jamk.fi

Jatkuva integrointi pelijulkaisussa

Julkaisun ja projektijohtamisen tehostaminen

Juha Möttönen

Opinnäytetyö
Joulukuu 2016
Luonnontieteiden ala
Tradenomi (AMK), tietojenkäsittelyn tutkinto-ohjelma

Jyväskylän ammattikorkeakoulu
JAMK University of Applied Sciences

jamk.fi

Kuvailulehti

Tekijä(t)	Julkaisun laji	Päivämäärä
Möttönen, Juha	Opinnäytetyö, AMK	Joulukuu 2016
	Sivumäärä 47	Julkaisun kieli Suomi
		Verkkojulkaisulupa myönnetty: x

Työn nimi
Jatkuva integrointi pelinkehityksessä
Julkaisun ja projektijohtamisen tehostaminen

Tutkinto-ohjelma
Tietojenkäsittelyn tutkinto-ohjelma

Työn ohjaaja(t)
Niko Kiviaho

Toimeksiantaja(t)
Kinahmi Games OY

Tiivistelmä

Julkaise nopeasti ja usein. Tarve kehittää työskentelyä peliprojekteissa loi tarpeen jatkuvan integroinnin tutkimiseen projektinhallinnan kautta. Tutkimuksen tavoitteena oli toimittaa toimeksiantajalle näkemystä, miten peliprojektien julkaisun eri työvaiheet voisivat hyötyä jatkuvan integroinnin käyttöönotosta. Toimeksiantajalle asian tutkiminen oli tärkeää, koska toimeksiantaja oli julkaissut ensimmäisen tuotteensa suosittuun pelien jakelualustaan vuonna 2015. Organisaatio tarvitsi näkemystä, kuinka tehostaa toimintaa jatkossa. Tavoitteeseen pääsy vaati teorian, dokumentaation ja haastattelujen tutkimista. Tämän lisäksi tuli pystyttää palvelu mittausta varten tutkimuksen pohjalta. Koska tavoite oli ilmiö, tutkimuksessa käytettiin laadullista kehittämistutkimusta.

Jatkuvan integroinnin käyttöönotto voisi tehostaa toimeksiantajan prosesseja, jos jatkuva integrointi otettaisiin käyttöön. Tutkimuksen aikana ilmeni, miten jatkuva integraatio saadaan sulautettua ketteriin projektinhallintamenetelmiin. Tuloksista ilmeni, kuinka työskentelyvaiheita voidaan automatisoida sekä tehostaa käytettäessä jatkuvaa integraatiota. Projektin hallinnan ulottuvuudet käytettäessä jatkuvaa integraatiota mahdollistavat esimerkiksi tehokkaamman suunnittelun Sprint Planningissä sekä kommunikaation parantamisen Daily Stand Up-tapaamisissa.

Ei ole olemassa vahvaa syytä, miksi itsenäisen peliyrityksen ei tulisi vakavasti harkita jatkuvan integroinnin käyttöönottoa. Tutkimuksen aikana selvinneet ongelmat ja riskit rajoittuvat vain ajankäytölliseen maailmaan, ja ongelmat ovat yleensä vain organisaation asenteista kiinni. Tutkimuksessa suoritettu mittaus antoi viitteitä kuinka paljon jo tuotteen julkaisu jakelualustaan tehostuu jatkuvan integroinnin käyttöönoton myötä.

Avainsanat (asiasanat)

Jatkuva, integrointi, peli, kehitys, organisaatio, hallinta, projekti, julkaisu

Muut tiedot

jamk.fi

Description

Author(s)	Type of publication	Date
Möttönen, Juha	Bachelor's thesis	December 2016
		Language of publication: Finnish
	Number of pages: 47	Permission for web publication: x

Title of publication

Continuous Integration in game development

Project deployment and management enhancement

Degree programme

Business Information System

Supervisor(s)

Kiviaho, Niko

Assigned by

Kinahmi Games Ltd.

Abstract

Release early, release often is today's rule in software development. The need to study continuous integration in the use of game development organizations emerged from Kinahmi Games' actual need. Kinahmi Games launched their first video game to PC platforms late 2015. Kinahmi Games needed to find out how to make their performance within releases of the new product increments better and less time consuming with the constant delivery of the new versions of the product. The objectives were to find out how to implement continuous integration to a game development company, what could be the benefits of using it, and how continuous integration might enhance project management in product version publishing. The objectives were met by researching the theory of the continuous integration, collecting data of the current situation within Kinahmi Games and implementing the continuous integration tools for the organization's workflow. With these tasks, using mixed methodology, the assignee origination should have a better view how continuous integration allows the organization members work more efficiently.

The results of this thesis pointed towards the benefits of using the continuous integration to save individual employees' worktime. The project management and project workflow were multiple times more performant after the continuous integration implementation. The results were based on the meters implemented before and after the continuous integration pipeline integration to workflow.

There is no valid reason why any organization should not consider implementing continuous integration. Automating the process provides more efficient workflows for all the individuals within company. Continuous integration provides also new ways to monetize the workflow in other areas than in game development.

Keywords/tags (subjects)

Continuous, integration, game, development, organization, management, project, delivery

Miscellaneous

Sisältö

Käsitteet 3
1 Johdanto 5
2 Tutkimusasetelma 6
2.1 Metodologia 7
2.2 Jatkuvan Integraation vaikutuksen mittaaminen 12
3 Jatkuva integraatio 14
3.1 Jatkuvan integroinnin haasteet 15
3.2 Jatkuvan integroinnin palvelut 16
4 Mikä on indie-peliyritys? 20
5 Jatkuvan integroinnin käyttöönotto 23
5.1 Alkutilanteen mittaus 24
5.2 Jatkuvan integroinnin infrastruktuuri 26
5.3 Asennus 29
6 Johtopäätökset 33
7 Pohdinta 37
Lähteet 44

Kuviot

Kuvio 1. Opinnäytetyön triangulaatio 9
Kuvio 2. Alku- ja lopputilan mittausprosessin vaiheet tuotteen julkaisussa. 12
Kuvio 3. Jatkuvan integroinnin työskentely 16
Kuvio 4. JIRA Issue Collector 18
Kuvio 5. Kehitysympäristön infrastruktuuri 27
Kuvio 6. Jatkuvan integroinnin palvelinarkkitehtuuri isommalle organisaatiolle 28
Kuvio 7. Jatkuvan integroinnin käyttöönoton haasteet organisaatioissa. 39

Taulukot

Taulukko 1. Lähtötilanteen mittauksessa mitatut ajat. 24

Taulukko 2. Lopputilanteen mittauksessa mitatut suoritusajat. 32

Käsitteet

Scrum on projektinhallinnan kehys, jossa henkilöt voivat keskittyä hankalasti adaptoitaviin ongelmiin. Scrum tähtää tehokkuuteen ja tuotejulkaisuihin tarjoten parasta mahdollista laatua. (Takeuchi & Nonaka 1986.) Scrumin tiimi koostuu tuotteen omistajasta, scrummasterista sekä kehittäjätiimistä. (Certified Scrum Master n.d.)

Product owner ajaa tuotteen ominaisuuksia scrumissa kohti liiketoiminnallista hyötyä asiakkaalle. Product owner toimii yhteistyössä projektinhallinnan, markkinakartoituksen ja muiden tukevien osien välillä. Product ownerin tarkoituksena on varmistaa, että kehittäjät tekevät oikeita asioita asiakkaan tarpeiden mukaisesti. (Certified Scrum Master n.d.) Tässä opinnäytetyössä Product ownerista käytetään suomenkielistä termiä tuotteen omistaja.

Scrum master työskentelee kehitystiimin fasilitoijana sekä rajapintana tuotteen omistajan ja kehittäjien välillä. Scrum master toimii valmentajana scrumin projektinhallinnan kehyksessä. (Certified Scrum Master n.d.) Tässä opinnäytetyössä Scrum masterista käytetään termiä scrummaster.

Sprint on kehityksessä tietty lyhyt ajanjakso. Sprintin pituus voi vaihdella yhden viikon ja kolmenkymmenen päivän välillä. Sprint koostuu useista välivaiheista: Sprint Planning, Daily Scrum, Sprint Review sekä Sprint Retrospective. (Cerified Scrum Master n.d.) Tässä opinnäytetyössä Sprintistä käytetään termiä sprint.

Sprint planning aloittaa sprintin. Sprintin suunnittelu on jaettu kahteen osioon. Ensimmäinen osio on sprintin tehtävien määrän määrittely. Toisen vaiheen tarkoituksena on tavat toteuttaa sprintin tehtävät sprintin aikarajassa. (Certified Scrum Master n.d.) Tässä opinnäytetyössä Sprint planningistä käytetään termiä sprintin suunnittelupalaveri.

Sprint Review tarkoittaa ensimmäistä scrum sprintin päättävää tapaamista. Sprint Review- tapaamisen tarkoituksena on käsitellä sprintissä tehty versio tuotteesta. Tapaamisen tarkoituksena on erityisesti antaa tietoa tuotteen omistajalle, mutta myös antaa tietoa esimerkiksi markkinointitiimille. (Certified Scrum Master n.d.) Tässä opinnäytetyössä Sprint Reviewistä käytetään termiä sprintin katselmointi.

Daily stand up on päivittäinen scrumin kehittäjien järjestämä lyhyt palaveri. Palaverin tarkoituksena on päivittää tiedot meneillä olevista tehtävistä sprintissä. (Certified Scrum Master n.d.) Tässä opinnäytetyössä Daily stand up –tapaamisesta käytetään termiä päiväpalaveri.

Scrum Backlog Grooming on scrum-tiimien tapaaminen, jossa suunnitellaan ennakoivasti tulevien sprinttien tehtäviä. Scrum Backlog Grooming ei ollut alkuperäisesssä scrumin määrityksessä mukana, mutta Ken Swhaber suosittelee Grooming-tapaamisen pitämistä. Scrum Backlog Gooming tulisi pitää jokaisessa sprintissä Swhaberin mukaan. (Scrum Backlog Grooming n.d.) Tässä opinnäytetyössä Scrum Backlog Groomingistä käytetään termiä kehitysjonon työstöpalaveri.

Steam on yksi vanhimmista ja isoimmista tietokonealustoihin keskittyvistä jakelualustoista. Nykyään Steam-palvelulla on monia pelejä, pelaajien välisiä kommunikointivälineitä, elokuvia ja muuta. (Wilson 2016.) Toimeksiantaja välittää omaa tuotettaan Steam-palvelussa.

Unity3D on pelikehitysympäristö, jolla tehdään pelejä Windows- ja macOS -alustoilla. Unity3D mahdollistaa ohjelmien kääntämisen suoraan yli kahdellekymmenelle alustalle. (Create games, connect with your audience and achieve success n.d.) Toimeksiantajan tuotteet luodaan Unity3D-ohjelmalla.

1 Johdanto

Jatkuva integraatio on verrattain nuori käsite ohjelmistotuotannon saralla. Käsite jatkuva integraatio esiintyi ensimmäistä kertaa vuonna 1991 Grady Boochin toimesta (Booch 1991, 209). Tämän jälkeen jatkuvasta integroinnista on linkitetty useasti ketteriin kehitysmenetelmiin, vaikka konsepti vain muutti ohjelmistokehityksen käytänteitä (Lee 2005).

Jatkuvan integroinnin käyttäminen pelinkehityksessä on yleistynyt nopeasti käsitteen käyttöönoton jälkeen. Fabian Röken sekä Dag Frommhold (2005) kertovat Gamasutra-artikkelissaan jatkuvan integroinnin hyödyistä omien kokemuksien pohjalta. Artikkelissa todetaan tuotekehityksen laatuvaatimuksissa olleen ongelmia, joiden takia projektit voisivat joutua julkaisuun joko keskeneräisinä tai viallisina. Jatkuvan integroinnin avulla projekteissa voitiin ottaa tarkemmat testaukset käyttöön ennen uusien versioiden julkaisua. Myös Fewster ja Graham (1999, 346-347) kertovat, kuinka automaattisten testien avulla riskit pienenevät, koska testien suorittamisen työvaiheita ei laiminlyödä inhimillisistä piirteistä johtuen. Tämän opinnäytetyön aiheena on pohtia jatkuvan integroinnin mahdollisuuksia indie-peliyrityksen projektinhallinnassa tuotteiden julkaisujen yhteydessä. Opinnäytetyössä ei keskitytä luvussa kaksi esitetyn rajauksen vuoksi testauksen automaatioon, koska tavoitteena on tutkia konkreettista hyötyä julkaisutyövaiheisiin. Toimeksiantaja suorittaa yksikkötestit ennen julkaisutoimenpiteitä.

Kuinka paljon jatkuva integrointi yhdistettynä jatkuvaan julkaisuun voi tehostaa projektien tuotekehitystä? Voiko jatkuvan integroinnin avulla saavuttaa mitattavia hyötyjä verrattuna tilanteeseen, jossa jatkuvaa integrointia ei käytetä? Ongelma monesti pelialan yrityksissä on skeptisyys saatuja hyötyjä kohtaan (Frommhold & Röken 2005). Opinnäytetyön aikana pyritään etsimään höytyjä peliprojektien ja projektien hallinnan tehostamiseen.

Opinnäytetyön aihe on tällä hetkellä kiinnostava, koska Suomen hallitus antoi digitalisaation kehittämisen yhdeksi hallituksen kärkihankkeista (Ratkaisujen Suomi 2015). Jotta digitalisaatiossa voidaan edetä, tulee muutoksen tapahtua erityisesti pienissä ja keskisuurissa yrityksissä, jotka voivat tulevaisuudessa olla uusia Suomen Nokioita. Koska peliteollisuus on ollut 2000-luvun ajan nopeimmin kasvava

viihdeteollisuuden haara (Tietoja toimialasta 2016), jatkuvan integroinnin kehittäminen indie-peliyrityksissä voi nostaa yritysten kilpailukykyä Suomessa ja maailmalla.

2 Tutkimusasetelma

Toimeksiantaja

Opinnäytetyön toimeksiantajana toimii Kinahmi Games OY. Kinahmi Games on jyväskyläläinen pelialan yritys, joka on perustettu vuonna 2014. Kinahmi Gamesin ensimmäinen tuote on julkaistu Steam Early Access -palvelussa elokuussa 2015 (Galactic Conquerors on Steam, n.d.). Early Access on Steam-jakelualustan palvelu, jossa asiakkaat pääsevät osallistumaan pelien kehittämiseen ennen virallista julkaisua. Early Access ei tarkoita pelin ennakko-ostoa, vaan peli ostetaan kuten se olisi virallisesti julkaistu. (Mitä tarkoittaa Early Access? n.d.)

Toimeksiantajan kehittäjät työskentelevät viikoittain uusien versioiden parissa Steam-palvelussa, kun kehittäjät testaavat ja esittelevät uusia versioita asiakkaista valituille testaajille. Toimeksiantajan kehitystyössä käytetään Scrumia projektinhallinnassa, mikä lisää toimeksiantajan ketteryyttä reagoida muutoksiin. Koska toimeksiantajan aikaa kuluu useita kymmeniä minuutteja aina uuden version julkaisussa, toimeksiantajalla on tarve etsiä julkaisuiden ja hallinnan tehostamista.

Tutkimuksen lähtökohdat

Miten jatkuva integrointi voi tehostaa indie-peliyrityksien ketterää projektinhallintaa pelien versiojulkaisuissa? Tähän tutkimusongelmaan haetaan vastausta seuraavien kysymysten avulla.

1. Mitä on jatkuva integraatio?
2. Kuinka jatkuvaa integrointia voidaan käyttää peliprojektien versiojulkaisussa?
3. Miten jatkuvaa integrointia voi hyödyntää peliprojektien julkaisuhallinnassa?

Tutkimuksella pyritään tuomaan uutusarvoa toimeksiantajan sekä indie-peliyrityksien käyttöön, joille mahdollisesti pienillä resursseilla toimiminen tarkoittaa mahdollisimman pitkälle vietyä työn tehostamista. Opinnäytetyön tarkoituksena ei

ole rakentaa yksiselitteitä näkemystä tutkimusongelmaan. Tarkoituksena on luoda kuva, kuinka jatkuva integrointi voi tehostaa julkaisua ja projektinhallintaa indie-peliyrityksissä. Opinnäytetyössä keskitytään erityisesti indie-peliyrityksiin, joissa jatkuvaa integrointia ei ole otettu vielä käyttöön.

Rajaus

Opinnäytetyön aihealue rajataan mahdollisimman lähelle toimeksiantajan tilannetta. Tutkimusongelmaan haetaan vastausta jatkuvan integraation mahdollisuuksista toimeksiantajan hektiseen tilanteeseen. Yhdistämällä jatkuvan integraation käyttö ja projektinhallinta, opinnäytetyö voi tuoda uutuusarvoa toimeksiantajalle nykyiseen tilanteeseen ja tulevaisuuteen. Tutkimustulokset ovat käytettävissä suoraan toimeksiantajan seuraaviin projekteihin, kun projektit etenevät kehitystyössä samaan tilanteeseen.

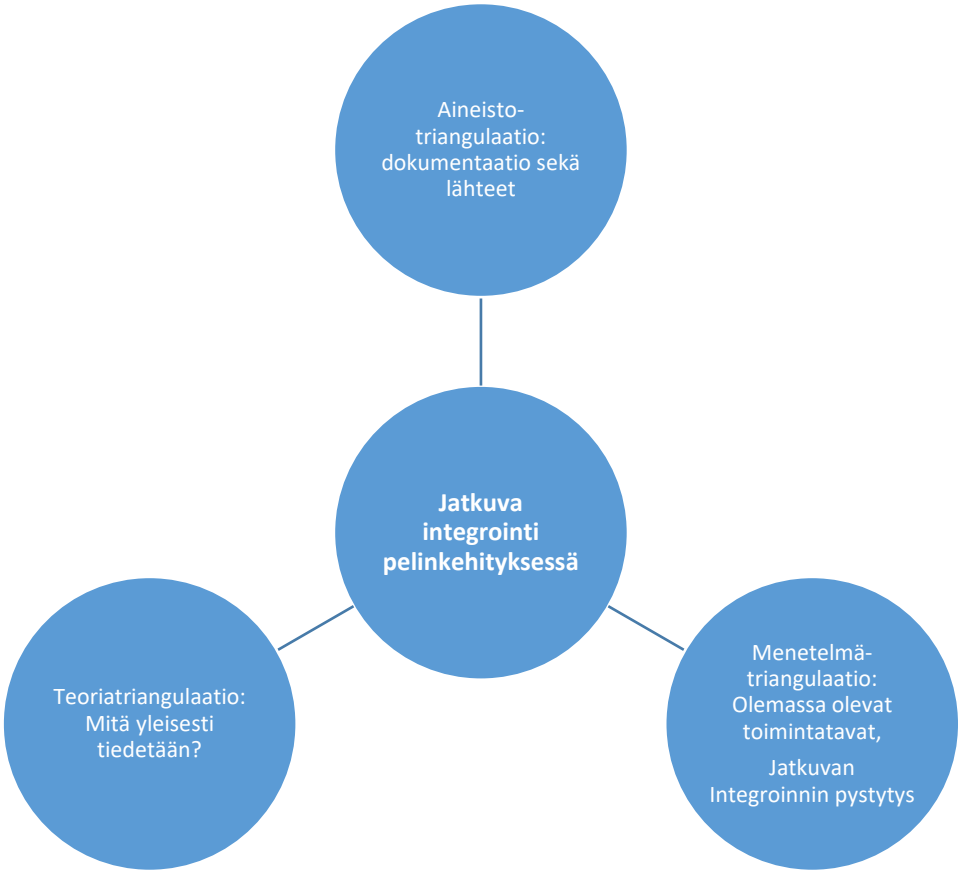
2.1 Metodologia

Koska aihealue vaatii laajaa ymmärrystä aihepiiristä, tutkimuksen luotettavuutta tutkimusilmiöön vahvistetaan käyttämällä triangulaatiota. Jos triangulaation eri lähestymistavat antavat samoja lopputuloksia, tutkimuksen johtopäätöksiä voidaan pitää luotettavina (Kananen 2014a, 121). Tämän opinnäytetyön triangulaatio käsittää vain piirteitä triangulaatiosta. Triangulaation avulla ei pyritä hakemaan täysin samoja vastauksia, mutta täydentämään tutkimusta useista eri aineistoista. Selkeyden vuoksi opinnäytetyössä puhutaan triangulaatiosta, koska tekstistä pyritään tekemään helposti luettavaa. Lukijan on hyvä huomata, että triangulaatio tarkoittaa vain piirteitä triangulattiosta.

Koska opinnäytetyössä pyritään etsimään vastausta tutkimusongelman ilmiöön, opinnäytetyön tutkimusotteeksi on valittu kvalitatiivinen eli laadullinen tutkimustapa. Opinnäytetyön aihe sisältää jatkuvan integroinnin ilmiön syvällistä ymmärtämistä projektitoimintojen eri prosessien kautta. Kvalitatiivinen tutkimusote mahdollistaa paremman tiedon keräämisen ja analysoinnin ilmiöstä, kun tiedon kerääminen ja analysointi kulkevat käsi kädessä (Kananen 2014a, 98). Kyseessä ei ole kuitenkaan puhdas kvalitatiivinen työ, vaan laadullinen kehittämistyö. Kehittämistutkimusta on

monenlaista (Kananen 2015, 33). Kehittämistutkimuksessa pyritään kehittämään tuotetta, menetelmää tai esimerkiksi organisaatiota (mts. 40), kuten tässä opinnäytetyössä. Koska toimeksiantajan prosesseja pyritään tehostamaan, kehittämistutkimuksen piirteisiin kuuluen järjestetään mittauksia ennen ja jälkeen tilanteista. Tästä opinnäytetyöstä käytetään työn aikana laadullista nimikettä selkeyden ja luettavuuden vuoksi. Koska aihealue käsittää projektinhallintaa, tekniikoita ja dokumentteja, laajempaa ymmärrystä ilmiöön haetaan triangulaatiota muistuttavan tutkimisen kautta.

Triangulaatio on eri aineistojen, menetelmien ja teorioiden käyttämistä tutkimuksessa. Yhdistelemällä eri menetelmiä tutkimuksen aikana, opinnäytetyön reliabiliteettia ja validiteettia voidaan vahvistaa. (Kananen 2014a, 121.) Tässä opinnäytetyössä käytetään aineisto -, teoria- sekä menetelmätriangulaatiota, kuten kuviossa yksi on kuvattu. Tutkimusmalliin päädyttiin, koska käytettävissä olevien resurssien määrä oli vähäinen suhteessa opinnäytetyön laajuuteen. Koska haastattelujen otanta ei tarjoa saturaatiota eli aineiston kyllästymistä, tutkimusta täydennetään triangulaatiolla. Opinnäytetyön runko on lähellä kehittämistutkimusta, koska menetelmätriangulaatiossa tutkitaan osallistuvasti palvelun kehittämistä. Kuitenkin opinnäytetyön pääpaino on havainnoinnissa sekä haastatteluissa, mikä asettaa opinnäytetyön lähemmäs laadullista tutkimusta kuin kehittämistutkimusta.



Kuvio 1. Opinnäytetyön triangulaatio

Aineisto-otteessa yhdistellään useanlaisia aineistoja keskenään (Eskola & Suoranta 2008, 69). Jotta opinnäytetyöhön saadaan validia dataa, aineistoa kerätään projektinhallinnan tuntevia henkilöitä haastattelemalla. Tämän lisäksi aineistoa kerätään tutkimalla toimeksiantajan dokumentaatiota. Aineiston keräämiseen liittyy käytännön haasteita, koska osa haastattelujen ja dokumentaation aiheista voi tulla käsittämään salassa pidettävää materiaalia. Materiaalin käyttöön liittyvä problematiikka selvitetään käyttämällä vain yleisiä ja hyväksyttyjä viittauksia kerättyyn aineistoon.

Aineiston kerääminen tapahtuu syklissä teoriaotteen kanssa. Kun teoriaotteesta nousee esiin uusia kysymyksiä liittyen aineistoon, voidaan haastateltaville esittää jatko-

kysymyksiä. Kun haastattelujen aihepiireistä nousee uusia kysymyksiä, voidaan opinnäytetyön haastatteluista saada kattavampi näkemys ilmiöön. Kanasen (2014a, 76) mukaan tämä on myös normaali ilmiö haastatteluja tehdessä. Kun aineistoa kerätään opinnäytetyön aikana, voidaan löytää uusia kysymyksiä teoriaotteen tarkoituksia varten. Tutkimusongelmaan kohdistetut triangulaatiomenetelmien tulokset vahvistuvat, kun aineiston, haastattelujen ja teorian syklinen tutkiminen mahdollistaa kehittyvän aineiston keräämisen.

Haastatteluista kerätty aineisto tullaan opinnäytetyön aikana käsittelemään litteroimalla. Aineisto koodataan propositiotasolla, jotta opinnäytetyön tutkimuksesta voidaan löytää ilmiön osa-alueita helposti. Koodauksen avulla pyritään selvittämään ilmiöön liittyviä aihealueita nykyisestä tilanteesta, sekä etsiä uusia tutkittavia aihepiirejä teoriaosuuteen. Koodauksen avulla saadaan yleisnäkemys uusiin hakusanoihin suoraan käyttämällä Kanasen (2014a, 107) esiteltyä mallia.

Teoriaotteessa oletetaan jatkuvan integroinnin käytöstä peliprojekteissa löytyvän vähän tietoa, mutta tietoa voidaan löytää yleisesti ohjelmistotuotannon saralta. Tutkimalla yleisesti tiedossa olevaa teoriapohjaa, voidaan etsiä vastauksia yleisiin kysymyksiin. Teorian avulla voidaan tutkia jatkuvan integroinnin mahdollisuuksia käyttöönottoon, käyttämiseen ja jatkokehitykseen olemassa olevalla tiedolla. Kerätyn aineiston avulla pystytään löytämään opinnäytetyön pohja. Teorian tarkoituksena on selvittää, miten jatkuvaa integrointia voidaan hyödyntää indie-peliyrityksien projekteissa. Teorian avulla pyritään löytämään vastausta, mitä on jatkuva integraatio. Ymmärtämällä, mitä jatkuva integraatio on, voidaan selvittää jatkuvan integroinnin käyttömahdollisuuksia. Teorian avulla pystytään työskentelemään paremmin menetelmäotteen parissa, kun jatkuvaa integrointia voidaan reflektoida teorian pohjalta nykyiseen käytäntöön ja tavoitetilaan.

Teoriaotteessa ilmiöön sovelletaan eri teorioita ja tieteenaloja (Kananen 2014b, 123). Tässä opinnäytetyössä teoriaa pyritään tuomaan tietotekniikan ja johtamisen alalta osaksi opinnäytetyötä. Teoriassa tutkitaan olemassa olevaa tietoa nykyaikaisista projektinjohtamismalleista ja yhdistää tietoteknillistä tutkimusta osaksi johtamisen kulttuuria. Teoria ei ole syvällistä yliopistotason tutkimista. Keskittymällä kirjallisuuteen ja muihin lähteisiin opinnäytetyön aikana, opinnäytetyön koko pystyy kompaktina.

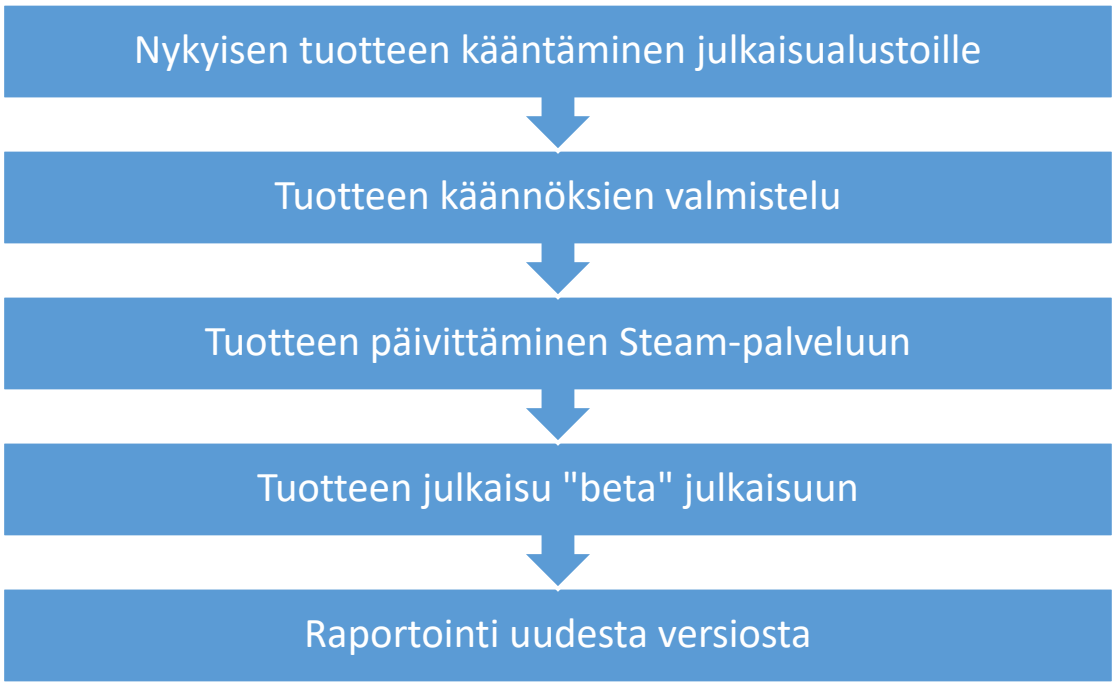
Menetelmäotteessa tutkitaan tutkimuskohdetta useilla eri aineistonhankinta- ja tutkimusmenetelmillä (mts. 70). Koska opinnäytetyön ongelma on peliprojektien tehostaminen käyttämällä jatkuvaa integraatiota, opinnäytetyön menetelmäotteessa haetaan tietoa käytännön esimerkkien kautta. Tärkeimpänä käytännön esimerkkinä toimii havainnointi toimeksiantajan nykyisistä prosesseista. Prosessien avulla pystytään havainnoimaan tämän hetkiset ajalliset pullonkaulat, eri työvaiheet projektiversioiden julkaisuissa sekä mahdolliset olemassa olevat projektityöskentelyn työvaiheet. Havainnoinnin avulla voidaan saada tutkimusta varten mittaus toimeksiantajan tilanteesta ennen jatkuvan integroinnin käyttöönottoa. Tällä tutkimuksella voidaan saada viitteellistä dataa, kuinka paljon ajallisesti jatkuvan integroinnin käyttöönotto vaikuttaa peliprojektien projektinhallintaan ja julkaisuun. Mittauksen vertailuun luodaan dataa, kun opinnäytetyön aikana pystytetty järjestelmä voidaan ottaa käyttöön toimeksiantajalla.

Menetelmäotteen tukena toimii jatkuvan integroinnin käyttöönotto, koska mittaukselle tarvitaan vastadataa ja jatkuvan integroinnin käyttöönottoa voidaan tutkia käytännön esimerkin kautta. Koska toimeksiantajalla ei ole jatkuvan integroinnin käyttöönottoa, voidaan järjestelmän pystytyksellä tutkia, vaikuttaako pystytys peliprojektien julkaisun ja projektinhallinnan tehostamiseen. On tärkeää saada tietää työaika, joka käytetään järjestelmän pystytykseen. Tiedolla voidaan tehdä johtopäätöksiä, tehostaako jatkuva integrointi pitkällä aikavälillä indie-peliyrityksien projektinhallintaa ja tuotteiden julkaisua. Tämä on tavoitetila, johon opinnäytetyön aikana pyritään. Tavoitetilan avulla voidaan suorittaa vertailua alkutilanteen kanssa ja reflektoida muutoksia, jotka johtuvat jatkuvan integroinnin käyttöönotosta.

Suorittamalla ylläolevia toimenpiteitä pyritään hyvään laadulliseen tutkimukseen. Kaikki tutkimus mitä opinnäytetyön aikana tehdään, tulee toimimaan interaktiossa toimeksiantajan kanssa. Interaktiossa tulevat olemaan teemahaastattelut, joiden avulla pyritään etsimään tärkeitä yhdistäviä tekijöitä tutkittaessa opinnäytetyön ilmiötä. Kun toimeksiantaja on kaiken toiminnan keskipisteessä havainnoinnissa, haastatteluissa ja aineistossa, voidaan resurssien puitteissa tutkia ilmiötä mahdollisimman syvällisesti.

2.2 Jatkuvan Integraation vaikutuksen mittaaminen

Opinnäytetyön tutkimusongelma jatkuvan integroinnin vaikutuksesta peliprojektien hallinnan ja julkaisun tehostamiseen, tarvitsee validia dataa. Jotta voidaan todentaa miten jatkuva integrointi vaikuttaa ajallisesti tehokkuuteen, tarvitsee saada tietää käytetty aika alkutilanteessa. Alkutilanne kertoo käytetyn ajan julkaisuun ilman jatkuvan integroinnin käyttöä. Mittauksen avulla pystytään havainnoimaan kulutettu aika tuotteen version julkaisuun yksittäiseltä henkilöltä. Mittaus käsittää kaikki työvaiheet, jotka toimeksiantaja on ilmoittanut etukäteen (katso kuvio kaksi). Mittaus käsittää kehittäjien sekä projektin hallinnallisen ajankäytön.



Kuvio 2. Alku- ja lopputilan mittausprosessin vaiheet tuotteen julkaisussa.

Toimeksiantajan ilmoittamat prosessit julkaisussa

Kuten kuviossa kaksi on havainnollistettu, uuden version päivittäminen on monivaiheinen prosessi. Toimeksiantaja on toimittanut alustavan dokumentin, jossa nämä

työvaiheet on esitetty yleisesti. Prosessit tullaan varmistamaan vielä varsinaisessa alkutilanteen mittauksessa. Mittauksen aikana pyritään havainnoimaan muita työvaiheita sekä tämän hetkistä ajankäyttöä julkaisussa. Toimeksiantajan ilmoittamat prosessit ovat

1. nykyisen tuotteen kääntäminen julkaisualustoille
2. tuotteen käännöksien valmistelu
3. tuotteen päivittäminen Steam-palveluun
4. tuotteen julkaisu beta-julkaisuun
5. raportointi uudesta versiosta.

Nykyisen tuotteen kääntäminen julkaisualustoille on projektin koostamista eri käyttöjärjestelmille. Käyttöjärjestelmät ovat Windows, Linux ja macOS. Tässä työvaiheessa kehitysympäristönä käytetty Unity3D-ohjelma koostaa käyttöjärjestelmille oman version. Koonnissa käytetään toimeksiantajan luomaa koodia käännöksien tekemiseen.

Tuotteen käännöksien valmistelu tarkoittaa toimenpiteitä ennen version julkaisua. Valmistelun aikana koontiversioiden kansiorakenne ja tietotiedostojen rakenne validoidaan manuaalisesti. Työvaiheen tarkoituksena oli estää pelin rikkoutuminen väärän rakenteen takia, koska käännöskoodi ei osannut validoida kansiorakennetta.

Tuotteen päivittäminen Steam-palveluun tarkoittaa varsinaista versiopäivitystä jakelualustalle. Tässä työvaiheessa tekijän tarvitsee käynnistää Steam-palvelun antamat ohjelmat. Nämä ohjelmat toimeksiantaja oli muokannut oman tuotteen tarpeisiin valmiiksi.

Tuotteen julkaisu beta-julkaisuun tarkoittaa manuaalista työtä Steam-palvelussa, jotta versio voidaan ottaa käyttöön testaajilla. Tässä työvaiheessa tekijän tulee kirjautua Steam-kehittäjien palveluun, ja muokata pelin Steam-dataa käyttämään uutta versiota beta-julkaisusta. Vain tietyt asiakkaat pääsevät käsiksi tuotteen versioon beta-julkaisussa.

Raportointi uudesta versiosta käsittää toimeksiantajan sisäiset toimenpiteet. Julkaisun jälkeen tekijän tulee merkitä uusi versio laskentataulukkoon. Tämä onnistuu toimeksiantajan omilla tunnuksilla, joilla on pääsy laskentataulukkoon. Laskentataulukkoa säilytetään Google Drive -palvelussa.

Lopputilanne

Lopputilanteen mittaus järjestelmän pystyttämisen jälkeen luo vertailukelpoista dataa päivittäisestä työskentelystä organisaatiossa. Lopputilanteen mittauksella voidaan luoda johtopäätöksiä esitettyyn tutkimusongelmaan konkreettisten tilastojen kautta. Näin vahvistetaan opinnäytetyön validiteettia ja reliabiliteettia.

Mittauksen tukena toimii menetelmätriangulaatio-tutkimuksessa pystytetty jatkuvan integroinnin järjestelmä. Pystytetyn järjestelmän tarkoituksena on toimia havainnoinnin apukeinona, kun järjestelmää käytetään. Suoritettavan jälkitilan mittauksen avulla voidaan havainnoida konkreettiset ajalliset vaikutukset. Palvelun pystyttäminen auttaa ymmärtämään syvällisesti vaikutuksia organisaatiossa, ei pelkästään yksittäisten tehtävien tehostamisvaikutuksia. Toimeksiantajan mukaan tämä on tärkeä tieto. Pienet vaikutukset yksittäisissä tehtävissä eivät välttämättä kompensoi palvelun pystyttämiseen käytettyä aikaa.

Koska toimeksiantajalla ei ole käytössä olevaa järjestelmää, tulee selvittää mahdolliset ongelmat ja riskit järjestelmän käyttöönotossa. Koska järjestelmän pystytys sitoo sekä kehittäjien että hallinnoijien työaikaa, tulee tietää kuinka paljon jatkuvan integroinnin käyttöönotto voi vaikuttaa negatiivisesti julkaisujen ja hallinnoimisen tehostamiseen. Toimimalla näin saadaan validia dataa toimeksiantajan kaltaisille organisaatioille, joiden tarvitsee miettiä myös kokonaisvaikutuksia eikä pelkästään päivittäisen työskentelyn vaikutuksia jatkuvan integroinnin käytöstä.

3 Jatkuva integraatio

Olemassa olevan teorian tutkimuksen tarkoituksena on löytää vastauksia erityisesti kysymykseen mitä on jatkuva integraatio. Tutkimuskysymystä lähestytään tutkimalla erityisesti olemassa olevia palveluja. Ymmärtämällä palveluja saadaan tietoa, kuinka palveluja voidaan hyödyntää, ja mitkä palvelut sopivat erityisesti indie-peliyrityksille.

Palvelujen kartoittamisella ja haastattelujen käymisellä voidaan vahvistaa menetelmätriangulaatiota, kun käytännön järjestelmän valinta on mahdollisimman lähellä toimeksiantajan toiveita ja käytettävissä olevia resursseja.

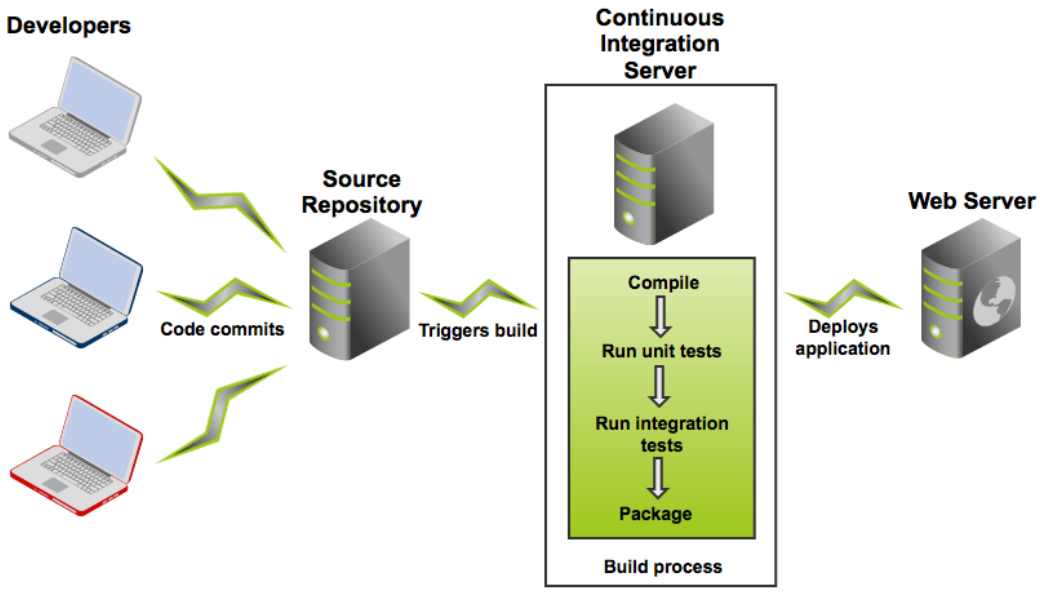
Jatkuvan integraatio jakautuu kahdeksi käsitteeksi, joiden erot ovat häilyviä ja merkitykset voivat olla tulkinnasta riippuen enemmän ja vähemmän samoja (Lukkarinen 2011, 25). Jatkuva integraatio voidaan nimetä myös jatkuvana käyttöönottona. Jatkuva käyttöönotto tarkoittaa jatkuvan integroinnin koontiprosessien luomuksien käyttöönottamista tuotantoympäristössä (Caum 2013). Jatkuva integraatio tarkoittaa vain ohjelman koontia, esimerkiksi Unity3D-pelin kääntämistä ja testien ajamista. Palvelut, joita esitellään luvussa 3.2 toimivat sekä jatkuvan integroinnin palveluina ja jatkuvan käyttöönoton palveluina. Palvelut käyttävät yleisesti termiä jatkuva integraatio kuvauksissaan. Tämän vuoksi tässä käytetään yhdenmukaisuuden takia käsitteenä jatkuvaa integraatiota. Opinnäytetyön aikana jatkuva integraatio käsitteenä kuitenkin tarkoittaa myös jatkuvaa käyttöönottoa.

3.1 Jatkuvan integroinnin haasteet

Jatkuvan integroinnin suurimpiin haasteisiin liittyy ajatusmalli, jossa ajankäytön tehostamista jatkuvan integroinnin avulla epäillään (Continuous Delivery Adaption Barriers 2013). Tämä on yleinen ajatus pelialalla (Fabian & Röken 2005). Toimeksiantajan mukaan yrityksien on vaikea nähdä jatkuvan integroinnin hyötyjä, koska jatkuvan integroinnin käyttöönotosta voi seurata haasteita koulutuksien ja toimintatapojen muutoksien takia. Tähän kokonaisvaltaiseen ongelmaan jatkuvan integroinnin mahdollisista vaikutuksista julkaisujen ja hallinnan tehostamiseen tutkitaan koko opinnäytetyön ajan triangulaatiota käyttäen.

Kuviossa kolme esitellään Atlassianin näkemys, miten projektin kehittäminen suoraviivaistuu käytettäessä jatkuvaa integraatiota internet-pohjaisten ratkaisujen tuottamisessa. Kuviosta nähdään, mitä työvaiheita voidaan siirtää työntekijän omalta työpisteeltä automatisoituun jatkuvan integroinnin palveluun. Kun jatkuva integrointi toimii tuotteen julkaisun vaiheiden automatisoijana, työntekijän tarvitsee vain huolehtia versioiden päivittämisestä versionhallintaan. Hallinnallisesti tärkeä laadunvar-

mistus toimii tehokkaammin, kun jatkuvan integroinnin palvelu varmistaa aina tuotteen toimivuuden ennen julkaisua. Jos tuote ei toimi jatkuvan integroinnin palvelussa, uutta versiota tuotteesta ei julkaista asiakkaille. Hallinnallisesti tämä voi tehostaa tiedon välitystä kehittäjiltä hallinnoijille, kun jatkuvan integroinnin palvelu voi ilmoittaa versioiden toimivuudesta heti projektihallintaan (Friedman 2016). Opinnäytetyön tutkimusongelmaan tämä malli antaa näkökulman, jossa hallinnallisesti projektityöskentely tehostuu, kun kehitys on avoimempaa organisaation eri osien välillä.



Kuvio 3. Jatkuvan integroinnin työskentely (Understanding the Bamboo CI Server n.d.)

3.2 Jatkuvan integroinnin palvelut

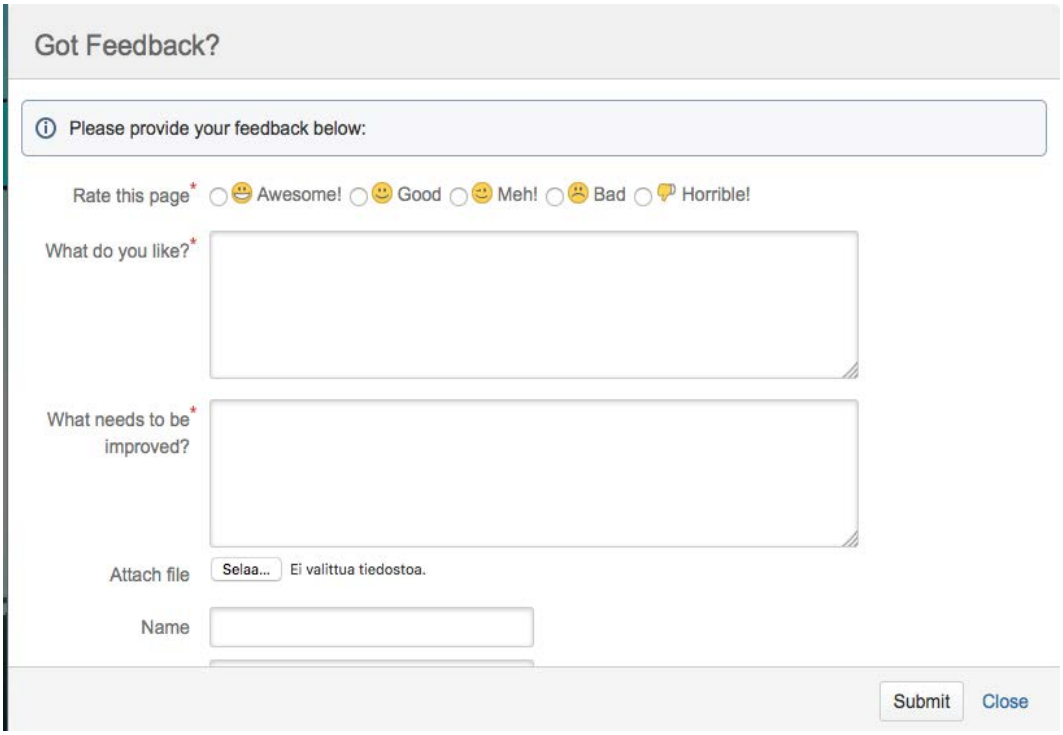
Jatkuvan integroinnin ymmärtäminen projektityöskentelyn automatisoijana on vasta ensimmäinen vaihe opinnäytetyön tutkimusongelmaan. Tutkittaessa jatkuvan integroinnin mahdollistamaa tehokkuutta, tarvitsee tietää toimeksiantajan tilanteeseen

parhaiten soveltuva palvelu. Palveluja kartoittamalla ja haastatteluja käymällä saadaan kartoitettua palvelu, joka voidaan pystyttää menetelmätriangulaatiota varten. Löytämällä spesifinen palvelu juuri toimeksiantajan ketterään peliprojektityöskentelyyn, voidaan opinnäytetyön mittauksista saada luotettavampaa dataa haastattelujen avulla. Haastattelut ovat yksi tärkeimmistä laadullisen tutkimuksen menetelmistä (Kuusela & Rintamäki 2002, 143). Kun palvelu on mahdollisimman lähellä toimeksiantajan toiveita, opinnäytetyöstä on hyötyä jatkokehityksen kannalta.

Atlassian Bamboo

Atlassian Bamboo on australialaisen Atlassian-pörssiyhtiön kehittämä jatkuvan integroinnin työkalu. Atlassian Bamboo julkaistiin ensimmäistä kertaa vuoden 2007 helmikuussa (Chaimungkalanont 2007), ja kehitys jatkuu yhä Atlassianin toimesta. Erona Bamboon ja kilpailevien tuotteiden välillä on syvä integraatio muihin Atlassianin tuotteisiin.

Atlassian Bamboon haittana on palvelun korkea hinta. Aloittelevan indie-peliyrityksen ei kannata aluksi ostaa kallista lisenssiä, koska hinta voi kymmenellä käyttäjällä olla useita satoja euroa vuositasolla (Bamboo Pricing 2016). Vakiintuneelle yhtiölle Atlassian Bamboo on kuitenkin varteenotettava vaihtoehto, jos yhtiössä on otettu tai otetaan käyttöön muitakin Atlassianin tuotteita. Hyvänä esimerkkinä toimii kuviossa neljä esitelty JIRA ”Issue Collector”, jonka avulla palvelun käyttäjät voivat lähettää virheilmoituksia sekä palautetta suoraan kehittäjille (Using the issue collector n.d.). Hallinnallisesti palvelu tehostaa itsenäisten peliyritysten peliprojektien adaptiivisuutta asiakkaiden toiveisiin ja virheilmoituksiin. Kun asiakkaat voivat lähettää suoraan järjestelmästä palautetta kehittäjille, eri julkaisujen ongelmatilanteet voidaan selvittää ilman formaaleja kyselyjä käyttäjille.



Kuvio 4. JIRA Issue Collector

Jenkins

Tällä hetkellä yksi suosituimmista jatkuvan integroinnin palveluista, joka alkujaan tunnettiin nimellä Hudson. Oraclen hallussa ollut projekti alkoi vuonna 2010 jakautua yhteisökehittäjien kesken erinäisten ristiriitojen takia. Vuonna 2011 yhteisön äänestyksellä Hudson-projektista irrotettiin erillinen Jenkins-projekti (Blewitt 2011).

Jenkins-palvelun suurin etu on sen avoimen lähdekoodin malli. Kuka tahansa voi kehittää palveluun omia lisäosiaan ja osallistua palvelun kehittämiseen ilman sitoumuksia tai ongelmia. Jenkinsille on satoja lisäosia (Berg 2015, 318), joita käytetään aktiivisesti. Jenkinsille on saatavilla useita lukuisia eri kieliversioita suomen lisäksi. Vuoden 2016 syyskuussa Jenkins-lisäosia oli asennettuna ja aktiivisena noin kuusi miljoonaa kappaletta statistiikkojen mukaan (Some statistics on the usage of Jenkins n.d.).

Lisäosien avulla Jenkins-palvelusta saa toiminnoiltaan lähes yhtä integroidun ratkaisun Atlassian JIRA-palveluun kuin Bamboo. Jenkins-palvelu ei pysty käyttämään JIRA-

palvelun käyttäjätunnuksia suoraan. Keskisuurissa organisaatioissa tämä ei kuitenkaan ole ongelma käytettäessä omaa LDAP-palvelinta, jota sekä JIRA että Jenkins pystyvät molemmat hyödyntämään. Käyttämällä LDAP-palvelinta voi palvelut liittää käyttämään yhteistä käyttäjähallintaa, jota molemmat palvelut ymmärtävät. (Connecting to an LDAP directory n.d.; Standard Security Setup 2015.)

Tutkimuksessa havaittiin, että Jenkins ei tue oikeusjärjestelmiä eri töiden välillä. Erillisillä lisäosilla on mahdollista kiertää tämä ongelma, jos organisaatiolla on tarvetta eri töiden välisille oikeuksille. Tämä kuitenkin koskee vain manuaalista töiden ylläpitämistä. Oikeusjärjestelmien käyttö on tarpeellinen tilanteissa, joissa kehittäjän ei haluta pääsevän käsiksi organisaation muihin projekteihin.

Ansible

Jatkuvan kehityksen työkalu Ansible jakautuu vapaan lähdekoodin versioon sekä maksulliseen versioon. Ansiblen avoimen lähdekoodin versiota työstää yli 1000 yhteisön jäsentä. Maksullisen Ansible Tower -version ylläpitäjänä toimii Ansible-yhtiö. Ansiblen vahva kasvu saattaa tulevaisuudessa haastaa Jenkins-projekti, kun Red Hat ilmoitti lokakuussa 2015 ostavansa Ansible-yhtiön (Novet 2015).

GitLab CI

GitLab CI on jatkuvan integroinnin palvelu, joka on integroitu suoraan GitLab-versionhallintapalveluun. GitLab CI on saatavilla sekä verkkoversiona sekä GitLabin ladatussa Enterprise-versiossa. GitLabin versionhallinnan parissa työskentely on integroitu suoraan GitLab CI -palveluun. (GitLab Features n.d.)

Palvelun suurin etu kilpailijoiden nähden on sen syvä integraatio GitLab-versionhallintaportaaliin ja Git-versionhallintaan. Palvelun käyttäminen on helppoa ja Git hook-asennukset on helppo tehdä suoraan GitLabissa. Git hook on mekanismi, jonka avulla on mahdollista käynnistää ohjelmakoodeja. Ohjelmakoodit suoritetaan, kun tietyt tärkeät tapahtumat tapahtuvat joko Git-palvelimella tai kehittäjän ympäristössä. Esimerkiksi Git-kommitointi on tärkeä tapahtuma, joka aktivoi hook-ominaisuuden. (Customizing Git – Git Hooks n.d.)

Käytettäessä GitLab CI -palvelun verkkoversiota suurin haitta on GitLabin Git-säiliön koko. Rajan voi kiertää ostamalla vuosimaksullisen GitLab Enterprise Edition -version.

Yrityksille suunnatun version avulla yritys pystyy sekä käyttämään LDAP-palvelinta ja säilyttämään teoriassa rajattomasti tietoa eri projektien versionhallinnoista. Tämä on tärkeä ominaisuus isommissa peliyrityksissä, jotka pyrkivät mahdollisimman yksinkertaiseen käyttäjähallinnointiin.

Team Foundation Server

Team Foundation Server on Microsoftin kehittämä työkalu, jonka avulla kehitystiimien on helpompi työskennellä ohjelmistoprojektien parissa. Microsoftin palvelu ei ole pelkästään erikoistunut jatkuvan integroinnin palveluihin. Sovelluskehittäjätiimit pystyvät hyödyntämään esimerkiksi ketteriä projektinhallintamenetelmiä suoraan Team Foundation Serveristä. Tuotteen asennus ja ylläpitäminen Windows -palvelinympäristöön on helppoa.

Suurin hyöty käytettäessä Team Foundation Serveriä tulee sen integroinnista Microsoftin työkaluihin. Microsoftin palvelu on tarvittaessa helposti laajennettavissa kolmansiin palveluihin, kuten Jenkins-palveluun. Pienille tiimeille Team Foundation Server tarjoaa kilpailijoitaan paremmat työkalut projektinhallintaan, koska esimerkiksi Git-palvelun käyttö on mahdollista suoraan integroiduista työkaluista.

Järjestelmän toimivuus on palvelun suurin ongelma sekä haitta. Team Foundation Server on tehty toimimaan vain Windows-ympäristössä. Tämän vuoksi Team Foundation Serverin asentaminen ilman virtuaalisoitua ympäristöä on mahdotonta muille käyttöjärjestelmille. Vaatimukset voivat asettua esteeksi, koska Windows Serverin kustannukset voivat olla liikaa indie-peliyrityksille. Muita ongelmia palvelusta ei tutkimuksen aikana ilmennyt, vaan palvelu näyttää ja tuntuu varteenotettavalta vaihtoehdolta ohjelmistokehityksen tueksi.

4 Mikä on indie-peliyritys?

Tarkkaa määritystä, mikä on indie-peliyritys eli itsenäinen peliyritys, ei ole tähän päivään asti toteutettu. Juan Gril (2008) esimerkiksi kertoo omasta mielestään Indie-peliyrityksen olevan vapaasti ajatteleva organisaatio. Grillin määrityksen pohjalta indiepeliyritys ei välttämättä ole taloudellisesti itsenäinen tai pieni. Tällä määrityksellä peliyritys voi käsittää kymmeniä työntekijöitä ja monta miljoonaa sijoittajapääomaa.

Tämä kuitenkin on hyvin erilainen muiden määritysten mukaan. Esimerkiksi Fred Dutton (2012) esittelee keräämiään pelaajien mielipiteitä, jotka esittelevät vaihtoehtoisia määrityksiä indie-peliyritykselle. Duttonin keräämien määrityksien mukaan yhtiötä voidaan kutsua itsenäiseksi, jos yhtiö on itsenäinen taloudellisesti. Duttonin esimerkissä itsenäisellä peliyrityksellä voi olla kymmeniä työntekijöitä, mutta ei saa sisältää pääomasijoituksia. Koska Grillin ja Duttonin määritykset ovat eriäviä, käytetään tässä opinnäytetyössä indie-peliyrityksestä toimeksiantajan määritystä.

Toimeksiantajan määritys itsestään indie-peliyrityksenä on itsenäinen taloudellisesti sekä päätäntävaltaisesti. Määrityksessä toimeksiantaja työllistää enintään kaksikymmentä henkilöä vakituisesti. Toimeksiantaja määrittelee indie-peliyrityksen suomenkielisellä termillä itsenäinen peliyritys. (Strategy 2016–2020 2015.) Toimeksiantajan määrityksen vuoksi tässä opinnäytetyössä indie-peliyrityksestä käytetään termiä itsenäinen peliyritys. Koska toimeksiantaja määrittää oman yrityksensä tällä tavoin, opinnäytetyössä tutkitaan jatkuvan integraation käyttämistä ketteriä kehitysmenetelmiä käyttävien itsenäisen peliyritysten näkökulmaa.

Itsenäisen peliyrityksen projektijohtaminen

Projektinhallinta on tietojen, taitojen, työkalujen ja tapojen käyttämistä uniikin ja väliaikaisen työskentelyn saattamiseksi valmiiksi (What Is Project Management n.d.). Projektinhallintaan voivat vaikuttaa projektitiimin ulkopuolelta tulevat vaikutukset, jotka voivat ohjata, vaikuttaa tai rajata projektitiimin työskentelyä (A Guide to the Project Management Body of Knowledge 2013). Itsenäisen peliyrityksen projektijohtaminen on näiden määrityksien mukaista toimeksiantajan mukaan. Peliprojektit ovat aina uniikkeja ja projektit kestävät vain tietyn ajan. Projekteilla on viisi vaihetta Project Management Instituten mukaan

1. määrittely
2. suunnittelu
3. tuottaminen
4. monitorointi ja ylläpito
5. lopetus. (What Is Project Management n.d.)

Tässä opinnäytetyössä keskitytään kolmanteen ja neljänteen vaiheeseen. Koska toimeksiantajan ensimmäinen tuote on Steam Early Access-julkaisussa, projekti on kolmannen ja neljännen tason välimaastossa. Tämän vuoksi tässä opinnäytetyössä keskitytään tilanteeseen, jossa asiakkaille tarjotaan uusia päivityksiä kehitystyön ollessa vielä kesken.

Ketterä kehitys on muodostunut muotisanaksi pelinkehityksessä viime vuosina, koska ketterä kehitys mahdollistaa iteratiivisen kehityksen. Ketterässä kehityksessä ei tarvitse kattavaa suunnittelua etukäteen. (Romanos 2014.) Tässä opinnäytetyössä keskitytään ketterän kehityksen malliin, kun tutkitaan miten jatkuva integraatio voi tehostaa sekä projektinhallintaa että versiojulkaisuja. Jatkuvan integroinnin mahdolliset tehostamisvaikutukset henkilöstöjohtamiseen jätetään huomioimatta tässä opinnäytetyössä. Luvussa kaksi esitetyn rajauksen vuoksi projektinhallinnalla käsitetään opinnäytetyössä vain scrum-projektihallinta. Vaikutuksissa otetaan huomioon vain tilanne, jossa kehitystiimi on julkaissut tuotteen asiakkaille. Tuotetta päivitetään korjauksien ja ominaisuuksien osalta, mutta projekti ei ole vielä edennyt täysin monitorointi- ja ylläpitovaiheeseen.

Ketterä kehitys, mitä se on?

Ketterä kehitys on ohjelmistotuotannon projektinhallintamalli, jossa korostetaan toimivaa ohjelmistoa toimivan dokumentaation sijasta. Ketterä kehitys on valmiutta muutokseen suunnitelmien seuraamisen sijasta. (Beck, Beedle, Bennekum, Cokcburn, Cunningham, Fowler, Grenning, Highsmith, Hunt, Jeffries, Kern, Marick, Martin, Mellor, Schwaber, Sutherland & Thomas 2001.) Ketterä kehitys voi tarkoittaa esimerkiksi Kanbania (Radigan n.d.), mutta opinnäytetyön aikana keskitytään scrum-projektinhallintaan. Kinahmi Games käyttää scrumia projektinhallinnassaan (Strategy 2016–2020 2015).

Scrumin organisaatio koostuu tuotteen omistajasta, scrummasterista sekä tekijätiimistä (Radigan n.d.). Poiketen vesiputousmallisesta työskentelystä, tekijätiimi on vastuussa scrumin sprintin aikana toteutuneista tuotoksista. Tuotteen omistaja sekä scrummaster eivät varsinaisesti ole esimiehiä projektinhallinnassa. Scrummasterin rooli on olla ohjaajana sekä mahdollistajana projektissa työskenteleville henkilöille.

Tuotteen omistajan rooli on kerätä asiakkaan mielipiteitä ja ohjata projektin etenemistä asiakkaan ja resurssien mukaisesti. (Certified Scrum Master 2016, 15.) Peliprojekteissa tuotteen omistaja saa koottua dataa käyttämällä palveluja, joiden avulla käyttäjät voivat lähettää suoraan palautetta tekijöille. Koska tuotteen omistajalla ei ole yhtä varsinaista asiakasta kuluttajille suunnattujen peliprojektien saralla, tuotteen omistajan tulee saada helposti koottua käsitystä asiakkaiden mielipiteistä nykyisten julkaisujen osalta. Esimerkkinä tämänlaisesta koontipalvelusta toimii Atlassian JIRA-palvelun ”Issue Collector”, joka esiteltiin luvussa kolme.

Scrum pohjautuu iteratiiviseen malliin, jossa jokaista vaihetta kutsutaan sprintiksi. Jokainen sprint jakautuu eri vaiheisiin riippuen sprintin tilanteesta. Jokainen sprint alkaa sprintin suunnittelupalaverilla, jossa kehittäjät yhdessä tuotteen omistajan kanssa määrittävät alkavan sprintin tehtävät. Sprintin aikana järjestetään päiväpalavereita, jossa kehittäjät keskustelevat edellisen päivän tuotoksista. Päiväpalaverissa kehittäjät keskustelevat myös tulevista tehtävistä, ongelmista ja mahdollisista riskeistä kehitystyöhön liittyen. Sprint päättyy sprintin katselmointiin, jossa esitellään tuotetut tehtävät tuotteen omistajalle, asiakkaille sekä mahdollisille sidosryhmille. Sprintin katselmoinnin jälkeen kehittäjät keskustelevat retrospektiivissä, mitä kehittäjät tekivät hyvin ja missä on parannettavaa seuraavaan kehitysvaiheeseen.

Mitä on peliteollisuus?

Raja peliteollisuuden sekä ohjelmistoteollisuuden välillä on työkuvan erilaisuudessa. Zackariasson ja Wilson (2012, 18) määrittävät pelinkehityksen olevan luovaa yhdessä työskentelyä sisältäen tietyn organisaation toimenpiteitä, saaden aikaan luovia ja kulttuurisesti tärkeitä töitä. Ohjelmistojen ollessa helposti lähestyttäviä koodauksen puolelta, peliteollisuuden kehitystyössä pelkkä koodin tuottaminen ei riitä. Peliliiketoimintaa harjoittava yritys tarvitsee monen eri alan osaajan samanaikaista yhteistyötä. Toimeksiantajan mukaan peliteollisuus on ohjelmistokehitystä, mutta on samalla uniikki ohjelmistoteollisuuden ala.

5 Jatkuvan integroinnin käyttöönotto

Toteutuksessa otettiin huomioon itsenäiset peliyritykset ja toimeksiantajan esittämät toiveet. Toimeksiantajan mukaan itsenäisten peliyritysten liikevaihto ja tulos voivat

olla epävakaalla pohjalla. Tämän takia jatkuvan integroinnin aiheuttamien muutoksien käyttöönotto voi olla haastavaa epävarmuustekijöiden takia. Epävarmuustekijöitä ovat muun muassa taloudelliset sekä organisaation epävarmuustekijät. Tutkimalla jatkuvaa integrointia toimeksiantajan toiveiden mukaisesti voitiin tutkia, onko palvelun pystyttämisestä taloudellisia riskejä. Palvelun pystyttämisellä saatiin menetelmätriangulaatiosta tutkimusta ilmiötä kohtaan. Menetelmätriangulaation tarkoituksena oli tutkia kokonaisvaltaista hyötyä, joten oli tärkeää tutkia miten palvelun pystyttäminen voi vaikuttaa organisaation tehokkuuteen. Koska jatkuva integroinnin käyttöönottoon kuluu aikaa, on tärkeää verrata palvelusta saatuja hyötyjä palvelun pystyttämiseen käytettyihin resursseihin. Näin opinnäytetyön validiteetti kasvaa, kun tutkittaessa tehokkuuden lisäämistä voidaan tutkia ilmiötä kokonaisvaltaisesti.

5.1 Alkutilanteen mittaus

Alkutilanteen mittaus järjestettiin toimeksiantajan kannettavalla tietokoneella. Mittaus suoritettiin yhteensä neljä kertaa. Ensimmäisellä mittauskerralla tutustuttiin tarkasti prosessiin, ja muilla mittauskerroilla mitattiin tarkasti kulutettua aikaa. Koska tässä vaiheessa julkaisu tapahtui manuaalisten prosessien kautta, henkilökohtainen ja yhteensä käytetty aika ovat kaikissa tapauksissa lähes identtiset. Henkilökohtaisella ajalla tarkoitetaan aikaa, jonka aikana käyttäjä voi käyttää työpistettään heikosti tai ei ollenkaan. Käytetty aika yhteensä tarkoittaa kuinka paljon mittauksen aloituksen ja hyväksymisen välillä on kulunut aikaa. Ajat ovat pyöristettyinä alkavalle minuutille.

Taulukko 1. Lähtötilanteen mittauksessa mitatut ajat.

Mittaus #	Käytetty aika henkilökohtaisesti	Käytetty aika yhteensä
1	29 minuuttia	30 minuuttia
2	28 minuuttia	29 minuuttia
3	24 minuuttia	24 minuuttia
4	24 minuuttia	25 minuuttia

Havaitut prosessit mittauksessa

Mittauksen aikana pyrittiin myös löytämään eri prosesseja, jotka vievät aikaa julkaisusta. Prosesseista pyrittiin havainnoimaan manuaalisia työvaiheita, joiden takia työn tehokkuus voi laskea. Prosessit noudattivat toimeksiantajan etukäteen ilmoittamia nykyisiä prosesseja. Havainnoidut prosessit olivat

1. tuotteen kääntäminen Windows-ympäristöön
2. tuotteen kääntäminen macOS-ympäristöön
3. tuotteen kääntäminen Linux-ympäristöön
4. tuotteen tarvittavien osasten kääntäminen jokaiselle yllä olevalle ympäristölle
5. tuotteen kansiorakenteen validointi
6. tuotteen integraation testaus
7. tuotteen hyväksymistestaus macOS-ympäristössä
8. tuotteen lähettäminen Steam-palveluun käyttäen Steam-palvelun työkaluja
9. tuotteen uuden version lisääminen beta-kehityshaaraan Steam-palvelussa
10. tuotteen uuden version lisääminen projektinhallintajärjestelmään, joka on laskentataulukko.

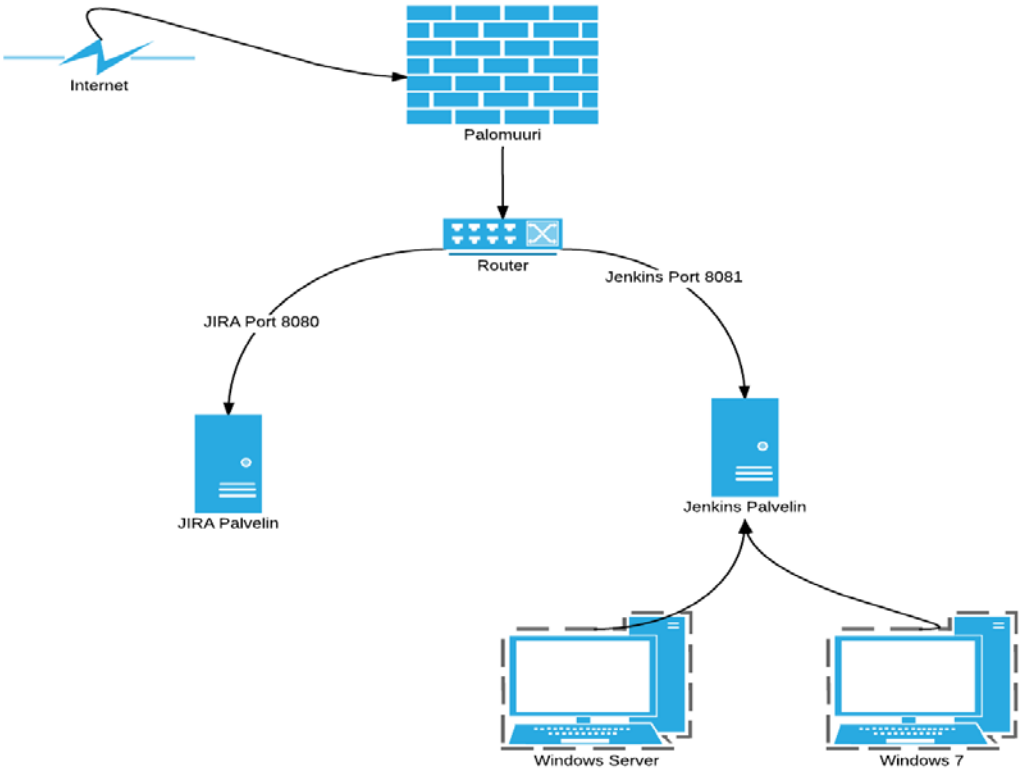
Prosesseista huomattiin tuotteen kääntämisen ja hyväksymistestauksen kuluttavan kehittäjän aikaa paljon. Tuotteen kääntämiseen kului jokaisella mittauskerralla noin seitsemän minuuttia yhtä käyttöjärjestelmäympäristöä kohti. Tämän jälkeen kehittäjän tuli vielä kääntää tarvittavat osat jokaiselle käyttöjärjestelmälle erikseen. Tämä lisäsi noin viisi minuuttia lisäaikaa jokaiseen käännökseen. Toimeksiantajalla oli valmis ohjelmakoodi käännöksien tekemiseen jokaiselle alustalle automaattisesti. Ohjelmakoodia käytettäessä kehittäjältä ei kulunut aikaa jokaisen käännöksen manuaaliseen käynnistykseen. Ohjelmakoodia hyödynnettiin jatkuvan integroinnin käyttöönotossa. Tuotteen version jakelukanavaan lähettämisessä kului aikaa noin kaksi minuuttia. laskentataulukon päivitys vei aikaa noin minuutin. Alkumittauksen jälkeen voitiin lähteä triangulaation mukaisesti hakemaan tutkimustuloksia palvelun pystytyksestä.

5.2 Jatkuvan integroinnin infrastruktuuri

Jatkuvan integroinnin infrastruktuuri rakennettiin toimeksiantajan toiveiden mukaisesti, kuten kuviossa viisi on esitelty. Rakennettu infrastruktuuri voi olla myös monelle pienelle sovelluskehitysyritykselle toimiva ratkaisu, koska infrastruktuuri on helppo rakentaa ja ylläpitää. Infrastruktuuria voidaan myös päivittää tarpeen tullen uusilla fyysisillä palvelimilla, kun organisaatioiden tarpeet muuttuvat ajan myötä.

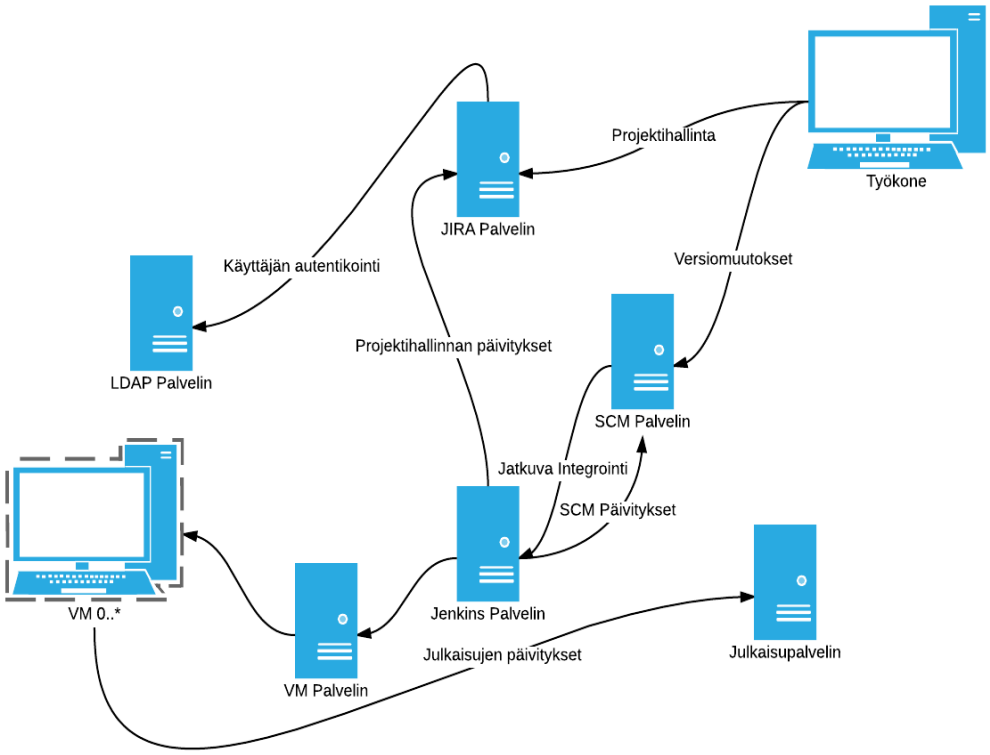
Tuotekehitykseen valittu Unity3D-pelimoottori asetti haasteita infrastruktuurin kehittämiselle. Unity3D-ohjelmaa ei ole tätä kirjoittaessa saatavilla Linux-käyttöjärjestelmille virallisesti, joten infrastruktuurin pitää tukea joko macOS- tai Windows-pohjaisia ratkaisuja. Koska toimeksiantajan haastatteluissa tuli ilmi toive Linux-palvelimen käytöstä omassa organisaatiossa, virtuaalisoitu palvelinympäristö ratkaisi haasteen.

Tämän opinnäytetyön aikana luotu jatkuvan integroinnin infrastruktuuri käsittää Windows-pohjaisen ratkaisun virtuaalisoidussa ympäristössä. Palvelininfrastruktuuria pystytettiin lähiverkkoon, jolloin palveluihin pääsee käsiksi vain toimeksiantajan tiloista. Tämä oli toimeksiantajan toiveiden mukainen malli. Malli mahdollistaa itsenäiselle peliyritykselle tietoturvallisen ratkaisun sekä skaalautuvuuden. Valittu infrastruktuuri poistaa mahdollisuuden etäkäyttöön, mutta etäkäyttö on huomioitu tämän opinnäytetyön tutkimuksessa. Malli on skaalautuva jatkokehitystä ajatellen.



Kuvio 5. Kehitysympäristön infrastruktuuri

Skaalautuvassa mallissa on mahdollista myös ottaa käyttöön LDAP-palvelin. Ottamalla käyttöön LDAP-järjestelmän yritys pystyy myöhemmässä vaiheessa kasvattamaan käyttäjämääräänsä helpommin. Koska jokainen tässä opinnäytetyössä esitelty palvelu pystyy keskustelemaan LDAP-palvelimen kanssa, yritys pystyy käyttämään eri palveluissa keskitetysti samoja tunnuksia. Tästä on hyötyä yrityksen kasvun aikana, koska yrityksen ei tarvitse päivittää tunnuksia ja käyttäjäryhmiä jokaiseen palveluun erikseen. Kuviossa kuusi esitellään esimerkki, miten jatkuvan integroinnin palveluinfrastruktuuri voidaan ottaa käyttöön pienissä ja keskisuurissa yrityksissä. Skaalautuvassa mallissa voidaan toteuttaa myös etähallintaa, jolloin työskentely julkaisun parissa ei ole enää sidottu fyysisesti lähiverkkoon. Kun projektin hallitsija ei ole enää sidottu peliyrityksen lähiverkkoon kuten kuviossa viisi, projektin hallitsijat voivat saada tietoa julkaisuista esimerkiksi asiakastapaamisissa.



Kuvio 6. Jatkuvan integroinnin palvelinarkkitehtuuri isommalle organisaatiolle

Jatkuvan integroinnin palvelun valinta

Opinnäytetyössä päädyttiin esitutkimuksen jälkeen käyttämään Jenkins-palvelua, koska Jenkins tarjoaa vakaan yhteisön, joka päivittää palvelua. Jenkinsiä voidaan pitää myös vakiintuneena standardina jatkuvan integroinnin palveluista. Palvelu oli lähimpänä teorian sekä haastattelujen aikana kerätystä toiveista toimeksiantajan suunnalta. Vapaan lähdekoodin Ansible oli vaihtoehtona, mutta valinta kallistui Jenkins-ympäristöön. Koska Jenkins-projektilla on vakiintunut kehittäjäyhteisö (Berg 2012, 1), toimeksiantajan on helpompi ylläpitää valittua järjestelmää. Tämä vähentää kuolleiden palveluiden riskiä toimeksiantajan jatkon kannalta, koska Jenkins-palvelun voi-

daan olettaa toimivan elinvoimaisen yhteisön takia. Kuollut palvelu tarkoittaa palvelua, jolle ei voi enää saada tukea tai päivityksiä (Miller, Vandome & McBrewster 2009, 1).

Projektinhallinta

Projektinhallinta peliteollisuudessa on ensisijaisen tärkeää. Siksi on tärkeää valita laadukas ohjelmistopaketti jatkuvan integroinnin palvelun tueksi. Monet palvelut tarjoavat nykyään laadukasta projektinhallintaa, mutta valinta päätyi JIRA-palveluun esitutkimuksen seurauksena. Valitsemalla JIRA voitiin luoda ammattimainen palvelu, joka oli lähimpänä toimeksiantajan toiveita projektihallinnan palvelusta.

5.3 Asennus

Palvelimien alustaksi valittiin Ubuntu Server 10.15-versio. Vaikka Ubuntu Server on laajalti käytössä, kannattaa yritysten harkita Red Hat Enterprise -version käyttämistä. Red Hat Enterprise on erikoistunut alusta alkaen palvelinympäristöjen ylläpitämiseen. Jos yritys miettii Ansible-ohjelman käyttämistä jatkuvan integroinnin palveluna, yrityksen tulisi harkita Red Hat Enterprise Edition -version käyttämistä. Ubuntu Serveriin asennettiin OpenSSH-palvelu, jonka käyttäminen on suositeltavaa kaikilla palvelimilla. OpenSSH muodostaa avainarvoyhdistelmän, jonka avulla palveluun voidaan kirjautua (Features n.d.).

Asennettua palvelinympäristöt molemmille tietokoneille sekä verkon toimivuuden testauksen jälkeen, jatkettiin työtä asentamalla suljetussa ympäristössä Atlassian JIRA-ympäristön. JIRA-ympäristön toimivuuden varmistamiseksi palvelimelle tuli asentaa sekä MySQL-tietokanta että yhteysajuri. MySQL-yhteysajuri on pakollinen, jotta JIRA voi muodostaa yhteyden tietokantaan (Connecting JIRA to MySQL 2016). JIRA-palveluun asennus onnistui helposti, koska asennuksen mukana tulee valmiit asennukseen tarkoitetut ohjelmakoodit. Kun serveripuolen asennus oli valmiina, viimeisteltiin asennus selaimessa. JIRA-palvelun asennus suoritettiin perusversiolla, sisältäen ketterän kehityksen työkalut.

Jenkins-palvelulle varattiin toinen palvelin kuviossa viisi kuvatun infrastruktuurin mukaisesti. Seuraamalla Jenkinsin asennusohjeita Ubuntulle, Jenkinsin asennus oli help-

poa. Jenkinsin asennusohjeissa kerrotaan, miten Jenkinsin yhteysportti voidaan vaihtaa käyttämään muuta kuin oletusporttia. Jos Jenkins asennetaan samalle palvelimelle JIRA-palvelun kanssa, tulee Jenkinsin tai JIRA-palvelun käyttämä yhteysportti vaihtaa. Molemmat palvelut kuuntelevat normaalisti samaa yhteysporttia, mikä estää toiseen palveluun pääsyn. Kun Jenkins oli toiminnassa oikealla portilla, asennettiin Git-versiohallinta Ubuntun pakettijärjestelmästä. Ubuntun pakettijärjestelmä mahdollistaa kattavan ohjelmien asennuksiin ja ylläpitoon tarkoitetut kirjastot (Package Management, n.d.).

Seuraavaksi työvaiheena oli virtuaalikoneympäristön asentaminen Jenkins-palvelimelle. Tutkittaessa vaihtoehtoja päädyttiin asentamaan Kernel Virtual Manager -ohjelma, jonka avulla virtuaalikoneita voidaan ajaa Linux-palvelimella (What Is KVM n.d.). Palvelimien asetukset aiheuttivat ongelmia SSH-yhteyden kautta ja komentokehotteinen palvelimen konfiguraatio ei onnistunut. Tämä ratkaistiin käyttämällä graafista SSH-yhteyttä.

Käyttämällä graafista SSH-asiakasohjelmaa kuten X11, voidaan palvelinta käyttää graafisesti etäyhteyden kautta. Virtuaalikoneiden hallinta on näin helpompaa etäyhteyden kautta. Seuraavaksi työvaiheena oli käyttöjärjestelmän asentaminen käyttäen virt-manager-ohjelmaa. Windows 7-asennus onnistui, mutta järjestelmän käyttämisessä esiintyi ongelmia. Windows saattoi lopettaa vastaamisen, jolloin ainoa vaihtoehto oli pakotettu uudelleenkäynnistys. Virtuaalikoneen järjestelmän sai toimintaan ainoastaan tällä tavoin. Ongelma toistui monesti ensimmäisen tunnin aikana, joten Windows 7-asennuksen rinnalle asennettiin Windows Server 2012. Tällä virtuaalikoneella ei syntynyt samanlaista ongelmaa.

Windows Server 2012 -asennuksen jälkeen uudelle virtuaalikoneelle asennettiin tarpeelliset ohjelmat. Ohjelmat käsittivät Unity3D-ohjelman ja jatkuvan integroinnin palvelujen käyttämistä varten tarpeelliset Javan, Blender-ohjelman sekä versionhallinnan. Jenkins-palvelussa voidaan käyttää versionhallintaa käynnistämään julkaisuja sekä Mercurial-, SVN- että Git-versiohallintaa käyttäen. Windows Serverille asennettiin Git opinnäytetyötä varten, koska Git oli yhteensopiva toimeksiantajan nykyisten järjestelmien kanssa. Jenkins-virtuaalikoneella täytyy olla versionhallinta käytössä, koska Jenkins-palvelu ei itsessään lähetä projektitietoja virtuaalikoneelle.

Jatkuvan integroinnin käyttämistä varten palvelulle tulee luoda projekteja. Jenkins tarjoaa projektien luomiseen kaksi vaihtoehtoa, mutta lisäosien avulla projektien malleja voidaan muokata. Menetelmätutkimusta varten käytettiin ”Freestyle Project”-projektimuotoa, jonka avulla projektin konfigurointi opinnäytetyön testausta varten oli kattava. Muokatessa versionhallinnan asetuksia esiin nousi ongelma, jossa Jenkins ei oletettavasti saanut yhteyttä Git-palvelimeen. Ongelma oli mielenkiintoinen, koska Git-palvelin oli asennettu samalle tietokoneelle. Ongelma ei kuitenkaan toistunut ajettaessa testiajoja itse käännöspalvelimilla. Ongelma voi johtua Jenkinsin ollessa samalla tietokoneella Git-palvelun kanssa, jolloin Jenkins ei pysty muodostamaan SSH-yhteyttä itseensä. Virtuaalisten käännöskoneiden tapauksessa SSH-yhteys muodostetaan taas ulkoverkon kautta, jolloin käännöskoneilla ongelmaa ei esiinny.

Jenkins tarvitsee orjakoneyhteyden ennen Jenkins-projektin käyttöönottoa. Yhteys muodostettiin Java Web Start -palvelulla, koska tämä toimintamalli on suositeltu Jenkinsin dokumentaatiossa. Palvelun käyttäminen tarvitsee Java EE -ympäristön Windows-palvelimelle. Java EE -asennuksen jälkeen orjakone saa heti yhteyden isäntäkoneeseen. Käytettäessä yhtä konetta pitempijaksoisesti, yhteysportti kannattaa vaihtaa vakaaksi yhteysportiksi. Käytettäessä vakaata yhteysporttia, järjestelmän ylläpitäminen helpottuu. Jos Jenkins käyttää vakaata yhteysporttia orjakoneille yhteyden kanssa, palomuuria ei tarvitse määrittää kuuntelemaan useita eri portteja. Palvelimien tietoturva paranee myös käytettäessä vain yhtä avointa porttia eri käännöskoneiden yhteyttä varten.

Jenkins-palvelimelle määritettiin uusi tyhjä Git-versiosäiliö. Git-versiosäiliöön alustettiin uusi Unity3D-projekti, jonka avulla pystyttiin esitestaamaan Jenkins-palvelun toimivuutta. Tutkimuksessa testattiin Jenkins-palvelun toimivuutta uusien peliversioiden luonnissa, automaattisten Asset Bundle -pakettien luomista sekä yksikkötestien ajamista. Unity3D voidaan käynnistää ilman graafista käyttöliittymää, jolloin palvelimen laskentatehoa ei missään käännöstyön vaiheessa hukata käyttöliittymän piirtämiseen (Command line Arguments n.d.). Jenkinsille on olemassa lisäosa, jonka avulla Unity3D-ohjelman käynnistäminen on helposti muokattavissa suoraan Jenkinsistä (Lacoste 2016).

Testiprojektin käännös onnistui ongelmitta. Testiprojekti ajoi testausta varten määritellyt ohjelmakoodit lävitse. Testiprojekti päivitti pelin tiedostot WebGL-ympäristölle

ongelmitta seuraavassa vaiheessa. Tämän jälkeen testiprojekti siirsi kootun version palvelimelle. Aikaa kokonaisuudessaan projektin rakentamisen alusta lopetukseen meni alle kymmenen minuuttia.

Lopuksi luotiin versiohallinnan päivittäminen asennetulle JIRA-palvelimelle. Versioiden päivitystä varten Jenkinsille tuli asentaa lisäosa, jonka avulla pystytään lähettämään versioiden päivitystiedot suoraan JIRA-palvelun projekteihin. Lisäosa löytyi palveluja tutkittaessa. Jenkinsin lisäosaa varten tarvitsi JIRA-palveluun luoda projekti, ja määrittää JIRA-palveluun yhteysasetuksista mahdollisuus Jenkinsille ottaa yhteys JIRA-palveluun. Viimeisenä työvaiheena toteutettiin toimeksiantajan toive tutkia ja mahdollisuuksien salliessa ottaa käyttöön Git Tag-ominaisuus. Ominaisuuden avulla versiohistoriaan voidaan merkitä tärkeitä pisteitä (Git Basics - Tagging n.d.). Tag-ominaisuuden käyttöönotto voidaan toteuttaa suoraan Jenkinsin Freestyle Project -projektin kautta.

Lopputilanteen mittaus

Lopputilanteen alkutilanne mittaukselle oli sama kuin alkutilanteessa. Mittauksessa käytetyt ohjelmistot ja tietokone olivat samoja. Kehittäjälle annettiin kirjalliset ohjeet edellisenä päivänä, kuinka jatkuva integraation prosessi toimii kehitystyön jälkeen. Ensimmäinen mittauskerta toimi prosessiin tutustumisena, jotta käytetty aika saataisiin mahdollisimman vertailukelpoiseksi alkutilanteen mittauksen kanssa. Alkutilanteen tapaan käytetty aika henkilökohtaisesti tarkoittaa aikaa, jolloin työpistettä voi käyttää heikosti tai ei ollenkaan. Käytetty aika yhteensä tarkoittaa aikaa, jolloin mittauksen prosessi on hyväksytysti läpäisty.

Taulukko 2. Lopputilanteen mittauksessa mitatut suoritusajat.

Mittaus #	Käytetty aika henkilökohtaisesti	Käytetty aika yhteensä
1	6 minuuttia	18 minuuttia
2	4 minuuttia	20 minuuttia
3	5 minuuttia	18 minuuttia
4	5 minuuttia	24 minuuttia

Jatkuvan integroinnin käyttöönotto on helppoa, mutta käyttöönotossa menee aikaa. Tässä opinnäytetyössä esitellyn infrastruktuurin tekemiseen meni aikaa noin kuusi tuntia. Aika on laskettu käyttöjärjestelmien asennuksesta päättyen versioiden päivittämisen JIRA- ja Git-palveluihin.

Mittauksien analysoinnin jälkeen voidaan todeta, että toimeksiantajan prosessit kehittyivät 425 % nopeammiksi. Toimeksiantaja voi näin ollen säästää jopa 18,2 tuntia vuositasolla kehittäjien aikaa. Säästöt voivat olla satoja euroa vuositasolla toimeksiantajan strategiassa ohjeistamalla 2700 euron keskipalkalla (Strategy 2015–2016 2015). Laskennassa on oletettu yhden kehittäjän tekevän yhden käännöksen viikoittain. Mittauksien sekä palvelun pystyttämisen avulla voitiin tutkia miten jatkuva integrointi voi tehostaa peliprojektityöskentelyä tuotannon eri vaiheissa ottamatta kantaa ylläpitovaiheeseen.

Saadut hyödyt kompensoivat tehokkaasti käytettyä aikaa järjestelmän pystyttämiseen. Useammalla henkilöllä ja nopeammalla julkaisutahdilla laskettuna aikaa säästyy vielä enemmän. Tämän takia jatkuva integrointi tehostaa julkaisua ja projektinhallintaa pitkällä aikavälillä tämän opinnäytetyön rajauksen puitteissa. Kritiikkiä mittauksiin ja ongelmatilanteita tähän havainnointiin löytyy pohdintakappaleesta.

Tehostaminen ulottuu scrumin luonteeseen monella eri tasolla. Jatkuvaa integrointi voidaan sulauttaa luvussa kolme esiteltyjen scrumin vaiheisiin. Kehittäjät voivat aina saada päiväpalavereissa tiedot uusista versioista ja testauksista, joiden avulla kehittäjät voivat keskustella paremmin, mitä on tehty. Päiväpalavereihin saadaan tietoa, onko julkaisussa tällä hetkellä ongelmia esimerkiksi testaukseen liittyen. Suunnittelupalavereissa ja katselmoinneissa voidaan heijastaa aina sen hetkisen tuotteen toiminallisuutta ja ominaisuuksia tulevan suunnitteluun. Jatkuva integrointi vaikuttaa kokonaisvaltaisesti itsenäisen peliyrityksen ketterään projektinhallintaan.

6 Johtopäätökset

Suurin hyöty jatkuvan integroinnin ottamisesta käyttöön on versiojulkaisujen eri vaiheiden virtaviivaistaminen ja nopeuttaminen. Weiss (2013) toteaa toistettavien manuaalisten töiden olevan hitaita ja alttiita inhimillisille virheille. Tämän tutkimuksen aikana on huomattavissa tehostamisen vaikutus, kun manuaalisista vaiheista päästiin

eroon. Jatkuvan integroinnin käyttöönoton avulla manuaaliset työvaiheet tehostuivat moninkertaisesti. Järjestetyt mittaukset auttoivat havainnoimaan suoria sekä epäsuoria tehostamisen vaikutuksia julkaisuun ja hallintaan. Suorat tehostamiset ovat nähtävissä numeroista, joista kävi ilmi jatkuvan integroinnin parantavan 425 % työvaiheiden tehoa. Epäsuoriin vaikutuksiin voidaan lukea jälkitilanteen mittauksessa havaittu työvaiheiden muistamisen väheneminen. Kun samat työvaiheet vaativat vähemmän, kului tekijältä vähemmän aikaa miettimiseen. Tämä voi osittain laskea stressiä kovan paineen alla. Kovan paineen alla tekijän tai hallinnoijan ei tarvitse keskittyä jokaiseen yksityiskohtaan. Automaatio hoitaa yksityiskohdat.

Jatkuvan integroinnin nopeuttavat vaikutukset ovat nousseet ennenkin esiin muiden tutkimuksien perusteella. Lukkarinen (2011, 141) kertoo opinnäytetyössään, kuinka opinnäytetyön aikana kehitetty ympäristö oli mahdollistanut parantaa kehitysprosessien tehokkuutta, ajantasaisuutta ja todenmukaisuutta huomattavasti. Lukkarisen opinnäytetyö keskittyy ohjelmistotekniikan opiskelijoiden koulutukseen (mts. 1), mutta vaikutukset kehitysprosessien tehokkuuteen vaikuttavat projektinhallinnan tehokkuuteen. Kun kehittäjät tekevät työnsä kurinalaisemmin ja tuottavat laadukkaampaa ohjelmistoa, projektinhallitsijoiden tarvitsee osallistua vähemmän tekemisen valvontaan. Tässä opinnäytetyössä ei keskitytty Lukkarisen tutkimaan testauksen osa-alueeseen opinnäytetyön rajauksen vuoksi.

Toinen iso hyöty jatkuvan integroinnin käyttöönotosta oli tämän opinnäytetyön alussa esitetty olettamus projektijohtamisen tehostamisesta. JIRA-palvelun käytössä oli huomattavissa organisaation sisäisen viestinnän nopeutuminen. Nopeutuminen tehostaa projektijohtoa, koska johtajat voivat reagoida ongelmiin nopeammin. Projektijohtajan ei tarvitse esimerkiksi kysyä tietoa projektin julkaisun onnistumisesta tekijöiltä. Kun johtaja voi tarkistaa julkaisun tilanteen Jenkinsistä, projektijohtaja ei ole enää edes fyysisesti sidottu samaan paikkaan tekijöiden kanssa. Tämä helpottaa esimerkiksi aikataulutusta, kun esimerkiksi palaverissa on mahdollista tarkistaa julkaisun eteneminen. Tämä vaatii luottamista toimivaan laadunhallintaan.

Laadunhallinnallista etua jatkuvan integroinnin käyttö peliprojekteissa voi luoda kehittäjien tekemien testien avulla. Kun esimerkiksi yksikkötestit ajetaan automaattisesti, inhimilliset virheet testien ajon aikana poistuvat (Fewster & Graham 1999, 346-347). Projektinhallinta tehostuu, kun läpinäkyvyys tuotetussa ohjelmassa lisääntyy

jatkuvan integraation tuottamien raporttien kautta. Reagoiminen ajoissa tarkoittaa stabiilimpaa toimintaympäristöä koko organisaatioon. Tästä seuraa koko organisaation mittainen tehostuminen, kun reagointi tapahtuu vähemmän kaoottisesti.

Jatkuva kehitys osana pelikehitystä

Peliprojektit voivat olla suuria ja jatkuvan integroinnin edut nousevat huomattavaan arvoon peliprojekteissa. Koska projektien koot voivat olla useita kymmeniä tuhansia rivejä ohjelmistokoodia, pelkästään ohjelmakoodin manuaalinen testaaminen olisi hyvin vaivalloista. Yksikään ihminen ei pystyisi tehokkaasti jokaisella kerralla suorittamaan testejä, mutta jatkuvan integroinnin palvelujen avulla testit tulisi aina suoritettua. Tämä poistaa inhimillisiä epävarmuustekijöitä ohjelmistojen koonnissa ja käyttöönotossa. (Lukkarinen 2011, 38.)

Pelien päivittäminen voi olla raskasta jakelualustoille, kun kyseessä on useiden satojen tai tuhansien megatavujen kokoisista peleistä. Pelien kääntäminen voi lähtökohtaisesti kestää useita kymmeniä minuutteja. Version päivittämisen jälkeen peliversio tulisi vielä manuaalisesti päivittää julkaisualustalle. Tähän päivitykseen voi mennä myös useita minuutteja. Yhden version julkaisuun voi kestää tunteja. Tänä aikana tehokas työskentely voi rajoittua tai estyä kokonaan työpisteellä.

Jatkuvan integroinnin käyttö peliversioiden päivittämisessä on suositeltavaa yllä mainitun ajan säästön lisäksi myös peliprojektien asiakashallinnan takia. Jos peli tarvitsee versiotietoja esimerkiksi moninpeliä varten, pelin tulee tietää versionumeronsa toimintojen takia. Pelin versiotiedon tietämisen avulla pystytään esimerkiksi antamaan pelaajalle viestejä mahdollisista päivityksistä sekä estämään moninpelissä kahden eri version yhteydenmuodostukset. Peliversion päivittäminen pelille itsessään onnistuu helposti Unity3D-ohjelmakoodilla, joka suoritetaan ennen varsinaisen peliversion tekemistä. Ohjelmakoodi hakee Jenkinsiltä tiedon, mikä versio on lähdössä rakentumaan ja päivittää tiedon pelin sisäiseen data-kansioon tai kooditiedostoon. Tämän jälkeen peli pystyy version päivittämisen jälkeen hakemaan Jenkinsin päivitetyn tiedon omista sisäisistä tiedoistaan ja käyttämään tätä hyväksi.

Indie-peliyritysten projektijohtamisen tehostaminen

Koska prosessit ovat automaattisia tuotteiden koonnissa ja julkaisussa, projektijohtajien epävarmuustekijät vähenevät. Indie-peliyritysten tapauksessa automaatio voi tarkoittaa esimerkiksi tiedon siirron välitystä tehokkaammin, kuten pystytetyssä palvelussa projektiversioiden päivitykset projektinhallintaan tehostuvat. Tiedon välityksen avulla projektin hallinnoijat voivat käynnistää seuraavat vaiheet tehokkaammin, kun tietoja uusista testatuista ja hyväksytyistä versioista saadaan parhaimmillaan minuuteissa. Koska projektin hallinnoijat voivat olla myös fyysisesti pois paikalta, ei aikaa kulu kommunikaatiovälineiden käyttämiseen.

Kun epävarmuustekijät vähenevät, projektijohtaja voi keskittyä muuhun kuin ongelmanratkaisuun. Tämä auttaa itsenäisten peliyritysten projektihallinnan tehokkuutta, kun hallinnoijat voivat keskittyä esimerkiksi asiakkaidenhallintaan. Kun projektin hallinnoijat voivat keskittyä enemmän asiakkaisiin ja heidän tarpeisiinsa, luottamus peliprojektien asiakkaiden ja tekijöiden välillä kasvaa. Tästä voi seurata lyhyempi sykli asiakkaan lähettämässä palautteessa, joka heijastuu ketteriin projektinhallintamenetelmiin. Saamalla lisää palautetta tuotteen omistajan käyttöön sprintin suunnittelupalaveriin ja kehitysjonon työstöpalaveriin, voi projektin suuntaa muuttaa tehokkaammin asiakkaiden toiveiden mukaisesti. Kun tuotteen omistaja saa tietoa asiakkaiden toiveista, voi jokainen luvussa neljä kuvattu scrumin prosessi tehostua.

Indie-peliprojektien versiojulkaisujen tehostaminen

Kuten luvussa kolme kuvattiin, tässä opinnäytetyössä jatkuvaa integrointia käsiteltiin myös jatkuvana käyttöönottona. Jatkuvan integroinnin käyttäminen versiojulkaisuiden tekemisessä voi hyödyntää projekteja monella eri tasolla. Koonnin ja käyttöönoton avulla jo pelkästään ajalliset hyödyt käytetyssä työajassa ovat merkittäviä kuten luvussa viisi on kuvattuna. Kun aikaa kuluu vähemmän odottamiseen, voi koko organisaatio keskittyä tärkeämpiin asioihin.

Versiojulkaisujen koonti on aina varmempaa ja tehokkaampaa. Kerran tehdyt koontikoodit tekevät aina versiojulkaisut tietyssä ja samassa järjestyksessä. Kun koonti ja julkaisu tehdään aina samassa järjestyksessä, kehittäjien ei tarvitse käyttää ongelman etsimiseen ja ratkaisuun yhtä paljon aikaa. Tämä on tärkeää erityisesti nopean syklin

päivityksissä, joissa päivitysten välillä kehittäjillä ei välttämättä ole aina aikaa tehdä manuaalisesti tarvittavia toimenpiteitä ennen julkaisua.

Kuten Lukkarinen (2011, 20) kuvaa, koontijärjestelmille voidaan siirtää monia prosesseja suoritettavaksi automaattisesti. Kuten tämä opinnäytetyö, myös Lukkarisen opinnäytetyö keskittyy jatkuvan integroinnin käyttämiseen kehitystyön yhteydessä. Jatkuvan integroinnin edut itsenäisten peliyritysten versiojulkaisujen ja projektin hallinnan tehostamisessa eivät välttämättä rajoitu tämän opinnäytetyön tuloksiin.

7 Pohdinta

Tämän opinnäytetyön tarkoituksena oli tutkia, miten jatkuvaa integrointia voidaan käyttää tehostamaan peliprojektien hallintaa ja julkaisuversioiden päivittämistä. Toimeksiantajana toiminut Kinahmi Games OY on tätä lukua kirjoittaessa ottanut opinnäytetyössä tuotetut tulokset käyttöön, ja on tyytyväinen jatkuvan integroinnin käyttöönottoon. Toimeksiantajan mukaan tutkimus toimii hyvänä pohjana organisaation nykyiseen tilanteeseen, ja tutkimus avaa mahdollisuuden jatkotutkimuksen työstämiseen.

Lähtökohdat

Toimeksiantajan esittämä tutkimusongelma oli laaja. Valituilla tutkimuskysymyksillä pystyttiin rajaamaan mahdollisimman tehokkaasti käsiteltävät asiat mahdollisimman lähelle toimeksiantajan toiveita. Tutkimusongelma toi uutuusarvoa, koska tutkimuksen aikana kävi ilmi edellisten tutkimusten vähyys johtamisen ja peliprojektien puolelta.

Kuten tutkimuksen luvussa kaksi on todettu, teoriaa odotettiin löytyvän vähän peliprojektien osalta. Tämän vuoksi teorian sovittaminen opinnäytetyöhön tapahtui ohjelmistotuotannon tutkimuksia käyttäen. Tämä ei ollut aina hyvä asia, mutta teorian sovittaminen onnistui aihealueeseen. Jatkotutkimuksena olisikin hyvä toteuttaa hypoteesiin pohjautuvaa tutkimusta, mutta tällainen tutkiminen on järkevä suorittaa väitöskirjoissa.

Kerätty teoria antoi hyvät lähtökohdat opinnäytetyön varsinaiselle tutkimusosalle. Kuten luvussa kaksi on mainittu, teoriaosio sovelsi vain olemassa olevaa tietoa. Tämän työn kannalta rajaus oli hyvä, mutta jatkokehitykselle jäi varaa. Esimerkiksi miten johtamisen eri teoriat vaikuttaisivat hyötyihin? Miten jatkuva integrointi tehostaa tilanteessa, jossa johtaja on tyylillisesti asiakeskeinen ja toimii autoritaarisesti? Scrumissa johtaja mielletään aina ihmiskeskeisenä ja demokraattisena.

Teorian avulla pysyttiin määrittelemään, mitä jatkuva integraatio on yksinkertaisimmillaan. Määrittelyssä apuna toimi nykyisten palvelujen tutkiminen, ja mitä ominaisuuksia nykyiset palvelut tarjoavat. Palveluiden tutkiminen mahdollisti myös muiden tutkimuskysymysten vastausten löytämisen. Palveluita tutkimalla ja haastatteluja tekemällä iteratiivisesti saatiin näkemys jatkuvan integroinnin mahdollisuuksiin julkaisussa ja julkaisujen hallinnassa. Yhdistettynä tutkimukseen ketteristä projektinhallintamenetelmistä ja projekteista yleisesti, teoriaosion saavutukset ovat hyvät.

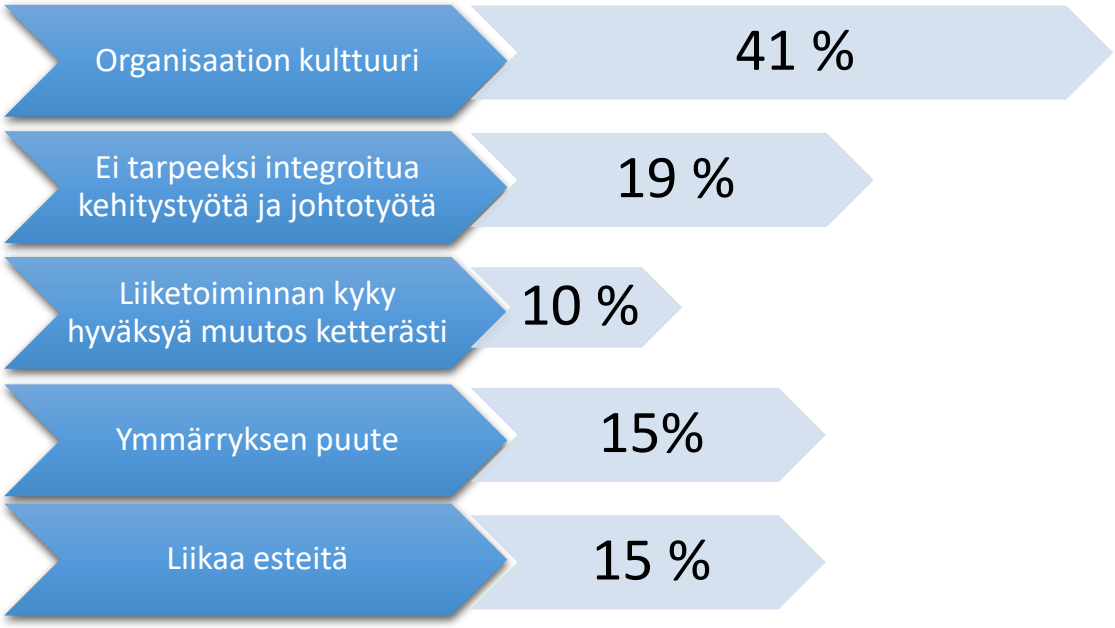
Teoriaosio ei kuitenkaan osoittautunut muuttumattomaksi. Tätä kirjoittaessa moni palveluista on alkanut tarjota uusia ominaisuuksia. Osa palveluista saatetaan myös korvata uusilla palveluilla kehittäjien toimesta. Esimerkiksi Atlassianin Bitbucket Git -versiosäiliöpalvelu on alkanut tarjota jatkuvan integroinnin ominaisuuksia. Tämän vuoksi teoriaosion yleistettävyys ja muuttumattomuus ei ole hyvällä tasolla. Jatkotutkimuksissa palveluiden tutkimiseen ei kannata käyttää aikaa, vaan korvata käytetty aika esimerkiksi haastatteluilla tai syventymällä projektinhallintaan. Projektinhallinnallisten kappaleiden uskon pysyvän stabiilina.

Palvelun pystytys

Kuten kappaleessa kaksi on todettu, palvelun pystyttäminen ei ole kvalitatiivisen tutkimuksen ominaispiirre. Tämän vuoksi palvelun pystyttämisen kuvaamista ei suositella ylemmän tason tutkimuksille. Palvelun pystyttäminen on enemmän monistrategista tutkimusta, mutta pystytyksellä saatiin käytännönläheistä uutta tutkimustietoa.

Uusi tutkimustieto perustuu kokonaisvaltaiseen ajankäyttöön, joka on merkittävä asia jatkuvaa integrointia miettiville organisaatioille. Saamalla tietoa palvelun pystyttämisestä, voidaan hälventää epäluuloja palvelun tehostamisesta organisaation sisällä. Kuten kuviossa seitsemän on näytetty, tämä on yleinen ongelma. Näyttämällä

toteen, mitä työvaiheita palvelun pystyttämiseen liittyy, pystyttiin todentamaan tehostamisvaikutukset epäileville organisaatioille. Palvelun pystytys toi uutuusarvoa uuden tiedon myötä, kun palvelun pystyttäminen oli suunnattu itsenäisille peliyrityksille. Palvelun pystyttämisen kuvatut työvaiheet ovat hyödynnettävissä itsenäisiin peliyrityksiin, vaikka palvelut muuttuisivatkin. Jokainen olemassa oleva palvelu perustuu samanlaiseen arkkitehtuuriin.



Kuvio 7. Jatkuvan integroinnin käyttöönoton haasteet organisaatioissa (Continuous Delivery Adaption Barriers 2013).

Mittaus

Mittaus oli ensisijaisesti havainnoinnin apuväline tutkimuksessa. Mittaus on kvantitatiivisen tutkimuksen piirteitä. Kvantitatiivinen tutkimus pyrkii tutkimaan tapausten joukkoa (Kananen 2014a, 19), kuten edellä mainittua mittauksia. Tämä ei kuitenkaan ole luotettavuusongelma, koska molemmissa mittauksissa suoritettiin sekä mittaus

että havainnointi. Käyttämällä tarkkaa ajallista dataa vaikutuksista, pystyttiin tekemään parempia johtopäätöksiä alkuperäiseen tutkimusongelmaan. Koska toimeksiantajan mukaan aikaa kului eniten tuotteen julkaistun version päivittämiseen jakelualustaan, oli myös järkevintä keskittyä tähän tiettyyn työvaiheeseen.

Olivatko mittauksen vaiheet tarpeeksi tarkkaan määritelty kvantitatiivisen sisältövaliditeetin toteutumiseen? Opinnäytetyön rajoissa kyllä, mutta aukkoja on, kuten edellä on mainittu. Koska mittareiden tavoite on mitata juuri oikeaa asiaa (Kananen 2014, 149), tässä opinnäytetyössä tutkimuksen sisältövaliditeetti toteutui. Jatkokehitettävää on, koska edellä mainitut ongelmat kyseenalaistavat saadut tulokset.

Mittaus ei ollut kuitenkaan täysin aukoton, koska mittauksessa mitattiin prosessi työn alkamisesta. Käytetty aika ei välttämättä kuitenkaan jää tähän. Mitä toimenpiteitä vaaditaan työntekijän saamiseksi samaan tilaan? Tämä riippuu toimeksiantajan mukaan tilanteesta. Mitä toimenpiteitä tulee suorittaa, kun versio on tiedossa projektinhallinnassa? Tarvitseeko esimerkiksi asiakkaille saattaa tieto uudesta versiosta? Näihin kysymyksiin tulisi keskittyä myös jatkotutkimuksissa. Jatkuvan integroinnin tehostamisen vaikutukset projektin hallintaan ja tuoteversioiden julkaisuun eivät välttämättä ole vain tulokset, jotka tässä opinnäytetyössä on esitelty.

Mittauksen tulokset ovat hyödynnettävissä organisaatioissa, jotka miettivät jatkuvan integroinnin käyttöönottoa. Tuloksissa mittaukset suoritettiin laajalla ohjelmistolla, jossa oli yli seitsemänkymmentä tuhatta riviä ohjelmakoodia. Ohjelmakoodin ja muiden työvaiheiden tehostamisvaikutukset jatkuvan integroinnin koonnissa ovat moninkertaiset. Vaikutukset voivat kuitenkin vaihdella riippuen ohjelmasta, mutta myös käytetystä palvelininfrastruktuurista. Tämän vuoksi mittauksen ajat eivät ole välttämättä toisessa projektissa samat, mutta tehostamisen vaikutukset tulevat näkymään silti. Jos ajallinen mittaus olisikin eri, tehostaa jatkuvan integroinnin käyttö poistamalla inhimilliset virheet tuotteiden julkaisusta. Kun tiedetään että julkaisu toimii aina samalla tavalla, tarve manuaaliselle testaukselle voi vähentyä.

Johtopäätökset

Johtopäätöksillä pystyttiin vastaamaan tutkimusongelmaan, mutta jatkotutkimusta aihealueesta tarvitaan. Tulokset kertovat, että jatkuva integrointi tehostaa pelkästään ajallisesti moninkertaisesti tuoteversioiden julkaisua. Lisäksi jatkuva integraatio

vähentää hitaita ja manuaalisia prosesseja, mikä lisää varmuutta prosesseihin ja tekijöihin. Kun ihmiset eivät pääse enää tekemään virheitä tai unohtamaan työvaiheita julkaisussa, voi organisaatio olettaa parempaa laatua. Tämä voi muodostaa positiivisen ilmiön työskentelyn tehokkuudessa, kun laadunvarmistukseen tarvitsee käyttää vähemmän aikaa.

Ovatko tulokset yksiselitteisiä? Eivät. Edellä mainitun mittauksen problematiikan kanssa, myös resurssien vähyys voi vaikuttaa negatiivisesti tuloksien pysyvyyteen. Resurssien puutteen takia ei pystytty tutkimaan pitkän aikavälin hyötyjä, mikä olisi myös tärkeää mietittäessä tehokkuuden lisäämistä. Muodostuuko jatkuvan integroinnin ylläpito ongelmalliseksi jossakin vaiheessa? Tuleeko palveluissa ongelmia, kun käytetyt työkalut päivittyvät ja muuttuvat? Miten jatkuva integroinnin käyttö projektinhallinnassa tulee muuttamaan projektin hallitsijoiden työskentelyä? Tuleeko tuotteen omistajille liikaa tietoa, kun versioita päivitetään useita kertoja päivätasolla? Näitä kysymyksiä tulisi tutkia lisää, jotta voidaan antaa aukoton vastaus tutkimusongelmaan. Jatkuva integrointi ei välttämättä lisääkään tehokkuutta julkaisussa ja projektien hallinnassa pitkällä aikavälillä.

Triangulaation ottaminen tutkimukseen osoittautui hyväksi valinnaksi tutkimuksellista kasvamista ja opinnäytetyön tuloksia ajatellen. Ottamalla triangulaation piirteitä käyttöön, tiettyihin aineistoihin keskittyminen helpottui. Tämä auttoi erityisesti teoriaosion ja haastattelujen tekemistä opinnäytetyön aikana.

Soveltuivatko triangulaation valinnat hyvin? Oliko tutkimus ristiriidaton tulkinta ilmiöstä? Aineiston puolelta kyllä, mutta aineiston määrää olisi ollut hyvä kasvattaa. Tämä olisi taas vaatinut useampaa organisaatiota, joka oli resurssien puitteessa vaikeaa. Menetelmä- ja teoriatriangulaatio soveltuivat hyvin ongelman ratkaisemiseen, mutta triangulaation osissa oli edellä mainittuja ongelmia. Puusniekka & Saaranen-Kauppinen (2006) toteavat, että oikeita vastauksia kvalitatiivisen tutkimuksen valintoihin ei voida löytää, mutta apuvälineitä on. Työkaluina toimivat tämän opinnäytetyön aikana vapaamuotoiset kyselyt asiantuntijoilta, sekä oma kyseenalaistaminen käytettyjä tapoja kohtaan. Opinnäytetyön aikana ei ollut tarkoitus tehdä puhdasoppista triangulaatiota. Käyttämällä piirteitä triangulaatiosta pyrittiin hakemaan paremmin tietoa tutkimuksen tekoon. Jatkotutkimukset osoittavat, kuinka hyvä tämä opinnäytetyö on.

Toteutuiko aineiston saturaatio? Ei. Kuten edellä mainittiin, haastatteluita olisi pitänyt lisätä. Aineistoa olisi voinut kerätä lisää muilta tekijöiltä, mikä olisi saanut aikaan laajemman tutkimuksen ilmiöön. Tämä on ehdottomasti jatkotutkimuksissa asia, joka täytyy korjata.

Oliko tutkimus puhdas kvalitatiivinen tutkimus? Ei, mutta käsitevalinta auttoi tutkimaan ilmiötä ymmärtämisen kautta. Kananen (2014a, 23) määrittää perusjaottelun laadulliseen ja kvantitatiiviseen tutkimukseen. Koska puhtailla tilastoilla oltaisiin saatu vain tilastoja tehostamisesta, ei oltaisi voitu keskittyä välttämättä ymmärtämään vaikutuksia organisaatioon. Ymmärtämällä vaikutuksia laajemmin toimeksiantajan prosesseihin, pystyttiin löytämään myös kehitettäviä kohteita. Opinnäytetyön laji oli enemmän kehittämistutkimus, joka sisälsi laadullisen ja määrällisen tutkimuksen piirteitä.

Olisiko opinnäytetyön tutkimusotteen fokusoinnista ollut hyötyä tutkimuksen kannalta? Osittain kyllä, mutta tutkimusote pakotti tutkimaan opinnäytetyön aikana monia eri tutkimusmenetelmiä. Tästä on hyötyä jatkossa, kun tuotetaan lisätutkimuksia. Objektiivinen käsittely tutkimusotteen valinnasta jätetään tämän työn arvostelijoille.

Jatkokehitys

Opinnäytetyön tulokset eivät ole yksiselitteinen näkemys, miten jatkuva integrointi voi tehostaa peliprojektien työskentelyä niiden eri julkaisuvaiheissa. Opinnäytetyö tarjoaa vastauksen projektijohtamisen ja toimeksiantajan tilanteen näkökulmasta. Organisaation tulisi tutkia vielä lisää eri mahdollisuuksia, miten jatkuvaa integrointia voidaan hyödyntää esimerkiksi markkinoinnin automatisoinnissa. Testauksen suorittamista Unity3D-ympäristössä käyttäen jatkuvaa integrointia tulisi myös tutkia vielä lisää olemassa olevien projektien kohdalla.

Jatkuva integrointi tehostaa itsenäisten peliyritysten prosesseja julkaisussa ja projektin hallinnassa. Jatkuva integrointi parantaa tuotannon läpinäkyvyyttä, laadun hallintaa sekä yleistä työskentelyä automatisoimalla tuotteen julkaisun vaiheita. Automatisoimalla tuotantoa ja organisaatiota, kilpailukyky toisia yrityksiä vastaan paranee. Kehitystyön automaatio on vasta ensimmäinen askel kohti kilpailukykyisempää ja tehokkaampaa organisaatiota.

Tuotteita ja pelejä ei ole olemassa ilman asiakkaita. Opinnäytetyön aikana ilmeni mahdollisuuksia saada jatkuva integraatio toimimaan automaattisesti myös asiakasrajapinnassa. Onko esimerkiksi mahdollista saada jatkuvan integroinnin palvelu tuottamaan asiakkaille suunnattuja versiopäivitysten muutoslokeja? Kehittäjille tämä on jo mahdollista Jenkinsissä, mutta asiakkaille nämä muutoslokit eivät ole luettavia eivätkä edusta hyvää asiakaskokemusta. Olisiko myös mahdollista saada markkinoinnin automaation työkalu kuten Mautic toimimaan automaattisesti versiopäivitysten yhteydessä? Mautic on avoimen lähdekoodin sovellus (Powerful Marketing Automation n.d.), joten itsenäisellä peliyrityksellä markkinoinnin automaation käyttöönotto olisi halpaa. Itsenäisille peliyrityksille markkinoinnin automaation kehittäminen voi olla tärkeä jatkokehityksen kohde. Organisaatio toimii tehokkaammin, kun työtaakkaa asiakasrajapinnasta siirretään automaatiolle. Asiakasrajapinnan automaation tutkiminen toisi asiakasläheisemmän näkemyksen jatkuvan integroinnin käyttöön itsenäisissä peliyrityksissä.

Loppusanat

Tutkimus oli mielenkiintoinen, mutta jatkotutkimuksissa tulee syventyä erilaisiin tapoihin. Aihealueesta pitäisi tehdä puhdas kvalitatiivinen tutkimus, jossa haastatteluilla ja havainnoinnilla pyrittäisiin löytämään tarkat tehostamisen vaikutukset. Vaikutuksia pitäisi myös tutkia pitkällä aikavälillä, tai tutkimuksen tulisi vaihtoehtoisesti pohjautua kirjattuun aineistoon pitkältä aikaväliltä. Vaikka aineistoa kuitenkin olisi, tutkimuksen tulisi käsittää enemmän haastatteluja ja havainnointia.

Kuten johtopäätöksissä on kuvattu, jatkokehityksiä löytyy asiakkaiden hallinnan parista. Esimerkiksi problematiikka mittauksen rajauksesta voitaisiin tutkia asiakkaiden hallinnan näkökulmasta. Mitä jatkuva integrointi aiheuttaisi asiakassuhteille ja organisaation projektien hallinnointiin, kun tieto uusista versioista tulisi automaattisesti asiakkaille? Miten sosiaalisen median automaattiset päivitykset versiojulkaisujen yhteydessä vaikuttaisivat? Miten johtaminen ja julkaisu tehostuisi, kun tutkimuksessa otettaisiin huomioon asiakkaiden hallinta? Itsenäisille peliyrityksille vastaukset näihin kysymyksiin voisivat olla tärkeitä. Asiakkaat ovat peliprojektien kulmakivi.

Lähteet

A Guide to the Project Management Body of Knowledge. 2013. 5. p. Pennsylvania: Project Management Institute. Viitattu 13.10.2016. https://janet.finna.fi/Record/nelli27_jamk.2670000000343855.

Beck, K., Beedle, M., Bennekum, A., Cokcburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., Kern, J., Marick, B., Martin,R., Mellor, S., Schwaber, K., Sutherland, J. & Thomas, D. 2001. Manifesto for Agile Software Development. Viitattu 21.10.2016. http://agilemanifesto.org/.

Berg, A. 2015. Jenkins Continuous Integration Cookbook – Second Edition. Birmingham: Packt Publishing.

Blewitt, A. 2011. Hudson Renames to Jenkins. Viitattu 15.4.2016. http://www.infoq.com/news/2011/01/jenkins.

Booch, G. 1991. Object Orientated Design: With Applications. Kalifornia: Benjamin/Cummings Pub.

Caum, C. Continuous Delivery Vs. Continuous Deployment: What's the Diff? Viitattu 19.9.2016. https://puppet.com/blog/continuous-delivery-vs-continuous-deployment-what-s-diff.

Certified Scrum Master. 2016. Scrum Alliancen tuottama ja Reaktor-yhtiön käyttämä koulutusmateriaali scrummaster sertifikaatin koulutuksessa. Koulutusmateriaali.

Chaimungkalanont, M. 2007. Bamboo 1.0 Released. Viitattu 28.6.2016. http://blogs.atlassian.com/2007/02/bamboo_10_released/.

Command line Arguments. N.d. Unity3D-ohjelman dokumentaatio ohjelman käynnistämiseen komentokehotteesta. Viitattu 16.6.2016. http://docs.unity3d.com/Manual/CommandLineArguments.html.

Connecting JIRA to MySQL. 2016. Atlassianing dokumentaatio JIRA-palvelun yhdistämisestä MySQL-tietokantaan. Viitattu 14.4.2016. https://confluence.atlassian.com/adminjiraserver070/connecting-jira-applications-to-mysql-749382640.html.

Connecting to an LDAP directory. N.d. Atlassianin dokumentaatio LDAP-palvelun käyttämiseen. Viitattu 12.10.2016. https://confluence.atlassian.com/adminjiraserver071/connecting-to-an-ldap-directory-802592350.html.

Continuous Delivery Adaption Barriers. 2013. Blogikirjoitus DevOpsGuys Limited:n kotisivuilla 13.3.2016. Viitattu 13.8.2016. https://blog.devopsguys.com/2013/03/13/continuous-delivery-adoption-barriers/.

Create games, connect with your audience and achieve success. N.d. Unity3D-ohjelman esittelysivusto. Viitattu 28.10.2016. https://unity3d.com/unity.

Customizing Git – Git Hooks. N.d. Git-versionhallintaorganisaation esittely Git-palvelun muokkausmahdollisuuksiin. Viitattu 15.10.2016. https://git-scm.com/book/it/v2/Customizing-Git-Git-Hooks.

Eskola, J. & Suoranta, J. 2008. 8.p. Johdatus laadulliseen tutkimukseen. Tampere: Osuuskunta Vastapaino.

Fewster, M. & Graham, D. 1999. Software test automation, effective use of test execution tools. New York: ACM Press.

Friedman, P. 2016. How to Make Your Scrum Master Happy Using JIRA Integrations. Viitattu 1.10.2016. http://www.agiletrailblazers.com/blog/how-to-make-your-scrum-master-happy-using-jira-integrations.

Galactic Conquerors on Steam. N.d. Toimeksiantajan tuotteen Steam-jakelualustan virallinen sivusto. Viitattu 16.6.2016. http://store.steampowered.com/app/385680.

Git Basics - Tagging. N.d. Git-versiohallintaorganisaation esittely Git-palvelun perustoiminallisuuksiin. Viitattu 11.1.2016. https://git-scm.com/book/en/v2/Git-Basics-Tagging.

GitLab Features. N.d. GitLab-palvelun esittelysivu. Viitattu 22.7.2016. https://about.gitlab.com/features/.

Gril, J. 2008. The State of Indie Gaming. Viitattu 1.9.2016. http://www.gamasutra.com/view/feature/132041/the_state_of_indie_gaming.php.

Kananen, J. 2015. Kehittämistutkimuksen kirjoittamisen käytännön opas. Miten kirjoitan kehittämistutkimuksen vaihe vaiheelta. Jyväskylän ammattikorkeakoulun julkaisuja –sarja.

Kananen, J. 2014. Laadullinen tutkimus opinnäytetyön aiheena. Miten kirjoitan kvalitatiivisen opinnäytetyön vaihe vaiheelta. Jyväskylän ammattikorkeakoulun julkaisuja –sarja.

Kananen, J. 2014. Toimintatutkimus kehittämistutkimuksen muotona. Miten kirjoitan toimintatutkimuksen opinnäytetyönä. Jyväskylän ammattikorkeakoulun julkaisuja –sarja.

Kuusela H., Rintamäki T. 2002 Arvoa tuottava asiointikokemus. Tampere: Tampereen yliopisto.

Lacoste, J. 2016. Unity3dBuilder Plugin. Viitattu 22.3.2016. https://wiki.jenkins-ci.org/display/JENKINS/Unity3dBuilder+Plugin.

Lee, K. 2005. Realizing continuous integration. Viitattu 6.10.2016. http://www.ibm.com/developerworks/rational/library/sep05/lee/.

Lukkarinen, A. 2011. Iteratiivinen sovelluskehitys. Kehitysympäristön toteutus ja ylläpito. Opinnäytetyö. AMK. Viitattu 11.9.2016. http://urn.fi/URN:NBN:fi:amk-201105269796.

Miller, F., Vandome, A. & McBrewster, J. 2009. Abandonware. Computer software, Copyright, Office suite, Public domain, List of commercial video games released as freeware, Orphan works. Lontoo: Alpha Press.

Mitä tarkoittaa Early Access? N.d. Steam-jakelualustan tietopaketti Early Access –palvelusta. Viitattu 15.10.2016. http://store.steampowered.com/earlyaccessfaq/?l=finnish.

Novet, J. 2015. Source: Red Hat is buying Ansible for more than $100M. Viitattu 29.3.2016. http://venturebeat.com/2015/10/15/source-red-hat-is-buying-ansible-for-more-than-100m/.

OpenSSH Features. N.d. OpenSSH-palvelun ominaisuuksien esittelysivusto. Viitattu 16.10.2016. http://www.openssh.com/features.html.

Package Management. N.d. Ubuntu-käyttöjärjestelmän dokumentaatio pakettijärjestelmään. Viitattu 20.10.2016. https://help.ubuntu.com/lts/serverguide/package-management.html.

Powerful Marketing Automation. N.d. Mautic-markkinoinnin ja myynninautomaatio-ohjelman esittelysivu. Viitattu 10.10.2016. https://www.mautic.org/.

Puuniekka, A. & Saaranen-Kauppinen A. 2006. Tampere: Yhteiskuntatieteellinen tietoarkisto. Viitattu 19.10.2016. http://www.fsd.uta.fi/menetelmaopetus/kvali/L1.html.

Radigan, D. N.d. Scrum. A brief look into using the scrum framework for software development. Viitattu 31.5.2016. https://www.atlassian.com/agile/scrum.

Ratkaisujen Suomi. 2015. Valtioneuvosto. Viitattu 12.2.2016. http://valtioneuvosto.fi/documents/10184/1427398/Ratkaisujen+Suomi_FI_YHDISTETTY_netti.pdf/801f523e-5dfb-45a4-8b4b-5b5491d6cc82.

Romanos, E. 2014. Project Management. There is no right or wrong. Viitattu 12.10.2016. http://www.develop-online.net/opinions/project-management-there-is-no-right-or-wrong/0197141.

Röken, F. & Frommold, D. 2005. Automated Tests and Continuous integration in Game Projects. Viitattu 24.1.2016. http://www.gamasutra.com/view/feature/130682/automated_tests_and_continuous_.php.

Scrum Values. N.d. Scrum Alliancen ylläpitämä sivu scrumin arvoihin. Viitattu 8.7.2016. https://www.scrumalliance.org/why-scrum/core-scrum-values-roles.

Some statistics on the usage of Jenkins. N.d. Jenkins-palvelun ylläpitämä statistiikkasivusto palveluun liittyvistä tilastoista. Viitattu 26.10.2016. http://stats.jenkins.io/jenkins-stats/svg/svgs.html.

Standard Security Setup. 2015. Viitattu 12.10.2016. https://wiki.jenkins-ci.org/display/JENKINS/Standard+Security+Setup.

Strategy 2016–2020. 2015. Kinahmi Gamesin sisäinen asiakirja yhtiön kehittämistä varten vuosien 2016 ja 2020 välillä. Viitattu 17.2.2016.

Takeuchi, H. & Nonaka, I. 1986. The New New Product Development Game. Viitattu 21.10.2016. https://hbr.org/1986/01/the-new-new-product-development-game.

Tietoja toimialasta. 2016. Suomen pelialan kattojärjestön Neogamesin tarjoama tietopaketti pelialaan Suomessa. Viitattu 20.4.2016. http://www.neogames.fi/tietoa-toimialasta/.

Understanding the Bamboo CI Server. N.d. Atlassianin dokumentaatio Bamboo-palvelun käyttöön. Kuvio jatkuvan integroinnin perusteista. Viitattu 14.6.2016. https://confluence.atlassian.com/bamboo/understanding-the-bamboo-ci-server-289277285.html.

Using the issue collector. N.d. Atlassianin dokumentaatio Issue Collectorin käyttöön. Viitattu 16.10.2016. https://confluence.atlassian.com/adminjiraserver071/using-the-issue-collector-802592637.html.

Weiss, M. 2013. The Benefits of Continuous Integration. Viitattu 24.1.2016. https://blog.codeship.com/benefits-of-continuous-integration/.

What Is KVM. N.d. Open Virtualization Alliancen ylläpitämä KVM-palvelun esittelysivu. Viitattu 24.9.2016. https://openvirtualizationalliance.org/what-kvm.

What is Project Management? N.d. Project Management Insituten esittelysivu projekteihin ja niiden hallintaan. Viitattu 14.10.2016. https://www.pmi.org/about/learn-about-pmi/what-is-project-management.

Wilson, J. 2016. Steam (for PC). Viitattu 28.10.2016. http://uk.pcmag.com/steam/70940/review/steam-for-pc.

Zackariasson, P. & Wilson, T. 2012. The Video Game Industry. Formation, Present State, and Future. New York: Routledge.