Pelinkehitysprosessin ongelmia ja ratkaisuja

Oulun yliopisto
Tietojenkäsittelytieteen laitos
Kandidaatin tutkielma
Aarni Alatalo
14.4.2021

Tiivistelmä

Videopelien kehittäminen on muuttunut yksinkertaisista, muutamien ihmisten tekemistä peleistä kymmenien tai satojen ihmisten kehittämiin, monimutkaisiin ja sisällöltään laajoihin peleihin. Teknologian kehittymisen myötä aiemmat, tiukat rajoitukset laitteistolle ovat vähentyneet ja työkalut pelien kehittämiseen ovat parantuneet. Silti pelien tekeminen ei ole helpottunut.

Kehitysprosessi on yksi ratkaisevista tekijöistä kaupallisesti menestyvän ja laadullisesti hyvän pelin tuottamiseen. Tämän tutkimuksen tarkoituksena on selvittää, millaisia pelinkehitysprosessit ovat, millaisia ongelmia niissä esiintyy sekä miten ongelmia on yritetty ratkaista. Tutkimus suoritettiin kirjallisuuskatsauksena. Tutkimuksen motivaationa on etsiä tuoretta tietoa pelinkehitykseen liittyvistä vaikeuksista, koska peliala muuttuu nopeasti ja täten tutkimustieto vanhenee nopeasti. Lisäksi pelinkehityksestä on tehty yllättävän vähän tutkimusta, vaikka se on ohittanut esimerkiksi elokuva- ja musiikkiteollisuuden vuosittaisella liikevaihdolla.

Tutkimuksen tuloksena löydettiin, että pelialalla ei ole standardisoitua prosessimallia. Osa pelistudioista käyttää yhä vesiputousmallia, mutta iteratiivisten ja ketterien kehitysmenetelmien käyttäminen on lisääntynyt. Peliala kärsii samoista ongelmista kuin ohjelmistokehitys, mutta myös uniikkeja ongelmia esiintyy. Kenties merkittävimmät ongelmat syntyvät pelinkehityksen monialaisuudesta: pelistudioiden tiimit koostuvat sekä teknisten että puhtaasti luovien alojen henkilöistä. Eri alat käyttävät erilaisia metodologioita ja hallintakeinoja, ja niiden yhdistäminen keskenään on haasteellista. Ratkaisuina tutkijat ehdottavat tyypillisesti ohjelmistokehityksen puolelta muokattuja ketteriä kehityskeinoja, kuten Scrumia. Näiden käyttäminen käytännössä on jäänyt vähäiselle tutkimukselle, joten niiden tehokkuudesta ei voida sanoa varmuudella mitään. Tämän lisäksi teknologisten ongelmien väheneminen on nostanut luovat osuudet pelinkehityksessä tärkeämpään asemaan; pelien luovista prosesseista on kuitenkin tehty erittäin vähän kokonaisvaltaista tutkimusta.

Peliala vaikuttaa päälle päin ohjelmistokehitykseltä, mutta uniikit aspektit, kuten omaperäisyys, innovaatio ja luovuus tekevät pelien kehittämisestä hyvin erilaista ohjelmistokehitykseen verrattuna. Pelien ja ohjelmistokehityksen tavoitteet ovat filosofisesti myös hyvin erilaiset, sillä pelien päämääränä on tarjota käyttäjälle viihdyttävyyttä. Johtopäätöksenä onkin, että peliala tulisi nähdä omana alanaan, ei osana ohjelmistokehitystä.

Avainsanat
pelinkehitys, kehitysprosessi, prosessimallit

Ohjaaja
Lehtori Antti Juustila

Sisällysluettelo

Tiivistelmä 2
Sisällysluettelo 3
1. Johdanto 4
2. Tutkimusmenetelmät 5
3. Keskeiset käsitteet 6
4. Pelinkehitysprosessi 7
5. Pelinkehitysprosessin ongelmia 9
6. Pelinkehitysprosessin ongelmien ratkaisuja 13
7. Pohdinta 19
8. Yhteenveto 21
Lähteet 22

1. Johdanto

Pelien kehittäminen on vaikeaa. Peliala on muuttunut viime vuosikymmenten aikana muutamien henkilöiden tekemistä, yksinkertaisista peleistä kymmenien tai satojen ihmisten kehittämiin projekteihin, joiden kehittäminen kestää kuukausista vuosiin. Samalla haasteet ovat muuttuneet ohjelmoinnin monimutkaisuudesta ja laitteistorajoituksista muille osa-alueille. Pelien lisääntynyt monimutkaisuus, pelaajien korkeat odotukset ja alan kova kilpailu ovat nykyisen pelinkehityksen suurimpia haasteita.

Videopelien kehittäminen on yleensä ajateltu ohjelmistokehityksenä, jossa on vain joitakin uniikkeja piirteitä. Viime vuosina on kuitenkin huomattu, että nämä uniikit piirteet vaikuttavat kehitykseen niin paljon, että ohjelmistokehityksessä käytettävät metodologiat on sellaisenaan todettu epäsopiviksi pelien kehittämiseen. Yksi syy tähän on pelien filosofinen ero ohjelmistoihin: pelien tärkein tavoite on viihdyttävyys, kun taas ohjelmistojen on tarkoitus tarjota käyttäjälle keinoja jonkin tehtävän suorittamiseen. Viihdyttävyys, tarinankerronta ja estetiikka ovat pelinkehityksen avainvaatimuksia, joita ei ohjelmistokehityksessä huomioida lainkaan (Lehtonen, Lu, Nummenmaa & Peltonen, 2019).

Teknologisesta näkökulmasta pelien kehittäminen on nykyään helpottunut. Kehittyneitä pelimoottoreita ja muita työkaluja tarjotaan pelinkehittäjille jopa ilmaiseksi ja ne tukevat erilaisia kohdealustoja laidasta laitaan. Pelejä voidaan kehittää vaikkapa puhelimille, konsoleille ja tietokoneille. Usein samaa peliä tarjotaan monelle alustalle asiakasmäärän ja myynnin lisäämiseksi (Politowski, Petrillo, Montandon, Valente & Guéhéneuc, 2021a). Tämän lisäksi digitaaliset kaupat ovat nykyään lähes standardi tapa myydä ja ostaa pelejä – tätä kautta pelejä on tarjolla huikeita määriä. Samalla kuitenkin suuri määrä pelejä jää julkaisematta, tai pelit epäonnistuvat joko laadullisesti tai myynniltään. Mikäli teknisesti pelien kehittäminen ja jakelu on helpompaa, mikä johtaa peliprojektien epäonnistumisiin?

Yksi tärkeimmistä syistä onnistuneen peliprojektin valmistumiseen liittyy kehitysprosessiin. Kehitysprosessi määrittää käytännöt ja menetelmät, joilla peli valmistetaan ideasta myytäväksi tuotteeksi tai palveluksi. Toimivan kehitysprosessin avulla pelin laatua voidaan parantaa, kehitysaikaa lyhentää ja ennen kaikkea hioa pelien tärkeintä elementtiä – viihdyttävyyttä. Tutkimuksissa on kuitenkin havaittu, että pelinkehittäjät noudattavat harvoin virallisia prosessimalleja: esimerkiksi Kasurinen, Palacin-Silva ja Vanhala (2017) kirjoittavat, että 61% kyselytutkimuksen vastaajista ei noudattanut mitään virallista kehitystapaa; Musil, Schweda, Winkler ja Biffl (2010) kertovat yleisimpien ongelmien johtuvan pelinkehitysprosessien rakenteettomuudesta.

Tämän tutkimuksen tarkoituksena on selvittää pelinkehityksessä käytettäviä prosessimalleja, prosessissa esiintyviä ongelmia sekä ongelmien ratkaisuja. Tavoitteena on saada yleiskuva pelien kehittämisestä ja saada vastaus siihen, millä tavoilla pelinkehityksessä esiintyviä ongelmia on aikaisemmassa tutkimuksessa yritetty ratkaista. Tutkimuksen motivaationa on tarjota pelinkehittäjille tuoretta tietoa, sillä peliala on nopeasti muuttuva ja tutkimustieto mahdollisesti vanhenee nopeasti. Saatuja tietoja voidaan käyttää pohjana pelinkehitysprosessin parhaiden käytäntöjen muodostamiseen.

2. Tutkimusmenetelmät

Tutkimuksen tavoitteena on selvittää pelinkehitysprosessin käytäntöjä, ongelmia ja ratkaisuja. Tutkimuskysymyksenä on ”miten pelinkehitysprosessia on yritetty parantaa?” Avustavana kysymyksinä käytetään ”millainen on pelinkehitysprosessi?”, ”mitä ongelmia pelinkehitysprosessissa on?” ja ”miten pelinkehitysprosessin ongelmia on ratkaistu?”

Tutkimus suoritetaan muodostamalla ensin tutkimusongelmaan liittyviä hakusanoja ja niiden yhdistelmiä. Hakusanoja ovat muun muassa ”video game development process”. Hakusanojen muodostamisen jälkeen hakusanoilla etsitään tieteellisistä viitetietokannoista Scopus ja Google Scholar aiheeseen liittyviä aiempia tutkimuksia. Löytyneiden hakutulosten määrä pyritään hakusanojen avulla rajaamaan noin sataan tulokseen. Tämän lisäksi vain suomen- ja englanninkielisiä artikkeleita käytetään. Myös kirjallisuutta hyödynnetään, ja aiheeseen liittyvää kirjallisuutta etsitään yleisiä hakukoneita ja Oula-Finna-kirjastohakua hyödyntäen. Tuloksista valitaan sopivia tieteellisiä artikkeleita, konferenssipapereita ja kirjoja. Valittujen tulosten luotettavuutta arvioidaan Julkaisufoorumi-verkkosivun avulla etsimällä valitun teoksen ISSN-numeroa tai konferenssin nimeä. Tason 1-3 saaneet julkaisut hyväksytään käytettäväksi tässä tutkimuksessa. Jos julkaisulle ei löydy Jufo-tasoa, sitä käytetään silti riippuen sisällön hyödyllisyydestä. Edellä mainitun suodatuksen jälkeen julkaisu luetaan ja siitä löytyvät olennaisimmat tiedot kirjataan ylös. Kirjatut tiedot analysoidaan, yhdistetään ja tämän jälkeen raportoidaan Aiempi tutkimus -osiossa. Tämän jälkeen muodostetaan omia johtopäätöksiä aiheeseen liittyen ja vastataan tutkimuskysymyksiin sekä avustaviin kysymyksiin ja ratkaistaan tutkimusongelma.

Tutkimus on rajattu käsittelemään pelien menestymisen saavuttamista erityisesti tuotteen laadukkuuden kannalta. Täten muita menestymiseen liittyviä tekijöitä, kuten markkinointia, ei käsitellä.

3. Keskeiset käsitteet

Pelinkehityksessä käytetään joitakin ohjelmistokehityksestä poikkeavia termejä. Selvyyden vuoksi nämä termit esitellään tässä kappaleessa.

Pelinkehitysprosessilla tarkoitetaan pelattavan pelin luomiseksi tarvittavia vaiheita, tekniikoita ja käytäntöjä. Pelinkehitysprosessi noudattaa yleisesti ottaen ohjelmistokehityksen vaiheita: konsepti, esituotanto, tuotanto ja jälkituotanto.

Jälkiselvittely (engl. post mortem) on pelinkehitysprosessin jälkeen projektin johtajan tai senioritason kehittäjien kirjoittama vapaamuotoinen tekstidokumentti, jossa kehitysprosessin hyviä ja huonoja puolia kuvataan. Jälkiselvittelydokumentteja voidaan käyttää virheistä oppimiseen ja prosessissa esiintyneiden ongelmien parantamiseen tulevaisuuden projekteissa.

Pelimoottori on ohjelmisto, jonka tarkoituksena on tarjota yleisiin pelinkehityksen aikana tapahtuviin, toistuviin ongelmiin valmiita ratkaisuja. Pelimoottorit tarjoavat esimerkiksi grafiikan piirtämisen, äänten soittamisen ja syötteen lukemisen.

4. Pelinkehitysprosessi

Pelejä voidaan kehittää monilla erilaisilla tavoilla. Kehitysprosessit pohjautuvat usein ohjelmistokehityksen käytäntöihin. Koutonen ja Leppänen (2013) tosin mainitsevat, että pelien kehitykseen ei ole olemassa standardia prosessimallia. Politowskin, Fontouran, Petrillon ja Guéhéneucin (2018) suorittamassa verkkotutkimuksessa yksi haastateltu meni jopa niin pitkälle, että sanoi jokaisella pelillä olevan oma kehitysprosessinsa ja kehitysprosessin tekemisen olevan osa suunnittelua.

Pelinkehitysprosessi on tyypillisesti jaettu erilaisiin vaiheisiin. Rabin (2009) mainitsee viisi erilaista vaihetta: konsepti, esituotanto, tuotanto, jälkituotanto ja jälkimarkkinat. Pelinkehittäjien ja julkaisijoiden väliset sopimukset perustuvat tyypillisesti virstanpylväisiin (Koutonen & Leppänen, 2013; Rabin, 2009).

Kehitysprosessi on jaettu erilaisiin vaiheisiin, jotka sisältävät erilaisia aktiviteetteja. Prosessimalleja on puolestaan monia erilaisia. Koutonen ja Leppänen (2013) tutkivat haastattelujen avulla, kuinka paljon suomalaiset peliyhtiöt käyttävät ketteriä kehitysmenetelmiä ja miten ne vaikuttavat pelien kehitykseen. Tutkimukseen osallistui 20 peliyhtiötä. Heidän kyselyssään yli 50% vastanneista yhtiöistä ilmoitti käyttävänsä Scrum-kehitysmenetelmää tuotannossa, esituotannossa ja jälkituotannossa. Yli 30% yhtiöistä ilmoitti käyttävänsä Scrum-kehitysmenetelmää myös konseptin määrittelyssä.

Osborne O'Hagan, Coleman ja O'Connor (2014) tutkivat systemaattisella kirjallisuuskatsauksella pelinkehityksessä käytettäviä uusimpia tekniikoita. He totesivat pelinkehityksessä useimmin käytettyjen kehitysprosessien olevan ketteriä kehitysmenetelmiä tai hybridiprosesseja. He tutkivat myös syitä erilaisten kehitysprosessien käyttöönottoon. Ketterät kehitysmenetelmät kuten Scrum, XP (extreme Programming) ja Kanban ovat sopivia, kun nopea markkinoille saaminen ja innovaatio ovat tärkeitä. Spiraalimalli sopii suuriin projekteihin. Hybridimallit ovat hyviä, kun investointi ajan ja hinnan puolesta ovat vakaiden vaatimusten ja pidemmän eliniän myötä taattuja. He mainitsevat myös esimerkkinä mallipohjaisen kehitystavan (engl. model-driven development), mutta tutkimus osoitti, että tätä kehitystapaa ei ole käytännössä käytetty peliteollisuudessa.

Politowski, Fontoura, Petrillo ja Guéhéneuc (2016) puolestaan selvittivät jälkiselvittelyiden (engl. post mortem) analysoimisen ja kartoittamisen avulla pelinkehityksessä käytettäviä prosesseja. He löysivät neljä erilaista kategoriaa prosessimalleista: vesiputous, iteratiivinen, hybridi ja adhoc-prosessi. Yleisimmin käytetty prosessimalli oli iteratiivinen kehitys ja toiseksi eniten käytetty vesiputousmalli. Tutkimuksen päätelmänä Politowski et al. (2016) totesivat, että pelinkehitys ja ohjelmistokehitys jakavat samanlaisia prosesseja ja käytäntöjä. Iteratiiviset prosessit ovat yleistyneet ja ketterien kehityskeinojen tekniikoita on otettu enemmän käyttöön, mutta samalla näiden tuomia hyötyjä ei täysin ymmärretä. Myös myöhemmässä jälkiselvittelyjä tutkivassa paperissa Politowski, Petrillo, Ullmann ja Guéhéneuc (2021b) totesivat, että vesiputousmalli on yhä laajassa käytössä, mutta ketterien kehityskeinojen käyttäminen alkaa vähitellen yleistyä.

Kasurinen, Palacin-Silva ja Vanhala (2017) tutkivat kvantitatiivisen kyselytutkimuksen avulla pelien kehitysprosesseja ja bisnesaspekteja. He esittelevät pelien kehittämisen samankaltaiseksi ohjelmistokehityksen kanssa siten, että molemmat vaativat suunnittelua, kehitystä ja laadunvarmistusta. Merkittävämpänä erona ohjelmistokehitykseen on prosessien rakenteettomuus. Kuitenkin ketterät

kehitysmenetelmät tuntuvat olevan kasvava trendi pelinkehityksessä. Murphy-Hill, Zimmermann ja Nagappan (2014) mainitsevat mielenkiintoisena yksityiskohtana, että osa pelinkehittäjistä saattaa ilmaista ketterien kehitysmenetelmien käyttämisen kiertoilmaisuna selkeän prosessin puutteelle.

Engström, Berg Marklund, Backlund ja Toftedahl (2018) tutkivat systemaattisella kirjallisuuskatsauksella videopelien kehitysprosessia näkökulmasta, jossa se nähdään sekä ohjelmistona että luovana tuotteena. Heidän mukaansa pelinkehittäjät voivat lainata vaikutteita ohjelmistokehityksestä ja luovilta aloilta, mutta niiden yhdistäminen projekteissa on monimutkaista. He toteavat tutkimuksessaan, että ketterät kehityskeinot alkavat näkyä eräänlaisena standardina pelienkehityksessä, mutta toisaalta niiden heikkoudet alkavat näkyä erityisesti luovan prosessin kannalta.

Lehtonen et al. (2019) suorittivat systemaattisen kirjallisuuskatsauksen ja analysoivat pelien jälkikirjoituksia. Heidän mukaansa perinteistä ohjelmistokehitystä on tutkittu yli 40 vuotta, mutta pelinkehitystä on tutkittu hyvin vähän. On yleisesti hyväksyttyä, että ohjelmistokehitys ja pelinkehitys jakavat yhteisiä piirteitä, mutta niissä on toisaalta täysin uniikkeja piirteitä. Heidän tuloksissaan todetaan, että pelinkehittäjät eivät käytä virallisia, teoriapohjaisia metodeja ja prosesseja. Mikäli jokin prosessimalli on käytössä, se on muokattu versio Scrumista tai XP:stä.

Yhteenvetona pelinkehitysprosesseista voidaan sanoa, että alalla ei ole standardisoitua prosessia, mutta tyypillisesti kokonaisprosessi on jaettu erilaisiin vaiheisiin. Iteratiiviset prosessimallit ja ketterät kehitysmenetelmät ovat jo yleisessä käytössä ja niiden käyttäminen lisääntyy, mutta koska pelinkehityksessä on joitakin merkittäviä, uniikkeja piirteitä ohjelmistokehitykseen verrattuna, ohjelmistokehityksen prosessimalleja ei voida suoraan hyödyntää pelinkehityksessä.

5. Pelinkehitysprosessin ongelmia

Pelien kehittämiseen sovelletaan samoja menetelmiä kuin ohjelmistokehityksessä, jonka vuoksi niissä esiintyy myös samoja ongelmia. Toisaalta pelinkehityksessä esiintyy myös ohjelmistokehityksestä eroavia ongelmia. Tässä kappaleessa esitellään pelinkehityksen eroja ohjelmistokehitykseen ja erityisesti pelinkehitykseen liittyviä yleisiä ongelmia.

Petrillo, Pimenta, Trindade ja Dietrich (2009) tutkivat kirjallisuuskatsauksen avulla pelinkehitysalan ongelmia. Tutkimuksessa selvitettiin ohjelmistokehityksessä esiintyviä ongelmia sekä pelinkehitykseen liittyviä ongelmia erikoiskirjallisuudesta ja peliprojektien jälkiselvittelyistä. Keskeisimmät tulokset olivat, että ohjelmistokehitystä ja pelinkehitystä vaivaavat samankaltaiset ongelmat. Näitä ovat epärealistinen laajuus ja ongelmat vaatimusmäärittelyssä. Erityisesti epärealistinen projektin laajuus ja siihen liittyvä liiallinen optimismi olivat universaali ongelma peliprojekteissa. Mielenkiintoista tutkimuksen tuloksissa oli, että Petrillo et al. (2009) päättelivät sekä ohjelmistoprojektien että peliprojektien ongelmien johtuvan hallinnollisista ongelmista, eivät niinkään teknologisista ongelmista.

Tärkein ero peliprojektien ja ohjelmistoprojektien ongelmissa on kommunikaatio tiimien välillä (Petrillo et al., 2009). Ohjelmistoprojektien tiimit ovat yleensä homogeenisiä, mutta pelinkehityksessä vaaditaan ohjelmistokehittäjien lisäksi ihmisiä puhtaasti luovilta ja taiteellisilta aloilta. Asiaan liittyen Musil et al. (2010) suorittivat peliyhtiöille kyselytutkimuksen, jonka tavoitteena oli analysoida pelinkehitystä monialaisessa ympäristössä ja ehdottaa uutta prosessimallia pelinkehityksen tukemiseen ja parantamiseen. Monialaisuus aiheuttaa vaikeuksia työnkulun integroinnissa eri alojen välillä, koska pelinkehityksen eri aloilla on monimutkainen ja toisistaan eristäytynyt metodologia. He mainitsevat suurimmiksi haasteiksi yhteisen näkemyksen saamisen pelistä, datan vaihtamisen eri alojen välillä ja virheentunnistuksen eri alojen ja työkalujen välillä. Engström et al. (2018) toteavatkin, että pelit eivät ole pelkästään sovelluksia, vaan luovia tuotteita, kulttuurillisia ilmaisukeinoja tai taideteoksia, josta seuraa kehitystiimien ristiriitaisuus.

Pelien kehittämisessä luovuus on tärkeässä asemassa. Ensgström et al. (2018) mukaan luovuuden ja kontrollin välillä on ristiriita, joten luovuuden hallinta on erilaista kuin perinteinen hallinta. Ohjelmistokehityksen vaikutusta luovaan prosessiin ei yleensä huomioida. He mainitsevatkin, että iso osa aiemmista pelinkehitykseen liittyvästä tutkimuksesta on tehty joko sovelluskehityksen tai projektinhallinnan näkökulmasta, ja taiteellisesta tai humanistisesta näkökulmasta huomattavasti vähemmän. Johtopäätöksenä he sanovat, että samalla tavalla kuin ohjelmistokehityksen näkökulmasta tehdyt tutkimukset eivät huomioi taiteellista osuutta, myös taiteellisesta näkökulmasta tehdyt tutkimukset eivät huomioi ohjelmistokehityksen monimutkaisuutta.

Osborne O'Hagan et al. (2014) kirjoittavat pelinkehityksessä tyypilliseksi ongelmaksi aikataulupaineen, koska peli pitää saada mahdollisimman nopeasti markkinoille. Aikataulupaineen vuoksi peliprojektit kärsivät huonosti arvioiduista aikatauluista ja täten aikataulujen ylityksistä. Myös Washburn, Sathiyanarayanan, Nagappan, Zimmermann ja Bird (2016) havaitsivat peliprojektien erittäin tiukat aikataulut tutkimuksessaan. Vaikkakin aikatauluongelmat ovat myös ohjelmistoprojekteissa tyypillinen ongelma, pelinkehityksessä aikataulupaine on Murphy-Hill et al. (2014) mukaan vielä vahvempi. Syynä tälle on muun muassa se, että pelijulkaisijat eivät halua lykätä pelin julkaisua (Lehtonen et al., 2019).

Furtado, Santos, Ramalho ja De Almeida (2011) kertovat ohjelmistokehyksen, kuten pelimoottorin, käyttämisen vähentävän kehityksen kustannuksia. Ongelmana on kuitenkin kaikkien tuotevarianttien vaatimusten kartoittaminen ohjelmistokehykseen – tämän tekemiseen tarvitaan yleensä senioriohjelmistoarkkitehti. Politowski et al. (2021a) kirjoittavat pelinkehittäjien käyttävän erikoistuneita ohjelmistoinfrastruktuureja kehityksessä, joista merkittävimpiä ovat pelimoottorit. Myös heidän mukaansa pelien eri lajityypit vaativat erilaisen pelimoottorin.

Schmalz, Finn ja Taylor (2014) tutkivat haastattelujen avulla viihdeohjelmistojen projektinhallintaa, erityisesti riskienhallintaa. He esittelevät pelinkehityksessä olevan samoja riskejä kuin ohjelmistokehityksessä, mutta pelinkehitykselle ominaisia riskejä syntyy kehityksen luovasta osuudesta, suhteista ulkoisiin sidosryhmiin ja yrityksen ulkoisesta ympäristöstä (esimerkiksi julkaisijalta, lisensoijilta tai julkaisualustan omistajilta). Näiden lisäksi tuloksissa selvisi kolme uutta riskiä: yleisön sopivuus, viihdyttävyystekijä ja alkuperäisyyden laajuus. Muita havaintoja olivat 1) yhdelläkään haastattelun antaneella tuottajalla (projektipäälliköllä) ei ollut IT-alan koulutuspohjaa, ja 2) epävirallisten riskienhallintakeinojen käyttäminen peliprojekteissa. Epävirallisista keinoista yleisimpiä olivat projektin rakenteeseen ja kehityskäytäntöihin liittyvät keinot (prototypointi, esituotantovaihe, aikainen ongelmallisten projektien hylkääminen). Nämä keinot tarkoittavat yksinkertaisuudessaan sitä, että pelinkehittäjät testaavat projektin aikaisessa vaiheessa prototyyppejä ja hylkäävät mahdollisesti ongelmalliset projektit nopeasti, jolloin kehittämiseen ei tarvitse investoida aikaa ja rahaa. Puolet vastanneista mainitsi myös ketterien kehityskeinojen käyttämisen, mutta haittapuolena ketteristä kehityskeinoista huomattiin alkuperäisen tarkoituksen menettäminen jatkuvan muuntautumisen takia ja ”epäpuhtaiden” käytäntöjen käyttäminen. Lopuksi he mainitsivat alan kovan kilpailun sekä kuluttajien ”räjähdysherkät”, nopeasti muuttuvat näkemykset ja korkeat laatuvaatimukset.

Washburn et al. (2016) analysoivat kvantitatiivisesti ja kvalitatiivisesti pelien jälkikirjoituksia. Heidän tavoitteenaan oli saavuttaa empiirisesti johdettu luokittelu pelinkehityksen piirteille tai ulottuvuuksille, syntesoida parhaita käytäntöjä ja pahimpia haasteita sekä tarjota suosituksia tulevaisuuden pelinkehittäjille entisten kokemusten perusteella. He esittelevät ohjelmistoprojektien ja peliprojektien välillä olevan useita eroja. Pelinkehityksessä vaatimukset ovat subjektiivisia (viihdyttämisen vaatimus). Ylläpidettävyyttä uhrataan usein suorituskyvyn parantamiseksi. Testauksessa ja laadunvarmistuksessa käytetään erilaisia menetelmiä (testaamiseen käytetään ihmisiä ja vähäisesti automaattisia testejä). Osborne O’Hagan et al. (2014) mukaan pelien käytettävyystestaus on haastavaa pelien toimintatavan vuoksi: pelit tarjoavat tyypillisesti vaikeutuvia haasteita, joiden ratkaisemisesta pelaaja saa iloa. Käyttäjäkokemusten avulla saatu palaute myös ohjaa suunnitteluiteraatioita. Politowski et al. (2021b) tekivät havainnon, että testauksesta löytyi jälkiselvittelyistä yllättävän vähän tietoa. He esittävät hypoteesina, että pelitestaus on onnistunutta, jonka takia siitä ei löydy mainintoja – toisaalta on yleisesti tiedettyä, että peliprojektit kärsivät matalasta laadusta ja aikatauluylityksistä, joten todennäköisemmin pelien testaus on ongelmallista. Tutkimuksen perusteella pelien testausta tehdään testaussessioiden avulla, jotka eivät ole skaalautuvia. Myöskään yksikkötestauksista ja automatisoinnista ei ollut mitään mainintoja.

Mitre-Hernández, Lara-Alvarez, González-Salazar ja Martín (2016) tekivät tapaustutkimuksen, jonka tavoitteena oli luoda pelejä varten uudelleentyöstöä vähentävä prosessimalli. Uudelleentyöstö on mikä tahansa ongelmien löytämisestä ja korjaamisesta syntyvä vaiva, joka suoritetaan sen jälkeen, kun dokumentit ja koodi on virallisesti allekirjoitettu osana konfiguraationhallintaa. He kirjoittavat joidenkin pelien olevan

monimutkaisia järjestelmiä, joiden kehittäminen vaatii esituotannossa ja tuotannossa merkittävän määrän vaivaa. He sanovat uudelleentyöstöön liittyvien ongelmien alkavan jo esituotantovaiheen alussa – syyksi he sanovat pelisuunnittelun määrittelemättömät tai epäselvät vaatimukset. Myös Politowski et al. (2021b) havaitsivat yhdeksi haastavimmista ongelmista epäselvän pelisuunnitteluvision. Epäselvä pelisuunnitteluvisio tarkoittaa sitä, että kehittäjätiimi ei ymmärrä selkeästi pelin visiota ja tämä aiheuttaa sivuoireena haittoja muun muassa hallintaan ja testaukseen.

Kasurinen et al. (2017) mainitsevat pelinkehitystä suoraan tukevien metodologioiden puutteen johtavan metriikan ylläpitämisen puuttumiseen. Tästä syystä esimerkiksi koodin uudelleenkäyttöä ei ole. Murphy-Hill et al. (2014) kirjoittavat myös vähäisestä koodin uudelleenkäytöstä peleissä. Syyksi he sanovat merkittävän panostuksen suorituskykyyn ja projektikohtaisen optimoinnin, mutta toisaalta myös ajatuksen siitä, että uudelleenkäyttö kuvastaa samankaltaisuutta – ja samankaltaisuus voidaan nähdä innovaation vastakohtana. Toisaalta he sanovat, että peliprojekteissa tapahtuu uudelleenkäyttöä, mutta eri tavalla kuin ohjelmistoprojekteissa: esimerkiksi pelimoottoreiden ja sisäisesti kehitettyjen työkalujen uudelleenkäyttöä sekä peräkkäisissä pelijulkaisuissa (jatko-osissa) koodin uudelleenkäyttöä. Washburn et al. (2016) kirjoittavatkin, että useimmissa peleissä tarvitaan kehitystyökaluja, jotka tehdään alusta asti.

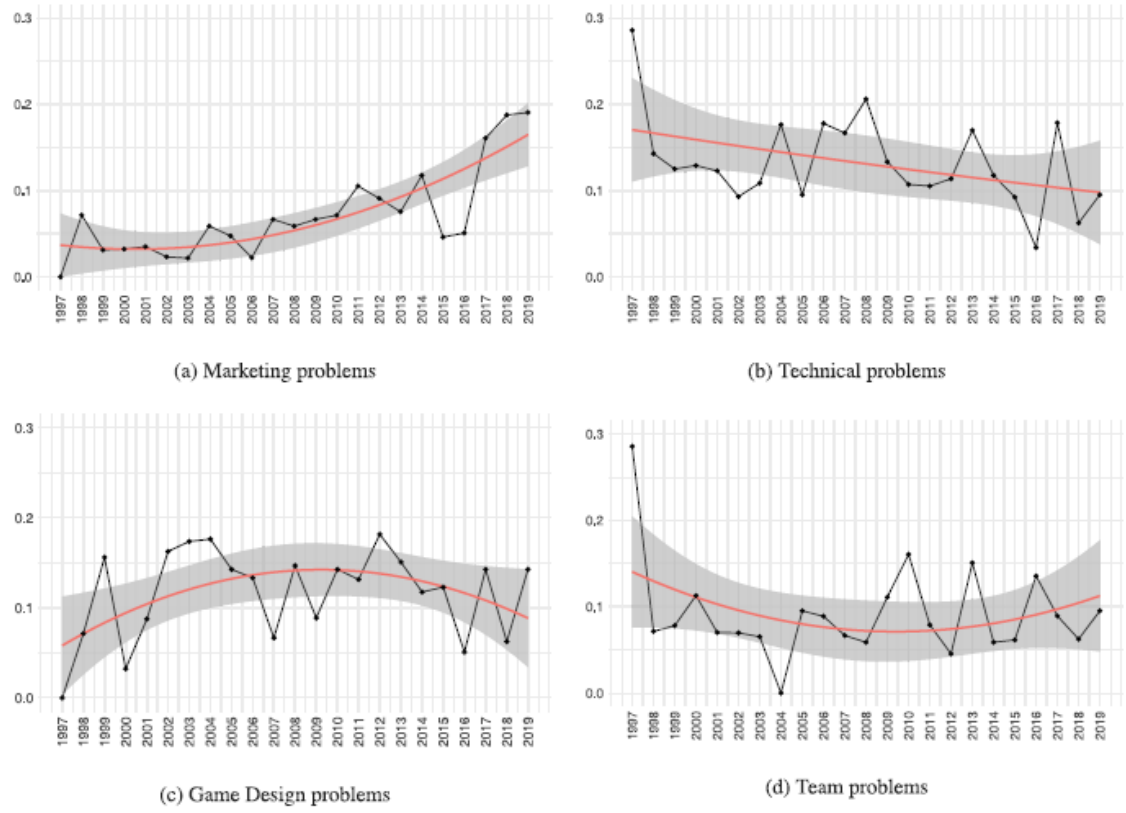
Lehtonen et al. (2019) kuvaavat pelinkehitystä hajanaiseksi alaksi, toisin kuin kurinalainen ja teoriapohjainen ohjelmistokehitys. He esittävät pelinkehityksen olevan ohjelmistokehitystä, jossa on kuitenkin uniikkeja piirteitä. Pelinkehityksen uniikkeja piirteitä ovat monialaisuus, filosofinen ero ohjelmistokehityksen sovelluksiin, massamarkkinointi viihdettä ja tunteellisia reaktioita aiheuttavana tuotteena, sekä ohjelmistokehityksessä käytettävien mallien ja teorioiden yhteensopimattomuus pelinkehitykseen. Suurimmat ongelmat pelinkehityksessä ovat pelisuunnitteludokumentin ja vaatimusmäärittelydokumentin yhteensopivuus, uniikit vaatimukset, muutosten haluttavuus ja virallisten, teoriapohjaisten metodien ja prosessien puuttuminen. Muutosten haluttavuudella viitataan siihen, että pelinkehittäjien pitää valmistautua tekemään kehityksen loppuvaiheessakin isoja muutoksia, kun viihdyttävyyttä etsitään pelin toimintojen, ominaisuuksien ja testauksen avulla. Tutkimuksen kannalta he mainitsevat ongelmaksi sen, että erityisesti isot pelejä kehittävät yritykset pitävät käyttämänsä metodit ja käytännöt salaisuutena. Osborne O'Hagan et al. (2014) huomasivat myös tutkimukseen liittyvän haasteen, sillä he kertovat suurimman osan varsinaisesta peliteollisuuden raporteista esiintyvän vain "harmaassa" kirjallisuudessa, kuten lehdissä ja verkkosivuilla.

Politowski et al. (2021b) tutkivat pelinkehityksessä esiintyviä ongelmia peliprojektien jälkiselvittelyistä vuosien 1997-2019 välillä. Ongelmat on ryhmitelty kolmeen ryhmään: tuotanto, hallinta ja bisnes. Näihin ryhmiin kuuluvat ongelmat on jaettu eri tyyppeihin, joista yleisimmät ovat pelisuunnittelu, tekniset ongelmat ja tiimiongelmat, jotka kattavat kokonaisuudessaan 30% ongelmista. Ongelmatyypit on jaettu vielä alatyyppeihin, jotka esittävät ongelman lähteen. Alatyyppeihin kuuluu jo tässä kappaleessa aiemmin esiteltyjä ongelmia, kuten aliarviointi (ohjelmistotuotanto-ongelma, esimerkiksi työmäärän aliarviointi), väärin muodostetut tiimit (kommunikaatio-ongelmat) ja pelisuunnittelun epäselvä visio (vaatimusmäärittely). Näiden lisäksi tekstin muita ongelmia ovat viihdyttävyyden puute, alusta- ja teknologiarajoitteet, pelisuunnittelun kompleksisuus sekä huonot tai puuttuvat työkalut.

Viihdyttävyyden puute tarkoittaa pelisuunnitteluun liittyvää ongelmaa, jossa pelille ei löydetä tuotannon aikana haluttua viihdearvoa. Alusta- ja teknologiarajoitteet ovat pelien

kohdealustojen (esimerkiksi pelikonsolit, mobiililaitteet tai tietokone) rajoitteisiin liittyvä ongelma – peli täytyy julkaista erilaisille alustoille asiakkaiden ja myynnin lisäämisen vuoksi, mutta jokaisella alustalla on omat laitteistorajoituksensa. Pelisuunnittelun kompleksisuudella tarkoitetaan sitä, että pelisuunnittelu on vaikeaa ja sille ei ole selkeää noudatettavaa prosessia. Huonot tai puuttuvat työkalut tarkoittavat esimerkiksi pelimoottoreiden huonoa käytettävyyttä sekä sitä, että pelimoottorit voivat nopeuttaa kehitystä, mutta toisaalta rajoittavat pelisuunnittelua.

Kuvassa 1 näytetään Politowski et al. (2021b) löytämistä ongelmista neljä kuvaajaa, jotka esittävät ongelmatyyppien muuttumista vuosien 1997-2019 välillä.



Kuva 1. Pelinkehitykseen liittyvien ongelmien muuttuminen vuosien 1997-2019 välillä (Politowski et al., 2021b, s. 5).

Kuvasta 1 nähdään, että markkinointiongelmat (a) ovat kasvaneet, tekniset ongelmat (b) vähentyneet, pelisuunnittelun ongelmat (c) vähentyneet ja tiimeihin liittyvät ongelmat (d) lievässä kasvussa. Punainen viiva on toisen asteen polynomifunktio ja harmaa alue luottamusväli. Vaikkakin punainen viiva osoittaa ongelmien trendien suunnan kokonaisuudessaan, on mielenkiintoista, että esimerkiksi pelisuunnitteluongelmien määrä vaihtelee lähes vuosittain eräänlaisena sahakuviona.

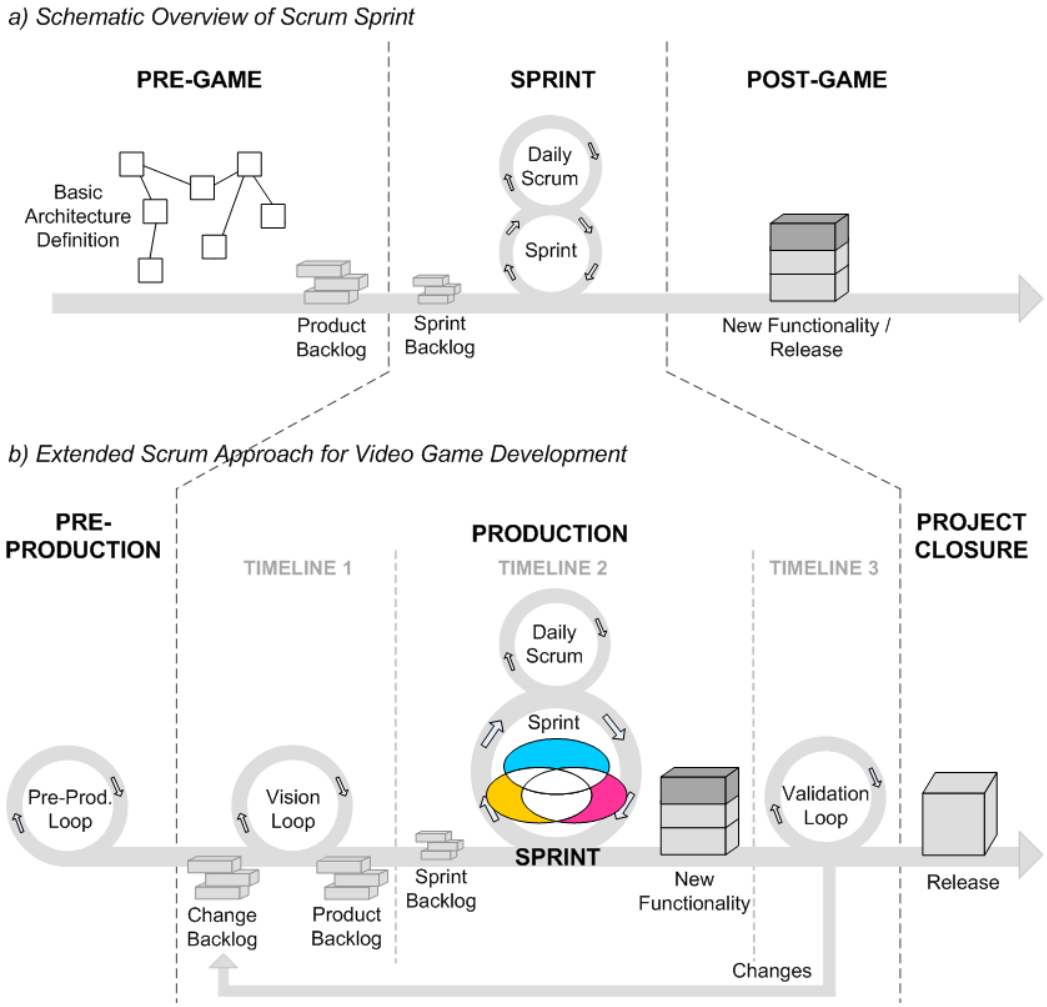
Yhteenvetona pelinkehityksen ongelmista voidaan sanoa, että ohjelmistokehitystä vaivaavat ongelmat ovat nähtävissä myös pelinkehityksessä. Toisaalta myös uniikkeja ongelmia esiintyy – suurimmat haasteet ovat aiempien tutkimusten perusteella tiimien monialaisuuteen liittyvät ongelmat (kommunikointi eri alojen välillä, erilaiset metodologiat eri aloilla, luovuuden hallinta), projektinhallintaan liittyvät ongelmat (projekti- ja riskienhallinnan epäviralliset keinot, tuottajien koulutuksen puute) sekä vaatimuksiin liittyvät ongelmat (viihdyttävyys on subjektiivista, pelisuunnittelun monimutkaisuus ja sen muuntaminen vaatimuksiksi, erilainen käyttäjätestaus).

6. Pelinkehitysprosessin ongelmien ratkaisuja

Petrillo et al. (2009) ehdottavat tutkittavaksi pelien vaatimusmäärittelyn parantamiseen tarkoitettuja tekniikoita. Myös Osborne O'Hagan et al. (2014) kirjoittavat, että perinteisen ohjelmistokehityksen vaatimusmäärittelytekniikoita pitää laajentaa pelinkehityksessä, jotta voidaan tukea pelien luovia prosesseja. Samoin Lehtonen et al. (2019) toteavat, että paremmalla vaatimusmäärittelyllä voitaisiin helpottaa tai eliminoida pelinkehityksen ongelmia. Lehtonen et al. (2019) kuitenkin kertovat tuloksissaan, että heidän tutkimassaan 340 jälkikirjoituksessa ei yhdessäkään mainittu vaatimusmäärittelyä terminä. He päättelivät tästä, että vaatimusmäärittely ei välttämättä ole huolenaihe pelinkehittäjille; tämä ei kenties kuitenkaan ole oikea johtopäätös, koska laajuus ja dokumentointi mainittiin hyvin monessa dokumentissa. He kirjoittavat lisäksi, että konkreettiset ratkaisut ovat hyvin harvassa: suurin osa tutkimuksessa esitetyistä ratkaisuista ovat ohjelmistopuolelta otettuja, hämäriä parhaita käytäntöjä.

Petrillo et al. (2009) ehdottavat, että pelinkehitykseen pitäisi määritellä ohjelmistojen kehitysmenetelmä, joka tukee pelialan kompleksisuutta, monialaisuutta ja luovuutta. Ketteriä kehitysmenetelmiä pidetään tyypillisesti hyvänä lähtökohtana. Esimerkiksi Koutonen ja Leppänen (2013) kertovat pelinkehittäjien uskovan ketterien kehityskeinojen tarjoavan useita etuja. Heidän suorittamassaan kyselyssä vastaajat uskoivat ketterien kehitysmenetelmien parantavan julkaisunopeutta, helpottavan muuttuvia vaatimuksia, vähentävän virheitä, parantavan tuotelaatua ja prosessin tuotettavuutta, nostavan asiakasarvoa sekä tarjoavan ylläpidettävän ja tasaisen työmäärän. Silti Scrum-menetelmässä on heidän mukaansa muutamia huonoja puolia: esimerkiksi Scrumin backlog aiheuttaa tuotantovaiheessa ongelmia. He ehdottavat Kanban- tai XP-menetelmien käyttämistä tuotantovaiheessa.

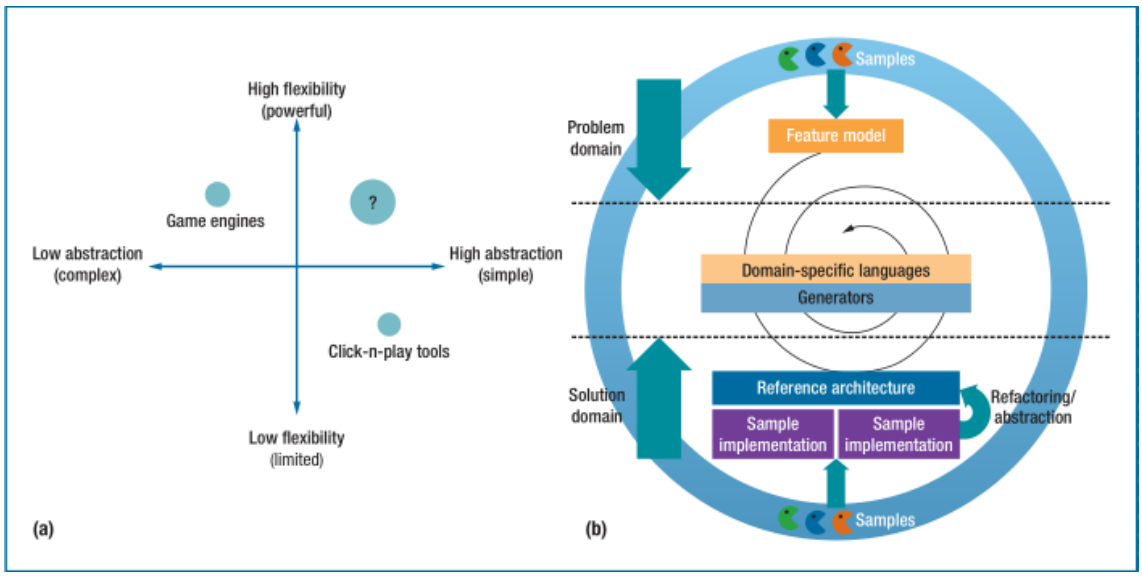
Musil et al. (2010) pitävät Scrum-kehitysmenetelmää hyvänä mallina, mutta siihen lisäyksenä täytyisi huomioida pelinkehityksen monialaisuus. He ehdottavat kehitysprosessia, joka on jaettu esituotanto-, tuotanto ja lopetusvaiheeseen. Kuvassa 2 esitetään visuaalisesti heidän ehdottamansa prosessimalli kohdassa b).



Kuva 2. Laajennetun Scrum-prosessin rakenne (Musil et al., 2010, s. 7).

Esituotantovaiheessa painotetaan keksimistä ja se on lyhyt kestoltaan. Tuotantovaihe on Scrum-menetelmään perustuva ja on jaettu kolmeen osaprosessiin: visiosilmukka, sprintti ja varmistussilmukka. Kaikilla osaprosesseilla on oma kestonsa ja tärkeimmät osapuolet eri aloilta ovat mukana kaikissa osaprosesseissa. Visiosilmukassa ylläpidetään projektin visiota ja tuotteen backlogia, johon luotavat tehtävät saadaan esituotannolta ja varmistussilmukalta. Sprintissä jokaiselle alalle tehdään oma sprint-työnkulku, jonka avulla monialaisuus saadaan integroitua. Luovilla aloilla voidaan käyttää Scrum-Ban -variaatiota. Koordinaatio alojen välillä tapahtuu päivittäisillä sprint-kokouksilla. Varmistussilmukan tehtävänä on huolehtia tuotteen arvioinnista, varmistuksesta, kokonaislaadusta ja virstanpylväiden noudattamisesta. Myös testaus tehdään varmistussilmukassa. Varmistussilmukan tulokset syötetään takaisin visiosilmukalle. Ehdotetulle prosessille tehtiin alustavia arvioita, mutta sen käyttöä käytännössä ei ole kokeiltu. Alustavien arvioiden mukaan prosessi vaikutti pelinkehittäjien mielestä hyödylliseltä. Hyödyiksi mainitaan muun muassa ylityön (engl. crunch time) vähentäminen, esituotannon tuoma tila ideoinnille ja prototypoinnille sekä prosessin aikana tapahtuva, jatkuva pelisuunnittelu.

Furtado et al. (2011) kehittivät pelejä varten tarkoitetun ohjelmistotuotantolinjan, jonka tarkoituksena on parantaa pelinkehitysprosessia. Kuvassa 3 näkyy pelinkehittäjien nykyinen tilanne ja ehdotettu tuotantolinja.



Kuva 3. Nykyinen pelinkehittäjien tilanne a) ja ehdotettu tuotantolinja b). (Furtado et al., 2011, s. 31).

Kuvan 3 a) osassa näytetään pelinkehittäjien ongelmallinen tilanne: pelimoottorit ovat kohtalaisen komplekseja, mutta samalla erittäin joustavia ja tehokkaita.”Click-n-play” -työkalut ovat yksinkertaisia käyttää, mutta rajoitettuja joustavuudeltaan. Furtado et al. (2011) tarjoavat tähän ongelmaan ratkaisua. Heidän ehdotuksensa näkyy kuvan 2 b) osassa.

Tuotantolinja esitellään tässä lyhyesti, koska kyseessä on hyvin kattava ja yksityiskohtainen prosessikuvaus. Kokonaisuudessaan ehdotus perustuu tiukkaan ”ylhäältä alas ja alhaalta ylös” domaintekniikan lähestymistapaan, jotka ovat yhdistetty spiraalimalliksi ja iteratiiviseksi reuna-keskus -prosessiksi. Tämä vähentää kirjoittajien mukaan kaikilla reunoilla tapahtuvaa kehitysresurssien investointia. Tuotantolinja on karkeasti jaettu neljään osa-alueeseen: domainin määrittely, pelidomainin analysointi, sovelluksen ydinsisällön luominen ja kehityssisällön luominen. Tuotantolinjan pääajatuksina on samaan kohdedomainiin kuuluvien pelien käyttäminen näytteinä, referenssiarkkitehtuurin käyttäminen, automatisoiminen ja täsmäkielten (engl. domain-specific language) luominen.

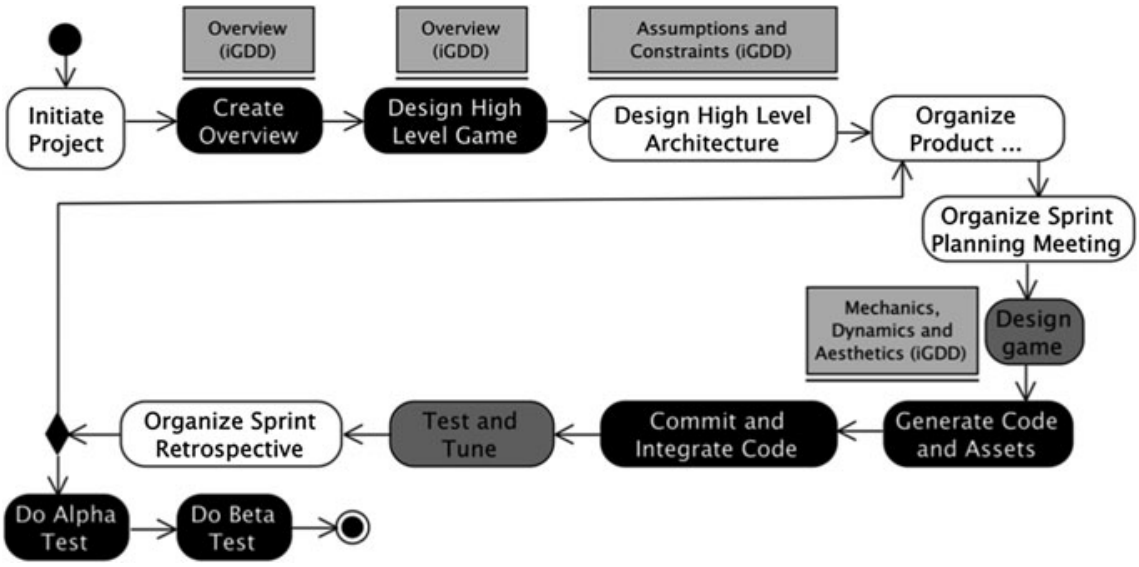
Kehittäjien luoma ArcadEx-ohjelmistokehys nopeutti kirjoittajien testien mukaan neljäs- tai viidesosan tuotantoa. Koska tuotantolinjan avulla on tarkoitus etsiä pelien lajityypeille tyypillisiä osia ja käyttää niitä uudelleen, syntyi huolenaihe, että tuotantolinja uhkaa tuotteen luovuutta ja erilaisuutta. Testien perusteella kuitenkin huomattiin, että automatisoimalla rutiiniosat ja virhealttiimmat alueet kehittäjät pystyivät käyttämään aikaa ja resursseja variaatioon ja uniikkiutta parantaviin laajennusosiin. Tutkijat mainitsevat, että pelinkehityksen uniikkiuden vuoksi suoria ohjelmistokehityksen tekniikoita ei tulisi käyttää, vaan luoda erikoistettuja, aluekohtaisia lähestymistapoja. Tuotantolinjan hyödyiksi mainittiin pelimoottorien vähempi monimutkaisuus, kehitystehtävien pilkkominen pienemmiksi ja automatisoiduiksi palasiksi, lisääntynyt arvo priorisoiduille pelin ala-alueelle, aluekohtaisen sisällön räätälöinti uniikeille (peli)tuoteperheille ja parantunut luotto siihen, että tulokset vastaavat alkuperäistä visiota ja vaatimuksia.

Wang ja Nordmark (2015) kirjoittavat ohjelmistoarkkitehtuurilla olevan tärkeä merkitys pelinkehityksessä. Ohjelmistoarkkitehtuuria käytetään peliohjelmiston kompleksisuuden hallintaan ja laadun saavuttamiseen suorituskyvyssä, saatavuudessa, turvallisuudessa ja

muokattavuudessa. Luovien prosessien hallintaan ja tukemiseen käytetään heidän mukaansa pelimoottoreita. Pelimoottorien avulla voidaan tukea dynaamista uusien elementtien lataamista, skriptausta sekä uusien ideoiden nopeaa prototypointia. Pelien kehitys on heidän mukaansa muuttunut viime vuosien aikana siten, että kolmansien osapuolten ohjelmistoja ja väliohjelmistoja käytetään enemmän, ja teknisesti pelien kehittäminen on muuttunut helpommaksi – muutoin kehittäminen itsessään ei ole helpottunut pelaajien suurten odotusten ja pelien lisääntyneen kompleksisuuden vuoksi. Politowski et al. (2021a) sanovat, että pelimoottorit ovat työkalu pelien kehittämisen helpottamiseen, mutta eivät ole laatuvaatimusten (kuten viihdyttävyyden) kanssa tekemisissä.

Washburn et al. (2016) tarjoavat pelinkehityksen parhaita käytäntöjä jälkiselvittelyjen perusteella. He tarjoavat ohjeita neljään eri kategoriaan: pelisuunnittelu, kehitysprosessi, tiimi ja taide. Pelisuunnittelussa kehittäjien pitäisi luoda hyvin määritelty konsepti ennen kehityksen aloittamista ja lisäksi keskittyä suunnittelussa asioihin, joilla pelaajan huomio saadaan kiinnitettyä. Kehitysprosessin parantamiseksi kehittäjien pitää investoida projektin alussa aikaa suunnittelulle ja mallintamiselle sekä rakentaa kehityksen aikana prototyyppejä, joita voidaan laajentaa iteratiivisten menetelmien avulla. Kehitystiimin pitäisi käyttää aikaa kouluttaakseen jäseniä toimimaan kehitysympäristössä ja huomioida palkkaamisessa työntekijän motivaatio. Taiteen osalta pelien korkeampi laatu voidaan saavuttaa hyödyntämällä ammattilaisartisteja, jotta taiteella voidaan kiinnittää pelaajien huomio.

Mitre-Hernández et al. (2016) ehdottavat kehitysprosessin parantamiseksi erilaista pelisuunnitteludokumenttia ja Scrum-menetelmään pohjautuvaa muunnosta. Päätavoitteena näiden käyttämisellä on uudelleentyöstön vähentäminen. Parannetun pelisuunnitteludokumentin nimi on iGDD (improved Game Design Document) ja kehitysmenetelmän nimi on Scrum sdPP (software development Project Pattern). Kuvassa 4 näytetään ehdotettu työnkulku.



Kuva 4. Scrum sdPP:n ja iGDD:n työnkulku. (Mitre-Hernández et al., 2016, s. 299).

iGDD-dokumentti on jaettu eri abstraktiotasoihin ja pohjautuu Chris Taylorin pelisuunnitteludokumenttiin (https://www.runawaystudios.com/articles/chris_taylor_gdd.php), mutta sisältää lisäksi MDA-rungon ja parhaat käytännöt vaatimusmäärittelyspesifikaatiosta. MDA-runko (mekaniikka, dynamiikka ja estetiikka, engl. mechanics, dynamics and aesthetics) on

iteratiivinen lähestymistapa videopelien suunnitteluun ja säätämiseen. Sen avulla määritellään ensin estetiikka, sitten estetiikkaa täydentävä dynamiikka ja lopuksi pelielementit, jotka tuovat tarvittavan interaktion. Scrum sdPP puolestaan sisältää useita aktiviteetteja. Aktiviteetit ovat yleiskatsauksen luominen, korkean tason pelin suunnittelu, korkean tason pelin arkkitehtuurin suunnittelu, pelin suunnittelu, koodin ja sisällön luominen, toteutuksen ja koodin integrointi, testaus ja säätäminen ja lopulta alpha- ja beta-testaus.

Kirjoittajien tekemän tapaustestin mukaan iGDD:n ja Scrum sdPP:n käyttäminen vähensi huomattavasti uudelleentyöstöä. iGDD:n käyttäminen selkeytti heidän mukaansa vaatimusten tekemistä yleiskatsauksen avulla. Lisäksi se määritteli selkeästi olettamukset ja rajoitukset, jonka avulla testikäyttäjät löysivät kontekstin ja rajoitukset korkean tason arkkitehtuurille. Myös selkeiden rajoitteiden ja rajojen määrittely parantaa vaatimusmäärittelyä. Määrittelemällä mekaniikan ja dynamiikan testikäyttäjät saivat hyvin organisoidut pelielementit ja hyvän suuntauksen kompleksisuudelle pelaajan profiilin avulla. Pelielementtien jäljitettävyys vähensi uudelleentyöstöä. Scrum sdPP puolestaan auttoi testikäyttäjiä jäljittämään iGDD:n eri osiot, joka auttoi löytämään olennaiset asiat milloin tahansa kehityksen aikana. iGDD:n ja Scrum sdPP:n käyttämisen todettiin olevan selkeämpi, yksityiskohtaisempi ja tarkempi kuin perinteinen lähestymistapa ja lisäksi dokumentaatio koettiin rakenteeltaan paremmaksi ja helpommin tulkittavaksi. Testillä oli kuitenkin vakava rajoitus: testikäyttäjät eivät kerenneet tehdä itse pelielementtejä, kuten ääniä.

Politowski et al. (2018) lähestyivät pelinkehitysprosessin parantamista mielenkiintoisesta näkökulmasta. He loivat ehdotusjärjestelmän, jonka tavoitteena on auttaa pelinkehitystiimejä rakentamaan kehitysprosesseja ehdottamalla aktiviteetteja ja piirteitä aikaisemmista projekteista, joissa on samanlainen konteksti. Ehdotusjärjestelmä ottaa syötteenä vastaan peliprojektien jälkikirjoituksia, jotka ovat muunnettu rakenteelliseen muotoon. Testien perusteella ehdotusjärjestelmä osasi kuvata hyvin valmiiden peliprojektien prosesseja. Toisaalta testien vastaajat sanoivat järjestelmän olevan hyödyllinen lähinnä suunnitteluvaiheessa, mutta ei tuotantolinjan luomisessa.

Politowski et al. (2021b) ehdottavat ratkaisuja yksittäisiin pelinkehitysprosessin ongelmiin. Aliarviointia voidaan helpottaa välttämällä luottamasta pelkkään ihmisasiantuntijuuteen ja investoimalla tietopankin (aiempien projektien historiadatasta saadun kokemuksen) rakentamiseen. Epäselvän pelisuunnitteluvision ratkaisuksi he ehdottavat olemaan hukkaamatta aikaa staattisiin dokumentteihin ja käyttämällä aikaa enemmän prototypoimiseen ja pelitestaukseen. Viihdyttämisen puute voidaan ratkaista varaamalla aikaa viihdyttävyyden löytämiseen pelimekaniikoista, parantamalla ja viilaamalla tarinaa ja taidetta sekä tekemällä laajaa pelitestausta heikkojen osakohtien löytämiseksi. Alustan ja teknologian rajoitteet voidaan havaita tutkimalla ja ymmärtämällä kohdealustoja (niiden arkkitehtuuri ja rajoitteet) ennen projektin aloittamista. Pelisuunnittelun kompleksisuutta voidaan keventää pitämällä pelisuunnittelu yksinkertaisena, huomioimalla pelaajien odotukset, käyttämällä jatkuvaa pelitestausta ja hankkimalla palautetta pelaajilta. Huonot tai puuttuvat työkalut voidaan ratkaista varaamalla aikaa useampien työkalujen ja pelimoottoreiden testaukseen ennen niiden valitsemista. Väärin muodostettujen tiimien ongelmia voidaan ratkaista harkitsemalla tiimin rakenteiden tai prosessien vaihtamista, mikäli tiimien tai osastojen välillä on kommunikaatio-ongelmia.

Yhteenvetona prosessien parantamisesta aiemmassa tutkimuksessa huomataan, että ketterät kehitysmenetelmät ovat aineistossa suosittuja keinoja. On kuitenkin huomattavaa, että pelinkehityksen uniikkien piirteiden vuoksi puhtaasti

ohjelmistokehitykseen tarkoitetut käytännöt eivät toimi sellaisenaan pelinkehityksessä, joten tutkijat ehdottavat monia erilaisia variaatioita ja parannuksia muun muassa Scrum-kehitysmenetelmään. Luovan prosessin parantamiseen mainittiin lähinnä pelimoottoreiden tarjoamat keinot, kuten dynaaminen sisällön lataaminen ja skriptaus. Vaatimusmäärittelyn parantamiseen voidaan käyttää esimerkiksi muunneltua pelisuunnitteludokumenttia ja tekemällä laajaa pelitestausta. Kaiken kaikkiaan pelinkehittäjien käyttämien epämuodollisten keinojen muuntaminen rakenteellisemmaksi vaikuttaisi tutkimusten mukaan hyödyttävän kehitystä.

7. Pohdinta

Esiteltyjen tutkimustulosten perusteella on nähtävissä, että pelinkehittäjät käyttävät kehitystyössään omia, epäformaaleja prosessimalleja. On mielenkiintoista huomata, että erityisesti vanhemmissa tutkimuksissa ja kirjallisuudessa pelinkehitys nähdään vain ohjelmistokehityksenä, johon on ”liimattu” päälle taiteellinen tai luova osa. Toisin sanottuna pelien on ajateltu olevan enimmäkseen ohjelmistoja. Myöhemmissä tutkimuksissa kuitenkin nähdään, että pelinkehitystä on alettu pitämään omana alanaan ja luovan osuuden merkitys on kasvanut. Tämä on kehitysprosessin kannalta tärkeä asia, koska useassa tutkimuksessa mainittu omaperäisyys, innovaatio ja luovuus ovat peleissä tärkeitä tavoitteita. Pelien lisääntynyt monimutkaisuus, alan kilpailu ja pelaajien korkeat odotukset ovat tehneet menestyvien pelien tekemisestä yhä vaikeampaa. Pelimoottorit tarjoavat pelinkehittäjille lajityyppikohtaiset toiminnalliset vaatimukset, joten tunnepohjaisten vaatimusten osuus on huomattavasti suuremmassa merkityksessä. Nämä vaatimukset syntyvät erityisesti luovista osista: tarinasta, immersiosta, grafiikasta, äänistä ja niin edelleen.

Pelinkehitys vaatii ihmisiä sekä teknisiltä että puhtaasti luovilta aloilta. Tutkitusta aineistosta on kuitenkin nähtävillä, että luovan prosessin tutkiminen on jäänyt hyvin vähälle huomiolle. Luovan osuuden hallinta on vaikeaa, koska kontrollia pidetään luovuutta heikentävänä. Jopa ketterien kehitysmenetelmien käyttäminen nähdään luovan alan ihmisten silmissä kankeana. Vaikuttaisi siltä, että luovaa osuutta ja sen prosessia pitäisi tutkia enemmän ja kehittää konkreettisia käytäntöjä ja ohjenuoria sen parantamiseksi. Tutkimuksissa mainittiin luovuuden hallintaan käytettävän pelimoottoreita: tämä on kuitenkin selkeästi myös ohjelmistopuolen näkökulmasta ilmaistu asia, koska tutkimuksissa esitetyt luovan alan ongelmat liittyvät mielestäni enneminkin sisällöntuotantoon ja erilaisten alojen toisiinsa yhdistämiseen. Esimerkiksi käsikirjoittajien tekemä työ poikkeaa merkittävästi graafikon työstä, ja nämä puolestaan poikkeavat merkittävästi vaikkapa ohjelmoijan työstä. Eri alojen käytännöt ja aikataulut eivät sovi toisiinsa, ja on tyypillistä, että esimerkiksi ohjelmoijat joutuvat odottamaan grafiikan luomista, tai toisin päin.

Vaatimusmäärittelyn monimutkaisuus mainittiin useissa tutkimuksissa. Erityisen suuri ongelma oli pelisuunnitteludokumentin muuntaminen vaatimusdokumentiksi. Käytännössä tämä tarkoittaa vapaamuotoisen pelin korkean tason kuvausten muuntamista rakenteelliseksi, tekniseksi dokumentiksi, josta puolestaan voidaan johtaa esimerkiksi eri aloille liittyvät tehtävät. Kaiken kaikkiaan tämän kaltainen muuntaminen ei näyttäisi päälle päin eroavan ohjelmistokehityksen tavasta muodostaa käyttäjien vaatimuksista virallista vaatimusmäärittelydokumenttia. Asiassa on kuitenkin pieni mutta merkittävä ero: vaatimukset eivät tule kohdeyleisön edustajilta (pelaajilta), vaan pelisuunnittelijalta. Toisin sanottuna pelaajat itse eivät määrittele peleille vaatimuksia. Tästä päätellen pelisuunnitteludokumentti vaikuttaisi olevan syypäänä vaikeuksiin, joten esimerkiksi Mitre-Hernández et al. (2016) määrittelemä iGDD vaikuttaa hyvältä idealta. iGDD-dokumentille tulisi kuitenkin suorittaa laajempi tutkimus esimerkiksi tapaustutkimuksen avulla, jotta saadaan konkreettisesti selville sen tuomat hyödyt. Toisaalta Politowski et al. (2021b) eivät suosittele staattisten dokumenttien käyttämistä kehitysvaiheessa, vaan käyttämään aikaa enemmän prototypointiin ja pelitestaukseen. Pelisuunnittelusta tekee haastavaa myös Engström et al. (2018) mainitsema asia: jokaisen pelin pitäisi olla uniikki. Vaikkakin esimerkiksi pelien jatko-osissa voidaan hyödyntää edellisen pelin komponentteja, jatko-osassa täytyy silti olla jotain erilaista ja uutta edelliseen osaan

verrattuna. Tämän lisäksi Politowski et al. (2021b) sanovat pelisuunnittelun olevan vaikeaa, koska pelisuunnittelulle ei ole selkeää prosessia.

Koska pelien vaatimukset ovat subjektiivisia, ainoa tapa suorittaa validointitason testausta koko tuotteelle on tehdä käyttäjätestausta. Käyttäjätestaus itsessään on myös ohjelmistopuolella yleistä, mutta tavoitteet ovat erilaiset. Tyypillisesti ohjelmistokehityksessä testataan toiminnallisia vaatimuksia ja laatuvaatimuksille on selkeät metriikat – yleisesti ottaen koko testausprosessi on rakenteellinen ja teoriapohjainen. Pelien vaatimuksissa puolestaan ei ole olemassa selkeitä metriikoita: esimerkiksi, miten mitataan tärkein vaatimus ”onko peli viihdyttävä?” Tutkimuksissa mainittiin, että myös pelinkehittäjien pitäisi noudattaa formaaleja käytäntöjä pelien testauksessa. Kysymykseksi jääkin, millaisia käytäntöjen pitäisi olla, koska esimerkiksi Politowski et al. (2021b) totesivat, että pelitestauksen käytäntöjä (tai niiden puutetta) pitäisi tutkia enemmän, koska he eivät löytäneet ollenkaan mainintoja ohjelmistotestauksesta, kuten yksikkö- ja integraatiotestauksesta. Tämän lisäksi pelien testausta tehdään tyypillisesti testaussessioiden avulla ja ”ad-hoc”-tyylillä, luottaen pelitestaajien tuntumaan ja järkeilyyn. Automaation puute johtaa pelitestauksen skaalautumattomuuteen.

Monet tutkijat ehdottavat tyypillisesti ketteristä kehitysmenetelmistä ja ohjelmistokehityksen parhaista käytännöistä muunneltuja versioita avuksi pelien kehittämiseen. Tutkimuksissa esitettiin useita muokattuja versioita Scrum-menetelmästä. Niiden tuomat hyödyt käytännössä sen sijaan jäivät hyvin vähälle huomiolle. Ketterät kehitysmenetelmät tuntuvat olevan muutenkin jo paljon käytössä pelinkehitysprosessin eri vaiheissa. On myös huomattavissa, että kehitysmenetelmiä ehdottavat tutkijat ovat tehneet ohjelmistokehityksen näkökulmasta omat ehdotuksensa. Esimerkiksi Musil et al. (2010) sanovat luovan osuuden olevan vaikeaa sovittaa heidän ehdottamaansa prosessimalliin, ja antavat vain ehdotuksen esimerkiksi Scrum-Ban-yhdistelmän käyttämisestä luovan osuuden tuottamisessa. Myöskään Furtado et al. (2011) tuotantolinjassa ei varsinaisesti käsitelty pelin luovan sisällön tuottamiseen liittyviä tekijöitä, vaan ennemminkin teknistä puolta, kuten pelin arkkitehtuuria. Toisaalta heidän tuotantolinjansa käyttäminen mahdollisti toisen tärkeän osa-alueen: omaperäisyyden löytämisen pelille.

Tärkeänä huomiona pidän sitä, että vaikka tässä kirjallisuuskatsauksessa pelinkehitys vaikuttaa kokonaisuudessaan hyvin monimutkaiselta ja hajanaiselta alalta, on silti muistettava, että sekä rahallisesti että laadullisesti menestyneitä pelejä julkaistaan kuitenkin vähän väliä. Uskoisin, että erityisesti monet suuret peliyhtiöt ovat löytäneet menestymiseen liittyvät tekijät kehitysprosesseissaan, mutta eivät halua paljastaa näitä julkisesti kilpailuedun säilyttämisen vuoksi. Toimiva prosessimalli on valtti, jolla peliyritykset kykenevät säilyttämään kilpailullisuuden epävakailla markkinoilla. Kokonaisuudessaan prosessien ongelmat eivät näyttäisi johtuvan niinkään alalla työskentelevien teknisen ja luovan alan ihmisten taidon puutteesta, vaan hallinnallisista ongelmista. Täten prosessien rakenteettomuus saattaa olla merkki tuottajien kokemuksen tai tiedon puutteesta. Tutkijoiden ehdotukset muokatuista prosessimalleista ovat askel oikeaan suuntaan, mutta luovan ja taiteellisen osuuden huomioiminen prosessissa tulisi olla jopa korkeammalla prioriteetilla.

8. Yhteenveto

Tämän tutkimuksen tavoitteena on selvittää pelinkehityksessä käytettäviä prosesseja sekä niissä esiintyviä ongelmia ja ratkaisuja. Tutkimuskysymyksenä on ”miten pelinkehitysprosessia on yritetty parantaa?” Tutkimus suoritettiin kirjallisuuskatsauksena.

Peliyhtiöiden salaileva luonne tekee tutkimuksen tekemisestä vaikeaa. Useassa tutkimuksessa tiedon lähteenä ovat pelinkehittäjien kirjoittamat vapaamuotoiset jälkiselvittelyt. Jälkiselvittelyiden rajoituksina mainitaan kuitenkin 1) jälkiselvittely on kirjoitettu vain julkaistuille ja useimmiten kaupallisesti menestyneille peleille, 2) jälkiselvittelyissä ei välttämättä kerrota koko totuutta ja 3) jälkiselvittelystä ilmenee vain kirjoittajan roolin näkemys prosessista. Hyvin toimiva kehitysprosessi on kilpailuvaltti, jolla peliyhtiö voi kilpailla epävakailla markkinoilla, joten sen paljastaminen julkisesti ei ole yhtiölle kannattavaa. Tästä huolimatta tutkijat ovat löytäneet pelinkehittäjien suosimia käytäntöjä ja niihin liittyviä ongelmia, joiden perusteella erilaisia ratkaisuja ehdotetaan.

Pelinkehityksessä ei ole olemassa standardia prosessimallia, mutta kehitysprosessi on yleensä jaettu vaiheisiin, joihin myös kehittäjien ja julkaisijoiden väliset sopimukset perustuvat. Erikoista kyllä, vaikuttaisi siltä, että osa pelinkehittäjistä käyttää yhä vesiputousmallia. Toisaalta ketterät kehitysmenetelmät ovat yleistyneet: eniten käytettyjä kehitysmenetelmiä ovat Scrum ja eXtreme Programming. Tyypillisesti pelinkehittäjät kuitenkin käyttävät muokattuja versioita näistä, koska niiden katsotaan olevan sellaisenaan huonosti pelinkehitykseen sopivia. Syynä tähän on se, että ohjelmistokehitys ei huomioi peleille tärkeitä aspekteja, kuten omaperäisyyttä, innovaatiota, estetiikkaa ja viihdyttävyyttä. Aiemmista tutkimuksista löydettiin useita muokattuja versioita ketteristä kehitysmenetelmistä, joiden tavoitteena on lisätä pelinkehityksen uniikit osa-alueet osaksi kokonaisprosessia. Osa esitetyistä prosesseista, kuten videopelien kehitykseen laajennettu Scrum-prosessi tai Scrum sdPP vaikuttavat teoriassa helpottavan kehityksessä todettuja ongelmia, mutta näiden käyttämisestä käytännössä ei kuitenkaan ole tarpeeksi tietoa, jotta voitaisiin todeta niiden tuomat hyödyt.

Pelinkehitys eroaa perinteistä ohjelmistokehityksestä useilla tavoilla ja sisältää täten oman tyyppisiä ongelmia. Tutkimuksista ilmeni monialaisuuteen, projektinhallintaan ja vaatimuksiin liittyviä ongelmia. Näistä kenties merkittävin ja uniikki haaste on alan monialaisuus. Pelien kehittäminen vaatii sekä teknisiltä että täysin luovilta aloilta ihmisiä, ja näiden alojen yhdistäminen on haastavaa. Syynä tälle on alojen erilaiset metodologiat ja niiden erilaiset hallintamenetelmät. Lisäksi pelien vaatimus viihdyttämiselle eroaa merkittävästi hyötyohjelmistojen tarkoituksesta. Viihdyttävyys syntyy erityisesti luovuudesta, kuten tarinankerronnasta, taidetyylistä, pelin toiminnasta tai immersiosta. Aiemmissa tutkimuksissa mainitaan usein näistä ongelmista, mutta luovien prosessien tutkiminen on jäänyt vähäiseksi.

Pelinkehityksen ongelmat ovat muuttuneet erilaisiksi vuosien myötä. Teknologiset ongelmat ja pelisuunnitteluun liittyvät ongelmat ovat vähentyneet, mutta tiimeihin liittyvät ongelmat ovat lisääntyneet. Yhdistettynä hallinnollisiin ongelmiin tämä viittaisi siihen, että toimivan monialaisen kehitysprosessin löytäminen on ongelmallista. Osaltaan kirjallisuudessa ja tutkimuksissa esiintyvä asenne siitä, että pelien kehitys on vain ohjelmistokehitystä johtaa tilanteeseen, jossa tutkijat yrittävät tuoda ohjelmistokehityksen puolelta muunnettuja käytäntöjä pelien kehitykseen. Tämän sijaan pelien kehittäminen pitäisi ajatella omana alanaan ja käyttää ongelmien ratkaisemisessa näkökulmaa, jossa pelien luova osuus on jopa korkeammalla prioriteetilla.

Lähteet

Engström, H., Berg Marklund, B., Backlund, P., & Toftedahl, M. (2018). Game development from a software and creative product perspective: A quantitative literature review approach. *Entertainment Computing*, *27*, 10–22.

Furtado, A. W. B., Santos, A. L. M., Ramalho, G. L., & De Almeida, E. S. (2011). Improving digital game development with software product lines. *IEEE Software*, *28*(5), 30–37.

Kasurinen, J., Palacin-Silva, M., & Vanhala, E. (2017). What Concerns Game Developers? A Study on Game Development Processes, Sustainability and Metrics. *International Workshop on Emerging Trends in Software Metrics, WETSoM*, 15–21.

Koutonen, J., & Leppänen, M. (2013). How are agile methods and practices deployed in video game development? A survey into finnish game studios. *Lecture Notes in Business Information Processing*, *149*, 135–149.

Lehtonen, M., Lu, C., Nummenmaa, T., & Peltonen, J. (2019). Adoption of Requirements Engineering Methods in Game Development: A Literature and Postmortem Analysis. *Interactivity, Game Creation, Design, Learning, and Innovation*, 436–457.

Mitre-Hernández, H. A., Lara-Alvarez, C., González-Salazar, M., & Martín, D. (2016). Decreasing rework in video games development from a software engineering perspective. *Advances in Intelligent Systems and Computing*, *405*, 295–304.

Murphy-Hill, E., Zimmermann, T., & Nagappan, N. (2014). Cowboys, ankle sprains, and keepers of quality: How is video game development different from software development? *Proceedings - International Conference on Software Engineering*, *1*, 1–11.

Musil, J., Schweda, A., Winkler, D., & Biffl, S. (2010). Improving Video Game Development: Facilitating Heterogeneous Team Collaboration through Flexible Software Processes. In A. Riel, R. O'Connor, S. Tichkiewitch, & R. Messnarz (Eds.), *Systems, Software and Services Process Improvement* (Vol. 99, pp. 83–94). Springer Berlin Heidelberg.

Osborne O'Hagan, A., Coleman, G., & O'Connor, R. V. (2014). Software Development Processes for Games: A Systematic Literature Review. *Communications in Computer and Information Science*, *425*, 182–193.

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

Politowski, C., Fontoura, L. M., Petrillo, F., & Guéhéneuc, Y.-G. (2018). Learning from the past: A process recommendation system for video game projects using postmortems experiences. *Information and Software Technology*, *100*, 103–118.

Politowski, C., Fontoura, L., Petrillo, F., & Guéhéneuc, Y.-G. (2016). Are the Old Days Gone? A Survey on Actual Software Engineering Processes in Video Game Industry. *Proceedings - International Conference on Software Engineering*, *16-May-2016*, 22–28.

Politowski, C., Petrillo, F., Montandon, J. E., Valente, M. T., & Guéhéneuc, Y.-G. (2021a). Are game engines software frameworks? A three-perspective study. *Journal of Systems and Software*, *171*.

Politowski, C., Petrillo, F., Ullmann, G. C., & Guéhéneuc, Y.-G. (2021b). Game industry problems: An extensive analysis of the gray literature. *Information and Software Technology*, *134*.

Rabin, S. (2009). *Introduction to game development* (2nd ed). Course Technology/Cengage Learning.

Schmalz, M., Finn, A., & Taylor, H. (2014). Risk management in video game development projects. *Proceedings of the Annual Hawaii International Conference on System Sciences*, 4325–4334.

Wang, A. I., & Nordmark, N. (2015). Software architectures and the creative processes in game development. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *9353*, 272–285.

Washburn, Jr., M., Sathiyanarayanan, P., Nagappan, M., Zimmermann, T., & Bird, C. (2016). What went right and what went wrong: An analysis of 155 postmortems from game development. *Proceedings - International Conference on Software Engineering*, 280–289.