmiin liittyviä toimintoja, kuten dokumentointityökaluja, katselmointityökaluja, raportointipalveluita, Scrum-työkaluja yms.. Ohjelmistojen näkymiä ja toimintoja voi myös muuttaa tarpeisiin sopiviksi. Tehtävienhallintatyökalujen toimintoja käydään tarkemmin läpi esimerkin kautta työn toteutusvaiheessa.

Kaupallisia tehtävienhallintatyökaluja ovat mm. TargetProcess, Trello, Jira, dapulse, Aiveo ja Polarion (Top Project Management Tools. 2017.). Kaikilla työkaluilla on eri määrä toiminnallisuuksia, jotka vaikuttavat mm. työkalun käytettävyyteen ja kustannuksiin, jonka vuoksi työkalu tuleekin valita yrityksen tarpeiden mukaisesti. Toimeksiantajan toimesta oli jo aiemmin tehty evaluointia eri työkalujen välillä ja valittu ominaisuuksien, laajennusmahdollisuuksien ja kustannuksien vuoksi Jira.

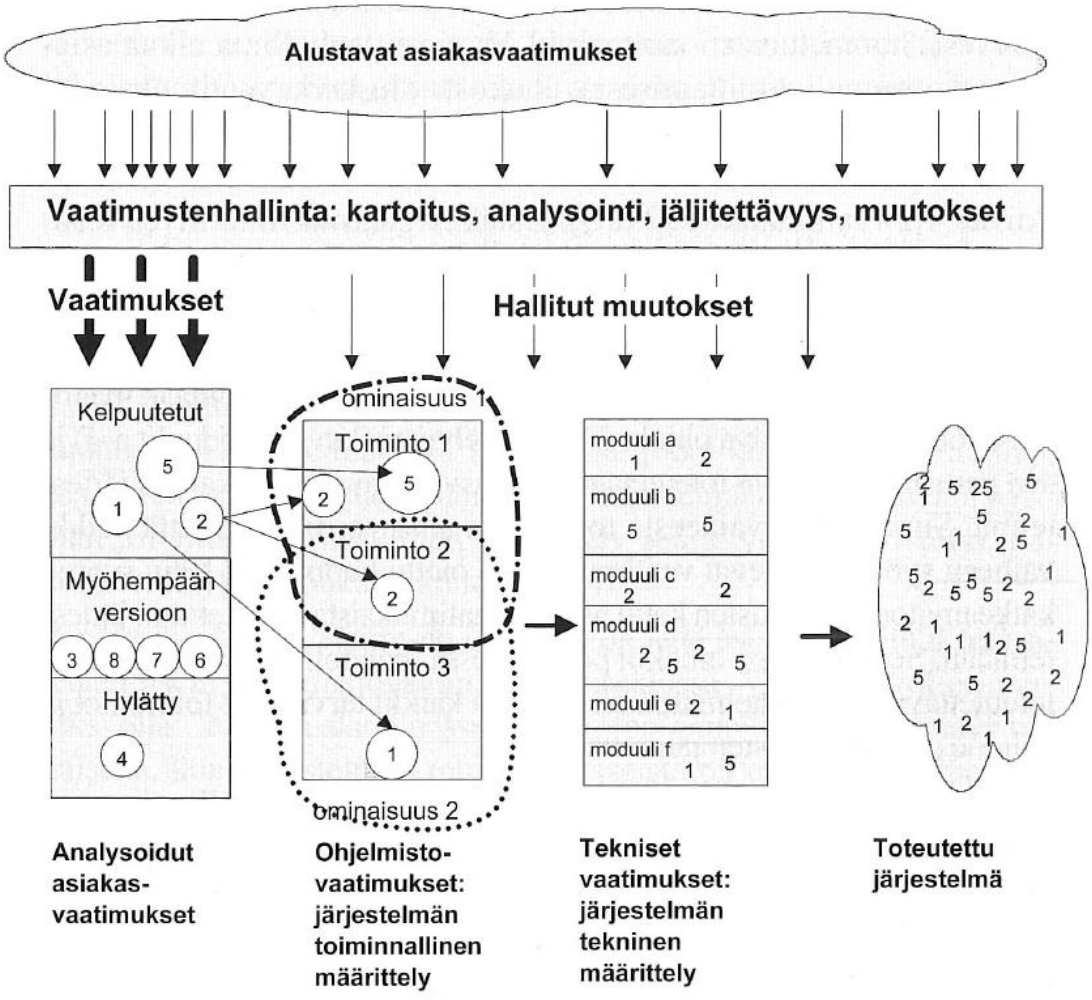
Jira on kokonaisvaltainen tehtävienhallintaohjelmisto (Issue Tracking System), jonka on kehittänyt australialainen vuonna 2002 perustettu ohjelmistoyritys Atlassian. Työkalulla voidaan kirjata käyttäjätarinoita ja tehtäviä, suunnitella kehitysperiodeja ja jakaa tehtäviä kehitystiimeille. Sen avulla tiimien työtä voidaan priorisoida ja siitä voidaan keskustella läpinäkyvästi. Työkalun avulla voidaan lisäksi ylläpitää julkaistujen ohjelmistokomponenttien informaatiota, sekä tuoda raportteja tehtävistä ja tiimien

työskentelystä. Jiraan on integroitavissa useita lisäosia liittyen mm. dokumenttienhallintaan, katselmointiin ja viestipalveluihin. (JIRA Software, 2017.)

2.5 Vaatimustenhallinta

Riippumatta tavasta vaiheistaa ohjelmistokehitystä, vaatimustenhallinta liittyy oleellisena osana menetelmien ja prosessien läpivientiin. Menetelmien yhteinen tavoite on aina asiakasvaatimusten ja lopullisen tuotoksen kohtaaminen. Tämän määränpään saavuttamiseksi tarvitaan vaatimustenhallintaa.

Tavallisesti vaatimustenhallinta mielletään tuotekehitysprosessin alkuvaiheessa tehtäväksi määrittelyksi ja esisuunnitteluksi, mutta käytännössä sen tulisi olla erillinen toiminto, joka kattaa koko tuotekehitysprosessin aikaisen vaatimusten- ja muutostenhallinnan (ks. kuvio 7).



Kuvio 7. Vaatimustenhallinta erillisenä toimintona (Haikala & Märijärvi 2004, 93)

Vaatimustenhallinnan avaintekijänä on, että pystytään määrittelemään mahdollisimman oikein ymmärretyt ja muuttumattomat asiakasvaatimukset. Etenkin suuremmissa projekteissa kuitenkin on tavallista, ettei kaikkia vaatimuksia pystytä tunnistamaan tai määrittelemään projektin alkuvaiheessa, vaan ne ilmenevät tai muuttuvat sen aikana. Vaatimusten muutoksiin voi liittyä myös liiketoiminnallisia aspekteja, kuten markkinatilanteiden muutokset ja kustannus/hyöty-näkykulmat. (Haikala & Märijärvi 2004, 91 - 93.)

Vaatimuksiin kohdistuvat muutokset aiheuttavat muutoksia kaikkiin ohjelmistokehitysprosessin vaiheisiin, joten mitä myöhemmin muutoksia tehdään, sitä enemmän lisätyötä ne aiheuttavat.

Vaatimusten analysointi tapahtuu yleensä esitutkimusvaiheessa, mutta myös myöhemmissä vaiheissa muutostenhallinnassa. Analysoinnin tarkoitus on selvittää kunkin vaatimuksen tarve ja arvioida prioriteetti, sekä sovittaa ristiriitaiset vaatimukset. Vaatimusten priorisoinnilla päätetään, mitä vaatimuksia otetaan mukaan seuraavaan kehitettävään versioon (vrt. kehitysjono). Analysoinnilla voidaan myös arvioida vaatimuksen muutosherkkyyttä ja tällä tapaa määritellä vaatimukseen liittyviä riskejä.

Asiakasvaatimukset voidaan liittää ohjelmistovaatimuksiin käyttötapausten (use case) avulla. Käyttötapausten tarkoitus on kuvata käyttäjien tapaa käyttää järjestelmää ja järjestelmän toimintaa käyttötilanteissa. Vaatimusmäärittelyn ja käyttötapausten perusteella voidaan luoda ohjelmistovaatimukset ja toimintokuvaukset. (Haikala & Märijärvi 2004, 94 - 97.)

Asiakasvaatimusten tulee olla testattavia ja asiakasvaatimukset verifioidaankin vertaamalla syötedokumentteja (vaatimukset) sen tulosdokumentteihin (testitulokset). Ominaisuutta, jolla osoitetaan testien kautta, että toteutetut kohdat toteuttavat tietyn asiakasvaatimuksen sanotaan jäljitettävyydeksi (traceability). Jäljitettävyys voi olla tietyn tyyppisissä sovelluksissa omana vaatimuksenaan (esimerkiksi turvakriittiset sovellukset) ja tulla kattaa koko ominaisuuden elinkaaren varhaisesta asiakasvaa-

timuksesta tarkempiin määrittelyihin, toteutukseen ja testaukseen. Luvussa 2.4 mainituilla tehtävienhallintaohjelmistoilla on ominaisuuksia, joilla jäljitettävyyttä voi ylläpitää. (Haikala & Märijärvi 2004, 97 - 98.)

2.6 Ketterä kehitys ja liiketoiminta

Usein oletetaan, että ketterä kehitys toimii kustannuksien säästäjänä. Toisaalta sitä myydään työkaluna, joka nimensä mukaisesti ketteröittää kehitystä ja mahdollistaa muutoksiin mukautumista kehityksen aikana. Liiketoiminnalle arvokkaina asioina voidaan myös pitää läpimenoaikaa, asiakastyytyväisyyttä, laatua ja läpinäkyvyyttä, jota seuraa ohjattavuus ja ennustettavuus. Mikään ketterä menetelmä ei näitä suoraan kuitenkaan takaa.

Monilla (etenkin suuremmilla) yrityksillä voi olla raskaita päätöksentekomenettelyitä, jotka itsessään hidastavat läpivientiaikaa viivästyttämällä kehittämisen aloittamista. Tämä koskee niin kehittäjä- kuin asiakasyrityksiä.

Läpimenoaikaa lyhentää valmiina oleva kehitystiimi, eikä tällöin esimerkiksi resursoinnista aiheudu viivästyksiä. Läpimenoaikaa saadaan lyhennettyä myös kiinnittämällä kehitystiimin jäsenet yhteen tiimiin kerrallaan. Sama pätee myös muihin projekteihin osallistuviin, kuten tuoteomistajiin, joiden kapasiteetti voi olla useampi projekti, mutta kuitenkin rajallinen.

Yhtäaikaisten kehittämisien ja projektien määrä on syytä pitää läpimenokyvyn kokoisena. Kullakin kehittäjällä saa olla vain yksi työ kerrallaan käynnissä. Kehittämisiä tehdään enemmän, kun ne ovat peräkkäin sen sijaan, että niitä yritetään tehdä rinnakkain. Se, että asiat aloitetaan, ei takaa, että ne saadaan päätökseen.

(Auer, ym. 2013, 27)

Läpimenoaikaan vaikuttavat myös prosessien monimutkaisuus ja vaatimustenhallinta. Pitäisi pyrkiä välttämään turhien vaatimusten korkealle priorisointia ja tiukkaa ulkopuolista ohjaamista ja hyväksymiskäytänteitä, vaan pitää ohjaus relevanteissa ja tärkeissä asioissa, esimerkiksi varmistamalla, että kehitystiimeistä löytyy jo valmiiksi tarvittava osaaminen ja itseohjautuvuus.

Asiakastyytyväisyys tarkoittaa käytännössä oikeita tuotteita oikea-aikaisesti. Tätä ajatellen tulee ymmärtää, millä ominaisuuksilla asiakkaalle on suurin hyöty ja osata priorisoida tekeminen sen mukaisesti. Ketterä kehitys mahdollistaa varhaisen palautteen saannin ja muutoskyvykkyyden, jonka vuoksi asiakas tulisi olla mukana jo kehityksen alkuvaiheissa.

Laatua voidaan mitata monilla asioilla, mutta yleisesti sillä tarkoitetaan teknistä toimivuutta ja käyttökokemuksia. Nämä varmistetaan kyvykkäällä ja motivoituneella kehittämistiimillä ja lyhyillä kehittämisiteraatioilla, joiden tuotoksena on aina valmis ja testattu tuotos. Tiimin tulee jatkuvasti kehittää myös omaa toimintaansa laadun takaamiseksi. Menetelminä voidaan käyttää esimerkiksi yksittäisten tuotosten ja yleisten toimintatapojen katselmointia. Auer ym., 2013, kuvaa onnistunutta kehitysprojektia liiketoiminnassa esimerkiksi seuraavilla mittareilla:

- Sijoitetun pääoman tuotto
- Kehityksen ja ylläpidon kokonaiskustannukset
- Markkinoillemenoaika
- Liikevoitto ja markkinaosuus
- Tuotoksen laatu
- Aikataulu
- Käyttäjätyytyväisyys
- Syntynyt kehitysprosessi
- Organisaatiossa syntyneet ihmissuhteet

Järvinen ym., 2002, kuvaa onnistuneen projektin kriteereiksi samantyyppisiä kohtia:

1. Projekti on toteutettu määritellyssä laajuudessaan, aikataulun sekä budjetin mukaisesti.
2. Projektinhallinta ja –toteutus on hoidettu laadukkaasti ja tehokkaasti.
3. Projekti on saavuttanut sille asetetut tavoitteet.
4. Projekti on toteutettu yrityskulttuurin mukaisesti häiritsemättä muuta liiketoimintaa.
5. Projekti on ollut yritykselle hyödyllinen lyhyellä tähtäimellä (esimerkiksi voitto, markkinaosuus) tai pitkällä tähtäimellä (esimerkiksi valmistautuminen tulevaisuuteen).
6. Projektiyrityksen sisäiset osapuolet ovat tyytyväisiä lopputulokseen.
7. Asiakas on hyväksynyt projektin ja asiakas sekä käyttäjät ovat tyytyväisiä projektiin.
8. Suoritettua projektia voidaan käyttää referenssinä markkinointitarkoituksessa.

Ketterä kehittäminen mahdollistaa painopisteen siirtymisen menneisyyden seuraamisesta tulevaisuuden ennustamiseen. Koska toimivaa ohjelmistoa syntyy jo varhaisessa vaiheessa, voidaan tuotteen valmiusaste ja tiimin kyvykkyys todeta ja sitä

kautta tehdä luotettavasti ennustuksia tulevien ominaisuuksien osalta. Ohjattavuudesta ja ennustettavuudesta syntyy toiminnan läpinäkyvyys. (Auer, ym. 2013, 26 - 29.)

2.7 Lähteiden luotettavuus

Työn tietolähteenä on käytetty enimmäkseen ohjelmistokehityksen ja ohjelmistotuotannon kirjallisuutta ja internetartikkeleita. Kirjallisuus koostuu niin kansallisista kuin kansainvälisistä tietokirjateoksista, joissa asiantuntijat avaavat tai tutkivat ohjelmistokehitykseen ja ohjelmistotuotantoon liittyviä toimintamalleja ja käsitteitä.

Kirjallisuuslähteiden luotettavuutta parantaa se, että ne ovat niin kansallisia kuin kansainvälisiä. Informaation oikeellisuutta pystytään varmentamaan vertaamalla informaatiota muiden samaa asiaa käsittelevien kirjojen vastaavaan aineistoon. Aineisto on pyritty käsittelemään mahdollisimman objektiivisesti ja avoimesti. Useissa kirjoissa on enemmän kuin yksi kirjoittaja, jotka ovat yleensä alansa asiantuntijoita, joka myös parantaa luotettavuutta. Tällä tavoin lähteistä on myös pyritty karsimaan kaupallista tai aatteellista yksipuolisuutta pois. Käytettäviksi lähteiksi on pyritty valikoimaan mahdollisimman ajantasaisia teoksia siinä määrin, mitä esimerkiksi kirjastot pystyvät niitä tarjoamaan. Lähdemerkintöihin on aina merkattu kirjoittaja(t), julkaisuvuosi, teoksen nimi, julkaisupaikka ja julkaisija.

Internetlähteitä ei ole käytetty kuin pienempiin kokonaisuuksiin, kuten käsitteiden selittämiseen ja kaupallisten tuotteiden läpikäymiseen. Esimerkiksi Scrum-viitekehyksen lähde on viitekehyksen kehittäjien kirjoittama julkinen internetartikkeli. Yleensä käsitteitä tukee vielä kirjallisuudesta saatava varmentava informaatio.

3 Tehtävienhallinnan kehittämisen määrittely, suunnittelu ja toteutus

Tehtävienhallinnan kehittämisprojekti käynnistyi kesällä 2015. Tuolloin oli käynnistynyt Vision Systemsin uuden konenäkökamera-alustan kehittämisprojekti VisiCORE, jonka toteuttamista varten tarvittiin kehittämisprosessi ja työkalut kehittämisprosessin seurantaan ja toteuttamiseen.

Lisäksi Vision Systemsillä ja Vision Developmentilla oli useita samanaikaisia asiakasprojekteja, joiden hallitsemiseen prosessia ja työkalua voitaisiin käyttää. Näitä asiakasprojekteja varten oli jo aiemmin perustettu kehitystiimi nimeltä Team AHMA (TAHMA), johon tehtävienhallinnan kehitysprojektin raportoinnissa tullaan useasti viittaamaan. TAHMA:n vastuualueisiin kuuluivat ohjelmisto-, järjestelmä ja sähkösuunnittelu PC-sovelluksiin, CoDeSys-projekteihin teollisuuteen ja liikkuviin koneisiin, sekä testausjärjestelmiin. TAHMA:lle delegoidut tehtävät olivat siis laaja-alaisia ja yhtäaikaisia projekteja saattoi olla meneillään useampia. Kehitystiimin koko vaihteli tehtävienhallinnan kehitysprojektin aikana kolmesta viiteen kehittäjään.

3.1 Esisuunnittelu ja määrittely – Toimeksiantajan vaatimukset

Tehtävienhallinnan kehitysprojektin alussa VisiCORE-projektin puitteissa oli jo tehty tehtävienhallintatyökalujen alustavaa evaluointia, valittu työkalu tehtävienhallintaan ja määritelty alustava suunnitelma kehitysprosessille. VisiCORE tuli viedä läpi käyttäen Scrum-viitekehystä Jira-työkalun avulla.

Jiraan luodaan projekteja ja niille näkymiä (Board). Näkymien avulla kehitystiimit hallitsevat tehtäviään (Issue). Tehtävätyyppejä on useita, niiden kompleksisuutta voidaan estimoida ja niille määritellään työnkulku (Workflow). Jiraan on liitettävissä muita joko Atlassianin tai kolmannen osapuolen lisäosia, jotka integroivat Jiran esimerkiksi vaatimustenhallintaan, dokumenttien hallintaan tai versionhallintajärjestelmään.

VisiCORE:en tehdyn prosessisuunnitelman integrointimahdollisuus tehtävienhallintaan oli yhtenä vaatimuksena määriteltäessä vaatimuksia työkalulle. Eli TAHMA:lle toteutettavan kehitysprosessin ja tehtävienhallintatyökalun tulisi pystyä toimimaan samanaikaisesti perinteisten tuotekehitysprojektien, kuten VisiCORE, kanssa.

TAHMA-tiimin tehtävienhallintaprosessin työkalu oli siis jo määrittelyvaiheessa valittu, mutta työkalun käyttö ja itse prosessi tulisi määritellä tarkoituksen mukaiseksi. Määrittely alkoi sähköpostikeskustelulla viiden henkilön toimesta; VisiCORE:n projektipäällikkö, VD:n tuotekehityspäällikkö, ohjelmistoasiantuntija ja kaksi TAHMA-kehitystiimin jäsentä, joista toinen on siis allekirjoittanut. Ensimmäiset vaatimukset ja määritelmät muotoiltiin hyvin työkalupainotteisiksi ja spesifisiksi, koska käytettävä

työkalu oli jo tiedossa ja henkilöillä oli ennestään kokemuksia sen hetkisen työskentelyn kehityskohteista:

1. *Product ownerit eivät näe Team Ahma boardia:*
    - *Käyttöönottovaiheessa se olisi hyvä*
    - *Jatkossa ei ole tarpeen, pidetään PO:t pois sotkemasta ja scrum master säätää taulua!*
2. *Sprinttien kesto ja toteutus*
    - *Voitaisiin kokeilla siirtoa 1 viikon sprintteihin -> sprintti olisi oikeasti lukittu (pienellä hätätyöjoustolla..)*
    - *Maanantaipalaverin jälkeen Product Ownerit tekisivät backlogien groomaukset*
    - *Sprintin suunnittelupalaveri vaikka tiistai-aamuna(!) josta sprintti alkaisi*
3. *Jiran projektit*
    - *Ei tarvetta olla scrum-tyyppisiä kun ei tarvita sprinttejä -> voisiko olla kanban-tyyppinen?*
    - *Ainakin kanban boardeihin pystyy konffaamaan estimaatit näkyviin (testattu..)*
    - *Projektien backlogissa voisi näkyä vain Storyt:*
        - *Epicit ovat jo piilossa:*
            - *=Feature, muoto: ”User interface”*
            - *Ei tarvitse näkyä backlogissa/sprintissä/boardissa*
        - *Story:*
            - *=User Story, muoto: ”As a PO...”*
            - *Ainoa joka näkyy backlogissa*
            - *Storyä ei voi siirtää sprintissä, vain sub-taskia voi*
        - *Sub-taskit:*
            - *=Task*
            - *Saisi näkyviin backlogissa vasta klikkaamalla storyä (onnistuuko tällainen?)*
            - *Sprintissä vain sub-taskeja siirrellään sarakkeissa!!!*
4. *Team Ahma board*
    - *Voisiko Team Ahma board olla scrum-tyyppinen? -> saataisiin oikeat sprintit*
    - *Saisiko Tahman backlogiin nostettua muiden projektien storyjä*

(Sähköpostikeskustelu elo-syyskuu 2015)

Todettiin, että vaatimuslistaus sisälsi liian spesifisiä ja jo toteutukseen kantaa ottavia vaatimuksia (kuten viittauksia valittuun työkaluun tai Scrum-viitekehykseen), eikä varsinaisesti määritellyt yrityksen ja tiimin toiminnalle tärkeitä vaatimuksia. Tärkeinä asioina pidettiin mm. sitä, ettei tiimin jäsenen tarvinnut nähdä tulevaa työkuormaa, vaan pystyttäisiin rauhoittamaan kehitystyö yhteen asiaan kerrallaan. Haluttiin myös, että ainoastaan eri asiakasprojektien projektipäälliköt (tai Product Ownerit, PO) näki-

sivät vastuullaan olevien projektien tulevat tehtävät ja vastaisivat niiden priorisoinnista. Lisäksi haluttiin, että tehtävien työmääräarviot olisivat mahdollisia ja että tehtävienhallintaprosessi ottaisi huomioon kiireelliset tukityöt.

Näin ollen vaatimusmäärittely kehittyikin keskustelun edetessä edellä esitetystä huomattavasti suppeampaan ja abstraktimpaan suuntaan. Lopullisen vaatimusmäärittelyn toteuttamiseksi järjestettiin palaveri, jota varten koottiin kehitykseen osallistuneiden henkilöiden ideoita ja toiveita TAHMA-tiimin tehtävienhallintaprosessin toteuttamiseksi. Palaverin perusteella toteutettu vaatimusmäärittelydokumentti sisälsi seuraavat vaatimukset:

VD-REQ-0.1. TAHMA-projektin tiimin jäsenten tulee nähdä vain periodille suunniteltu työkuorma

VD-REQ-0.2. Asiakasprojektien, joista TAHMA:lle tehtävät linkataan, tulee olla hallinnoituna erillisissä tehtäväjonoissaan, joiden prioriteetit järjestää projektin PO

VD-REQ-0.3. Projektien tehtäville tulee voida estimoida pisteet

VD-REQ-0.4. Prosessiviolaatiot (eli ”tulipalohommat”) tulee eritellä

Työkalukohtaiset vaatimukset ja ominaisuudet päädyttiin eriyttämään erilliseen dokumenttiin, koska niiden tarkoitus oli ennemminkin auttaa työkalun käyttöönotossa kuin palvella yrityksen tarpeita. Näitä vaatimuksia käsitellään kappaleessa 3.3.

3.2 Suunnittelu – Prosessikuvaus

3.2.1 Team Ahma

Tehtävienhallinnan prosessikuvaus tehtiin vaatimusmäärittelyn ja määrittelyvaiheessa esille tulleiden huomioiden perusteella. Aluksi alettiin määrittelemään tarkempia vaatimuksia erilliseen dokumenttiin, mutta lopulta nähtiin tarkoituksenmukaisemmaksi tehdä prosessikuvaus Power Point –esitysmuotoon, jota voitaisiin jatkossa käyttää myös prosessiin osallistuvien perehdyttämiseen (ks. Liite 1). Tehdyn esityksen mukaisen prosessin tulisi vastata seuraaviin tarkennettuihin vaatimuksiin:

VD-REQ-1. Tehtävähallintaan ja ohjelmistokehitykseen tulee olla työkalu, jolla voidaan hallinnoida tehtäviä projektikohtaisesti

VD-REQ-6. Tiimien tehtävien priorisoinnista vastaavat projektien projektivastaavat

VD-REQ-10. Periodikohtaiset tehtävät pisteytetään suunnittelupalaverissa, joka järjestetään projekti-/tiimivastaavan tai erillisen Scrum Masterin toimesta tasaisin väliajoin

VD-REQ-11. Tiimi kehittää omaa toimintaansa järjestämällä retrospektiivin tasaisin väliajoin

VD-REQ-14. Tehtäviä ei tule lisätä työn alle kesken periodin

Kesken periodin tulevat tehtävät ilmoitetaan projektivastaavalle, joka tarpeen vaatiessa käy tehtävät läpi muiden projektivastaavien kanssa. Jos tehtävä on luonteeltaan erittäin korkealla prioriteetilla ja muut projektivastaavat sen toteuttamisen hyväksyvät, voidaan tehtävä ottaa työn alle kesken periodin. Tällöin projektivastaava kirjaa tehtävän työkaluun ja antaa sen tehtäväksi.

Kesken periodin tulleet työtehtävät tulee kirjata niille tarkoitetun projektin alle (esimerkiksi Customer Support) erittäin korkealla prioriteetilla.

Team Ahma –tyyppisen kehityksen prosessikuvauksen mukainen prosessi ja sen käyttöönotto käydään läpi luvussa 3.5.

3.2.2 VisiCORE

Kuten edellä mainittiin, VisiCORE-projektin prosessikuvaus oli jo ennalta määritelty ja se mukaili Scrum-viitekehystä. Tässä luvussa käydään läpi pääkohtia määritelmästä, joka siis toimi yhtenä tapana toteuttaa tehtävienhallintaa ja näin ollen oli vaatimuksena toteutettavalle työkalun konfiguroinnille.

Tuotetta kehitetään yleensä 2 viikon mittaisissa, tasapituisissa sprinteissä. Sprintin sisältö suunnitellaan yhdessä, sprintin alussa pidettävässä suunnittelupalaverissa johon otetaan kehitysjonosta eli backlogista storyjä. Kun sprintti on alkanut, tekemistä hallitaan säännöllisissä palavereissa, daily scrumeissa, joita järjestetään 3-5krt viikossa. Palavereissa päivitetään tieto tiimin jäsenten etenemisestä ja poistetaan esteet työn tekemiseltä, ei esim. suunnitella tuotetta. Sprintin päätyttyä tuotokset eli toimiva ohjelmisto esitellään katselmointipalaverissa ja pidetään retrospektiivi, jossa tiimi pohtii edellisen sprintin työskentelytapoja ja parannusehdotuksia niihin.

Tuotteen backlogin luomisesta ja ylläpidosta vastaa product owner ja Scrumin mukainen backlogin ylläpito tehdään erillisissä ylläpito, grooming-palavereissa. Erillinen aika ylläpidolle on varattu, ettei siihen liittyviä asioita selvitellä sprintin suunnittelussa tai daily scrumeissa.

Sprinttiin valitut stoorit pisteytetään yhdessä tiimin kanssa suunnittelupalaverissa, mutta kehittäjät voivat itsenäisesti luoda tarvittavat alitehtävät. Toteuttamalla stooriin luodut alitehtävät, stoorin hyväksyntäkriteerit täyttyvät ja ko. ominaisuus tulee tehdyksi.

Määritelmä sisältää myös Scrumin mukaiset roolit, jotka ovat esitelty teoriaosuudessa. Lisäksi määritelmässä käydään läpi Jira-työkalun tehtävätyypit (Issuet), sekä niiden hierarkia ja työnkulku. Näihin palataan tarkemmin työkalun käyttöönoton toteutuksessa.

3.3 Suunnittelu – Työkalujen evaluointi ja valinta

Kuten luvussa 3.1 mainittiin, työkalujen alustavaa evaluointia oli tehty jo VisiCORE-projektin puitteissa ja sen perusteella päädytty käyttämään Jira-tehtävienhallintatyökalua. Ennen varsinaista käyttöönottoa haluttiin varmistaa, että Jira pystyisi vastaamaan myös TAHMA-tiimin prosessin mukaisiin vaatimuksiin. Jirasta oli käytössä evaluointiversio, jonka avulla sen ominaisuuksia voitiin verrata tehtävienhallinnan työkalulle asettamiin vaatimuksiin:

VD-REQ-2. Tehtävähallintaa ja ohjelmistokehitystä tulee voida toteuttaa työkalun puitteissa Scrum- ja Kanban-prosessien mukaisesti

VD-REQ-3. Projektihenkilöstön roolitukset ja vastuut tulee kirjata työkalussa projektin kuvaukseen

VD-REQ-4. Projekteilla tulee olla ns. projektivastaavat (vrt. product owner), jotka kirjaavat ja priorisoivat ko. projektin tehtävät

VD-REQ-5. Työkalulla voidaan hallinnoida tehtävälistaa tiimeille, jotka ovat mukana useammissa projekteissa yhtä aikaa

VD-REQ-7. Työkalulla tulee voida suorittaa useampien projektien Scrum-prosessin mukaisia Sprinttejä yhtä aikaa

VD-REQ-8. Tehtävät eivät voi olla useammassa projektissa tai tiimissä työn alla yhtäaikaisesti

VD-REQ-9. Tiimien jäsenien tulee nähdä ainoastaan periodille tai työn alle suunniteltu työkuorma

VD-REQ-12. Tiimien jäsenet määrittelevät tehtäville alitehtävät ja kirjaavat ne tehtäville

VD-REQ-13. Tiimien tehtävien seurannasta vastaavat projektivastaavat

VD-REQ-15. Työkalussa tulee voida määritellä projektille näkymä käytettävän prosessin mukaisesti

VD-REQ-16. Projektit ja projektitunnisteet tulee nimetä työkaluun yhdenmukaisesti

Projektin nimi muotoa [projektinumero] [projektin nimi]

VD-REQ-17. Työkalussa tulee voida määritellä tehtävälle tyyppi, esim. Epic, Story, Bug

VD-REQ-18. Työkalussa tulee voida määritellä tehtävälle tila, esim. To Do, In Progress, To Review, In Review, Reviewed, Done, Postponed

VD-REQ-19. Tehtävälle tulee kirjoittaa muotoilultaan yhdenmukainen otsikko ja kuvaus

VD-REQ-20. Tehtäville tulee voida asettaa tekijä

VD-REQ-21. Työkalussa tehtävien tilan vaihdokset tulee rajata

VD-REQ-22. Tilanvaihdokset tapahtuvat tehtävän tekijän toimesta

VD-REQ-23. Tilanvaihdokset toteutetaan pienimmälle tehtävälle

VD-REQ-24. Mikäli tehtävän sisältö muuttuu prosessin aikana, sille tehdään uusi ticket ja vanha suljetaan

VD-REQ-25. Työkalussa tulee voida pisteyttää kaiken tyyppisiä tehtäviä kompleksisuuden mukaan

VD-REQ-26. Työkalussa tulee voida määritellä tehtävälle prioriteetti

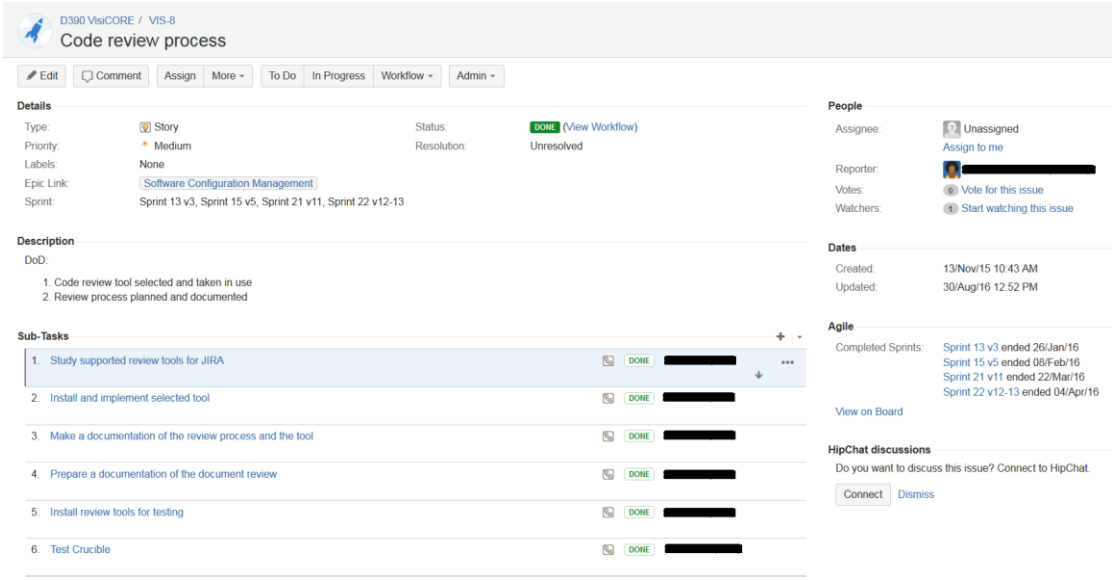
Evaluoinnin aikana todettiin, että suurin osa em. vaatimuksista pystytään Jirassa täyttämään pienellä konfiguroinnilla, mutta joihinkin kohtiin vaaditaan erillinen ratkaisu tai lisäosan asentamista.

Esimerkiksi useamman tiimin ja projektin tehtävienhallinnan synkronointi vaatisi suurempaa konfigurointia tai prosessimäärityksen, mikäli osa projektin tehtävistä olisi toteutumassa projektitiimin ulkopuolella eli toisen tiimin toimesta (VD-REQ-6-8). Ongelman ratkaisemiseksi päädyttiin sopimaan, että tehtäviä ei voida tehdä ristiin kahdella tiimillä, vaan tällaiset tapaukset toteutetaan ensisijaisesti projektitiimissä. Tällä parannetaan myös tehtävän jäljitettävyyttä, koska Jira raportoi minkä projektin (tiimin) kautta tehtävä on tehty ja milloin. Päädyttiin myös eriyttämään pienemmät asiakasprojektit Kanban-menetelmällä toteutettaviksi projekteiksi, koska Kanban ei sisällä erillisiä kehitysperiodeja, joita kuitenkin tiimeissä tulisi tehdä. Tällä varmistetaan, ettei tehtävä pysty olemaan yhtä aikaa työn alla kahden eri tiimin sprintin näkymässä.

Evaluoinnin aikana havaittiin myös, ettei esimerkiksi Kanban-näkymässä pysty estimoimaan (pisteyttämään) tehtäviä (VD-REQ-25). Scrum-näkymässä tämä on mahdollista. Ongelman ratkaisemiseksi asennettiin ja testattiin Agile Estimates –lisäosaa, jonka avulla tehtävien estimointi onnistui myös Kanban-taululla.

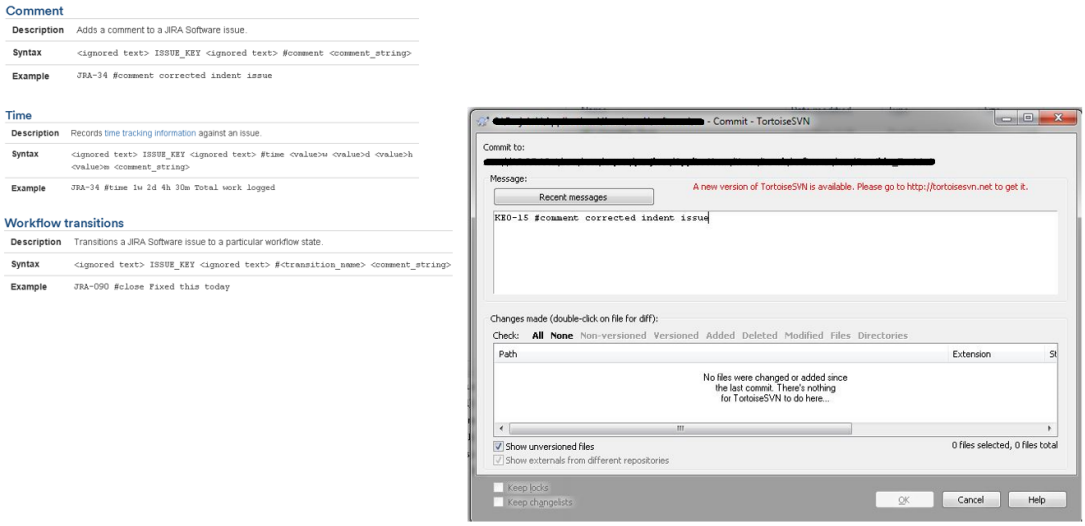
3.3.1 Jira Crucible ja FishEye

VisiCORE-projektissa tuli ottaa käyttöön katselmointityökalu ohjelmakoodinkatselmointia varten. Vaatimuksena oli selvittää, valita ja käyttöönottaa Jira-yhteensopiva katselmointityökalu. Työkalun evaluointitoimeksiannon alkaessa Jira oli jo käyttöönotettu ja toimeksiannon tehtävää hallinnoitiinkin Jirassa. Kuviossa 8 on Jira storyn ”ticket”, joka kuvaa hyväksytysti suoritetun toimeksiannon vaatimukset (DoD – Definition of Done), sen alle määritellyt alitehtävät, tehtävien suorittajat ja muut toimeksiantoon liittyvät ominaisuudet.



Kuvio 8. Katselmointiprosessin ja –työkalun story Jirassa

Evaluontia varten yrityksen palvelimelle asennettiin Jira Crucible ja Fisheye maaliskuussa 2016. Crucible on Jiraan integroitava katselmointien hallinnointityökalu ja Fisheye puolestaan integroi katselmointityökalun versionhallintaan, jossa mm. toteutettu ohjelmakoodi sijaitsee. Evaluoinnissa selvitettiin ja testattiin, että integrointi on mahdollinen toimeksiantajan käyttämään Subversion-versionhallintajärjestelmään. Testausta varten Crucible ja Fisheye konfiguroitiin niin, että se oli integroituna toimeksiantajan Jira-instanssiin ja versionhallintaan, joka mahdollisti Jira-tehtävien linkityksen katselmointitoimeksiantoihin ja katselmointitoimeksiantojen linkityksen versionhallinnassa sijaitsevaan ohjelmakoodiin. Integraation testauksessa Jiraan luotiin esimerkkitehtävä ja versionhallintaan esimerkkitekstitiedosto, joka simuloi ohjelmakoodia. Versionhallintaan lisättäessä (”commitoitaessa”) katselmoitava ohjelmakoodi (tässä tapauksessa esimerkkitekstitiedosto) voidaan käyttää Smart Commit –komentoja, jotka automaattisesti linkittävät commitoidun tiedoston halutulle Jira-tehtävälle (ks. Kuvio 9). Komennoilla on myös mahdollista toteuttaa monimutkaisempiakin toimintoja, kuten tehtävän tilanvaihdot Jirassa (esim. In Progress → To Review) ja katselmointitoimeksiannon luominen Crucbileen.



Kuvio 9. Jira-integroitu versionhallinnan käyttö

Katselmointitoimeksianto luodaan joko Smart Commit –komennon avulla tai Jira-tehtävän kautta. Luodessa määritellään mm. katselmoijat, jotka saavat katselmointitoimeksiannosta tiedoksi sähköpostin. Katselmointi suoritetaan ja siinä havaitut asiat kommunikoidaan Cruciblessa. Crucible tarjoaa poikkeavuuksien vertailuun työkalun, jonka avulla katselmoija voi verrata tuotettua ohjelmakoodia edelliseen versioon (ks. Kuvio 10). Tämän mahdollistaa Fisheye-versionhallinta integraatio. Kun katselmointi on suoritettu, kehittäjä saa tapahtumasta sähköpostin ja voi mennä katsomaan katselmoinnin tulokset.

Kuvio 10. Poikkeavuuksien vertailun työkalu Cruciblessa

Evaluoinnin tuloksena toteutettiin Power Point –esitys, jossa kuvataan valittu työkalu, työkalun käyttö ja esitellään muita havaintoja evaluoinnista. Dokumenttia käytettiin esittelypalaverissa (huhtikuu 2016), jossa katselmointityökalun ominaisuuksia esiteltiin toimeksiantajalle ja työkalun hankintaehdotus valmisteltiin. Lisäksi dokumenttia käytettiin pohjana VisiCORE-projektin katselmointikäytänteissä.

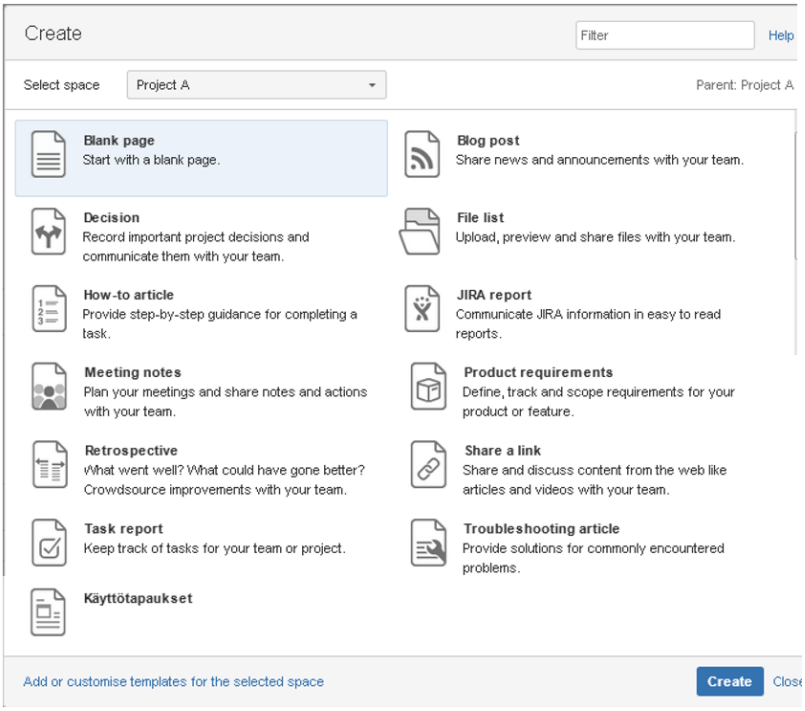
3.3.2 Jira Confluence

Toimeksiantajalla on ollut työn alla kehittää dokumenttien- ja vaatimustenhallintaa. Tehtävienhallintatyökalun käyttöönoton ohella haluttiin tehdä lyhyt selvitys siitä, olisiko Jiraan integroitavissa dokumentointijärjestelmää ja kuinka sitä voisi hyödyntää toimeksiantajan prosessissa. Nopeasti selvisi, että Jiraan on integroitavissa Atlassianin oma dokumenttien hallintajärjestelmä Confluence, jonka perusteella määriteltiin listaus selvitettävistä asioista:

1. Selvitys Confluencen käytettävyydestä VD:n/VS:n dokumenttien hallinnasta
    a. Myynnin dokumentaatio (vrt. nykyään Erp/verkkolevy)
    b. Projektikansiot, asiakasdokumentit, tilaukset yms. (vrt. nykyään verkkolevy)
    c. Projektidokumentaatio, vaatimusmäärittelyt, spesifikaatiot (vrt. nykyään verkkolevy/svn)
    d. Jäljitettävyys myyntivaiheesta svn-tuotoksiin
    e. Tuloksena esimekkiprojekti ja ppt-esitys
2. Selvitys Confluencen muista mahdollisuuksista/toiminnallisuuksista
    a. Intra
    b. (Liitettävyys Erpiin)
    c. Tuloksena ppt-esitys
    d. Kustannusarvio Confluencesta (otetaan huomioon myös suuremmat käyttäjämäärät)

Confluencesta asennettiin instanssi toimeksiantajan palvelimelle elokuussa 2016 ja siihen aktivoitiin evaluointilisenssi. Evaluoinnissa testattiin yleisellä tasolla Confluencen toiminnallisuuksia, verrattiin niitä toimeksiantajan vaatimuksiin ja luotiin esimerkkitiloja ja –dokumentteja.

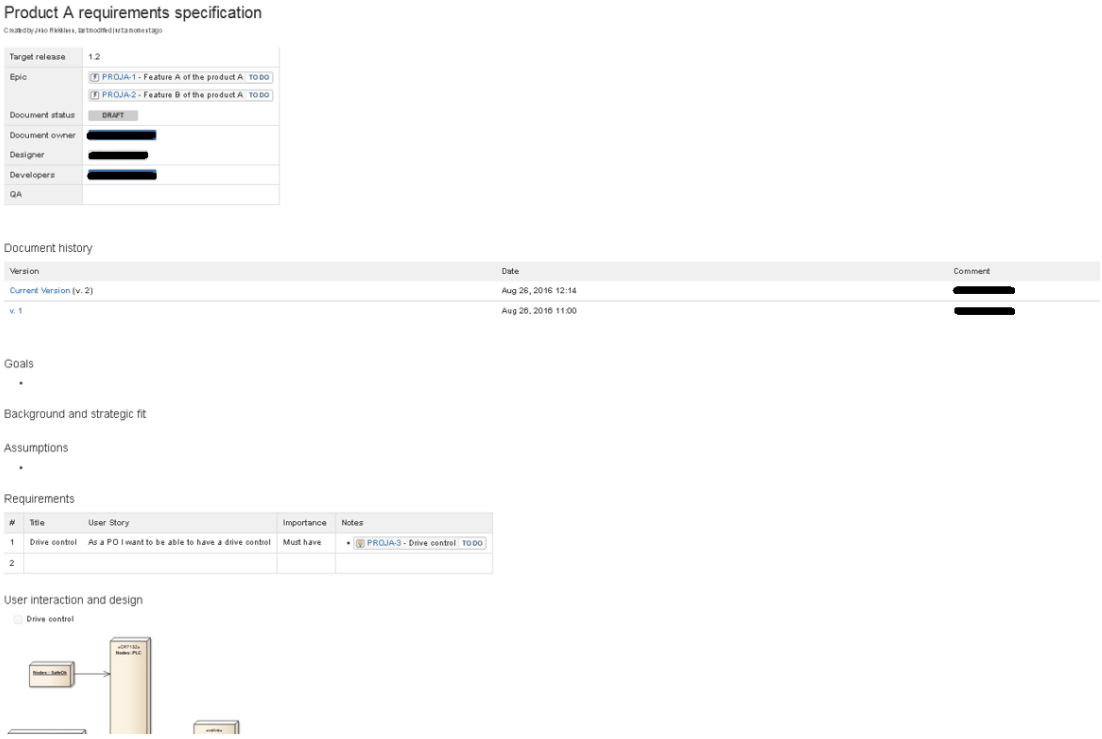
Confluenceen luodaan tiloja (space), joiden alle voidaan luoda sivuja (page). Tilat voidaan määritellä kuuluvaksi ryhmään (esim. projektit, HR jne.), joka helpottaa sivujen selailua. Sivun (tai dokumentin) voi luoda tyhjälle pohjalle tai käyttää valmiita template-pohjia. Kuviossa 11 näkyvät valmiit pohjavaihtoehdot ja lisäksi alimpana itse luotu dokumenttipohja.



Kuvio 11. Confluencen sivupohjat

Sivuista pystyi siis muokkaamaan käytännössä millaisia tahansa dokumentteja tai esimerkiksi intra-sivuja käyttäjä halusi. Evaluoinnissa testattiin suurin osa valmiista pohjista, joista toimeksiantajan kannalta kiinnostavin oli vaatimusmäärittely (Product requirements). Vaatimusmäärittelypohjaan määriteltiin linkityksiä Jira-tehtäviin ja muihin dokumentteihin. Tällä haluttiin selvittää, kuinka jäljitettävyyttä voitaisiin parantaa dokumenttien ja tehtävien välillä. Lisäksi vaatimusmäärittelyyn luotiin muita objekteja, kuten kuvia, versiohistoria ja taulukoita (ks. Kuvio 12).

Confluence tarjoaa lisäksi dokumenttien versiohallinnan ja työkalun poikkeavuuksien vertailuun versioiden välillä. Dokumenteista voidaan tulostaa pdf- ja doc-tiedostoja ja dokumenttien hakemiseen on valmiita toimintoja.



Kuvio 12. Dokumenttiesimerkki Confluencessa

Evaluoinnista toteutettiin Power Point –esitys, joka esiteltiin katselmointipalaverissa. Yhteenvetona evaluoinnista voidaan sanoa, että dokumenttien luonti ja hallinta on ainakin pienellä määrällä helppoa ja yksinkertaista. Valmiit pohjat ja linkitykset muihin dokumentteihin helpottavat usean dokumentin muokkaamista, vaikkakin sivujen muokkaaminen yleisesti ei ole monipuolista (esimerkiksi tekstiobjektien muokkaaminen rajoittunutta ja työlästä). Linkitykset mahdollistavat myös jäljitettävyyden parantamisen ja dokumentaatio voidaan yhdistää suoraan Jira-tehtäviin. Kun yhdistää Jiran, Confluencen, Cruciblen ja Fisheyen, niin koko ketju Confluencen asiakasvaatimusdokumentaatiosta, Jira-tehtävien, Crucible-katselmointien ja Fisheyen versionhallintaintegraation kautta toteutukseen olisi jäljitettävissä.

3.4 Toteutus – Työkalujen käyttöönotto

3.4.1 Tehtävienhallintatyökalu

Jiran käyttöönotto aloitettiin Vision Developmentissa elokuussa 2015 asentamalla Jira-instanssi toimeksiantajan palvelimelle IT-tuen toimesta. Aluksi työkaluun tuli

konfiguroida yleisiä asetuksia, kuten aktivoida lisenssi, määritellä sähköpostipalvelin, luoda käyttäjätilit, määritellä käyttöoikeudet ja muokata aloitusnäkymä. Jiran natiiviominaisuudet olivatkin melko nopeasti käytössä ja sitä voitiin alkaa koekäyttämään TAHMA:n toimesta vanhan Exceliin pohjautuvan tehtävänhallintataulukon ohessa.

Seuraava tehtävä oli muokata ja käyttöönottaa Jira kappaleessa 3.3 mainittujen vaatimusten vastaiseksi. Työkaluun tuli luoda mahdollisuuksien mukaan valmiit pohjat ja tavat toteuttaa sinne luotavia objekteja ja niihin liittyviä toimintoja. Seuraavassa tullaan esittämään nämä tavat ja toiminnot, joita käyttäjät tulisivat käyttämään. Toiminnot esitetään viittaamalla liitteenä 2 olevaan ohjeistukseen ja niitä verrataan vaatimusmäärittelyyn.

Projektin luonti ja projektinäkymä

Vaatimusmäärittelyn mukaan tehtävienhallintaa ja ohjelmistokehitystä tuli voida toteuttaa Scrum- ja Kanban-prosessien mukaisesti (VD-REQ-2), projektinäkymä tuli voida määritellä valitun prosessin mukaisesti (VD-REQ-15) ja projektit, sekä projektitunnisteet tuli voida nimetä muodossa [projektinumero] [projektin nimi] (VD-REQ-16).

Jokainen projekti eritellään Jirassa projektitunnisteella ja projektin nimellä (ks. Liite 2, diat 13 ja 14). Jirassa on valmiita pohjia Scrum- ja Kanban-taulujen luomiseksi. Projektinäkymät mukautuvat oletuksena näitä menetelmiä tukeviksi, mutta ovat myös myöhemmin muokattavissa omien tarpeiden mukaisiksi. Projektinäkymiä voidaan muokata suodattimien avulla, jotka määrittelevät mm. projektit joiden tehtäviä näkymässä esitetään, sekä halutut tehtävätyypit, tehtävien tilat ja tehtävien tekijät (ks. Liite 2, diat 20 – 22).

Tehtävät

Vaatimusmäärittelyn mukaan tehtäville tulee voida määrittää tyyppi, otsikko ja kuvaus (VD-REQ-17 ja VD-REQ-19), tehtävillä tulee olla valmiusastetta kuvaava tila (VD-REQ-18) ja tehtäville tulee voida määrittää tekijä (VD-REQ-20).

Kun tehtävä luodaan, Jira määrittää sille automaattisesti tunnisteen, joka on muotoa [projektitunniste]-[juokseva numero]. Projektitunniste luodaan sen mukaan, minkä projektin näkymässä tehtävää ollaan luomassa ja juoksevasta numerosta työkalu

huolehtii automaattisesti. Näin tehtävillä ei voi olla identtisiä tunnisteita. Käyttäjän vastuulla on kirjata tehtävälle kuvaava nimi ja asianmukainen selite, sekä valita tehtävän tyyppi (ks. Liite 2, diat 6 – 8 ja 24 – 27).

Jirassa on natiivina tehtävätyypit Epic, New Feature, Improvement, Story, Task, Sub-task ja Bug, joiden käyttöä voidaan rajata teemojen (Scheme) avulla. Määriteltiin, että käyttöön otetaan Epic suurille kokonaisuuksille (esim. UI Development), Story (vrt. User Story) normaalikokoisille vaatimuksesta johdetuille tehtäville, Sub-task storysta johdetuille alitehtäville ja Bug virheille ja korjauksille. Työkaluun määriteltiin teema, joka kattaa nämä tehtävätyypit ja ohjeistettiin käyttämään ainoastaan tätä teemaa (ks. Liite 2, diat 18 – 19).

Tehtävän suorittaja voidaan valita Jirassa tehtäväkohtaisesti, mutta määriteltiin että suorittaja määritellään aina pienimmälle tehtävälle eli sub-taskille (jonka luonti on myös kehittäjän vastuulla, VD-REQ-12).

Kehitysjonot

Vaatimuksena oli myös, että tehtäviä tulee voida estimoida kompleksisuuden mukaan (VD-REQ-25) ja tehtäville tulee voida antaa prioriteetti (VD-REQ-26). Jotta kaikkia tehtäviä pystyttiin estimoimaan kaiken tyyppisissä projekteissa, tuli projekteissa ottaa käyttöön Agile Estimates –lisäosa (ks. Liite 2, diat 9 ja 23). Prioriteetin tehtäville pystyy asettamaan tehtävän asetuksista, mutta tätä toimintoa ei todettu tarpeelliseksi, vaan priorisointi tapahtui kehitysjonoa muokkaamalla niin, että kehitysjonon ylin alkio oli korkeimmalla prioriteetilla.

Kehitysjonolle erityisvaatimuksena oli, että työkalulla tuli voida hallinnoida useammassa projektissa olevan tiimin kehitysjonoa (VD-REQ-5). Koska tiimeille luotiin oma projekti (esim. TAHMA-projekti) ja näkymä, tuli muiden projektin tehtävät linkittää jollain tapaa tiimin omalle tehtävälistalle. Kehitettiin menetelmä, jossa projekteista toiselle tiimille tulevat tehtävät linkitettiin tiimin kehitysjonolle tiimivastaavan (tai muun tiimin jäsenen) kautta. Esimerkiksi tiimivastaavalle pystyttiin linkittämään useasta projektista useita storyja, jotka näkyivät tiimin omalla kehitysjonolla. Storyista voitiin luoda tiimin toimesta sub-taskit, jotka pystyttiin osoittamaan kehittäjille. Tähän vaadittiin myös, että tiimin projektinäkymä oli suodatettu niin, ettei siinä näkynyt kuin tiimin jäsenille osoitetut työt, eivät usean projektin työt. Mikäli näkymä olisi

suodatettu näyttämään useiden projektien töitä, siinä olisi näkynyt kaikki projektien tehtävät, joka oli vastoin vaatimusmäärittelyä (VD-REQ-0.1, VD-REQ-0.2 ja VD-REQ-9).

Työnkulku

Työnkululla tarkoitetaan tehtävien tilaa, tilanvaihdoksia ja tilanvaihdosten sääntöjä (ks. Liite 2, diat 11 – 12).. Työnkulku on toteutettu Jirassa Workflow-teemojen avulla. Työkalussa on oletuksena muutamia valmiita teemoja, mutta sinne on myös mahdollista luoda omia. Oletusteemoissa ei ollut kaikkia toimeksiantajan määrittelemiä tiloja;

To Do – Tehtävää ei ole vielä aloitettu

In Progress – Tehtävä on työn alla

To Review – Tehtävä on valmis katselmoitavaksi (tämän tilan avulla katselmoijaa informoidaan milloin tehtävän voi katselmoida)

In Review – Tehtävä on katselmoitavana

Reviewed – Tehtävä on katselmoitu (tämä tila informoi kehittäjää tarkastamaan katselmoinnin tulos)

Testing – Tehtävä on testauksessa

Done – Tehtävä on valmis

Closed/Postponed – Tehtävää ei voitu suorittaa tai se suoritetaan myöhemmin (VD-REQ-24)

Jiraan toteutettiin teema, joka sisältää em. tilat ja säännöt jotka sallivat siirtymisen kaikista tiloista kaikkiin tiloihin. Käyttöönoton aikana havaittiin, että liian tiukat säännöt siirtymisten välillä tuottivat ongelmia esimerkiksi vahingossa siirrettyjen tehtävien palauttamiseen haluttuun tilaan. Määriteltiin ja ohjeistettiin, että toteutettua teemaa käytetään kaikissa projekteissa ja tarkoituksenmukaiset tilanvaihdot ovat tiimien kehittäjien vastuulla (VD-REQ-21 – 23, ks. Liite 2, diat 15 – 17).

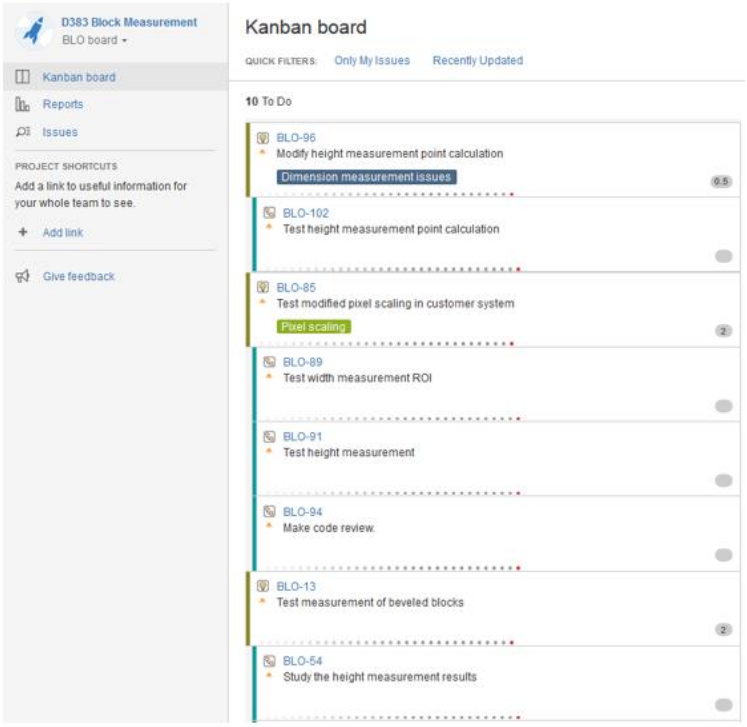
3.5 Toteutus – Prosessin käyttöönotto

3.5.1 Case Team Ahma

Tehtävienhallintaprosessin käyttöönotto aloitettiin elokuussa 2015. Aluksi tehtävienhallintaan käytössä oli vanha Excel-taulukkopohja, jonka rinnalla tehtäviä alettiin kirjaamaan Jiraan. Ensimmäinen versio prosessisuunnitelmasta ja työkalun käyttömenetelmistä oli valmis lokakuussa ja ensimmäinen Jirassa toteutettava sprintti käynnistettiin lokakuun puolessa välissä.

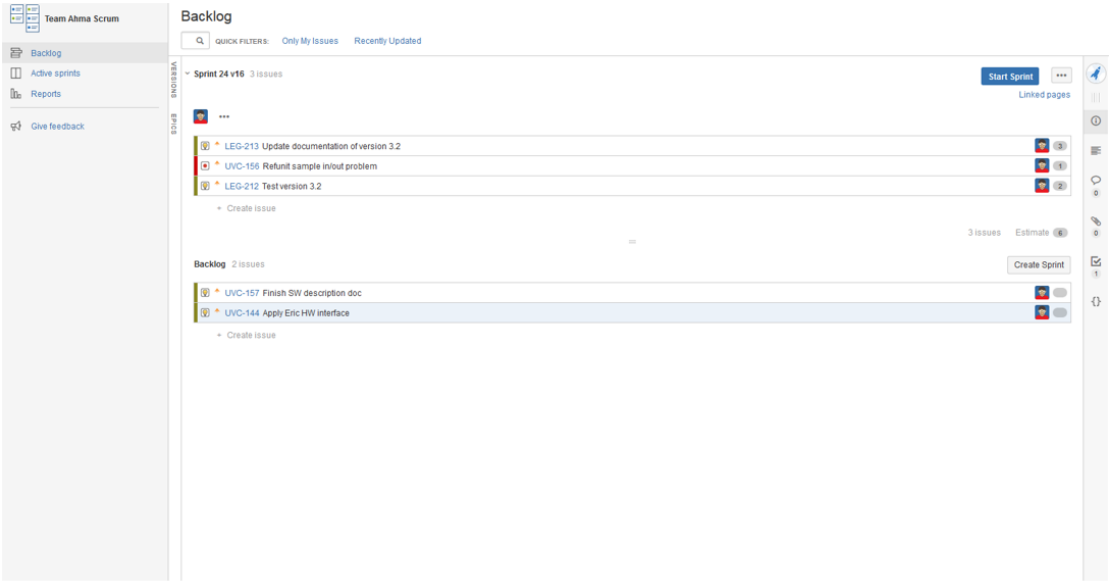
Tiimin työskentely toteutettiin prosessiohjeistusta noudattaen (ks. Liite 1). Projektivastaavia tiimissä oli kolme, kehitystiimin edustajia yksi ja kehitystiimin jäseniä kolme. Projektivastaavien tehtävä on oman projektinsa kohteen ominaisuuksista ja toiminnallisuuksista vastaaminen, kehitysjonon luonti, ylläpito ja priorisointi, sekä periodin tavoitteiden läpikäynti asiakkaan kanssa ja tavoitteiden esilletuonti sprintin suunnittelupalaverissa. Tiimin edustajan vastuulla on esteiden poistaminen kehitystiimin työskentelystä, prosessikäytänteiden noudattaminen, palavereiden järjestäminen ja projektivastaavien tukeminen kehitysjonojen ylläpidossa ja periodin suunnittelussa. Kehitystiimi vastaa kehitystyön tekemisestä, työmäärien arvioinnista (estimointi), storyjen pilkkomisesta alitehtäviin, alitehtävien tilojen ylläpidosta ja katselmoinneista.

Ennen varsinaista periodin aloitusta projektivastaavat järjestelivät oman projektinsa ”kehitysjonon” (prosessissa käytettiin projektin kehitysjonona Jiran Kanban-taulun To Do –saraketta) vastaamaan projektin vaatimuksia, niin että ylimmäinen tehtävä on tärkein ja alimmainen tehtävä on vähiten tärkein (ks. Kuvio 13).



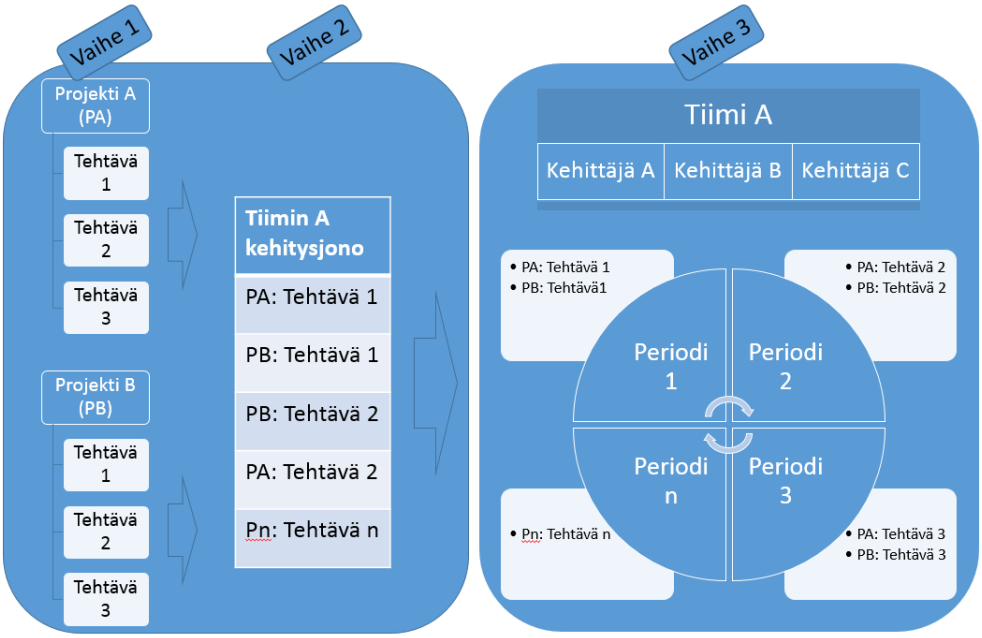
Kuvio 13. Projektin kehitysjono

Kehitysperiodi alkoi projektivastaavien ja tiimin edustajan palaverilla (ks. Kuvio 15, Vaihe 2). Tuossa palaverissa projektivastaavat esittelivät ko. kehitysperiodille etukäteen priorisoimansa (ks. Kuvio 15, Vaihe 1) tehtävät (storyt). Tehtävät tuli myös olla valmiiksi estimoitu kehitystiimin avustuksella. Näiden tehtävien väliltä kehitystiimin kehitysjonoon valittiin se määrä tekemistä, mitä yhteen viikon mittaiseen sprinttiin mahtui (ks. Kuvio 14).



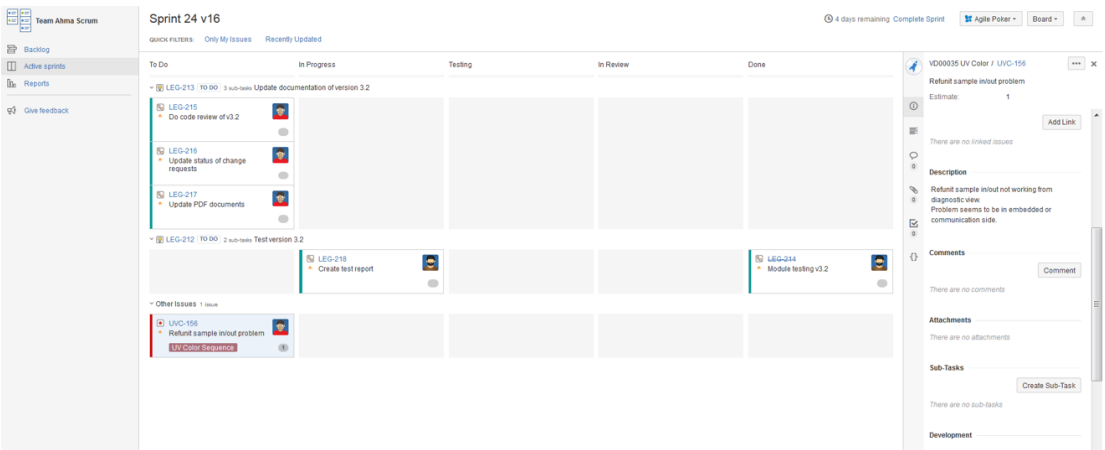
Kuvio 14. Kehitystiimin kehitysjono ja kehitysperiodiin valittavat tehtävät

Seuraavana päivänä pidettiin kehitystiimin ja tiimin edustajan kesken sprintin suunnittelupalaveri, jossa tiimin edustaja esitteli sprinttiin valitut tehtävät (ks. Kuvio 14). Suunnittelupalaverissa sovittiin jokaiselle tehtävälle vastuuhenkilöt ja käynnistettiin sprintti. Suunnittelupalaverin jälkeen kehitystiimin tehtävä oli pilkkoa nuo story-tasoiset tehtävät pienemmiksi yksittäisiksi alitehtäviksi (sub-task) ja aloittaa alitehtävien mukainen kehitystyö (ks. Kuvio 15, Vaihe 3). Suunnittelupalaverissa myös suljettiin edellinen sprintti ja tarkasteltiin sprintin aikana tehtyjen tehtävien määrä (ks. Kuvio 17).



Kuvio 15. Tiimin prosessikuvaus

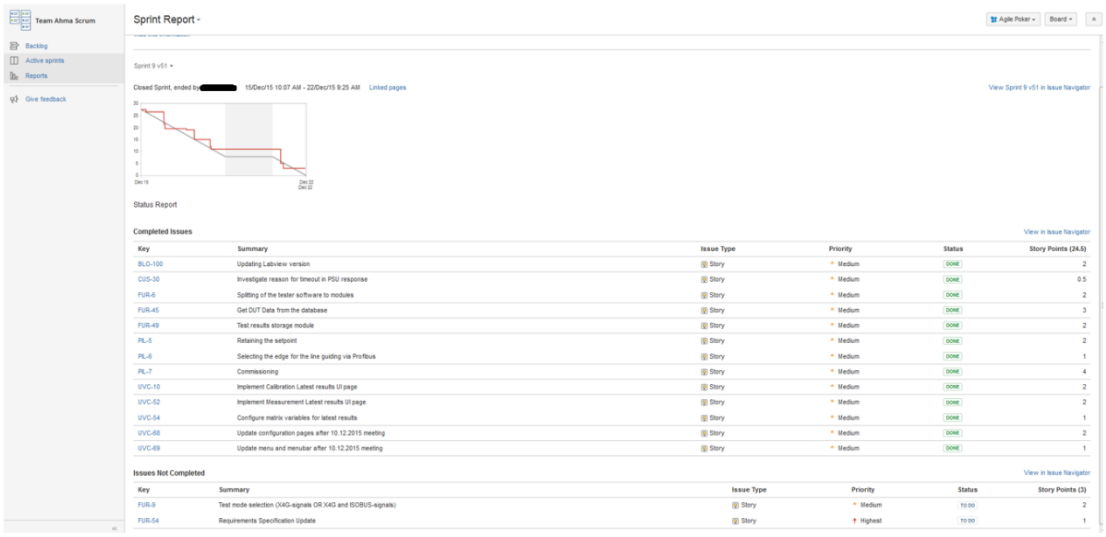
Sprintin aikana kehitystiimi ylläpitää alitehtävien tilaa siirtämällä niitä sarakkeesta toiseen ja mahdollisesti kirjaamalla niihin kommentteja ja tarkempia määrityksiä (ks. Kuvio 16). Alitehtävien tila on sprintin alussa To Do ja kun kehittäjä ottaa sen työn alle, siirretään se In Progress –tilaan. Kyseisen tehtävän tila vaihtuu myös projektin Kanban-taululla, josta projektivastaava voi oman projektinsa tehtävien edistymistä seurata. Sprintin aikana pidettiin myös vaihtelevasti ns. daily scrum –palavereita, esimerkiksi kaksi kertaa viikossa, joissa kehittäjien oli tarkoitus tuoda esille mitä on tehty, mitä tullaan tekemään ja onko tekemiselle mahdollisesti esteitä.



Kuvio 16. Kehitystiimin sprintin näkymä

Mikäli havaittiin, että jonkun tehtävän suorittamiselle oli esteitä tai sen työmäärärvio kasvoi sprintin aikana, ilmoitettiin siitä tiimin edustajan kautta projektivastaaville, jotka päättivät miten kyseisen ja muiden tehtävien osalta toimitaan eli jatketaanko tehtävän toteutusta vai siirretäänkö se myöhempään ajankohtaan. Toinen prosessissa määritelty prosessiviolaation käsittely oli kesken sprintin ilmoitetut tehtävät. Mikäli projektivastaava, tiimin edustaja tai tiimin jäsen sai periodille määrittelemättömän tehtävän, tuli se ilmoittaa muille projektivastaaville tiimin edustajan toimesta, jotka päättivät onko ko. tehtävä riittävän suurella prioriteetilla, että se tehtäisiin kyseisessä sprintissä jonkun toisen tehtävän sijaan vai siirrettäisiin myöhempään ajankohtaan. Yksi prosessimäärittelyn tavoitteita oli juuri tällaisten tilanteiden hallittu käsittely sen sijaan, että tehtävät tulisivat automaattisesti välittömästi työn alle.

Kehitysperiodin jälkeen, sprinttiä suljettaessa, siinä toteutuneet ja toteutumattomat tehtävät raportoidaan Jirassa automaattisesti (ks. Kuvio 17). Raportteja voi tulostaa myös myöhemmin.



Kuvio 17. Kehitysperiodin raportti

Tiimin edustaja järjesti kahden viikon välein (tai tarvittaessa) retrospektiivipalaverin, jossa käytiin läpi tiimin ja prosessin onnistumisia, epäonnistumisia ja kehitettäviä asioita edellisten kehitysperiodien osalta. Palaverien pohjalta kirjattiin muistio, joihin kehitystä pystyi vertaamaan seuraavissa retrospektiiveissä. Lisäksi kehitettävät asiat koottiin erilliseen dokumenttiin, jota käydään tarkemmin läpi tulosten käsittelyssä.

4 Prosessin ja työkalujen käyttöönoton tulokset

Tehtävienhallintaprosessin kehittämisen tuloksina ovat toteutetut määrittelyt ja ohjeistukset toimintatapoihin, käyttöönotettu tehtävienhallintatyökalu, käyttöönotettu prosessimalli ja siinä toteutuneet kehitysperiodit, sekä kaikista em. kohdista saatu käyttäjäpalaute.

4.1 Tehtävienhallintaprosessin määritelmä ja ohjeistukset

Tehtävienhallintaprosessista toteutettiin määritelmä (ks. Liite 1), jossa kuvataan Team Ahma –tyyppinen prosessi. Lisäksi toteutettiin ohjeistus tehtävienhallintatyökalun käyttöön (ks. Liite 2) ja evaluointiraportit katselmointi- ja dokumenttien hallintatyökaluista (ks. luvut 3.3.1 ja 3.3.2). Kaikki em. dokumentit toteutettiin Power Point –esitysmuotoon selkeyden vuoksi ja käyttötarkoitusta, eli perehdytystä ja esittelyä, ajatellen.

Tehtävienhallintaprosessin määritelmään pyrittiin kokoamaan tarvittavat määrittelyt prosessissa mukana olevien rooleista, termeistä, käytännöistä ja poikkeustilanteista. Määritelmän mukaan tulisi pystyä viemään läpi Team Ahma –tyyppisen tiimin kehitystyötä. Määritelmän lisäksi toteutettiin lyhempiä, noin kahden kalvon mittaisia pikaohjeita tietyn prosessin roolin omaaville, esimerkiksi kehitystiimin jäsenen pikaohje.

Dokumentteja muokattiin käyttöönoton yhteydessä korjausehdotusten ja ohjeiden testauksen myötä. Dokumentit on sijoitettu toimeksiantajan verkkolevylle, jossa ne ovat kaikkien työntekijöiden nähtävissä. Esityksiä on esitelty palavereissa, joissa on käyty läpi tehtävienhallintaprosessin kehitystä ja työkalujen evaluoinnin tuloksia. Lisäksi dokumenteista voidaan esittää tarvittavia kohtia perehdytystilaisuuksissa.

4.2 Työkalut

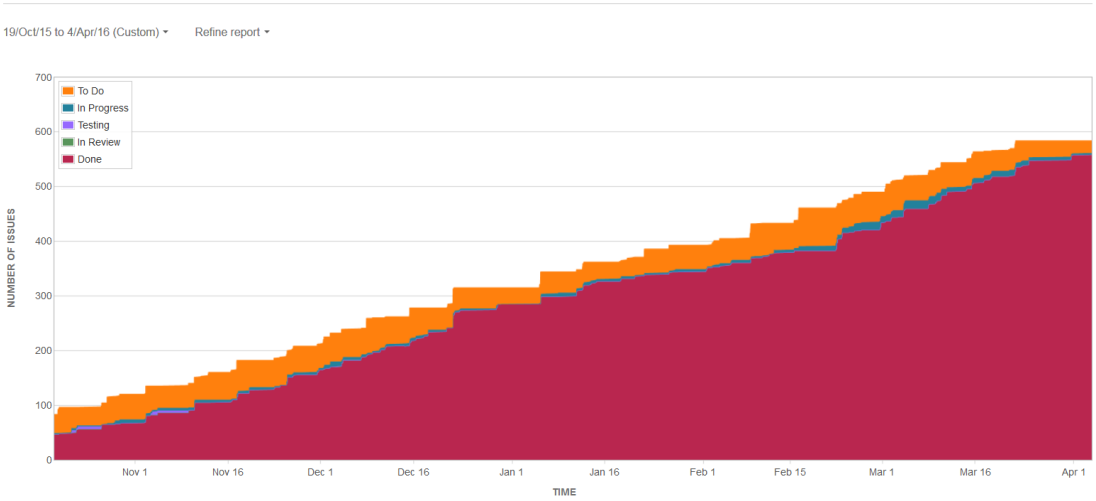
Toimeksiantajalla on käytössä Jira tehtävienhallintatyökalu, joka on asennettu yrityksen palvelimelle. Käytössä on 25 lisenssiä, joista aktiivisessa käytössä on vaihteleva määrä, riippuen projektien määrästä joita Jirassa hallitaan. Lisäksi käytössä on Crucible katselmointityökalu, sekä FishEye integroimaan Jira toimeksiantajan versionhallintajärjestelmään. Jira ja lisäosat on konfiguroitu toimeksiantajan vaatimusten mukaisesti ja tehtävienhallinnan toteuttaminen on annettujen ohjeistusten mukaisesti mahdollista.

Jiraan on luotu tilat noin kolmellekymmenelle projektille, joista osa on jo valmistunut ja osa aktiivisia. Projekteihin on kirjoittamishetkellä luotu yli 1600 tehtävää mukaan lukien myös alitehtävät. Alitehtävät pois lukien tehtäviä on yli 700.

Jira on käytössä kaikissa projekteissa, joiden tehtäväjonoja hallitaan sisäisesti eli sisäisissä projekteissa ja projekteissa joissa asiakkaalla ei ole käytössä omaa tehtävienhallintatyökalua ko. projektille.

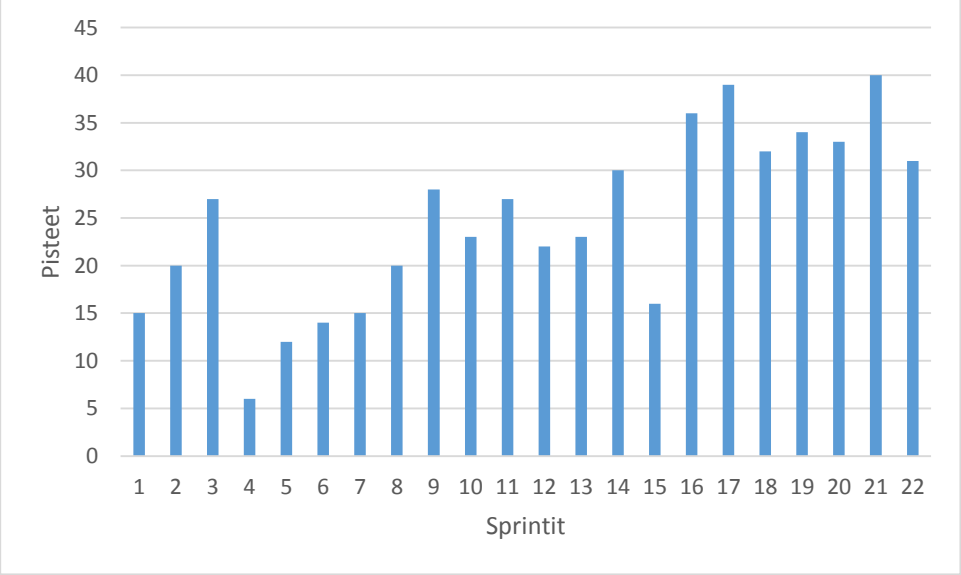
4.3 Prosessimallin käyttöönotto ja kehitysperiodit

Toimeksiantajan vaatimusmäärittelyn mukaisesti kehitettyä prosessimallia käyttöönotettiin ja koekäytettiin noin puoli vuotta. Käyttöönoton aikana toteutettiin 22 yhden tai kahden viikon mittaista kehitysperiodia eli Scrum-termein sprinttiä. Sprinttien aikana toteutettiin kehitystyötä usealle projektille ja hallinnassa oli lähes 600 tehtävää (ks. Kuvio 18). Tiimiin osallistui 5 – 10 henkilöä (ml. projektivastaavat, tiiminedustaja ja kehitystiimin jäsenet) kerrallaan riippuen aktiivisina olevien projektien lukumäärästä. Tiimi kehitti toimintaansa jatkuvasti prosessimallin mukaisten palaverien ja toimintatapojen avulla.



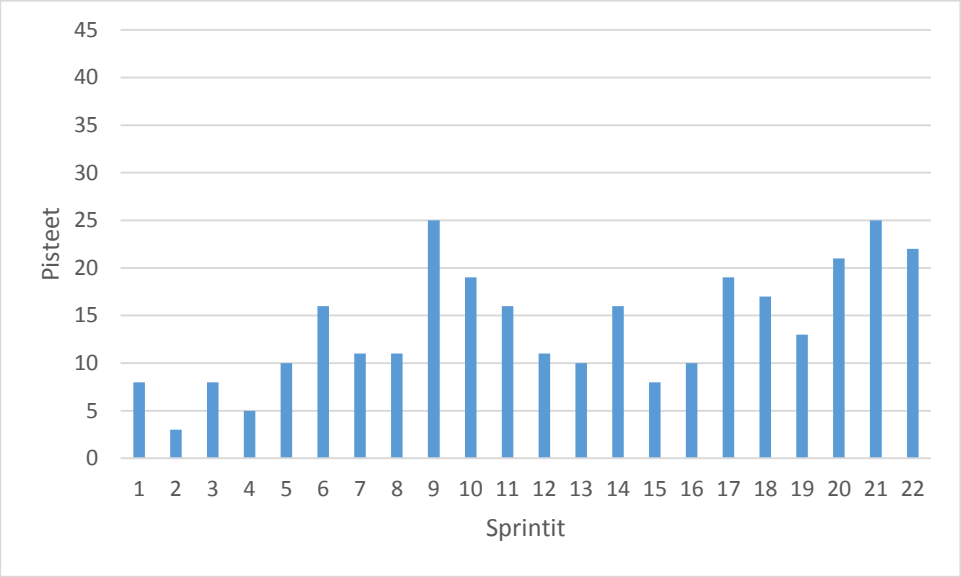
Kuvio 18. Jira-raportti tehtävien määrästä ja tilasta prosessimallin käyttöönoton aikana

Jokaisen kehitysperiodiin eli sprinttiin suunniteltiin tietty määrä toteutettavaa työtä riippuen kehitystiimin kapasiteetista. Projektien tehtävät estimoitiin eli pisteytettiin Jiraan tehtäväkohtaisesti sen kompleksisuuden ja työaika-arvion mukaan. Ao. taulukoihin on koottu Jiran sprinttiraporteista tilastot sprinteissä hallittujen tehtävien pistemääristä. Kuviossa 19 kuvataan kaikkien käyttöönoton aikana toteutettujen sprinttien suunnitellut pistemäärät eli työmäärä joka kuhunkin sprinttiin otettiin toteutettavaksi. Suunniteltujen työmäärien keskiarvo käyttöönoton ajalta on 24,7 pistettä per sprintti.



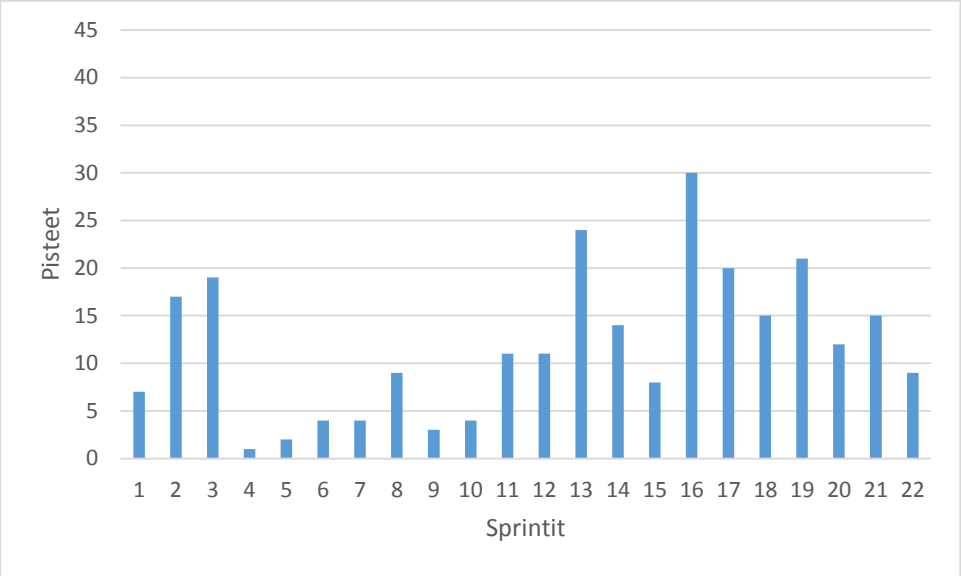
Kuvio 19. Sprintteihin suunniteltujen tehtävien yhteispistemäärät

Kuvio 20 kuvaa käyttöönoton aikana sprinteissä toteutettujen tehtävien pistemäärät. Toteutuneiden työmäärien keskiarvo käyttöönoton ajalta on 13,8 pistettä per sprintti.



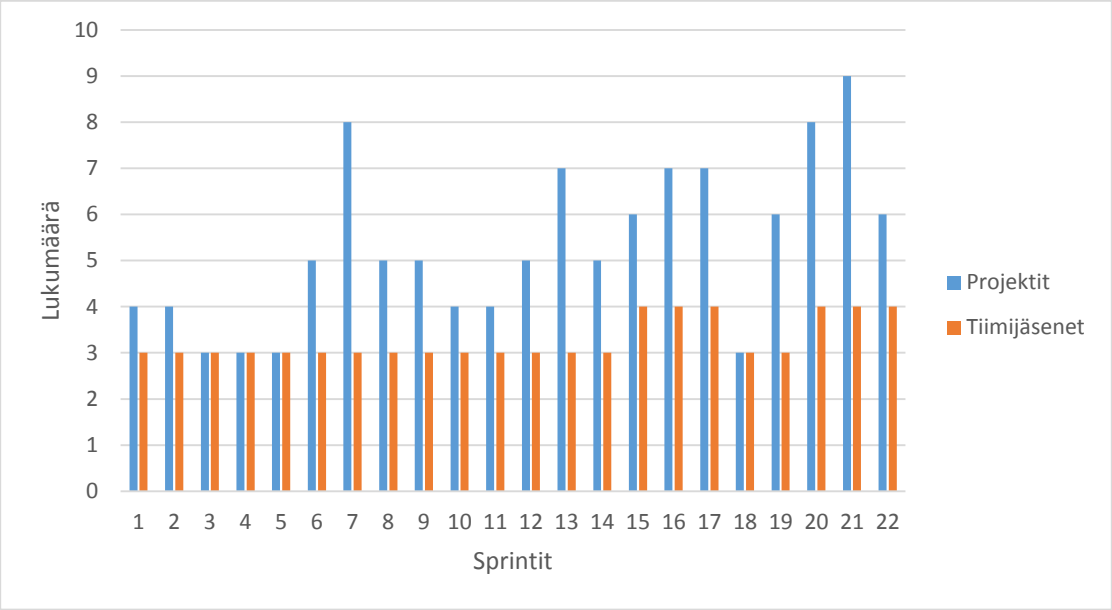
Kuvio 20. Sprinteissä toteutettujen tehtävien yhteispistemäärät

Kuviossa 21 esitetään sprintteihin suunniteltujen ja niissä toteutuneiden työmäärien pisteiden erotus eli ts. työmäärä, joka sprinteissä jäi toteuttamatta. Toteutumatta jääneiden työmäärien keskiarvo käyttöönoton ajalta on 11,8 pistettä per sprintti.



Kuvio 21. Sprinteissä suunniteltujen ja toteutettujen tehtävien yhteispistemäärien erotukset

Tiimin tehtävänkuvaan kuului useiden pienempien projektien tai projektin osien kehitystyö. Projektien lukumäärä vaihteli käyttöönoton aikana kolmesta yhdeksään yhtäaikaiseen projektiin. Projektivastaavia oli sprintistä riippuen mukana kahdesta neljään (yhdellä projektivastaavalla saattoi olla vastuulla useampi projekti) ja kehitystiimin jäsenten lukumäärä pysyi lähes vakiona, vaikkakin henkilövaihdoksia tapahtui käyttöönoton loppupuolella. Yhdellä henkilöllä oli koko käyttöönoton ajan kaikki kolme prosessimallin mukaista tehtävää; projektivastaava, kehitystiimin edustaja ja kehitystiimin jäsen. Kuviossa 22 on esitetty kehitystiimin jäsenten lukumäärä suhteessa sprintteihin valittujen tehtävien projektien lukumäärään.



Kuvio 22. Sprintteihin valittujen tehtävien projektien ja kehitystiimin jäsenten lukumäärä

Käyttöönottojakson jälkeen Team Ahma –tiimin työskentelyä ei ole enää toteutettu johtuen henkilöstön siirtymisestä asiakasprojekteihin, joiden tehtäviä ei hallita sisäisesti. Pienempiä sisäisesti hallittavia projekteja toteutetaan edelleen ja niiden toteutus ei ole enää tiimipainotteista tai Team Ahma –prosessin mukaista, joten tuo kehitystyö ei ole enää tämän kehitysprojektin alaista.

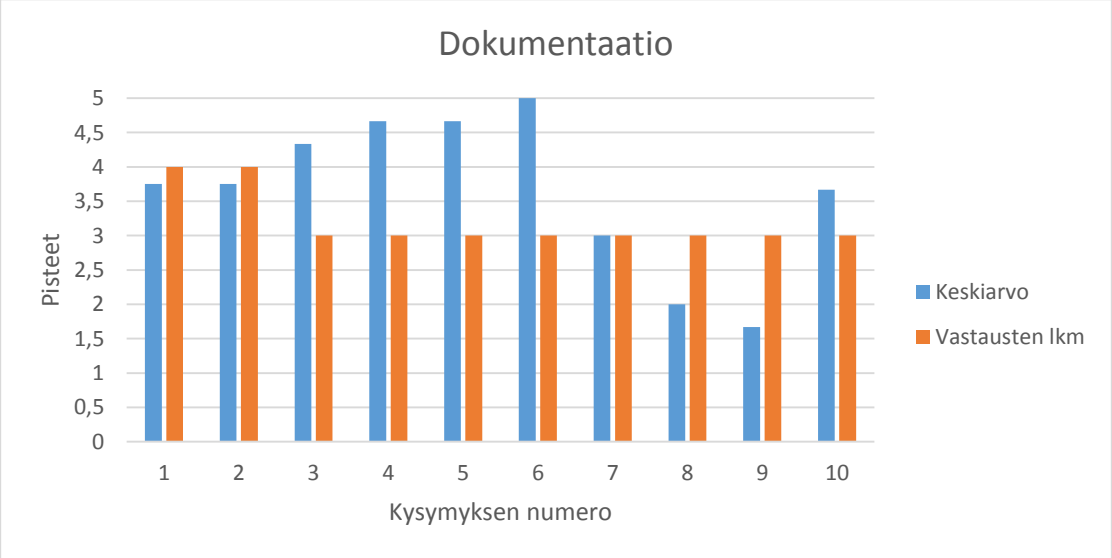
Prosessimallin käyttöönoton ja kehitysperiodien aikana, sekä kyselyaineiston perusteella havaittiin useita haasteita mm. kehitystiimin kokoon, tehtävien laatuun, tehtävien estimointiin, tiimin rooleihin sitoutumiseen ja muuttuviin olosuhteisiin liittyen. Kehitystiimissä oli suurimmillaan 4 henkilöä, joista yhdellä oli lisätehtävänä tiimin edustajan ja projektivastaavan rooli. Myös projektivastaavien työkuorma oli huomattavaa, joka hankaloitti prosessinmukaista projektin kehitysjonon ylläpitoa. Prosessin oli tarkoitus rauhoittaa sprintin kehitystyö suunnittelupalaverissa sovitun mukaiseksi, mutta välillä uusia tehtäviä oli tarve saada työn alle kesken sprintinkin; käyttöönoton aikana sprinteissä 6 ja 13 tehtäviä lisättiin sprinttiin jälkikäteen. Tehtävien estimoinnin epäonnistuminen aiheutti sen, että osa tehtävistä saattoi olla kehityksen alla useammassa sprintissä, joka puolestaan viivästytti muiden tehtävien valmistumista. Käyttöönoton sprintissä 16 osa tehtävistä estimoitiin kesken sprintin uudelleen, joka

nosti alun perin suunniteltua pistemäärää. Näiden ongelmien vuoksi prosessimallin käyttöönoton tulokset ovat osittain puutteelliset, koska prosessin toimivuutta oletetuissa (ns. täydellisissä) olosuhteissa ei pystytty todentamaan. Prosessin käyttöönoton perusteella ohjelmistokehityksen kehitystä on kuitenkin pystytty jatkamaan hyväksi havaituin menetelmin. Tuloksia ja niiden validiutta, vaatimustenvastaisuutta ja havaittuja haasteita analysoidaan tarkemmin tulosten käsittelyssä ja johtopäätöksissä.

4.4 Käyttäjäpalaute

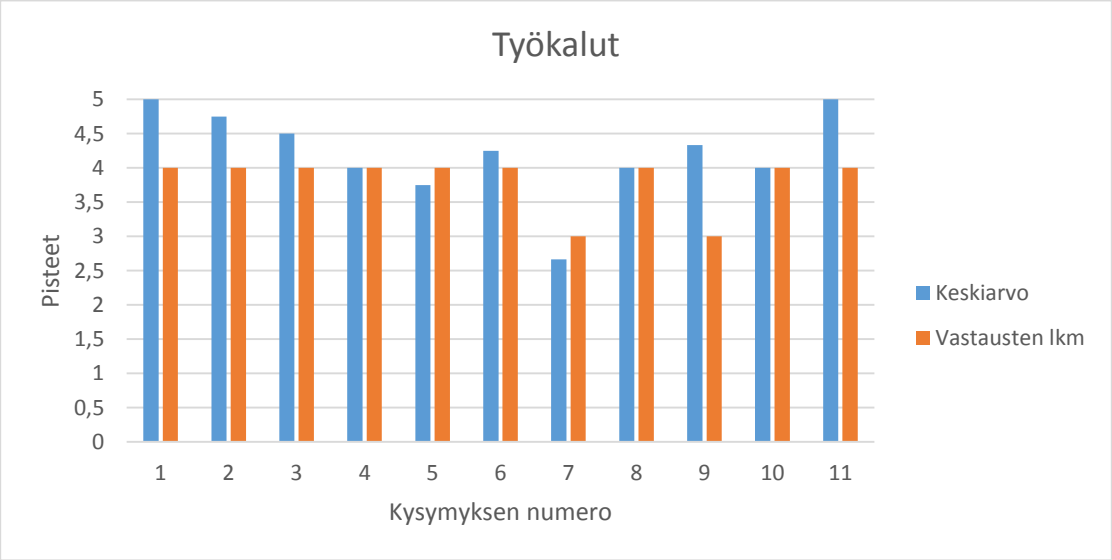
Tehtävienhallintaprosessin kehittämisen aikana mukana olleilta henkilöiltä saatiin suullista palautetta retrospektiivipalavereiden, haastatteluiden ja ”käytäväkeskustelujen” kautta. Lisäksi prosessin käyttöönotossa mukana olleilta kerättiin kyselylomakkeella (ks. Liite 3.) tarkempaa tietoa määritelmien, dokumentaation, työkalujen ja itse prosessin tuloksista.

Dokumentaatio-osuuden tavoitteena oli saada tietoa siitä, ovatko dokumentit helposti tavoitettavissa, onko käyttöönoton henkilöstö perehtynyt niihin, onko dokumenteista ollut apua prosessin käyttöönotossa ja onko dokumentaation formaatti ja laatu tarpeenmukaisia. Kuviossa 23 esitellään kyselyn dokumentaatio-osuuden pisteiden keski-arvo ja vastausten lukumäärä suhteessa kysymyksen numeroon.



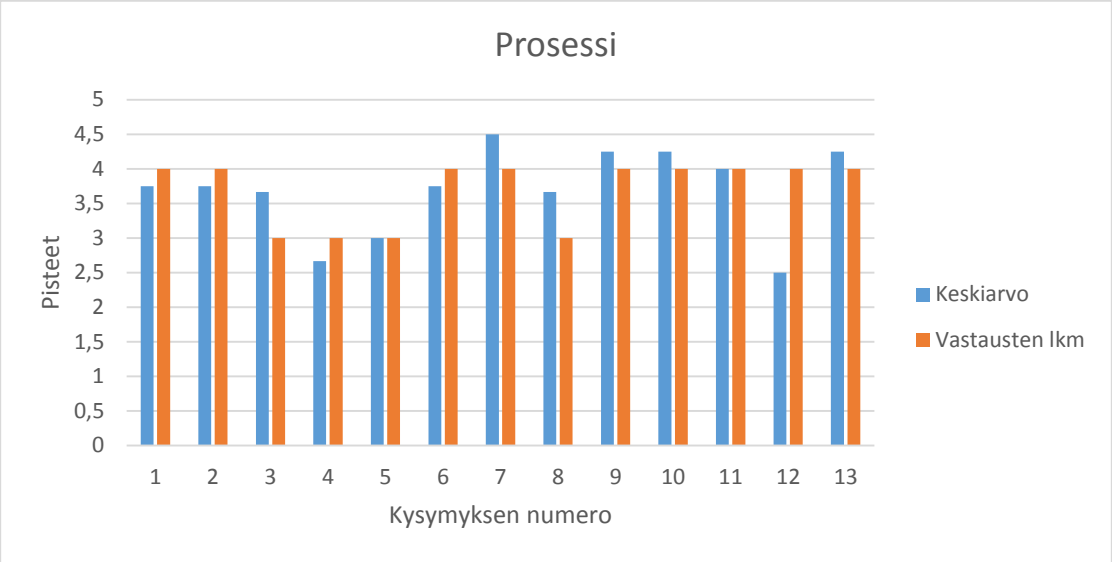
Kuvio 23. Kyselyn vastausten jakauma dokumentaatiota koskien

Työkaluista haluttiin tietää, ovatko ne auttaneet projektien tehtävienhallinnassa, ovatko ne käytettäviä ja käyttäjäystävällisiä, sekä auttavatko ne projektinseurantaa ja läpinäkyvyyttä. Kuviossa 24 esitellään kyselyn työkaluosuuden pisteiden keski-arvo ja vastausten lukumäärä kuhunkin kysymykseen.



Kuvio 24. Kyselyn vastausten jakauma työkaluja koskien

Prosessimallin käyttöönoton kyselyosuus oli tärkein, koska sen käyttöönotossa oli suurin työ, sillä oli toimeksiantajan toimintaan suurin vaikutus ja sen tuloksiin vaikuttivat myös dokumentaatio ja työkalut. Prosessin kyselyosuudessa päädyttiin keräämään tietoa siitä, kuinka hyvin mukana olleet henkilöt ovat mallin sisäistäneet, ovatko he pystyneet olemaan käyttöönotossa riittävästi mukana, onko prosessimalli ollut toimiva ja kuinka se on vaikuttanut toimintaan. Kuviossa 25 esitellään kyselyn prosessiosuuden pisteiden keski-arvo ja vastausten lukumäärä suhteessa kysymyksen numeroon.



Kuvio 25. Kyselyn vastausten jakauma prosessia koskien

Kyselyn lisäksi kahdelta prosessin käyttöönotossa vahvimmin mukana olleelta henkilöltä pyydettiin arviot työn hyödyllisyydestä toimeksiantajan tuotekehitysprosessissa ja nykytilanteesta verrattuna vanhoihin toimintatapoihin. Toinen vastanneista toimi prosessin käyttöönotossa kehitystiimin edustajana ja suunnittelijana ja hän totesi vastauksena kysymykseen seuraavasti:

Vanhaan tilanteeseen verrattuna työkalu ohjeistuksineen on selkeyttänyt rooleja ja niiden vastuita, jolloin suunnittelijaa ei kuormiteta usean projektipäällikön (tai PO:n) samanaikaisilla vaatimuksilla vaan keskitetysti backlogin kautta.

Vastaavasti projektipäälliköillä on selkeämpi näkyvyys yksittäisen suunnittelijan työkuormasta ja projektien kokonaistilanteesta, kun projektien tehtävälistaa ylläpidetään ja seurataan avoimesti.

Agile-menetelmiin kuuluu myös prosessin jatkuva arviointi ja kehittäminen ja tätä osa-aluetta on edelleen kehitettävä toiminnan ohessa.

Toinen vastanneista toimii tuotekehitysprosessin projektin omistajana ja toimi prosessin käyttöönotossa usean projektin projektivastaavana. Hän toteaa arviona työn hyödyllisyydestä toimeksiantajalle ja vertaa nykytilannetta aiempaan seuraavasti:

Suunnittelijan työtä helpottaa se, että tekeminen on valmiiksi priorisoitua ja tietää, mitä pitää tehdä seuraavaksi. Parhaimmillaan, mikäli projektit ovat tarpeeksi suuria, keskittymisen voi suunnata 100 % yhteen asiaan.

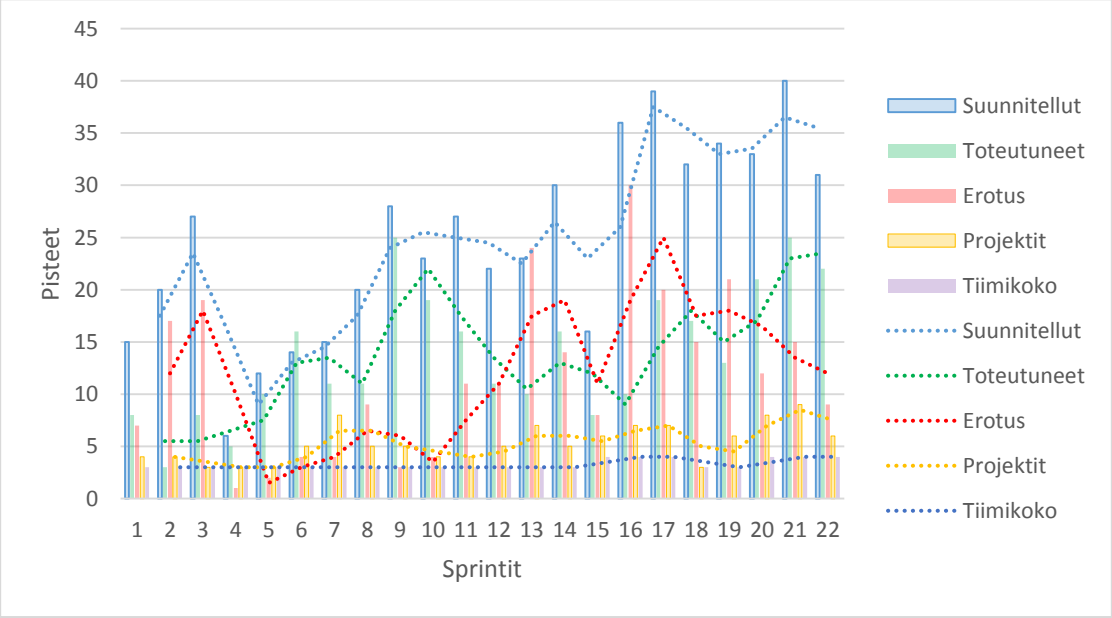
TAHMA-prosessin tarkoitus oli selkeyttää projektointia siten, että roolitukset olisivat selkeät. Uusien työkalujen myötä tekijälle saatiinkin muodostettua selkeämpi kuva ”projektiviidakosta”.

Prosessia pyöritti Scrum master, mutta TAHMA-backlogin hallinnasta puuttui alkuvaiheessa Product Owner. Eri asiakasrajapinnoissa olevat henkilöt eivät pystyneet keskenään sopimaan oikeista prioriteeteista, joten tekeminen oli haastavaa. Myöhemmässä vaiheessa backlogin hoitoon on otettu PO ja tämä on selkeyttänyt toimintaa.

5 Tulosten käsittely ja johtopäätökset

5.1 Kehitysperiodien tulokset

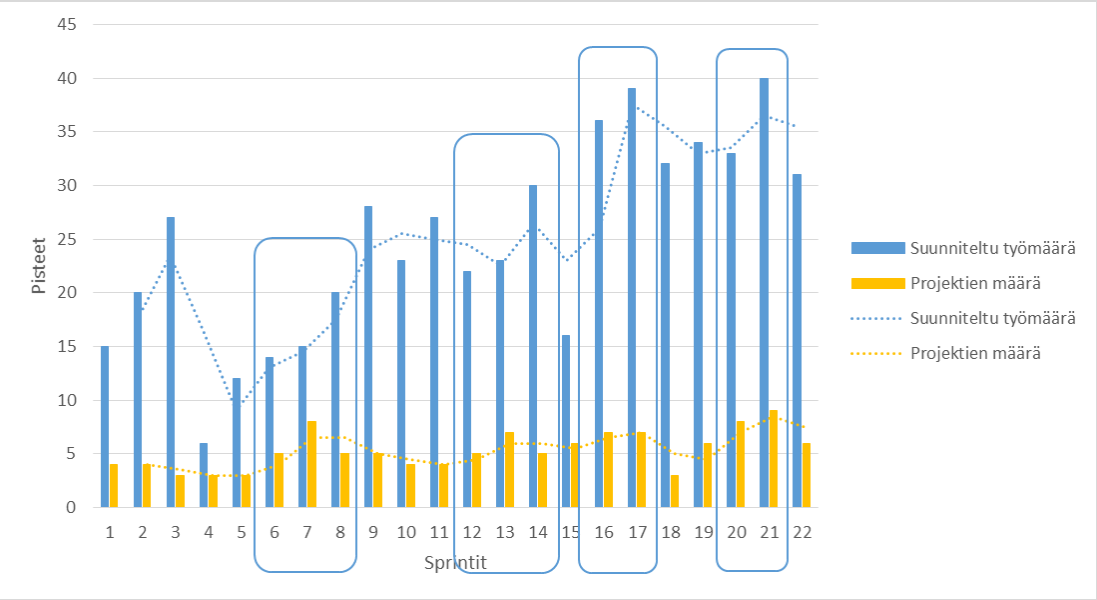
Kappaleessa 4 esiteltiin tilastoja tehtävienhallinnan prosessimallin käyttöönoton ja kehitysperiodien aikana hallituista tehtävämääristä, sekä kehitystiimin koosta ja projektien määrästä. Kuvioon 26 on koottu ko. tilastoista kooste.



Kuvio 26. Sprinteissä hallittujen tehtävien pistemäärien ja projektien kooste

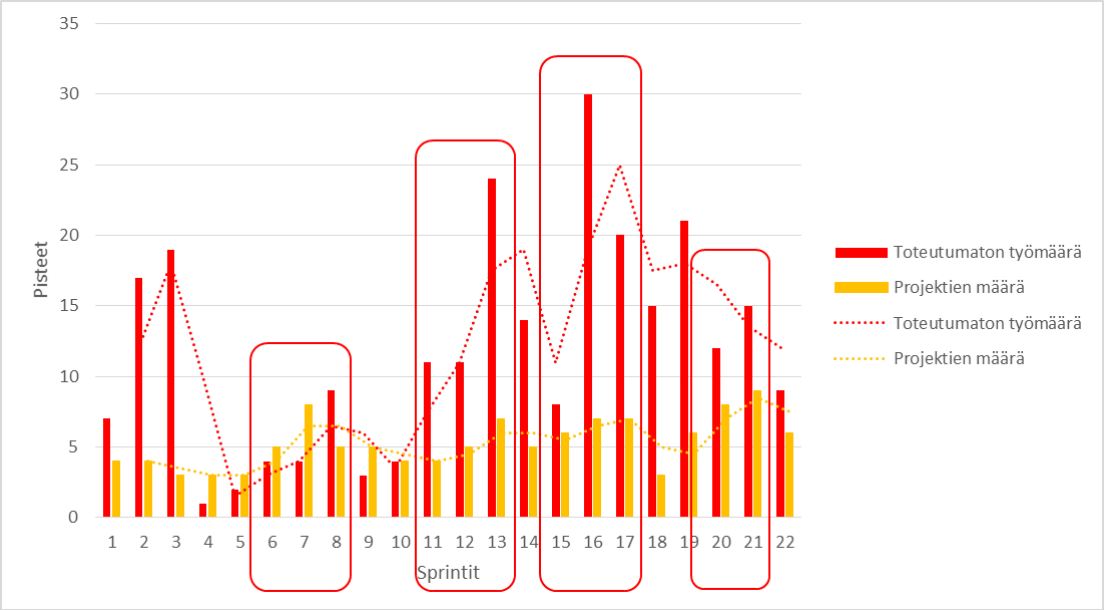
Kuviosta 26 voidaan todeta, että sprintteihin suunniteltu työmäärä pisteinä vaihteli kuuden (sprintti 4) ja neljänkymmenen (sprintti 21) pisteen välillä. Toteutunut työmäärä vaihteli kolmen (sprintti 2) ja kahdenkymmenenviiden (sprintit 9 ja 21) pisteen välillä ja näiden erotus, eli sprinteissä toteutumaton työmäärä, puolestaan vaihteli yhden (sprintti 4) ja kolmenkymmenen (sprintti 16) pisteen välillä. Ainoa kehitysperiodi, jossa työmäärä ylsi suunniteltuun, oli sprintti 6, jossa toteutunut työmäärä ylitti suunnitellun. Ko. sprintissä työmäärää kasvatettiin kesken periodin.

Kun verrataan kuviossa 27 esiteltyjä suunniteltuja työmääriä projektien määrään, voidaan todeta, että mitä enemmän projekteja otetaan yhtä aikaa toteutukseen, sitä enemmän kehitysperiodien suunniteltu työmäärä kasvaa.



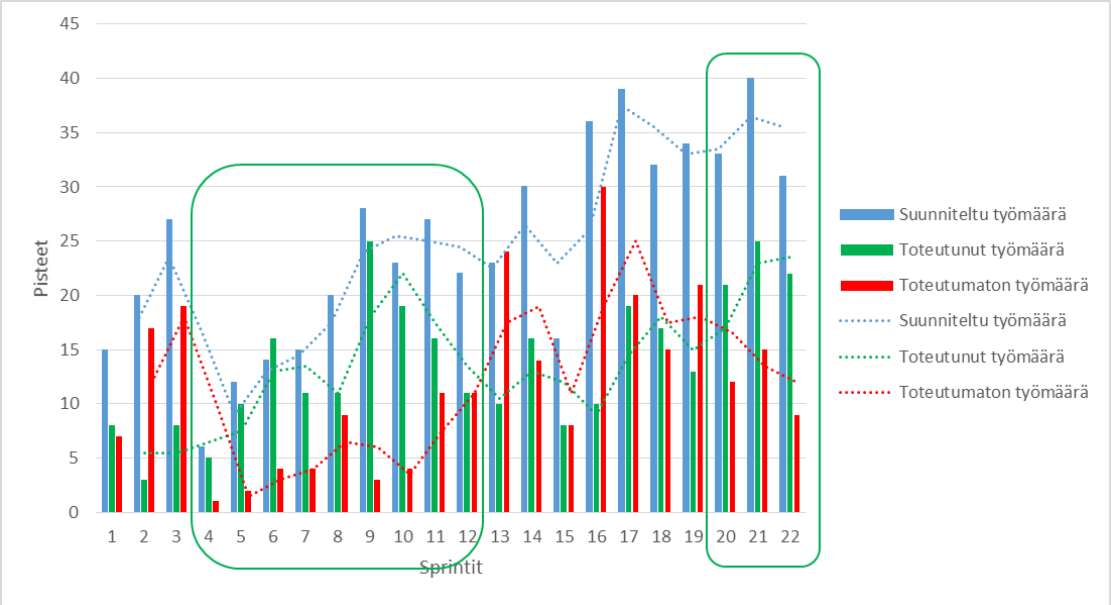
Kuvio 27. Sprintteihin suunniteltujen tehtävien ja projektien lukumäärän korrelaatio

Tämä kasvattaa molempien, toteutettujen ja toteutumattomien töiden määrää. Suhteessa kuitenkin toteutumattomien töiden määrä kasvaa enemmän. Kuvioon 28 ympyröidyistä kohdista voidaan havaita, että kun projektien lukumäärä alkaa kasvamaan, alkaa myös tekemättä jääneiden tehtävien määrä kasvamaan.



Kuvio 28. Sprinteissä tekemättä jääneiden tehtävien ja projektien lukumäärän korrelaatio

Kun pienemmän projektien lukumäärän myötä sprintteihin suunniteltu työmäärä pysyy maltillisena (ks. Kuvio 29, sprintit 4 – 12), toteutunut työmäärä on lähimpänä suunniteltua. Kun projektien lukumäärä ja suunniteltu työmäärä kasvavat (ks. Kuvio 28, sprintit 12 – 18), toteutumattomien töiden määrä on suurempi kuin toteutuneiden. Tiimikoon kasvaessa toteutuneiden töiden määrä kasvaa (ks. Kuvio 29, sprintit 20 – 22).



Kuvio 29. Sprintteihin suunniteltujen, toteutettujen ja toteuttamatta jääneiden tehtävien lukumäärien korrelaatio

Etenkin sprinteissä 13 ja 16 toteutumattomien töiden määrä on suuri. Kyseisissä sprinteissä työmäärää kasvatettiin kesken kehitysperiodin.

5.2 Kysely- ja haastattelutulokset

Luvussa 4.4 esitellystä käyttäjäpalautteesta ja kyselyn tuloksista tulee huomioida, että vastaajilla on ollut erilainen rooli prosessimallin käyttöönotossa. Vastaajissa oli kolme projektivastaavaa, joista yksi oli myös kehitystiimin edustaja ja jäsen, sekä yksi kehitystiimin jäsen. Projektivastaavat ovat oletettavasti kokeneet prosessin erilaisena kuin tiimin jäsenet, eikä koko päivittäisen kehitystyön taustalla pyörivä prosessi ollutkaan kaikille kehitystiimiläisille täysin selvillä.

Kyselyn dokumentaatio-osuuden kysymykset 1, 2 ja 7 tarkastelevat, onko dokumentit helposti saatavilla ja kuinka hyvin henkilöt ovat niihin perehtyneet. Tulosten perusteella voidaan todeta, että suurin osa henkilöistä on perehtynyt dokumentteihin, mutta eivät ole täysin tyytyväisiä niiden sijaintiin tai tavoitettavuuteen. Tähän voi olla syynä se, että kunnollista dokumenttien hallintajärjestelmää ei ole käytössä ja dokumentaatio on kontekstista riippuen eri sijainneissa (verkkolevy, versionhallinta, Sharepoint). Kehitysprojektin aikana evaluointiinkin Jiraan integroituvaa Confluencea,

josta saatiin hyviä kokemuksia ja jolla voisi dokumenttien hallintajärjestelmää parantaa.

Dokumentaatio-osuuden kysymyksillä 3, 4, 8 ja 9 tarkastellaan, onko dokumentaation formaatti selkeää ja riittävän kattavaa. Tulosten perusteella dokumentaation formaatti on koettu tarpeenmukaiseksi ja selkeäksi, eikä liian suppeaksi tai yksityiskohtaiseksi. Kysymykset 5 ja 6 puolestaan varmentavat dokumenteista olleen apua perehtymisessä prosessiin ja Jira-työkaluun. Myös evaluointiraportit koettiin tarpeenmukaisina, joskin yksi kolmesta vastaajasta kertoi perehtyneensä raportteihin huonosti. Evaluointiraporttien tärkein tehtävä oli toimia pohjana evaluointien tulosten esittämisessä.

Seuraavassa kyselyn kohdassa pyrittiin keräämään työkalujen (Jira ja Crucible) käyttöönoton tuloksia. Kysymyksillä 1, 2, 3, 6 ja 9 pyrittiin selvittämään, ovatko työkalut auttaneet projektien tehtävänhallintaa, -seuraamista, -visualisointia, -raportointia ja –suunnittelua. Tulosten perusteella voidaan yleisesti todeta, että työkalut ovat olleet apuna tehtävien käsittelyssä.

Kysymykset 4, 5, 7 ja 8 liittyvät työkalujen käytettävyyteen ja toteutettuun konfigurointiin. Työkalun käyttäjäystävällisyys ja etenkin projektin luominen koettiin hieman vaivalloiseksi. Käyttöönoton aikana käyty keskustelu puoltaa kyselyn tuloksia ja automaattisesti konfiguroidusta projektipohjasta vastaanotettiin kehitysehdotuksia. Projektipohjan luontia selvitettiinkin, mutta havaittiin että se olisi vaatinut suurempaa konfigurointia Jira-instanssin asennukseen ja näin ollen lykättiin kehitys myöhempään ajankohtaan. Työkalussa on siis selkeästi vielä kehitettävää ja automatisoitavaa tämän kehityshankkeen jälkeenkin.

Hyvänä puolena työkalusta koettiin projektikommunikaation ja läpinäkyvyyden paraneminen, joka johtunee siitä, että kaikilla tahoilla on nähtävissä yksi sijainti, josta tehtävien määrittely ja tila voidaan todeta. Projektien aikataulutukseen ja resursointiin työkalun koettiin vaikuttavan kohtuullisen positiivisesti. Kysymyksen 10 tarkoitus olikin selvittää, voidaanko työkalulla parantaa ennustettavuutta. Ennustettavuuteen vaikuttaa toki moni muukin prosessiin liittyvä asia, jonka vuoksi lähes sama kysymys toistettiin prosessi-osuudessa.

Prosessimallin ja sen käyttöönoton osuus kyselystä oli merkittävin tulosten lähde ja aihe herätti yleisestikin eniten keskustelua. Kysymykset 1-3 perehtyivät siihen, oliko vastaaja sisäistänyt prosessimallin ja oman roolinsa siinä. Vastauksista tulee huomioida että vastaajat, jotka eivät olleet määrittelemässä prosessin vaatimuksia ja olivat prosessin käyttöönotossa vain hyvin vähän aikaan, tiedostivat heikoiten kuinka prosessimallia tulisi käyttää ja mikä heidän roolinsa siinä on. Tämä lienee luonnollista, koska mitä ”syvemmällä” vastaajat olivat prosessin suunnittelussa ja käytössä, sitä paremmin he hallitsivat sen. Tämä kuitenkin korostaa myös perehdyttämisen ja sitouttamisen tärkeyttä prosessimallia käytettäessä.

Kysymysten 6-9 ja 13 tarkoitus oli kysyä mielipidettä itse prosessimallin toteutuksesta ja siitä kuinka hyvin malli on auttanut työskentelyä, läpinäkyvyyttä, aikataulutusta ja resursointia. Projektien läpinäkyvyys ja prosessimallin toteutus koettiin positiivisena ja prosessimallin vaikutus työskentelyyn ja projektien aikataulutukseen, sekä resursointiin kohtalaisen positiivisena. Työskentely prosessin mukaan voi tuntua lyhyellä aikavälillä raskaalta, koska se tuo väistämättä lisää työkuormaa ja vastuita. Yleisesti prosessin vaikutus työskentelyyn kuitenkin koettiin yllättävänkin hyvänä siihen nähden, että toimintatapa oli huomattavan erilainen kuin aikaisemmin. Projektien aikataulutukseen ja resursointiinkin prosessimallista saatava hyöty nähtäisiin todennäköisesti vasta pidemmällä aikavälillä, jolloin tiimi ja sen toiminta olisi ehtinyt vakiintumaan (vrt. tehtävien estimointi, sprintin työmäärät) ja ennustettavuus näin parempaa.

Heikoimpina kohtina prosessimallin käyttöönotossa koettiin kohdat 4, 5 ja 12. Etenkin kohdat 4 ja 12 viittaavat henkilöiden kokemaan resurssimäärään prosessin käyttöönotossa. Ainoa hyvän arvion henkilömäärästä antanut oli yksi kehitystiimin jäsenistä. Tämä arvio voi pohjautua edelläkin mainittuun seikkaan eli siihen, ettei koko taustaprosessi näkynyt kehitystiimiin asti, joka puolestaan oli yksi vaatimuksista prosessia suunniteltaessa. Lisäksi kysymyksen 10 tulokset varmentavat, että kehitystyö koettiin rauhoitetuksi kehitysperiodien aikana. Voidaan siis päätellä, että etenkin projektivastaavien resurssitilanne käyttöönoton aikana oli riittämätön. Tämä havaittiin myös prosessimallia toteutettaessa esimerkiksi siten, ettei projektivastaavilla ollut riittävästi aikaa työstää projektiensa kehitysjonoa, jolloin ne saattoivat olla puutteellisia esim. tehtäväkuvausten tai tehtävien hyväksyntäkriteereiden osalta. Eräs

kommentti vastauksissa tiivistää käyttöönoton aikaista resurssitilannetta, sitoutumista ja perehdytyksen (sekä sitouttamisen) tärkeyttä: *”Henkilöstöä tiimin pyörittämiseen ei vielä ollut riittävästi käyttöönoton aikana. Kaikki tiimit eivät sitoutuneet prosessimallin käyttöön tai sitten heille ei järjestetty tarvittavaa perehdytystä.”*

Kun verrataan toteutettuja tuotoksia ja saatua käyttäjäpalautetta Team Ahma –tyyppisen tiimin tehtävienhallintaprosessin vaatimuksiin (eli testataan tai mitataan toteutusta), voidaan todeta että vaatimukset täyttyivät hyvin. Kehitystiimin jäsenet eivät näe kuin periodille suunnitellun työkuorman, asiakasprojektien kehitysjonoja hallitaan erillisissä tehtäväjonoissaan, projektien tehtäville voidaan estimoida pisteet ja prosessiviolaatioille on eriteltynä toimintatapa. Lisäksi aiemmin esitellyt tarkemmat tekniset vaatimukset ja käyttöönoton aikana tarkennetut määrittelyt ovat toteutuneet.

Kun taas verrataan käyttäjäpalautetta (luku 4.4) käyttöönotossa toteutuneiden kehitysperiodien (luku 4.3) tilastoihin, huomataan niissä yhtäläisyyksiä. Mukana olleet henkilöt kokivat, etteivät heidän omat resurssinsa täysin riittäneet prosessin vaatimien toimien suorittamiseksi, eikä tiimin koko ollut riittävä projektien (ja muiden tehtävien) määrään nähden. Kehitysperiodeilla hallittujen projektien ja tehtävien määrä oli liian suuri tiimin (ml. projektivastaavat ja kehitystiimin jäsenet) kokoon nähden, joka aiheutti tehtävien jatkumista sprintistä toiseen eli teoriaosuudessakin mainittuja ”häntiä”. Osasyynä tähän on luultavasti myös kehitystiimin virheellinen työmäärien arviointi eli estimointi. Lisäksi kesken sprintin lisätyt tehtävät eli prosessiviolaatiot estävät kyseiselle sprintille suunnitellun työn tekemisen, jonka vuoksi se siirtyy seuraavaan sprinttiin. Tällaisia tilanteita oli harvoin, mutta kuten tilastoista huomataan, vaikutukset voivat olla silti pitkäaikaisia.

Yhtenä päätuloksena esiteltiin kehitystiimin edustajan ja suunnittelijan, sekä tuotekehitysprosessin projektin toimeksiantajan edustajan arviot työn hyödyllisyydestä toimeksiantajalle ja vertaus nykytilanteesta aiempaan, jotka olivat positiivisia. Toimeksiantajan edustaja oli maininnut jo useampaan otteeseen projektin valmistuttua, että se on toteutettu hyvin ja siitä on ollut suuresti apua luotaessa perustaa toimeksiantajan tuotekehitysprosessimallille. Molempien henkilöiden arvioista käy ilmi, että vanhaan tilanteeseen verrattuna kehittäjien työtä saatiin rauhoitettua ja useamman projektin yhtäaikaista hallintaa selkeytettyä.

5.3 Tulosten luotettavuus

Tutkimuksessa on yhdistetty useampia tapoja kerätä tutkimusaineistoa; tilastot kehitysperiodeissa hallituista tehtävistä, projekteista ja tiimin koosta, prosessissa mukana olleille teetetty kysely ja avainhenkilöiden haastattelut. Tutkimusaineiston triangulaatiolla (monipuolisilla aineistotyypeillä) pyritään mahdollistamaan tulosten keskenään vertailu ja näin parantamaan niiden luotettavuutta. Lisäksi luotettavuutta parantaa tutkimusaineiston osalta sen keräämiseen käytetty pitkä ajanjakso, etenkin kehitysperiodien osalta.

Tulosten luotettavuutta on pyritty parantamaan myös teorian ja toteutuksen tarkalla raportoinnilla, sekä yksityiskohtaisesti esitellyillä ja käsitellyillä tuloksilla. Yksityiskohtainen raportointi edistää tutkimuskohteen ja tutkimustuloksiin päätymisen ymmärtämistä ja sitä kautta tutkimuksen luotettavuutta.

Tutkimuksen ongelmana oli se, ettei toteutettua prosessimallia voitu täysin todentaa. Tämän vuoksi tuloksista voidaan havaita erilaisia ominaisuuksia kuin mitä optimaalisissa olosuhteissa olisi havaittu ja sen vuoksi prosessimallin toimivuuden arviointi jää kyselyjen ja haastattelujen varaan ja näin ollen heikentää tutkimustulosten luotettavuutta. Toisaalta prosessin käyttöönotossa ilmenneet ongelmat (tiimin resursointi, tehtävien estimointi, rooleihin sitoutuminen) antoivat tuloksia, joita voidaan käyttää jatkossa prosessia kehitettäessä.

Kyselyn tulosten luotettavuutta voi heikentää vastaajien pieni määrä. Prosessinkäyttöönotossa olleiden henkilöiden kokonaismäärän suhde vastanneiden määrään oli kuitenkin suuri, joten mahdollisuutta huomattavan suureen vastaajien määrään ei käytännössä edes ollut. Toisaalta tätä pyrittiin myös kompensoimaan kysymysten määrällä, käymällä keskusteluja ja esittämällä tarkentavia kyselyitä avainhenkilöille. Lisäksi aineiston monipuolisuus ja niiden keskenään vertailu parantaa näidenkin tulosten luotettavuutta.

5.3.1 Ohjelmistokehitysprosessien jatkokehittäminen

Opinnäytetyönä tehty ohjelmistosuunnittelun tuotekehitysprosessien kehittäminen on toiminut toimeksiantajalle ensimmäisenä suurempana ko. prosessin kehittämis-

hankkeena, jonka pohjalta on tarkoitus standardoida toimintamalleja. Hankkeen perusteella on otettu ja tullaan edelleen ottamaan käyttöön hyväksi havaittuja menetelmiä ja rooleja yrityksessä. Etenkin projektivastaavien rooli ja resursointi on huomioitu tarkemmin hankkeen tulosten perusteella. Myös tuloksinakin havaittua kehittäjien työn rauhoittamista prosessin avulla on priorisoitu korkeammalle, koska sen on nähty vaikuttavan laatuun, aikatauluihin, asiakastyytyväisyyteen ja työntekijöiden hyvinvointiin.

Tehtävienhallintatyökalua tullaan kehittämään käyttäjäystävällisempään suuntaan esimerkiksi projektien luonnin osalta ja käyttämään työkalusta vain hankkeen perusteella havaitut tarpeelliset ominaisuudet. Lisäksi työkaluun voidaan jatkossa vähällä työllä integroida dokumenttienhallinta- ja katselmointityökalut, jotka parantavat asiakasprojektien vaatimustenhallintaa ja tuotetun ohjelmiston laatua ja edelleen jäljitettävyyttä.

Hankkeen jälkeen on käynnistetty myös laadunhallintajärjestelmien vaatimuksia käsittelevän ISO 9001-standardin mukaisen toiminnan kehittämishanke, jossa kehityshankkeen aikana toteutetut toimintamallit ja työkalut huomioidaan.

6 Pohdinta

Vision Systems Oy:n ja myöhemmin myös Vision Development Oy:n projektien- ja tehtävienhallinta oli kehittynyt jo vuosia muun tekemisen ohessa ja etenkin viime vuosina (2013–2014) on pyritty siirtymään ketterän kehityksen menetelmiin. Menetelmiä ei ole kuitenkaan pystytty tarkasti määrittelemään ja käyttöönottamaan mm. projektien luonteen, henkilöstömäärän ja työkalujen puutteen vuoksi.

Tämän kehityshankkeen myötä toimeksiantajalle on onnistuneesti otettu käyttöön ja dokumentoitu työkaluja tehtävienhallintaan, sekä luotu pohjaksi prosessimalli pienten ja keskisuurten asiakasprojektien läpiviemiseksi. Prosessimallia testattiin noin puolen vuoden ajan ja sen avulla saatiin tärkeää tietoa itse prosessimallin toimivuuden lisäksi asioista, joita vastaavien projektien läpiviemisessä tulee huomioida. Kehitettyjen toimintatapojen ja työkalujen pohjalta tuotekehitysprosessia on pystytty edelleen kehittämään ja ylläpitämään hyväksi havaittuja menetelmiä.

Kehityshankkeen tuloksista voidaan havaita, ettei tehtävienhallintaprosessin käyttöönotto toteutunut täysin optimaalisissa olosuhteissa. Koko mukana ollut tiimi, eli prosessivastaavat ja kehitystiimi edustajineen, oli melko kuormitettu koko käyttöönoton ajan, joka näkyi tuloksissa niin kehitysperiodien tilastojen kuin käyttäjäpalautteen osalta. Lähinnä huomiota olisi tullut kiinnittää tehtävien määrittelyihin ja estimointiin. Kun tehtävä on huolellisesti määritelty ja estimoitu yhdessä projektivastaavan ja kehitystiimin jäsenen kanssa, on se helpompi toteuttaa kehitysperiodilla ja kehitysperiodeille suunniteltu työkuorma on kooltaan tarkemmin määritelty.

Projektien määrän kasvaessa myös periodien kuormitus kasvaa. On tärkeää huomioida tuloksista, että kehitystiimin koon pysyessä vakiona, mitä enemmän projekteja on aktiivisena, sitä enemmän periodeilla on aktiivisia tehtäviä ja sitä todennäköisimmin toteutumattomien töiden määrä on suurempi kuin toteutuneiden töiden määrä.

Tässäkin voidaan teoriaan nojaten todeta, että ”kaikki vaikuttaa kaikkeen”; tehtävien ollessa tarkemmin ja laadukkaammin määriteltynä, ovat kehitysperiodit tarkemmin suunniteltavissa, jatkokehitysperiodeilla ei ole ”häntiä”, asiakasprojektit pysyvät paremmin aikataulussa, asiakastyytyväisyys kasvaa, yleisesti tiimin työ on mielekkäämpää jne..

Kehityshankkeen tiimin asiakasprojektit eivät välttämättä olleet muutenkaan helpoimpia projekteja hallita. Projektien pieni koko vaihteli ajallisesti muutamasta päivästä useisiin kuukausiin, joka aiheuttaa mm. sen, että kehitystiimiä on riskialtista pitää kovin suurena, koska tällöin kehittäjien käyttöaste voi kärsiä projektien välisinä aikoina. Lisäksi osa projektien tehtävistä oli luonteeltaan tukipyyntöjä korkealla prioriteetilla ja nopeallakin aikataululla, joka hankaloittaa pienen kehitystiimin työskentelyä entisestään. Tällaisille tukipyynnöille (eli prosessiviolaatioille) olisikin tarpeen jatkossa kehittää edelleen toimintatapoja esimerkiksi suunnittelemalla tiettyjen kehittäjien työ aina tukipyyntövarauksella.

Työkalujen käyttö on helpottanut projektien tehtävien visualisointia ja seurantaa, sekä auttanut tehtävien määrittelyssä ja työmääräarvioissa. Työkaluissa on vielä kehitettävää esimerkiksi projektien luonnin automatisoinnin ja dokumenttien hallinnan osalta. Jiraan liitettävästä dokumenttien hallintatyökalusta saatiin opinnäytetyöprojektin puitteissa hyviä kokemuksia, joita voidaan mahdollisesti käyttää dokumenttien

hallintaa kehitettäessä. Kehityshankkeen aikana toteutetut dokumentit määrittelyistä ohjeistuksiin saivat hyvää palautetta ja toimivat apuna uusien kehittäjien perehdyttämisessä tehtävienhallintaan ja työkaluihin.

Pelkkä työkalujen kehittäminen ei yksistään riitä selvittämään projektien- ja tehtävienhallinnan haasteita:

“A fool with a tool is still a fool.

Projektinhallintaa ei voi automatisoida. Työkaluilla voi joitakin asioita tehdä tehokkaammin. Fiksu ja kokenut projektipäällikkö, jolla on ruutupaperia ja kynä, päihittää ohjelmistolla varustetun aloittelijan kirkkaasti. Mutta toki kokenut ja fiksu, jolla on oikeat työkalut käytettävissään, on kaikkein paras vaihtoehto.”

Lehtimäki, 2006, 199.

Kehitysprojektin ja opinnäytetyön aikana olen saanut erittäin paljon tietoa ko. aiheesta ja kasvanut nykyaikaiseen ohjelmistokehitykseen ja projektinhallintaan. Olen havainnut saman asian minkä siteerasin Lehtimäen, 2006 kirjasta, ettei projektinhallintaa todella voi automatisoida millään työkalulla, vaan lopulta toteutuksesta vastaavat ihmiset. Työkalut ja prosessimallit voivat mm. tehostaa työskentelyä ja parantaa laatua, mutta lopulta onnistumiseen vaaditaan ihmisten sitoutuminen toimintamalleihin ja työkalujen käyttöön, kuten myös itse kehitystyöhön. Toimintatapoihin ja ihmisten käyttäytymiseen vaikuttavat todella monet asiat, joihin tulisi kiinnittää suurin huomio työkaluja tai prosessimalleja suunniteltaessa.

Opinnäytetyöprojekti oli yleisesti tuloksien perusteella onnistunut ja toimeksiantajan palautteen mukaankin koettu erittäin hyödylliseksi. Opinnäytetyön toteutusvaihe eteni aikataulullisesti lähes suunnitellusti, mutta raportoinnin aloituksen viivästyminen venytti opinnäytetyöprosessia yleisesti 0,5 – 1 vuodella. Opinnäytetyön toteutus oli allekirjoittaneelle erittäin mielenkiintoinen kokemus lähinnä siitä syystä, että pääsi suunnittelemaan yrityksen (ja samalla omaan työskentelyyn liittyviä) toimintatapoja pienten asiakasprojektien toteutuksessa ja tehtävienhallinnassa. Tämän kaltaisen prosessin suunnittelu eroaa huomattavasti normaalista suunnittelutyöstä, joka myös toi hyvää vaihtelua työskentelyyn. Oli myös mielenkiintoista nähdä suunniteltu

prosessi käytännössä, saada viikoittain palautetta toteutetusta työstä ja kehittää sitä edelleen.

Lähteet

Appelo, J. 2011. Management 3.0: leading Agile developers, developing Agile leaders. Boston: Pearson Education, Inc.

Astels, D., Granville M., Novak, M. 2002. A Practical Guide to Extreme Programming. New Jersey: Prentice Hall.

Auer, A., Auer, L., Heinäsmäki, M., Hölttä, J., Kalliala, E., Laanti, M., Laine, K., Lekman, L. 2013. Ketterää kehitystä. Helsinki: Finn Lectura.

Braude, E., Bernstein, M. 2011. Software Engineering, Modern Approaches. New Jersey: John Wiley & Sons, Inc.

Cohen, G. 2010. Agile Excellence for Product Managers. Cupertino: Superstar Press.

Cohn, M. 2009. Succeeding with Agile, Software Development Using Scrum. Boston: Pearson Education, Inc.

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

Haikala, I., Märijärvi, J. 2004. Ohjelmistotuotanto. Helsinki: Talentum.

Hazzan, O., Dubinsky, Y. 2008. Agile Software Engineering. London: Springer.

Hedberg, J., Söderberg, A., Tegehall, J. 2011. How to design safe machine control systems – a guideline to EN ISO 13849-1.

Hyvönen, E. 2003. Ohjelmistoliiketoiminta. Vantaa: Dark Oy.

JIRA Software. Jira-tehtävienhallintatyökalun esittely Atlassianin sivustolla. Viitattu 16.5.2017. https://www.atlassian.com/software/jira.

Järvinen, P., Kronström, V., Poskela, J., Karlos, A. 2002. Suorituskyvyn mittaaminen ja mittareiden kehittäminen projektiliiketoiminnassa. Espoo: Otamedia.

Kanban. Kanban-määritelmä Crispin sivustolla. Viitattu 14.3.2017. https://www.crisp.se/gratis-material-och-guider/kanban.

Lehtimäki, T. 2006. Ohjelmistoprojektit käytännössä. Jyväskylä: Gummerus Kirjapaino.

Lehtonen, T., Tuomivaara, S., Rantala, V., Känsälä, M., Mäkilä, T., Jokela, T., Könnölä, K., Kaisti, M., Suomi, S., Isomäki, M. ja Ylitolva, M. 2014. Sulautettujen järjestelmien ketterä käsikirja.

Lekman, L. 2009. Mikä ihmeen Kanban? Lekman Consultingin Kanban-määritelmä 26.9.2009. Viitattu 14.3.2017. https://lekman.fi/2009/09/26/mika-ihmeen-kanban/.

Schwaber, K., Sutherland, J. 2013.The Scrum Guide. Scrum-viitekehyksen määritelmä. Viitattu 15.3.2017. http://www.scrumguides.org/docs/scrumguide/v1/scrum-guide-us.pdf

Stepanek, G. 2005. Software Project Secrets: Why Software Projects Fail. New York: Springer-Verlag.

Top Project Management Tools. Listaus projektien- ja tehtävienhallintatyökaluista Capterran sivustolla. Viitattu 16.5.2017. http://www.capterra.com/project-management-software/.

Tuomi, J., Sarajärvi, A. 2009. Laadullinen tutkimus ja sisällönanalyysi. Jyväskylä: Gummerus Kirjapaino.

Vision Development presentation. Yritysesittely. 5.10.2016. Vision Development Oy.

What is V-model- advantages, disadvantages and when to use it? Artikkeli V-mallista ISTQB Certification Examin sivustolla. Viitattu 15.3.2017. http://istqbexamcertification.com/what-is-v-model-advantages-disadvantages-and-when-to-use-it/

Liitteet

Liite 1. Tehtävähallintaprosessin määrittely ja ohjeistus

Dia 1

Ohjeistus VS:n ja VD:n
tehtävienhallintaprosesseihin

Vision Development Oy
Juho Riekkinen 30.09.2015

Dia 2

Sisältö

Dia 3

Johdanto

VISION Development
Member of Vision Systems company

- Tämän esityksen tarkoitus on
  - Määritellä Vision Systemsin ja Vision Developmentin järjestelmäkehityksen tehtävienhallintaprosessi
  - Toimia ohjeistuksena prosessin ja tehtävienhallintatyökalun käyttöönotossa
  - Kerätä kehitysehdotuksia em. aiheisiin
- Tehtävienhallintaprosessi- ja työkalu tarvitaan, jotta
  - Pystytään rauhoittamaan kehittäjien työ
  - Pystytään ennustamaan tehtävien ja projektien etenemistä paremmin
  - Roolitukset ja vastuut ovat kaikille selvillä
  - Projektin johto pysyy ajan tasalla tehtävistä
- Vaikutuksia mm.
  - Asiakaspalveluun
  - Projektien resursointiin ja aikataulutukseen
  - Omaan tuotekehitykseen
  - Työn tehokkuuteen
  - Työntekijöiden hyvinvointiin

Dia 4

Määritelmät – käsitteitä

VISION Development
Member of Vision Systems company

- Projekti (Project) – projekti, tuote tai muu kokonaisuus, joka määrittelee ison joukon tehtäviä
- Tehtäväjono (Backlog) – Projektin tai tiimin tehtäväjono
- Yleinen tehtävä (Issue) – Yksittäinen tehtävä, ominaisuus tai iso tehtäväkokonaisuus (kaikkien tehtävätyyppien yleisnimitys)
- Tehtävätyypit:
  - Kokonaisuus (Epic) – Suurempi kokonaisuus, joka sisältää useampia toimintoja (ks. seuraava)
  - Toiminto (Story) – Yksittäinen kohteeseen kehitettävä toiminnallisuus/toiminto. Osa suurempaa kokonaisuutta (ks. edellä).
  - Alitehtävä (Sub-Task) – Yksittäinen toiminnon toteutumista edellyttävä tehtävä, jonka kehittäjä tekee. Toiminto voi sisältää useita alitehtäviä.
  - Tehtävä (Task) – Irrallinen tehtävä, jota ei voida liittää osaksi suurempaa kehitettävää kokonaisuutta, eikä liity toiminnallisuuden kehittämiseen (luonteeltaan esimerkiksi käyttöönotto, huolto, tutkimus, selvitys, tukipyyntö)
- Periodi (Sprint) – Iteraatio tai ajanjakso, jonka aikana suoritetaan x-määrä tehtäviä
- Siistiminen (Grooming) – Tehtäväjonon ylläpito, siistiminen ja priorisointi
- Tehtävän tila (Status) – Tehtävän valmiusaste (esim. To Do, In Progress, Done)
- Työnkulku (Workflow) – Tehtävän työnkulkumalli (esim. To Do → In Progress → Done)

Dia 5

Määritelmät – roolit

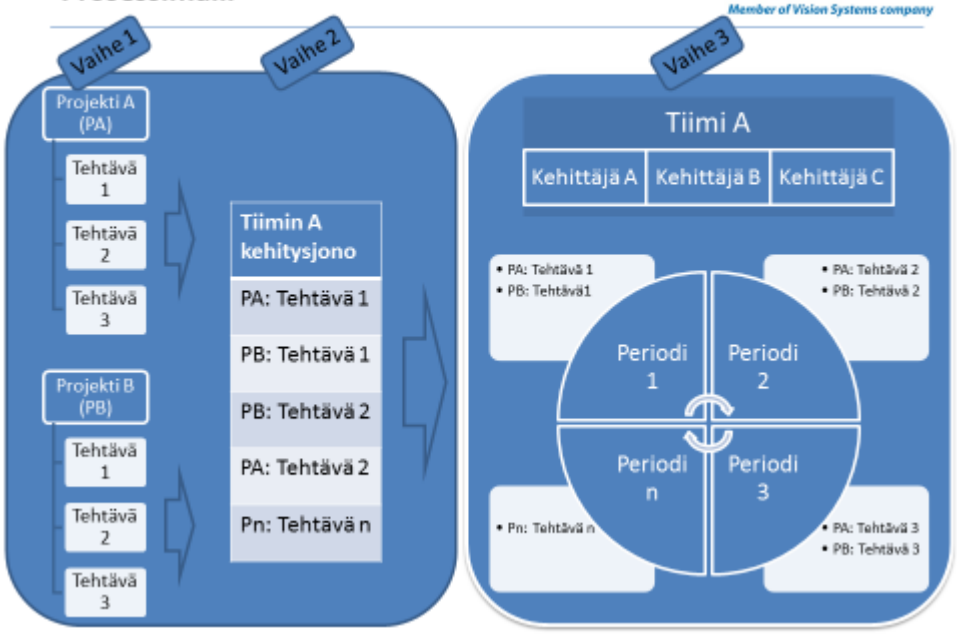
VISION Development
Member of Vision Systems company

- Projektivastaava (Product Owner, Project Lead)
  - Vastaa projektin kohteen ominaisuuksista ja toiminnallisuuksista
  - Tehtäväjonon luonti, ylläpito ja priorisointi
  - Periodin tavoitteet, tavoitteiden toteutuminen asiakkaan kanssa
- Tiimin edustaja (Scrum Master)
  - Poistaa työn tekemisen esteet tiimin tieltä
  - Varmistaa prosessikäytäntöjen noudattamisen
  - Järjestää periodin suunnittelupalaverit (ja mahdollisen katselmoinnin ja retrospektiivin)
  - Avustaa projektivastaavaa tarvittaessa kehitysjonon ylläpidossa ja periodin suunnittelussa
  - Toisin kuin yleisesti tiimin edustajalla/vetäjällä, Scrum Masterilla ei ole suoranaista määräysvaltaa tiimiin
- Tiimin jäsen, kehittäjä
  - Vastaa kehitystyön tekemisestä
  - Osallistuu periodin suunnittelussa työmääräarviointiin
  - Luo kokonaisuuksille alitehtävät ja varmistaa, että kokonaisuuden hyväksyntäkriteerit tulee toteutetuksi näiden alitehtävien kautta
  - Katselmoi muiden kehittäjien toteuttamia alitehtäviä

Dia 6

Prosessimalli

VISION Development
Member of Vision Systems company



Dia 7

Prosessimalli

VISION Development
Member of Vision Systems company

- Vaihe 1 (valmistautuminen periodiin):
  - Projektivastaavat muokkaavat (tehtävien lisääminen, pisteytys, muokkaaminen) ja priorisoivat projektin tehtävälistan itsenäisesti. Projektivastaavat käyttävät tarvittaessa apuna muita resursseja, kuten kehittäjiä.
- Vaihe 2 (miekkailu):
  - Projektivastaavat ja tiimin edustaja yhdessä muokkaavat ja priorisoivat tiimin kehitysjonon. Mikäli syntyy ristiriita, on tiimin edustajan selvitettävä ristiriita johdon avustuksella. Periodiin valitut tehtävät annetaan tehtäväksi tiimille.
- Vaihe 3 (periodin toteutus):
  - Tiimin edustaja käy tiimin jäsenten kanssa läpi kehitysjonolta periodiin valitut tehtävät. Tehtävät pilkotaan alitehtäviin, annetaan tehtäväksi ja toteutetaan periodin aikana. Tehtävien edistymistä seurataan tarvittaessa määräajoin palavereissa. Tiimin jäsenet vastaavat alitehtävien tilojen ajantasaisuudesta.

(ks. tarkemmin; JIRA – Yleistä)

Dia 8

Prosessimalli

VISION Development
Member of Vision Systems company

- Tiimin jäsenet toimivat lähinnä vaiheessa 3, jossa toteutetaan kehitystyö prosessin mukaisesti
- Tiimin jäsenet eivät näe projektien tai tiimin kehitysjonoa, ainoastaan periodille valitut tehtävät
- Tarkoituksena on rauhoittaa tiimin jäsenten työskentelyä periodin ajaksi

Dia 9

Prosessimalli

VISION Development
Member of Vision Systems company

- Määrätyin väliajoin tiimin edustaja järjestää retrospektiivin, jossa käydään läpi toimintamallin toteutuksen onnistuminen edellis(t)en periodi(e)n osalta
- Retrospektiivissä tuodaan esille myös ideat ja kehitysehdotukset tiimin ja toimintamallin toimintaan liittyen
- Retrospektiiviin osallistuu ainakin tiimin edustaja ja tiimin jäsenet, sekä tarpeen vaatiessa projektivastaavat

Dia 10

Prosessimalli – Case Team Ahma

VISION Development
Member of Vision Systems company

- Vaihe 1 – 3 etenevät viikon (tai muun tiimille sopivan ajan) periodeissa
  - Projektivastaavat siistivät projektiensa tehtävälistat maanantaihin mennessä
  - Maanantaina projektivastaavat kokoontuvat tiimin edustajan kanssa ja siistivät tiimin tehtävälistan seuraavaa periodia varten. Lisäksi tiimin edustaja käy läpi meneillään olevan periodin etenemisen.
  - Tiimi pitää tiistaina suunnittelupalaverin, jossa käydään läpi edellisen periodin tehtävien tila, seuraavan periodin tehtävälista, pilkotaan tehtävät alitehtäviksi, asetetaan alitehtävät periodiin työn alle ja aloitetaan periodi
  - Tiimin jäsenet seuraavat ja päivittävät tehtävien etenemistä henkilökohtaisesti työkalun avulla ja vähintään kahdesti viikossa järjestetään n. 5 minuutin tilannekatsaus, jossa tiimi käy läpi tehtävien etenemisen, mahdollisesti ilmenneet ongelmat ja muut tehtävien valmistumiseen vaikuttavat asiat
  - Retrospektiivi järjestetään periodin lyhyydestä johtuen ainoastaan joka toinen periodi

Dia 11

Poikkeuskäsittely

VISION Development
Member of Vision Systems company

- Kesken periodin tulevat tehtävät ilmoitetaan tiimivastaavalle, joka tarpeen vaatiessa käy tehtävät läpi muiden projektivastaavien kanssa. Jos tehtävä on luonteeltaan erittäin korkealla prioriteetilla ja muut projektivastaavat sen toteuttamisen hyväksyvät, voidaan tehtävä ottaa työn alle kesken periodin. Tällöin projektivastaava kirjaa tehtävän työkaluun, lisää periodille ja antaa sen tehtäväksi (seurauksena scope change). Kesken periodin tulleet työtehtävät tulee kirjata niille tarkoitetun projektin alle (esimerkiksi Customer Support) erittäin korkealla prioriteetilla ja otsikko etuliitteellä [Urgent]. Merkinnän tarkoitus on erotella prosessiviolaatiot normaalitehtävistä.

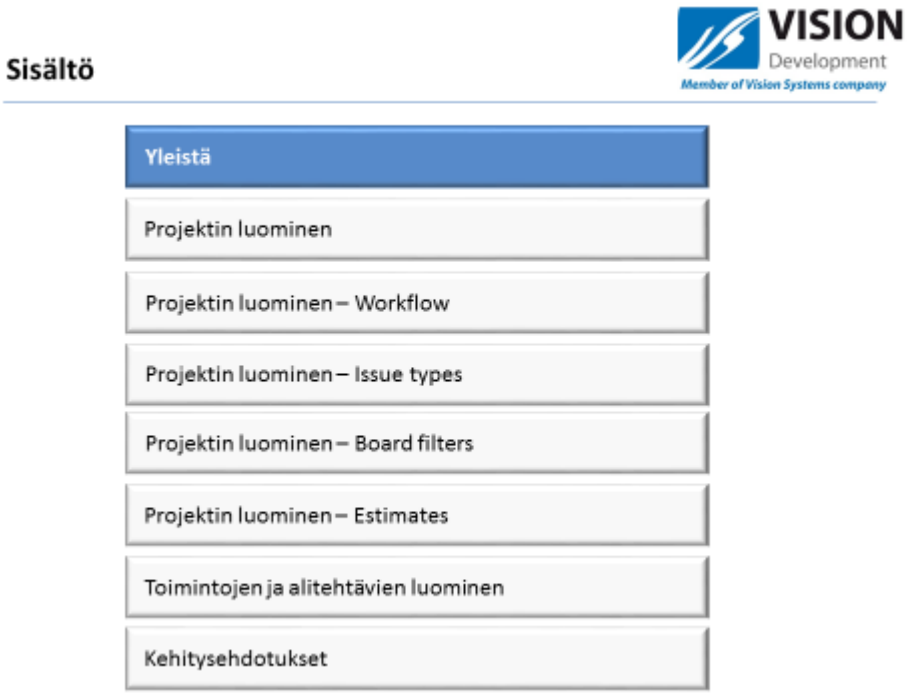
Liite 2. Tehtävienhallintatyökalun ohjeistus

Dia 1 - Kansilehti

Ohje Jiran tehtävienhallintaan

Vision Development Oy
Juho Riekkinen 14.10.2015

Dia 2 - Sisältö



Dia 3

JIRA – Yleistä

- JIRA on tehtävienhallintaohjelmisto, jonka on tehnyt Atlassian, australialainen vuonna 2002 perustettu ohjelmistoyritys
- JIRAssa on projekteja (Project) ja niillä näkymät (Board)
- Projektin alle luodaan tehtävät (Issue)
- Tehtävien etenemistä seurataan näkymien avulla

Dia 4

JIRA – Yleistä

- Tämän ohjeistuksen tarkoitus on
  - Opastaa Vision-konsernin JIRA-käyttäjää
    - Projektien ja toimintojen luonnissa
    - Yleisesti JIRAn käytössä
  - Tarkentaa toimintojen kuvaukset ja työnkulku

Dia 5

JIRA – Yleistä

VISION Development
Member of Vision Systems company

- Visionin JIRA-instanssi on asennettu sisäiselle serverille ja löytyy osoitteesta:

http://10.85.10.4:8085/

- Katselmointityökalu JIRA Crucible löytyy osoitteesta:

http://10.85.10.4:8060/

- Dokumenttienhallinta JIRA Confluence löytyy osoitteesta:

http://10.85.10.4:8090/

Dia 6

JIRA – Yleistä – Issues

VISION Development
Member of Vision Systems company

- Tehtävätyypit:
  - Kokonaisuus (Epic) – Suurempi kokonaisuus, joka sisältää useampia toimintoja (ks. seuraava) Kokonaisuudet luo projektivastaava.
    - Toiminto (Story) – Yksittäinen kohteeseen kehitettävä toiminnallisuus/toiminto. Osa suurempaa kokonaisuutta (ks. edellä). Toiminnot luo kehitystiimin jäsen.
    - Bugi (Bug) – Virhe toiminnallisuudessa. Bugi voi liittyä myös osaksi suurempaa kokonaisuutta. Bugeja voi luoda kuka tahansa projektissa työskentelevä.
      - Alitehtävä (Sub-Task) – Yksittäinen toiminnon toteutumista edellyttävä tehtävä, jonka kehittäjä tekee. Toiminto voi sisältää useita alitehtäviä. Alitehtävät luo kehitystiimin jäsen.

Dia 7

JIRA – Yleistä – Issues

VISION Development
Member of Vision Systems company

Muotoiluesimerkkejä:

Epic:
Otsikko ja yhteenveto (Epic Name ja Summary):
Communication interface
Kuvaus (Description):
General description of the feature.

Bug:
Otsikko (Summary):
General description of the bug.
Kuvaus (Description):
Detailed description of the bug.

Story:
Otsikko (Summary):
Driver module development
Kuvaus (Description):
As ... I want to be able to ...
Definition of done:
1. Definition
n. Definition

Sub-task:
Otsikko (Summary):
General description of the sub-task.
Kuvaus (Description):
-

Dia 8

JIRA – Yleistä – Issues

VISION Development
Member of Vision Systems company

- Toimintoja (Story) luotaessa, niille asetetaan linkki kokonaisuuteen (Epic)

- Lisäksi toiminto (Story) tai bugi (Bug) tulee ilmoittaa tehtäväksi tiimin edustajalle

Dia 9

JIRA – Yleistä – Issues

- Jokainen toiminto (Story) ja bugi (Bug) tulee estimoida (pisteyttää) kompleksisuuden mukaan, ei aikasidonnaisesti. Pisteytyksessä tulee huomioida periodin pituus; toiminto tulee voida suorittaa yhdessä periodissa. Suuremmat toiminnot tulee pilkkoa pienempiin osiin.

  0.5, 1, 2, 3, 5 (, 8, 13)

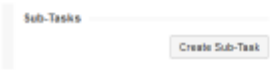
- Toimintojen estimointi on projektivastaavan vastuulla. Projektivastaava voi käyttää estimoinnissa apuna tarvittavaa asiantuntemusta.

- Estimointi tehdään projektin kehitysjononäkymässä (Backlog) estimoitava toiminto aktiivisena

Dia 10

JIRA – Yleistä – Issues

- Kehittäjä luo toiminnoille alitehtävät (Sub-Task) toiminnon asetuksien kautta suunnittelupalaverissa



- Periodissa suoritettavat alitehtävät valitaan periodiin suunnittelupalaverissa ja käynnistetään periodi (Create Sprint → Start Sprint)

Dia 11

JIRA – Yleistä – Workflow

VISION Development
Member of Vision Systems company

- Projektin tehtävien työnkulku suoritetaan tietynlaisen mallin mukaisesti
- Mallissa siniset, keltaiset ja vihreät laatikot kuvaavat tehtävien tiloja ja niihin osoittavat nuolet tilanvaihdoksia (mistä eri tiloista kohdetilaan pystytään siirtymään)
- Oheinen malli on nimeltään *Vision Workflow* ja kyseinen malli otetaan käyttöön jokaiseen projektiin, joka luodaan

Dia 12

JIRA – Yleistä – Workflow

VISION Development
Member of Vision Systems company

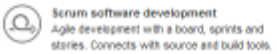
- Kun tehtävä luodaan, se saa tilan *To Do*
- Kun tehtävä otetaan tiimille työn alle, se saa tilan *In Progress*
- Jos tiimissä suoritetaan katselmointia, kehittäjä joka antaa tehtävän katselmoitavaksi, asettaa sen tilaan *To Review*
  - Kehittäjä joka ottaa tehtävän katselmoitavaksi, asettaa sen tilaan *In Review*
  - Kehittäjä joka katselmoinut tehtävän, asettaa sen tilaan *Reviewed*
- Kun tehtävä on tehty ja katselmoitu, muttei testattu, se saa tilan *Testing*
- Kun tehtävä on valmis ja kaikki hyväksyntäkriteriat toteutuvat, se saa tilan *Done*
- Mikäli tehtävän sisältö muuttuu, sen toteutuksen ajankohta siirtyy kesken periodin tai sen toteuttaminen ei muuten ole mahdollista, se saa tilan *Closed* (Suljettu, ei oteta enää työn alle) tai *Postponed* (Lykätty, voidaan ottaa uudestaan työn alle myöhemmin)
  - Mikäli tehtävä tulee uudestaan työn alle, se saa tilan *To Do*

Dia 13

JIRA – Projektien luominen

- Projekti luodaan valitsemalla ylävalikosta Projects → Create Project

- JIRAan voidaan luoda Scrum- ja Kanban-prosessinmukaisia Boardeja
- Tiimeille tulee luoda Scrum-prosessin mukainen projekti ja Board
  - Esim.
    Team Ahma Project
    Team Ahma Board
- Muut projektit (joilla ei ole omaa projektitiimiään) ja niiden Boardit tulee luoda Kanban-prosessin mukaisesti



Dia 14

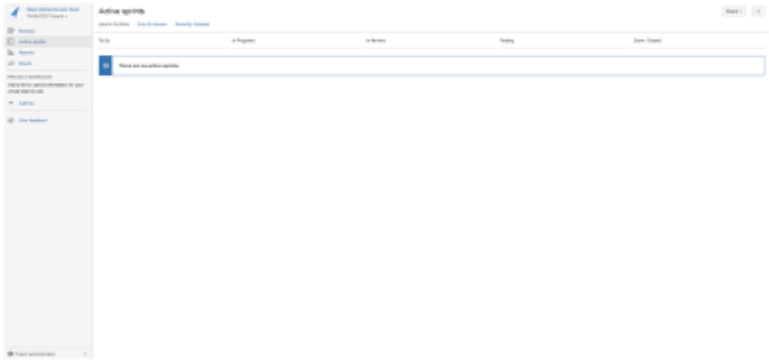
JIRA – Projektien luominen

- Kun projektityyppi on valittu, JIRA esittää projektille oletuksena annettavan työnkulun ja tehtävätyyppirakenteen (nämä voidaan muuttaa myöhemmin). Valitse Select.
- Projektin nimen syntaksi on *[Projektinumero] [Projektinimi]*
- Jokaisella projektilla on yksilöity tunniste. Projektin tunnisteen tulee olla kolme kirjainta, numeroita tunnisteeseen ei sallita.

Dia 15

JIRA – Projektien luominen – Workflow

- Kun projekti on luotu sen tehtävien työnkulku otetaan käyttöön seuraavasti (ks. JIRA – Workflow)

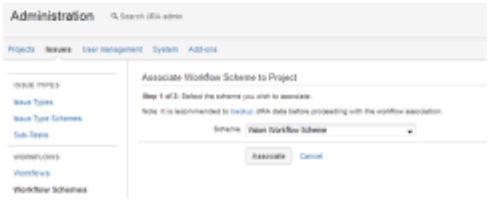


- Valitse vasemmalta alhaalta Project administration
- Valitse Workflows
- Valitse Switch Scheme

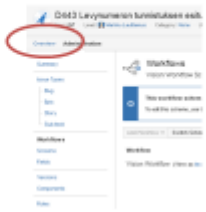
Dia 16

JIRA – Projektien luominen – Workflow

- Valitse alasvetovalikosta Vision Workflow Scheme ja valitse Associate



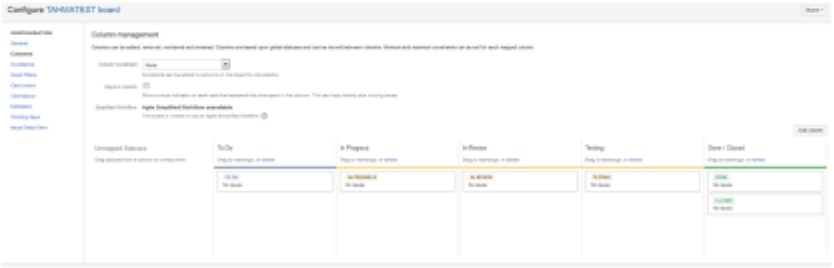
- Valitse vasemmalta ylhäältä Overview



Dia 17

JIRA – Projektien luominen – Workflow

- Valitse oikealta ylhäältä Board → Configure
- Valitse vasemmalta Columns
- Lisää sarakkeita Add Column –painikkeesta (nimeä sarakkeet kuvan mukaisesti)
- Liitä tehtävien tilat sarakkeisiin (Unmapped Statuses)

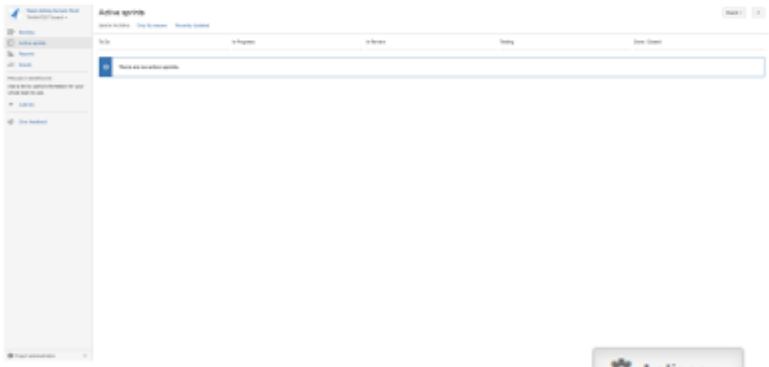


- Valitse vasemmalta ylhäältä Boardisi nimi 

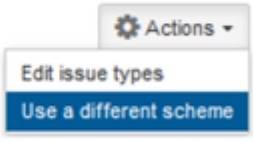
Dia 18

JIRA – Projektien luominen – Issue Types

- Seuraavaksi otetaan käyttöön tehtävien tyypit (ks. JIRA – Issues)



- Valitse vasemmalta alhaalta Project administration
- Valitse Issue Types
- Valitse Actions → Use a different Scheme

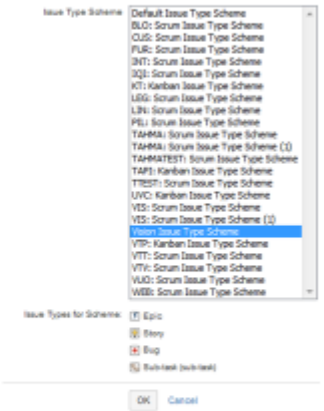


Dia 19

JIRA – Projektien luominen – Issue Types

VISION Development
Member of Vision Systems company

- Valitse listalta Vision Issue Type Scheme ja valitse OK



- Valitse vasemmalta ylhäältä Overview

Dia 20

JIRA – Projektien luominen – Board Filters

VISION Development
Member of Vision Systems company

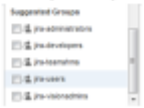
- Seuraavaksi suodatetaan projektin näkymää
- Valitse oikealta ylhäältä Board → Configure

- Valitse General-välilehdeltä Edit Filter Query tai
- Klikkaa Saved Filter –kohtaa ja valitse valmis suodatin listalta
- Seuraavalla slidellä käsitellään suodattimen muokkaamista

Dia 21

JIRA – Projektien luominen – Board Filters

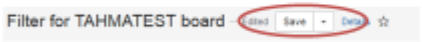
- Valitse ensimmäisestä (Project) alasvetovalikosta, minkä projektien tehtävien haluat Boardillasi näkyvän (esim. tiimillä työn alla olevat projektit tai ainoastaan ko. projekti)
- Valitse toisesta (Type) alasvetovalikosta, minkä tyyppiset tehtävät haluat Boardillasi näkyvän (valitse Story, Sub-Task ja Bug)
- Valitse kolmannesta (Status) alasvetovalikosta, minkä tilan omaavat tehtävät haluat Boardillasi näkyvän (oletuksena kaikki)
- Valitse neljännestä (Assignee) alasvetovalikosta käyttäjät/ryhmät, joille määrätyt tehtävät haluat Boardillasi näkyvän (esim. jos olet tekemässä näkymää ainoastaan tiimillesi, valitse tiimin nimi. Jos teet perusprojektia, suodattimen voi jättää oletukseksi → kaikki)



Dia 22

JIRA – Projektien luominen – Board Filters

- Tallenna suodattimet valitsemalla ylälaidasta Save



- Palaa takaisin (esim. selaimen Back-nappi) Board Configure –ikkunaan ja klikkaa Edit Filter Shares

- Valitse Add Shares –alasvetovalikosta haluamasi käyttäjät/ryhmät, joiden haluat näkevän Boardisi ja klikkaa Add
- Lopuksi hyväksy klikkaamalla Save

Dia 23

JIRA – Projektien luominen – Estimates

- Jotta toimintoja ja alitehtäviä voidaan estimoida, tulee estimointipisteet ottaa käyttöön boardissa
- Mene boardin konfigurointiin Board-valikosta → Configuration
- Valitse Estimation Statistic –alasvetovalikosta Story Points

Dia 24

JIRA – Toimintojen ja alitehtävien luominen

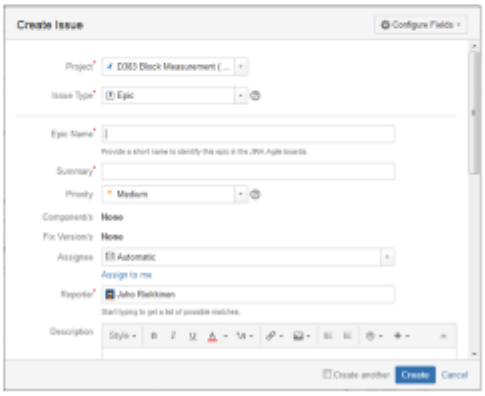
- Kokonaisuudet (Epic), toiminnot (Story) ja bugit (Bug) luodaan Create-painikkeesta
- Create-painiketta klikattaessa JIRA luo tehtävän oletuksena sillä hetkellä aktiivisena olevalle projektille

Dia 25

JIRA – Toimintojen ja alitehtävien luominen

VISION Development
Member of Vision Systems company

- Create Issue –ikkunassa määritellään tehtävän ominaisuudet (punaisella tähdellä merkatut ovat pakollisia)
- Tehtävälle tulee antaa nimi ja yhteenveto ohjeistuksen mukaisesti (Epicissä nimessä ja yhteenvedossa voi olla sama, lyhyt ja kuvaava nimi, ks. JIRA – Yleistä – Issues)

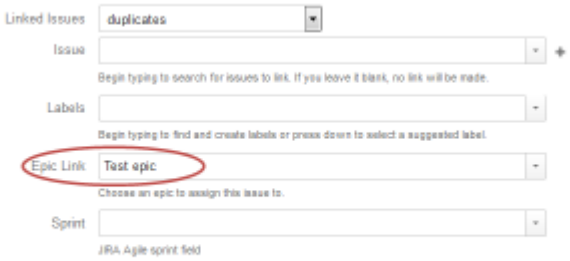


Dia 26

JIRA – Toimintojen ja alitehtävien luominen

VISION Development
Member of Vision Systems company

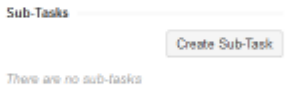
- Mikäli toiminto/bugi on isompaan kokonaisuuteen liittyvä, tulee Epic Link –kohtaan valita Epic, jonka alle toiminto/bugi kuuluu
  - Epic Linkin voi määrittää myös myöhemmin, mutta ensin siis kannattaa tehdä Epic-tyypit, jonka alle määrittää toiminnot



Dia 27

JIRA – Toimintojen ja alitehtävien luominen

- Kun toiminto (Story) on luotu, sille voidaan määritellä alitehtävät
- Alitehtävän määrittelyyn pääset valitsemalla Boardilla toiminnon ja selaamalla oikeanpuoleisessa ikkunassa kohtaan Create Sub-Task



- Kuten toiminnot, alitehtävät kirjataan ohjeistuksen mukaisesti Create Sub-Task –napilla avautuvan dialogin kautta (alitehtävä menee oletuksena valittuna olevan toiminnon alle)

Dia 28

JIRA – FAQ – Move-toiminto

- JIRAssa tehtävän voi siirtää projektista toiseen Move-toiminnolla (esim. tehtävän sivupalkista … → More Actions… → Move)
- Joissain tapauksissa voi tulla seuraavanlainen ilmoitus:

- Workaround:
  - Vaihda Issue type → Bug
  - Tee Move-toiminto uudestaan (Current issue type Bug → New issue type Bug)
  - Kun tehtävä on uudessa projektissa, vaihda Issue type takaisin → Story
- Lisätietoa:

https://confluence.atlassian.com/jirakb/unable-to-move-issue-to-another-issue-type-due-to-the-issue-type-selected-is-invalid-error-227413483.html

Liite 3. Tehtävienhallintaprosessin kehittämisen kysely

TEHTÄVIENHALLINTAPROSESSIN KEHITTÄMISEN KYSELY

Tämän kyselyn tarkoitus on kartoittaa, kuinka tehtävienhallintaprosessin ja -työkalujen kehittämisen projekti on onnistunut ja millaisia kehityskohtia nähdään jatkoa ajatellen.

Projektin tuloksina ovat toteutetut määrittelyt ja ohjeistukset toimintatapoihin, käyttöönotettu tehtävienhallintatyökalu, käyttöönotettu prosessimalli (Team Ahma) ja sen aikana toteutuneet kehitysperiodit.

Kirjoita alle roolisi ja rastita väitteitä mielestäsi parhaiten kuvaavat vaihtoehdot asteikolla:
1 - Täysin eri mieltä ... 5 - Täysin samaa mieltä
Mikäli et osaa vastata johonkin kohtaan, jätä se tyhjäksi. Kommentit ja kehitysehdotukset ovat enemmän kuin tervetulleita!

Rooli (projektivastaava, kehitystiimin edustaja, kehitystiimin jäsen):

1. Tehtävienhallintaprosessin määritelmä ja ohjeistukset

Nro.	Väite	1	2	3	4	5
1	Olen perehtynyt prosessimääritelmään					
2	Olen perehtynyt Jira-ohjeistukseen					
3	Dokumentit ovat selkeitä					
4	Dokumenttien formaatti on tarpeenmukainen					
5	Dokumenteista on apua perehtymisessä tehtävienhallintaprosessiin					
6	Dokumenteista on apua perehtymisessä Jiraan					
7	Dokumentit löytyvät helposti					
8	Dokumentit ovat liian yksityiskohtaisia					
9	Dokumentit ovat liian suppeita					
10	Evaluointiraporteista (Crucible, Confluence) saa hyvän käsityksen työkaluista					
11						
12						
13						
14						
15						

Kommentit ja kehitysehdotukset:

2. Työkalut (Jira, Crucible)

Nro.	Väite	1	2	3	4	5
1	Työkalut helpottavat usean projektin tehtävienhallintaa					
2	Työkalujen avulla projektin tehtävien tilan seuraaminen on vaivatonta					
3	Työkalu auttaa tehtävien visualisointia ja suunnittelua					
4	Työkalu on konfiguroitu tarpeiden mukaisesti					
5	Työkalu on käyttäjäystävällinen					
6	Työkalu auttaa projektien ja kehitysperiodien raportoinnissa					
7	Projektin luominen työkaluun on vaivatonta					
8	Tehtävien luominen ja määrittely työkaluun on vaivatonta					
9	Työkalun avulla voidaan seurata kehitysperioideja tarpeenmukaisesti					

10	Työkalu auttaa projektien aikataulutuksessa ja resursoinnissa					
11	Työkalu auttaa projektikommunikointia ja luo läpinäkyvyyttä					
12						
13						
14						
15						

Kommentit ja kehitysehdotukset:

2. Prosessimallin käyttöönotto ja kehitysperiodit

Nro.	**Väite**	**1**	**2**	**3**	**4**	**5**
1	Olen ollut mukana prosessimallin käyttöönotossa					
2	Tiedän kuinka prosessimallia käytetään					
3	Tiedostin roolini prosessissa käyttöönoton aikana					
4	Pystyin edesauttamaan prosessimallin käyttöönottoa hyvin					
5	Prosessimallin käyttöönotto oli vaivatonta					
6	Prosessimalli ja kehitysperiodit auttoivat työskentelyä käyttöönoton aikana					
7	Prosessimalli paransi projektien tilanteen läpinäkyvyyttä käyttöönoton aikana					
8	Prosessimalli auttoi projektien aikataulusta/suunnittelua ja resursointia käyttöönoton aikana					
9	Prosessimallin toteutus (suunnittelu) oli onnistunut					
10	Prosessimalli rauhoittaa kehitystyötä					
11	Tiimi (ml. projektivastaavat ja tiimin edustaja) olivat sitoutuneet noudattamaan prosessimallia käyttöönoton aikana					
12	Tiimin henkilömäärä oli riittävä prosessimallin toteutukseen käyttöönoton aikana					
13	Prosessimalli on toimiva pienten asiakasprojektien hoitamiseen					
14						
15						

Kommentit ja kehitysehdotukset: