Eetu Vilhunen

KETTERIEN MENETELMIEN HYÖDYNTÄMINEN VIDEOPELIEN JULKAISUN JÄLKEISESSÄ YLLÄPITOPROSESSISSA

JYVÄSKYLÄN YLIOPISTO
TIETOJENKÄSITTELYTIETEIDEN LAITOS
2016

TIIVISTELMÄ

Vilhunen, Eetu
Ketterien menetelmien hyödyntäminen videopelien julkaisun jälkeisessä ylläpitoprosessissa
Jyväskylä: Jyväskylän yliopisto, 2016, 114 s.
Tietojärjestelmätiede, pro gradu
Ohjaaja(t): Leppänen, Mauri

Viime vuosien aikana peliala on kasvanut voimakkaasti ja siitä on kehittynyt viihteen ala, jonka asiakaskunta vaatii entistä hauskempia ja viihdyttävämpiä pelejä. Pelien odotetaan myös hyödyntävän nopeasti kehittyvän teknologian luomia uusia mahdollisuuksia. Näiden haasteiden ohella pelikehitykseen osallistuu monien eri alojen toimijoita, mikä lisää entisestään pelien kehittämisen haastavuutta. Tästä johtuen pelistudion kehitys- ja ylläpitoprosessien tulee olla huippuluokkaa. Tehokkuutta ja luovuutta on ohjelmistokehityksen alueella tavoiteltu ketteriä menetelmillä ja käytänteillä. Ketteriä menetelmiä on käytetty jossain määrin myös pelikehityksessä. Pelin julkaisun jälkeisestä toiminnasta ei ole kovin paljon tutkimustietoa.

Tämän tutkielman tarkoituksena on selvittää, millä tavoin ketteriä menetelmiä voidaan hyödyntää videopelien julkaisun jälkeisessä ylläpito-/ jatkokehittelyprosessissa. Tutkielma koostuu kahdesta osasta, kirjallisuuskatsauksesta ja haastattelututkimuksesta. Kirjallisuuskatsauksella valotetaan pelien, peliteollisuuden, pelikehityksen, ketterien menetelmien ja pelien ylläpidon taustoja. Haastattelututkimus on tehty neljässä suomalaisessa pelistudiossa, jotka poikkeavat toisistaan kooltaan ja pelialustoiltaan.

Haastateltavien keskuudessa pelien ylläpito koettiin enemmän jatkokehityksenä, kuin ylläpitona. Tästä huolimatta jatkokehittelyyn kuuluu oleellisena osana virheiden korjaus sekä erinäisten palvelujen ja pelaaja suhteiden ylläpito, vaikka pääpainona onkin uuden sisällön tuottaminen. Suosituin lähestymistapa ylläpitoon ja jatkokehitykseen on toteuttaa kahta eri kehityslinjaa, joista toinen keskittyy uuden sisällön luomiseen ja toinen ylläpidollisiin tehtäviin. Ketterien menetelmien osalta yritykset eivät käyttäneet mitään yksittäistä menetelmää vaan olivat kokeilun ja kokemuksien kautta räätälöineet omiin tarkoituksiinsa toimivat käytänteet. Yritykset kokivat ketterien menetelmien käytön hyödylliseksi, mutta käyttöön sisältyi kuitenkin haasteita. Saatuja tuloksia voidaan hyödyntää hyödyntää pelistudioissa pelikehityksen parantamiseksi ja jatkotutkimuksen pohjana.
Asiasanat: ketterät menetelmät, peliteollisuus, videopelien ylläpito

ABSTRACT

Vilhunen, Eetu
Usage of Agile Methods in post-launch process of Video games
Jyväskylä: University of Jyväskylä, 2016, 114 s.
Information Systems, Master's Thesis
Supervisor(s): Leppänen, Mauri

In recent years, the game industry has developed into a highly competitive entertainment industry with the market demanding more and more interesting and fun games. In addition, these games should take advantage of rapidly evolving technological opportunities available. These qualitative and technological challenges, together with problems with participating stakeholders from multiple different fields, make game development very challenging. As a result, game development and maintenance prosesses used by game studios should be top-notch. In software engineering, productivity and creativity has been pursuited through agile methods and practices. Also in game development agile methods have, to some extent, been used. There are only a few studies on agile game development. These studies do not, however, concern activities carried out after the first release or in game maintenance.

The purpose of this study is to find out how agile methods can be deployed in game further development or game maintenance. The thesis consists of two main parts, a literature review and an empirical part. The literature review sheds light onto games, gaming industry, game development, agile methods and game maintenance. The empirical part comprises interviews conducted in four game studios differing in terms of size and platform of games.

The study concludes that game maintenance was seen more as further development rather than as maintenance. Despite this, different maintenance activities such as bug fixing, support activities and player feedback collection were an integral part of this process, although the main focus was on the development of new content. The most popular approach to the maintenance and further development of the game was to establish two lines of development: one focused on the creation of new content, and the other concerned with maintenance activities. In regard to agile methods, companies did not use any specific methods, but rather were using their own constructs created through experimentation and experience. The usage of agile methods was, in general seen to be beneficial, although the methods posed certain limitations and challenges. The results of this study can be used to make

improvements in game development and maintenance processes in game studios, and as a basis for further research.
Keywords: Agile methodologies, gaming industry, game maintenance

KUVIOT

TAULUKOT

SISÄLLYS

1. JOHDANTO

Pelialasta on viime vuosien aikana kehittynyt viihteen osa-alue, jonka voidaan sanoa saavuttaneen valtavirran suosion. Pelien kehitykseen käytetään suuria summia, ja näiden pelien tuotanto-prosessi on tyypillisesti hyvin pitkä, jopa useita vuosia. Kasvanut asiakasmäärä on tarkoittanut sitä, että myös erityylisten pelien määrä ja kysyntä on kasvanut voimakkaasti. Samalla laitteisto on kehittynyt, mikä on vaikuttanut paljon pelien grafiikkaan sekä kehittäjien käytössä olevaan teknologiaan ja tehnyt mahdolliseksi yhä vaikuttavampien pelien kehittämisen.

Pelillä tarkoitetaan tässä kaikkia digitaalisia laitteilla pelattavia pelejä. Digitaalisina peleinä voidaan käsittää karkeasti ne pelit, jotka on toteutettu jollekin teknologiselle alustalle, kuten esimerkiksi PC:lle (Salen & Zimmerman, 2003). Digitaalisista peleistä puhutaan myös usein nimityksellä videopeli. Pelit voidaan jakaa useisiin eri genreihin, jotka voivat erota toisistaan hyvin paljon sisällöllisesti (Ye, 2004). Pelien genreihin kuuluu esimerkiksi simulaatiot tai strategiapelit.

Pelikehitystä varten ei ole olemassa mitään yhtä yksittäistä pelikehitysprosessia, mutta yleisellä tasolla pelikehitysprosessin voidaan nähdä koostuvan seuraavista vaiheista: konseptin valmistelu, esituotanto, tuotanto, laadunvarmistus, julkaisu sekä ylläpito (Manninen ym., 2006). Pelikehitys nähdään poikkeavan perinteisestä ohjelmistokehityksestä, sillä pelit ovat viihdetuotteita. Pelien kehitystyö yhdistääkin monen eri alan osaajia yhteen pelikehitysprosessin ajaksi. Pelikehityksessä ilmenee ohjelmistotuotannon tavoin monenlaisia ongelmia (Petrillo ym., 2009; Keith, 2010; Kanode & Haddad, 2009). Esimerkkejä näistä ongelmista ovat ongelmat mittakaavojen määrittelyssä, viivästymiset, ominaisuuksien kasvu, ominaisuuksien karsiminen, dokumentaation puute ja kommunikaatio-ongelmat. Pitkät

kehitystyöjaksot, monialaiset projektitiimit ja uudet teknologiat nostavat pelikehityksen haastavuutta (Petrillo ym., 2009; Keith, 2010; Kanode & Haddad, 2009).

Ketterää pelikehitystä käsitteleviä julkaisuja löytyy vähän. Keith on kirjoittanut aiheeseen liittyen kirjan Agile Game Development with Scrum (Keith, 2010). Musil ym. (2010a) ovat laatineet yhteistyössä itävaltalaisten pelistudioiden kanssa esityksen ketterästä pelikehitysprosessista. Nämä esitykset ovat kuitenkin vain yksittäisiä näkemyksiä ketterästä pelikehityksestä. Lisäksi Musil ym. (2010b) ovat selvittäneet ketteryyden asemaa itävaltalaisessa peliteollisuudessa tutkimuksessaan, jossa todetaan valtaosan pelistudioista hyödyntävän ketteriä menetelmiä. Goboy ja Barbosa (2010) puolestaan ovat tutkineet juuri pelikehityksen tarpeisiin ja ongelmiin vastaavia ketteriä menetelmiä ja luoneet näiden pohjalta esityksen Game-Scrum nimisestä lähestymistavasta pelikehitykseen. Flood (2003) on tehnyt kuvauksen GUP (game unified process) nimisestä lähestymistavasta pelikehitykseen, joka yhdistelee XP:n ja RUP:n (Rational Unified Process) käytänteitä. Gibson (2007) on puolestaan tutkinut pelikehityksen ongelmia ja tehnyt kokeen, kuinka Scrum pystyy vastaamaan niihin.

Pelien ylläpitoa koskevia tutkimuksia on hyvin vähän (vrt. Ampatzogloun ja Stamelosin (2010) kirjallisuuskatsaus). Pelien ketterästä ylläpidosta ei ole löytynyt yhtään julkaisua. Tämä herättää kysymyksen, minkä luonteista pelien ylläpito itse asiassa on ja mikä sen suhde on itse pelikehitykseen. Erottuuko ylläpito toiminnallisesti ja organisationaalisesti muusta kehitystoiminnasta vai onko kysymyksessä pelin jatkokehitys? Ja miten tähän voisivat kytkeytyä ketterien menetelmät ja käytänteet?

Tämän tutkielman tarkoituksena on selvittää, millä tavoin ketteriä menetelmiä voidaan hyödyntää pelien julkaisun jälkeisessä ylläpito-/jatkokehittelyprosessissa. Tutkimuksen tavoite voidaan esittää tutkimusongelmina seuraavasti:

Millä tavoin ketteriä menetelmiä voidaan hyödyntää videopelien julkaisun jälkeisessä ylläpitoprosessissa?

Tutkimusongelma voidaan jakaa seuraaviksi tutkimuskysymyksiksi:

1. Millaista on videopelien ylläpito/jatkokehittely?
2. Millaisia ongelmia esiintyy videopelien ylläpidossa/jatkokehittelyssä?
3. Millaisia hyötyjä ja ongelmia ketterien menetelmien ja käytänteiden käytöstä videopelien ylläpidossa/jatkokehittelyssä on koettu?

Tutkimus koostuu kahdesta osuudesta, käsitteellis-teoreettisesta ja empiirisestä osuudesta. Käsitteellis-teoreettisessa osuudessa pyritään luomaan aiheelle teoriapohja. Empiirinen osuus puolestaan toteutetaan haastattelututkimuksena, joka on suunnattu suomalaisille pelistudioille ja sillä pyritään saamaan vastauksia tutkimusongelmiin. Haastattelun kohteeksi on valittu neljän pelistudion henkilöitä. Tiettävästi tämänlaatuista tutkimusta ei ole aiemmin tehty.

Tämä tutkielma jakautuu kahdeksaan lukuun. Luvussa 2 tarkastellaan pelejä, peliteollisuutta ja pelikehitystä. Luvussa 3 kuvataan ketterää lähestymistapaa ja kolmea ketterää menetelmää, Scrumia, XP:tä ja Kanbania. Lisäksi kerrotaan ohjelmiston ylläpidosta yleisesti ja ketterästä ylläpidosta erityisesti. Luvussa 4 käsitellään ketterää pelikehitystä aiheesta julkaistun kirjallisuuden pohjalta. Luvussa 5 kerrotaan empiirisen osuuden tutkimusmenetelmän valinnasta, haastateltavien valinnasta sekä kuvataan tutkimusasetelma ja haastattelurunko. Luvussa 6 raportoidaan haastattelututkimuksen tulokset. Luvussa 7 tiivistetään tulokset vastauksiksi tutkimuskysymyksiin, verrataan tuloksia aiempien tutkimusten tuloksiin ja tarkastellaan tutkimuksen luotettavuutta. Luvussa 8 esitetään yhteenveto ja jatkotutkimusaiheita

2. PELIT JA PELIKEHITYS

Tässä luvussa keskitytään taustoittamaan pelejä, peliteollisuutta, pelikehitystä ja pelien ylläpitoa. Liikkeelle lähdetään käsittelemällä pelien suosion kasvua ja peliteollisuuden kehittymistä sekä peleihin liittyviä toimijoita. Tämän jälkeen esitetään pelin määritelmä ja perehdytään pelien luokitteluun. Kolmanneksi käsitellään pelikehitystä, sen vaiheita sekä siihen liittyviä haasteita. Lopuksi tarkastellaan vielä pelien ylläpitoa ja luodaan yleiskuva sen sisällöstä ja tavoitteista.

2.1. Pelaamisen yleistyminen ja peliteollisuuden kasvu

Digitaalisten pelien myynti on kasvanut huomattavasti koko 2000-luvun ajan. Suomessa pelimyynnin kehittymistä seuraa Figma Ry, mikä on suomalainen peliohjelmisto- ja multimediajärjestö. Figma Ry julkaisee vuosittain tilastokoosteen Suomen pelimarkkinoiden myyntimäärästä sekä sen rahallisesta arvosta. Figman tilastojen (Figma Ry, 2009) mukaan 2000-luvun alussa Suomessa Figman jäsenet myivät 1 538 000 peliä ja kokonaismyynnin rahallinen määrä oli 54 miljoonaa euroa. Vuoteen 2009 mennessä Figman jäsenten myynti oli kasvanut 2 088 000 peliin ja myynnin rahallinen kokonaisarvo oli kasvanut 75,4 miljoonaan euroon. Vuonna 2009 Figma julkaisi myös tilaston kaikista Suomessa myydyistä peleistä. Tuona vuonna Suomessa oli myyty kaikkiaan 2 783 000 peliä 10 110 000 euron edestä (Figma Ry, 2009).

Yhdysvalloissa kehitys on ollut Peltoniemen (2009) keräämien lähteiden mukaan hyvin samankaltaista. 2000-luvun alussa Yhdysvaltojen pelimyynnin arvo oli noin 5,7 miljoonaa dollaria. Vuoteen 2007 mennessä pelimyynnin arvo

Yhdysvalloissa oli kasvanut noin 9,5 miljoonaan dollariin. Kasvua on nähty myös massiivisten verkossa moninpelattavien eli lyhyemmin MMO-pelien niin sanottujen tilaajien määrän kasvussa. Peltoniemen (2009) lähteistä ilmenee, että vuonna 2000 MMO-pelien tilauksia oli noin 1 miljoona. Vuoteen 2008 mennessä tällaisten tilauksien määrä oli kasvanut noin 16 miljoonaan. Näiden tilastojen perusteella voimme päätellä, että pelaaminen on kasvattanut suosiotaan huomattavasti ja kasvanee vielä lisää. Suomen tasolla suosion kasvusta viestii myös muun muassa se, että YLE on alkanut tuoda ohjelmistoonsa niin sanottua elektronista urheilua, mikä käytännössä on digitaalisten pelien kilpapelaamista.

Syitä suosion kasvuun lienee useita. Suomen Pelinkehittäjät Ry:n ja Neogamesin tuottamassa Suomen pelialanstrategia -julkaisussa (Kuittinen ym., 2010) erääksi syyksi nähdään teknologian kehittyminen pelialalla. Markkinoille on tullut valtavasti uusia pelilaitteita ja varsinkin jakelukanavat ovat kehittyneet suuresti, mikä on luultavasti vaikuttanut kasvuun. Myös pelisisältöjen kehityksellä uskottiin olevan vaikutusta, mikä on mahdollistanut sisällön tarjoamisen uusille kohderyhmille. Lisäksi verkkopelaaminen ja verkkojakelun voimakas kasvu nähtiin yhdeksi syyksi. Tässä suhteessa oletettavaa on, että ihmisten vahva tarve sosiaalisuuteen on merkittävä tekijä. Havaintoja tähän suuntaan on nähty myös muissa tutkimuksissa (Jansz ym., 2005; Vorderer ym., 2003).

Teknologisten alustojen, julkaisukanavien ja pelisisältöjen kehitys on mahdollistanut sen, että suuremmalla yleisöllä on pääsy heille merkittäviin peleihin. Pelaaminen näkyy jo televisiossa, kuten voimme huomata YLEn elektronisen urheilun lähetyksistä. Pelaamisen voidaan nähdä saavuttaneen valtavirran suosion (Peltoniemi, 2009) ja olettaa voidaan, että pelaamisen suosio tulee kasvamaan entisestään.

2.2. Pelin määritelmä

Pelaamista ja pelejä on historiallisesti tutkittu useista eri lähtökohdista (Salen & Zimmerman, 2003). Tästä johtuen kirjallisuudessa löytyykin monenlaisia määritelmiä pelille. Usein määritelmiin on pyritty sisällyttämään niin pelit, leikkiminen ja myös myöhemmin digitaaliset pelit (Juul, 2003).

Salen ja Zimmerman (2003) ovat esittäneet kirjassaan Rules of Play: Game Design Fundamentals oman määritelmänsä pelille. Heidän määritelmänsä pohjautuu kahdeksaan kirjallisuudessa aiemmin esitettyyn määritelmään. Määritelmään on päädytty poimimalla aiemmista määritelmistä yhtäläisyydet

ja poistamalla heidän mielestään tarpeettomat osat. Heidän määritelmänsä on seuraavanlainen :

”Peli on järjestelmä, jossa pelaajat osallistuvat teennäiseen konfliktiin, joka on määritelty säännöin ja jolla on laskettavissa oleva lopputulos” (Salen & Zimmerman, 2003, s.80)

Määritelmään kuuluu kuusi eri käsitettä, jotka ovat Järjestelmä, Pelaajat, Keinotekoisuus, Konflikti, Säännöt sekä Laskettavissa oleva lopputulos. Salen ja Zimmerman (2003) ovat määritelleet käsitteet kirjassaan seuraavasti:

Järjestelmä on useista toisiinsa yhteydessä olevista osista koostuva monimutkainen kokonaisuus. Kaikki järjestelmät sisältävät objekteja, attribuutteja, sisäisiä suhteita sekä ympäristön. Tapa, jolla nämä osat tunnistetaan yksittäisissä peleissä, riippuu siitä, kuinka peli itsessään kehystetään järjestelmänä. Pelien kontekstissa järjestelmä voidaan ymmärtää joko formaaliksi, empiiriseksi tai kulttuurilliseksi järjestelmäksi. Järjestelmä voi olla joko avoin tai suljettu. Formaalit järjestelmät ovat suljettuja, empiiriset järjestelmät joko suljettuja tai avoimia ja kulttuurilliset järjestelmät ovat avoimia järjestelmiä.

Pelaajat ovat peliin osallistuvia henkilöitä, jotka ovat kanssakäymisessä järjestelmän kanssa kokeakseen pelin pelaamisen. Pelaajia voi olla yhtäaikaisesti yksi tai useampi.

Keinotekoisuus tarkoittaa sitä, että peli pitää yllä rajoja reaalimaailman ja pelimaailman välillä, niin ajallisesti kuin myös tilallisesti. Vaikka useat pelit esiintyvät reaalimaailmassa, keinotekoisuus on silti yksi pelejä määrittelevistä piirteistä.

Konflikti voi peleissä ilmetä usealla tavalla. Kaikki pelit ovat jossain määrin eri voimien välistä kisailua. Tämä kisailu voi ottaa useita eri muotoja yhteistyöstä kamppailuun. Lisäksi konfliktit voivat olla yhden pelaajan konfliktia pelijärjestelmää vastaan tai monen pelaajan sosiaalista konfliktia. Konflikti on erittäin keskeisessä osassa pelejä.

Säännöt ovat myös oleellinen osa peliä. Säännöt määrittävät rakenteen pelille, mitä kautta itse pelaaminen ilmenee. Sääntöjen tarkoitus on kertoa pelaajalle, mitä he voivat tehdä ja mitä he eivät voi tehdä.

Laskettavissa oleva lopputulos tarkoittaa, että peleissä on laskettavissa oleva maali tai lopputulos. Pelin lopussa pelaaja on joko voittanut, hävinnyt tai mahdollisesti saanut jonkinlaisen numeerisen pisteytyksen. Laskettavissa oleva lopputulos on yleensä se tekijä, mikä erottaa pelit muista vähemmän formaaleista pelaamiseen liittyvistä aktiviteeteistä.

Määritelmään kuitenkin liittyy joitain rajatapauksia, jotka Salen ja Zimmerman (2003) kuitenkin käsittävät peleiksi. Näihin rajatapauksiin

kuuluvat puzzle- ja roolipelit sekä lelunomaiset pelit kuten SimCity. Puzzle-pelien kriittisin ero on se, että niillä on ennalta määrätty lopputulos, jonka pelaaja yrittää niin sanotusti avata (Salen & Zimmerman, 2003). Roolipelit taas eroavat perinteisistä peleistä siten, että niissä ei ole yhtä mahdollista lopputulosta, vaan käytännössä pelaaja itse asettaa itselleen tavoittelemansa lopputulokset (Salen & Zimmerman, 2003). SimCity-tyyliset pelit taas eivät varsinaisesti sisällä mitään selkeää päämäärää, joten ne ovat tässä suhteessa lelunomaisia (Salen & Zimmerman). Tärkeää näissä tapauksissa on kuitenkin huomata se, että niiden sisältyminen Salenin ja Zimmermanin (2003) määritelmään riippuu siitä, mistä näkökulmasta näiden alalajien pelejä tarkastellaan (Salen & Zimmerman, 2003).

Tässä tutkielmassa pelit ymmärretään Salenin ja Zimmermanin (2003) määritelmän mukaisesti sillä erotuksella, että myös ääritapaukset katsotaan peleiksi. Ääritapaukset ovat olleet oleellinen osa pelikehitys- ja pelaamiskulttuuria (Salen & Zimmerman, 2003), joten niiden poissulkeminen olisi lyhytnäköistä.

Digitaaliset tai elektroniset pelit voivat esiintyä useissa erilaisissa muodoissa ja useilla eri alustoilla (Salen & Zimmerman, 2003). Näihin peleihin voidaan laskea pelit, jotka on julkaistu esimerkiksi PC:lle, Playstation-konsolille, Nintendo 3DS-käsikonsolille, mobiililaitteelle tai arcade-kabinetille. Digitaaliset pelit, siinä missä muutkin pelit, ovat Salenin ja Zimmermanin (2003) mukaan järjestelmiä. Digitaalisilla peleillä on kuitenkin piirteitä, jotka erottavat ne ei-digitaalisista peleistä. Salen ja Zimmerman (2003) esittelevät neljä piirrettä, jotka erottavat digitaaliset pelit ei-digitaalisista peleistä.

Ensimmäinen piirre heidän mukaansa on välitön mutta rajoitettu interaktiivisuus. Käytännössä tämä tarkoittaa sitä, että digitaaliset pelit reagoivat välittömästi pelaajan antamaan syötteeseen ja näin ollen tarjoavat pelaajalle reaaliaikaista pelattavuutta, joka reagoi dynaamisesti pelaajan valintoihin (Salen & Zimmerman, 2003). Rajoittuneisuudella tässä yhteydessä tarkoitetaan sitä, että vaikka digitaalinen teknologia mahdollistaakin välittömän palautteen saamisen, rajoittuu interaktiivisuus kuitenkin ainoastaan esimerkiksi hiirellä sekä näppäimistöllä annettuun syötteeseen ja näytölle ilmestyneeseen ja mahdollisesti kaiuttimista kuuluneeseen palautteeseen.

Toisena piirteenä digitaalisissa peleissä on Salenin ja Zimmermanin (2003) mukaan informaation manipulointi. Digitaaliset pelit sisältävät yleisesti paljon dataa ja hyödyntävät sitä jatkuvasti. Digitaalisissa peleissä on paljon tekstiä, kuvia, videoita, audiota, animaatioita, 3D-sisältöä ja paljon muuta dataa. Näiden lisäksi voidaan myös nähdä, että digitaaliset pelit manipuloivat esimerkiksi pelin omaa sisäistä logiikkaa, pelaajainteraktiota sekä muistia.

Kaikki nämä voidaan myös nähdä eräänlaisena informaation manipulointina. Digitaaliset pelit kykenevät myös piilottamaan informaatiota erittäin hyvin. Useissa reaaliaikaisissa strategiapeleissä, kuten Command and Conquer tai Age of Empires, hyödynnetään niin sanottua ”sodan sumu”-mekaniikkaa, mikä piilottaa alueet, joilla pelaajalla ei ole yksiköitä.

Kolmanneksi piirteeksi Salen ja Zimmerman (2003) näkevät monimutkaisten järjestelmien automatisoinnin. Tämä tarkoittaa sitä, että digitaaliset pelit automatisoivat useita eri prosesseja. Ei-digitaalisissa peleissä pelaajien täytyy itse viedä peliä eteenpäin, joko liikuttamalla pelin palasia tai toimimalla annetun ohjeistuksen mukaisesti. Digitaaliset pelit taas voivat automatisoida nämä prosessit ja liikuttaa peliä eteenpäin ilman pelaajan antamaa syötettä. Muita hyviä esimerkkejä automatisoinnista on reaaliaikaisten strategiapelien tietokonevastustajien tekoäly tai eri pelimoottorien, kuten Unrealin valotehosteiden mallintaminen.

Viimeisenä erottavana piirteenä on Salenin ja Zimmermanin (2003) mukaan kommunikointi tietoliikenneyhteyksiä hyödyntäen. Vaikka kaikki digitaaliset pelit eivät hyödynnä tietoliikenneyhteyksiä, on niiden hyödyntäminen kuitenkin oleellinen osa useita digitaalisia pelejä. Tämä kommunikointi voi ilmetä useilla tavoilla, kuten esimerkiksi pelin sisäisinä pikaviesteinä, konsolien välisenä viestimisenä moninpelatessa tai jossain mielessä jopa pelkästään pelaamisena, jos ajattelemme pelaamisen eräänlaisena kommunikaation muotona.

Salenin ja Zimmermanin (2003) mukaan on hyvä huomata, että nämä piirteet täyttyvät myös jossain määrin ei-digitaalisissa peleissä. Merkittävänä tekijänä kuitenkin on se, että yleisesti ottaen edellä mainitut piirteet ilmenevät erittäin vahvasti digitaalisissa peleissä, joten ne voidaan laskea digitaalisten pelien erikoispiirteiksi verrattuna ei-digitaalisiin peleihin. Digitaalisista peleistä käytetään myös usein nimitystä videopeli. Jatkossa kun tässä tutkimuksessa puhutaan videopeleistä, viitataan niihin termillä peli.

2.3. Pelien genret

Genrejä on usein käytetty luokitteluun ja analyysiin elokuva- ja musiikkiteollisuudessa. Elokuvateollisuudessa genret perustuvat suureksi osin elokuvien sisältöön kuten aiheisiin tai teemoihin sekä niiden muotoon eli tyyliin ja esitystapaan (Ye, 2004). Elokuvien yhteydessä hyvinä esimerkkeinä genreistä ovat komedia, toiminta tai seikkailu. Musiikkiteollisuuden yhteydessä

Trance, Pop tai Rock ovat hyviä esimerkkejä genreistä. Genren voidaan siis nähdä tarkoittavan tyylisuuntaa. Jokaisella näistä tyylisuunnista on tietyt sisällölliset ominaispiirteet, jotka esiintyvät kaikissa kyseisen tyylisuunnan edustajissa. On myös hyvä huomata, että esimerkiksi elokuvat voivat kuulua useaan eri genreen.

Luonnollisesti myös pelit voidaan luokitella omiin genreihin. Tutkimuksellisessa kontekstissa voidaan nähdä, että pelien genremäärittelyn suhteen on olemassa kaksi erilaista lähestymistapaa. Ensimmäinen on kerrontaa, tyyliä sekä ulkoasua painottava lähestymistapa, jossa lähestytään pelejä samoilla käsitteillä kuin musiikkia tai elokuvia. Toinen lähestymistapa painottaa pelien vuorovaikutuksellisuutta (Apperley, 2006).

Yen (2004) mukaan pelien genret ovat muotoutuneet luonnollisesti parin viimeisen vuosikymmenen aikana pelaajien ja pelin tekijöiden välillä. Hän toteaa myös, että suurin osa näistä genremäärittelyistä perustuu pelaajan ja pelin väliseen vuorovaikutukseen sekä pelikokemukseen. Apperley (2006) on kuitenkin eri mieltä tässä suhteessa, ja hänen mielestään pelien genremäärittely pohjautuu edelleen kuvaukseen eikä niinkään vuorovaikutukseen.

On vaikea sanoa, mitä lähestymistapaa kuluttajien keskuudessa yleisesti hyväksytyt ja käytetyt peligenret edustavat. On myös huomattava, että niin kuin elokuvien yhteydessä, myös pelit voidaan määritellä kuuluvaksi useampaan eri genreen, joten pelien genremäärittelyt eivät ole ehdottomia. Nykyinen yleisesti kuluttajien sekä kehittäjien käytössä oleva luokittelu voidaan kuitenkin nähdä riittävänä, sillä molemmilla osapuolilla vaikuttaa olevan hyvin samankaltainen näkemys siitä, millainen tyypillinen genren edustaja on.

Yleisesti kuluttajien ja kehittäjien käytössä olevat peligenret vaikuttavat olevan näiden kahden ryhmän välistä hiljaista tietoa. Internet tarjoaa useita listauksia siitä, mitä peligenrejä löytyy. Tässä yhteydessä esitellään ne genret, jotka koetaan keskeisimmiksi ja jotka perustuvat laajalti pelialalla hyväksyttyihin genreihin. Ne löytyvät muun muassa ESA:n eli Enterntainment Software Associationin vuonna 2014 julkaisemassa ”Essential facts about the computer and video game industry” -julkaisussa (ESA, 2014). Seuraavaksi esitellään tässä julkaisussa esiteltyjä, keskeisiksi koettuja peligenrejä. Kuvaukset pohjautuvat Wikipedia-artikkeleihin sekä suomalaiseen pelitieto.net-sivustoon.

Räiskintäpelit ovat hyvin usein kolmiulotteiseen maailmaan sijoittuvia pelejä, joissa pelaajahahmon tehtävänä on taistella vihollisia vastaan erilaisilla aseilla (Mäyrä ym., 2009; Wikipedia, 2015). Taistelemisen lisäksi pelaajalla on muita vaihtelevia tavoitteita, jotka hänen täytyy mahdollisesti saavuttaa. Nykyisin tärkeimpinä räiskintäpelien alagenrejen edustajina voidaan pitää

ensimmäisen ja kolmannen persoonan räiskintäpelejä Näiden kahden keskeisimpänä erona on se, että kuvakulma vaihdetaan ensimmäisestä persoonasta kolmanteen persoonaan, mikä käytännössä tarkoittaa kuvakulman vetämistä pelihahmon taakse, jolloin saavutetaan laajempi näkymä sekä mahdollistetaan pelaajahahmon monipuolisempi toiminta.

Ensimmäisen ja kolmannen persoonan räiskintäpelien lisäksi löytyy paljon muitakin alalajeja, kuten Shoot 'em up tai shmup-pelit, kuten Gradius tai G-Darius, valopistoolipelit, joissa ruutuun tähdätään erillisellä aseohjaimella ja pelaajaa kuljetetaan eteenpäin ikään kuin raiteilla, sekä taktiset räiskintäpelit, joiden lähestymistapa räiskintää kohtaan on enemmän taktinen, jopa jossain määrin realistinen ja pelaaja yleensä asetetaan osaksi sotilasryhmää joko jäsenenä tai komentajana (Wikipedia, 2015). Näistä edellä mainituista peleistä on niin yhden kuin myös useamman pelaajan toteutuksia.

Tietokoneroolipelit voidaan nähdä kehittyneiksi perinteisemmistä kynää ja paperia hyödyntävistä roolipeleistä kuten Dungeons & Dragons (Mäyrä ym,. 2009; Wikipedia, 2015). Hyvin usein tietokoneroolipelit alkavat siitä, että pelaaja luo itselleen niin sanotun avatarin. Yleisesti avatarilla on useita erilaisia ominaisuuksia, kykyjä sekä piirteitä, joihin pelaaja kykenee vaikuttamaan luontitilanteessa. Pelin edetessä pelaaja kehittää ja muuttaa hahmoaan valintojensa mukaisesti. Tietokoneroolipelit sijoittuvat yleensä johonkin kuvitteelliseen maailmaan, jonka tapahtumat vaikuttavat pelaajahahmoon. Pelaajahahmo pystyy vaikuttamaan näihin tapahtumiin ja tätä kautta myös itse pelin juonen kulkuun(Mäyrä ym., 2009).

Tietokoneroolipeleissä on havaittavissa kulttuurillisia eroja, ja sitä kautta on syntynyt kaksi erillistä suuntausta (Wikipedia, 2015). Länsimaisten roolipelien peruspiirteitä ovat oman avataren luonti ja seikkailu maailmassa seuraten epälineaarista juonta. Itämaisissa roolipeleissä taas pelaajahahmo on ennalta määrätty ja pelaaja asettuu tämän ennalta määrätyn hahmon ”saappaisiin” ja seuraa lineaarista juonipolkua. Molemmissa suuntauksissa pelaajahahmon ympärille kerätään seurua, jonka jäsenillä on erilaisia, mahdollisesti toisiaan täydentäviä kykyjä.

Yhtenä merkittävänä alalajina tietokoneroolipeleille voidaan nähdä massiiviset usean pelaajan verkkoroolipelit eli niin sanotut MMORPG-pelit, joihin kuuluvat muun muassa World of Warcraft. Aiemmin tämä kyseinen genre oli olemassa teknologisesti vaatimattomammin toteutettuina tekstipohjaisina MUD-peleinä. Eräänä merkittävänä piirteenä MMORPG-alalajin peleillä on sosiaalisuus (Mäyrä ym., 2009).

Seikkailupeleillä tarkoitetaan yleisesti pelityyppiä, joka keskittyy erityisesti tarinankerrontaan sekä ongelmanratkaisuun. Pelaajan tehtävänä on ratkaista

ongelma joko puhumalla tai esineitä hyödyksi käyttämällä ja ratkaisun löydyttyä pelaajahahmo pääsee tarinassa eteenpäin. Yleensä seikkailupelien pelitilanteet ovat erittäin verkkaisia ja pelaaja voi näin ollen pohdiskella ratkaisujaan kiireettömästi. 2000-luvulle tultaessa käsitys seikkailupelistä on jokseenkin muuttunut. Seikkailupeleihin on tuotu hyvin paljon toiminnallisia elementtejä ja valtavirrassa onkin siirrytty niin sanottuihin toimintaseikkailuihin (Mäyrä ym., 2009). Tässä suhteessa on hieman sekavuutta, sillä toimintaseikkailuita nimitetään joskus virheellisesti seikkailupeleiksi, vaikka seikkailupelien keskiössä on ongelmanratkaisu, kun taas toimintaseikkailuissa taistelu on huomattavasti isommassa roolissa. Seikkailupelien elementit näkyvät paljon myös useissa muissa peligenreissä. Esimerkiksi roolipeleissä ja räiskintäpeleissä löytyy nykyisin paljon samoja piirteitä kuin seikkailupeleistä (Mäyrä ym., 2009).

Strategiapelit perustuvat yleisesti kahden tai useamman osapuolen väliseen kilpailuun selviytymisestä tai jonkin voittotavoitteen saavuttamisesta. Keskiössä strategiapeleissä on strateginen ajattelu sekä erilaisten resurssien tehokas hyödyntäminen. Strategiapelit voidaan nähdä johdannaisina vanhoista klassisista lautapeleistä kuten shakista tai mahjongista (Mäyrä ym., 2009). Strategiapeleissä ilmenee kaksi lähestymistapaa, vuoropohjainen lähestymistapa sekä reaaliaikainen lähestymistapa. Erot lähestymistapojen välillä voivat olla hyvinkin suuria. Usein strategiapelit, erityisesti reaaliaikaiset strategiapelit, saattavat sisältää piirteitä myös seikkailu- ja roolipeleistä (Mäyrä ym., 2009). Strategiapelien alalajiksi voidaan myös erottaa niin sanotut taktiikkapelit (Mäyrä ym., 2009). Tämän alalajin pelit eroavat strategiapeleistä pääosin siten, että niiden mittakaava on huomattavasti pienempi verrattuna strategiapeleihin. Käytännössä pelaajalle annetaan mahdollisuus ohjata yksilöitä, ryhmiä tai komppaniaa, eikä niinkään useita joukkueita, divisioonia tai armeijoita.

Simulaatio voidaan nähdä hyvin monimuotoisena genrenä. Siihen voidaan laskea kuuluviksi erilaiset urheilu-, ajoneuvo-, rakentelu-, elämä-, jumal- ja valtakuntasimulaatiot. Yhdistävänä tekijänä simulaatiopeleissä on se, että niissä pyritään jäljittelemään jotain reaalimaailman tapahtumaa, esinettä tai ilmiötä. Simulaatiopeleillä ei varsinaisesti ole ulkoasun, tarinan tai juonen osalta yhdistäviä tekijöitä, joten genreen voidaan laskea kuuluviksi niin kaupunkien rakenteluun keskittyvät pelit, elämäsimulaattorit, autoilupelit sekä esimerkiksi lentosimulaattorit (Wikipedia, 2015).

Edellä mainittujen genrejen lisäksi on myös paljon muita peligenrejä kuten oppimis-, ajanviete-, seura-, kuntoilu- ja musiikkipelit. Näiden lisäksi löytyy

myös paljon muita genrejä, mutta koska niiden kirjo on niin laaja, ei niitä lähdetä tässä luettelemaan tai avaamaan.

Genrejen osalta on vielä tärkeää huomata se, että pelaajilla on niihin sisältyviä odotuksia, jotka pelinkehittäjän tulee ottaa huomioon. Tämä voidaan nähdä taakkana kehittäjille, koska tämä rajoittaa kehittäjien luovuutta. Tärkeää on, että kehittäjät esittelevät uusia ideoita siten, etteivät ne riko liikaa pelaajan odotuksia tai käsityksiä tietystä genrestä (Ye, 2004). Lisäksi jatkuva teknologian kehitys vaikuttaa paljon peligenreihin. Teknologian kehittyessä jotkin genret saattavat vanhentua samalla kun uusia genrejä saattaa syntyä uusien laitteiden tai interaktiivisten tekniikoiden kehittyessä (Ye, 2004).

2.4. Pelialan toimijat

Pelialalta voidaan tunnistaa kolme keskeistä toimijaa: pelikehittäjä, julkaisija ja kuluttaja. *Pelikehittäjä* on se osapuoli, joka on vastuussa pelien suunnittelusta ja toteutuksesta. Pelikehittäjää voidaan joskus kutsua myös pelistudioksi. Pelikehittäjän päämääränä on kehittää resurssien ja budjetin rajoissa mahdollisimman korkealaatuinen peli. *Julkaisija* voi osittain toimia rahoituksen lähteenä peliprojekteille. Tarkemmin julkaisijan rooli on määritelty julkaisijan ja pelikehittäjän välillä tehdyssä julkaisusopimuksessa. Pääasiallisesti julkaisijan tehtäviin kuuluu pelin markkinointi, painatus sekä jakelu. Voidaan ajatella, että julkaisijan pääasiallinen tavoite on tuottaa hyvin myyvä peli ja samalla minimoida omat riskit pitämällä kulut mahdollisimman pieninä. *Kuluttaja* on se taho, pelaaja tai muu asiakas, joka ostaa pelin tai mahdollisen pelipalvelun. Kuluttaja on harvoin suoraan yhteydessä kehittäjiin, sillä julkaisija on yleensä se taho, joka hoitaa liiketoimintaan liittyviä toimintoja (Manninen ym., 2006). Pelikehittäjien ja julkaisijoiden päämäärät ovat yleisesti ottaen samat, eli tarkoituksena on tuottaa mahdollisimman hyvä peli. Näiden osapuolien välillä voi kuitenkin olla ristiriitoja rahoituksen ja aikataulutuksen suhteen.

Pelialalla on tunnistettavissa Mannisen ym. (2006) mukaan myös muita toimijoita. Pelialaa lähemmin tarkasteltaessa huomataan, että alalla vaikuttavat myös *alusta,-* tai *pelimoottorikehittäjät*, *rahoittajat*, *jakelijat* sekä *jälleenmyyjät*.

Alustakehittäjät voidaan jakaa pelialustojen, kuten pelikonsolien, kehittäjiin, ja pelimoottorien kehittäjiin. Tämän sidosryhmän tuotosten tarkoituksena on helpottaa pelikehittäjien työtä, luomalla teknologiakeskeiset ratkaisut, jotta pelinkehittäjät itse voivat keskittyä varsinaisen pelisisällön luomiseen (Manninen ym., 2006).

Rahoittaja on mikä tahansa pelikehitykseen rahaa sijoittava kolmas osapuoli. Mahdollisia kolmansia osapuolia ovat muun muassa riskipääomasijoittajat, yleishyödylliset organisaatiot tai vaikkapa yksittäiset sijoittajat (Manninen ym., 2006).

Jakelijoiden tai *toimittajien* tehtävänä on toimittaa peli kuluttajille tai mahdollisille jälleenmyyjille. Jakelijat voivat joissain tapauksissa osallistua kehityskustannuksiin, mutta yleensä jakelijat ovat projektissa mukana julkaisijan kanssa tehdyn sopimuksen mukaisesti (Manninen ym., 2006).

Jälleenmyyjät ovat vastuussa pelin myymisestä kuluttajille. Yleisesti jälleenmyyjät ostavat pelit jakelijoilta tai mahdollisesti julkaisijoilta ja myyvät ne eteenpäin (Manninen ym., 2006).

2.5. Pelikehitys

Pelikehityksellä tarkoitetaan ohjelmistokehitysprosessia, jonka aikana luodaan peli jollekin alustalle, kuten PC:lle (Bethke, 2003). Pelikehitys voidaan ajatella uniikkina sovelluskehityksen osa-alueena, johon tarvitaan panostusta useilta eri alueilta, kuten musiikista, taiteesta, näyttelystä ja ohjelmoinnista. Pelikehitystä on myös verrattu elokuva- ja televisiosarjatuotantoon. Syynä tähän on ollut se, että pelikehityksen lopputuote on viihdettä, ja voidaan nähdä, että luovuus on hyvin kriittisessä asemassa pelikehityksessä (Manninen ym., 2006). Olennaisena osana pelikehitystä ovat myös prototyypitys ja iteratiivisuus (Kanode ym., 2009). Vaikeaksi pelikehityksen tekee se, että siihen tarvitaan panostusta useilta eri alueilta mikä tarkoittaa, että kehitystyössä tarvitaan monitaitoinen kehitystiimi, joka toimii saumattomassa yhteistyössä keskenään ja muiden oleellisten sidosryhmien kanssa.

Pelikehityksessä ei varsinaisesti ole mitään yksittäistä vallalla olevaa menetelmää, jonka mukaisesti pelejä tehtäisiin, vaan useimmissa tapauksissa pelistudioilla on jokin oma, mukautettu menetelmä (Manninen ym., 2006). Ketterien menetelmien on kuitenkin havaittu olevan suosittuja peliteollisuudessa (Musil, ym., 2010). Vaikka yksittäistä standardia prosessille ei ole, ovat Manninen ym. (2006) koostaneet kirjallisuuden pohjalta yleisen mallin siitä, mitä vaiheita pelikehitysprosessiin kuuluu. Mallin tarkoituksena on esittää ne vaiheet, jotka ovat läsnä pelikehityksessä menetelmästä riippumatta. Mannisen ym. (2006) kuvauksen mukaiseen yleisluontoiseen pelikehitysprosessiin kuuluvat seuraavat vaiheet:

1. Konseptin valmistelu

2. Esituotanto
3. Tuotanto
4. Laadunvarmistus ja testaus
5. Julkaisu
6. Ylläpito

Pelikehityksen lähtöpisteenä toimii aina peli-idea. Peli-idea on pelin elinkaaren läpi säilyvä visio pelistä, joka voi muuttua ja kehittyä sen aikana (Manninen ym., 2006).

Konseptin valmistelu-vaihe alkaa peli-idean kehittämisestä ja kypsyttämisestä. Tarkoituksena on koetella, hioa ja muokata ideaa, jotta päästäisiin ymmärrykseen pelin ydinideasta ja siitä, voidaanko idean pohjalta tuottaa peli. Kun idea on hiottu toteuttamiskelpoiseksi, luodaan alustava yhteenveto-dokumentti pelin ydinideasta, kohdeyleisöstä, kehitysprosessista ja siihen liittyvistä mahdollisista haasteista. Tämän dokumentin perusideana on saada oleelliset sidosryhmät, kuten julkaisija, rahoittajat ja asiakkaat, kiinnostuneeksi peli-ideasta. Tässä vaiheessa tuotetaan myös alustava pelisuunnitteludokumentti, alustavaa konseptitaidetta sekä alustavia prototyyppejä pelistä. Tässä vaiheessa kehitystiimillä ei tarvitse olla kaikkia mahdollisia yksityiskohtia peliin liittyen, mutta selkeä ymmärrys peli-ideasta täytyy olla (Manninen ym., 2006).

Seuraavana vaiheena on *Esituotanto*. Peliprojektien monimutkaisuuden sekä isojen kustannusten takia, on huolellinen suunnittelu elintärkeää peliprojekteissa. Esituotannossa perusideana on suunnitella, testata ja arvioida kaikki peliin liittyvä ennen varsinaisen tuotannon aloittamista. Hyvin toteutetussa esituotannossa saadaan parhaimmillaan lopputuloksena pelattava demo. Esituotannossa kaikki pelilliset vaatimukset tiivistetään ja näiden vaatimuksien seuraukset analysoidaan sekä karsitaan ne vastaamaan liiketoiminnallisia rajoitteita. Lisäksi vaatimusten osalta luodaan tarkat suunnitelmat teknisestä, äänellisestä sekä taiteellisesta toteutuksesta (Manninen ym,. 2006).

Esituotannon aikana pelin lähtökohtia ja teemoja syvennetään tekemällä tutkimusta muun muassa kirjallisuuden, elokuvien tai musiikin parissa sekä saatetaan tehdä pieni muotoisia retkiä pelimaailmaa muistuttaviin paikkoihin. Tämän lisäksi tehdään markkina-analyysi markkinoista ja kilpailevista tuotteista, hiotaan tuotantosuunnitelmadokumentaatiota, luodaan tarvittavia kehitystyökaluja sekä luodaan ja huotaan käsikirjoitusta. Esituotanto voidaan nähdä kriittisimpänä vaiheena peliprojektissa, ja ideaalisesti sen tulisi kuluttaa noin 25-40% julkaisua edeltävästä kehitysajasta (Manninen ym,. 2006).

Tuotanto on pelikehitysprosessin raskain vaihe henkilö- ja aikaresurssien osalta. Tuotantovaiheessa pelille luodaan kaikki käytettävät audiovisuaaliset ja pelimekaaniset elementit. Sen aikana luodaan konseptitaidetta, jonka perusteella pelin ympäristöt ja objektit, kuten pelaajahahmot, luodaan. Tarvittaessa objekteille luodaan animaatiot, joiden lisäksi tarvitaan myös mahdollisesti välianimaatiota kuljettamaan eteenpäin pelin juonta. Oleellisena osana tuotantovaihetta ovat myös kenttäsuunnittelu, 3D-suunnittelu, tekstuurien ja grafiikan luominen sekä äänimaailman suunnittelu ja toteutus. Pelielementit yhteensitovana tekijänä toimii taustalla ajettava ohjelmakoodi, joka luodaan myös tuotantovaiheessa. Näiden osa-alueiden lisäksi suoritetaan myös pelitestaamista, jonka tarkoituksena on arvioida ja analysoida pelin pelattavuutta ja tuottaa tietoa käyttökokemuksesta ja käytettävyydestä (Manninen ym., 2006).

Laadunvarmistus ja testaus on vaihe, joka limittyy koko tuotanto-vaiheen kanssa. Käytännössä tämä vaihe keskittyy tuotettavan tuotteen laatuun ja sen tarkoituksena on varmistaa, että käytettävissä olevien resurssien puitteissa syntyy paras mahdollinen lopputulos. Pääasiallisesti pelin ohjelmakoodin testaus toimii samankaltaisilla periaatteilla kuin ohjelmistotuotannossa. Testaus voidaan jakaa yksikkö-, integrointi-, järjestelmä-, konfiguraatio- ja regressiotestaukseen. Pelitestauksesta puhuttaessa taas tarkoitetaan pelin pelaamista. Kun peli on ominaisuuksiltaan valmis, aloitetaan niin sanottu alfatestaus. Alfatestausta seuraa Betatestaus, mikä alkaa siinä vaiheessa kun pelissä ei ole enää tiedossa olevia virheitä. Betatestaus voidaan jakaa avoimeen ja suljettuun beta-testaukseen, joiden pääasiallisena erona on se, että suljettuun beta-testaukseen voi osallistua vain, jos tulee valituksi, kun taas avoimeen beta-testaukseen ei tarvitse tulla valituksi, joten käytännössä kaikki halukkaat voivat osallistua testaukseen. Testauksen aikana löydetyt virheet listataan ja korjataan vakavuuden ja kiireellisyyden perusteella. Virheitä korjattaessa on myös hyvin tärkeää, että korjaukset suoritetaan huolellisesti uusien virheiden ilmaantumisen estämiseksi (Manninen ym., 2006).

Koko pelikehitysprosessi huipentuu *Julkaisuun*. Julkaisuvaiheessa peli viimeistellään ja toimitetaan julkaisijalle monistusta ja jakelua varten. Viimeistään tässä vaiheessa luodaan käyttäjälle tarkoitetut dokumentit kuten ohjekirjat tai apua-tiedostot. Lisäksi luodaan mahdolliset jakelua tukevat materiaalit, tukipalvelut sekä lokalisointi eri markkina-alueille (Manninen ym., 2006).

Julkaisun jälkeen peliprojekti etenee *ylläpitovaiheeseen*. Ylläpitovaiheessa julkaistaan mahdollisia korjauksia poistamaan pelistä virheitä, korjaamaan pelielementtien välistä tasapainoa tai estämään mahdollisten huijauksien

käyttämistä. Tässä vaiheessa peliin voidaan myös julkaista uutta sisältöä, joko maksullisesti tai ilmaiseksi. Uusi sisältö voidaan julkaista korjauksenomaisesti tai mahdollisina lisäosina. Kehittäjät voivat osallistua tässä vaiheessa myös edustustehtäviin erilaisissa markkinatapahtumissa. On myös tärkeää, että tässä vaiheessa kehittäjät huolehtivat pelinsä yhteisöstä ja kuuntelevat heidän mielipiteitään (Manninen ym., 2006). Nämä mielipiteet voivat olla erittäin kriittisiä esimerkiksi uuden sisällön, virheiden korjaamisen ja pelielementtien tasapainon muokkaamisen kannalta.

Kanode ja Haddad (2009) puolestaan esittävät pelikehitysprosessin koostuvan esituotannosta, tuotannosta ja testauksesta. Kanoden ja Haddadin esituotanto kattaa peli-idean luomisen ja pelin prototyyppien tuottamisen. Tässä vaiheessa tuotetaan myös pelisuunnitteludokumentti (Kanode ym. 2009). Tuotannon Kanode ja Haddad (2009) kuvaavat vaiheeksi, jonka aikana tuotetaan suurin osa teknillisistä, taiteellisista ja äänellisistä vaatimuksista. Vaiheen aikana luodaan usein uusia prototyyppejä tuotettavasta pelistä ja tällä tavoin peliä kehitetään iteraatioissa valmista tuotetta kohti (Kanode ym. 2009). Testauksen Kanode ja Haddad (2009) esittävät vaiheeksi, joka edeltää pelin julkaisua. Testaus-vaiheen aikana peliä muun muassa stressitestataan ja virheet pyritään poistamaan ennen julkaisua. Kanode ja Haddad (2009) mainitsevat, että vaiheet ovat voimakkaasti sidoksissa toisiinsa ja esimerkiksi tuotannossa voi esiintyä aktiviteetteja jotka voidaan nähdä kuuluvan esituotantoon tai testaukseen. Mannisen ym. (2006) esityksestä poiketen, Kanode ja Haddad (2009) eivät esitä mainintaa julkaisu- tai ylläpito-vaiheesta. Lisäksi konseptin valmistelu-vaihetta ei mainita ollenkaan, mutta Kanoden ja Haddadin (2009) kuvauksen esituotanto vaikuttaa sisältävän Mannisen ym. (2006) kuvaaman konseptin valmistelu-vaiheen tehtäviä.

2.6. Pelikehityksen ongelmia

Ohjelmakehityksen tavoin myös pelikehityksessä on ongelmia. Keith (2010) näkee pelikehityksen isoimmiksi ongelmiksi muun muassa vaatimusten liiallisen kasvun, huonon aikatauluttamisen sekä erinäiset tuotannon haastavuudet kuten tehokkuuden maksimointi ja niin sanotun hukan minimoiminen. Sinänsä vaatimusten lisääntymistä ei nähdä ongelmaksi, vaan ongelman ytimessä on vaatimusten lisääminen ilman, että budjettia tai aikataulua muutetaan vastaamaan muuttunutta tarvetta. Tällaiseen tilanteeseen voidaan päätyä hyvinkin huomaamatta, jos projektin johto ei pidä huolellista

kirjaa vaatimusten määrästä. Huono aikataulutus voi taas johtua yllättävien tekijöiden jättämisestä huomiotta. Yllättäviin tekijöihin voi kuulua esimerkiksi vaatimusten kasvu. Luonnollisesti myös mahdolliset koodausvirheet ja muut vastaavat ongelmat lisäävät pelikehitysprosessiin tarvitun ajan määrää (Keith, 2010).

Pelikehityksen yleisimmistä ongelmista saa hyvän käsityksen perehtymällä pelien jälkiselvittelyraportteihin (vrt. Gamasutra). Tällaisten jälkiselvittelyraporttien pohjalta Petrillo ym. (2009) ovat koonneet listan pelikehityksessä usein esiintyvistä ongelmista. Heidän mukaan pelikehityksen ongelmia ovat ongelmat mittakaavan määrittelyssä, epärealistinen laajuus, ominaisuuksien kasvu, ominaisuuksien karsiminen kehityksen aikana, ongelmat suunnitteluvaiheessa, viivästykset, tekniset ongelmat, loppurutistukset, dokumentaation puute, kommunikaatio-ongelmat, työkaluongelmat, testausongelmat, ongelmat ryhmänkoostumuksessa, ohjelmistovirheiden määrä, kehittäjien menettäminen ja budjetin ylitys. Seuraavassa kuvataan ongelmia hieman tarkemmin.

Ongelmat mittakaavan (engl. scope) *määrittelyssä* voivat johtua useista tekijöistä kuten kehitystiimin kokemattomuudesta tai siitä, ettei pelillisiä ominaisuuksia kiinnitetä tarpeeksi ajoissa (Petrillo ym., 2009). Hyvin usein nämä ongelmat johtuvat myös pelisuunnitelmadokumentin huonosta kääntämisestä projektisuunnitelmaksi (Kanode & Haddad, 2009).

Epärealistinen mittakaava tarkoittaa yleisesti sitä, että kehitystiimi on ollut alkuvaiheessa liian kunnianhimoinen ja peli on hahmoteltu liian haastavaksi ottaen huomioon kehitystiimin kyvyt ja resurssit (Petrillo ym., 2009).

Jos peliin lisätään ominaisuuksia ilman tarkempaa suunnittelua, voidaan puhua *ominaisuuksien kasvusta*. Tämä johtaa siihen, että projektin mittakaava kasvaa, jolloin myös työmäärä kasvaa, mikä voi johtaa esimerkiksi viivästyksiin tai budjetin ylityksiin. Tällöin on hyvin tyypillistä, että *ominaisuuksia karsitaan*. Ominaisuuksien karsiminen voi myös johtua epärealistisesta mittakaavasta. Hyvin tyypillistä on myös karsia ominaisuuksia aikataulun tai budjetin säilyttämiseksi (Petrillo ym., 2009). Ominaisuuksien kasvu ei kuitenkaan itsessään ole ongelma, jos lisättävät ominaisuudet voidaan toteuttaa budjetin ja aikataulun puitteissa (Keith, 2010). Hyvin usein ongelmana on kuitenkin se, että ominaisuuksia lisätään vähitellen, jolloin voi olla hyvin vaikeaa huomata niiden vaikutus aikatauluun tai budjettiin (Keith, 2010).

Suunnitteluvaiheen ongelmat johtuvat pääosin liian vähäisestä tai liiallisesta etukäteissuunnittelusta. Peliprojekteihin osallistuu myös hyvin paljon erilaisia ihmisiä kuten ohjelmoijia, taiteilijoita ja tuottajia, jotka kaikki saattavat osallistua pelisuunnitteluun. Osallistujilla voi myös olla erilaisia näkemyksiä

siitä, mihin suuntaan peliä tulisi viedä, mikä tuottaa myös ongelmia (Petrillo ym., 2009; Kanode & Haddad, 2009).

Viivästykset ovat yleensä muiden pelikehityksessä esiintyvien ongelmien seuraus. Epärealistiset odotukset, ongelmat suunnitteluvaiheessa, ominaisuuksien kasvu tai kehittäjien menetys vaikuttavat suoraan aikatauluun ja voivat myös tarpeeksi vakavina aiheuttaa viivästyksiä (Petrillo ym., 2009). Viivästykset voivat myös johtua ylioptimistisista aikatauluista. Yksinkertaisten tehtävien kohdalla aikataulutus on helpompaa, mutta tarkkuus vähenee sitä mukaa, mitä vaikeammaksi tehtävä tulee (Keith, 2010),

Tekniset ongelmat ovat myös mahdollisia peliprojektin aikana. Jälkiselvittelyraporttien mukaan monet näistä ongelmista johtuvat kolmannen osapuolen komponenttien ohjelmointirajapintojen heikkouksista. Lisäksi hyvin usein pelikehittäjät työskentelevät uusimpien teknologisten innovaatioiden parissa, mikä tekee kehittämisestä vaativampaa ja enemmän aikaa vievää (Petrillo ym., 2009). Ongelmat voivat liittyä myös kolmannen osapuolen teknologioihin, kuten pelimoottoreihin. On tapauksia, joissa pelille on valittu pelimoottori, jolla peliä ei lopulta pystytty tuottamaan tai pelimoottori toimi hidastavana rajoitteena (Kanode & Haddad, 2009).

Loppurutistuksella (engl. crunch time) tarkoitetaan ajanjaksoa, jolloin tehdään kiivaasti töitä, jotta peli saataisiin valmiiksi. Pelikehityksen yhteydessä tämä tarkoittaa hyvin usein valtaisaa määrää ylityötunteja hyvinkin pitkillä aikaväleillä. Petrillon ym. (2009) tutkimissa raporteissa nämä ajanjaksot vaihtelivat muutamista viikoista jopa vuoden mittaisiin rutistusjaksoihin. Rutistusjaksot ovat kuitenkin ongelmallisia, koska ne heikentävät niin työkykyä kuin myös pitemmillä aika-väleillä päätöksen tekoa.

Dokumentaation puute tarkoittaa usein sitä, ettei jotain ominaisuutta ole dokumentoitu tarpeeksi hyvin, jolloin ainoastaan ominaisuuden toteuttaja kykenee laajentamaan tai muokkaamaan sitä. Lisäksi joissain tapauksissa voi myös käydä niin, että joudutaan käyttämään aikaa ominaisuuksien toimivuuden hahmottamiseen (Petrillo ym., 2009).

Peliprojekteissa *kommunikaatio-ongelmat* voivat liittyä pelistudion sisäiseen kommunikointiin tai pelistudion ja sidosryhmien väliseen kommunikointiin. Pelistudion sisällä ongelmat liittyvät lähinnä kehitystiimien monimuotoisuuteen. Pelikehityksen parissa työskentelee paljon eritaustaisia ja hyvinkin erilaisia taitoja omaavia ihmisiä (Kanode & Haddad, 2009). Tämän lisäksi tiimit jakaantuvat hyvin usein artisteihin, ohjelmoijiin ja suunnittelijoihin (Petrillo ym., 2009; Kanode & Haddad, 2009). Tämä johtaa helposti ”me vastaan he” -ajatteluun (Kanode & Haddad, 2009).

Testausongelmat liittyvät usein aikataulujen lipsumiseen, mikä taas pienentää testivaiheelle tarkoitettua aikaa. Lisäksi kehityksen loppuvaiheeseen sijoittuvan testivaiheen vaatima aika ja resurssit on usein aliarvioitu. Julkaisijaa pidetään osasyyllisenä, koska usein julkaisija on aliarvioinut laadunvarmistukseen vaaditut resurssit (Petrillo ym., 2009). Testaamisen voidaan nähdä olevan erittäin tärkeässä asemassa, sillä sen avulla peliprojektien aikana on löydetty suuria määriä *ohjelmistovirheitä*, joiden korjaaminen syö valtaosan pelikehityksen loppuvaiheesta (Petrillo ym., 2009).

Ryhmänkoostumuksellisiin ongelmiin kuuluvat ongelmat palkata tarpeeksi kehittäjiä, ongelmat tiimin yhteishengessä sekä myös tiiminjäsenten kokemattomuus (Petrillo ym., 2009). Koostumukselliset ongelmat voivat viitata myös tiimin jäsenten välisiin ihmissuhdeongelmiin (Kanode & Haddad, 2009). Kaikki eivät tule välttämättä toimeen toistensa kanssa, ja ”me vastaan he” -ajattelu on varmasti myös keskeinen tekijä ryhmänkoostumuksellisissa ongelmissa.

Kehittäjien menettämisellä tarkoitetaan ongelmia, jotka syntyvät, kun kehitystiimin muut tai korvaava jäsen eivät kykenekään jatkamaan projektista poistuneen tai mahdollisesti sairastuneen kehittäjän töitä (Petrillo ym., 2009). *Budjetin ylitys* ei Petrillon ym. (2009) suorittaman tutkimuksen mukaan ollut kovin yleinen ongelma pelikehityksen parissa. Petrillon ym. (2009) raportoivat, että budjetin ylitys oli ongelma ainoastaan kahdessa tapauksessa heidän otannassaan.

2.7. Yhteenveto

Peliala on useista eri toimijoista koostuva viihteen ala. Se on kasvanut viime vuosien aikana voimakkaasti, ja voidaankin jo väittää, että pelaaminen on saavuttanut valtavirran suosion. Pelialalta löytyy paljon erilaisia toimijoita kuten kehittäjiä, julkaisijoita sekä kuluttajia. Kehittäjiin voidaan lukea niin ohjelmoijat kuin myös artistit, jotka osallistuvat pelikehitykseen. Julkaisijat ovat yleensä niitä osapuolia, jotka rahoittavat pelikehityksen. Kuluttajat taas ovat joko itse pelaajia tai muita tahoja, jotka ovat ostaneet pelin tai pelipalvelun. Näiden toimijoiden lisäksi voidaan tunnistaa myös alusta- ja pelimoottorikehittäjiä, rahoittajia, jakelijoita sekä jälleenmyyjiä.

Peli voidaan määritellä järjestelmäksi, jossa pelaajat osallistuvat teennäiseen konfliktiin, joka on määritelty säännöin ja jolla on laskettavissa oleva lopputulos. Määritelmän ovat kehittäneet Salen ja Zimmerman (2003), ja

tämän tutkimuksen yhteydessä pelit käsitetään tämän määritelmän mukaisesti. Pelit itsessään voidaan jakaa hyvinkin monimuotoisiin genreihin, kuten räiskintäpeleihin, strategiapeleihin ja simulaatiopeleihin.

Pelikehityksessä ei ole käytössä yhtenäistä prosessimallia. Yleisellä tasolla pelikehitysprosessiin kuitenkin kuuluvat konseptin valmistelu, esituotanto, tuotanto, laadunvarmistus, julkaisu sekä ylläpito. Vaikka laadunvarmistus on erotettu omaksi kokonaisuudekseen, tapahtuu sitä luonnollisesti myös koko pelikehitysprosessin ajan, koska pelikehitys on luonteeltaan iteratiivista. Ylläpito puolestaan keskittyy päivityksien ja lisäosien tuotantoon, tiedotus- ja suhdetoimintaan sekä peliyhteisötoimintaan.

3. KETTERÄ OHJELMISTOKEHITYS

Tässä luvussa luodaan yleiskuva ketterästä lähestymistavasta, kuvataan ketteriä menetelmiä sekä kerrotaan ketterästä ohjelmiston ylläpidosta. Aluksi esitellään ketterää lähestymistapaa Agile-manifestin näkökulmasta ja kerrotaan yleisesti ketterien menetelmien ominaispiirteistä. Toiseksi kuvataan kolmea yleisintä ketterää menetelmää, jotka ovat Scrum, XP ja Kanban. Lopuksi kerrotaan ketterän lähestymistavan käytöstä ohjelmiston ylläpidossa.

3.1. Ketterä lähestymistapa

Ketterästä lähestymistavasta on esitetty useita näkemyksiä (vrt. Laanti 2013; Laanti ym. 2013). Tässä ei ole mahdollisuutta käsitellä niitä tarkemmin. Yhteistä käsityksille on se, että ne nojaavat enemmän tai vähemmän Agile-Manifestiin (Beck ym., 2001). Agile-manifestin on kirjoittanut joukko ohjelmisto- ja menetelmäkehittäjiä, jotka olivat luoneet kevyempiä ohjelmistokehitysmenetelmiä vastaamaan perinteisen ohjelmistokehityksen haasteisiin (Beck ym., 2001). Vaikka Agile-manifesti itsessään julkaistiin 2001, jotkut ketterät menetelmät olivat jo olemassa ennen sen julkaisua. Agile-manifestissa on esitetty neljä arvoa ja 12 periaatetta. Agile-manifestin arvot esitetään seuraavalla tavalla (Beck, ym., 2001):

- Yksilöt ja vuorovaikutus ennen prosesseja ja työkaluja
- Toimiva ohjelmisto ennen kattavaa dokumentaatiota
- Asiakasyhteistyö ennen sopimusneuvotteluita
- Muutokseen vastaaminen ennen suunnitelman noudattamista.

Ensimmäisellä arvolla painotetaan kehittäjien välisiä suhteita ja yhteistyötä sekä ihmisten roolia kehitysprosessissa. Toinen arvon tarkoituksena on motivoida kehittäjät pitämään tuotettu koodi mahdollisimman yksinkertaisena ja helposti ymmärrettävänä, jolloin ei tarvita kattavaa dokumentaatiota. Kolmannella arvolla halutaan painottaa kehittäjien ja asiakkaiden välistä suhdetta. Ennen kaikkea tulisi keskittyä yhteistyöhön ja toimivan yhteistyösuhteen rakentamiseen. Neljännellä arvolla tarkoitetaan, että projektitiimi on valmis tekemään tarvittaessa muutoksia. Lisäksi sopimukset tulisi räätälöidä muutokset mahdollistaviksi.

Tärkeää näissä arvoissa on huomioida se, että vaikka vasemman puoleiset arvot ovatkin enemmän arvostettuja, ei oikean puoleisia arvoja tule unohtaa täysin. Agile-manifestiin kuuluu myös kaksitoista periaatetta, joiden tarkoituksena on tukea ja konkretisoida edellä esitettyjä arvoja (Beck ym., 2001). Tämän tutkielman yhteydessä keskittyminen Agile-manifestissa esitettyihin arvoihin riittää, joten ei nähdä oleelliseksi esitellä periaatteita ja niiden sidoksia aiemmin esitettyihin arvoihin.

Ketterillä menetelmillä tarkoitetaan yleisesti ottaen niitä ohjelmistokehitysmenetelmiä, joiden perustana ovat Agile-manifestissa esitetyt arvot. Ydinajatuksena on luoda muutosta sekä vastata muutokseen (Highsmith & Cockburn, 2001). Tärkeää ketterässä ajattelussa on tunnistaa projektiin osallistuvat ihmiset menestymisen avaintekijöiksi. Tämä ei yksinään tietenkään riitä, vaan tarvitaan myös keskittymistä tehokkuuteen ja ohjattavuuteen. Ketterien menetelmien tavoitteena on vastata nykypäivän ohjelmistokehityksen haasteisiin kuten jatkuvaan muutokseen, markkinoiden vaatimuksiin sekä vaatimukseen tuottaa innovatiivisia ja korkeatasoisia ohjelmistoja (Beck ym., 2001; Abrahamsson ym. 2002).

Keskeistä ketterässä kehittämisessä ovat inkrementaalisuus, vuorovaikuttaminen, yksinkertaisuus sekä mukautuvuus (Abrahamsson, Salo, Ronkainen & Warsta, 2002). *Inkrementaalisuus* tarkoittaa sitä, että ohjelmistoa julkaistaan pienissä osissa, jotka on kehitetty lyhyissä sykleissä. *Vuorovaikuttamisella* tarkoitetaan asiakkaan ja kehittäjien jatkuvaa yhdessä työskentelyä ja kommunikointia ohjelmistoprojektin kaikkien vaiheiden aikana. *Yksinkertaisuudella* tarkoitetaan sitä, että menetelmä on helppo oppia, tarpeen vaatiessa mukauttaa ja myös sitä, että menetelmä on hyvin dokumentoitu. *Mukautuvuudella* tarkoitetaan, että tarvittaessa menetelmä mahdollistaa viime hetken muutokset sekä luonnollisesti myös mukautumisen muutoksiin koko projektin aikana.

Useimmissa ketterissä menetelmissä isot työkokonaisuudet jaetaan pienemmiksi osiksi, joita työstetään iteraatioissa. Iteraatioiden pituus riippuu menetelmästä, mutta tyypillisesti se vaihtelee yhdestä neljään viikkoon. Näiden

iteraatioiden aikana projektiryhmän kaikki osa-alueet ja niistä vastaavat henkilöt toimivat yhteistyössä. Jokaisen iteraation lopussa tuotetut tulokset esitellään sidosryhmän jäsenille. Näillä toimenpiteillä on tarkoitus minimoida projektiin liittyviä riskejä ja mahdollistaa nopea reagointi muutoksiin(Abrahamsson, Salo, Ronkainen & Warsta, 2002).

Esimerkkejä ketteristä menetelmiä ovat Scrum (Schwaber & Sutherland, 2013), Extreme Programming (XP) (Beck & Anders, 2004), Feature Driven Development (FDD) (Palmer & Felsing, 2002), Dynamic Systems Development Model (DSDM) (Stapleton, 1997), Crystal Methods(Cockburn, 2002) ja Adaptive Software Development (O'Reilly, 1999). Yleisesti ottaen nämä menetelmät toimivat kehyksinä sille, millaisiin vaiheisiin projekti jaetaan ja minkälaisia tehtäviä ne sisältävät. Menetelmien tarkoituksena on helpottaa ja parantaa suunnittelua ja hallinnointia. Jotkut menetelmistä sisältävät myös ennalta määrättyjä rooleja ja artefakteja eli tuotoksi. Näiden tarkoitus on helpottaa projektiryhmän organisointia ja kehitystyötä sekä antaa selkeät kehykset sille, mitä heiltä odotetaan.

3.2. Ketteriä menetelmiä

Nykyisin ketterät menetelmät ovat suuressa suosiossa niin perinteisen ohjelmistokehityksen kuin myös pelikehityksen yhteydessä. Vuonna 2015 julkaistun Stage of Agile (Analysis.Net Research, 2015) kyselytutkimuksen mukaan ohjelmistokehityksessä yleisimpiä ketteriä menetelmiä ovat Scrum, Lean, Kanban, FDD sekä XP. Näistä selkeästi suosituin on Scrum ja vähiten käytössä on XP. Näiden menetelmien lisäksi useat erilaiset yhdistelmät, kuten Scrumban ja Scrum/XP ovat suosittuja. Seuraavaksi kuvataan Scrum, XP ja Kanban.

3.2.1. Scrum

Scrum on ketterien menetelmien piiriin kuuluva monimutkaisten tuotteiden, kuten ohjelmistojen, kehitykseen ja ylläpitoon tarkoitettu prosessiviitekehys (Schwaber ym., 2013). Scrumia on käytetty ohjelmistotuotekehityksessä jo 1990-luvun puolivälistä lähtien (Schwaber, 1995). Scrum ei ole prosessi tai tekniikka, minkä mukaisesti tuote tuotettaisiin, vaan se tarjoaa viitekehyksen, jonka puitteissa voidaan hyödyntää erilaisia projektinhallinnallisia käytänteitä. Scrumin tarkoituksena on olla kevyt ja helposti ymmärrettävä, mutta sen

täydellinen ymmärtäminen ja toteuttaminen on vaikeaa. Scrumin pohjana toimii empirismi. Se tarkoittaa sitä, että tieto syntyy kokemuksesta ja että päätökset tehdään olemassa olevan tiedon perusteella. Scrum on iteratiivinen ja inkrementaalinen lähestymistapa, jolla voidaan optimoida ennakoitavuutta ja kontrolloida riskejä (Schwaber ym., 2013). Iteratiivisuus ja inkrementaalisuus on toteutettu jakamalla prosessi osiin, joita kutsutaan sprinteiksi.

Scrum-viitekehys koostuu rooleista, tapahtumista (engl. events), artefakteista (engl. artefacts) ja säännöistä. Jokainen viitekehyksen komponentti palvelee tiettyä tarkoitusta ja on näin ollen oleellinen osa Scrumin onnistunutta hyödyntämistä. Ydinideana Scrumissa on se, että järjestelmäkehityksen parissa on paljon ympäristöllisiä ja teknillisiä muuttujia, kuten vaatimukset, aikataulu, resurssit ja teknologia, ja on hyvin todennäköistä, että nämä muuttujat muuttuvat prosessin aikana. Tämä tekee kehitysprosessista vaikeasti ennakoitavan ja monimutkaisen ja siksi järjestelmäkehitysprosessilta vaaditaan joustavuutta (Schwaber ym., 2013;Abrahamsson ym., 2002). Seuraavassa kerrotaan hieman lähemmin rooleista ja tapahtumiin liittyvistä käytänteistä.

Roolit

Scrum-viitekehys sisältää seuraavat roolit: Scrum-tiimi, tuotteen omistaja, kehitystiimi ja Scrum-mestari. *Scrum-tiimi* on tuotteen omistajasta, kehitystiimistä sekä Scrum-mestarista koostuva kokonaisuus. Scrum-tiimi on itsestään organisoituva. Scrum-tiimin tarkoituksena on toteuttaa suunniteltu tuote iteratiivisesti ja inkrementaalisesti ja samalla pyrkiä maksimoimaan mahdollisuudet palautteen saamiselle. Scrum-tiimi on suunniteltu siten, että se on mahdollisimman joustava, innovoiva ja tuottelias (Schwaber ym., 2013).

Tuotteen omistajan (engl. Produt owner) vastuulla on maksimoida toteutettavana olevan tuotteen ja kehitystiimin työn arvo. Se miten tuotteen omistaja saavuttaa tämän, riippuu täysin organisaatiosta, Scrum-tiimistä ja projektiin osallistuvista muista sidosryhmistä. Tuotteen omistajan muihin vastuualueisiin kuuluu myös tuotteen työlistan (engl. Product backlog) ylläpitäminen. Tuotteen omistajan tulee olla yksittäinen henkilö, joka voi kuitenkin olla vastuussa esimerkiksi johtoportaalle. Jotta tuotteen omistaja voi onnistua tehtävässään, tulee koko organisaation kunnioittaa hänen päätöksiä (Schwaber ym., 2013).

Kehitystiimi koostuu ammattilaisista, joiden tehtävänä on tuottaa sprintin aikana julkaisukelpoinen osa tuotetta. Kehitystiimit on rakennettu ja valtuutettu siten, että ne organisoivat ja hallinnoivat itse omaa työtään. Kehitystiimin sisällä ei ole muita tittelejä kuin kehittäjä. Koon puolesta kehitystiimien tulisi olla tarpeeksi pieniä jotta ne säilyvät ketterinä, mutta että ne kuitenkin

kykenevät tuottamaan merkittäviä tuloksia sprintin aikana (Schwaber ym., 2013).

Scrum-mestari (engl. Scrum master) on yksittäinen henkilö, jonka vastuulla on huolehtia siitä, että Scrum-prosessi on täysin ymmärretty ja toteutuu. Scrum-mestari voi myös auttaa tuotteen omistajaa ja kehitystiimiä eri tavoin, kuten tuomalla esiin tehokkaampia menetelmiä tuotteen työlistan hallinnointiin tai valmentamalla kehitystiimiä olemaan itse organisoituva (Schwaber ym., 2013).

Käytänteitä

Scrum itsessään ei vaadi tai tarjoa mitään ohjelmistokehityskäytänteitä. Sen sijaan Scrum sisältää käytänteitä, jotka on tarkoitettu helpottamaan projektihallinnointia. Näillä käytänteillä pyritään estämään ennakoimattomuuden ja monimutkaisuuden aiheuttamia ongelmia (Schwaber yms., 2013).

Tuotteen työlista on luettelo kaikesta siitä, mitä lopullisen tuotteen tulisi sisältää perustuen sen hetkiseen tietämykseen. Voidaankin siis sanoa, että tuotteen työlista kuvaa kaiken sen työn, mitä projektin aikana täytyy tehdä. Se on lista priorisoiduista ja jatkuvasti päivitetyistä liiketoiminnallisista ja teknillisistä vaatimuksista ohjelmistolle tai järjestelmälle. Työlista voi sisältää muun muassa toimintoja (engl. functionalities), virheenkorjauksia, vikoja (engl. bugs) ja teknillisiä päivityksiä. Listan sisällön kartuttamiseen voi osallistua useita tahoja, mutta sen ylläpitäminen on tuotteen omistajan vastuulla (Schwaber ym., 2013). Tuotteen työlistan sisältämien vaatimuksien kehittämiseen kuluva aika tulee myös arvioida jokaiselle vaatimukselle erikseen aloittaen tärkeimmistä vaatimuksista. Scrum-tiimi on pääsiallisesti vastuussa toteutusaikojen arvioinneista, mutta tuotteen omistaja voi auttaa Scrum-tiimiä arvioinnin tekemisessä (Schwaber ym., 2013).

Sprintin työlista (engl. Sprint backlog) on jokaisen sprintin aloituspiste. Se on lista niistä tuotteen työlistan kohdista, jotka on valittu toteutettaviksi sprintin aikana. Sprintin työlista voi kehittyä sprintin aikana, kun Scrum-tiimi työstää työlistan vaatimuksia ja lisää uuden tiedon perusteella uusia vaatimuksia työlistaan. Koska sprintin työlista on kehitystiimin käyttöön tarkoitettu, voi ainoastaan kehitystiimi itse tehdä siihen muutoksia. (Schwaber ym., 2013).

Sprintti on Scrum prosessin ydin, ja se on noin kuukauden mittainen ajanjakso, jonka aikana Scrum-tiimi työskentelee tuottaakseen uuden käyttökelpoisen palan ohjelmistoa. Sprintin aikana ei tule tehdä muutoksia, jotka saattavat vaarantaa sille asetetut tavoitteet (Schwaber ym., 2013). Sprintit koostuvat sprintin suunnittelusta, päivittäisistä Scrum-tapaamisista,

kehitystyöstä, sprintin katselmoinnista ja sprintin retrospektiivistä (Schwaber ym. 2013).

Sprintin suunnittelutapaaminen on ennen sprinttiä järjestettävä sprintin sisältöä ja tavoitetta käsittelevä tapaaminen. Sprintin suunnittelutapaamisen osallistujiin kuuluu kehitystiimi, Scrum-mestari, tuotteen omistaja ja asiakas sekä tarpeen mukaisesti myös muut oleelliset sidosryhmät. Sprintin suunnittelutapaaminen on kaksi osainen. Ensimmäisessä osassa tavoitteena on asettaa sprintille tavoite ja päättää ohjelmistoon tuotettavista toiminnallisuuksista. Näiden päätösten pohjalta luodaan sprintin työlista. Toisessa osassa Scrum-mestari ja Scrum-tiimi keskittyvät siihen, kuinka sprintin työlistaa lähdetään toteuttamaan (Schwaber ym., 2013).

Päivittäiset Scrum-tapaamiset ovat lyhyitä, noin 15 minuuttia kestäviä kokoontumisia. Näiden tarkoituksena on seurata Scrum-tiimin etenemistä sekä toimia eräänlaisina suunnittelukokoontumisina. Kokoontumisten aikana käydään läpi, mitä on tehty viime kokouksen jälkeen ja mitä tulee tehdä ennen seuraavaa kokoontumista. Lisäksi käydään läpi mahdollisia ongelmia ja muita tärkeitä keskusteltavia asioita. Kaikki mahdolliset kehitysprosessin esteet pyritään tunnistamaan ja poistamaan, jotta kehitysprosessia voitaisiin parantaa. Scrum-mestari on vastuussa päivittäisistä Scrum-tapaamisista. Yleensä tapaamisiin osallistuvat Scrum-mestari sekä kehitystiimi, mutta myös johtoporras voi osallistua tapaamiseen halutessaan (Schwaber ym., 2013).

Sprintin katselmointi pidetään aina sprintin päätteeksi. Tapaamisen aikana Scrum-tiimi ja projektin kannalta oleelliset sidosryhmät käyvät läpi sprintin aikana tehdyt tulokset, sen aikana ilmentyneet ongelmat ja kuinka nämä ongelmat ratkaistiin. Lisäksi sprintin katselmoinnin aikana käsitellään sen hetkistä tuotteen työlistaa ja tehdään siihen muutoksia tarpeen vaatiessa. Sprintin katselmoinnin pituus riippuu pyrähdyksen pituudesta, mutta ohjeellinen maksimipituus on noin neljä tuntia kuukauden mittaiselle sprintille. (Schwaber ym., 2013)

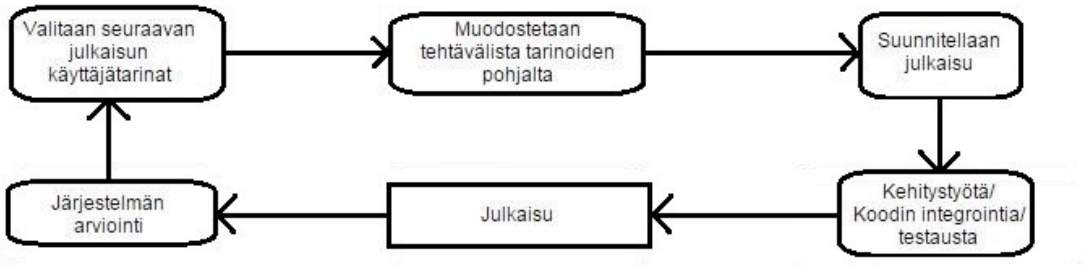
Sprintin retrospektiivin tarkoituksena on, että Scrum-tiimi tarkastelee ja arvioi omaa toimintaansa ja luo tämän pohjalta itselleen suunnitelman, jonka mukaan Scrum-tiimi parantaa toimintaansa seuraavan sprintin aikana. Sprintin retrospektiivi pidetään sprintin katselmoinnin jälkeen, mutta kuitenkin ennen seuraavaa sprintin suunnittelutapaamista. Sprintin retrospektiivin pituus riippuu täysin sprintin pituudesta, mutta pisimmillään se on ohjeellisesti kolme tuntia, kun sprintin pituus on noin yksi kuukausi (Schwaber ym., 2013).

3.2.2. XP

XP (Extreme Programming), on kevyt ohjelmistokehitysmenetelmä, jonka tarkoituksena on vastata ohjelmointikehityksen haasteisiin (Beck ym., 2004). XP sisältää joukon arvoja, periaatteita ja käytänteitä, joiden tarkoituksena on mahdollistaa korkeatasoisen ohjelmiston tuottaminen mahdollisimman nopeasti asiakkaalle ja näin samalla maksimoida siitä saatu arvo (Agarwal ym., 2008). XP:n nimi juontuu siitä, että se vie useat tunnetut ohjelmistokehityskäytänteet, kuten iteratiivisen kehityksen tai asiakkaan osallistumisen, ”äärimmilleen” (engl. extreme) (Sommerville, 2008; Agarwal ym., 2008). XP luotiin vastauksena ongelmallisiin ohjelmistokehityksen alueisiin, joissa vaatimukset muuttuivat jatkuvasti. XP voidaan nähdä yhtenä tehokkaimmista keinoista vastata muun muassa tilanteisiin, joissa asiakkaat eivät ole täysin varmoja siitä, mitä ominaisuuksia ohjelmistoon tarvitaan, jolloin vaatimukset todennäköisesti muuttumat usein ja hyvinkin nopeasti (Agarwal ym., 2008). Agarwal ym. (2008) toteavat, että XP on ideaali valinta uuden tyyppisten ohjelmistotuotteiden tuotantoon ja toimii myös hyvin silloin, kun projektissa joudutaan tekemisiin tiukkojen aikarajojen kanssa. XP:stä on muokattu erilaisia versioita, kuten niin sanottu PXP (Personal Extreme Programmin) (Agarwal ym. 2008), joka on tarkoitettu yksittäisen ohjelmoijan tarpeisiin (Agarwal ym., 2008). Seuraavassa kerrotaan hieman lähemmin XP:n prosessista ja käytänteistä.

XP-prosessi

XP keskittyy tilanteisiin, joissa pienet kehittäjätiimit suorittavat kehitystyötä. Useat XP:n käytänteet vaativat yhteistyötä. XP-prosessissa julkaisujen välinen aika on hyvinkin pieni (Agarwal ym., 2008; Sommerville, 2008). Yksinkertaisimmillaan yksi kehitysjakso voidaan kuvata kuvion 1 mukaisesti.



Kuvio 1: XP-prosessi (mukaillen Sommerville, 2008)

XP-prosessi alkaa seuraavassa julkaisussa toteutettavien käyttäjätarinoiden valinnalla. Kun käyttäjätarinat on valittu yhteistyössä asiakkaan kanssa, ne muunnetaan toteutettaviksi ominaisuuksiksi ja näiden ominaisuuksien pohjalta muodostetaan tehtävälista. Tehtävälistan valmistuttua suunnitellaan, kuinka julkaisu toteutetaan. Kun julkaisun yksityiskohdista on päästy yhteisymmärrykseen, alkaa kehitystyö. Kehitystyön aikana ominaisuudet toteutetaan, testataan ja kun toteutettu koodi on läpäissyt testit, integroidaan se osaksi olemassa olevaa ohjelmistoa. Kun toteutettavaksi valitut ominaisuudet on saatu onnistuneesti integroitua ohjelmistoon, julkaistaan uusi versio ohjelmistosta. Tämän jälkeen ohjelmistoa tarkastellaan ja pohditaan asiakkaan kanssa yhteistyössä, mitä uusia ominaisuuksia ohjelmistoon tulisi toteuttaa (Sommerville, 2008).

Vaatimusten osalta asiakasyhteistyö on myös erittäin oleellista. XP-prosessissa asiakkaat osallistuvat vaatimusten määrittelyyn ja priorisointiin. XP:ssä vaatimukset ilmaistaan skenaarioina, joita kutsutaan myös käyttäjätarinoiksi (engl. user stories), joiden pohjalta yhden kehitysjakson tehtävälistat toteutetaan (Sommerville, 2011). Nämä käyttäjätarinat muodostetaan yhdessä asiakkaan kanssa ja niiden tarkoituksena on kiteyttää asiakkaan tarpeet. Tehtävälistoja toteuttaessa, arvioidaan myös niiden toteuttamiseen vaadittu aika ja resurssit yhdessä asiakkaan kanssa (Sommerville, 2011). Tämän tarkoituksena on edesauttaa sitä, että asiakas kykenee helpommin priorisoimaan itselleen oleellisimmat ja eniten arvoa tuovat vaatimukset. Luonnollisesti vaatimusten muuttuessa ei-toteutettuja tehtäviä voidaan muuttaa tai ne voidaan poistaa kokonaan. Jos muutokset koskevat jo toteutettuja ominaisuuksia, luodaan muuttuneiden vaatimusten pohjalta uudet käyttäjätarinat ja tehtävälistat (Sommerville, 2011). Näissä tapauksissa asiakas myös yleensä päättää, priorisoidaanko muutokset toteutettaviksi ennen uusia ominaisuuksia. Kun ohjelmoijat työstävät tehtävälistoissa olevia tehtäviä, he työskentelevät pareina ja kirjoittavat yksikkötestit jokaiselle tehtävälle ennen koodin toteuttamista. Kaikkien testiajojen täytyy olla onnistuneita, ennen kuin uutta koodia voidaan yhdistää järjestelmään (Sommerville, 2011).

XP:n lähtökohta inkrementaaliseen kehitykseen on viedä se äärimmilleen. Tästä johtuen ohjelmiston uusia versioita saatetaan tuottaa hyvinkin monta muutaman päivän sisällä, ja yleisesti ottaen julkaisukelpoinen tuotos toimitetaan asiakkaalle aina parin viikon välein. Käytäntönä on, että julkaisu ajankohdista tulee pitää kiinni ja jos kehitysongelmia ilmenee, otetaan ensi tilassa yhteyttä asiakkaaseen ja mahdollisesti poistetaan tehtäviä kehitysjaksosta (Sommerville, 2011).

XP käytänteitä

XP:n yhteydessä on määritelty kaksitoista ydin-käytännettä (Sommerville 2011; Agarwal ym., 2008). Seuraavassa esitetään niistä lyhyet kuvaukset.

Suunnittelupelin ytimenä on kehitystiimin ja asiakkaan yhteistyönä tapahtuva käyttäjätarinoiden kirjoittaminen, joka voidaan nähdä eräänlaisena neuvottelutilanteena siitä, mitä vaatimuksia ohjelmistolle esitetään, kauanko niiden toteuttamiseen menee ja mitkä vaatimukset toteutetaan ensimmäisenä. Ideana on siis, että asiakas ja kehitystiimi yhdessä päättävät siitä, mitkä ominaisuudet ovat arvokkaimpia asiakkaalle (Agarwal ym., 2008).

Pienillä julkaisuilla tarkoitetaan sitä, että toteutetaan aina yksinkertainen osa ohjelmistoa, joka sisältää oleellisia ominaisuuksia, nopeasti ja liitetään se osaksi kokonaisuutta lyhyissä kehitysjaksoissa (Agarwal ym., 2008).

Vertauskuvilla (engl. metaphors) tarkoitetaan sitä, että luodaan lista nimityksistä ja niiden selityksistä helpottamaan ja nopeuttamaan kommunikointia osallistuvien osapuolien välillä (Agarwal ym., 2008).

Yksinkertainen suunnittelu tarkoittaa sitä, että tuotetta kehitettäessä käytetään aina mahdollisimman yksinkertaista toteutusta siten, että tuote vastaa sen hetkisiä vaatimuksia. Taustana on ajatus siitä, että vaatimukset muuttuvat kuitenkin tulevaisuudessa, joten yksinkertaisesti toteutettu tuote on helpompi muuttaa vastaamaan uusia vaatimuksia (Agarwal ym., 2008).

Testaaminen ja testausvetoinen kehitys (engl. Test-driven development, TDD) ovat yksi XP:n tärkeimmistä käytänteistä. Käytännössä kaikki ohjelmisto, mikä tuotetaan XP:n periaatteiden mukaisesti, validoidaan jatkuvasti. Ennen kuin mikään ominaisuus pääsee osaksi valmista kokonaisuutta, testataan ne erittäin tarkasti (Agarwal ym., 2008).

Refaktorointi on tekniikka, jonka tarkoituksena on parantaa olemassa olevan koodin rakennetta. Käytännössä koodiin tehdään parantavia muutoksia pienin askelin, jotta vältyttäisiin mahdollisilta virheiltä (Agarwal ym., 2008).

Pariohjelmointi on käytänne, jonka mukaisesti ohjelmoijat työskentelevät aina pareittain. Tarkoituksena on, että kun toimitaan pareittain, koodin rakennetta tarkastellaan jatkuvasti ja näin saadaan tuotettua laadukkaampaa koodia (Agarwal ym., 2008).

Yhteinen omistajuus tarkoittaa sitä, että kaikki tuotettu koodi kuuluu koodaustiimin jokaiselle jäsenelle. Kaikki tiimin jäsenet voivat vapaasti tehdä muutoksia mihin tahansa osioon milloin tahansa. Tämän käytänteen tarkoituksena on motivoida tiimin jäseniä tuottamaan uusia ideoita (Agarwal ym., 2008).

Jatkuvan integraation tavoitteena on estää ja vähentää koodin rönsyilyä tuotteen pääkoodista. Tämän tarkoituksena on estää mahdolliset eri poluille

ajautumiset. Koodin integrointia voidaan tehdä useita kertoja päivässä, mutta kuitenkin vähintään kerran päivässä. Jokainen uusi osa tulee testata ennen integroinnin suorittamista (Agarwal ym., 2008).

40-tuntinen työviikko on käytänne, jonka mukaisesti työviikot pidetään 40 tunnin mittaisina, jotta tuottavuus pystyttäisiin pitämään mahdollisimman korkeana ja vältyttäisiin ”loppuu palamisilta” (Agarwal ym., 2008).

XP:tä toteutettaessa on erittäin tärkeää, että yksi tai useampi asiakas on jatkuvasti mukana kehitystyössä. Tästä johtuu periaate *läsnäolevasta asiakkaasta* (engl. On-site customer). Asiakkaan tarkoituksena on auttaa ja opastaa kehitystiimiä vaatimusten osalta kirjoittamalla ja priorisoimalla niitä ja vastaamalla kehitystiimin kysymyksiin. Tarkoituksena on myös parantaa kommunikointia asiakkaan ja kehitystiimin välillä (Agarwal ym., 2008).

XP-projektissa kaikki jäsenet käyttävät *yhteisiä ohjelmointistandardeja.* Tarkoituksena on helpottaa pariohjelmointia ja koodin yhteistä omistajuutta. Ideana on, että kenenkään ei pitäisi kyetä erottamaan koodista, kuka kyseinen osion on työstänyt (Agarwal ym., 2008).

3.2.3. Kanban

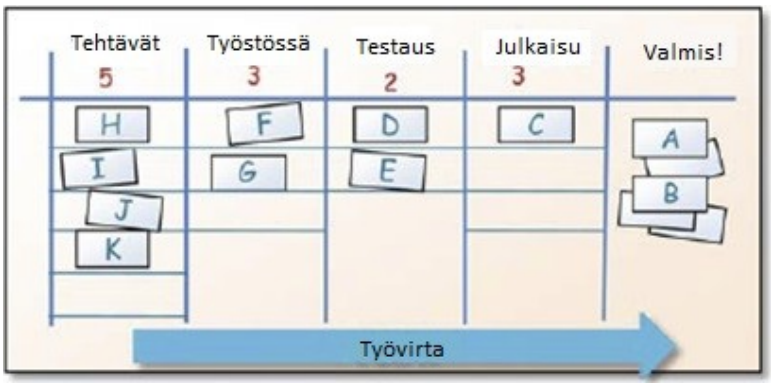
Kanban (Anderson, 2010; Kniberg & Skarin, 2009) on ketterään kehittämiseen kuuluva ajatusmalli, joka perustuu Lean-filosofiaan (Womack ym. 1990). Se ei tarjoa tukea projektinhallintaan eikä se ole varsinaisesti ohjelmistonkehitysmenetelmäkään (Anderson, 2010). Parhaiten Kanbania voidaan kuvata eräänlaiseksi työvirranhallintajärjestelmäksi. Kanban ei määrittele minkäänlaisia rooleja eikä esittele prosessia, jonka mukaisesti esimerkiksi ohjelmistokehitystä tulisi tehdä. Tässä suhteessa Kanban siis olettaa, että organisaatiossa on jo käytössä jokin prosessimalli, jonka yhteyteen Kanban voidaan sovittaa.

Kanban määrittää viisi keskeistä periaatetta ohjelmistokehitykseen (Anderson, 2010; Ahmad ym., 2013):

1. Visualisoi työvirta
2. Rajoita samanaikaisesti toteutuksessa oleva työmäärä (engl. Work-in-Progress, WIP)
3. Mittaa ja optimoi työvirta
4. Tee prosessikäytänteet selkeiksi
5. Käytä malleja parannusmahdollisuuksien tunnistamiseen

Työvirran visualisoinnilla tarkoitetaan sitä, että työtehtävät ja niiden tila on esitetty visuaalisesti esimerkiksi Kanban-taulua (ks. kuvio 2) hyväksi käyttäen. Kanban-taulu jaetaan sarakkeisiin, jotka osoittavat, missä tilassa kukin tehtävä

on sillä hetkellä. Sarakkeiden otsikot riippuvat täysin organisaation työtavoista. Yksinkertaisimmillaan työvirta voidaan jakaa esimerkiksi aloittamattomiin, työn alla oleviin ja valmiisiin tehtäviin (Kniberg & Skarin, 2009).



Kuvio 2: Esimerkki Kanban-taulusta

Jokaiseen taulun sarakkeeseen voidaan asettaa arvo kuvaamaan sitä, kuinka monta tehtävää kyseisessä vaiheessa voi yhtäaikaisesti olla. Tätä kutsutaan *meneillään olevan työn* rajoittamiseksi. Kanbanissa jokaisen tunnistetun työvaiheen tehtävien määrää rajoitetaan, jotta saataisiin estettyä pullonkaulojen syntyminen prosessiin. Sopivien tehtävien enimmäismäärää saraketta kohti on vaikea tietää etukäteen ja käytännössä määrä selviää vain kokeilemalla (Kniberg & Skarin, 2009).

Työvirran optimointia suoritetaan esimerkiksi *mittaamalla* jokaisen tehtävän läpikulkuaikaa. Läpikulkuajalla tarkoitetaan sitä aikaa, mikä tehtävällä kestää valmistua kulkemalla prosessin jokaisen vaiheen läpi. Läpikulkuajan mittaaminen alkaa tehtävän saapuessa Kanban-taulun vasemman puoleisimpaan sarakkeeseen, ja päättyy kun se saapuu oikean puoleisimpaan sarakkeeseen eli valmistuu. Mittaamalla läpikulkuaikaa saadaan tärkeää tietoa prosessin toimivuudesta. Sen perusteella voidaan muun muassa tehdä muutoksia sarakkeen työmäärän rajoituksiin vähentämällä tai nostamalla rajoitusta tilanteen mukaan. Tärkeää on tunnistaa mahdollisia pullonkauloja ja optimoida prosessia, jotta saavutettaisiin mahdollisimman lyhyt läpikulkuaika (Kniberg & Skarin, 2009).

Jotta kehitysprosessia voidaan parantaa, tulee kaikkien osapuolien ymmärtää täysin sen sisältö. Prosessikäytänteet tulee määritellä tarkasti ja kaikkien osapuolien täytyy kyetä tutustumaan niihin. Kun *prosessikäytänteet* on *tehty selkeiksi,* on kaikkien osapuolien helpompi ymmärtää kuinka toimia ja

toiminnasta ja sen kehittämisestä on paljon helpompi puhua jolloin toimintaa on helpompi kehittää positiiviseen suuntaan (Anderson, 2010).

Kanbanin yhteydessä *parannusmahdollisuuksia tunnistetaan* erilaisia *malleja* hyödyntäen. Malleja voidaan hakea useilta eri aloilta, kuten esimerkiksi sosiologiasta, psykologiasta tai vaikkapa riskien hallinnasta. Päätarkoituksena on oppia tunnistamaan ongelmakohtia ja poistaa ne. Usein käytettyjä malleja ovat esimerkiksi niin sanottu rajoiteteoria (engl. theory of constraints, TOC), systeemiajattelu sekä hukan (engl. waste) käsite (Anderson, 2010).

3.3. Ketterä ohjelmiston ylläpito

Ohjelmiston ylläpidolla tarkoitetaan ohjelmiston elinkaaren vaihetta, jolloin toimitetulle ohjelmistolle tarjotaan kustannustehokasta tukea. Se voi sisältää muun muassa virheiden korjausta, uusien ominaisuuksien lisäämistä tai ohjelmiston tehokkuuden, ylläpidettävyyden ja käytettävyyden parantamista (ISO/IEC, 2008; Choudhari & Suman, 2010). Ylläpito on välttämätön osa ohjelmiston elinkaarta. Se on jatkuva prosessi, jonka aikana ohjelmistoa parannellaan. Ylläpidon tarpeen laukaisevina tekijöinä voivat olla esimerkiksi teknologian kehittyminen, vaatimusten muuttuminen sekä sidosryhmien tiedon kasvu (Rajlich, 2014). Tässä alaluvussa kerrotaan ensin yleisesti ohjelmiston ylläpidosta. Sen jälkeen kuvataan, miten ohjelmiston ylläpitoa voidaan tehdä ketterästi.

3.3.1. Ohjelmiston ylläpidosta yleisesti

ISO/IEC 14765–standardissa (ISO/IEC, 2006) ylläpito jaetaan alalajeihin riippuen siitä tilanteesta, mikä laukaisee ylläpidon tarpeen. *Mukauttavaksi ylläpidoksi* (engl. adaptive maintenance) kutsutaan julkaisun jälkeistä ylläpitoa, jonka tarkoituksena on mukauttaa ohjelmisto toimivaksi muuttuneeseen tai muuttuvaan ympäristöön. *Korjaavaksi ylläpidoksi* (engl. corrective maintenance) määritellään ylläpito, jonka tarkoituksena on korjata ohjelmistosta julkaisun jälkeen löytyneet virheet. Korjaavaa ylläpitoa voi edeltää myös niin sanottu *kiireellinen ylläpito* (engl. emergency maintenance). Kiireellisen ylläpidon aikana ohjelmisto pidetään toimintakuntoisena niin kauan, kunnes varsinainen korjaava ylläpito voidaan aloittaa. *Estäväksi ylläpidoksi* (engl. preventive maintenance) ISO/IEC 14765–standardi määrittää ylläpidon, jossa ohjelmiston piilevät virheet tunnistetaan ja korjataan ennen kuin niistä kehittyy toiminnallisia virheitä. *Täydentävänä ylläpitona* (engl. perfective maintenance)

taas voidaan nähdä ylläpito, jonka tarkoituksena on tehdä parannuksia ohjelmaan. Tämä voi tarkoittaa esimerkiksi ohjelmiston tehokkuuden, ylläpidettävyyden ja käytettävyyden parantamista. Täydentävään ylläpitoon kuuluu myös uusien toiminnallisuuksien lisääminen.

Perinteisesti ohjelmistokehityksen ja ylläpidon suhde on voitu nähdä siten, että kaikki se työ mitä ohjelmiston parissa on tehty alustavan julkaisun jälkeen, on ollut ohjelmiston ylläpitoa (Rajlich & Bennett, 2000). Ohjelmistokehityksen ja ylläpidon välille voi kuitenkin olla vaikeaa vetää mitään tarkkaa viivaa. Hyvin usein käy niin, että ohjelmiston esikehitys on saatu loppuun ja ohjelmisto julkaistaan vasta sitten, kun se on käynyt läpi useita organisaation sisäisiä iteraatioita, joiden aikana ohjelmistosta poistetaan suurimmat puutteet ja varmistetaan sen vakaus. Tämä voidaan jossain määrin nähdä jo eräänlaisena ylläpitotyönä. Ohjelmisto voidaan myös julkaista useina eri versioina, kuten esimerkiksi alpha- ja beta-versioina (Rajlich & Bennett, 2000). Voidaan siis katsoa, että ohjelmistonkehityksen ja ylläpidon erottelu on jossain määrin häilyvää ja mielivaltaista. (Rajlich & Bennett, 2000).

Ylläpito voidaan nähdä myös palveluna. Niessin ja Vliet (2000) esittävät, että ohjelmistokehitys voidaan nähdä tuotteena ja ohjelmiston ylläpito palveluna tälle tuotteelle. Heidän näkemyksensä mukaan ohjelmiston ylläpito sisältää kaksi eri laatu-ulottuvuutta: tekninen laatu ja toiminnallinen laatu (Niessink ja Van Vliet, 2000). Teknisellä laadulla tarkoitetaan ylläpitopalvelun tuottamia tuloksia ja toiminnallisella laadulla taas sitä, kuinka palvelua toimitetaan. Ohjelmiston ylläpidon tuottama arvo nähdään ylläpitotehtävien asiakkaalle tuottamien hyötyjen kautta (Niessink ja Van Vliet, 2000). Nämä hyödyt voivat olla muun muassa virheiden korjauksia tai uusia ominaisuuksia. Ohjelmistokehityksen yhteydessä taas voidaan nähdä, että hyöty ilmenee itse tuotetun ohjelmiston kautta (Niessink ja Van Vliet, 2000).

Ylläpito voidaan jakaa ohjelmiston elinkaaren mukaan erilaisiin vaiheisiin. Kitchenham ym. (1999) ovat esittäneet, että ylläpito voidaan jakaa varhaislapsuuteen, murrosikään, aikuisuuteen ja seniiliyteen. Jokainen näistä vaiheista eroaa toisistaan pääasiallisesti siinä, mitkä ylläpitotehtävät ovat tärkeimpiä missäkin vaiheessa ja miten suuri käyttäjäkunta ohjelmistolla on. *Varhaislapsuus* (engl. infancy) on vaihe, joka alkaa yleisesti ottaen julkaisun jälkeen. Varhaislapsuuden aikana ohjelmiston käyttäjäkunta on varsin pientä, ja he tekevät yleensä ilmoituksia pienistä puutteista. Näin ollen pääasiallinen ylläpito onkin korjauksien tekeminen (Kitchenham ym., 1999). Murrosiässä (engl. adolescence) ohjelmiston käyttäjäkunta on kasvussa ja pääasiallisena ylläpitotoimena on vieläkin korjauksien tuottaminen. On kuitenkin mahdollista, että käyttäjäkunnan kasvusta johtuen ohjelmiston vaatimukset muuttuvat, joten ohjelmistoon voidaan myös joutua tekemään muutoksia (Kitchenham ym.,

1999). *Aikuisuuteen* (engl. adulthood) saavuttaessa ohjelmiston käyttäjäkunta on saavuttanut huippunsa ja ohjelmisto on suhteellisen vapaa mahdollisista puutteista. Jos kuitenkin suurin osa käyttäjäkunnasta niin toivoo, voidaan ohjelmistoon lisätä uusia toiminnallisuuksia. Lisäksi muutosten määrän kasvaessa ohjelmistoa voidaan myös uudistaa ja parannella sen koodirakennetta. Pääasiallisina ylläpidon toimenpiteinä ovat siis uusien vaatimusten tunnistaminen ja niiden tuottaminen (Kitchenham ym., 1999). Ohjelmiston ollessa seniiliys (engl. senility) -vaiheessa on markkinoilla jo uudempia vastaavia tuotteita ja ohjelmiston käyttäjäkunta vähenee tai mahdollisesti pysyy hyvin pienenä. Tämän vaiheen aikana ohjelmistolle tehdään pääasiassa vain korjaavia toimenpiteitä (Kitchenham ym., 1999).

Ylläpitoprosessi itsessään voidaan jakaa kuuteen vaiheeseen, prosessin suunnitteluun, ongelma ja muutos analyysiin, muutosten toteutukseen, ylläpidon katselmointiin, migraatioon ja eläköittämiseen. *Prosessin suunnittelu* (engl. process implementation) -vaiheessa ylläpidosta vastaava tekee vaaditut suunnitelmat ja muodostaa menettelyt, joita hyödynnetään itse ylläpitoprosessin aikana. Suotavaa olisi, että vaaditut suunnitelmat toteutettaisiin samanaikaisesti ohjelmiston kehityssuunnitelman kanssa. Ylläpitäjän tulisi myös luoda vaaditut kontaktit asianomaisten organisaatioiden välillä (ISO/IEC, 2006). *Ongelma- ja muutos-analyysin* (engl. problem and modification analysis) aikana ylläpitäjä analysoi saatuja ongelmaraportteja ja vahvistaa niissä kuvattujen ongelmien olemassa olon. Ylläpitäjä myös suunnittelee mahdolliset ratkaisut kyseisiin ongelmiin, dokumentoi vastaanotetut ongelmat ja niille suunnitellut ratkaisut ja hankkii hyväksynnän ehdotettujen muutoksien toteuttamiseksi (ISO/IEC, 2006). *Muutosten toteutus* (engl. modification implementation) -vaiheessa ylläpitäjä toteuttaa ja testaa valmiit muutokset, jotta uusia virheitä ei pääsisi syntymään (ISO/IEC, 2006). *Ylläpidon katselmoinnissa* (engl. maintenance review) varmistetaan, että tehdyt muutokset olivat oikeita ja lisäksi hankitaan hyväksyntä sille, että virheenkorjaukset on onnistuneesti suoritettu loppuun (ISO/IEC, 2006). *Migraatio* (engl. migration) ja *eläköittäminen* (engl. retirement) ovat ylläpitoprosessin viimeisiä vaiheita, ja ylläpitoprosessi etenee niistä toiseen tilanteesta riippuen. Migraatiovaiheeseen siirrytään silloin, kun järjestelmä tai ohjelmisto halutaan siirtää vanhasta toimintaympäristöstä uuteen. Jotta ohjelmisto saataisiin siirrettyä onnistuneesti, tulee ylläpitäjän suunnitella migraation vaatimat toimenpiteet, toteuttaa ne ja dokumentoida vaaditut toimenpiteet. Eläköittämisvaiheeseen edetään yleisesti ottaen silloin, kun ohjelmisto saavuttaa hyödyllisyyselinkaarensa lopun. Tässä vaiheessa yleensä arvioidaan onko viisasta jatkaa vanhentuneen teknologian käyttöä vai onko kustannustehokkaampaa siirtyä käyttämään uudempaa teknologiaa. Niissä

tapauksissa joissa ohjelmisto halutaan eläköittää lopullisesti, tulee ylläpitäjän tuottaa toimintasuunnitelma sille, kuinka eläköittäminen toteutettaisiin (ISO/IEC, 2006).

3.3.2. Ohjelmiston ylläpidon ongelmat

Ohjelmiston ylläpidossa löytyy useita ongelmia, jotka voidaan kategorisoida karkealla tasolla ulkoisiksi ongelmiksi ja sisäisiksi ongelmiksi. *Ulkoiset ongelmat* viittaavat asiakkaan näkökulmasta koettaviin ongelmiin, ja *sisäisillä ongelmilla* tarkoitetaan ohjelmistonkehittäjien sekä projektin johtajien kokemia ongelmia (April ja Abran, 2012).

SWEBOK (2014) kategorisoi ongelmat tarkemmin neljään luokkaan: tekniset ongelmat, hallinnolliset ongelmat, ylläpitokustannusongelmat sekä ylläpidon mittausongelmat. Dekleva (1992) puolestaan jakaa ohjelmiston ylläpito-ongelmat neljään hieman erilaiseen kategoriaan, jotka pohjautuvat ylläpitäjän näkökulmaan: ylläpidon hallinta, organisaationaalinen ympäristö, henkilöstö tekijät ja järjestelmän ominaisuudet.

Tekniset ongelmat voivat viitata esimerkiksi ylläpitäjien rajoitettuun käsitykseen ylläpidettävästä ohjelmistosta (SWEBOK, 2014) tai kunnollisen koulutuksen puutteeseen (Dekleva, 1992). Teknisiin ongelmiin liittyy myös haasteet testauksessa, kuten ongelmat regressiotestauksessa tai testaamisen vaatiman ajan löytäminen (SWEBOK, 2014). Lisäksi riittämättömät testausmenetelmät, vaikeudet vaikutusanalyysien toteutuksessa sekä ohjelmiston ylläpidettävyys ovat erittäin haasteellisia aiheita (Dekleva, 1992). Dekleva (1992) kuvaa myös muita teknisiä ongelmia kuten muun muassa monimutkainen ja huonorakenteinen koodi, vajaa tai olematon järjestelmädokumentaatio, yhteensopimattomien järjestelmien integrointi sekä vanhentuneet teknologiat.

Ohjelmiston ylläpito sisältää myös paljon *hallinnollisia ongelmia.* April ja Abran (2012) toteavat hallinnollisten ongelmien olevan yleisimpiä ongelmia ylläpidon parissa. Yksi hallinnollisista ongelmista on sovittaa organisaatiolliset tavoitteet siten, että ylläpitoon tehdyt sijoitukset saadaan tuottamaan arvoa (SWEBOK, 2014). Tämän lisäksi voi olla ongelmia henkilöstön suhteen, kuten ylläpitohenkilöstön hankkiminen ja motivoiminen. Myös kokeneen henkilöstön menetys, vähäisestä kunnioituksesta ja tunnustuksen puutteesta johtuva alhainen moraali sekä vähäinen hallinnollinen tuki nähdään ongelmiksi ylläpitäjän näkökulmasta (Dekleva, 1992). Ylläpitoprosessin aktiviteetit aiheuttavat hallinnollisia ongelmia (SWEBOK, 2014). Ylläpitomenetelmien, prosessistandardien ja työkalujen puute ovat isompia ongelmia ylläpidon

parissa (Dekleva, 1992). Muihin hallinnollisiin ongelmiin kuuluvat myös ulkoistukseen liittyvät haasteet (SWEBOK, 2014).

Kolmantena pääongelmana ovat *ylläpidon kustannusongelmat*. Ongelmia löytyy muun muassa kustannusten arvioinnissa, muuttujien mallinnuksessa sekä yleisesti kokemuksessa.

Ongelmat ylläpidon mittauksessa voidaan nimetä neljänneksi pääongelmaksi. Sopivat ja tarkat mittauskäytänteet tulee määritellä organisaatiokontekstin mukaisesti. Jokaisen mittauskäytänteen tulee myös ottaa huomioon muuttuvuus, analysoitavuus, vakaus ja testattavuus (SWEBOK, 2014).

Näiden ongelmien lisäksi löytyy myös *liiketoimintaan* liittyviä ongelmia. Nämä ongelmat liittyvät jatkuvasti muuttumaan liiketoimintaympäristöön, liiketoiminnallisten vaatimusten ymmärtämiseen ja niihin vastaamiseen sekä strategiseen suunnitteluun (Dekleva, 1992).

3.3.3. Ohjelmiston ylläpito ketterän lähestymistavan mukaisesti

Ketteriä menetelmiä on alettu hyödyntää ohjelmiston ylläpitotyössä 2000-luvun alusta alkaen. Ylläpito on joissakin tapauksissa myös sisällytetty oleellisena osana ketteriä menetelmiä (vrt. XP, Abrahamsson ym., 2002). Tämä yhteys ketterien menetelmien ja ylläpidon välillä juontuu siitä, että ketterä kehitystiimi tekee jatkuvia julkaisuja ja korjaa myös samalla yleisen kehitystyön aikana esille nousevia virheitä. Tämä lähestymistapa on erilainen verrattuna perinteisiin kehitysmenetelmiin ja ylläpitoon (Prochazka, 2011).

Ketterää ohjelmiston ylläpitoa koskevaa tutkimusta on tehty jonkin verran (Anderson ym., 2002, Choudhari ym., 2010, Jain, 2006. Kajko-Mattsson, 2008, Prochazka, 2011). Anderson ym. (2012) ovat tutkineet simuloimalla, kuinka Kanban ja Scrum mahdollisesti parantaisivat PSP/TSP-pohjaista ylläpitoprosessimallia, ja ovat soveltaneet Kanbania tapaustutkimuksessa tutkimuksen kohteena toimineeseen alkuperäisprosessiin. Choudhari ym. (2010) puolestaan ovat esittäneet, kuinka XP:hen pohjautuva iteratiivinen ylläpitoprosessi voisi auttaa ylläpidon parissa ilmaantuvien ongelmien ehkäisyssä. Tämän lisäksi Choudhari ym. (2010) ovat esittäneet mallin tälle XP:hen pohjautuvalle iteratiiviselle ylläpitoprosessille. Ketterien menetelmien hyödyntämistä ulkomaille ulkoistetussa ylläpidossa on tutkinut muun muassa Jain (2006). Jain (2006) on kirjoittanut aiheesta raportin pohjautuen hänen omiin kokemuksiinsa viisitoista kuukautta kestäneessä ulkoistetussa ylläpitoprojektissa. Kajko-Mattsson (2008) on puolestaan selvittänyt, ilmaantuvatko perinteisen ylläpidon parissa yleisesti esiintyvät ongelmat myös ketterän lähestymistavan mukaisesti ylläpitoa harjoittavissa organisaatioissa.

Prochazka (2011) on tutkinut perinteisen prosessiorientoituneen ylläpidon ongelmia ja kuvannut, kuinka kyseisiä ongelmia voitaisiin ehkäistä hänen omalla ketteriin periaatteisiin ja Lean-ajattelutapaan perustuvalla ylläpito lähestymistavalla.

Seuraavaksi kuvataan tarkemmin ketterän ohjelmiston ylläpidon periaatteita, käytänteitä sekä annetaan kuvaus ketterän ylläpitoprosessin vaiheista.

Prochazka (2011) on esittänyt ketterälle ohjelmiston ylläpidolle neljä periaatetta, jotka on tarkoitettu laajentamaan Agile-manifestia (Prochazka, 2011). Nämä periaatteet ovat:

1. Enemmän kurinalaisuutta, vähemmän byrokratiaa
2. Tiimien välinen yhteistyö
3. Proaktiivisuus
4. Riskipainotteinen lähestyminen

Kurinalaisuuden kasvattamiseksi ja byrokratian vähentämiseksi on Prochazkan (2011) mukaan tärkeää määritellä selkeät perussäännöt ylläpitoryhmän kanssa. Hyvä olisi, ettei määritellä mitään tarkkoja tehtäviä, vaan jätetään tilaa luovuudelle. Hierarkiarakennetta tulisi myös tasoittaa vähentämällä päälliköiden määrää ja antamalla enemmän päätösvaltaa tiiminvetäjille. Lisäksi retrospektiivien pitäminen ja sääntöjen muuttaminen tarvittaessa on tärkeää. *Tiimien välisen yhteistyön* tärkeimpänä hyötynä on käyttäjien tarpeiden täyttyminen. Tärkeää on, että organisaation kaikki osapuolet ovat mukana ylläpidossa tavalla tai toisella. Informaation jakaminen esimerkiksi kehitystiimin ja markkinointitiimin välillä on hyvin tärkeää, jotta voidaan tehdä informaatioon perustuvia päätöksiä, jotka luovat mahdollisimman paljon markkina-arvoa (Prochazkan, 2011). *Proaktiivisuudella* puolestaan tarkoitetaan sitä, että analysoidaan ja perehdytään oma-aloitteisesti kehittyvään teknologiaan ilman asiakkaan pyyntöä. Oleellista olisi esittää asiakkaille markkinoihin ja IT-alan kehitykseen perustuvia uudistuksia ja ratkaisuja, jotka toisivat asiakkaille mahdollisuuksia lisätä markkina-arvoa. Proaktiivisuuden selkeimpänä hyötynä on asiakastyytyväisyys, mikä voi johtaa esimerkiksi uuteen tilaukseen asiakkaan puolesta (Prochazkan, 2011). *Riskipainotteisen lähestymistavan* tarkoituksena on taas vähentää riskien vakavuutta. Lisäksi vältytään huomattavasti helpommin mahdollisilta myöhästymisiltä, taloudellisilta tappioilta ja laatuongelmilta.

Prochazka (2011) esittää myös neljätoista käytännettä ketterälle ohjelmiston ylläpidolle. Ne ovat: iteratiivinen lähestymistapa, pari-työskentely, vuorottelu, testauspainotteinen lähestymistapa, refaktorointi, päivittäiset tapaamiset, työssäoppiminen, liiketoimintaskenaarioiden simulointi, retrospektiivit, jatkuva integraatio, puolustava ohjelmointi, suunnittelupokeri,

visualisointi ja tulkinnanvaraisuuden poistaminen. Osa käytänteistä esiintyy yleisesti ketterässä ohjelmistokehityksessä, ja osa taasen tarjoaa tukea erityisesti ylläpitotyölle.

Iteratiivinen lähestymistavan ydin on ylläpidossa hyvin samankaltainen kuin ohjelmistokehityksessäkin. Tarkoituksena on hallinnoida ihmisresursseja sekä tehtävää työtä, julkaista usein, testata, esitellä ja arvioida tehtyä työtä ja parantaa prosessia kokemusten pohjalta (Prochazka, 2011). *Parityöskentely* toimii Prochazkan (2011) mukaan ylläpidossa samalla tavalla kuin XP:n yhteydessä. Työskennellään pareittain, mikä edesauttaa laadukkaamman koodin tuottamista. Tätä käytetään muun muassa kriittisten ongelmien ratkaisussa, ydinongelmien syiden selvittämisessä ja koodin laadun tarkastelussa (Prochazka, 2011). *Vuorottelulla* puolestaan tarkoitetaan kehitystiimin ja ylläpitotiimin jäsenten siirtymistä tiimistä toiseen tietyin väliajoin. Tällä edesautetaan tietämyksen leviämistä tiimien sisällä (Prochazka, 2011). *Testauspainotteisella lähestymistavalla* tarkoitetaan pääosin yksikkötestaamista. Kun korjataan virheitä, luodaan yksikkötestit korjauksen kohteena olevalle luokalle, pakkaukselle tai moduulille. Muutoksien yhteydessä tehtävien muutoksien toimivuus todennetaan yksikkötesteillä ja sama pätee myös täysin uusien ominaisuuksien lisäämisen kohdalla (Prochazka, 2011). *Refaktoroinnin* ydin on sama kuin ketterän ohjelmistokehityksen piirissä. Koodin rakennetta parannellaan inkrementaalisesti, jotta saavutettaisiin helposti ylläpidettävä ohjelmistorunko (Prochazka, 2011). *Päivittäisillä tapaamisilla* pyritään synkronoimaan tiimin toimintaa, jakamaan mielipiteitä ja tilannekatsauksia sekä tunnistamaan mahdollisia ongelmakohteita aikaisessa vaiheessa (Prochazka, 2011). *Työssäoppimisella* tarkoittaa käytännössä sitä, että tiiminjäsenet oppivat työnteon ohessa kasvattaen tietämystään nopeasti (Prochazka, 2011). *Liiketoimintaskenaarioiden simuloinnilla* pyritään olemaan valmiita mukauttamaan toimintaa alati kehittyvään IT-alaan sekä valmistautumaan tuottamaan kustannustehokkaita ratkaisuita uusien teknologioiden avulla (Prochazka, 2011). *Retrospektiiveillä* pyritään keräämään yhteen kaikki tietämys siitä, mitä on opittu ja kehittämään toimintaa opitun pohjalta (Prochazka, 2011). *Jatkuvan integraation* tavoitteena taas on edesauttaa jatkuvaa koodin kehittämistä ja näin tukea laadukkaan ohjelmiston tuottamista (Prochazka, 2011). *Puolustava ohjelmointi* tarkoittaa käytännössä sitä, että vältetään yleisimpiä virheitä, seurataan sovittuja käytänteitä ja tuotetaan helposti ymmärrettävää koodia ylläpidon helpottamiseksi (Prochazka, 2011). *Suunnittelupokerin* avulla on tarkoitus suunnitella jokaisen iteraation sisältöä tarkasti ja yhdessä tiiminjäsenten kanssa. Lisäksi tarkoituksena on pitää hauskuus mukana työnteossa (Prochazka, 2011). Tarkempaa kuvausta ei

suunnittelupokerille Prochazka (2011) anna, mutta hänen kuvauksen perusteella se muistuttaa paljon XP:n yhteydessä käytettyä suunnittelupeliä. *Visualisointi* on yleispätevä tekniikka ongelmien ja trendien löytämiseksi. Tähän tehtävään voidaan esimerkiksi käyttää Kanban-taulua (Prochazka, 2011). *Tulkinnanvaraisuuden poistaminen* tarkoittaa sitä, että hyödynnetään eri tekniikoita ja käytänteitä kommunikoinnin helpottamiseksi (Prochazka, 2011).

Choudharin ym. (2010) esittämä ketterä ylläpitoprosessi koostuu seuraavista vaiheista: analyysi, suunnittelu, rakenteen muutos, muutosten toteutus, järjestelmän testaus, vastaanottotarkastus ja toimitus.

Analyysivaiheessa (engl. analysis) on tarkoituksena analysoida käyttäjätarinamuotoiset, käyttäjiltä saadut virheraportit sekä muutospyynnöt. Analyysi suoritetaan kahdella tasolla, soveltuvuusanalyysissa (engl. feasibility analysis) ja yksityiskohtaisessa analyysissa (engl. detailed analysis). Soveltuvuusanalyysin tarkoituksena on tunnistaa korjausvaihtoehtoja sekä arvioida niiden vaikutuksia sekä kustannuksia. Yksityiskohtaisessa analyysissa taas analysoidaan muutoksen vaatimukset sekä luodaan testaus-strategia. Lisäksi analyysivaiheessa tehdään vaikutusanalyysi, jonka tarkoituksena on selvittää tarvittujen muutosten laajuus (Choudhari ym., 2010).

Suunnitteluvaiheessa luodaan julkaisusuunnitelma (engl. release plan), iteraatiosuunnitelma (engl. iteration plan) sekä suunnitellaan tarvittavat kokoontumiset. Julkaisusuunnittelun päätavoitteena on luoda lopullinen lista toteutettavista muutoksista ja korjauksista sekä määritellä ajankohta, jolloin kyseiset muutokset ja korjaukset on viimeistään suoritettu. Iteraatiosuunnitelmassa taas ilmaistaan se, millaisiin iteraatioihin toteutettavat muutokset ja korjaukset jaetaan (Choudhari ym., 2010).

Rakenteen muutos -vaiheessa (engl. change design) on tarkoituksena muuntaa ylläpidettävää järjestelmää toteutukseen valittujen rakenteellisten muutosten mukaisesti. Tässä vaiheessa muun muassa tunnistetaan ne moduulit, joihin muutokset vaikuttavat, muokataan dokumentaatiota vastaamaan muutoksia, luodaan testit uuden rakenteen testaamiseksi ja tunnistetaan tarvittavat regressiotestaukset.

Muutosten toteutus -vaiheessa toteutetaan käyttäjätarinoiden mukaiset muutokset. Muutosten toteutus sisältää muun muassa koodausta, yksikkötestaamista, muutetun koodin integrointia sekä integraatio- ja regressiotestausta. Muutokset toteutetaan XP:n käytänteiden mukaisesti. Oleellisia käytänteitä ovat muun muassa parikoodaus, koodausstandardit, testausvetoinen lähestymistapa ja asiakasyhteistyö.

Järjestelmän testaus -vaiheessa testataan koko muutosten alla ollut järjestelmä, jotta saataisiin varmuus siitä, että muutokset ja virheenkorjaukset

on toteutettu onnistuneesti. Lisäksi regressiotestauksen avulla tarkistetaan, ettei uusia ongelmia ole syntynyt tehtyjen muutosten johdosta.

Vastaanottotarkastuksessa luodaan testit sen varmistamiseksi, että järjestelmä tuottaa käyttäjätarinoiden mukaiset tulokset. Yleisesti toimitaan sitten, että asiakkaalta saadaan käyttäjätarina sekä sitä vastaava tuotos järjestelmältä ja näiden tietojen pohjalta ylläpito luo tarvittavat testit toimivuuden varmistamiseksi.

Toimitus-vaihe on se vaihe, jossa tehdyt muutokset ja korjaukset julkaistaan. Tämä vaihe voi myös sisältää ohjelmiston asennukset, käyttäjien kouluttamisen uusien ominaisuuksien tai muutosten osalta ja ohjelmiston varmuuskopion arkistoinnin.

3.3.4. Ketterien menetelmien hyödyt ylläpidossa

Ketterien menetelmien hyötyjä on tarkasteltu useissa ketterää ylläpitoa käsittelevissä tutkimuksissa. Esimerkiksi Seikola ym. (2011) ovat kirjoittaneet raportin Kanban lähestymistavan käyttöönottamisesta tuotteen ylläpito-organisaatiossa, ja esitelleet käyttöönotosta koituneita hyötyjä. Svensson ja Höst (2005) puolestaan ovat suorittaneet tapaustutkimuksen ketterän menetelmän käyttöön ottamisesta, ja raportoivat kohdeorganisaation kokemia hyötyjä menetelmän käyttöönotosta. Poole ym. (2001) raportoivat tutkimuksessaan omista kokemuksistaan ketterän menetelmän käyttöönotosta ja siitä seuranneista hyödyistä. Taulukkossa 1 on esitetty yhteenveto siitä, mitä hyötyjä ketterien menetelmien käytöstä ylläpidossa on edellä mainituissa tutkimuksissa koettu.

Taulukko 1: Ketterien menetelmien hyödyt

Ketterien menetelmien hyödyt	Tutkimus
Jatkuvan integraation kehittyminen	Seikola ym. 2011; Svensson ja Höst, 2005
Tuottavuuden parantuminen	Poole ym. 2001
Aikataulussa pysyminen	Poole ym. 2001
Julkaisujen mittakaavan määrittelyn parantuminen	Poole ym. 2001
Ylläpidettävän tuotteen vakauden kehittyminen	Poole ym. 2001
Ylläpidon helpottuminen	Poole ym. 2001
Muutoksiin vastaamisen helpottuminen	Poole ym. 2001
Kehityskäytänteiden parantuminen	Poole ym. 2001
Tiedon välittymisen parantuminen	Svensson ja Höst, 2005
Parantunut tietoisuus tyypillisistä projektinhallinnallisista ongelmista	Svensson ja Höst, 2005
Nopeammat julkaisut	Svensson ja Höst, 2005
Testauskäytänteiden kehittyminen	Svensson ja Höst, 2005
Koodausstandardien kehittyminen	Svensson ja Höst, 2005
Tiimiin liittyvät hyödyt	Seikola ym. 2011
Virheiden käsittelyn läpimenoajan parantuminen	Seikola ym. 2011
Työn näkyvyyden parantuminen	Seikola ym. 2011

Kaksi taulukossa esitellyistä tutkimuksista liittyen ketterien menetelmien hyötyihin ohjelmiston ylläpidossa keskittyy XP:n (Svensson ja Höst, 2005; Poole ym. 2001) ja yksi Kanbanin (Seikola ym. 2011) hyötyihin ylläpidossa. Poole ym. (2001) esittävät yhdeksi XP:n käyttöön liittyväksi hyödyksi tuottavuuden parantumisen. Tämän lisäksi aikatauluissa pysyminen koettiin yhdeksi hyödyksi XP:n käytöstä. Asiakkaan näkökulmasta XP:n käyttö vaikutti parantaneen ylläpidettävän tuotteen vakautta, ja ylläpitäjien näkökulmasta taas tuotetta oli helpompi ylläpitää. XP:n koettiin myös auttaneen muutoksiin vastaamisessa sekä parantaneen kehityskäytänteitä (Poole ym. 2001). Svenssonin ja Höstin (2005) tutkimuksessa tiedon välittymisen parantuminen todettiin yhdeksi XP:n hyödyistä. Tutkimuksessa todettiin myös, että ylläpitäjien tietoisuus tyypillisistä projektinhallinnallisista ongelmista parantui

ja lisäksi monien käytänteiden kuten testaamisen ja koodausstandardien koettiin kehittyneen (Svensson ja Höst, 2005).

Kanbanin käyttöön liittyvät parannukset sisälsivät muun muassa ylläpitotiimiin liittyviä hyötyjä kuten ryhmätyöskentelyn, vastuullisuuden ja yhteistyön kehittymisen. Muita Kanbanin käyttöön liittyneitä hyötyjä olivat virheiden käsittelyn läpimenoajan lyheneminen sekä työn näkyvyyden parantuminen.

Jatkuvan integraation kehittyminen puolestaan koettiin hyödyksi XP:n (Svensson ja Höst, 2005) sekä Kanbanin (Seikola ym. 2011) parissa.

3.4. Yhteenveto

Ketterää lähestymistapa juontaa juurensa Agile-manifestiin, jonka joukko ohjelmisto- ja menetelmäkehittäjiä julkaisi vuonna 2001 vastaesityksenä perinteisen ohjelmistokehityksen rajoittuneisuuteen. Agile-manifestin ytimenä toimivat neljä arvoa ja 12 periaatetta.

Ketteriksi menetelmiksi kutsutaan menetelmiä, joiden perustana ovat Agile-manifestissa esitetyt arvot ja periaatteet. Yleisesti ottaen nämä menetelmät toimivat kehyksenä sille, millaisiin vaiheisiin kehittämistyö jaetaan ja millaisia tehtäviä ne sisältävät. Keskeistä ketterässä kehittämisessä on inkrementaalisuus, vuorovaikuttaminen, yksinkertaisuus sekä mukautuvuus. Esimerkkejä ketteristä menetelmistä ovat muun muassa Scrum, Extreme Programming (XP), Feature Driven Development (FDD), Crystal Methods, Adaptive Software Development ja Dynamic Systems Development Model (DSDM). Yleisimpiä ketteriä menetelmiä ovat Scrum, Lean, Kanban, FDD sekä XP, joista suosituin on Scrum ja vähiten käytössä on XP. Näiden menetelmien lisäksi useat erilaiset yhdistelmämenetelmät, kuten Scumban ja Scrum/XP ovat suosittuja.

Ohjelmiston ylläpito on ohjelmiston elinkaaren vaihe, jolloin sille tarjotaan kustannustehokasta tukea, mikä voi sisältää muun muassa virheiden korjausta, uusien ominaisuuksien lisäämistä ja ohjelmiston tehokkuuden, ylläpidettävyyden ja käytettävyyden parantamista. Ylläpito voidaan jakaa useisiin eri ala-lajeihin riippuen siitä tilanteesta, mikä laukaisee ylläpidon tarpeen. Tämän mukaisesti voidaan tunnistaa mukauttava, korjaava, estävä ja täydentävä ylläpito. Ohjelmistokehityksen ja ylläpidon välille voi olla vaikea vetää selvää rajaa. Yksinkertaistettuna voidaan sanoa, että kaikki julkaisun jälkeinen työ on ylläpitoa. Ylläpito voidaan myös nähdä palveluna siten, että ohjelmistokehitys nähdään tuotteena ja ylläpito palveluna tälle tuotteelle.

Ketteryyttä ylläpidon parissa on alettu hyödyntää 2000-luvun alusta alkaen. Joissain tapauksissa ylläpito on myös oleellinen osa ketteriä menetelmiä. Ketterän ylläpidon hyötyihin kuuluvat muun muassa tuottavuuden parantuminen, aikatauluissa pysyminen, ylläpidettävän tuotteen vakauden kehittyminen, muutoksiin vastaamisen helpottuminen, tiedon välittymisen parantuminen ja nopeammat julkaisut.

4. KETTERÄ PELIKEHITYS

Edellä on käsitelty pelejä, pelaamista ja peliteollisuutta. Lisäksi on tutustuttu ketterään lähestymistapaan ja ketteriin menetelmiin kuten Scrumiin, XP:hen ja Kanbaniin. Tässä luvussa selvitetään, millaisia esityksiä löytyy ketterän lähestymistavan soveltamisesta pelikehitykseen. Aluksi käydään läpi Agile-arvoja ja niiden esiintymistä pelikehityksen parissa. Tämän jälkeen esitellään ketterän pelikehityksen vaiheita, menetelmiä, käytänteitä ja organisointia. Lopuksi tarkastellaan vielä pelien ketterää ylläpitoa.

4.1. Agile-arvot ja pelikehitys

Keith (2010) on tutkinut Agile-manifestissa (Agile Alliance, 2001) esitettyjen arvojen vastaavuutta pelikehityksen arvoihin. Tämän pohjalta Keith (2010) on esittänyt seuraavanlaisia päätelmiä ja mukaelmia.

Ensimmäinen arvo, ”Yksilöt ja vuorovaikutus ennen prosesseja ja työkaluja” (Agile Alliance, 2001), ei muutu ollenkaan. Pelikehitys vaatii panostusta usean eri alan osaajilta, mistä johtuen on tärkeää, että kommunikaatio toimii saumattomasti ja vaivattomasti. Erityisen tärkeää on, ettei kommunikaatiota hankaloiteta tarpeettomalla hierarkialla, mikä hyvinkin tavallisesti vallitsee yli 100 hengen pelistudiossa. Tarpeettoman hierarkian ongelmana on se, että se hidastaa kommunikaatiota, koska pienimmästäkin asiasta joudutaan kysymään hierarkiassa ylempänä olevalta. Ketterissä menetelmissä etuna on se, että tiimi valtuutetaan tekemään päätöksiä itse, jolloin vältetään hidastava hierarkia. Tällöin myös hierarkiassa ylempänä olevat voivat keskittyä yleisempiin asioihin ja kehitystiimi voi kasvaa vähän kerrallaan kantamaan yhä suurempaa vastuuta. On kuitenkin muistettava, että prosesseja

ja työkaluja tarvitaan tukemaan ketterää kehitystiimiä, mutta on kuitenkin arvokkaampaa, että yksilöt ratkovat ongelmia kollegojensa kanssa (Keith, 2010).

Toinen arvo pelialalle sovellettuna on hieman eri muotoinen: ”toimivaa peliä arvostetaan enemmän kuin kaiken kattavaa pelisuunnitteludokumenttia” (Keith, 2010, s. 27). Tämä muutos johtuu siitä, että peli on enemmän kuin pelkkä ohjelmisto (Keith, 2010). Pelisuunnitteludokumentti on kuitenkin syytä olla jossain muodossa, koska sitä tullaan tarvitsemaan eri sidosryhmiä varten. Julkaisijat, lisensoijat ja muut sidosryhmät haluavat saada selkeän kuvan projektin tavoitteista ja visiosta. Tämä on yksi esimerkki siitä, miksi pelisuunnitteludokumentti on hyvä olla olemassa jossain muodossa (Keith, 2010).

Pelikehityksen aikana pelistudion asiakkaana voidaan nähdä julkaisija, joka rahoittaa pelin (Keith, 2010). Kolmas arvo säilyy siis pelialalle sovellettuna muodossa: ”Asiakasyhteistyö ennen sopimusneuvotteluita” (Agile Alliance, 2001). Tämän arvon kohdalla on kuitenkin jonkin verran parantamisen varaa sen toteutumisessa. Yhtenä syynä tähän on se, että pelistudion ja julkaisijan väliset sopimukset ovat hyvin pitkään perustuneet etukäteen neuvoteltuihin ja sovittuihin vaatimuksiin eli virstanpylväisiin. Tavallinen käytäntö on ollut, että pelistudio saa lisää rahoitusta julkaisijalta, kun virstanpylväs on saavutettu. Tämä käytäntö on kuitenkin ongelmallinen. Etukäteen sovittujen virstanpylväiden takia pelistudiot eivät välttämättä pysty kehittämään peliä parempaan suuntaan. Toisaalta julkaisija ei voi myöskään muuttaa virstanpylvästä jälkikäteen, vaikka peli voisikin hyötyä muutoksesta. Kehittäjien kannalta uusien ominaisuuksien lisääminen voi tarkoittaa virstanpylvään ohittamista, jolloin kehittäjät eivät välttämättä saa palkkaansa, mikä luonnollisesti vähentää halukkuutta ottaa riskejä lisäämällä uusia ominaisuuksia. Ongelman ytimenä ovat siis tiukat sopimukset, mutta nykyään kuitenkin virstanpylväät määritellään kompromissina joustavimmiksi, mikä edesauttaa julkaisijan ja pelistudion välistä yhteistyötä edes jollain tasolla (Keith, 2010).

Neljäs arvo, ”Muutokseen vastaaminen ennen suunnitelman noudattamista” (Agile Alliance, 2010), säilyy myös samassa muodossa. Pelialalla teknologia kehittyy jatkuvasti ja olisi suotavaa, että lopputulos on innovatiivinen ja hauska peli, joten ei ole kovinkaan järkevää noudattaa etukäteen tehtyjä suunnitelma ja usein on niin, ettei niitä voitaisikaan noudattaa (Keith, 2010).

4.2. Ketterän pelikehityksen vaiheet

Luvussa 2 esiteltiin Mannisen ym. (2006) tunnistamat yleiset pelikehityksen vaiheet. Mannisen ym. (2006) esittämät vaiheet kattavat pelikehityksen elinkaaren konseptin luomisesta aina ylläpitoon ja sen päättämiseen saakka. Seuraavaksi kuvataan Keithin (2010) ja Musilin ym. (2010a) esittämät ketterän pelikehityksen vaiheet. Lopuksi niitä verrataan lyhyesti toisiinsa.

Kirjassaan Keith (2010) tunnistaa ketterälle pelikehitykselle Scrumiin pohjautuen neljä vaihetta: konsepti-, esituotanto-, tuotanto- ja jälkituotantovaihe. Vaiheet ovat jossain määrin päällekkäisiä, ja näin ollen vaiheiden rajat eivät ole täysin selkeitä. Esimerkiksi konseptivaihe läpäisee lähes koko kehitysajan (Keith, 2010).

Konseptivaihe tapahtuu ennen esituotantovaihetta. Konseptin kehittely on lähes tulkoon täysin iteratiivinen prosessi. Ideoita kehitellään ja mahdollisesti toteutetaan prototyyppeinä, ja niitä myös hylätään säännöllisin aikavälein. Tämä vaihe on yleisesti ottaen ajallisesti rajoitettu, ja tarkoituksena on tuottaa yksi tai useampi konsepti kehityssuunnitelmasta julkaisijan tai lisenssin haltijan hyväksyttäväksi. Ketterästä näkökulmasta voidaan ajatella, että konseptivaiheessa luodaan tietämystä kehitystiimille sekä sidosryhmille, eikä niinkään arvoa kuluttajille (Keith, 2010).

Esituotantovaiheessa hiotaan pelimekaniikkoja ja pyritään niin sanotusti löytämään pelin hauskuus sekä tuotetaan vaiheen tarpeiden mukaan mahdollisia kenttiä ja muita käytettäviä artefakteja. Tämä vaihe on täysin iteratiivinen ja inkrementaalinen. Kehitystiimi etsii iteratiivisesti pelin hauskuutta ja sopeuttaa kehityssuunnitelmaa inkrementaalisesti uuden tiedon puitteissa (Keith, 2010).

Tuotantovaiheessa tuotetaan pelin pelattava sisältö. Peli tuotetaan esituotantovaiheessa hiottujen ydinmekaniikkojen ja prosessien puitteissa. Tässä vaiheessa keskitytään tehokkuuteen ja inkrementaalisiin parannuksiin. Tärkeää on huomata, että tämän vaiheen aikana kehitystiimi keskittyy vähemmän iteroimaan ydinmekaniikkoja, koska tuotantovaiheessa luodaan hyvin paljon artefakteja ydinmekaniikkojen pohjalta. Tästä johtuen näiden mekaniikkojen muuttaminen tuotantovaiheessa on usein hyvin kallista ja epätaloudellista. Yhtenä esimerkkinä tästä Keith (2010) esittää pelihahmon hyppykorkeuden, minkä puitteissa esimerkiksi pelikentät joudutaan toteuttamaan. Jos hyppykorkeutta muutetaan sen jälkeen, kun on luotu sadoittain artefakteja aiemman hyppykorkeuden mukaisesti, aiheutuu näinkin yksinkertaisesta muutoksesta paljon ylimääräistä työtä, koska kaikkia

artefakteja joudutaan muuttamaan vastaamaan uutta hyppykorkeutta (Keith, 2010).

Jälkituotantovaiheessa pelin sisältö viimeistellään ja hiotaan julkaisukuntoon. Tässä vaiheessa tuotettavaa peliä parannellaan ja hiotaan inkrementaalisesti. Kun peli on vihdoin saatu tyydyttävälle tasolle, luovutetaan se laitteistotestaukseen (engl. hardware testing). Vaikka testaaminen on läsnä koko tuotantoprosessin ajan, ei kaikkea testaamista voida suorittaa kehitysprosessin aikana. Keith (2010) esittää tällaisesta tilanteesta esimerkkinä Microsoftin tai Sonyn laitteistotestaukset, jotka toteutetaan vasta muutamia kuukausia ennen pelin varsinaista julkaisua.

Musil ym. (2010a) esittävät Scrumiin pohjautuvan ketterän pelikehitysprosessin vaiheet. Musilin ym. (2010a) esittämät vaiheet ovat esituotanto (engl. pre-production), tuotanto (engl. production) ja projektin päättäminen (engl. project closure).

Musilin ym. (2010a) mukaan *esituotantovaiheessa* on tarkoituksena tunnistaa mahdolliset toteuttamiskelpoiset peli-ideat. Toteutettavat peli-ideat tunnistetaan loppukäyttäjä-orientoituneella prototyypittelyllä. Käytännössä siis luodaan prototyyppi mahdollisesta lopputuotteesta, joka sisältää samoja ominaisuuksia kuin tuleva lopullinen tuote. Lisäksi tehdään myös vaatimus- ja riskianalyysiä sekä hankitaan muita projektille olennaisia resursseja kuten rahoitus. Esituotantovaihe voi kestää viikoista jopa vuosiin ja siihen tulisi kulua noin 25% koko kehitysajasta (Musil ym., 2010a)

Tuotantovaiheessa tuotetaan itse peli. Tämän vaiheen aikana tuotannosta vastaavat osapuolet tuottavat esituotantovaiheessa määriteltyjen suuntaviivojen mukaisen pelin annettujen ajallisten, rahallisten ja laadullisten rajoitteiden puitteissa. Tämän vaiheen prosessi perustuu kokonaisvaltaisesti Scrumiin.

Projektin *päättämisvaiheessa* peli siirtyy jakeluun. Tässä vaiheessa mennyt projekti analysoidaan ja peliin liittyvät artefaktit kasataan pelin ylläpidon mahdollistavaksi paketiksi. (Musil ym., 2010a)

Musilin ym. (2010a) ja Keithin (2010) esittämät vaihejaot muistuttavat monilta osin toisiaan. Esimerkiksi Musilin ym. (2010a) esituotantovaihe on hyvin pitkälti yhdistelmä Keithin konseptin valmistelu- ja esituotantovaihetta. Erona on kuitenkin se, että Musil ym. (2010a) esittävät, että toteuttamiskelpoiset peli-ideat tunnistetaan prototyyppien avulla, kun taas Keithin esityksessä prototyyppejä ei välttämättä tehdä (Keith, 2010; Musil ym., 2010a). Tuotantovaiheiden osalta Musilin ym. (2010a) ja Keithin (2010) esitykset ovat samankaltaiset. Molemmissa esityksissä tuotantovaiheessa tarkoituksena on tuottaa itse peli esituotantovaiheessa määriteltyjen suuntaviivojen mukaisesti. Musilin ym. (2010a) esittämän tuotantovaiheen voidaan kuitenkin nähdä sisältävän osan tehtävistä, mitkä Keith (2010) on sisällyttänyt

jälkituotantovaiheeseen. Musilin ym. (2010a) päättämisvaihe ja Keithin (2010) esittämä jälkituotantovaihe taas eroavat toisistaan hyvinkin paljon. Musilin ym (2010a) päättämisvaiheessa keskitytään enemmän pelikehitysprosessin ulkopuolisiin asioihin kuten pelin jakeluun, menneen projektin analysoimiseen ja pelin ylläpidon järjestämiseen. Keithin (2010) esittämässä jälkituotantovaiheessa puolestaan huolehditaan vielä pelin hiomisesta ja testaamisesta (Keith, 2010; Musil ym., 2010a).

Goboy ja Barbosa (2010) ovat esittäneet XP:hen perustuvan mallin ketterästä pelikehityksestä. Goboy ja Barbosa (2010) esittävät ketterän pelikehityksen sisältävän esituotanto-, tuotanto- ja jälkituotantovaiheet. Vaihejako muistuttaa näiltä osin paljon Musilin ym. (2010a) esitystä.

Goboyn ja Barbosan (2010) mukaan *esituotantovaiheessa* on tarkoituksena ”löytää” peli. Pääasiallisesti vaiheessa hiotaan pelikonseptia, päätetään ohjelmointikielestä ja alustasta sekä muista määriteltävissä olevista seikoista. Käytännössä tämä vaihe määrittää, mihin suuntaan tuotantovaiheessa lähdetään etenemään. Tärkeää on löytää ideaalinen pelikonsepti, usein iteraatioiden kautta. Goboy ja Barbosa (2010) toteavat, että prototyypin luominen on hyvä keino tässä suhteessa. Tärkeää on kuitenkin huomata, että prototyyppi hylätään, kun se ei ole enää hyödyllinen (Goboy & Barbosa, 2010).

Tuotantovaiheessa pelikehitysprojektilla tulisi olla tarkoin määritelty käsitys (mittakaava), millainen peli lopulta tulee olemaan ja mitä tehtäviä täytyy tehdä (Goboy & Barbosa, 2010). Tässä vaiheessa suunnitteludokumentti tulee muuttaa tuotteen työlistaksi. Jokaisen iteraation alussa työlistan tärkeimmät tehtävät tulisi paloitella pienemmiksi osiksi ja tätä kautta määritellä iteraation tehtävät (Goboy & Barbosa, 2010).

Kun peli on saatu valmiiksi, siirrytään *jälkituotantovaiheeseen.* Tässä vaiheessa suoritetaan testaus, millä varmistetaan pelin laatu ja hauskuus. Lisäksi jälkituotantovaiheessa tehdään jälkiselvittelyraportti, joka pohjautuu koko kehitysprosessiin ja siitä kerättyyn palautteeseen (Goboy & Barbosa, 2010). Goboy ja Barbosa (2010) toteavat jälkiselvittelyraporttien olevan tärkeitä, koska niiden avulla pystytään arvioimaan kehitysprosessin vahvuuksia ja heikkouksia sekä näkemään, mitä ongelmia ilmaantui ja tätä kautta kehittämään toimintaa. Kerätty palaute auttaa myös tulevien projektien arvioinnissa sekä helpottaa tarvittavien muutoksien tekemistä tavoitteellisesti (Goboy & Barbosa, 2010).

4.3. Ketterien menetelmien käyttö pelikehityksen eri vaiheissa

Tässä alaluvussa tarkastellaa sitä, kuinka ketteriä menetelmiä hyödynnetään Keithin (2010) ja Musilin ym. (2010a) kuvausten mukaisissa pelikehitysprosesseissa. Kuvaukset perustuvat Keithin (2010) ja Musilin ym. (2010a) esityksiin.

Keithin (2010) esityksen mukaan *konseptivaiheessa* hyödynnetään Scrumia siten, että sprinttit pidetään mahdollisimman lyhyinä, työlistojen koot pieninä ja suurin osa tehtävistä aikarajataan (engl. time boxed). Aikarajattuja tehtäviä Keith (2010) kutsuu niin sanotuiksi piikeiksi (engl. spikes).

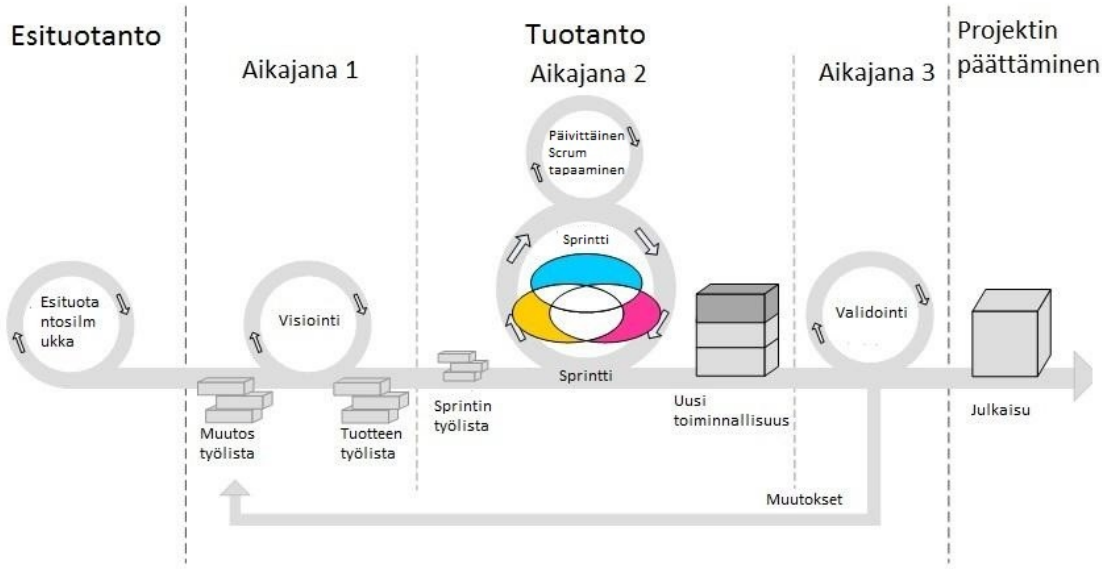
Esituotantovaiheessa Keith (2010) esittää, että Scrumia pystytään hyödyntämään puhtaimmillaan projektitiimin tuottaessa keskeisimpiä osia pelimekaniikoista, taiteesta ja ohjelmakoodista. Scrumin käytänteiden lisäksi ohjelmoijat hyödyntävät XP:n eri käytänteitä koko pelin kehityskaaren ajan. Jokaisen esituotantovaiheen sprintin aikana kehitystiimi tarkentaa arvioitaan tuotantovaiheeseen varattavasta ajasta, kun tieto tarvittavista artefakteista tarkentuu ja niiden tuottamiseen tarvittava tuotantolinja hioutuu (Keith, 2010).

Musilin ym. (2010a) esityksessä konseptivaihe on osana esituotantovaihetta joka on kuvattu iteraatiosilmukkana (kuvio 3). Tämän silmukan aikana pelin konseptia etsitään prototyyppien avulla, mutta Musil ym. (2010a) eivät suoranaisesti ehdota ketterien menetelmien hyödyntämistä esituotantovaiheen aikana.

Keithin (2010) kokemuksien mukaan Scrum ei vaikuta sopivan hyödynnettäväksi *tuotantovaiheen* tuotantolinjojen pääasialliseksi prosessiksi. Tämän sijaan Keith (2010) esittää, että Scrumin rinnalla tulisi hyödyntää Lean-ajatusmalleja ja Kanbania. Tuotantovaiheessa tuotantolinjatiimit hylkäävät sprintin työlistat ja siirtyvät niiden sijaan laajennettuun Kanban-tauluun. Lisäksi tiimit pyrkivät parantamaan prosessejaan Lean oppien mukaisesti. Kaikki Kanban-taulussa kuvatut tehtävät ovat aikarajattuja eli ne ovat piikkejä (Keith, 2010). Tuotantolinjan tiimit jatkavat kuitenkin toimimista muiden tiimien sprinttien tahdissa. Tuotantolinjan tiimit eivät luovu kaikista Scrumin käytänteistä vaan jatkavat joidenkin käytänteiden kuten päivittäisten Scrum-tapaamisten ja retrospektiivien hyödyntämistä. Lisäksi tuotantotiimit esittelevät aikaansaannoksensa sprinttien katselmoinneissa. Tuotantotiimit eivät kuitenkaan laadi sprinttien työlistaa, vaan nämä tiimit toimivat täysin Kanban-taulun mukaisesti. Kanban-taulun työtehtäviä työstetään, kunnes taulussa kuvattu tehtävälista on tyhjä, jonka jälkeen se täytetään uudestaan sovitulla määrällä uusia tehtäviä. Joissain tapauksissa voi olla niin, että tiimi ei ole puhtaasti tuotantotiimi. Näissä tapauksissa sprinttien aikana tuotantoon

osallistuville jäsenille käytetään Kanban-taulua. Tavallista sprintin työlistaa ja Kanban-taulua käytetään puolestaan muille jäsenille. Esimerkkinä tästä Keith (2010) esittää tiimikokoonpanon, joka sisältää ohjelmoijia ja kenttäsuunnittelijoita. Esimerkissä ohjelmoijat toimivat pääasiallisesti sprintin työlistan mukaisesti kun taas kenttäsuunnittelijat pääasiallisesti Kanban-taulun mukaisesti. Näissä tapauksissa sprintti tulee myös suunnitella ja sprintin työlista laatia niiden henkilöiden toimesta, joita ne koskevat (Keith, 2010).

Musil ym. (2010a) esittävät esituotanto- ja tuotantovaiheessa hyödynnettäväksi pelikehityksen tarpeisiin mukautettua Scrumia (kuvio 3). Tuotantovaihe perustuu kolmeen eri aikalinjaan, jotka ovat *visiointi* (engl. vision loop), *sprintti ja validointi* (engl. validation loop). Visiointia Musil ym. (2010a) vertaavat perinteisen Scrumin tuotteen omistajaan. Sen tarkoituksena on ylläpitää näkemystä siitä mihin suuntaan pelinkehitystä on tarkoitus viedä.



Kuvio 3: Pelikehitykseen mukautettu Scrum (mukaillen Musil ym. 2010a)

Sprintti on perinteisen Scrumin mukainen Scrum-sprintti, jonka tarkoituksena on tuottaa sprintin loppuun mennessä valmis lisäosa peliin (Musil ym., 2010a).

Validointi huolehtii pelin laadunhallinnasta. Vaikka testaamista tehdään myös sprintin aikana, ei yhteen sprinttiin kuitenkaan ole mahdollista sisällyttää kaikkia tarvittavia testejä, joten ne toteutetaan validointisilmukassa. Esimerkkejä näistä testeistä ovat muun muassa stressitestaus (engl. stress testing), tasapainotestaus (engl. balance testing), käyttäjätestaus (engl. user testing), kohderyhmäanalyysi (engl. focus group analyses), toiminnallisuustestaus (engl. functionality testing) ja lisenssin verifikaatio (engl.

license verification testing). Kun testit on saatu valmiiksi, ne suunnataan visiointiin tarkastelua ja vahvistusta varten (Musil ym., 2010a).

Jälkituotantovaiheessa prosessi muuttuu Keithin (2010) mukaan jälleen. Kun peli on artefaktien ja ominaisuuksiensa puolesta valmis, tiimit keskittyvät hiomiseen, säätämiseen ja virheiden korjaamiseen. Tämän vaiheen aikana tehtävät tunnistetaan lyhyemmällä aikavälillä kuin kokonaiset sprintit, ja tästä johtuen tehtävät lisätään päivittäiseen työlistaan sitä mukaa, kun ne havaitaan. Sprinttien katselmointi jatkuu edelleen, ja päämäärät määrittyvät päivittäisen työlistan mukaisesti (Keith, 2010). Musilin ym. (2010a) tuotantovaiheeseen esittämä prosessi jatkuu samankaltaisena myös jälkituotantovaiheessa.

4.4. Ketterä pelien ylläpito

Ketterästä pelien ylläpidosta ei ole tiedettävästi vielä julkaistu aiempaa tutkimusta. Ampatzoglou ym. (2010) tekemässä kirjallisuuskatsauksessa ei löytynyt pelien ketterää ylläpitoa käsittelevää tutkimusta. Ketterää pelikehitystä käsittelevässä kirjallisuudessa ylläpidosta on vain vähän mainintoja. Mannisen ym. (2006) tutkimuksessa ylläpito esiteltiin pelikehitysprosessin osaksi ja sen pohjalta voidaan todeta, että pelien ylläpito vaikuttaa sisältävän hyvin paljon samoja tehtäviä kuin ohjelmistojen ylläpitokin, kuten virheiden korjausta tai uusien ominaisuuksien lisäämistä (Manninen ym. 2006; Choudhari & Suman, 2010). Perinteisestä ylläpidosta poiketen pelien ylläpito kuitenkin vaikuttaa sisältävän myös uuden pelisisällön tuottamista, joko korjausten omaisesti tai mahdollisina lisäosina (Manninen ym. 2006). Keithin (2010) ketterää pelikehitystä käsittelevässä kirjassa ja Musilin ym. (2010a) esittämässä ketterässä pelikehitysprosessissa ylläpitoa ei mainita ollenkaan.

Tarkemman kirjallisuuden puuttuessa on vaikea määrittää, millaista pelien ylläpito tarkalleen ottaen on, kuinka ylläpitoprosessi jäsentyy suhteessa pelikehitykseen, millaisia ongelmia siihen liittyy tai toteutetaanko se ylipäätään ketterästi. Olemassa olevan kirjallisuuden pohjalta voidaan kuitenkin olettaa, että pelien ylläpito jäsentyy mahdollisesti kehitysprosessin ulkopuoliseksi prosessiksi (Manninen ym. 2006). Tähän suuntaan viittaisivat yhtäläisyydet perinteisen ylläpidon tehtävien (Choudhari & Suman, 2010; ISO/IEC, 2008) ja Mannisen ym. (2006) esittämien pelin ylläpito tehtävien välillä. Toinen mahdollinen vaihtoehto on, että pelien ylläpito nähdään osana pelikehitysprosessia. Ylläpitovaiheen puutos Keithin (2010) sekä Musilin ym. (2010a) pelikehitysprosesseissa viittaa tähän mahdollisuuteen. Tämän

mukaisesti pelikehityspiireissä ajateltaisiin siis, että kun peli on julkaistu, siirrytään pelin jatkokehittelyyn, mikä toimii prosessina samankaltaisesti kuin varsinainen pelikehitys.

4.5. Yhteenveto

Ketterä lähestymistapa on käytössä myös peliteollisuudessa. Pelikehityksessä osa Agile-arvoista voidaan mukauttaa, näkemyksestä riippuen, vastaamaan enemmän juuri pelikehityksen tarpeita. Muutokset johtuvat paljolti siitä, että pelin voidaan nähdä olevan enemmän kuin pelkkä ohjelmisto ja sidosryhmien väliset suhteet ovat erilaiset verrattuna perinteiseen ohjelmistokehitykseen.

Ketterälle pelikehitykselle ei ole olemassa yhtä yhtenäistä vaihejakoa. Yhden näkemyksen mukaan ketterään pelikehitykseen kuuluu konsepti-, esituotanto-, tuotanto- ja jälkituotantovaihe. Joissain tapauksissa konsepti- ja esituotantovaihe voidaan nähdä esituotantovaiheena, jälkituotantovaiheen tehtäviä on siirretty tuotantovaiheeseen ja jälkituotanto on korvattu projektin päättämisellä. Eroista huolimatta vaiheet muistuttavat sisällöllisesti ja päämäärällisesti toisiaan.

Ketterien menetelmien hyödyntäminen pelikehityksen eri vaiheissa vaihtelee sen mukaan, mitä menetelmää käytetään ja myös menetelmän esittäjän omien näkemyksien perusteella. Myös pelikehityksen vaihe vaikuttaa paljon siihen, millaisia ketteriä käytänteitä sovelletaan juuri kyseiseen vaiheeseen.

Pelien ketterästä ylläpidosta ei näytä olevan aiempaa tutkimusta. Tästä johtuen on vaikeaa sanoa, millaista pelien ketterä ylläpito tarkalleen ottaen on. Tämän aihealueen tutkimukselle näyttäisi olevan selvästi tarvetta.

5. HAASTATTELUTUTKIMUKSEN TOTEUTTAMINEN

Tämän tutkimuksen empiirisen osa tavoitteena on selvittää millaista on videopelien ylläpito, millaisia ongelmia siihen liittyy ja millaisia hyötyjä ja kokemuksia ketterien menetelmien käytöstä videopelien ylläpidossa on saatu suomalaisessa peliteollisuudessa. Tutkimuksella toivotaan saatavan esille uutta tietämystä siitä, kuinka videopelien ylläpito nähdään suhteessa pelikehitykseen ja miten ketteriä menetelmiä ja käytänteitä käytetään ylläpidossa. Tulosten toivotaan auttavan pelistudioita ylläpitoprosessin- ja käytänteiden parantamisessa. Seuraavissa alaluvuissa kuvataan ensin tutkimusmenetelmän valinta, haastateltavien valinta, tutkimusasetelma ja haastattelurunko.

5.1. Tutkimusmenetelmän valinta

Kuten aiemmin on todettu, ketterien menetelmien ja käytänteiden käyttöä ohjelmistojen ylläpidossa on tutkittu jonkin verran (esim. Anderson ym., 2002, Choudhari ym., 2010, Jain, 2006, Kajko-Mattsson, 2008, Prochazka, 2011), mutta pelien ylläpidossa tuskin lainkaan. Yhtenä syynä tähän voi olla se, että peliteollisuus eroaa tavallisesta ohjelmistoteollisuudesta monessa suhteessa. Tuotteita tehdään kuluttajamarkkinoille, jotka muuttuvat hyvin nopeasti ja usein ennakoimattomalla tavalla. Ensimmäiset peliversiot tulee saada julkaistua nopeasti. Pelejä laajennetaan ja monipuolistetaan myöhemmin. Suhtautuminen ylläpitoon voi tästä syystä olla erilainen. Tämä tekee tästä aiheesta tutkimuksellisesti kiinnostavan.

Ilmiötä, josta ei ole tarkempaa tietoa, kannattaa lähestyä kvalitatiivisen eli laadullisen tutkimuksen keinoin. Kirjallisuudessa todetaan, että laadullisen tutkimuksen avulla pyritään löytämään tai paljastamaan tosiasioita tutkimuskohteesta (Hirsjärvi, Remes & Sajavaara, 2007, 155). Tietoa laadulliseen tutkimukseen voidaan kerätä niin kyselyllä, haastattelulla havainnoinnilla kuin arkistomateriaalia käyttämällä (Järvinen & Järvinen, 2004).

Kyselytutkimuksessa (survey) käytetään määrämuotoista lomaketta tiedon keräämiseen ennalta määrätyltä otokselta (Kitchenham & Pfleeger, 2001). Tämän tyyppisellä menetelmällä on periaatteessa helppoa, edullista ja nopeaa kerätä tietoa laajalta joukolta. Jos otos on riittävän suuri ja edustava, päätelmät ovat yleistettäviä. Ongelmana voi kuitenkin olla matala vastausprosentti ja se, etteivät vastaajat välttämättä ymmärrä oikealla tavalla tutkijan esittämiä kysymyksiä. Kyselytutkimus ei anna myöskään mahdollisuutta esittää tarkentavia kysymyksiä, jolloin vastaustieto voi jäädä pinnalliseksi (Järvinen & Järvinen, 2004).

Tämän tutkimuksen yhteydessä haastattelu on luonnollinen valinta, koska kyseessä on vähän kartoitettu alue. Kirjallisuudessa haastattelun esitetään olevan oiva valinta juuri tällaisissä tilanteissa (Hirsjärvi ym., 2004; Hirsjärvi & Hurme, 2000). Toisena etuna on se, että haastattelujen yhteydessä voidaan säädellä aineiston keruuta joustavasti tilanteen ja vastaajien edellyttämällä tavalla (Hirsjärvi ym., 2004, Hirsjärvi & Hurme, 2000). Lisäksi haastatteluissa on enemmän mahdollisuuksia tulkita vastauksia (Hirsjärvi ym., 2004; Hirsjärvi & Hurme, 2000). Hirsjärvi ym. (2004) kuvaavat eduksi myös sen, että vastaajiksi suunnitellut henkilöt saadaan yleensä mukaan tutkimukseen. Haastateltavat on myös mahdollista tavoittaa myöhemminkin, jos on tarpeen täydentää aineistoa tai jos halutaan tehdä seurantatutkimusta (Hirsjärvi ym., 2004).

Haastatteluissa on myös ongelmia. Hirsjärvi ym. (2004) toteavat, että yli puolen tunnin haastatteluihin tuskin kannattaa ryhtyä. Lisäksi jos tutkittavana oleva ongelma on helppo ratkaista, on ehkä parempi turvautua kyselylomakkeeseen (Hirsjärvi ym., 2007). Haastattelut on myös suunniteltava erittäin huolellisesti, ja haastattelijan rooliin ja tehtäviin tulee perehtyä erittäin tarkasti (Hirsjärvi ym., 2004). Haastatteluissa voi ilmetä myös erinäisiä virhelähteitä. Haastateltava voi kokea haastattelutilanteen uhkaavaksi tai pelottavaksi. Tämän lisäksi haastattelun luotettavuutta heikentää se, että haastateltava saattaa antaa sosiaalisesti suotavia vastauksia (Hirsjärvi ym., 2004). Yhdeksi heikkoudeksi voidaan myös nähdä halukkaiden haastateltavien löytäminen.

Haastattelut voidaan kohdistaa yhteen organisaatioon tai useampaan. Jos kohteeksi otetaan yksi organisaatio (yritys, projekti tai vastaava), tutkimus voidaan tehdä tapaustutkimuksena. Tapaustutkimuksella pyritään löytämään

syvällisempää tietoa ja ymmärrystä tutkittavasta kohteesta. Tällöin pyrkimyksenä on usein ilmiön kuvaamisen ja ymmärtämisen lisäksi toiminnan kehittäminen ko. organisaatiossa (Yin, 2009; Runeson & Höst, 2009). Tämän tutkimusmenetelmän heikkoutena on se, että kerätyt tiedot ja niistä tehtävät päätelmät koskevat vain yhtä organisaatiota, eivätkä ole millään muotoa yleistettävissä.

Tässä tutkimuksessa pyritään löytämään tietoa useamman kaltaisesta peliyrityksestä. Tästä syystä haastateltavat valitaan useammasta yrityksestä. Pro gradu –työ ei laajuutensa rajallisuuden vuoksi anna mahdollisuutta toteuttaa tutkimusta monitapaustutkimuksena (multi-case study) (Yin, 2009), josta syystä on päädytty valitsemaan kustakin yrityksestä yksi haastateltava.

5.2. Haastateltavien valinta

Tutkimuksen perusjoukkoon kuuluvat suomalaiset pelistudiot, joita on arvioiden mukaan n. 200. Pelistudioiden koko vaihtelee kansainvälisessä mittakaavassa pienestä keskisuureen. Suomen suurimpiin pelistudioihin voidaan laskea Remedy Entertainment, joka työllistää noin 132 työntekijää (Wikipedia, 2015a), Rovio Entertainment, joka työllistää noin 800 työntekijää (Wikipedia, 2015b), Supercell, 152 työntekijää (Wikipedia, 2015c) ja Redlynx, 110 työntekijää (Wikipedia, 2015d). Nämä ovat kuitenkin verraten pieniä, kun niitä verrataan pelialan isoimpiin tekijöihin kuten Activision Blizzardiin, joka työllistää noin 6690 työntekijää (Wikipedia, 2015e). Suomen pelialan voidaan sanoa myös olevan monenkirjava joukko, sillä Suomesta löytyy pelistudioita, jotka kehittävät pelejä niin PC:lle, konsoleille kuin mobiililaitteillekin (Kuittinen ym., 2010).

Laadullisen tutkimuksen ominaispiirteisiin kuuluu, että kohdejoukko valitaan tarkoituksen mukaisesti (Hirsjärvi ym., 2004). Jotta kohdejoukosta saataisiin mahdollisimman kattava, tutkija lähetti sähköpostia parille pelistudioille henkilökohtaisten kontaktien kautta ja tämän lisäksi otti yhteyttä pelialan kattojärjestön, Suomen Pelikehittäjät ry (http://www.pelinkehittajat.fi/) jäseniin. Näitä kanavia myöten haastattelupyyntö välitettiin neljääntoista peliyritykseen. Tämän lisäksi tutkimukseen osallistuvia pelistudioita tavoiteltiin Keski-Suomen alueella toimivan peliosuuskunta EXPA:n (http://www.expa.fi/) kautta (vrt. Saate 1 liitteessä 1).

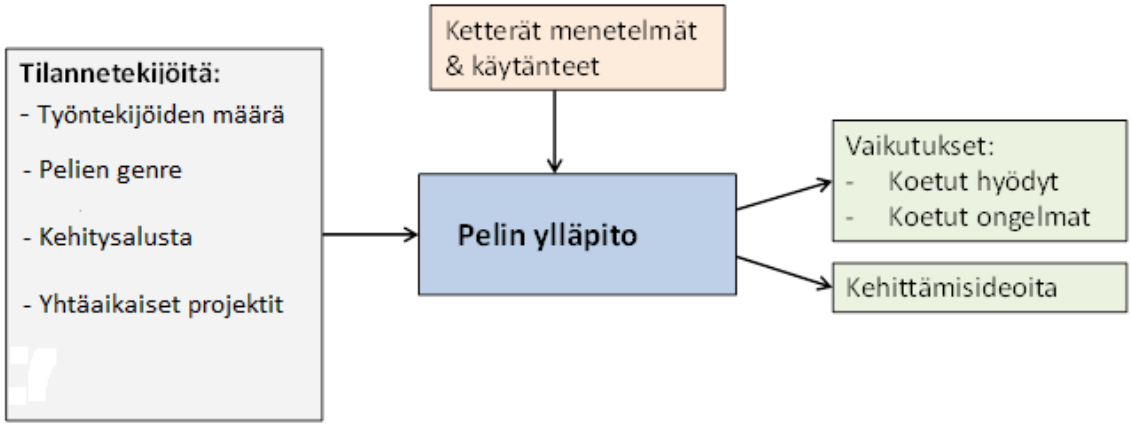
Edellä kuvattu haastattelupyyntökierros tuotti valitettavasti vain kaksi yhteydenottoa, joka nähtiin liian pieneksi. Tämän takia tutkija päätyi lähettämään saatteen 2 (Liite 2) mukaisen sähköpostin edellä mainituille

neljälletoista yritykselle. Toinen haastattelupyyntökierros tuotti kolme uutta yhteydenottoa, mikä nosti yhteydenottojen määrän viiteen. Näistä viidestä haastateltavaksi valittiin neljä yritystä perustuen yrityksien pääasialliseen kehitysalustaan. Tutkimukseen osallistui kaksi mobiilialustoille kehittävää yritystä, yksi konsolialustalle kehittävä ja yksi PC-alustalle kehittävä yritys. Alusta toimi pääkriteerinä valintaa tehtäessä, koska haluttiin nähdä alustan vaikutukset videopelien ylläpitoon. Yritysten koko oli myös yksi tekijöistä valintaa tehdessä, koska haluttiin nähdä yrityksen koon vaikutuksen videopelien ylläpitoon.

Pelistudioista haastateltaviksi pyrittiin saamaan sellainen henkilö, jonka työnkuva ja tietämys takasivat parhaat lähtökohdat haastatteluun osallistumiseksi.

5.3. Tutkimusasetelma ja haastattelurunko

Tämän tutkimuksen tutkimusasetelma on rakennettu kirjallisuuden pohjalta ja se perustuu pelialaa, pelikehitystä ja ylläpitoa koskeviin rakenteisiin ja jäsennyksiin. Se muistuttaa rakenteeltaan esimerkiksi Senepathin ja Srinivasanin (2011) tutkimusmallia. Asetelma koostuu viidestä osasta: taustatiedot eli tilannetekijät, pelin ylläpito, ketterät menetelmät ja käytänteet, ketterien menetelmien ja käytänteiden vaikutukset ja kehittämisideat (Kuvio 4). Ideana tutkimusasetelmassa on se, että sen mukaan voidaan tarkastella tietyssä kontekstissa pelien ylläpitoa, jossa käytetään tiettyjä ketteriä menetelmiä ja käytänteitä, sen selville saamiseksi, millaisia vaikutuksia (hyötyjä, ongelmia) siinä on havaittu ja mitä kehittämisideoita on löydettävissä.



Kuvio 4: Tutkimusasetelma

Haastattelurunko on rakennettu tutkimusasetelman rakenteen mukaisesti. Se koostuu viidestä pääkohdasta: taustatiedot, pelien ylläpito ja sen ongelmat, ketterät menetelmät ja käytänteet, vaikutukset ja kehittämisideat. Koska ylläpito voidaan ymmärtää peliteollisuudessa eri tavalla kuin perinteisten ohjelmistojen yhteydessä, on tehty kaksi vaihtoehtoista haastattelurunkoa, jotka ovat pääkohdiltaan samanrakenteisia. Ensimmäistä niistä on tarkoitettu käytettäväksi yrityksissä, joissa ylläpito nähdään perinteisellä tavalla (liite 3). Toinen runko on tarkoitettu käytettäväksi tilanteissa, joissa ylläpito nähdään saumattomana jatkumona pelikehitykselle (liite 4). Seuraavassa kuvataan tarkemmin haastattelurungon osia.

Taustatiedot-osion kysymysten tarkoituksena on selvittää pelistudion ominaisuuksia. Kiinnostuksen kohteina ovat pelistudion työntekijöiden määrä, kehitettävien pelien genre ja alusta ja samanaikaisesti meneillään olevien ylläpitoprojektien määrä. Kaikki nämä ovat oleellisia, koska niiden voidaan olettaa vaikuttavan pelistudion tapaan toteuttaa ylläpitoa. Esimerkiksi pelien genre ja kohdealustat vaikuttavat kehitystoimien haasteellisuuteen sekä tarvittavien artefaktien määrään ja on hyvin todennäköistä, että ne vaikuttavat näin ollen myös pelien ylläpitoon.

Pelien ylläpito-osiossa selvitetään tarkemmin, millainen pelin julkaisun jälkeinen ylläpitoprosessi on. Lisäksi tarkastellaan ylläpidon asemaa suhteessa pelikehitykseen sekä tilannetekijöiden vaikutusta ylläpitoon. Koska pelien ylläpidosta ei löydy juurikaan kirjallisuutta, lähtökohdaksi tässä osiossa on otettu ohjelmiston ylläpitoa käsittelevässä kirjallisuudessa (Kitchenham ym., 1999; Niessink ja Van Vliet, 2000; ISO/IEC, 2006) esitetyt prosessit ja käytännöt. Osiossa tarkastellaan myös pelien elinkaarta ja lähtökohdaksi on otettu kirjallisuudessa esitetty ohjelmiston elinkaari (Kitchenham ym., 1999). Osiossa perehdytään myös ongelmiin, joita pelien ylläpidossa mahdollisesti on. Lähtökohdaksi on otettu kirjallisuudessa esitetyt ohjelmiston ylläpidon ja pelikehityksen ongelmat. Tätä varten käytössä on ohjelmiston ylläpidon ongelmalista (liite 6) ja pelikehitys ongelmalista (liite 7).

Ketterät menetelmät ja käytänteet -osiossa pyritään selvittämään, missä määrin, jos ollenkaan, pelistudio hyödyntää ketteriä menetelmiä pelien ylläpidossa. Tätä varten on käytössä käytännelista (liite 5). Näiden lisäksi käydään läpi pelien ylläpidossa esiintyviä ongelmia ja peilataan niitä kirjallisuudessa esitettyihin ylläpidon ja pelikehityksen ongelmiin (April ja Abran, 2012; Dekleva, 1992; SWEBOK, 2014; Keith, 2010; Petrillo ym., 2009; Kanode & Haddad, 2009).

Vaikutukset-osiossa esitettyjen kysymysten on tarkoitus valottaa ketterien menetelmien vaikutusta pelien ylläpitoon. Osiossa tarkastellaan, mitä mahdollisia hyötyjä ja ongelmia ketterien menetelmien ja käytänteiden käytössä

ollaan mahdollisesti koettu. Pohjana osiolle käytetään pelien ylläpito-osiossa esiintulleita käytänteitä ja ylläpitoprosessia. Lisäksi osiossa tarkastellaan, kuinka pelikehityksessä ja ylläpidossa käynnissä oleva vaihe vaikuttaa ketterien menetelmien ja käytänteiden hyödyllisyyteen, haasteisiin ja ongelmiin.

Kehittämisideoita-osiossa pyritään selvittämään, tulisiko pelien ylläpitoa mahdollisesti lähestyä eri näkökulmasta ja kuinka pelien ylläpitoa voisi kehittää ja mikä rooli ketteryydellä olisi tässä. Tarkastelun kohteena on ylläpitoprosessi, toimijat sekä ketterät menetelmät ja käytänteet, jotka ovat tulleet esille haastattelun aiemmissa osissa.

Haastattelurunko luotiin iteratiivisen prosessin avulla. Tutkija loi ensimmäisen version rungosta, jota käytiin läpi yhdessä ohjaajan kanssa. Läpikäynnin jälkeen haastattelurunkoa muokattiin palautteen mukaisesti, jonka jälkeen uusi versio käytiin jälleen läpi ja sen nähtiin olevan sopiva testihaastattelua varten. Testihaastattelu suoritettiin ulkopuolisen, jyväskyläläisessä pelifirmassa työskentelevän, henkilön kanssa. Testihaastateltavan palautteen perusteella osaa kysymyksistä tarkennettiin, kysymysten järjestystä vaihdettiin johdattelun vähentämiseksi ja joitain hankalampia termejä varten määritelmät (Liite 8). Tämän jälkeen kysely käytiin vielä kerran läpi ohjaajan kanssa, jonka jälkeen haastattelut voitiin suorittaa. Haastattelut tehtiin kasvotusten 29.10.2015 ja 3.11.2015 välisenä aikana. Haastattelut kestivät keskimäärin noin 60 minuuttia.

Haastattelut nauhoitettiin älypuhelimella, ja haastattelun liitteenä olleet kyselylomakkeiden data oli tutkijan käytettävissä fyysisessä muodossa. Haastatteluissa esille tulleet asiat organisoitiin haastattelurungossa esiintyvien kysymyksien mukaisesti, ja tätä kautta lähdettiin hakemaan vastauksia tutkimuskysymyksiin. Osaa lomakkeiden tuloksista jouduttiin tarkastelemaan erityisen huolellisesti, sillä osa vastauksista sisälsi useampia valittuja vaihtoehtoja. Näiden osalta turvauduttiin litteroituihin haastatteluihin, joiden avulla pystyttiin tulkitsemaan syyt useampiin valittuihin vaihtoehtoihin.

6. TULOKSET

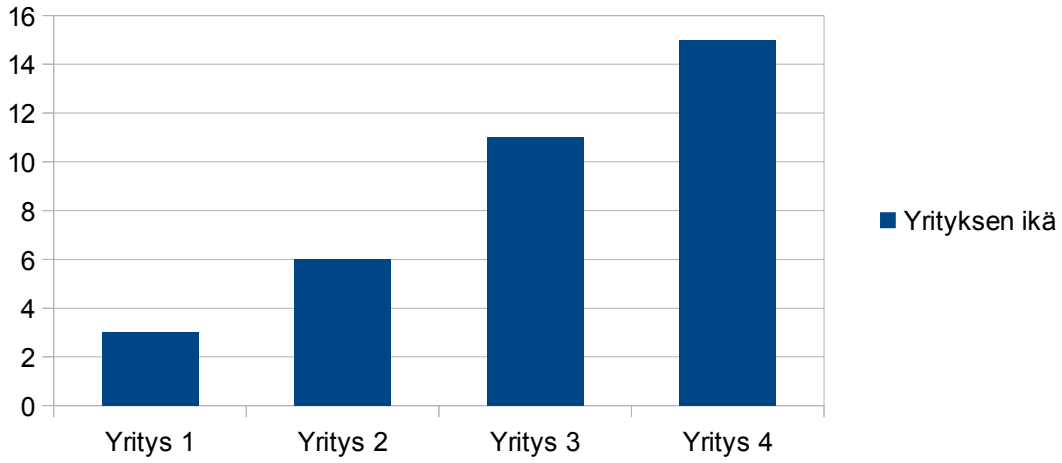
Tässä luvussa esitetään haastattelututkimuksen tulokset. Tulokset esitetään haastattelurungon mukaisessa järjestyksessä. Ensimmäinen alaluku käsittelee yrityksien taustatietoja, toinen pelien ylläpitoa, kolmas ylläpitoprosessia, neljäs ketterien menetelmien käyttöä pelikehityksessä, viides pelikehityksen ongelmia, kuudes ketterien menetelmien käytöstä koettuja hyötyjä ja haasteita ja viimeinen alaluku pelikehityksen kehittämisideoita.

Haastatteluun osallistui yhteensä neljä henkilöä, kukin eri videopelejä kehittävästä yrityksestä. Seuraavassa yrityksiin viitataan nimillä Yritys 1, Yritys 2, Yritys 3 ja Yritys 4.

6.1. Pelistudioiden taustatiedot

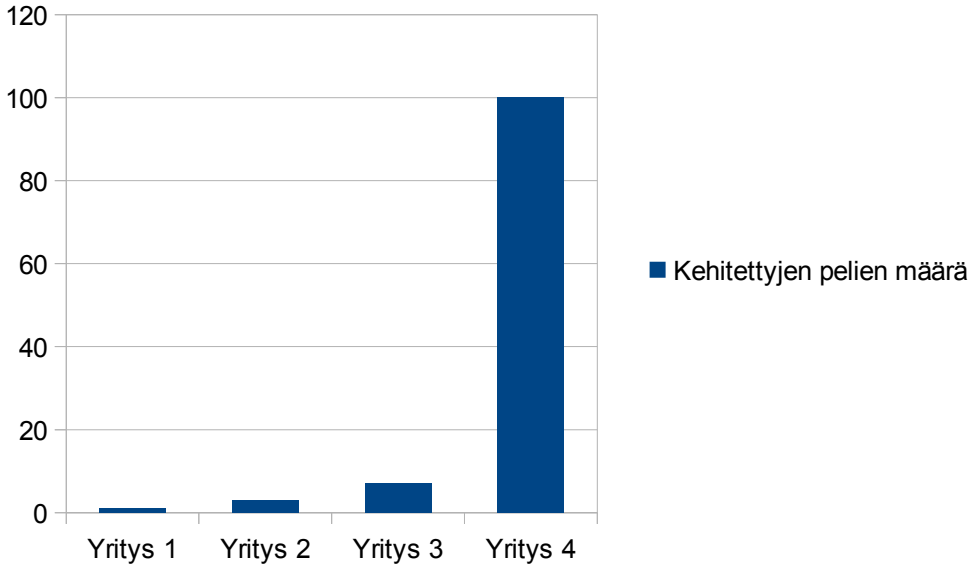
Pelistudion taustatietoina kysyttiin ikää, kehitettyjen pelien määrää, työntekijöiden määrää, yhtäaikaisten peliprojektien määrää, pelien laitealustaa ja genreä.

Tutkimuksen kohteena olleista pelistudioista kahta (Yritys 1, Yritys 2) voidaan pitää melko nuorina (3-6 vuotias) ja kahta (Yritys 3, Yritys 4) melko vanhoina (11-15 vuotias) (kuvio 5). Yrityksen 3 kohdalla on tärkeää huomioida, että yrityksen ydintiimi on ollut kasassa jo vuodesta 2004 alkaen, jolloin yritys toimi toisen yrityksen sisaryrityksenä, ja irtautui siitä vuonna 2007. Tämän perusteella voidaan ajatella, että Yritys 3:lla on 11 vuotta kokemusta yhteisestä tekemisestä.



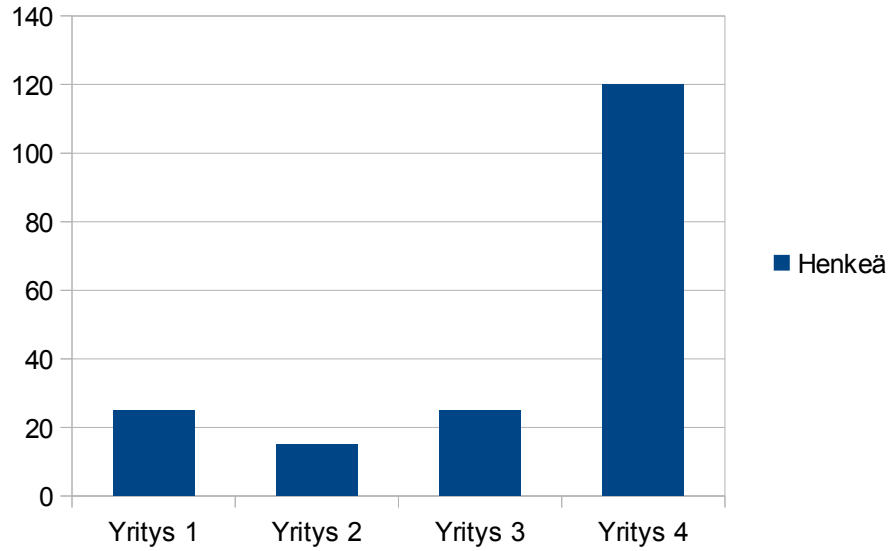
Kuvio 5: Yrityksien ikäjakauma

Yrityksissä kehitettyjen pelien määrä vaihteli myös paljon. Tutkimukseen osallistuneet nuorimmat yritykset (Yritys 1, Yritys 2) olivat kehittäneet keskimäärin 1-3 peliä ja vanhemmat (Yritys 3 ja Yritys 4) yritykset 7-100 peliä (Kuvio 6). Huomioitavaa on, että yhtä yritystä lukuun ottamatta yritykset mainitsivat kehittäneensä myös useita lisäpaketteja julkaisemiinsa peleihin, jotka määrittelystä riippuen voidaan myös laskea julkaisuiksi.



Kuvio 6: Kehitettyjen päätuotteiden määrä

Suurin osa tutkituista yrityksistä työllistää vähemmän kuin 30 henkeä (Kuvio 7). Yksi yrityksistä (Yritys 2) työllisti haastateltavan mukaan 15 henkeä, mutta hän mainitsi, että luku voi poiketa hieman ylös tai alas. Tässä yhteydessä oletetaan yrityksen työllistäneen 15 henkeä. Kaksi yrityksistä (Yritys 1, Yritys 3) ilmoitti työllistävän 25 henkeä. Yksi yritys (Yritys 4) ilmoitti työllistävänsä noin 120 henkeä, mutta täysin tarkkaa lukua haastateltava ei osannut ilmoittaa, joten tässä yhteydessä oletetaan yrityksen työllistäneen 120 henkeä.



Kuvio 7: Pelistudioiden työllistämien henkilöiden määrä

Yhtäaikaisia eri vaiheissa olevia kehitysprojekteja yrityksillä oli keskimäärin 2-4. Yksi yritys (Yritys 2) ilmoitti pyörittävänsä kahta samanaikaista projektia, yksi (Yritys 3) kolmea samanaikaista projektia ja kaksi yrityksistä (Yritys 1, Yritys 4) pyöritti neljää eri projektia.

Kehitettävien pelialustojen puolesta yrityksien välillä oli paljon vaihtelua. Yksi haastateltava (Yritys 2) ilmoitti yrityksen kehittävän PC-, Mac- ja Linux-alustoille. Kolme yrityksistä ilmoitti (Yritys 1, Yritys 3, Yritys 4) kehittävänsä mobiilialustoille kuten iOS ja Android. Yksi yrityksistä (Yritys 4) kehittää mobiilipelien ohella myös konsolipelejä keskittyen pääsääntöisesti mobiili- ja konsolialustoille. Yhden yrityksen (Yritys 3) kohdalla oli myös niin, että yritys oli aloittanut pelien kehittämisen PC-alustoille, mutta teki suunnan vaihdoksen vuonna 2013 mobiilialustoille.

Osion viimeinen kysymys koski pelien genrejä. Kahdessa yrityksessä (Yritys 1, Yritys 3) on linjattu keskittyttävän pääasiallisesti strategiapeleihin, ja toinen yrityksistä (Yritys 3) linjasi myös näiden ohella tekevänsä Action-pelejä.

Yksi yrityksistä (Yritys 2) keskittyy pelkästään simulaatio-peleihin. Yksi haastateltavista (Yritys 4) ilmoitti tämän hetkisiksi suuriksi genreiksi racing- ja platformer-pelit, mutta totesi, että tilanne voi tulevaisuudessa muuttua.

6.2. Videopelien ylläpito

Seuraavaksi tarkastellaan haastatteluosiota, jolla pyrittiin selvittämään näkemyksiä yritystensä videopelien ylläpidosta. Osion pääkysymyksenä toimi tutkimusongelma ”Millaista on videopelien ylläpito?”, joka on jaettu useampaan alakysymykseen.

Aluksi haastateltavilta kysyttiin, miten he näkivät ylläpidon aseman suhteessa pelikehitykseen. Kaikki haastateltavat linjasivat, että ylläpito tai vähintään erinäiset ylläpidolliset toimenpiteet, kuten virhekorjaukset, ovat tärkeä osa pelikehitystä. PC-puolen (Yritys 2) näkemys ylläpidon sisällöstä vastasi Mannisen ym. (2006) esittämää kuvausta ylläpitovaiheesta. Lisäksi jossain määrin alle viivattiin digijakelun osuutta pelituotteen helppoon päivittämiseen.

No se on ihanaa, että meillä on digijakelu, joten me pysytään päivittää helposti tuotteita ja vaikka tietysti tähdätään siihen että kun peli menee myyntiin, se olisi siinä kunnossa, että siellä ei ole bugeja, mutta ainahan siellä on jotain pientä. Et niitä me kyllä ehdottomasti halutaan fiksata, ja siihen on laskettu aikaa että sitä tehdään ja ihan vaan sitäkin, että käydään juttelemassa foorumeilla pelaajien kanssa, saadaan vähän selville, että mitä niillä on hätänä tai haluuko ne vaan kuulla, että heitä on huomioitu jotenkin. Et sekin on sitä ylläpitoa, ja mä ainakin ajattelisin, että siihen kuuluu se lisämatskun tekeminen. Et kyl meillä on ennen julkasua jo aina suunnitelma siitä, miten sitä jatketaan, eli minkälaisia paketteja tehdään. (Yritys 2)

Mobiilipuolella (Yritys 1, Yritys 3) esitettiin, että ylläpito on yhtä tärkeä osa pelikehitysprosessia kuin esimerkiksi tuotanto. Yhtenä syynä tällaiseen ajatteluun voidaan mahdollisesti nähdä näkemys pelistä enemmänkin palveluna kuin tuotteena. Tässä yhteydessä myös linjattiin (Yritys 1), että onnistuneessa palvelussa ylläpito on huomattavasti isompi ja vaikeampi osuus kuin itse tuottaminen.

No kyllä oikeastaan meidän tapauksessa me aatellaan tuo just silleen, et se on tavallaan eri vaihe siinä pelikehityksessä. Eli kehitysprosessi tavallaan jaetaan kolmeen eri osaan ja sitten

tää ylläpito ja liveops on niinkuin yksi osa sitä pelikehitystä. Eli sanotaan ylläpito on yhtä tärkeä kuin esimerkiksi prototyyppaus tai sitten itse tuotanto. (Yritys 3)

Meille se on tosi tärkeätä. koska me tehdään free-to-play-pelejä. Mikä sinänsä on olennaista ehkä noissa taustatiedoissa, et mehän tosiaan ei tehdä freemium-pelejä ollenkaan, eli pelit on palveluita ja palveluissa ylläpito on, no onnistuneessa palvelussa ylläpito huomattavasti isompi ja vaikeampi osuus kuin se tekeminen, ajallisesti, rahallisesti ja effortin määrällä, ja niin pois päin. Eli me valmistaudutaan jokasessa tuotteessa aika vakavasti siihen, että ylläpito tulee olemaan se leijona osa työstä. (Yritys 1)

Konsolipuolella (Yritys 4) ylläpitoon esitettiin näkemys palvelumallista, jossa idea oli jättää joitain peliominaisuuksia pois julkaisusta ja julkaista ne myöhemmin omina päivityksinään julkisuuden nostamiseksi ja pelin eliniän kasvattamiseksi. Huomioitavaa on myös, että haastateltava (Yritys 4) linjasi saman mallin olevan usein käytössä myös mobiilipuolella.

...niin siinä mentiin service modelilla, mikä tarkotti käytännössä vähän samaa kuin mikä on mobiilipuolella useinkin käytössä, että tarkoituksella jätetään featureja pois launchista ja tehdään ne launchin jälkeen omina päivityksinä, että saadaan lisää julkisuutta aina siitä ja porukat on tyytyväisiä ja peli pysyy pinnalla pidempään. (Yritys 4)

Konsolipuolen ylläpidon linjattiin myös poikkeavan jossain määrin esimerkiksi mobiili- ja PC-puoleen verrattuna. Vaikka digijulkaisut ovat myös mahdollisia konsolipuolella, haastateltava (Yritys 4) totesi, että pienempienkin virheiden korjaaminen on vaikeampaa konsolipuolella verrattuna PC- ja mobiilipuoleen. Syyksi esitettiin useat erilaiset sertifiointiprosessit, joista pelin ja mahdollisten päivitysten tulee päästä läpi. Ajallisesti nämä prosessit voivat haastateltavan mukaan kestää useitakin viikkoja pienempienkin korjauksien kohdalla, mikä tekee konsolipuolen päivityksien teosta huomattavasti raskaampaa.

joo kyllä se varmasti poikkeaa sen takia, että esimerkiksi konsolipuolella juuri on paljon isompi homma tehdä päivitys, että mobiilipuolella ja steamissakin ehkä pystyy korjaa yhden asian, jos tulee pieni moka, niin voi tehä hyvin halvalla uuden päivityksen ja korjata sen pienen mokan. PS4:lle on yksin 3 eri certtiä, tavallaan enemmänkin jos enemmän regioneita. Että se menee usemman 1st party certien läpi, mikä saattaa kestää viikosta kahdesta kolmeen ja sitten jos tulee joku blocker, niin että ei voida laittaa niin homma alkaa alusta. Siinä voi kestää, että päivityksen sisään saaminen voi kestää montakin viikkoa vaikka olisi kyse kuinka pienestä asiasta, niin se on raskaampaa tehdä päivityksiä konsolipuolella. Mobiilipuolella ja PC:llä se voi olla kevyempää vaikka mobiilipuolella on omat certit, mutta silti siellä pelintekijät on itse enemmän vastuussa siitä, että meneekö sinne rikkinäistä asiaa, kun konsolipuolella 1st partyt enemmän huolehtii siitä kokonaistoimivuudesta. (Yritys 4)

Vaikka pelialalla selkeästi nähdään ylläpidon olevan tärkeä osa pelikehitystä, ylläpito kuitenkin mielletään enemmänkin jatkokehittelynä, eikä niinkään ylläpitona kuten perinteinen ohjelmistokehitys sen ymmärtää. Mahdollisena syyllä tälle ajattelulle voi joissain tapauksissa olla häilyvä raja tuotannon ja julkaisun välillä. Varsinkin mobiilipuolella tuotiin esille, että kehityksen aikana tuotantoa pyritään lähestymään siitä näkökulmasta kuin peli olisi jo ulkona. Tämä tarkoittaa myös lopulta sitä, että itse pelikehitysprosessi ei muutu, vaikka peli olisikin ulkona. Kaikissa haastatelluissa yrityksissä tämä ei ole vielä todellisuutta, mutta vaikuttaa siltä, että tällaiseen tilanteeseen kuitenkin pyritään.

Me kokoajan pyritään enemmän ja enemmän sellaiseen prosessiin, että kehityksen aikana me käyttäydytään sitä peliä kohtaan kuin se olisi jo pihalla. On ne sitten testipelaajia firman sisällä tai muista firmoista tai mitä tahansa, mutta ikään kuin käyttäydytään kuin tämä olisi jo siellä, ja toteutetaan sitä jatkokehitysprosessia vaikka peli ei ole vielä ulkona. Jossain vaiheessa se vaan on ulkona, mutta prosessissa ei muutu mikään. (Yritys 1)

Toinen syy siihen, miksi ylläpito nähdään enemmän jatkokehittelynä, vaikuttaa johtuvan siitä, että esimerkiksi konsolipuolella ylläpidolliset tehtävät, ennen kaikkea virheiden korjaukset, suoritetaan juuri ennen julkaisua. Tarkoituksena on hioa peli siihen kuntoon, että se läpäisee mahdolliset sertifiontiprosessit. Konsolipuolella klassinen malli on haastateltavan (Yritys 4) mukaan se, että isommat virheet korjataan ja pienempien kohdalla voidaan toimia niin, että ne vain jätetään peliin. Julkaisun jälkeen keskitytään sitten enemmän juuri lisäominaisuuksien ja mahdollisen DLC:n (downloadable content, suom. ladattava sisältö) tekemiseen, kun suurin osa kehitystiimistä siirtyy jo uusien projektien pariin ja vain pieni osa alkuperäistiimistä jää työskentelemään vanhan projektin pariin. Puhtaasti ylläpidollisiksi toimenpiteiksi linjattiin erinäisiä palvelimien tai online-toiminnallisuuksien ylläpitämistä, joita tehtiin jossain määrin, mutta pääpaino kuitenkin oli juuri jatkokehittämisessä ja tarvittavien ylläpitotoimenpiteiden minimoimisessa.

Joo siis yleensä klassinen malli konsolipeleissä on, että peli tehdään, sitte jos siellä on isoja bugeja, niin ne korjataan mutta jos ei ole niin ne jää sinne. Että enempää ei ylläpitoa välttämättä ole, että ainoa mikä pelipuolella on yleensä tuo DLC tekeminen, koska tech porukka siirtyy jo tekemään seuraavaa projektia niin porukalla ei välttämättä ole mitään tekemistä, niin se on yksi syy minkä takia DLC:tä tehdään. Että porukka voi käyttää vanhoja työkaluja ja tehdä sen kanssa vielä lisäsisältöä. (Yritys 4)

Niin niin, että ei ole mitää järkee jos me tehtiin vuosi jatkokehitystä, niin ei olisi ollut mitään tolkkua pitää niin isoa porukkaa ylläpitämässä. Että jos ei ole tarkotus tehdä lisäsisältöä,

niin sitten kannattaa mahdollisimman nopeasti saada porukat pois ja mahdollisimman minimaalisella ylläpidolla tehtyä asiat. (Yritys 4)

Yhteenvetona voidaan todeta, että pelien ylläpito nähtiin tutkituissa yrityksissä liki saumattomana osana pelikehitystä. Tuotantoa voidaan lähestyä siitä näkökulmasta, että peli on jo ulkona, mikä tekee tuotannon ja julkaisun rajasta häilyvän. Lisäksi joissain tapauksissa ylläpidolliset tehtävät suoritetaan juuri ennen julkaisua, johtuen erinäisistä sertifiointivaatimuksista. Tästä syystä haastattelut etenivät jatkossa haastattelurungon B (liite 4) mukaisesti.

6.3. Pelien ylläpitoprosessi

Koska pelien ylläpito koettiin pikemminkin jatkokehittelynä, useimmissa yrityksissä pelikehitysprosessi jatkui samanlaisena pelin julkaisun jälkeenkin. Haastattelujen pohjalta vaikuttaa siltä, että pelikehitysprosessi itsessään jakaantuu yrityksissä pääpiirteittäin suunnittelu-, toteutus- ja julkaisuvaiheeseen, joiden sisällä toteutetaan yrityksen omaa näkemystä ketteryydestä. PC-puolella (Yritys 2) pelin julkaisun jälkeinen kehitysprosessi tapahtuu kahdessa eri linjassa. Toinen linja keskittyy pääasiallisesti niin sanottujen lisäpakettien, luultavasti jatko-osien, tuottamiseen, ja toinen linja keskittyy ilmaisten päivitysten ja virhekorjauksien tuottamiseen.

Ja meillähän on vähän niinkuuin 2 linjaa, että toisaalta tehdään niitä suunniteltuja lisäpaketteja ja sitten tehään toisella kaistalla sitten pätsejä, ilmaisia päivityksiä ja bugifikseja ja tämmöstä, että ne menee vähän niin kuin päällekkäisesti. (Yritys 2)

Mobiilipuolella (Yritys 3) oli havaittavissa samankaltainen lähestymistapa julkaisun jälkeiseen kehitysprosessiin. Käytännössä prosessi pyörii kahdessa linjassa siten, että on tuotantolinja, jossa tuotetaan uusia ominaisuuksia. Toisen linjan tarkoituksena on suorittaa virhekorjauksia. Käytännössä prosessi toimii siten, että aina kun peliin tahdottiin lisätä uusia ominaisuuksia, tuotantotiimi toteuttaa ne noudattaen yrityksen omaa näkemystä suunnittelu-, toteutus- ja julkaisuvaiheista. Ylläpitotiimi puolestaan keskittyy pääasiallisesti virheiden korjauksiin, mutta tiimiin kuuluu oleellisena osana myös markkinointipuoli. Tarkoituksena on kerätä dataa esimerkiksi pelaaja-käyttäytymisestä, kuten milloin pelaaja lopettaa pelin pelaamisen tai kuinka pelaajat suoriutuvat esimerkiksi opastuskentistä. Lisäksi kerätään dataa myös siitä, mitä pelistä mahdollisesti puuttuu pelaajien mielestä tai millaista sisältöä he peliin haluaisivat. Tärkeää on huomata se, että ylläpitotiimi itsessään ei tuota mitään

uutta sisältöä, vaan uuden sisällön tuottaa aina tuotantotiimi ylläpitotiimiltä saadun pelaajapalautteen perusteella.

Käytännössä meidän tapauksessa se tehdään siten, että meillä on tämmöinen liveops-tiimi, joka hoitaa bugifiksejä ja tietysti meidän markkinointiosasto on melkein isommassa roolissa kuin devaajat liveops tiimissä. Jos me halutaan jotain uutta kontenttia peliin, niin se käy itse tuotantotiimin läpi. Eli se lähtee sitten tavallaan alusta. (Yritys 3)

Se on enemmän tämmöstä analyyttistä duunia ja sitä kautta saadaan tuotantotiimille tarpeeksi tämmöisiä niinkuin, että tällästä pelaajat haluaa ja tällästä pelaajat olisivat odottamassa ja sitä kautta niinkuin päätöksentekoaineistoa kasataan analytiikan kautta. Liveopstiimi ei varsinaisesti, vaikka sielläkin on koodareita x määrä, niin se tosi, niinkuin, että esimerkiksi jos meillä on designer liveopstiimissä, niin hän ei varsinaisesti designaa mitään uutta, vaan balansoi peliä, analysoi pelaajan käyttäytymistä esimerkiksi miten tutoriaali viedään läpi, missä vaiheessa pelaajat alkaa putoamaan ja siis tämmöistä excelin pyörittämistä oikeestaan. (Yritys 3)

Myös konsolipuolella (Yritys 4) voitiin havaita kahden linjan jaottelu. Karkealla tasolla pelikehitysprosessi kulkee suunnittelu-, tuotanto- ja julkaisuvaiheiden kautta kuten PC- ja mobiilipuolella, mutta juuri ennen julkaisua prosessit vaikuttavat eroavan hiukan toisistaan. Konsolipuolella erinäiset sertifiointiprosessit vaikuttavat hyvin paljon siihen, kuinka julkaisuvaiheen aikana toimitaan. Pelikehitysprosessin alussa keskitytään perinteisesti peli-idean työstöön ja suunnitteluun. Tästä siirrytään sitten tuotantoon, jossa tarkoituksena on tuottaa toimiva versio pelistä. Loppua kohden kun aletaan siirtymään julkaisuvaiheeseen, pelikehitysprosessin sisältö alkaa muuttua sisällön ja ominaisuuksien tuottamisesta enemmän virheidenkorjaukseen keskittyväksi. Syynä tähän ovat juuri edellä mainitut sertifiointiprosessit, joista pelin tulee päästä läpi, ennen kuin se voidaan julkaista. Julkaisun jälkeen prosessi vaikuttaa jälleen jakaantuvan kahteen eri linjaan. Yhdessä linjassa jatketaan niin sanotusti pelin vakauttamista, eli poistetaan jäljelle jääneitä virheitä. Toisessa linjassa taas lähdetään toteuttamaan isompia ominaisuuksia samalla tyylillä kuin peliä itseään kehitettiin.

No jos siis puhutaan jatkokehittelystä, niin silloin kaikki muuttuu silleen varmistelevammaksi, että pitää aina olla tarkempi ettei oikeasti riko mitään olemassa olevaa, kaikki yhteensopivuus alaspäin pysyy aina voimassa ja siinä on kuitenkin tavallaan, ollaan hiukan siinä moodissa että on saatu julkaisu juuri ulos, niin meillä oli aikamoinen loppucrunch, missä korjattiin bugeja ja osa bugeista jää aina sitten seuraavaan updateen, niin siinä oli tietenkin jossain määrin ollaan vielä siinä vaiheessa. Sitten meillä pari isompaa featuree niin kuin esimerkiksi MP, jota tehtiin eri branchissa niin kuin pidemmän aikaisesti vähän samalla tyylillä kuin ennenkin.(Yritys 4)

Edellä kerrotun perusteella vaikuttaa siltä, että kahden linjan lähestymistapa on suosituin menetelmä pelien ylläpitoon niin PC-, mobiili- kuin konsolipuolella. Nuorimmassa haastatellussa yrityksessä (Yritys 1) kahden linjan lähestymistapa ei ollut vielä käytössä, mutta haastateltava totesi kuitenkin yrityksen harkitsevan ja todennäköisesti siirtyvän menettelyyn, jossa heillä olisi sekä ylläpitopuoli ja tuotantopuoli:

Se mitä me halutaan todennäkösesti, eli tämä asia on mitä me ei olla tehty, mutta mihin me kohta luultavasti tullaan menemään on se, että meillä on firman sisällä kehityspuoli ja ylläpitopuoli. (Yritys 1)

Pelinkehitysprosessi vaikuttaa pysyvän vakiona läpi koko kehitysajan. Siirryttäessä julkaisun jälkeiseen ajanjaksoon, voidaan havaita siirtymä nopea tempoisempaan ja varmistelevaan suuntaan. Prosessi itse pysyy siis samana, mutta kehityssyklit muuttuvat nopeammiksi ja kun kyse on uudesta tuotteesta, tulee ylläpitopuoli myös osaksi prosessia. Henkilöstötasolla muutokset vaikuttavat olevan hyvin yrityskohtaisia. Joissain tapauksissa henkilöstöä pyritään siirtämään välittömästi uusiin projekteihin, tai jos julkaistu peli on iso, voidaan henkilöstöä myös lisätä kyseiseen projektiin.

Elinkaarellisesti videopelit eroavat kirjallisuudessa esitetystä ohjelmiston elinkaaresta. Videopelien yhteydessä on tyypillistä, että pelin elinkaaren keskipisteenä on julkaisu, koska se on piste, jolloin pelin käyttäjäkunta saavuttaa maksiminsa. Tästä syystä haastateltavat kokivat, että pelien kohdalla julkaisu vastaa suunnilleen ohjelmiston elinkaaren aikuisuusvaihetta (Kitchenham ym., 1999). Jossain tapauksissa, riippuen yrityksen julkaisukäytänteistä, voitiin julkaisu nähdä myös murrosikävaiheena. Yleisesti ottaen lapsuus- ja murrosikävaiheita ei esiinny tai niiden voidaan ajatella kuuluvan osaksi pelikehitysprosessia itseään.

Minä luulen, että kun peli softlaunchataan, niin kuin me tehdään F2P-mobiilipelejä, missä on tapana tehä softlaunch-vaihe, niin tuota niin, ehkä sitä vois kutsua murrosiäksi, mutta aikuisuus on ehdottomasti se itse launch. (Yritys 3).

No näillä käsitteillä minä sanoisin, että me hypätään suoraan aikuisuuteen eli heti tuotteen launchista, siinä ei tämmösiä pienien korjauksien väliä ollenkaan, eli semmosta ei voi sanoa, että olisi elinkaaressa, vaan se on osana jokaista pientä sprinttiä tai milestonea. Eli suoraan tohon aikuisuuteen, missä katsotaan, mitä ihmiset tekee, mistä ne tykkää, koitetaan laittaa mahdollisimman paljon lisäarvoa tuotteelle ja kasvattaa kasvattaa kasvattaa. (Yritys 1)

Voidaan siis todeta, että olisi jonkin asteinen tarve luoda videopeleille oma malli kuvaamaan videopelien elinkaarta. Yksi näkemys, mikä nousi esille testihaastattelua toteuttaessa, oli, että jossain määrin lapsuus- ja

murrosikävaiheet voitaisiin nähdä pelin alpha- ja beta-vaiheina. Haastateltujen yrityksien keskuudessa koettiin, että tällainen näkemys voisi mahdollisesti toimia. Isompana ongelmana kuitenkin oli se, että lapsuus- ja murrosikävaiheen kuvaukset eivät vastanneet haastattelujen perusteella pelialan todellisuutta, mikä puoltaisi juuri pelialan todellisuuteen ja käytänteisiin perustuvan mallin luomista videopelien elinkaaren vaiheista.

Semmonen voisi olla ihan järkevä, mutta edelleen jos varhaislapsuus on semmonen, että tehdään vähän pieniä fiksailuja ja muuta, niin me itseasiassa suhtaudutaan sellaisena radikaalin innovaation aikana. Elikkä sanotaan, että meillä on se beta-vaihe ja siellä on niitä pieniä fiksejä kyllä, mutta tämä olettaa, että meidän visio siitä pelistä oli täydellinen, ja siinä ei ole mitään muutettavaa tai korjattavaa ylipäätään, mikä ei minun mielestä kuvaa pelialan todellisuutta.(Yritys 1)

Tärkeää on huomioida mahdollisen alustan, palvelumallin ja kustantajan vaikutus pelin elinkaaren pituuteen. Esimerkiksi kun puhutaan peleistä, jotka tarjotaan ikään kuin palveluina, on toivottavaa, että pelillä on niin sanotusti ”pitkä häntä”. Eli tarkoituksena on tarjota palvelua mahdollisimman pitkään ja sijoittaa siihen maksimiresurssit niin pitkään kuin kasvua on nähtävissä. Kun kasvu on laantunut ja palvelun suosio on stabiloitunut, aletaan huomiota siirtämään enemmän muihin projekteihin. Joissain tapauksissa kaikki resurssit on taas keskitetty juuri julkaisuun, ja julkaisun jälkeen huomio siirretään hyvinkin nopeasti uusiin projekteihin, koska on kustannustehokkaampaa alkaa tuottamaan uusia projekteja ja huolehtia minimiresurssein esimerkiksi siitä, että mahdolliset online-ominaisuudet toimivat.

Se miten me se on että näillä peleillä ja palveluilla on äärettömän pitkä longtail, eli siis ainakin meidän, kun mitä pelifirmat on ollut olemassa, niin todella todella pitkä longtail f2p-palveluilla. Jengi edelleen pelaa sitä farmvilleä ja muuta vastaavaa. Sitä mielellään ei haluta laittaa kokonaan pois, mutta jossain vaiheessa firman päähuomio siirtyy siitä pois. Eli jossain vaiheessa on nähtävissä, että tuote stabiloituu jollekkin tasolle, ja sen kehitysresurssit, jos siihen laitetaan enemmän resursseja sisään, niin siitä ei kuitenkaa saada enemmän ulos. (Yritys 1)

Toinen oli se mistä mainitsin tuossa, että usein se realiteetti on se, että koodaajat, koska suurin osa rahasta tehdään sillä versiolla, mikä menee pihalle, niin mitä järkeä on käyttää hirveesti paukkuja ylläpitoon muuta kuin, että homma, kaikki online-servicet ja muut toimii. (Yritys 4)

6.4. Ketterien menetelmien hyödyntäminen

Tämän osion tarkoituksena oli selvittää, missä määrin ketterää pelikehitystä käsittelevässä kirjallisuudessa esiintyneiden menetelmien käytänteet olivat käytössä haastatelluissa yrityksissä. Kirjallisuuden pohjalta suosituimpia menetelmiä ovat olleet Scrum, XP ja Kanban. Näiden pohjalta on laadittu käytännelista, joka esitettiin haastateltaville haastattelun yhteydessä. Käytännelistaan on kerätty kaikki Scrumin ja XP:n käytänteet. Kanbanin osalta on tyydytty ainoastaan työvirran visualisointiin, johon liittyen haastateltaville esitettiin kysymys. Käytännelistaan saadut vastaukset on kerätty alla olevaan taulukkoon 2. Numerot taulukossa osoittavat, kuinka moni yritys kutakin käytännettä missäkin määrin käyttää.

Taulukko 2: Ketterien käytänteiden hyödyntäminen yrityksissä

Käytänne	**Missä määrin käytössä 0 ei ollenkaan – 5 hyvin usein**					
	0	1	2	3	4	5
Tuotteen työlista	0	0	0	0	0	4
Sprintit	0	0	0	1	2	1
Sprintin työlista	0	0	0	2	1	1
Suunnittelutapaamiset	0	0	0	0	1	3
Katselmoinnit	1	0	0	0	2	1
Retrospektiivit	1	0	0	0	3	0
Suunnittelupeli	2	2	0	0	0	0
Pienet julkaisut	1	0	0	1	1	1
Vertauskuvat	0	1	0	0	2	1
Yksinkertainen suunnittelu	0	0	0	2	0	2
Testausvetoinen kehitys	1	0	0	0	1	2
Refaktorointi	0	0	1	2	1	0
Pariohjelmointi	1	0	1	2	0	0
Yhteinen omistajuus	0	0	0	0	2	2
Jatkuva integraatio	1	0	0	0	1	2
40-tuntinen työviikko	0	1	0	0	2	1

Haastattelun perusteella selvisi, ettei haastateltavissa yrityksissä yhdessäkään ollut käytössä mikään yksittäinen ketterä menetelmä sellaisenaan

kuin ne kirjallisuudessa esiintyvät. Yksi haastateltavista (Yritys 4) kertoi yrityksessä olevan käytössä oma Scrum-johdannainen niiltä osin kuin käytänteet heidän yritykseensä sopivat. Muissa yrityksissä vaikutti myös olevan käytössä yrityksien omat johdannaiset ketteristä menetelmistä. Tätä päätelmää tukevat myös taulukossa 2 esitetyt tulokset, joista voidaan huomata kaikkien käytänteiden olevan yhtä poikkeusta lukuunottamatta keskimäärin käytössä usein tai hyvin usein. Taulukosta voidaan huomata, että suosituimpia käytänteitä olivat tuotteen työlista ja suunnittelutapaamiset. Myös retrospektiivit, katselmoinnit ja yhteinen omistajuus olivat hyvin suosittuja. Yhdessä yrityksessä (Yritys 2), retrospektiivit ja katselmoinnit eivät olleet käytössä, koska yritys sai palautetta suoraan julkaisijalta ja kehitti toimintaansa tätä kautta. Testausvetoinen kehitys, vertauskuvat, 40-tuntinen viikko, pienet julkaisut ja jatkuva integraatio ovat taulukon perusteella myös suosittuja. Yhdessä yrityksessä (Yritys 2) testausvetoista kehitystä ei tehty, koska pelien testauksesta vastaa julkaisija. Vähemmän käytänteistä käytetään refaktorointia ja pariohjelmointia. Yhdessä yrityksessä (Yritys 2) pariohjelmointia ei käytetty, koska yrityksessä oli käytössä suunnittelija-koodaaja työpari käytänne. Suunnittelupeli ei ollut käytössä juuri yhdessäkään yrityksessä, vaikka sitä oltiinkin kokeiltu. Haastattelujen pohjalta suurimpana tekijänä suunnittelupelin suosion vähäisyydessä oli se, että sen ei koettu tuovan lisäarvoa suunnitteluprosessiin.

Työvirran visualisointia hyödynnettiin aktiivisesti kolmessa haastatellussa yrityksessä (Yritys 1, Yritys 3, Yritys 4). Yhdessä yrityksessä (Yritys 2) oli hyödynnetty työvirran visualisointia aiemmin listojen avulla, mutta haastateltavan mukaan kyseistä menettelyä ei oltu käytetty vähään aikaan syystä tai toisesta. Muissa haastateltavissa yrityksissä työvirtaa visualisoitiin eri tavoin riippuen tilanteesta ja yrityksestä. Yhdessä yrityksessä (Yritys 3) käytettiin perinteistä Kanban-taulua tai sprinttien to-do-tauluja, riippuen tilanteesta. Toisessa yrityksessä (Yritys 1) yksi tiimeistä käytti Kanban-taulua ja toinen tiimi yhteistä jaettua työlistaa. Pelikehityksen ulkopuoliset tiimit hyödynsivät myös Kanban-taulua silloin, kun tehtiin erityisen tärkeitä työtehtäviä. Kolmannessa yrityksessä (Yritys 4) oli käytössä JIRA-ohjelmisto, johon kuuluu työvirran visualisointiin tarkoitettu liitännäinen. Haastateltava totesi, että yrityksen mobiilipuolella saattoi mahdollisesti olla käytössä fyysinen Kanban-taulu, mutta hän ei ollut asiasta täysin varma.

6.5. Pelikehityksen ongelmat

Koska haastateltavat kokivat, että pelien ylläpito kohdeyrityksissä on enemmän jatkokehittelyä ja pääpaino on uuden sisällön luomisessa, päädyttiin haastattelussa keskittymään pääasiallisesti pelikehityksen ongelmiin (vt. Liite 7). Ongelmiin kuuluu mittakaavan määrittelyn ongelmat, epärealistinen mittakaava, ominaisuuksien kasvu, ominaisuuksien karsiminen, suunnitteluvaiheen ongelmat, viivästykset, tekniset ongelmat, loppurutistukset, dokumentaation puute, testausongelmat, ohjelmistovirheet, ryhmäkoostumukselliset ongelmat, kehittäjien menettäminen ja budjetin ylitys. Vastaukset on koottu esiintymistiheyden osalta taulukkoon 2 ja vakavuuden osalta taulukkoon 3.

Taulukko 3: Pelikehityksen ongelmien esiintymistiheys

Ongelmat	**Esiintymistiheys (Asteikolla 1-5, 1 hyvin harvoin – 5 hyvin usein)**				
	1	**2**	**3**	**4**	**5**
Mittakaavan määrittelyn ongelmat	1	0	0	1	2
Epärealistinen mittakaava	1	0	0	2	1
Ominaisuuksien kasvu	1	0	0	3	0
Ominaisuuksien karsiminen	1	0	0	2	1
Suunnitteluvaiheen ongelmat	0	1	1	0	2
Viivästykset	1	0	0	1	2
Tekniset ongelmat	0	1	0	1	2
Loppurutistukset	1	1	1	0	1
Ohjelmistovirheet	0	0	0	2	2
Dokumentaation puute	0	1	2	0	1
Testausongelmat	0	0	1	2	1
Ohjelmistovirheet	0	0	0	2	2
Ryhmäkoostumukselliset ongelmat	1	1	1	1	0
Kehittäjien menettäminen	2	0	0	2	0
Budjetin ylitys	1	0	1	2	0

Taulukko 4: Pelikehityksen ongelmien koettu vakavuus

Ongelmat	Vakavuus (Asteikolla 1-5, 1 ei kovin vakava – 5 hyvin vakava)				
	1	**2**	**3**	**4**	**5**
Mittakaavan määrittelyn ongelmat	0	0	1	2	1
Epärealistinen mittakaava	0	0	0	3	1
Ominaisuuksien kasvu	0	0	0	3	1
Ominaisuuksien karsiminen	3	1	0	0	0
Suunnitteluvaiheen ongelmat	0	3	1	0	0
Viivästykset	0	0	2	0	2
Tekniset ongelmat	0	0	3	1	0
Loppurutistukset	0	1	2	1	0
Ohjelmistovirheet	0	0	2	0	2
Dokumentaation puute	0	2	2	0	0
Testausongelmat	0	0	3	1	0
Ryhmäkoostumukselliset ongelmat	0	1	1	2	0
Kehittäjien menettäminen	0	0	2	1	1
Budjetin ylitys	0	0	3	1	0

Haastateltavien antamien vastauksien pohjalta voidaan huomata, että kirjallisuudessa esitellyistä ongelmista (vrt. alaluku 2.6) suurin osa esiintyi joko usein tai hyvin usein haastateltujen yritysten keskuudessa. Eniten esiintyi ongelmia mittakaavan määrittelyssä, ominaisuuksien kasvussa ja karsimisessa sekä viivästyksissä. Lisäksi usein ilmenee myös teknisiä ongelmia, ohjelmistovirheitä ja testausongelmia. Nämä ongelmat esiintyivät useimmissa yrityksissä (Yritys 1, Yritys 3, Yritys 4) joko usein tai hyvin usein. Yhdessä yrityksessä (Yritys 2) edellä mainitut ongelmat esiintyivät hyvin harvoin. Syyksi tähän haastateltava esitti kattavien suunnitelmien tekemisen ennen tuotannon aloittamista.

Vakavuuden osalta taas vastaukset olivat enimmäkseen hyvin neutraaleja. Suurin osa ongelmista koettiin olevan keskiluokkaa vakavuudeltaan. Mielenkiintoisena huomiona tuloksissa on, että vaikka ominaisuuksien karsimista ja suunnitteluvaiheen ongelmia esiintyi useimmissa

yrityksissä usein, ei niitä kuitenkaan koettu kovinkaan vakaviksi ongelmiksi. Viivästyksien ja ohjelmistovirheiden osalta yrityksien vastaukset jakaantuivat kahteen eri ryhmään. Hyvin vakavana viivästyksiä piti yritysten 1 ja 2 edustajat, kun taas yrityksissä 3 ja 4 viivästykset koettiin olevan vakavuudeltaan keskiluokkaa. Erot vastauksien välillä voivat johtua siitä, mistä näkökulmasta haastateltavat ovat ongelmalistaa täyttäneet. Esimerkiksi yksi haastateltava (Yritys 1) koki viivästyksien olevan vakavia aina, koska ne tarkoittavat resurssien hukkaamista. Toisessa yrityksessä (Yritys 3) haastateltava esitti, että viivästyksien vakavuus riippuu täysin tilanteessa, milloin se tapahtuu. Esimerkiksi julkaisun viivästyminen oli haastateltavan mukaan huomattavasti vakavampaa kuin sisäisten aikataulujen viivästyminen.

Haastateltavilta tiedusteltiin myös, ilmenivätkö ongelmat yleisellä tasolla vai mahdollisesti vasta julkaisun jälkeen. Haastattelujen pohjalta voidaan todeta, että ohjelmistovirheitä tapahtuu läpi pelikehitysprosessin ja niiden määrä yleensä kasvaa lähestyessä julkaisua ja julkaisun jälkeistä aikaa. Mittakaavalliset ongelmat puolestaan ilmenevät yleensä hyvin aikaisessa vaiheessa. Ominaisuuksien karsiminen puolestaan esiintyy yleensä loppua kohden. Rutistusajanjaksot taas puolestaan keskittyivät pääasiallisesti julkaisua edeltävään aikakauteen, mutta niitä voi myös esiintyä julkaisun jälkeen. Henkilöstöllisiä ongelmia, vaikka ne eivät olisikaan juuri kehittäjien menettämisiä, ilmenee yleensä juuri rutistusajanjaksojen aikana.

Pari haastateltavaa (Yritys 1, Yritys 3) toi myös esille näkemyksen siitä, että kirjallisuudessa esiintyvät pelikehityksen ongelmat ovat hyvin universaaleja ongelmia. Ensimmäisessä yrityksessä (Yritys 3) tämä näkemys perustui oletettavasti yrityksen lähestymistapaan pelikehitystä kohtaan. Yrityksessä oli käytössä kolmivaiheinen pelikehitysprosessi, joka jatkui myös samankaltaisena julkaisun jälkeen. Toisessa yrityksessä (Yritys 1) puolestaan koettiin, että esimerkiksi mittakaavan määrittely kuuluu aina osaksi suunnittelupalavereita, joita pidetään aika ajoin ketterästä lähestymistavasta johtuen.

Hmm. Toisaalta niin kuin sanoin, että kun meillä tuotekehitys jatkuu julkasun jälkeen samalla syklilla tällä kolme steppisyksillä, niin periaatteessa nämä kaikki pystytään jakamaan jotenkin niihin kolmeen eri steppiin, tai siis siten, että näitä kaikki varmasti esiintyy kaikissa stepeissä. Et en lähtisi erottelemaan tiettyihin steppeihin. Voi tietysti olla, että jotain esiintyy pikkusen enemmän jossain stepissä. (Yritys 3)

Itseasiassa ei, mittakaavan määrittely menee siihen suunnittelupalaveriin. Eli kun me tehään se peli useassa suunnitteluvaiheessa, suunnitellaan ja tehään jne. Aina siinä vaiheessa kun lähetään suunnittelemaan, niin se mittakaava on, voidaan sanoa, että se on prosentuaalinen ongelma, elikkä me, mitä tahansa suunnitellaan, niin siinä on aina tietty

määrä heittoa ja mitä isompi se isompi suunnitelma on, sitä isompi se heitto on luonnollisesti. (Yritys 1)

kehitysvaiheeseen ei, mutta se voi johtaa siihen että me vaan käydään ne, ne kehitysvaiheet voi olla olemassa mutta me vaan käydään ne tiheällä tahdilla läpi eikä kerran niin kuin kerran koko tuotteen syklissä koska ketterää. (Yritys 1)

6.6. Ketterien menetelmien käytöstä koetut hyödyt ja haasteet

Ketterien menetelmien käyttäminen koettiin yleisesti ottaen erittäin hyödylliseksi kaikkien haastateltavien keskuudessa. Yhdeksi hyödyksi koettiin yleisen toiminnan organisoinnin parantuminen ja helpottuminen. Esimerkiksi erilaisten työlistojen koettiin selkeyttävän työnteon hallintaa, koska näin pystyttiin helposti näkemään mitä pitäisi olla tehtynä ja milloin. Tuottajan näkökulmasta ketterien menetelmien koettiin auttavan hallinnointia ja estimointia. Erityisesti iteratiivinen lähestymistapa koettiin hyvänä, koska helpottuneen estimoinnin kautta työtaakat saadaan jaettua iteraatioihin sopiviksi paloiksi. Tämän puolestaan koettiin lisäävän muun muassa toiminnan stabiiliutta ja vähentävän mahdollisten rutistusajanjaksojen tarvetta. Työtaakkojen paloittelemisen todettiin myös mahdollistavan sen, että voidaan nähdä aikaisemmassa vaiheessa, saadaanko jokin työtaakka valmiiksi ajoissa. Erilaisten retrospektiivien koettiin myös olevan erittäin tärkeitä. Tähän syynä oli se, että oppimisen ja toiminnan jatkuvan kehittämisen koettiin olevan hyvin tärkeitä seikkoja. Seuraavassa on esitetty joitakin otteita haastatteluista:

No periaatteessä nämä tietysti auttaa estimoimaan, ja jos puhutaan tuottajan näkökannasta, niin nämä auttaa manageroimaan ja estimoimaan asioita huomattavasti paremmin. (Yritys 3)

Sit tietysti noin yleisesti niin kuin tiimin johtamisen kannalta retrospektiivit, vaikka ei välttämättä kutsuta retrospektiiveiksi, ne on ehdottoman tärkeitä, että aina opitaan tekemisestä jotain ja tehään ne jutut huomattavasti paremmin. (Yritys 3)

Se ajatus, että pidetään asiat stabiilina, lisätään uus iteraatio missä on uusi näkyvä hyödyllinen asia, niin tämä on silleen hyvä asia, että saadaan jaoteltua, että ei mene silleen, että ollaan puoli vuotta niin kuin koodaillaan jotain, ja sitten tulee yhtäkkiä kauheen iso milestone ja crunchi ja kaikkea muuta, että se tavallaan koittaa jakaa sitä, jakaa tätä tarvetta crunchata tai tehä isoa milestonea pienempiin palasiin ja pystyy aikasemmin näkemään, että onko joku asia valmistumassa ajoissa vai ei.(Yritys 4)

Yhdessä yrityksessä (Yritys 1) ketterien menetelmien koettiin olevan hyödyllisiä yleisellä tasolla, koska voidaan joutua toimimaan osittaisessa tietämättömyydessä, jolloin tarvitaan joustavia toimintatapoja. Haastateltava

koki myös ketterien menetelmien olevan luonnollinen ratkaisu heidän filosofiaansa, jonka mukaan he eivät tahtoneet olla parempia välttelemään ongelmia vaan parempia ratkomaan niitä:

No siis yleisesti miksi me näitä tehään, no sen takia kun ei me tiedetä. Eli jos ketteryyden vastakohta on jäykkyys, waterfall tai vastaava, niin eihän se toimisi ollenkaan, koska me ei vaan tiedetä. Niin sillon meidän on pakko tehdä jotain, mikä odottaa tätä epätietosuutta ja sitten meidän filosofia ylipäätään on, että me ei haluta olla parempia estämään ongelmia, vaan me halutaan olla parempia selviämään ongelmista, koko firman tasolla. (Yritys 1)

Hyödyistä huolimatta ketterät menetelmien käyttö sisälsi kuitenkin jonkin verran haasteita. Yhdeksi haasteeksi esitettiin se, että koska kyseessä on luova ala ja koska pelien välillä voi olla hyvin paljon eroavaisuuksia, ei voida olla täysin varmoja aiempien prosessien toimivan uuden projektin yhteydessä. Tästä johtuen toimintaa tulee kehittää jatkuvasti, mihin voidaan joutua käyttämään hyvin paljon resursseja, jotka mieluummin oltaisiin käytetty tuotteen tuottamiseen. Parissa tapauksessa (Yritys 3, Yritys 4) nousi esille myös joidenkin ketterien menetelmien jäykkyys. Erityisesti esille nousi Scrum ja kuinka Scrumin toteuttaminen vaatisi kaikkien käytänteiden kirjaimellista noudattamista. Tässä suhteessa alleviivattiin jälleen sitä, kuinka retrospektiivit, katselmoinnit ja muut työnulkopuoliset estimointitapaamiset syövät hyvin paljon aikaa itse työnteosta. Yhdeksi haasteeksi koettiin myös toiminnan ajoittainen teennäisyys, mikä johtui yhteisten ja konkreettisten sprinttipäämäärien asettamisen haastavuudesta. Haastateltava (Yritys 4) alleviivasi tässä, että projektin jäsenten työtehtävissä voi olla hyvinkin paljon eroja, jolloin voi olla vaikeaa asettaa työntekijöille yhteistä konkreettista päämäärää, mikä puolestaan voi johtaa toiminnan teennäisyyteen.

Minun mielestäni näissä on se ongelma, että jos orjallisesti ja kun pitäisi scrumia orjallisesti käyttää, niin siihen kuuluu tosi paljon tällaista niin kuin työn ulkopuolella tai niin kuin varsinaisen työn ulkopuolella olevaa estimointia, checkuppeja, istumista ja katsomista ja niin edelleen. Et siihen palaa tosi tosi paljon aikaa ja sitten siihen, että mikä hyöty siitä tulee, niin se ei välttämättä vastaa siihen investoitua aikaa. (Yritys 3)

No se että kun tiimit tekee aina jotain uutta, niin ihmiset on eri ja tuote on eri ja niitten tuotteen eroavuuden takia ne haasteet on erit, jolloin ei voida sanoa, että tässä on prosessi ja käytetään tätä prosessia, vaan se on jatkuvaa ehkäilyä. Eli toisinsanoen varmuus, sitä ei olo paljon, niin sitten se johtaa ehkäilyyn monenlaisella tasolla ja ehkäilyyn menee vaan energiaa ja aikaa, niin se on ehkä se haaste. (Yritys 1)

Kun scrumissa pitäis olla silleen, että tiimillä on ne goalit, ja sitten mitataan tiimin edistymistä storypointtien mukaan niin se on vähän teennäistä ja se ei, jos enginetiimistä yksi tyyppi tekee file systeemia ja threadaus systeemia ja yksi tyyppi ihan muuta, niin on vaikee

keksiä, mikä yhteinen goali niillä on. Joskus voi olla myös vaikea miettiä, mikä on nyt se tämän sprintin näkyvä uusi muutos tähän liittyen. Eli se ei välttämättä istu, et jos ei ole tämmöistä näkyvää selkeää featuree, niin se voi olla välillä vähän teennäistä. (Yritys 4)

6.7. Toiminnan kehittäminen

Toiminnan kehittämisen osalta haastateltavilta kysyttiin, miten pelikehitystä voisi kehittää ylläpidon osalta ja mikä rooli ketteryydellä olisi tässä. Aluksi haastateltavilta kysyttiin, olisiko järkevämpää lähestyä ylläpitoa eri näkökulmasta. Tältä osin eräs haastateltava (Yritys 2) koki, että ylläpito on oleellinen osa pelinkehitysprosessia, joka täytyy ottaa jo huomioon heti peliprojektin alussa. Syiksi tähän kuuluivat muun muassa aikataulupaineet pelin julkaisun jälkeen, jolloin ei ole enää aikaa miettiä ylläpitoa tarkemmin, vaan pitää olla valmiit toimintasuunnitelmat.

Mutta kyllä se minun mielestä pitää nähä osana sitä, että kun aloitetaan tekemään peliä, niin miettitään jo siinä vaiheessa ylläpitoa, laajennuksia ja näitä, koska se on epätodennäköistä, että henkilöstö yllättäen kasvaisi kauheasti siinä vaiheessa kun se julkaistaan, ja se on vaan huonoa bisnestä, että suunnitellaan siihen asti, että peli on ulkona ja sitten se peli on ulkona ja sitte asiat on et mitäs me nyt tehään. Kun siinä vaiheessa on kuitenki hoppu, ei siinä kukaan ehdi sillä sekunnilla keskittyy siihen, että mitä me tehdään lisää, minkälaisia laajennuksia ja muuta, niin se on tosi hyvä olla ne sopimukset ja suunnitelmat siinä vaiheessa. (Yritys 2)

Toinen haastateltava (Yritys 1) taas koki, että ylläpitoa on vaikea erottaa pelikehityksestä, koska voi olla epäselvää, milloin siirrytään jatkokehitykseen ja sen myötä ylläpidollisten tehtävien pariin. Haastateltava alleviivasi myös, että erottaminen erillisiksi on vaikeaa, koska prosessit yleisellä tasolla eivät välttämättä muutu ollenkaan, vaikka ollaan jo jatkokehityksen puolella. Haasteltava kuitenkin totesi lopuksi, että tämä tilanne on tarkoituksellisesti luotu heidän yritykseen, joten on vaikeaa sanoa, voidaanko yleisesti sanoa tilanteen olevan vastaava muissa alan yrityksissä.

Eli voidaan ehkä vaan huomata, havaita jonain päivänä, että kappas ehkä me ollaan jatkokehityksessä, mutta sekään ei välttämättä vaihda sitä prosessia itsessään. Mutta tämä on tarkotuksellista. Eli meidän kehitys muistuttaa vaan niin paljon jatkokehitystä syystä, niin ehkä sen takia ei niin kaivata. (Yritys 1)

Ylläpitoprosessin kehittymisen osalta haastateltavat kokivat, että tärkeintä on ajatella menetelmät ja käytänteet työkaluina ja taivuttaa ne firman omiin

työtapoihin eikä päinvastoin. Tärkeää on muuttaa prosesseja vastaamaan yrityksen kulloistakin kehitystä. Myös projektiryhmien tarpeet tulisi ottaa huomioon eikä pakottaa kaikkia ryhmiä samaan muottiin, vaan pyritään valitsemaan sellaiset käytänteet, jotka maksimoivat ryhmän työskentelytehokkuuden. Tässä suhteessa tuotiin esille esimerkiksi Scrum, jonka ei koettu soveltuvan niin hyvin ylläpidollisiin työtehtäviin.

Hmm, no minun mielestäni on tärkeetä, ettei välttämättä ajatella sitä työkalua, että ei tehdä sitä duunia sen työkalun periaatteiden mukaan, vaan löydetään se oma tapa tehä. (Yritys 3).

Vaan enemmänkin se, että me nähdään, että firma kehittyy johonkin suuntaan ja meidän pitää muuttaa prosesseja vastaamaan sitä firman silloista kehitystä, ja me niin kuin mietitään sitä tulevaisuuteen aika pitkälle. (Yritys 1)

Kun meilläkin on useampi tiimi sprintissä, niin koitetaan löytää tämmösiä yhteisiä isoja goaleja missä porukat voisi yhdessä miettiä ketkäs tähän tiimiin kuuluisivat. Jotkut saa allergiaa siitä liiallisesta taskiajattelusta ja planaamisesta, että jos johonkin hommiin sopii paremmin joku muu juttu, niin mietitään uudelleen ne asiat sillon. Et ei pakoteta kaikkia samaan muottiin, jos se tuntuu tehottomalta. (Yritys 4)

Organisoinnin osalta suurin osa haastateltavista koki, että ylläpidollisiin tehtäviin olisi hyvä saada toinen projektitiimi, joka huolehtii pääasiallisesti juuri ylläpidollisista toimenpiteistä. Yhden näkemyksen (Yritys 4) mukaan ideaalinen tilanne olisi, että kehitystehtävien pariin laitetaan firman kärkitiimi ja ylläpidollisiin tehtäviin puolestaan junioritiimi. Tällaiseen tilanteeseen pääseminen on kuitenkin vaikeaa. Yhdeksi syyksi tähän esitettiin muun muassa henkilöstölliset rajoitteet, kuten tarpeeksi pätevien henkilöiden löytäminen.

Mieluiten olisi toinen tiimi, käytännössä se on tosi vaikeeta. Ehkä jonain päivänä, että me voidaan jakaa ne. Mutta kun meillä kaikilla on aika selkeät osa-alueet, niin sekin, että sitten meillä pitäsi olla melkein kaksi ihmistä kaikessa, sitten niiden pitäisi kommunikoida keskenään ja se voi hidastaa ja näin. Et pitäisi melkein tuplata koodauksen määrä ja hyvät koodaajat on kaikkein vaikein löytää. (Yritys 2)

Mut ideaalisesti se voisi mennä siten, että se ihan kärkitiimi, joka kehittää uutta, menisi mahollisimman nopeasti jo kehittämään sitä seuraavaa projektia ja sit juniorimman, jos puhutaan oikeesti ylläpidosta, korjataan bugeja ja pidetään yllä, että kaikki hommat toimii, niin se voisi olla semmoinen pienempi porukka. (Yritys 4)

6.8. Yhteenveto

Tässä luvussa esiteltiin tuloksia haastattelututkimuksesta, joka koski ketterien menetelmien ja käytänteiden hyödyntämistä julkaisun jälkeisessä ylläpitoprosessissa pelistudioissa. Haastattelulla pyrittiin valottamaan pelistudioiden taustoja, videopelien ylläpitoa, ylläpitoprosessia, ketterien menetelmien hyödyntämistä, pelikehityksen ongelmia, ketterien menetelmien käytöstä koettuja hyötyjä ja haasteita sekä toiminnan kehittämistä. Seuraavassa luvussa esitetään tuloksien pohjalta tehdyt johtopäätelmät ja pohdintaa.

7. POHDINTA

Tämän haastattelututkimuksen tarkoituksena oli selvittää, millaista on videopelien ylläpito, millaisia ongelmia videopelien ylläpidossa esiintyy, missä määrin pelien kehittämisessä käytetään ketteriä menetelmiä ja käytänteitä sekä millaisia hyötyjä ja ongelmia ketterien menetelmien ja käytänteiden käytöstä on koettu videopelien ylläpidossa. Tutkimuksen kohteeksi valittiin neljä videostudiota, joista kustakin valittiin yksi kokenut henkilö haastateltavaksi. Haastattelut tehtiin käyttäen puolistruktiroituja kysymyksiä. Tutkimuksesta tekee kiinnostavan se, ettei pelien ylläpidosta ole tiettävästi ennen tehty empiirisiä tutkimuksia.

Seuraavissa alaluvuissa esitetään ensin pääasialliset tulokset teemoittain, verrataan niitä aiempiin tutkimuksiin ja vedetään johtopäätöksiä. Toisessa alaluvussa pohditaan tutkimuksen reliabiliteettia ja validiteettia.

7.1. Tulokset ja johtopäätökset

Tässä luvussa vastataan aiemmin johdannossa esitettyihin tutkimuskysymyksiin. Tutkimuskysymysten avulla oli tarkoitus kartoittaa pelien ylläpidon sisältöä, pelien ylläpidon ongelmia ja ketteryyden hyötyjä ja haittoja ylläpidon parissa.

7.1.1. Millaista on videopelien ylläpito / jatkokehittely?

Haastattelututkimuksen perusteella vaikuttaa siltä, että pelien ylläpito mielletään enemmän jatkokehittelyksi kuin perinteisen ohjelmistotuotannon tarkoittamaksi ylläpidoksi. Toisin sanoen julkaisun jälkeen toiminnan

pääpainona on uuden sisällön kehittäminen. Tästä huolimatta, vaikka haastateltavat kokivat pelien julkaisun jälkeisen ajan jatkokehittelyksi, olivat myös erinäiset ylläpidolliset toimenpiteet osa tätä jatkokehitysprosessia.

Sisällöllisesti videopelien ylläpito vaikuttaa vastaavan Mannisen ym. (2006) esittämää kuvausta ylläpitovaiheen sisällöstä. Videopelien ylläpitoon kuuluu virheiden korjausta, pelielementtien tasapainon muokkaamista, huijauksien estämistä sekä uuden sisällön luomista, joko lisäpakettien tai lisäosien muodossa. Lisäksi oleellisena osana on myös pelaajayhteisöstä huolehtiminen sekä palautteen kerääminen, mikä auttaa kehittäjiä määrittämään esimerkiksi sen, millaista uutta sisältöä pelaajat haluavat.

Prosessillisesti pelien jatkokehittely vaikuttaa jakaantuvan kaksilinjaiseksi. Ensimmäisen linjan ytimenä toimii pääasiallinen tuotantotiimi, joka keskittyy tuottamaan lisäsisältöä ja uusia ominaisuuksia. Toisen linjan ytimenä on puolestaan ylläpitotiimi, jonka tehtäviin kuuluvat kaikki ylläpidolliset tehtävät, pelaaja suhteiden ylläpito ja pelaaja palautteen kerääminen. Ylläpitotiimin tärkeänä tehtävänä on kerätä dataa esimerkiksi pelaajakäyttäytymisestä, pelin puutteista ja siitä mitä uutta sisältöä pelaajat peliin haluaisivat. Tuotantotiimi hyödyntää tätä kerättyä dataa suunnitellessaan, mihin suuntaan peliä tullaan kehittämään. Tällainen linjajaottelu oli käytössä kolmessa neljästä haastatellusta yrityksestä ja yritys, jossa menettely ei vielä ollut käytössä, aikoi todennäköisesti siirtyä kyseiseen menettelyyn tulevaisuudessa. Jatkokehittelyprosessi vaikuttaa pääosin toimivan täysin samalla tavalla kuin itse pelikehitysprosessi ennen julkaisua. Yleistetysti tuotantotiimi käy läpi suunnittelu-, tuotanto- ja julkaisuvaiheet, joiden sisällä ovat käytössä yritysten omat näkemykset ketterästä kehityksestä. Siitä miten ylläpitotiimin prosessi toimii, ei saatu tarpeeksi tietoa, mikä voidaan nähdä yhtenä tämän tutkimuksen puutteena. Virheiden korjauksien osalta saatiin jonkin verran tietoa ja vaikuttaa siltä, että virheidenkorjaus muistuttaa hyvin paljon ohjelmiston ylläpidon korjaavaksi ylläpidoksi määriteltyä ylläpitoa (ISO/IEC, 2006). Toisin sanoen tarkoituksena on korjata niin ennen julkaisua kuin julkaisun jälkeenkin löytyneet virheet.

Pelien ylläpidon elinkaaren osalta haastattelussa kysyttiin, kuinka hyvin ohjelmiston ylläpidon elinkaari (varhaislapsuus, murrosikä, aikuisuus,seniiliys) (Kitchenham ym., 1999) sopisi pelien ylläpidon yhteyteen. Tuloksien perusteella voidaan sanoa, että esitys ohjelmiston ylläpidon elinkaaresta ei vastaa peliteollisuuden todellisuutta. Useimmissa tapauksissa haastattelijat kokivat pelin julkaisun sijoittuvan ohjelmiston elinkaaressa esitettyyn aikuisuusvaiheeseen. Tämä johtuu siitä, että pelien käyttäjäpotentiaali saavuttaa yleisesti ottaen huippunsa juuri julkaisun yhteydessä. Lisäksi aktiviteettien kannalta pelin julkaisun jälkeiset toimenpiteet vastasivat

aikuisuusvaiheeseen listattuja aktiviteetteja. Lapsuus- ja murrosikävaiheet voitiin jossain määrin ajatella pelin alpha- ja beta-vaiheiksi, mutta tämä näkemys oli jokseenkin ontuva. Tästä johtuen olisikin siis tarpeellista luoda erillinen malli pelien ylläpidon elinkaaresta vastaamaan peliteollisuuden todellisuuksia. Tässä mallissa olisi tärkeää huomioida muun muassa mahdollinen alustan, palvelumallin ja kustantajan vaikutus pelin ylläpidon elinkaareen.

7.1.2. Millaisia ongelmia esiintyy videopelien ylläpidossa / jatkokehittelyssä?

Videopelin ylläpidon / jatkokehittelyn ongelmat vaikuttivat olevan hyvin samankaltaisia verrattuna itse pelikehitysprosessin ongelmiin (Petrillo ym., 2009; Kanode & Haddad, 2009). Tämä johtui siitä, että jatkokehittelyprosessi oli hyvin samankaltainen pelikehitysprosessiin nähden. Eniten ongelmia vaikutti esiintyvän erityisesti mittakaavan määrittelyssä, ominaisuuksien kasvussa, suunnitteluvaiheessa yleisesti, viivästyksissä, teknisissä ongelmissa ja ohjelmisto- ja testausongelmissa. Yllättävää kuitenkin oli, että suurin osa ongelmista koettiin vakavuudeltaan hyvin neutraaleiksi muutamaa poikkeusta lukuun ottamatta. Vakavimmiksi ongelmiksi koettiin mittakaavan määrittelyn ongelmat ja ominaisuuksien kasvu. Huomioitavaa on myös, että vaikka ominaisuuksien karsiminen ja suunnitteluvaiheen ongelmat olivat hyvin yleisiä haastatelluissa yrityksissä, ei niitä koettu kuitenkaan kovinkaan vakaviksi ongelmiksi. Tutkimuksesta saadut tulokset ovat linjassa aiempaan tutkimukseen nähden. Esimerkiksi Petrillon ym. (2009) suorittamassa tutkimuksessa suurimmiksi ongelmiksi todettiin mittakaavan määrittely ja ominaisuuksien kasvu, mikä pitää paikkansa myös tässä tutkimuksessa. Petrillon ym. (2009) tutkimuksessa yleisimmiksi ongelmiksi siteerattiin myös suunnitteluvaiheen ongelmat, viivästykset ja tekniset ongelmat, mitkä olivat yleisimpiä ongelmia myös tämän tutkimuksen yhteydessä. Tulokset vastaavat myös Kanoden ja Haddadin (2009) tutkimusta, jossa pelikehityksen yhdeksi haasteeksi esitetään suunnitteluvaiheen ongelmat, johon voidaan katsoa kuuluvan myös ongelmat mittakaavan määrittelyssä. Kanoden ja Haddadin (2009) tutkimuksessa nostettiin esille myös erityisesti tekniset ongelmat, jotka olivat myös tämän tutkimuksen mukaan yleisiä.

Näkemykset siitä, miten ongelmat jakaantuivat prosessin eri vaiheisiin, oli erilaisia näkemyksiä haastateltavien joukossa. Yhden näkemyksen mukaan esimerkiksi ohjelmistovirheitä tapahtuu läpi pelikehitysprosessin ja niiden määrä yleensä kasvaa lähestyttäessä julkaisua ja julkaisun jälkeistä aikaa. Mittakaavalliset ongelmat taas ilmenevät yleensä hyvin aikaisessa vaiheessa prosessia. Ominaisuuksien karsiminen sijoittuu myös yleensä prosessin

loppuun. Rutistusajanjaksot taas keskittyvät voimakkaasti julkaisua edeltävään ajanjaksoon. Henkilölliset ongelmat ilmenevät yleensä näiden rutistusajanjaksojen aikana. Toisen näkemyksen mukaan ongelmat ovat hyvin universaaleja. Esimerkiksi mittakaavan määrittely voi kuulua osana kaikkia suunnittelupalavereita, joten sen voidaan nähdä olevan jatkuvasti läsnä ketterästä lähestymistavasta johtuen.

7.1.3. Millaisia hyötyjä ja ongelmia ketterien menetelmien ja käytänteiden käytöstä videopelien ylläpidossa / jatkokehittelyssä on koettu?

Yleisesti ottaen ketterät menetelmät ja käytänteet koettiin haastateltavien keskuudessa erittäin hyödyllisiksi pelikehityksessä. Yhdeksi höydyksi koettiin muun muassa yleisen toiminnan organisoinnin helpottuminen ja parantuminen. Erilaisten työlistojen koettiin selkeyttävän työnteon hallintaa, koska ne selkeyttivät aikataulujen ja tehtävien suhteuttamista toisiinsa. Tuottajien näkökulmasta iteratiivinen lähestymistapa koettiin hyväksi, koska helpottuneen estimoinnin kautta työtaakat saatiin helpommin jaettua iteraatioihin sopiviksi paloiksi minkä puolestaan koettiin lisäävän stabiiliutta ja vähentävän rutistusajanjaksojen tarvetta. Työtaakkojen paloittelemisen koettiin myös mahdollistavan nopeamman reagoimisen mahdollisiin myöhästymisiin. Retrospektiivien puolestaan koettiin olevan hyvin tärkeitä oppimisen ja toiminnan jatkuvan kehittämisen kannalta. Ketterien menetelmien voidaan myös sanoa olevan hyvin hyödyllisiä, koska pelikehityksen parissa voidaan joutua toimimaan osittaisessa tietämättömyyden tilassa, jolloin joustavat menetelmät ovat erityisen tärkeitä. Ketterien menetelmien voidaan myös sanoa kehittävän toimijoita vastaamaan ongelmiin tehokkaasti, eikä niinkään välttelemään niitä.

Saadut tulokset ovat linjassa aiempaan tutkimukseen nähden. Esimerkiksi muutoksiin vastaaminen nähtiin ketterien menetelmien eduksi Poolen ym. (2001) tutkimuksessa, mikä taas näkyi tässä tutkimuksessa parantuneena reaktionopeutena potentiaalisiin myöhästymisiin ja näkemyksenä siitä, että ketterät menetelmät auttavat toimijoita vastaamaan ongelmiin tehokkaammin. Erilaisten työlistojen selkeyttävä luonne voidaan myös katsoa kuuluvan tiedon välittämisen parantumiseen, mikä nähtiin etuna Svenssonin ja Höstin (2005) tutkimuksessa. Seikola ym. (2011) esittivät ketterien menetelmien tuovan tiimiin liittyviä etuja, jotka oletettavasti tämän tutkimuksen yhteydessä ilmenivät yleisen toiminnan organisoinnin helpottumisena ja parantumisena.

Hyödyistä huolimatta ketterien menetelmien tehokas hyödyntäminen on myös hyvin haastavaa. Peliala on hyvin luova viihteen ala, jossa toimii henkilöitä useilta erilaisilta toimialoilta ohjelmoijista säveltäjiin. Pelien välillä

löytyy myös paljon eroja, mistä johtuen ei voida olla täysin varmoja aiempien prosessien toimivuudesta uuden projektin parissa. Tästä johtuen toimintaa täytyy kehittää jatkuvasti ja aktiivisesti, mikä taas syö paljon resursseja erityisesti tuotteen tuottamiselta. Jossain tapauksissa ketterät menetelmät voivat myös olla liian jäykkiä. Esimerkiksi Scrum ja sen vaatima käytänteiden, kuten retrospektiivien, katselmointien ja muiden työnulkopuolisten estimointitapaamisten kirjaimellinen noudattaminen syö hyvin paljon aikaa itse kehitystyöltä. Pelialalla toimitaan usein tiukkojenkin aikataulujen puitteissa. Jos peli ei valmistu esimerkiksi joulumarkkinoille, on mahdollista, että peli voidaan julkaista järkevästi vasta usean kuukauden päästä. Tästä johtuen on hyvin vaikeaa tasapainottaa toisaalta tarpeelliset ja hyödylliset estimointitapaamiset työnteon kanssa. Jossain määrin on myös vaikeaa asettaa konkreettisia päämääriä hyvinkin erilaisissa työtehtävissä toimiville projektin jäsenille, mikä voi johtaa toiminnan teennäisyyteen. Samoja ongelmia on esitelty myös Millerin (2008) artikkelissa. Esimerkiksi juuri päivittäiset Scrum-tapaamiset olivat haasteellisia, koska niihin käytetään potentiaalisesti turhaan aikaa ja käytetystä ajasta saadut hyödyt voivat olla hyvinkin kyseenalaisia. Miller (2008) esitti ongelmallisiksi myös muut estimointitapaamiset ajanhukasta johtuen, mikä vastaa myös tässä tutkimuksessa saatuja tuloksia.

7.1.4. Kehittämisideoita

Haastatteluissa tuli ilmi, että toiminnan kehittämisen kannalta nähtiin tärkeäksi, että käytettävät menetelmät ja käytänteet koettaisiin työkaluina, jotka taivutetaan pelistudion omiin tarpeisiin. Tärkeää on myös ottaa huomioon projektiryhmien tarpeet, eikä ryhmiä tule pakottaa käyttämään samoja menetelmiä ja käytänteitä, vaan käytänteet tulee valita ryhmäkohtaisesti siten, että ne maksimoivat kunkin ryhmän työskentelytehon.

Ylläpito / jatkokehittely on tärkeä ja oleellinen osa pelikehitysprosessia, ja se tulisi näin ollen ottaa huomioon heti peliprojektin alussa. Kun peli julkaistaan ja täytyy siirtyä ylläpidon / jatkokehittelyn pariin, voi ilmaantua muun muassa aikataulu paineita, joista johtuen ei mahdollisesti enää ole aikaa miettiä ylläpitoa / jatkokehittelyä tarkemmin. Tästä johtuen toimintasuunnitelmat tulee laatia hyvissä ajoin.

Toimijoiden osalta ylläpitoon / jatkokehittelyyn olisi hyvä asettaa toinen projektitiimi, joka huolehtii pääasiallisesti joko ylläpitoon tai jatkokehittelyyn liittyvistä työtehtävistä. Yksi mahdollisuus olisi sijoittaa pelistudion kärkitiimi huolehtimaan kehitystehtävistä ja sijoittaa hieman kokemattomampi junioritiimi ylläpidon pariin. Tämä voi kuitenkin osoittautua ongelmalliseksi

muun muassa henkilöstöllisten rajoitteiden takia, joihin kuuluu esimerkiksi pätevien henkilöiden löytäminen.

7.2. Tutkimuksen reliabiliteetti ja validiteetti

Empiirisiä tutkimuksia arvioidaan yleensä arvioidaan reliabiliteetin ja validiteetin perusteella. *Reliabiliteetilla* kuvataan sitä, kuinka samalla tavalla toteutetut, samaa ilmiötä kuvaavat tutkimukset tuottavat yhtenevät tulokset (Hirsjärvi ym. (2009, 232-233). Reliabiliteetin tarkoituksena on minimoida virheiden, vääristymien ja sattumanvaraisten tulosten määrää tutkimuksissa (Yin, 2009; Hirsjärvi ym. 2009). Reliabiliteettia pyrittiin parantamaan tässä tutkimuksessa ensiksikin sillä, että tutkimusprosessi alkaen tutkimusasetelmasta ja haastatteluringosta ja päätyen tiedonkeruutapojen kuvaamiseen ja tulosten esittelyyn on esitetty yksityiskohtaisesti. Kuka tahansa voi toistaa tutkimuksen. Tosin tulokset voivat olla erilaisia, koska ne riippuvat kontekstista. Uhkana reliabiliteetille voi olla myös haastattelijan vaikutus haastateltaviin (Hirsjärvi ym. 2009). Haastattelujen aikana ei ilmaantunut erityisiä ongelmia, ja haastateltavat vastasivat tyhjentävästi kaikkiin esitettyihin kysymyksiin ongelmitta. Haastateltavien rehellisyyttä ei ole syytä epäillä, sillä haastattelu koski yrityksille tärkeitä aiheita. Haastateltaviksi saadut henkilöt olivat perehtyneitä haastattelun aihealueeseen. Haastattelukysymysten asettelussa pyrittiin käyttämään ymmärrettävää sanastoa, ja erilaiset termit esitettiin suomeksi ja englanniksi tarpeen vaatiessa väärinkäsitysten välttämiseksi. Joitakin kysymysten sanamuotoja olisi voitu hioa vielä paremmiksi, sillä pelikehityksen ongelmia käsittelevässä käsiteluettelossa oli termi ”ongelmat suunnitteluvaiheessa”, mikä voidaan nähdä liian tulkinnanvaraisena ilmaisuna. Toisaalta termi oli kyseisessä muodossa myös pelikehityksen ongelmia. Haastattelut litteroitiin ja analysoitiin haastattelukysymyksiin perustuneella teemoittelulla. Haastattelujen sulavasta kulusta huolimatta on mahdollista, että haastattelijan kokemattomuus vaikutti negatiivisesti haastattelujen tuloksiin eikä kaikkea mahdollista tietoa ole saatu selville. Yhdessä tapauksessa haastateltavalla oli hyvin tiukka aikataulu, mistä johtuen haastattelu jouduttiin suorittamaan hyvin ripeästi, mikä on voinut vaikuttaa heikentävästi kyseisen haastattelun tuloksiin.

Validiteetti ilmaisee tutkimustulosten luotettavuutta ja ilmaisee kuinka tarkasti teoria, malli tai käsite kuvaa todellisuutta (Runeson & Höst, 2009). Validiteetti voidaan jakaa konstruktiovaliditeettiin, sisäiseen ja ulkoiseen valididititeettiin (Yin, 2009;Runeson & Höst, 2009). *Konstruktiovaliditeetilla*

tarkoitetaan sitä, onko käytetty oikeaa mittaria eli miten systemaattisesti tutkimus on toteutettu alkaen tutkimuskysymyksistä ja päätyen johtopäätöksiin. Pelien ketterästä ylläpidosta ei ole aiemmin tehty tutkimusta, joten käytettävissä ei ollut valmista tutkimusmallia tai teoriaa. Tässä työssä rakennettiin tutkimuskysymyksistä johtaen tutkimusasetelma, joka kokoaa yhteen Senapathin ja Srinivasanin (2011) mallirakenteen mukaisesti tutkimuskohteeseen keskeisesti liittyviä osatekijöitä. Haastattelurunko rakennettiin huolellisesti kirjallisuuden pohjalta, ja se testattiin yhdellä testikäyttäjällä, josta saadun palautteen pohjalta haastattelurunkoa kehitettiin edelleen. Haastateltaville annettiin mahdollisuus ilmaista mielipiteensä myös muista asioista. Tutkimusasetelma on alustava, mutta se osoittautui sopivaksi tämän tutkimuksen eksploratiiviseen tarkoitukseen.

Sisäinen validiteetti kuvaa, missä määrin teorian esittämä kausaalianalyysi ja selitykset vastaavat todellisuutta, eli tutkitaanko oikeasti sitä, mitä halutaan tutkia (Yin, 2009; Runeson & Höst, 2009). Tämän tutkimuksen tarkoituksena ei ollut testata käsitteiden ja muuttujien välisiä kausaalisuhteita, joten tämän validiteetin tarkastelu ei ole tässä työssä relevanttia.

Ulkoisella validiteetilla taas kuvataan sitä, missä määrin tutkimuksen tulokset ovat yleistettävissä (Yin, 2009; Runeson & Höst, 2009). Aineisto kerättiin haastattelututkimuksen avulla etelä-Suomen alueella toimineiden peliyritysten työntekijöiltä. Pyyntö haastatteluun osallistumiseksi lähetettiin kaikille Suomen pelialan kattojärjestön, Suomen Pelikehittäjät ry:n (http://www.pelinkehittajat.fi/) jäsenille sekä Keski-Suomen alueella toimivan peliosuuskunta EXPA:n (http://www.expa.fi/) jäsenille. Haastattelupyyntöön vastasi viisi erikokoista yritystä, joista neljä valittiin haastattelun kohteeksi. Kustakin yrityksestä valittiin yksi henkilö haastateltavaksi. Kaikki yritykset edustivat eri genrejä ja pelialustoja. Tästä johtuen voidaan todeta, että tutkimus antaa osviittoja pelien ylläpidon sisältöön, mutta mitään yleistyksiä ei voida tehdä. Tämä johtuu siitä, että otanta on Suomenkin mittakaavassa hyvin pieni.

Koska tutkittavana aiheena oli pelien ylläpito, tuli tutkimukseen osallistujien olla riittävän kokeneita, jotta aiheesta oltaisiin saatu luotettavaa tietoa. Suurin osa tutkimukseen osallistuneista yrityksistä oli useita vuosia vanhoja ja julkaissut useamman kuin yhden pelin. Nuorin yrityksistä oli julkaissut vasta yhden pelin, mistä johtuen yrityksen näkemys pelien ylläpidosta ei välttämättä edusta yleistä näkemystä aiheesta. Toisaalta yrityksen näkemys sisälsi paljon yhtäläisyyksiä ja samoja kehityssuuntia muihin kohdeyrityksiin verrattuna. Kustakin yrityksestä saatiin vain yhden henkilön näkemys asioista. Tästä johtuen mahdolliset erot näkemyksissä eivät tulleet julki. Yrityksiltä kuitenkin pyydettiin haastatteluun osallistujaksi ketteristä

menetelmistä, yrityksen pelikehitysprosessista ja ylläpidosta hyvin perillä olevia henkilöitä.

Yhteenvetona yllä esitetystä reliabilliteetin ja validiteetin tarkastelusta voidaan todeta, että tietyistä puutteistaan huolimatta tutkimuksen tuloksia voidaan pitää suhteellisen luotettavina. Esitettyjen näkemysten ja johtopäätösten vahvistamiseksi tarvitaan kuitenkin jatkotutkimusta.

8. YHTEENVETO

Peliala on voimakkaasti kasvava viihteen ala. Toisin kuin pelien kehittämisestä, niiden ylläpidosta on tehty tähän mennessä hyvin vähän tutkimusta (Ampatzoglou ym., 2010). Pelien ylläpidon tai jatkokehittelyn merkitys tulee kuitenkin entisestään kasvamaan digitaalisen jakelun tuomien mahdollisuuksien myötä.

Tämän tutkimuksen tavoitteena on ollut selvittää, millä tavoin ketteriä menetelmiä voidaan hyödyntää videopelien julkaisun jälkeisessä ylläpito-/ jatkokehittelyprosessissa. Tutkimukselle asetettiin kolme tutkimuskysymystä: Millaista on videopelien ylläpito? Millaisia ongelmia esiintyy videopelien ylläpidossa? Millaisia hyötyjä ja ongelmia ketterien menetelmien ja käytänteiden käytöstä videopelien ylläpidossa on koettu?

Tutkielmassa tarkasteltiin ensin, mitä tarkoitetaan pelillä, millaisia pelejä on olemassa, mitä rooleja liittyy peleihin sen elinkaaren aikana, miten pelikehitys on jäsennettävissä sekä millaisia ongelmia pelikehityksessä on koettu. Peli voidaan määritellä järjestelmäksi, jossa pelaajat osallistuvat teennäiseen konfliktiin, joka on määritelty säännöin ja jolla on laskettavissa oleva lopputulos. Pelit voidaan luokitella genreihin, kuten räiskintäpeleihin, strategiapeleihin ja simulaatiopeleihin. Pelialan toimijoiden osalta voidaan tunnistaa kehittäjät, julkaisijat ja kuluttajat. Yleisellä tasolla pelikehitysprosessi jakaantuu konseptin valmistelu-, esituontanto-, tuotanto-, laadunvarmistus-, julkaisu- ja ylläpitovaiheisiin. Vaikka laadunvarmistus erotetaankin omaksi kokonaisuudeksi, tapahtuu sitä koko pelikehitysprosessin ajan, johtuen pelikehityksen iteratiivisesta luonteesta. Pelikehityksen yhteydessä ilmaantuu monenlaisia ongelmia, esimerkiksi mittakaavan määrittelyssä, ominaisuuksien kasvussa ja karsimisessa, viivästyksissä ja dokumentaation puutteessa.

Toiseksi tutkielmassa selvitettiin, mitä tarkoitetaan ketterällä lähestymistavalla ohjelmistokehityksessä ja millaisia ovat kolme ketterää

menetelmää (Scrum, XP ja Kanban). Samassa yhteydessä kuvattiin myös, miten ohjelmiston ylläpito jäsennetään , mitkä ovat sen tyypilliset ongelmat sekä miten ohjelmiston ylläpito voidaan tehdä ketterän lähestymistavan mukaisesti. Ketterällä lähestymistavalla tarkoitetaan ohjelmistokehityksessä toimintaa, joka on Agile-manifestissa esitettyjen arvojen mukaista. Ketterien menetelmien ydinajatuksena on luoda muutosta sekä vastata muutokseen. Näistä menetelmistä Scrum on empirismiin perustuva iteratiivinen ja inkrementaalinen lähestymistapa, jolla voidaan optimoida ennakoitavuutta ja kontrolloida riskejä. Iteratiivisuutta ja inkrementaalisuutta Scrumissa toteutetaan jakamalla prosessi sprinteiksi kutsuttuihin osiin. XP puolestaan on kevyt ohjelmistokehitysmenetelmä, joka pyrkii vastaamaan tilanteisiin, joissa vaatimukset muuttuvat usein ja hyvinkin nopeasti. Kanban puolestaan on ketterään kehittämiseen kuuluva ajatusmalli, joka perustuu Lean-filosofiaan. Se ei tarjoa tukea projektinhallintaan eikä se ole varsinainen ohjelmistokehitysmenetelmä. Parhaiten Kanbania voidaan kuvata eräänlaiseksi työvirranhallintajärjestelmäksi.

Kolmanneksi tutkielmassa kuvattiin ketterää pelikehitystä. Pelikehityksen parissa osa Agile-arvoista voidaan mukauttaa, näkemyksestä riippuen, vastaamaan paremmin pelikehityksen tarpeita. Muutokset johtuvat siitä, että pelin voidaan nähdä olevan enemmän kuin pelkkä ohjelmisto ja sidosryhmien väliset suhteet ovat erilaiset kuin perinteisessä ohjelmistokehityksessä. Mitään yhtenäistä vaihejakoa ketterälle pelikehitykselle ei ole. Yhden näkemyksen mukaan ketterään pelikehitykseen kuuluvat konsepti-, esituotanto-, tuotanto- ja jälkituotantovaiheet. Jossain tapauksissa konsepti- ja esituotantovaihe voidaan nähdä esituotantovaiheena, jälkituotantovaiheen tehtäviä on siirretty tuotantovaiheeseen ja jälkituotanto on korvattu projektin päättämisellä. Eroista huolimatta vaiheet muistuttavat tarkoituksen ja sisällön osalta toisiaan. Ketterien menetelmien hyödyntäminen pelikehityksen eri vaiheissa vaihtelee käytettävän menetelmän ja menetelmän esittäjän omien näkemyksien perusteella. Pelikehityksen vaihe vaikuttaa paljon siihen, millaisia ketteriä käytänteitä sovelletaan.

Edellä raportoidussa kirjallisuuskatsauksessa päädyttiin siihen, että ketterästä pelien ylläpidosta on aiemmin tuskin lainkaan tehty tutkimusta. On ollut jopa epäselvyyttä siitä, minkä luonteista ylläpito on pelien yhteydessä. Asian valottamiseksi tehtiin tässä tutkimuksen yhteydessä haastattelututkimus, jossa haastateltiin neljän pelistudion henkilöitä. Pelistudiot oli valittu kehitysalustan ja koon mukaisesti. Tämä johtui siitä, että haluttiin nähdä, kuinka kehitysalusta ja pelistudion koko vaikuttavat videopelien ylläpitoon / jatkokehittelyyn. Tutkimuksen pohjaksi rakennettiin

tutkimusasetelma, ja siitä johdettu haastattelurunko puolistrukturoituine kysymyksineen.

Haastattelututkimuksen tulosten perusteella vaikuttaa siltä, että pelien ylläpito mielletään enemmän jatkokehittelyksi kuin perinteisen ohjelmistotuotannon tarkoittamaksi ylläpidoksi. Sisällöllisesti ylläpito / jatkokehittely vastaa Mannisen ym. (2006) esittämää kuvausta ylläpitovaiheen sisällöstä. Prosessillisesti pelien jatkokehittely jakaantuu kaksilinjaiseksi, joista yhdessä linjassa tuotetaan lisäsisältöä ja toisessa linjassa huolehditaan ylläpidollisista tehtävistä. Pelien elinkaaren osalta voidaan sanoa, että esitys ohjelmiston elinkaaresta ei vastaa peliteollisuuden todellisuutta, mistä johtuen olisikin tarpeellista luoda erillinen malli pelien ylläpidon elinkaaresta.

Ongelmien osalta ylläpito / jatkokehittely on samankaltaista verrattuna pelikehitykseen. Eniten ongelmia esiintyi erityisesti mittakaavan määrittelyssä, ominaisuuksien kasvussa, suunnitteluvaiheessa ja viivästyksissä. Lisäksi esiintyi myös teknisiä ja ohjelmisto- sekä testausongelmia. Nämä tulokset ovat linjassa Petrillon ym. (2009) tekemän tutkimuksen kanssa, jossa suurimmiksi ongelmiksi todettiin mittakaavaan määrittely ja ominaisuuksien kasvu. Myös suunnitteluvaiheen ongelmat, viivästykset ja tekniset ongelmat olivat yleisiä Petrillon ym. (2009) suorittaman tutkimuksen mukaan. Tulokset ovat myös linjassa Kanoden ja Haddadin (2009) tutkimuksen kanssa, jossa nostettiin esille suunnitteluvaiheen ongelmat ja tekniset ongelmat.

Ketterät menetelmät koettiin hyödyllisiksi, koska pelikehityksessä voidaan joutua toimimaan osittaisessa tietämättömyyden tilassa, jolloin joustavat menetelmät ovat tärkeitä. Hyödyiksi koettiin muun muassa yleisen toiminnan organisoinnin helpottuminen ja parantuminen. Erilaisten työlistojen taas koettiin selkeyttävän aikataulujen ja tehtävien suhteuttamista toisiinsa. Tuottajien näkökulmasta taas ketterät menetelmät helpottivat estimointia, lisäsivät stabiiliutta ja vähensivät rutistusajanjaksojen tarvetta. Tulokset ovat linjassa muun muassa Poolen ym. (2001) tutkimuksen kanssa, jossa ketterien menetelmien eduksi nähtiin muutoksiin vastaaminen. Tulokset ovat linjassa myös Svenssonin ja Höstin (2005) tutkimuksen kanssa, jossa linjattiin hyödyksi tiedon välittymisen parantumista. Seikolan ym. (2011) tutkimuksessa esitettiin ketterien menetelmien hyödyksi tiimiin liittyviä etuja, mikä voitiin nähdä tässä tutkimuksessa yleisen toiminnan organisoinnin helpottumisena.

Ketterien menetelmien tehokas hyödyntäminen koettiin kuitenkin haastavaksi. Esimerkiksi Scrum ja sen vaatima käytänteiden kirjaimellinen noudattaminen koettiin haastavaksi. Erityisesti estimointi- ja suunnittelutapaamisten määrän koettiin syövän paljon aikaa itse kehitystyöltä. Tämä tulos on myös linjassa Millerin (2008) artikkelin kanssa, jossa tuodaan esille esimerkiksi päivittäisten Scrum-tapaamisten haasteellisuutta johtuen

potentiaalisesta ajan tuhlaamisesta ja niistä saatujen hyötyjen kyseenalaisuudesta. Lisäksi artikkelissa nostettaan esille estimointitapaamiset, juuri ajanhukasta johtuen, mikä nousi esille myös tässä tutkimuksessa.

Toiminnan kehittämisen kannalta todettiin, että on tärkeää nähdä käytettävät menetelmät ja käytänteet työkaluina, jotka taipuvat pelistudion omiin tarpeisiin. Projektiryhmien tarpeet tulee myös ottaa huomioon, eikä ryhmiä tule pakottaa käyttämään samoja menetelmiä ja käytänteitä, vaan käytänteet tulee valita ryhmäkohtaisesti. Ylläpito ja / jatkokehittely nähtiin tärkeäksi ja oleelliseksi osaksi pelikehitysprosessia, ja se tulisi ottaa huomioon heti peliprojektin alussa. Yhdeksi syyksi siteerattiin julkaisun jälkeisiä ajallisia paineita, joista johtuen ei mahdollisesti enää ole aikaa miettiä ylläpitoa / jatkokehittelyä tarkemmin. Toimijoiden osalta ylläpitoon / jatkokehittelyyn olisi hyvä asettaa toinen projektitiimi, joka vastaa pääasiallisesti ylläpitoon / jatkokehittelyyn liittyvistä tehtävistä. Yksi mahdollisuus on sijoittaa pelistudion kärkitiimi kehitystehtävien pariin ja kokemattomampi junioritiimi ylläpidon / jatkokehittelyn pariin. Tämä on kuitenkin hankalaa muun muassa henkilöstöllisten rajoitteiden, kuten pätevien henkilöiden löytämisen takia.

Tämän tutkimuksen tuloksia voivat hyödyntää esimerkiksi pelistudiot, jotka haluavat lisätietoa pelien ylläpidosta / jatkokehittelystä niiden kehittämiseksi. Tutkimus tarjoaa myös hyviä lähtökohtia jatkotutkimuksen tekemiselle.

Tämän tutkimuksen yhteydessä on esitetty yksi näkemys siitä, millaista pelien jatkokehittely on perustuen neljän eri yrityksen edustajien haastatteluihin. Haastattelun kohteena olleiden yritysten määrä on hyvin pieni, mistä johtuen yleistyksiä pelien jatkokehittelyn tarkasta sisällöstä ei voida tehdä. Tuloksia voidaan pitää kuitenkin suuntaa antavina ja niiden pohjalta tutkimusta voidaan jatkaa. Tutkimukselle luonnollinen jatkotutkimusaihe olisi siirtyä tarkastelemaan millaista pelien ylläpito ja jatkokehittely on maailmanluokan pelistudioiden sisällä. Yhtenä mahdollisena aiheena olisi myös lähteä luomaan peliteollisuuden todellisuuksiin perustuvat kuvaukset ylläpidon alalajeista, ylläpidon ja pelikehityksen suhteesta sekä pelien elinkaaresta. Tässä tutkimuksessa ei ole myöskään päästy tutkimaan, missä määrin pelin genre vaikuttaa ylläpitoon ja jatkokehittelyyn. Esimerkiksi erilaisten verkossa pelattavien MMORPG-pelien jatkokehittelyä ja ylläpitoa olisi kiintoisaa tutkia.

LÄHTEET

Abrahamsson, P., Salo, O., Ronkainen, J. & Warsta J. (2002). Agile software development methods: Review and analysis. *VTT Publications, 478*. Haettu 4.2.2015 osoitteesta http://www.vtt.fi/inf/pdf/publications/2002/P478.pdf

Agarwal, R. & Umphress, D. (2008) Personal extreme programming for a single person team. Teoksessa *Proceedings of the 46th Annual Southeast Regional Conference on XX* (ACM-SE 46), 82-87.

Ahmad, M., O., Markkula, J., & Oivo, M. (2013). Kanban in software development: A systematic literature review. Teoksessa *Proceedings of 39th Euromicro Conference Series on Software Engineering and Advanced Applications*, 9-16.

Ampatzoglou, A. & Stamelos, I. (2010) Software engineering research for computer games: A systematic review. *Information and Software Technology*, 52(9) : 888-901

Analysis.Net Research. (2015). *State of agile survey 2015* VersionOne. Haettu 3.4.2015 osoitteesta http://scrumgroup.org/wp-content/uploads/2015/03/state-of-agile-development-survey-ninth.pdf

Anderson, D., J. (2010). *Kanban - Successful Evolutionary Change for Technology Organizations*. US: Blue Hole Press

Anderson, D. J., Concas, G., Lunesu, M. I., Marchesi, M., & Zhang, H. (2012). A comparative study of Scrum and Kanban approaches on a real case study using simulation. Teoksessa *Agile Processes in Software Engineering and Extreme Programming* (pp. 123-137). Berlin: Springer Heidelberg.

April, A. Abran, A. (2012). *Software Maintenance Management – Evaluation and Continuous Improvement.* USA: John Wiley & Sons, Inc.

Beck, K. & Andres, C. (2004). *Extreme programming explained: Embrace change*. (2. painos). Boston: Addison-Wesley Professional.

Beck, K., Grenning, J., Martin, R. C., Beedle, M., Highsmith, J., Mellor, S. ym. (2001). Manifesto for agile software development. Haettu 04.2.2015 osoitteesta http://agilemanifesto.org/

Choudhari, J. & Suman, U. (2010). Iterative Maintenance Life Cycle Using eXtreme Programming. Teoksessa *International Conference on Advances in Recent Technologies in Communication and Computing, ARTCom, October 16-17* (pp. 401-403). Los Alamitos: IEEE Computer Society.

Cockburn, A. (2002). Agile Software Development. Boston, MA, USA: Addison-Wesley Professional.

Dekleva, S. M. (1992). Delphi Study of Software Maintenance Problems. Teoksessa *Proceedings of the IEEE Conference of Software Maintenance (ICSM 1992)*, 10-17.

Figma Ry. (2011, 3. helmikuuta). *Suomen pelimarkkinoiden kehitys.* Haettu 1.3.2015 osoitteesta http://www.figma.fi/index.php/tilastot

Flood K. 2003. Game unified process. Haettu 16.1.2016 osoitteesta http://www.gamedev.net/reference/articles/article1940.asp

Gibson A. 2007. Agile game development and fun. Haettu 16.1.2016 osoitteesta http://programmedevelopment.com/public/uploads/files/agile_game_development_and_fun.pdf

Godoy A., Barbosa E. 2010. GameScrum: an approach to agile gamedevelopment. Teoksessa Proceedings of SBGames 2010 Conference (s. 292-295).

Highsmith, J. & Cockburn, A. (2001). Agile software development: The business of innovation. *Computer*, *3*4(9), 120-127.

Hirsjärvi, S. & Hurme, H. (2000) *Tutkimushaastattelu.* Helsinki: Helsinki University Press.

Hirsjärvi, S., Remes, P. & Sajavaara, P. (2004). *Tutki ja kirjoita.* (10. osin uudistettu painos). Helsinki: Kusannusosakeyhtiö Tammi.

ISO/IEC. (2006). *Software Engineering – Software Life Cycle Processess – Maintenance.* Haettu 17.2.2015 osoitteesta http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=1703974

ISO/IEC. (2008). *Systems and software engineering – Software life cycle processes.* Haettu 4.6.2015 osoitteesta http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4475826

Jain, N. (2006). Offshore agile maintenance. Teoksessa *Proceedings of Agile 2006 Conference, July 23-28* (pp. 7, 336). Los Alamitos: IEEE Computer Society.

Jansz, J. Martens, L. (2005). Gaming at a LAN event: The social context of playing video games. *New Media & Society, 7*(3), 333-355.

Kajko-Mattsson, M. (2008). Problems in agile trenches. Teoksessa *Proceedings of the Second ACM-IEEE international symposium on Empirical software engineering and measurement, ESEM '08* (pp. 111-119) New York: ACM.

Kanode, C. M. & Haddad, H. M. (2009). Software engineering challenges in game development. Teoksessa Latifi, S. (toim.), *Information Technology: New Generations, 2009. ITNG '09. Sixth International Conference on. Las Vegas, Nevada. 27.-29. huhtikuuta.* (s. 260-265). Los Alamitos: IEEE Computer Society.

Kitchenham, B. & Pfleeger, S. (2001). Principles of survey research. Part 1: Turning lemons in lemonade. *SIGSOFT Software Engineering Notes* 26(6), 16-18.

Kitchenham B., Travassos G., von Mayrhauser A., Niessink F. Schneidewind N., Singer J. & Yang H. (1999). Towards an ontology of software maintenance. *Journal of Software Maintenance 11(6),* 365-389.

Kuittinen, I., Virtala, T., Hyvärinen, L., Heikkinen, H., Lyytikäinen, M., Mäkinen, J., ym. (2010). *Suomen pelitoimialan strategia 2010-2015 - visio 2020*. Helsinki: Neogames, Suomen Pelinkehittäjät ry. Haettu 1.3.2015 osoitteesta http://www.neogames.fi/wp-content/uploads/2013/05/Pelistrategia-2010-2015.pdf

Laanti, M. (2013). *Agile Methods in Large-Scale Software Development Organizations – Applicability and Model for Adoption.* Dissertation Thesis, University Of Tampere, Tampere: Juvenes Print.

Laanti, M., Similä, J., Abrahamsson, P. (2013). Definitions of agile software development and agility. Teoksessa F. McCaffery, R. V. O'Connor, R. Messnarz (toim.), *Systems, Software and Services Process Improvement*, EuroSPI 2013 (s.247-258). Berling: Springer-Verlag.

Manninen, T., Kujanpää, T., Vallius, L., Korva, T. & Koskinen, P. (2006). *Game production process: A preliminary study* (Raportti v. 1.0 – 28.2.2006). Oulu: University of Oulu, Department of Information Processing Science, LudoCraft.

Miller P. (2008) Top 10 pitfalls using Scrum methodology for video game development. http://www.gamasutra.com/view/feature/3724

Musil, J., Schweda, A., Winkler, D. & Biffl, S. (2010a). Improving video game development: Facilitating heterogenous team collaboration through flexible software processes. Teoksessa Riel, A., O'Connor, R., Tichkiewitch, S. & Messnarz, R. (toim.), *Proceedings of the 17th European Conference on System, Software and Services Process Improvement. Grenoble, Ranska. 1.-3. syyskuuta, 2010.* (s. 83-94). Berlin: Springer.

Musil, J., Schweda, A., Winkler, D., Biffl, S. (2010b): A survey on a state of the practice in video game development, Report IFS-QSE 10/04. Vienna: Institute of Software Technology and Interactive Systems.

Mäyrä, F., Sihvonen, T., Saarenpää, H., Kultima, A., Paavilainen, J., Stenros, J., ym. (2009). *Pelitieto - pelien peruskurssi*. Haettu 10.3.2015 osoitteesta http://pelitieto.net/

Niessink, F., Van Vliet, H. (2000). Software maintenance from a Service Perspective. *Journal of Software Maintenance: Research and Practice.* 12(2): 103-120.

O'Reilly, T. (1999). Lessons from Open Source Software Development. *Communications of the ACM,* 42(4): 32-37

Palmer, S. R. & Felsing, J. M. (2002). A Practical Guide to Feature-Driven Development. Upper Saddle River, NJ, Prentice-Hall.

Peltoniemi, M. (2009). *Industry life-cycle theory in cultural domain: Dynamics of the games industy*. Väitöskirja. Tampereen Teknillinen Yliopisto. Haettu 1.3.2015 osoitteesta http://dspace.cc.tut.fi/dpub/bitstream/handle/123456789/223/peltoniemi.pdf

Petrillo, F., Pimenta, M. Is Agility out there? Agile Practices in Game Development. Teoksessa *Proceedings of the 28th ACM International Conference on Design of Communication (SIGDOC '10)(s. 9-15)*. New York, USA.

Petrillo, F., Pimenta, M., Trindade, F. & Dietrich, C. (2009). What went wrong?; A survey of problems in game development. *Computers in Entertainment*, 7(1), 1-22.

Poole C. J., Murphy, T., Huisman, J. W., Higgins, A. (2001) Extreme maintenance. Teoksessa *Proceedings of IEEE International Conference on Software Maintenance (s. 301-309).*

Prochazka, J. (2011). Agile Support and Maintenance of IT Services. Teoksessa J. Pokorny, et al. (eds.), *Information Systems Development*, (pp. 597-609). New York: Springer.

Rajlich, V.T. & Bennett, K.H. (2000). A staged Model for the Software Life Cycle. *Computer, 33(7),* 66-71.

Runeson, P. & Höst, M. (2009). Guidelines for conducting and reporting case study research in software engineering. *Empirical Software Engineering* 14(2), 131-164.

Salen, K & Zimmerman, E. (2003). Rules of play. US: The MIT Press.

Schwaber, K. 1995. SCRUM Development Process, OOPSLA'95 workshop. Haettu 18.4.2015 osoitteesta http://agilix.nl/resources/scrum_OOPSLA_95.pdf

Schwaber, K., Sutherland, J. (2013) The Scrum Guide. Haettu 6.4.2015 osoitteesta http://www.scrumguides.org/docs/scrumguide/v1/Scrum-Guide-US.pdf#zoom=100

Seikolam M., Loisa, H., Jagos, A. (2011). Kanban Implementation in a Telecom Product Maintenance. *37th EUROMICRO Conference on Software Engineering and Advanced Applications (SEAA), (s.321-329).* Los Alamitos: IEEE Computer Society

Senapathi, M. & Srinivasan, A. (2011). Understanding Post-Adoptive Agile Usage – an Exploratory Cross-Case analysis. Teoksessa *Agile Conference (AGILE)*, 2011 (s. 117-126). Salt Lake City, UT.

Smith, T., Cooper, K. M. L. , Longstreet C. S. (2011). Software Engineering Senior Design Course: Experiences with Agile Game Development in a Capstone Project. Teoksessa *Proceedings of the 1st Internatiuonal Workshop on Games and Software Engineering (GAS '11)(s. 9-12).* New York, USA.

Sommerville, I. (2011). Software Engineering (9.painos). US: Addison-Wesley.

Stapleton, J. (1997). Dynamic systems development method – The method in practice. Addison Wesley.

Svensson, H. Höst, M. (2005). Introducing an Agile Process in a Sfotware Maintenance and Evolution. Teoksessa *Proceedings of the Ninth European Conference on Software Maintenance and Teengineering (CSMR'05) (s. 256-264).* Manchester, Yhdistynyt kuningaskunta.

SWEBOK. (2014). *Guide to the Software Engineering Body of Knowledge.* IEEE. Haettu 05.10.2015 osoitteesta http://www.computer.org/web/swebok/v3

Vorderer, P., Hartmann, T., Klimmt, C. (2003). Explaining the enjoyment of playing video games: the role of competition. Teoksessa *Proceedings of the second international conference on Entertainment computing* (ICEC '03), 1-9.

Ye, Z. (2004). Genres as a tool for understanding and analyzing user experience in games. Teoksessa Dykstra-Erickson, E. & Tscheligi, M. (toim.), *CHI '04 extended abstracts on Human factors in computing systems. Vienna, Austria. 24.-29 huhtikuuta.* (s. 773-774). New York, NY: ACM.

Yin, R.K. (2009). *Case study research: Design and methods* (5. painos). Sage Publishing, Beverly Hills CA

Wikipedia. (2015, 22. helmikuuta). *Video game genres*. Haettu 10.3.2015 osoitteesta http://en.wikipedia.org/wiki/Video_game_genres

Wikipedia. (2015a, 11. lokakuuta). *Remedy Entertainment.* Haettu 25.11.2015 osoitteesta https://en.wikipedia.org/wiki/Remedy_Entertainment

Wikipedia. (2015b, 17. marraskuuta). *Rovio Entertainment.* Haettu 25.11.2015 osoitteesta https://en.wikipedia.org/wiki/Rovio_Entertainment

Wikipedia. (2015c, 25. marraskuuta). *Supercell.* Haettu 25.11.2015 osoitteesta https://en.wikipedia.org/wiki/Supercell_(video_game_company)

Wikipedia. (2015d. 30.heinäkuuta). *Redlynx.* Haettu 25.11.2015 osoitteesta https://en.wikipedia.org/wiki/RedLynx

Wikipedia. (2015e. 17 marraskuuta 2015). *Activision Blizzard.* Haettu 25.11.2015 osoitteesta https://en.wikipedia.org/wiki/Activision_Blizzard

LIITE 1 SAATE 1

Arvoisa vastaanottaja,

Olen tekemässä tietojärjestelmätieteen pro gradu -tutkielmaa aiheesta "Ketterien menetelmien hyödyntäminen videopelien julkaisun jälkeisessä ylläpito/-jatkokehitysprosessissa". Ohjaajani on lehtori Mauri Leppänen, KTT Jyväskylän yliopistosta. Tarkoituksenani on selvittää millä tavoin ketteriä menetelmiä voidaan hyödyntää videopelien julkaisun jälkeisessä ylläpito-/jatkokehitysprosessissa. Tarkemmin tutkimuksessa perehdytään seuraaviin asioihin: millaista videopelien ylläpito/jatkokehitys tarkalleen on, millaisia ongelmia siinä esiintyy ja millaisia hyötyjä ja ongelmia ketterien menetelmien ja käytänteiden käytöstä videopelien ylläpidossa/jatkokehityksessä on koettu. Tutkielmassa tarkastellaan myös mm. seuraavia asioita: pelikehityksen vaiheita, eri vaiheissa hyödynnettäviä menetelmiä ja käytänteitä ja pelikehityksen ongelmia. Tutkielmaan sisältyy haastattelututkimus, jonka kohteena ovat suomalaiset pelistudiot.

Pyytäisin, että yrityksenne osallistuisi haastatteluun. Haastateltavana voisi toimia henkilö, jolla on hyvä tuntemus yrityksenne pelikehitysprosessista ja käyttämistänne kehitysmenetelmistä, erityisesti ketteristä menetelmistä. Haastatteluun osallistumalla autatte keräämään tärkeää tietoa videopelien ylläpidosta/jatkokehityksestä ja ketterien menetelmien hyödyllisyydestä. Tutkimuksen valmistuttua lähetän teille työn, jonka pohjalta voitte verrata toimintatapojanne muiden haastatteluun osallistuvien toimintatapoihin. Samalla saatte uusinta tutkimustietoa aiheesta, koska työhön sisältyy laaja kirjallisuuskatsaus.

Haastattelu olisi tarkoitus suorittaa kasvokkain yrityksenne edustajan kanssa. Jos haastattelun toteuttaminen kasvokkain ei teidän tapauksessanne käy, voidaan haastattelut toteuttaa myös Skype-ohjelman välityksellä. Haastatteluun osallistumiseen kuluu noin 60 minuuttia. Löytyisikö yrityksenne edustajalta tämän verran aikaa osallistua haastatteluun lokakuun aikana?

Olen valmis vastaamaan kaikkiin asiaa koskeviin kysymyksiin.

Kiittäen,
Eetu Vilhunen
eetu.k.vilhunen@student.jyu.fi
+358408235032

LIITE 2 SAATE 2

Hei!

Tiedustelin aiemmin osallistuisiko teidän yrityksenne pro gradu -tutkielmaani haastatteluosuuteen aiheesta "Ketterien menetelmien hyödyntäminen videopelien julkaisun jälkeisessä ylläpito/-jatkokehitysprosessissa". Etsin vielä haastateltavia ja jos yrityksenne olisi kiinnostunut osallistumaan tutkielmani haastatteluosuuteen, tarvitsisin vain noin 60 minuutta aikaa henkilöltä jolla on hyvä tuntemus yrityksenne pelikehitystprosessista ja käyttämistänne kehitysmenetelmistä, erityisesti ketteristä menetelmistä.

Olen valmis vastaamaan kaikkiin asiaa koskeviin kysymyksiinne.

Kiittäen
Eetu Vilhunen
eetu.k.vilhunen@student.jyu.fi
+358408235032

LIITE 3 HAASTATTELURUNKO, OSA A

I TAUSTATIEDOT

1. Minkä ikäinen yrityksenne on? Kuinka monta peliä yrityksessänne on kehitetty tähän mennessä?
2. Kuinka paljon yrityksessänne on työntekijöitä?
3. Kuinka paljon samanaikaisia projekteja yrityksessänne on tällä hetkellä? Vaihteleeko projektien määrä paljonkin?
4. Mille alustoille yrityksenne tuottaa pelejä?
5. Mitä genrejä yrityksenne pelit pääasiallisesti edustavat?

II MILLAISTA ON VIDEOPELIEN YLLÄPITO

1. Miten jäsentäisitte pelien kehitysprosessin teidän yrityksessä?

- Jäsentyykö se julkaisun jälkeiseksi, pelikehitysprossessin ulkopuoliseksi prosessiksi?
- Vai onko se osa pelikehitysprosessia? Eli nähdäänkö julkaistu versio esimerkiksi 1.0 versiona, jota lähdetään julkaisun jälkeen jatkokehittelemään?
- Kirjallisuudessa kuvataan yhdenlaista ylläpitoprosessia seuraavasti: alustava kehittely → evoluutio, versio 1 → huolto, versio 1 (korjaukset) → luopuminen, versio 1 → sulkeminen, versio 1 ja siirtyminen versioon 2. Toisinsanoen ylläpito voidaan nähdä kehityksen jälkeen alkava ajanjakso jolloin julkaistua versiota ylläpidetään niin kauan, että voidaan ja ollaan valmiita siirtymään uudempaan versioon ohjelmistosta.
- Poikkeaako menettely mahdollisesti peli genrejen tai alustojen mukaan?
  - Mitkä ovat kriittisimmät vaikuttavat tekijät tässä suhteessa?
- *Haastattelijan vastauksista riippuen: Jos ylläpito nähdään erillisenä, jatketaan tämän dokumentin mukaisesti. Jos taas koetaan, että ylläpitoa ei varsinaisesti ole vaan jatketaan pelikehitysprosessia, siirrytään osaan B.*

2. Ylläpidossa on perinteisesti tunnistettu seuraavat luokat: mukauttava, korjaava, kiireellinen, estävä ja täydentävä.

- Missä määrin mitäkin näistä luokista teidän yrityksenne pelien ylläpidossa esiintyy?

- Kuinka ne ilmenevät pelien ylläpidossa teidän yrityksessä?
- Tunnistatteko mahdollisesti muita mainitsemattomia luokkia, jotka ovat erityisen tärkeitä pelien ylläpidossa?

3. Miten kuvailisitte pelin elinkaarta teidän yrityksessä?

- Kirjallisuudessa ohjelmiston elinkaari jäsennetään seuraavasti: varhaislapsuus, murrosikä, aikuisuus ja seniiliys. Missä määrin koette pelien elinkaaren olevan edellä mainitun jäsennyksen mukainen?
- Olisiko teidän mielestä jokin muunlainen jäsennys parempi pelien kohdalla?
- Mitkä seikat vaikuttavat ohjelmiston elinkaareen?

4. Miten jäsentäisitte pelien ylläpidon teidän yrityksessä?

- Onko ylläpito jäsennelty yrityksessänne ketteränä prosessina? Kirjallisuudessa ketterä ylläpito voi sisältää esimerkiksi seuraavia tehtäviä: Analyysi, suunnittelu, rakenteen muutos, muutosten toteutus, järjestelmätestaus, vastaanottotarkastus ja toimitus.

5. Missä määrin yrityksenne hyödyntää ketteriä käytänteitä ylläpidossa?

- Käydään läpi liitteen 1 mukainen käytännelista
- Hyödyntääkö yrityksenne mahdollisesti työvirran visualisointia, esimerkiksi Kanban-taulua?

6. Muita ylläpitoon liittyviä kysymyksiä

- Miten kauan pelin ylläpito yleisesti ottaen kestää teidän yrityksessä?
  - Mitkä seikat vaikuttavat tähän erityisesti?
- Mitkä seikat vaikuttavat yleisesti ylläpitoon?

III MILLAISIA ONGELMIA ESIINTYY VIDEOPELIEN YLLÄPIDOSSA TOIMINNALLISESTA NÄKÖKULMASTA KATSOTTUNA?

1. Mitkä asiat nousevat ensimmäisenä mieleen ongelmina pelien ylläpitoon liittyen?
2. Kirjallisuudessa on esitetty ylläpidon ongelmiksi seuraavia ongelmia (*Liite 2: Ylläpidon ongelmat),* miten koette niiden esiintyvän pelien ylläpidossa ja kuinka vakavia ne ovat?
3. Kirjallisuudessa on esitetty pelikehitykselle ongelmiksi seuraavia ongelmia (*Liite 3: Pelikehityksen ongelmat)*, miten koette niiden esiintyvän pelien ylläpidossa ja kuinka vakavia ne onvat?

IV MILLAISIA HYÖTYJÄ JA ONGELMIA KETTERIEN MENETELMIEN JA KÄYTÄNTEIDEN KÄYTÖSTÄ VIDEOPELIEN YLLÄPIDOSSA ON KOETTU?

1. Mitä hyötyjä koette olevan ketterien menetelmien ja käytänteiden käytöstä pelien ylläpidon parissa?
2. Mitä haasteita koette ketterien menetelmien ja käytänteiden tuovan pelien ylläpitoon?

V MITEN PELIKEHITYSTÄ VOISI KEHITTÄÄ YLLÄPIDON OSALTA? MIKÄ ROOLI KETTERYYDELLÄ OLISI TÄSSÄ?

1. Olisiko mielestänne syytä lähestyä ylläpitoa eri näkökulmasta?
2. Kuinka kehittäisitte nykyistä pelien ylläpitoprosessianne?

• Miten mielestänne ketteryyttä voisi hyödyntää tässä suhteessa?

3. Kuinka kehittäisitte pelien ylläpitoa toimijoiden osalta?
4. Kuinka kehittäisitte pelien ylläpitoa käytänteiden osalta?

LIITE 4 HAASTATTELURUNKO, OSA B

I YLLÄPITOA EI VARSINAISESTI OLE, VAAN JATKETAAN PELIKEHITYSPROSESSIA

1. Miten jäsentäisitte pelien kehitysprosessin teidän yrityksessä? Onko kehitysprosessi jäsennelty ketteränä prosessina?
- Kirjallisuudessa ketterä pelikehitys jäsennetään esimerkiksi seuraavasti: konseptivaihe, esituotantovaihe, tuotantovaihe ja jälkituotantovaihe, joiden sisällä prosessia toteutetaan esimerkiksi scrumin mukaisesti sprintteinä. Jäsentyykö pelien kehitysprosessi edellä mainitulla tavalla vai mahdollisesti täysin eri tavoin?
- Kuinka jäsentelyyn ollaan päädytty?
2. Muuttuuko pelikehitysprosessi mitenkään julkaisun jälkeen?
- Lisätäänkö prosessiin mahdollisesti uusia aktiviteettejä vai poistetaanko niitä?
- Tapahtuuko muutoksia henkilöstörakenteissa?
3. Miten kuvailisitte pelin elinkaarta teidän yrityksessä?
- Kirjallisuudessa ohjelmiston elinkaari jäsennetään seuraavasti: varhaislapsuus, murrosikä, aikuisuus ja seniiliys. Missä määrin koette pelien elinkaaren olevan edellä mainitun jäsennyksen mukainen?
- Olisiko mielestänne jokin muunlainen jäsentely parempi pelien kohdalla?
- Mitkä seikat vaikuttavat pelien elinkaareen?
4. Missä määrin yrityksenne hyödyntää ketteriä käytänteitä pelien kehityksessä?
- Käydään läpi liitteen 1 mukainen käytännelista
- Hyödyntääkö yrityksenne mahdollisesti työvirran visualisointia, kuten esimerkiksi Kanban-taulua?
5. Muita pelikehitykseen liittyviä kysymyksiä
- Mitkä seikat vaikuttavat yleisesti pelikehitykseen
- Kuinka kauan pelin kehitysprosessi yleisesti ottaen kestää teidän yrityksessä? Mitkä seikat vaikuttavat tähän erityisesti?

II MILLAISIA ONGELMIA ESIINTYY PELIEN KEHITYKSESSÄ TOIMINNALLISESTA NÄKÖKULMASTA KATSOTTUNA?

1. Mitkä seikat nousevat ensimmäisenä mieleen ongelina pelien kehitykseen liittyen?
2. Kirjallisuudessa on esitetty pelikehitykselle seuraavia ongelmia (*Liite 3: Pelikehityksen ongelmat).* Miten koette niiden esiintyvän pelien kehityksessä ja kuinka vakavia ne ovat?
3. Ilmenevätkö ongelmat yleisellä tasolla vai kenties vasta julkaisun jälkeen? Mitkä ongelmat esiintyvät missäkin vaiheessa?

III MILLAISIA HYÖTYJÄ JA ONGELMIA KETTERIEN MENETELMIEN JA KÄYTÄNTEIDEN KÄYTÖSTÄ VIDEOPELIEN YLLÄPIDOSSA ON KOETTU?

1. Mitä hyötyjä koette olevan ketterien menetelmien ja käytänteiden käytöstä pelikehityksen parissa?
2. Mitä haasteita koette ketterien menetelmien ja käytänteiden tuovan pelikehitykseen?
3. Miten pelikehityksessä menossa oleva vaihe vaikuttaa käytänteiden hyödyllisyyteen tai ongelmallisuuteen?

IV MITEN PELIKEHITYSTÄ VOISI KEHITTÄÄ YLLÄPIDON/JATKOKEHITTELYN OSALTA? MIKÄ ROOLI KETTERYYDELLÄ OLISI TÄSSÄ?

1. Olisiko mielestänne syytä lähestyä ylläpitoa/jatkokehittelyä eri näkökulmasta?
2. Kuinka kehittäisitte nykyistä pelien ylläpito/jatkokehittelyprosessia?
3. Kuinka kehittäisitte pelien ylläpitoa/jatkokehittelyä toimijoiden osalta? Olisiko ylläpito/jatkokehittely parasta järjesteää kenties erillisellä tiimillä tai mahdollisesti pelin kehittäneen tiimin toimesta?
4. Kuinka kehittäisitte pelien ylläpitoa/jatkokehittelyä käytänteiden osalta? Olisiko ketterien menetelmien käytänteet hyödyllisiä myös pelien ylläpidossa?

LIITE 5 KÄYTÄNNELISTA

Käytänne	**Missä määrin hyödynnetty (0 ei ollenkaan – 5 hyvin usein)**					
Tuotteen työlista	0	1	2	3	4	5
Sprintit	0	1	2	3	4	5
Sprintin työlista	0	1	2	3	4	5
Suunnittelutapaamiset	0	1	2	3	4	5
Katselmoinnit	0	1	2	3	4	5
Retrospektiivit	0	1	2	3	4	5
Suunnittelupeli	0	1	2	3	4	5
Pienet julkaisut	0	1	2	3	4	5
Vertauskuvat	0	1	2	3	4	5
Yksinkertainen suunnittelu	0	1	2	3	4	5
Testausvetoinen kehitys	0	1	2	3	4	5
Refaktorointi	0	1	2	3	4	5
Pariohjelmointi	0	1	2	3	4	5
Yhteinen omistajuus	0	1	2	3	4	5
Jatkuva integraatio	0	1	2	3	4	5
40-tuntinen työviikko	0	1	2	3	4	5
	0	1	2	3	4	5
	0	1	2	3	4	5
	0	1	2	3	4	5

LIITE 6 OHJELMISTON YLLÄPIDON ONGELMAT-LIITE

Ongelma	**Esiintymistiheys** **(Asteikolla 1-5, 1 hyvin harvoin – 5 hyvin usein)**	**Vakavuus** **(Asteikolla 1-5, 1 ei kovin vakava – 5 hyvin vakava)**
Puutteellinen tietämys ylläpidettävästä ohjelmistosta	1 2 3 4 5	1 2 3 4 5
Ajalliset ongelmat testauksessa	1 2 3 4 5	1 2 3 4 5
Riittämättömät testaus käytänteet	1 2 3 4 5	1 2 3 4 5
Ylläpitotoiminnan suuntaaminen organisaation tavoitteiden mukaisesti	1 2 3 4 5	1 2 3 4 5
Ylläpitäjien kokemattomuus	1 2 3 4 5	1 2 3 4 5
Alhainen moraali (Esim. kunnioituksen puutteesta)	1 2 3 4 5	1 2 3 4 5
Ammattilaisten menetykset	1 2 3 4 5	1 2 3 4 5
Riittämätön hallinnollinen tuki	1 2 3 4 5	1 2 3 4 5
Hallinnolliset ongelmat	1 2 3 4 5	1 2 3 4 5
Yhteisten standardien, käytänteiden ja työkalujen puute	1 2 3 4 5	1 2 3 4 5
	1 2 3 4 5	1 2 3 4 5
	1 2 3 4 5	1 2 3 4 5
	1 2 3 4 5	1 2 3 4 5

LIITE 7 PELIKEHITYKSEN ONGELMAT-LIITE

Ongelma	**Esiintymistiheys** **(Asteikolla 1-5, 1 hyvin harvoin – 5 hyvin usein)**	**Vakavuus** **(Asteikolla 1-5, 1 ei kovin vakava – 5 hyvin vakava)**
Ongelmat mittakaavan määrittelyssä	1 2 3 4 5	1 2 3 4 5
Epärealistinen mittakaava	1 2 3 4 5	1 2 3 4 5
Ominaisuuksien liiallinen lisääminen	1 2 3 4 5	1 2 3 4 5
Ominaisuuksien karsimiseen käytetty ylimääräinen aika	1 2 3 4 5	1 2 3 4 5
Ongelmat suunnitteluvaiheessa	1 2 3 4 5	1 2 3 4 5
Viivästykset	1 2 3 4 5	1 2 3 4 5
Tekniset ongelmat, esim kolmannen osapuolen komponenttien ohjelmointirajapinnoissa	1 2 3 4 5	1 2 3 4 5
Loppurutistukset (Engl. Crunch time)	1 2 3 4 5	1 2 3 4 5
Dokumentaation puute	1 2 3 4 5	1 2 3 4 5
Testausongelmat	1 2 3 4 5	1 2 3 4 5
Ohjelmistovirheet	1 2 3 4 5	1 2 3 4 5
Ryhmäkoostumukselliset ongelmat	1 2 3 4 5	1 2 3 4 5
Kehittäjien menettäminen	1 2 3 4 5	1 2 3 4 5
Budjetin ylitys	1 2 3 4 5	1 2 3 4 5
	1 2 3 4 5	1 2 3 4 5
	1 2 3 4 5	1 2 3 4 5
	1 2 3 4 5	1 2 3 4 5

LIITE 8 TERMISTÖSELVENNYKSET

Konseptivaihe tapahtuu ennen esituotantovaihetta. Konseptin kehittely lähes tulkoon täysin iteratiivinen prosessi, ideoita kehitellään ja ne mahdollisesti toteutetaan prototyyppeinä ja niitä myös hylätään säännöllisin aikavälein. Tämä vaihe on yleisesti ottaen ajallisesti rajoitettu ja tarkoituksena on tuottaa yksi tai useampi konsepti kehitys-suunnitelmasta julkaisijan tai lisenssin haltijan hyväksyttäväksi. Ketterästä näkökulmasta voidaan ajatella, että konseptivaiheessa luodaan tietämystä kehitystiimille sekä sidosryhmille, eikä niinkään arvoa kuluttajille (Keith, 2010).

Esituotantovaiheessa hiotaan pelimekaniikkoja ja pyritään niin sanotusti löytämään pelin hauskuus ja tuotetaan vaiheen tarpeiden mukaan mahdollisia kenttiä ja muita käytettäviä artefakteja. Tämä vaihe on täysin iteratiivinen ja inkrementaalinen. Kehitystiimi etsii iteratiivisesti pelin hauskuutta ja sopeuttaa kehityssuunnitelmaa inkrementaalisesti uuden tiedon puitteissa (Keith, 2010).

Tuotantovaiheessa tuotetaan pelin pelattava sisältö. Peli tuotetaan esituotantovaiheessa hiottujen ydinmekaniikkojen ja prosessien puitteissa. Tässä vaiheessa keskitytään tehokkuuteen ja inkrementaalisiin parannuksiin.

Jälkituotantovaiheessa pelin sisältö viimeistellään ja hiotaan julkaisu kuntoon. Tässä vaiheessa tuotettavaa peliä parannellaan ja hiotaan inkrementaalisesti. Kun peli on vihdoin saatu tyydyttävälle tasolle, luovutetaan se laitteistotestaukseen.

Varhaislapsuus (engl. infancy) on vaihe, joka alkaa yleisesti ottaen julkaisun jälkeen. Varhaislapsuuden aikana ohjelmiston käyttäjäkunta on varsin pientä ja he yleisesti ottaen tekevät ilmoituksia pienistä puuutteista ja näin ollen pääasiallinen ylläpito toimi onkin korjauksien tuottaminen (Kitchenham ym., 1999).

Murrosiässä (engl. Adolescence) ohjelmiston käyttäjäkunta on kasvussa ja pääasiallisena ylläpitotoimena on vieläkin korjauksien tuottaminen. On kuitenkin mahdollista, että käyttäjäkunnan kasvusta johtuen ohjelmiston vaatimukset muuttuvat, joten ohjelmistoon voidaan myös joutua tekemään muutoksia (Kitchenham ym., 1999).

Aikuisuuteen (engl. Adulthood) saavuttaessa ohjelmiston käyttäjäkunta on saavuttanut huippunsa ja ohjelmisto itsessään on suhteellisen vapaa mahdollisista puutteista. Jos kuitenkin suurin osa käyttäjäkunnasta niin toivoo, voidaan ohjelmistoon lisätä uusia toiminnallisuuksia. Lisäksi muutosten määrän kasvaessa ohjelmistoa voidaan myös uudistaa ja parannella sen

koodirakennetta. Pääasiallisina ylläpito toimenpiteinä ovat siis uusien vaatimusten tunnistaminen ja niiden tuottaminen (Kitchenham ym., 1999).

Ohjelmiston ollessa *seniilivs* (engl. Senility) vaiheessa, on markkinoilla jo uudempia vastaavia tuotteita ja ohjelmiston käyttäjäkunta on vähenee ja mahdollisesti pysyy hyvin pienenä. Pääasiassa tämän vaiheen aikana ohjelmistolle korjaavia toimenpiteita (Kitchenham ym., 1999).

Analyysi-vaiheessa (Engl. Analysis) on tarkoituksena analysoida käyttäjätarinamuotoiset käyttäjiltä saadut virheraportit sekä muutospyynnöt.

Suunnittelu-vaiheessa luodaan julkaisusuunnitelma (engl. Release plan), iteraatiosuunnitelma (engl. Iteration plan) sekä suunnitellaan tarvitut kokoontumiset.

Rakenteen muutos-vaiheessa (engl. Change design) on tarkoituksena muuntaa ylläpidettävää järjestelmää toteutukseen valittujen rakenteellisten muutosten mukaisesti. Tässä vaiheessa muun muassa tunnistetaan ne moduulit joihin muutokset vaikuttavat, muokataan dokumentaatiota vastaamaan muutoksia, luodaan testit uuden rakenteen testaamiseksi ja tunnistetaan tarvitut taantumistestaukset.

Muutosten toteutus-vaiheessa toteutetaan käyttäjätarinoiden mukaiset muutokset. Muutosten toteutus-vaihe sisältää muun muassa koodausta, yksikkötestaamista, muutetun koodin integrointia sekä integraatio ja taantumustestausta.

Järjestelmä testaus-vaiheessa testataan koko muutosten alla ollut järjestelmä jotta saataisiin varmuus siitä, että muutokset ja virheenkorjaukset on toteutettu onnistuneesti.

Vastaanottotarkastuksessa luodaan testit testaamaan, että järjestelmä palauttaa käyttäjätarinoiden mukaiset tulokset. Yleisesti toimitaan sitten, että asiakkaalta saadaan käyttäjätarina sekä sitä vastaava tuotos järjestelmältä ja näiden tietojen pohjalta ylläpito luo tarvitut testit toimivuuden varmistamiseksi.

Toimitus-vaihe on se vaihe missä tehdyt muutokset ja korjaukset julkaistaan. Tämä vaihe voi myös sisältää ohjelmiston asennukset, käyttäjien kouluttamisen uusien ominaisuuksien tai muutosten osalta ja ohjemiston varmuuskopion arkistoinnin.