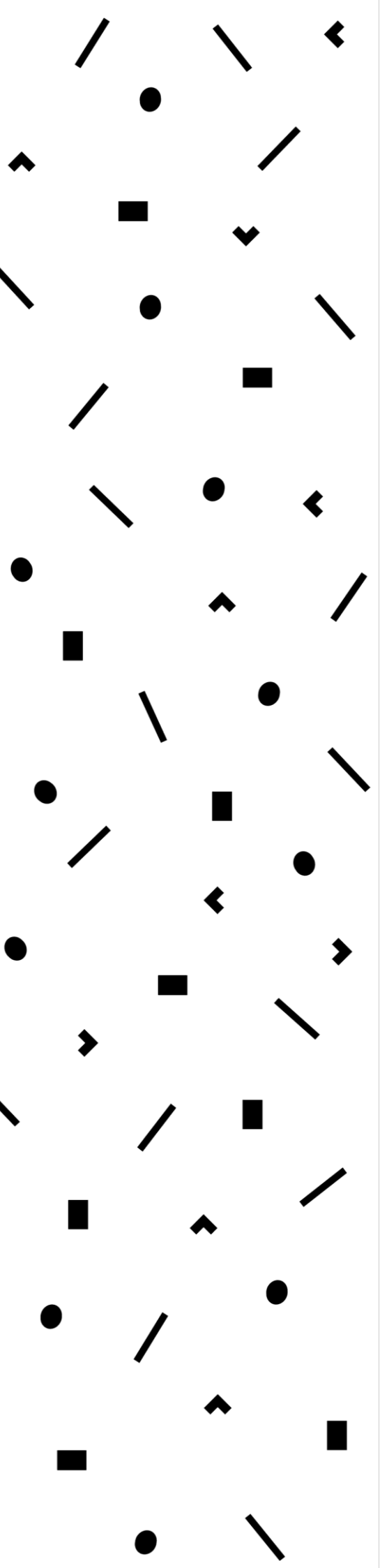


LAMK Lahden ammattikorkeakoulu
Lahti University of Applied Sciences

DRUPAL-JULKAISUJÄRJESTELMÄN KÄYTTÖÖNOTTOPROSESSIN AUTOMATISOINTI

LAHDEN AMMATTIKORKEAKOULU
Liiketalouden ala
Tradenomi, tietojenkäsittelyn koulutus
Kevät 2018
Jaakko Luhtamäki

Tiivistelmä

Tekijä(t)	Julkaisun laji	Valmistumisaika
Luhtamäki, Jaakko	Opinnäytetyö, AMK	Kevät 2018
	Sivumäärä	
	47	

Työn nimi

Drupal-julkaisujärjestelmän käyttöönottoprosessin automatisointi

Tutkinto

Tradenomi, tietojenkäsittelyn koulutus

Tiivistelmä

Opinnäytetyössä perehdyttiin verkkosovellusten käyttöönottoprosessiin ja siinä keskityttiin tarkemmin Drupal-julkaisujärjestelmään. Tavoitteena työssä oli löytää työkaluja ja menetelmiä käyttöönottoprosessin automatisointiin, jotta siitä saataisiin tehokkaampi ja vähemmän virhealtis. Työn toimeksiantaja oli Reason Solutions Oy.

Työn teoriaosuudessa selvitettiin aluksi kirjallisuuden avulla, mitä ohjelmiston käyttöönotto on, miksi se kannattaa automatisoida ja kuinka automatisointi voidaan toteuttaa. Kun nämä perusteet oli saatu selville, perehdyttiin seuraavaksi Drupalin toimintaan ja käyttöönottoprosessiin.

Työn tutkimusosuus toteutettiin suunnittelutieteellistä tutkimusmenetelmää noudattaen, jonka tuotoksena toteutettiin ratkaisu automaatiota hyödyntävästä käyttöönottoprosessista. Toteutetulla ratkaisulle asetettiin tavoitteeksi aktiivisen ajankäytön ja avainhenkilöihin sidonnaisuuden vähentäminen sekä luotettavuuden ja jäljitettävyyden parantaminen.

Ratkaisun toimivuutta arvioitiin rakentamalla työtilannetta mukaileva testiympäristö, jossa suoritettiin muutostöitä esimerkkiprojektille. Muutostöistä kerättiin havainnoimalla laadullista aineistoa, jonka perusteella tehtiin yleistyksiä käyttäen induktiivista päättelyä. Päätelmien perusteella työssä toteutettu ratkaisu saavutti sille asetetut tavoitteet.

Löydösten perusteella todettiin, että automaatiolla voidaan saavuttaa merkittäviä parannuksia Drupal-julkaisujärjestelmän käyttöönottoprosessissa ja se voidaan toteuttaa Git-versionhallintajärjestelmällä, Jenkins-automaatiopalvelimella ja Ansible-komentorivityökalulla. Toteutetun ratkaisun suurimmaksi ongelmaksi havaittiin virheentarkistuksen puute.

Avainsanat

julkaisujärjestelmä, käyttöönottoprosessi, automaatio, Drupal

Abstract

Author(s)	Type of publication	Published
Luhtamäki, Jaakko	Bachelor's thesis	Spring 2018
	Number of pages	
	47	

Title of publication

Automating the deployment process of Drupal content management system

Name of Degree

Bachelor's Degree Programme in Business Information Technology

Abstract

The thesis examined the deployment process of web applications and focused on the Drupal content management system. The goal was to find out tools and methods that could be used to automate the deployment process to make it more efficient and less error prone. The study was commissioned by Reason Solutions Oy.

The theoretical part of the thesis discusses the concept of application deployment, the possible benefits achieved by automating the process, and how to do this. After discussing these basic concepts, the thesis focuses on the usage, requirements, and the deployment process of Drupal. This information was gathered from literary material related to the topic, mainly from books, reports and the Internet.

A study was then conducted using design science research method, in which a possible solution for automating the deployment process was produced as an artifact. Reducing active work time and reliance on key personnel along with increasing reliability and traceability were set as goals for the artifact.

The produced solution was evaluated by doing development tasks to an example project in an environment resembling work situation. Qualitative data was collected by making observations during the development tasks and this data was then analyzed using inductive reasoning. It was concluded that the produced solution achieved the set goals.

Based on the findings it was concluded that notable improvements can be achieved by automating the deployment process of the Drupal content management system and it can be done by using the Git version control system, Jenkins automation server and Ansible command line tool. Lack of error checking was found to be the main problem of the produced solution.

Keywords

content management system, deployment process, automation, Drupal

SISÄLLYS

1 JOHDANTO

Verkkosivustoista puhuttaessa käytetään nykyään usein myös termiä verkkosovellus, sillä sivustot eivät enää yleensä ole vain staattisia dokumentteja. W3Techs sivuston julkaisemien tilastojen (W3Techs 2018) mukaan noin puolet kaikista verkkosivustoista käyttävät taustalla jotain julkaisujärjestelmää, joista suurimmat ovat WordPress, Joomla ja Drupal.

Julkaisujärjestelmät tarjoavat monia edistyneitä ominaisuuksia sisällön luontiin ja hallintaan, joiden ansiosta sivustolle voidaan tehdä muutoksia koskematta järjestelmän lähdekoodiin. Tästä huolimatta järjestelmiin täytyy aika ajoin tehdä päivityksiä ja muokkauksia, jotka vaativat koodimuutoksia.

Nämä muutokset voivat olla pieniäkin, mutta koska järjestelmät ovat monimutkaisia kokonaisuuksia, joissa pienelläkin muutoksella voi olla kauaskantoisia seurauksia, ei niitä yleensä tehdä suoraan julkisessa tuotantoympäristössä. Yleinen työtapa on toteuttaa muutokset ensin paikallisessa kehitysympäristössä ja varmistaa niiden oikeanlainen toiminta, jonka jälkeen ne voidaan ottaa käyttöön myös tuotantoympäristössä.

Käyttöönottoprosessin suorittaminen vaatii tarkkaavaisuutta ja se on luonteeltaan hyvin virhealtista työtä. Vaikka yksittäinen käsityönä suoritettu käyttöönotto ei välttämättä ole kovin työläs, kasvaa prosessiin käytettävä työaika projektimäärän kasvaessa huomattavaksi. Koska prosessissa toistetaan usein samoja vaiheita, voidaan se automatisoida hyvin pitkälle, mutta syystä tai toisesta automaatiota ei kuitenkaan aina hyödynnetä. Tässä opinnäytetyössä perehdytään käyttöönottoprosessin automatisointiin ja sen hyötyihin sekä myös käytännön toteutukseen.

2 TUTKIMUSSUUNNITELMA

2.1 Tausta

Tutkimuksen toimeksiantajana on Lahdessa ja Helsingissä toimiva ohjelmistoalan palveluja tarjoava Reason Solutions Oy, joka työllistää n. 15 henkilöä. Yrityksen tarjoamiin palveluihin kuuluu avoimen lähdekoodin julkaisujärjestelmillä toteutettujen verkkosivustojen kehitys ja ylläpito. Käytössä olevia julkaisujärjestelmiä ovat mm. Drupal, WordPress ja Concrete5.

Näiden järjestelmien kehitykseen yrityksessä käytetään yleensä paikallisesti pystytettyjä kehitysympäristöjä, joista järjestelmät viedään ajoittain loppukäyttäjien saataville julkiseen tuotantoympäristöön. Tämä työtapa on asiakkaan kannalta hyvä, sillä sen avulla palvelukatkot voidaan minimoida, mutta yritykselle tämä tuottaa kuitenkin lisätyötä. Julkaisujärjestelmien käyttöönotto suoritetaan yrityksessä pitkälti käsityönä ja vaikka se on toimiva menetelmä, on se havaittu aikaa vieväksi ja virhealttiiksi.

Idea tähän opinnäytetyöhön syntyi työharjoittelun yhteydessä keväällä 2018. Yrityksessä oli jo aiemmin tiedostettu tarve kehittää käyttöönoton työmenetelmiä ja siitä syntyi luontevasti aihe tälle opinnäytetyölle. Päätarkoituksena työssä oli kartoittaa mahdollisia keinoja avoimen lähdekoodin julkaisujärjestelmien käyttöönottoprosessien automatisointiin. Työssä käsiteltäväksi julkaisujärjestelmäksi valittiin Drupal, koska työn kirjoittajan työtehtävät liittyivät työharjoittelun aikana pääosin Drupaliin, joten työ toimi näin myös oppimisalustana järjestelmään yleisesti.

2.2 Tutkimuskysymys

Tutkimuskysymys tässä opinnäytetyössä on:

Kuinka automaatiota voidaan käytännössä hyödyntää Drupal-julkaisujärjestelmän käyttöönottoprosessin kehityksessä?

Käyttöönottoprosessin lähtökohtana on käsityönä suoritettava käyttöönotto, jossa siis työntekijä itse suorittaa käyttöönottoon tarvittavat toimenpiteet vaihe vaiheelta. Tavoitteena on löytää käytännöllisiä menetelmiä ja työkaluja, joilla toistuvasti suoritettavia työvaiheita voidaan automatisoida ja siirtää tietokoneiden vastuulle. Työssä keskitytään Drupal-julkaisujärjestelmään, mutta löydöksiä voidaan mahdollisesti soveltaa muihinkin julkaisujärjestelmiin ja verkkosovelluksiin.

Käyttöönottoprosessilla viitataan tässä työssä niihin vaiheisiin ja menetelmiin, joilla kehitysympäristössä tehty työ viedään tuotantoympäristöön loppukäyttäjien saataville. Tuotantoympäristöllä viitataan tässä julkiseen verkkopalvelimeen ja tutkimuksessa oletetaan, että tälle palvelimelle on jo tehty Drupalin käyttöön tarvittavat asennukset ja määrittelyt.

Tietokoneen käyttöjärjestelmää lukuun ottamatta kaikki tässä työssä käytettävät ohjelmistot ja palvelut ovat joko kokonaan ilmaisia tai niistä käytetään saatavilla olevaa ilmaisversiota. Drupalista käytetään uusinta versiota 8 ja kun tässä työssä puhutaan Drupalista, viitataan sillä nimenomaan versioon 8, ellei toisin mainita. Keskittymällä vain yhteen versioon voidaan käyttöönottoprosessia tutkia syvemmin ja näin ollen saadaan kerättyä käytännöllisempää tietoa.

2.3 Tutkimusmenetelmä

Tässä opinnäytetyössä tutkimuskysymyksellä tavoitellaan tietoa siitä, kuinka automaatiota voidaan käytännössä hyödyntää Drupal-julkaisujärjestelmän käyttöönottoprosessin kehityksessä. Tavoitteena ei siis ole pelkästään selvittää, mitä hyötyä automaatiosta on, vaan myös miten käyttöönottoprosessin automatisointi käytännössä toteutetaan.

Tutkimuskysymykseen voitaisiin todennäköisesti löytää vastaus monenlaisilla tutkimusmenetelmillä. Koska verkkosovellusten käyttöönottoprosessien automatisointia on toteutettu aiemminkin, voisi menetelmiä ja ohjeita käytännön toteutukseen mahdollisesti löytää käymällä läpi aiempia tutkimuksia, haastattelemalla alan ammattilaisia ja selvittämällä kyselyillä, millaisia menetelmiä alan yrityksissä on käytössä. Näillä menetelmillä voisi kuitenkin olla hankala huomioida työn lähtökohdat ja kohdistaa tutkimusta tarkasti kohteeksi valittuun julkaisujärjestelmään. Lisäksi tämän opinnäytetyön kirjoittajan tavoitteena on myös itse tutustua Drupal-julkaisujärjestelmän toimintaan, joten tutkimusmenetelmän tulisi olla käytännönläheinen.

Näiden pohdintojen perusteella voidaan käytettäväksi tutkimusmenetelmäksi valita suunnittelutieteellinen (engl. design science) tutkimusmenetelmä, sillä siinä rakennetaan tuotoksia tiettyyn tarkoitukseen ja arvioidaan, kuinka hyvin toteutettu tuotos toimi käytännössä (March & Smith 1995, 254). Tällä tutkimusmenetelmällä voidaan tutkimus suorittaa toteuttamalla ratkaisu automaatiota hyödyntävästä Drupal-julkaisujärjestelmän käyttöönottoprosessista ja arvioimalla sen toimivuutta käytännössä.

2.3.1 Suunnittelutieteellinen tutkimusmenetelmä

Suunnittelutieteellinen tutkimusmenetelmä on teknologiaan suuntautunut tutkimusmenetelmä, jonka tarkoituksena on tuottaa ihmisten toimintaa palvelevia tuotoksia. Näitä tuotoksia arvioidaan niiden toimivuuden ja käytännöllisyyden kannalta. (March & Smith 1995, 253.)

Suunnittelutieteellinen tutkimusmenetelmä voidaan jakaa kuuteen aktiviteettiin, jotka on lyhyesti kuvattu taulukossa 1. Taulukossa vasemmassa sarakkeessa on listattu aktiviteetti ja oikeassa sarakkeessa toimet, jotka aktiviteetissa suoritetaan.

Taulukko 1.Suunnittelutieteellinen tutkimusmenetelmä. (Peffers,Tuunanen,Rothenberger & Chatterjee 2008, 12-14)

Aktiviteetti	**Suoritettavat toimet**
1. Ongelman tunnistaminen ja motivaatio.	Määritellään tutkittava ongelma ja perustellaan, miksi sen ratkaiseminen on arvokasta.
2. Ratkaisun tavoitteiden määrittely.	Tutkittavan ongelman perusteella määritellään tavoitteet, jotka ratkaisun tulisi saavuttaa.
3. Suunnittelu ja toteutus.	Suunnitellaan ja toteutetaan tuotos, joka ratkaisee tutkittavan ongelman.
4. Demonstrointi.	Demonstroidaan tuotoksen kyvykkyys ratkaisemalla tutkittavan ongelman ilmentymä.
5. Arviointi.	Arvioidaan tuotoksen toimivuutta vertaamalla havainnoituja tuloksia määriteltyihin ratkaisun tavoitteisiin.
6. Kommunikointi.	Tiedotetaan tutkitusta ongelmasta ja saavutetusta ratkaisusta sekä sen toimivuudesta asiasta kiinnostuneille tahoille.

Vaikka aktiviteetit ovat numeroitu, ei niitä välttämättä suoriteta järjestyksessä, vaan tutkimuksen aloituskohdan voi valita tapauskohtaisesti. Ensimmäisestä vaiheesta aloitettava ongelmakeskeinen lähestymistapa sopii tilanteeseen, jossa tutkimuksen idea on syntynyt havaitusta ongelmasta. (Peffers ym. 2008, 14.)

2.3.2 Valitun tutkimusmenetelmän soveltaminen

Tämän tutkimuksen idean voidaan katsoa syntyneen havaitusta ongelmasta, joten tutkimus voidaan luvun 2.3.1 perusteella aloittaa tutkittavan ongelman määrittelystä. Ongelman määrittelyn jälkeen loput aktiviteetit suoritetaan järjestyksessä.

Ongelman tunnistaminen tehdään perehtymällä yleisellä tasolla verkkosovellusten käyttöönottoprosesseihin ja niiden automatisointiin. Kerätyn tiedon pohjalta määritellään suurimmat ongelmakohdat, ottaen huomioon myös tämän opinnäytetyön toimeksiantajalta kerätyt taustatiedot ja nykyisen käyttöönottoprosessin lähtötaso.

Ongelman tunnistamisen jälkeen seuraava aktiviteetti on ratkaisun tavoitteiden määrittely, jossa määritellään millainen tutkimuksessa toteutettavan käyttöönottoprosessin tulisi olla, jotta se voitaisiin katsoa onnistuneeksi. Luvun 2.3.1 perusteella tavoitteet määritellään tutkittavan ongelman perusteella, eli tavoitteet määräytyvät edellisessä aktiviteetissa määriteltyjen ongelmakohtien perusteella.

Kun tavoitteet on määritelty, suunnitellaan automaatiota hyödyntävä käyttöönottoprosessi, jonka tavoitteena on ratkaista sille asetetut tavoitteet mahdollisimman hyvin. Tämän suunnitelman pohjalta toteutetaan ratkaisu myös käytännössä. Suunnittelussa käytetään pohjana kirjallisuudesta kerättävää tietoa verkkosovellusten käyttöönotosta ja Drupal-julkaisujärjestelmästä sekä myös tunnistettuja ongelmakohtia ja ratkaisulle asetettuja tavoitteita. Ratkaisun käytännön toteutuksessa käytetään apuna erilaisia verkosta löytyviä ohjeistuksia sekä ohjelmistojen dokumentaatioita.

Luvun 2.3.1 perusteella suunnittelutieteellisessä tutkimuksessa toteutetun ratkaisun kyvykkyys osoitetaan ratkaisemalla tutkittavan ongelman ilmentymä. Tässä työssä tämä toteutetaan rakentamalla testiympäristö, joka mukailee käytännön tilannetta, eli se koostuu kehitys- ja tuotantoympäristöistä. Tässä testiympäristössä ratkaisun toimivuutta esitellään suorittamalla muutostöitä Drupal-pohjaiselle verkkosivustolle kehitysympäristössä, jonka jälkeen muutokset viedään tuotantoympäristöön toteutetun ratkaisun välityksellä. Muutostöiden suorituksesta ja toteutetun ratkaisun toimivuudesta tullaan keräämään havainnoimalla laadullista ainestoa. Havainnot kirjataan osaksi tätä opinnäytetyötä ja ne kerätään tämän työn kirjoittajan toimesta, eli käytännössä siis kirjataan ylös työn kirjoittajan näkemyksen mukaan merkittävimmät huomiot toteutettavan ratkaisun toimivuudesta. Havainnoissa keskitytään etenkin ensimmäisessä aktiviteetissa määriteltyihin ongelmakohtiin, mutta myös muut huomionarvoiset asiat kirjataan ylös, sillä ne voivat olla hyödyllisiä mahdollisen jatkokehityksen kannalta.

Ratkaisun toimivuutta arvioidaan induktiivisen päättelyn avulla, jossa luodaan yleistyksiä rajallisen aineiston perusteella (Peda.net 2018). Aineistona toimii edeltävässä aktiviteetissa kerätty aineisto. Arvioinnissa käydään ratkaisulle asetetut tavoitteet läpi yksitellen ja päätellään kerätyn aineiston perusteella, kuinka hyvin toteutettu ratkaisu pystyi ne ratkaisemaan.

Lopuksi tutkimuksen tuloksista luodaan johtopäätöksiä tarkastelemalla, saavuttiko toteutettu ratkaisu sille asetetut tavoitteet sekä pohtimalla, mitä vaikutuksia tuloksilla on käyttöönottoprosessin kehityksen kannalta. Jos käyttöönottoprosessi voidaan katsoa kehittyneeksi lähtötilanteeseen verrattuna, toimii toteutettava ratkaisu vastauksena tutkimuskysymykseen. Jos taas käyttöönottoprosessi ei ole kehittynyt merkittävästi, ei silloin myöskään voida vastata tutkimuskysymykseen kokonaisuudessaan.

3 KÄYTTÖÖNOTTOPROSESSI

3.1 Ohjelmiston käyttöönotto

Ohjelmiston käyttöönotto on monivaiheinen prosessi, jossa kehitetty ohjelmisto saatetaan sen loppukäyttäjien saataville. Hyvin sujuneen käyttöönoton jälkeen voidaan iloita työn tuloksista, mutta jos prosessi ei kuitenkaan suju toivotulla tavalla ja testiympäristössä toiminut ohjelma hajoaa tositilanteessa, muuttuu tilanne nopeasti painajaiseksi. (Stephens 2015, 203.)

Onnistuneeseen käyttöönottoon vaadittavat resurssit ja toimenpiteet ovat tapauskohtaisia. Suuren projektin käyttöönottoon voidaan ajatella kuuluvan mm. tilat, laitteistot, dokumentaatio, koulutus, tietokannat, ulkoiset ohjelmistot ja tietenkin oma ohjelmisto. (Stephens 2015, 209-210.). Tämä työ käsittelee käyttöönottoa enemmän käytettävien työkalujen ja menetelmien kannalta, mutta on hyvä ymmärtää käyttöönoton käsittävän paljon muutakin.

3.2 Automaation hyödyt käyttöönottoprosessissa

Manuaalisen käyttöönottoprosessin ongelmia ja automaation hyötyjä on pohdittu Octopus Deploy yrityksen vuonna 2014 julkaisemassa raportissa (Octopus Deploy 2014). Raportti käsittelee automaation hyötyjä käyttöönottoprosessissa ja siinä kerrotaan yrityksen tekemästä kyselystä, johon vastasi 72 käyttöönottoprosessin hiljattain automatisoinutta yritystä.

Manuaalisen käyttöönottoprosessin ongelmiksi raportissa mainitaan mm. ihmisten erehtyväisyys, seurannan puutteellisuus, hitaus ja liiallinen sidonnaisuus avainhenkilöihin. Avainhenkilöihin sidonnaisuudella tarkoitetaan tässä tilannetta, jossa käyttöönoton suorittaminen on vain tiettyjen henkilöiden vastuulla. Jos henkilöt ovat jostain syystä estyneitä, voi se aiheuttaa suuria viiveitä tai pakottaa muuta henkilöstöä paikkaamaan tilannetta ilman riittävää ymmärrystä asioista. (Octopus Deploy 2014.)

Kun yrityksiltä kysyttiin manuaaliseen käyttöönottoprosessiin käytettävästä ajasta, 40% arvioi käyttävänsä 1-3 tuntia, 27% 15-30 minuuttia ja 20% 4-8 tuntia tai enemmän. Automaation käyttöönoton jälkeen 90% yrityksistä kertoi prosessiin kuluvan alle 30 minuuttia ja 64 prosentissa näistä tapauksista prosessin kerrottiin kestävän vain muutaman minuutin. (Octopus Deploy 2014.)

Automaation vaikutuksista kysyttäessä yritykset kertoivat mm. seuraavia asioita:

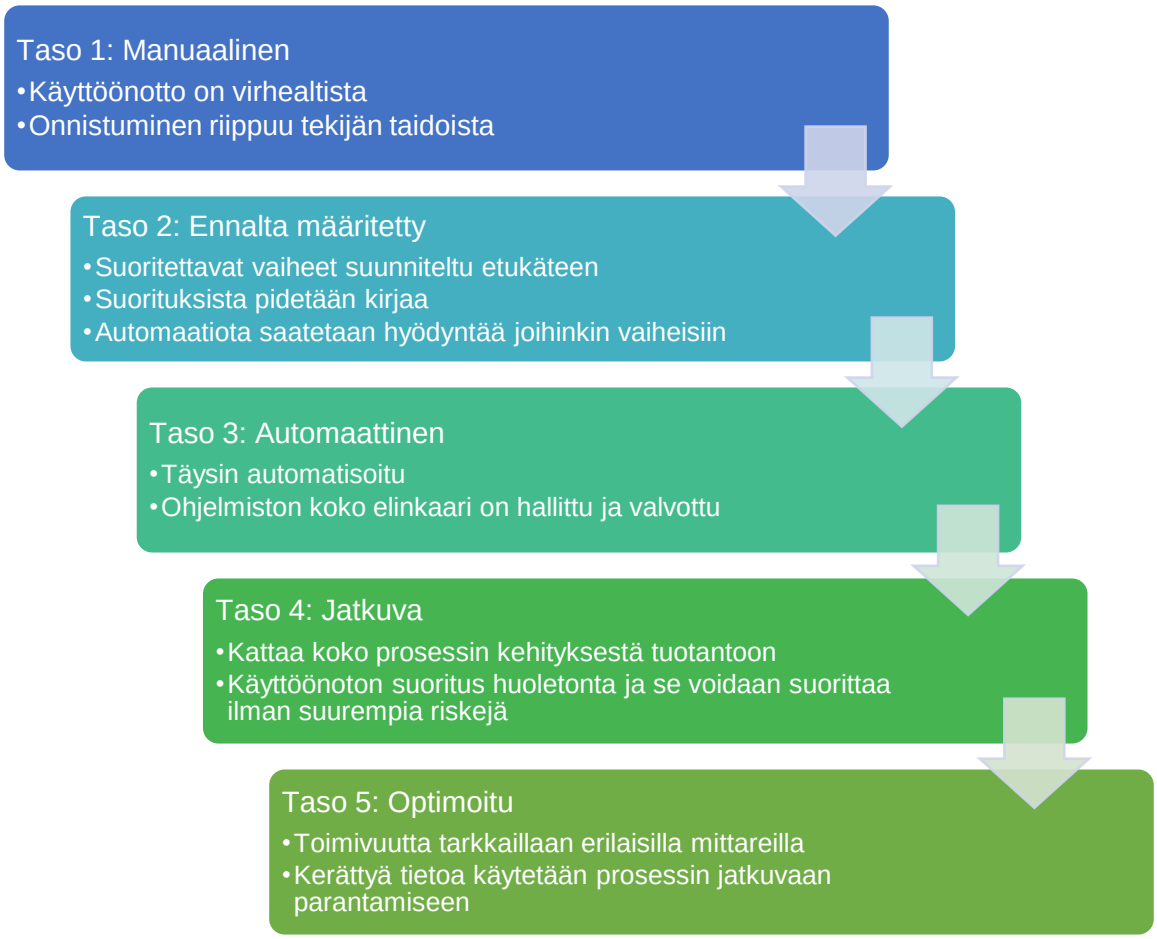
- 94% koki käyttöönottoprosessin vähemmän riskialttiiksi.

- 90% kertoi tiimin luottamuksen käyttöönottoprosessin toimivuuteen kasvaneen.
- 72% kertoi hyötyneensä automaatiojärjestelmän tuomasta selvyydestä käyttöönoton tilan seurantaan.
- 56% kertoi saavansa nopeammin palautetta testaajilta ja loppukäyttäjiltä tiheämmän käyttöönoton ansiosta. (Octopus Deploy 2014.)

Raportin perusteella käyttöönottoprosessin automatisointi tuo useissa tapauksissa merkittäviä parannuksia manuaaliseen prosessiin verrattuna, mutta hyödyllisyyden aste kuitenkin vaihtelee.

3.3 Käyttöönottoprosessin automaation tasot

Kirjassa DevOps for Digital Leaders (Ravichandran,Taylor & Waterhouse 2016, 91-92) käsitellään ohjelmistojen käyttöönottoprosessin automatisointia ja siihen käytettäviä menetelmiä. Kirjassa käyttöönottoprosessin kehittyneisyys on jaettu viiteen tasoon, joiden pääkohdat on kuvattu kuviossa 1.



Kuvio 1. Käyttöönottoprosessin kehittyneisyys. (Ravichandran ym. 2016, 91-92).

Ensimmäisellä tasolla käyttöönotto rajoittaa yrityksen toimintaa ja hidastaa uusiin haasteisiin reagointia. Toisella tasolla käyttöönotto on suunnitelmallista ja käyttöön on todennäköisesti otettu aputyökaluja, kuten versionhallintajärjestelmä tai erilaisia automaatiotyökaluja. Kolmannella tasolla käytössä on yleiskäyttöinen automaattinen käyttöönottoprosessi, jonka tuotoksien koko elinkaari on jäljitettävissä. Neljännellä tasolla käyttöönotto on ennakoitavaa ja yritys voi huoletta suorittaa ohjelmiston käyttöönoton monenlaisiin ympäristöihin. Viidennellä tasolla kaikki käyttöönottoprosessin osat toimivat yhdessä luotettavasti ja mahdollistavat useiden ohjelmistojen käyttöönoton ja hallinnan keskitetysti. Kehittämällä käyttöönottoprosessin tasoa työnteosta saadaan tehokkaampaa ja aikaa voidaan käyttää enemmän esimerkiksi uusien ominaisuuksien kehittämiseen. (Ravichandran ym. 2016, 92.)

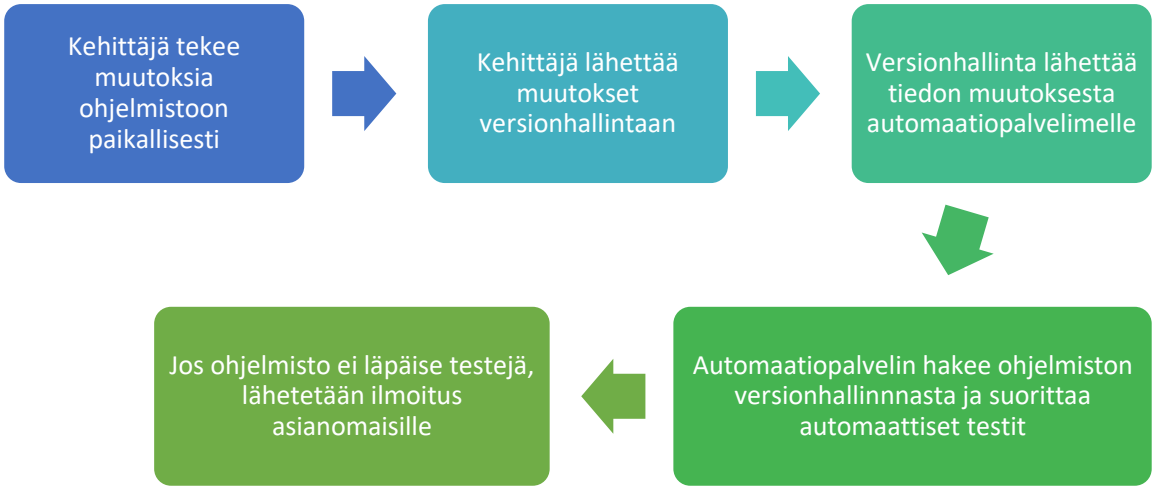
3.4 Työkalut ja menetelmät

Verkkosovelluksen käyttöönottoprosessi voidaan suorittaa hyvin monella tavalla. Keinot vaihtelevat helppokäyttöisistä palveluista monimutkaisiin komentorivityökaluihin ja käyttöönottojärjestelmiin. Yksinkertaisimmillaan prosessi voi sisältää tiedostojen siirron verkkopalvelimelle FTP-ohjelman välityksellä. Monimutkaisempien verkkosovellusten käyttöönotossa pelkkä uusien tiedostojen tuonti ei välttämättä riitä, vaan verkkopalvelin täytyy valmistella käyttöönottoa varten. Lisäksi voi olla tarpeellista suorittaa erilaisia toimenpiteitä, kuten tiedosto-oikeuksien muokkaamisia ja palveluiden uudelleenkäynnistyksiä. (Coyier 2013).

Ohjelmistokehityksessä tapahtuu virheitä ja jotta niistä voitaisiin palautua, täytyy sovelluksen tiedostoista säilyttää eri versioita, jotka voidaan tarvittaessa palauttaa. Tätä varten on kehitetty versionhallintajärjestelmät, jotka pitävät kirjaa tiedostomuutoksista ja mahdollistavat paluun aiempaan versioon. (Git 2018a.) Useat versionhallintajärjestelmät tukevat myös työhaaroja, joiden avulla ominaisuuksia voidaan kehittää erillään ohjelmiston päätyöhaarasta ja näin pitää päätyöhaara siistinä. (Git 2018b.) Versionhallintajärjestelmää voidaan käyttää myös tiedostojen synkronointiin eri ympäristöjen välillä. Sovelluksen tiedostot voidaan esimerkiksi siirtää kehitysympäristöstä verkossa sijaitsevaan versionhallintajärjestelmään, josta ne voidaan puolestaan noutaa tuotantoympäristöön. (Coyier 2013.)

Verkkosovelluksen käyttöönotossa voidaan myös hyödyntää monenlaisia komentorivityökaluja, joiden avulla voidaan suorittaa toimintoja kirjoittamalla käskyjä terminaalin. Käskyjä voidaan antaa myös etäyhteyden välityksellä verkkopalvelimelle. Yksinkertaisemmat komentorivityökalut suorittavat vain tiettyjä rajattuja tehtäviä, kun taas monimutkaisemmat vaativat toimiakseen erillisiä tiedostoja, joissa määritetään suoritettavat toiminnot. (Coyier 2013.)

Ohjelmistojen käyttöönoton toimivuuden parantamiseksi on kehitetty jatkuvan integraation työmenetelmä, jossa kehittäjät yhdistävät tekemänsä työn tiheään tahtiin keskitettyyn työhaaraan. Yhdistämiseen kuuluu yleensä automaattinen testausprosessi, jonka tarkoituksena on varmistaa koodin yhteensopivuus ja estää ongelmien eteneminen pidemmälle. Jatkuva integraatio vaatii tuekseen automaatiopalvelimen, joka suorittaa sille määritetyt toimenpiteet automaattisesti aina, kun kehittäjät tekevät muutoksia keskitettyyn työhaaraan. (Pittet 2018.) Esimerkki jatkuvan integraation työmenetelmän etenemisestä on esitetty kuviossa 2.



Kuvio 2. Jatkuvan integraation työmenetelmä. (Ellingwood, 2017).

Jatkuvan integraation työmenetelmään käytettäviä automaatiopalvelimia voidaan hyödyntää myös verkkosovellusten käyttöönottoprosessissa. Automaatiopalvelin voidaan esimerkiksi ohjelmoida suorittamaan määritelty käyttöönottoprosessi automaattisesti aina, kun versionhallintajärjestelmään lähetetään muutoksia. (Coyier 2013.)

3.5 Käyttöönottoprosessin suunnittelu

Ennen käyttöönottoprosessin suunnittelua on järkevää miettiä projektin laajuutta. Projektin laajuus voi vaihdella omaan käyttöön tehdystä työkalusta miljoonien ihmisten käyttämiin sovelluksiin, kuten selaimet ja käyttöjärjestelmät. Mitä suurempi projekti on, sitä todennäköisemmin jokin käyttöönotossa menee vikaan. Varsinkin suurempien projektien kohdalla käyttöönottoprosessin suunnittelu kannattaa tehdä huolella. (Stephens 2015, 204.)

Koska jotain kuitenkin yleensä menee vikaan, on käyttöönottoprosessin suunnittelussa hyvä miettiä, kuinka ongelmatilanteissa toimitaan. Jos ongelmiin on varauduttu, voidaan kohdatut ongelmat mahdollisesti ratkaista ja jatkaa käyttöönottoprosessin suorittamista.

Pelkkä käyttöönottoprosessin keskeyttäminen tai peruminen voi olla vaikeaa tai mahdotonta, jos siihen ei ole varauduttu. (Stephens 2015, 204-205.)

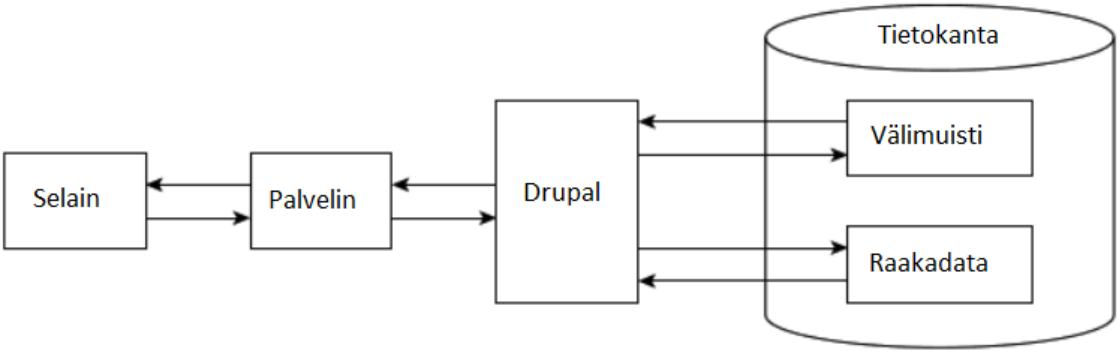
Käyttöönottoprosessin suunnittelu voidaan aloittaa listaamalla vaiheet, jotka halutaan suorittaa. Tämän jälkeen listataan vaiheet, jotka voivat epäonnistua ja ratkaisut niihin. Ongelmia voi syntyä myös odottamattomista syistä, kuten laiterikoista tai verkkokatkoksista. Tästä syystä on tärkeää suunnitella, kuinka ongelmista palaudutaan. Ohjelmistojen tapauksessa voidaan esimerkiksi tehdä varmuuskopiot ennen käyttöönoton suorittamista, joiden avulla järjestelmä voidaan palauttaa alkutilanteeseen. (Stephens 2015, 205.)

4 DRUPAL-JULKAISUJÄRJESTELMÄ

4.1 Perusteet

Drupal on PHP-ohjelmointikielellä kirjoitettu modulaarinen julkaisujärjestelmä, eli se rakentuu monesta pienestä osasta. Sitä voidaan kuvailla myös sovelluskehykseksi (engl. framework), sillä se on suunniteltu joustavaksi ja muokattavaksi. (Bucher, Dunwoodie & Shreves 2011, 10-11.)

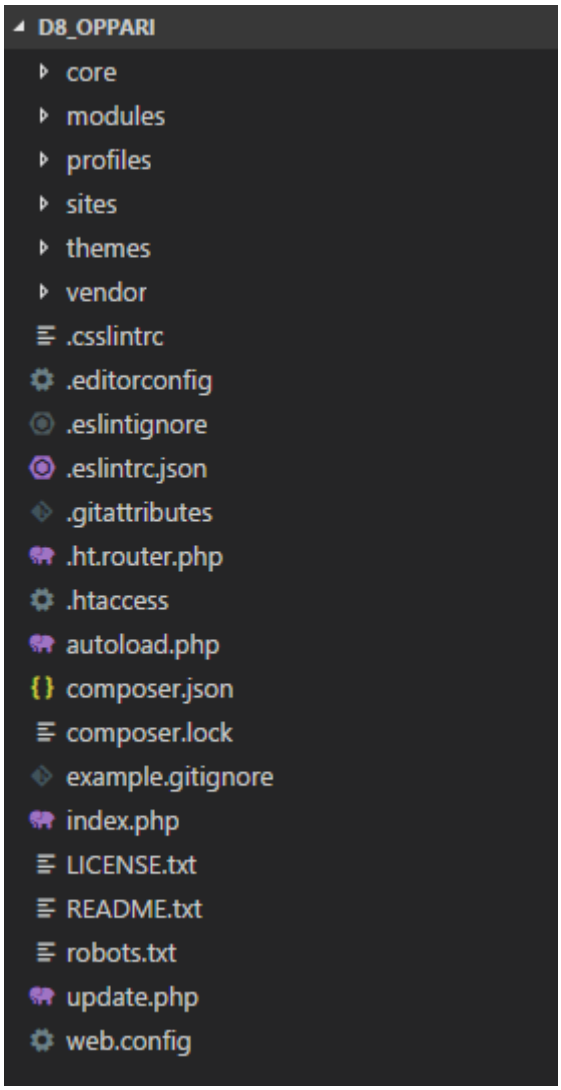
Drupal on verkkosovellus, joka vaatii toimiakseen verkkopalvelimen ja tietokannan. Tietokantaan tallennetaan mm. käyttäjien luoma sisältö ja järjestelmän konfiguraatio. Kun käyttäjä tekee pyynnön sivustolle esimerkiksi verkkoselaimen kautta, välittää verkkopalvelin sen Drupalille. Drupal käsittelee pyynnön ja noutaa pyydetyn resurssin tietokannasta tai välimuistista ja palauttaa sen verkon välityksellä käyttäjälle. Tämä prosessi on kuvattu kuvassa 1. (Bucher ym. 2011, 15.)



Kuva 1. Drupalin toiminta. (Bucher ym. 2011, 15).

4.2 Tiedostorakenne

Drupalin tiedostot on jaettu kansioihin määrätyllä tavalla. Käyttöönoton kannalta ei välttämättä ole oleellista käydä tiedostorakennetta läpi syvällisesti, mutta tutustumalla perusteisiin saadaan järjestelmän toiminnasta selkeämpi yleiskuva. Drupalin perustason tiedostorakenne näkyy kuvassa 2.



Kuva 2. Drupalin tiedostorakenne

Tiedostorakenteen pääkohdat ovat:

- Pohjatasolla sijaitsee mm. index.php-tiedosto, jota kautta kaikki pyynnöt saapuvat verkkopalvelimelta Drupalin käsiteltäväksi. (Drupal.org 2018b).
- Core-kansiossa sijaitsee kaikki Drupalin peruskäyttöön tarvittavat tiedostot, ellei niillä ole erityistä syytä sijaita pohjatasolla (kuten index.php) (Drupal.org 2016).
- Modules-kansioon tallennetaan kaikki ulkopuoliset moduulit, jotka voivat olla joko itsetehtyjä tai Drupal-yhteisön tekemiä. Näillä moduuleilla voidaan laajentaa Drupalin toimintoja entisestään. (Drupal.org 2016).
- Themes-kansioon puolestaan tallennetaan kaikki ulkopuoliset teemat, joilla määritellään Drupalin ulkoasu (Drupal.org 2016).

- Vendor-kansio sisältää ulkoiset kirjastot, jotka Drupalin ydintoiminnot vaativat toimiakseen. Näiden hallintaan Drupalissa käytetään Composer-pakettienhallintaa. Composerin avulla riippuvuudet voidaan asentaa pohjatasolta löytyvien composer.json- ja composer.lock-tiedostojen perusteella. (Drupal.org 2016).
- Sites-kansioon tallennetaan sivustokohtaisia asetuksia ja mm. käyttäjäien luomat tiedostot. Sivustokohtaiset moduulit ja teemat voidaan myös halutessaan siirtää tänne. (Drupal.org 2016)

4.3 Asennus ja käyttöönotto

Drupalin asennus voidaan suorittaa monella tavalla, mutta pääpiirteiltään uuden Drupal-sivuston asennus koostu seuraavista vaiheista:

- Tiedostojen siirto verkkopalvelimelle.
- Riippuvuuksien asentaminen Composer-pakettienhallintatyökalulla.
- Tietokannan luonti.
- Palvelimen konfigurointi.
- Drupalin asennusohjelman suoritus verkkoselaimen kautta.

Asennusohjelman suorituksen jälkeen Drupal-sivusto on valmis käytettäväksi. (Drupal.org 2018c.) Sivustoa voidaan hallinnoida suoraan verkkoselaimen kautta, mutta koska se käy pidemmän päälle työlääksi, on hallintaa varten kehitetty myös komentorivityökaluja, joista kenties suosituin on Drush. Drush on komentorivi ja skriptaus käyttöliittymä Drupaliin, jonka avulla ylläpitotoimintoja voidaan suorittaa antamalla käskyjä komentoriviltä. (Drupal.org 2018a.)

4.3.1 Konfiguraatio koodina

Kuten luvussa 4.1 todettiin, tallentaa Drupal järjestelmän konfiguraation tietokantaan. Tämä mahdollistaa järjestelmän monipuolisen muokkaamisen ja hallinnan ilman lähdekoodin muokkausta, mutta siitä seuraa myös ongelmia käyttöönoton kannalta.

Järjestelmän konfiguraatioksi luetaan esimerkiksi teeman asetukset, sisältöjen asettelut ja valikot. Käyttäjien luoma sisältö, kuten blogikirjoitukset, eivät ole osa konfiguraatiota. Konfigurointi suoritetaan usein verkkoselaimen kautta klikkailemalla valikoita ja asetuksia. Tämä on käyttöönoton kannalta ongelmallista, sillä kun järjestelmää ollaan siirtämässä

esimerkiksi kehitysympäristöstä tuotantoympäristöön, joudutaan konfigurointi suorittamaan myös tuotantoympäristössä. (Taylor 2016.)

Koska konfiguroinnin teko valikkojen kautta moneen kertaan on työlästä, voidaan konfiguraatio viedä tietokannasta koodina, jonka avulla se voidaan siirtää toiseen ympäristöön helposti. Näin konfiguraatio voidaan myös sisällyttää versionhallintaan, jolloin se on paremmin hallittavissa ja kaikkien saatavilla. (Taylor 2016.)

Drupalissa konfiguraation vienti koodiksi on mahdollista suorittaa järjestelmän ydinmoduuleihin kuuluvalla ”Configuration Manager”-moduulilla. Sen avulla konfiguraation tuonti ja vienti voidaan suorittaa esimerkiksi järjestelmän asetuksista selaimen välityksellä tai komentorivityökaluilla. (Drupal.org 2018d.)

4.3.2 Muutostöiden käyttöönottoprosessi

Jos sivusto on jo asennettu, ei muutostöiden yhteydessä enää luoda tietokantaa tai ajeta Drupalin asennusohjelmaa. Uuden ominaisuuden toteutukseen ja käyttöönottoon voi kuulua esimerkiksi seuraavat vaiheet:

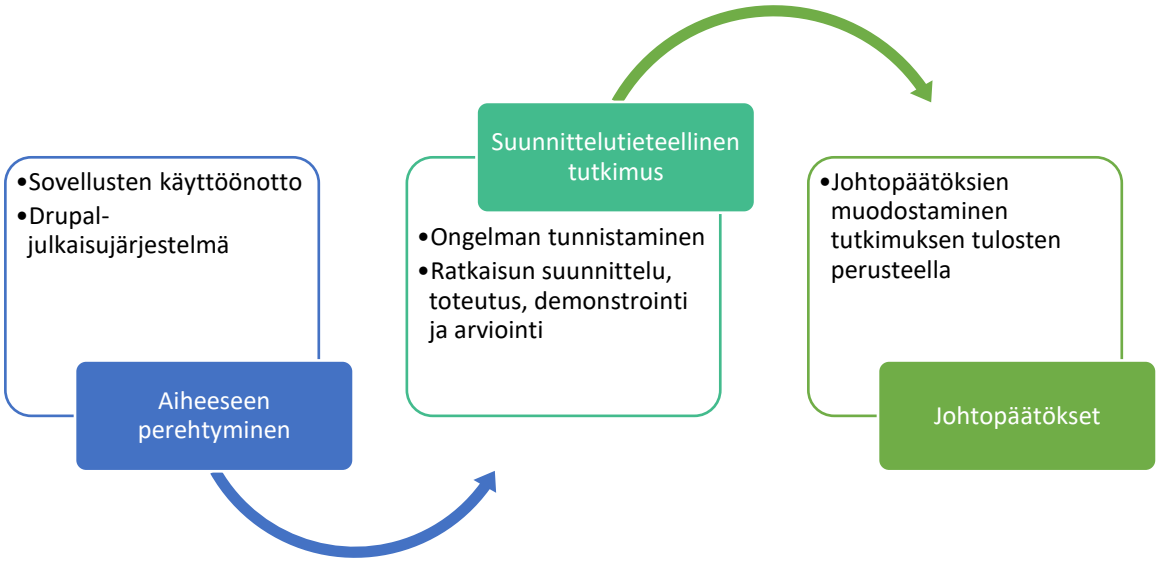
- Konfiguraation ja tiedostojen muokkaus kehitysympäristössä.
- Konfiguraation vienti koodiksi.
- Tiedostojen lähetys versionhallintajärjestelmään.
- Muuttuneiden tiedostojen haku versionhallintajärjestelmästä tuotantoympäristöön.
- Tietokantapäivitysten teko.
- Konfiguraation tuonti koodista.
- Välimuistin tyhjennys.

Tätä prosessia voidaan kehittää automatisoimalla sitä erilaisilla työkaluilla, kuten Drush-komentorivityökalulla, Git-versionhallintajärjestelmällä ja automaatiopalvelimella. Tavoitteena on luoda prosessi, joka on luotettava, nopea ja vähentää ihmisen tekemien virheiden mahdollisuutta. (Lullabot Education 2018.)

5 TUTKIMUKSEN TOTEUTUS

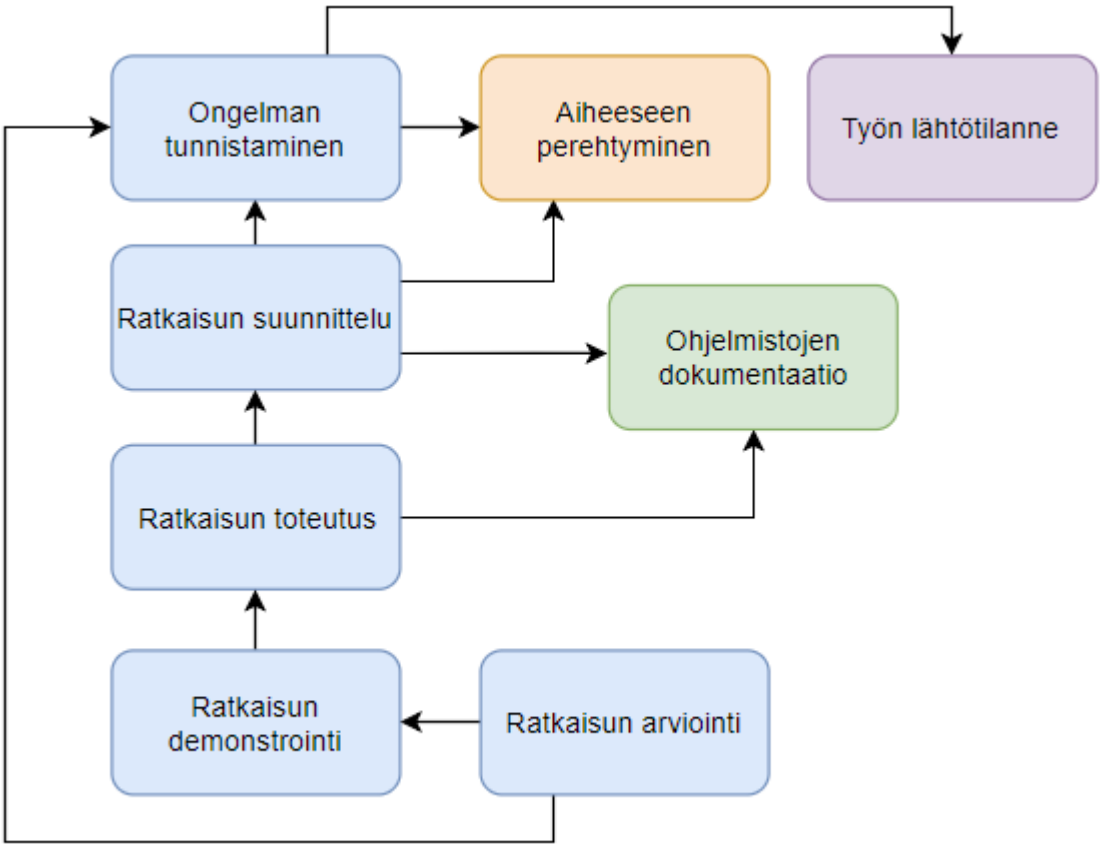
5.1 Työprosessi

Opinnäytetyön toteutus aloitettiin perehtymällä sovellusten käyttöönottoon ja Drupal-julkaisujärjestelmään erilaisten kirjallisten lähteiden kautta (luvut 3 ja 4). Näiden tietojen pohjalta lähdettiin toteuttamaan suunnittelutieteellistä tutkimusmenetelmää luvussa 2.3.2 määritetyllä tavalla ja lopuksi muodostettiin johtopäätöksiä tutkimuksen tulosten perusteella. Prosessin etenemistä on havainnollistettu kuviossa 3.



Kuvio 3. Työprosessi

Suunnittelutieteellisen tutkimuksen aktiviteettien suorittamiseen tarvittiin monenlaista tietoa, jonka käyttöä on kuvattu kuviossa 4, jossa aktiviteetit on kuvattu sinertävissä laatikoissa vasemmalla. Aktiviteeteista lähtevät nuolet osoittavat, mitä tietoa niiden suorittamiseen käytettiin.



Kuvio 4. Tiedon käyttö tutkimuksen toteutuksessa

Työn lähtötilanne perustuu toimeksiantajalta työn aloituksen ja työharjoittelun suorituksen yhteydessä saatuun suulliseen tietoon. Aiheeseen perehdyttiin luvuissa 3 ja 4, joissa tiedon hakuun käytettiin pääasiassa verkkohakuja, jotka suoritettiin erilaisilla aiheeseen liittyvillä avainsanoilla. Tietoa pyrittiin hakemaan monenlaisista lähteistä, kuten kirjoista, raporteista ja verkkokirjoituksista, jotta aiheesta saataisiin kattava näkemys. Ratkaisun suunnitteluun ja toteutukseen käytetyt ohjelmistojen dokumentaatiot haettiin pääosin kyseisten ohjelmistojen verkkosivustoilta. Ratkaisun demonstroinnin yhteydessä tehdyt havainnot dokumentoitiin osaksi tätä opinnäytetyötä ja ratkaisun arviointi suoritettiin vertaamalla näitä havaintoja tunnistettujen ongelmien perusteella määriteltyihin ratkaisun tavoitteisiin.

5.2 Käyttöönottoprosessin lähtötilanne

Luvussa 2.2 määriteltiin lähtötilanteeksi tähän työhön manuaalinen käyttöönottoprosessi, jossa työntekijä suorittaa tarvittavat vaiheet itse ja on siten myös vastuussa lopputuloksesta. Käyttöönottoprosessin vaiheet voivat olla ennalta määritelty, jolloin työntekijän vastuulla on vain seurata prosessia ohjeiden mukaan. On myös mahdollista, ettei työvaiheita

ole määritelty ennakkoon, vaan työntekijällä on tiedossa vain tavoite, eli työn vienti kehitysympäristöstä tuotantoympäristöön. Tämä antaa työntekijälle vapauden suorittaa työ parhaaksi katsomallaan tavalla, mutta samalla se myös lisää vastuuta ja tekee prosessista vaikeasti jäljitettävän ongelmatilanteissa.

Käytännössä manuaalista käyttöönottoprosessia on vaikea määritellä yleispätevästi, koska sen sisältö riippuu mm. kohdeprojektista, tuotantoympäristöstä, käytössä olevista työkaluista, yrityksessä sovituista työmenetelmistä ja työn suorittajasta. Muuttujia on siis paljon ja ne voivat vaihdella yrityksen sisäisestikin projektikohtaisesti.

Jotta tutkimukseen saatiin selkeä lähtökohta, päätettiin lähtötilanteeksi ottaa käyttöönottoprosessi, jota tämän työn kirjoittaja työharjoittelun aikana käytti useissa toimeksiantajayrityksen Drupal-projekteissa. Lähtötilanteena oleva käyttöönottoprosessi koostui siis pääpiirteiltään seuraavista vaiheista:

1. SSH-terminaaliyhteyden ottaminen tuotantoympäristöön.
2. Tuotantoympäristössä olevan projektin varmuuskopiointi.
3. Tuotantoympäristön tietokannan varmuuskopiointi.
4. Tiedostojen siirto tuotantoympäristöön SFTP-ohjelman välityksellä.
5. Riippuvuuksien asennus Composer-komentorivityökalulla.
6. Drupalin tietokantapäivitysten suorittaminen Drush-komentorivityökalulla.
7. Drupalin välimuistin tyhjennys Drush-komentorivityökalulla.
8. Drupalin konfigurointimuutosten teko tuotantoympäristöön.

Käytännössä yllä kuvattu käyttöönottoprosessi on sekoitus määriteltyä ja määrittelemätöntä prosessia, eli siinä suoritetaan tietyt vaiheet, mutta niiden suoritustapa on työntekijän vastuulla.

5.3 Ongelman tunnistaminen ja motivaatio

Kuten luvussa 2.3.1 käytiin läpi, suunnittelutieteellisen tutkimusmenetelmän ensimmäisessä aktiviteetissa määritellään tutkittava ongelma ja perustellaan, miksi sen ratkaiseminen on arvokasta. Luvun 2.3.2 suunnitelmaa seuraten ongelman tunnistamiseen käytettiin luvussa 3.2 kirjallisuudesta kerättyä tietoa automaation hyödyistä, luvussa 2.1 esiteltyjä tutkimuksen taustoja sekä luvussa 5.2 määriteltyä käyttöönottoprosessin lähtötilannetta.

Tutkimuksen taustoja käsittelevässä luvussa 2.1 todettiin, että käsityönä tehtävä käyttöönotto oltiin toimeksiantajayrityksessä havaittu aikaa vieväksi ja virhealttiiksi. Luvussa 3.2 esitellyssä raportissa (Octopus Deploy 2014) manuaalisen käyttöönottoprosessin ongelmiksi todettiin myös hitaus ja virhealttius, joiden lisäksi siinä mainittiin ongelmallisiksi myös seurannan puutteellisuus ja sidonnaisuus avainhenkilöihin.

Hitaudessa ei tutkittavan tapauksen osalta ole kyse pelkästään käyttöönottoprosessiin kuluvasta kokonaisajasta, vaan tärkeää on etenkin aktiivisen työn määrä. Jos työntekijä voi tehdä muita töitä käyttöönottoprosessin pyöriessä taustalla, ei työn tehokkuus kärsi paljoa, vaikka prosessi kestäisikin yhtä kauan kuin lähtötilanteessa.

Virhealttiuden voidaan ajatella käsittävän kaikenlaiset käyttöönottoa suorittavan työntekijän mahdollisesti tekemät virheet. Näitä virheitä on vaikea ennakoida, mutta niihin voivat kuulua esimerkiksi varmuuskopioinnin unohtaminen tai väärän tiedoston poistaminen.

Seurannan puutteellisuus on ongelmallista, sillä kaikkea ei aina muisteta kirjata ylös. Jos tarvitaan tieto esimerkiksi jonkin sivuston senhetkisestä ohjelmistoversiosta, joudutaan se mahdollisesti selvittämään esimerkiksi käyttöönoton suorittaneelta henkilöltä, joka on jo saattanut unohtaa koko asian.

Avainhenkilöihin sidonnaisuuden ongelmallisuutta selvitettiin jo luvussa 3.2, ja sillä siis viitataan tilanteeseen, jossa käyttöönottoprosessin osaa suorittaa vain tietyt henkilöt. Tästä muodostuu ongelma, jos kyseiset henkilöt eivät syystä tai toisesta ole käytettävissä.

Kun näitä ongelmia mietitään luvussa 5.2 määritellyn lähtötilanteen kannalta, voidaan siitä tunnistaa yllä mainituista ongelmista etenkin virhealttius ja seurannan puutteellisuus. Koska käyttöönottoprosessia ei ole tarkoin määritelty, riippuu sen onnistuminen paljolti työntekijän tiedoista ja taidoista. Yrityksen kannalta tämä on hyvin riskialtista, sillä esimerkiksi kokematon työntekijä saattaa vahingossa poistaa tuotantoympäristössä olevan verkkosivuston kokonaan. Virheen tässä tekee työntekijä, mutta vian voidaan myös katsoa olevan prosessissa itsessään, sillä ihmisten tekemiltä virheiltä ei voida koskaan välttyä kokonaan. Tämä on toki ääritapaus, mutta myös pienempien virhetilanteiden ratkaisun kannalta seurannan puutteellisuus on ongelmallista, koska käytännössä on hyvin vaikea selvittää, mitkä toimenpiteet johtivat virheen esiintymiseen.

Yhteistä näille kaikille ongelmille on niiden vaatima asioiden muistaminen ja tiedon hallinta. Oli kyse sitten unohtuvista komentorivikäskyistä, vaikeasti muistettavista salasanoista tai monimutkaisista työvaiheista, vievät ne kaikki osansa työntekijän huomiosta. Koska ohjelmointityö vaatii jo itsessään paljon keskittymistä ja asioiden muistamista, olisi

hyödyllistä löytää keinoja, joilla tätä muistitaakkaa ja vastuuta voidaan siirtää tietokoneiden hallittavaksi.

5.4 Ratkaisun tavoitteiden määrittely

Kuten luvussa 2.3.1 todettiin, suunnittelutieteellisessä tutkimuksessa ratkaisun tavoitteet määritellään tutkittavan ongelman perusteella. Tässä opinnäytetyössä tavoitteet toteutettavalle ratkaisulle määriteltiin luvussa 5.3 määriteltyjen ongelmakohtien perusteella ja ne näkyvät taulukossa 2.

Taulukko 2. Ratkaisun tavoitteet

Määritelty ongelma	**Ratkaisun tavoite**
Hitaus.	Ei vaadi tekijältä aktiivista osallistumista.
Virhealttius.	Toteuttaa määritetyt työvaiheet toistettavasti ja luotettavasti.
Seurannan puute.	Pitää kirjaa käyttöönottoprosessin suorittamisesta ja lopputuloksista.
Avainhenkilöihin sidonnaisuus.	Ei vaadi tekijältä tapauskohtaista tietoa.

Saavuttamalla nämä tavoitteet saadaan todetut ongelmakohdat ainakin osittain ratkaistua ja toteutettu ratkaisu voidaan katsoa onnistuneeksi. Käyttöönottoprosessin automaatiota voitaisiin viedä paljon pidemmällekin, kuten automaation tasoja käsitelevästä luvusta 3.3 selviää. Luvussa 3.1 kuitenkin huomioitiin, että ohjelmiston käyttöönottoon sisältyy käytettyjen työkalujen lisäksi mm. henkilöstön koulutusta ja ohjeistusta, joten ei välttämättä ole järkevää tai edes mahdollista siirtyä suoraan manuaalisesta prosessista pitkälle automatisoituun prosessiin. Tämän perusteella tässä työssä päätettiin lähteä kehittämään prosessia pienin askelin ratkaisemalla vain luvussa 5.3 määritellyt ongelmakohdat.

5.5 Ratkaisun suunnittelu

Ratkaisun suunnittelussa käytetään perustana luvussa 3.5 esiteltyjä käyttöönottoprosessin suunnittelun periaatteita, eli suunnittelu aloitetaan pohtimalla kohdeprojektien laajuutta. Tämän jälkeen listataan vaiheet, jotka halutaan suorittaa ja pohditaan mitä ongelmia niistä voi seurata sekä kuinka ne ratkaistaan.

Näiden pohdintojen ja määriteltyjen ratkaisun tavoitteiden perusteella muodostetaan suunnitelma toteutettavasta ratkaisusta. Lopuksi valitaan vielä sopivat työkalut suunnitelman toteuttamiseen.

5.5.1 Kohdeprojektin laajuus

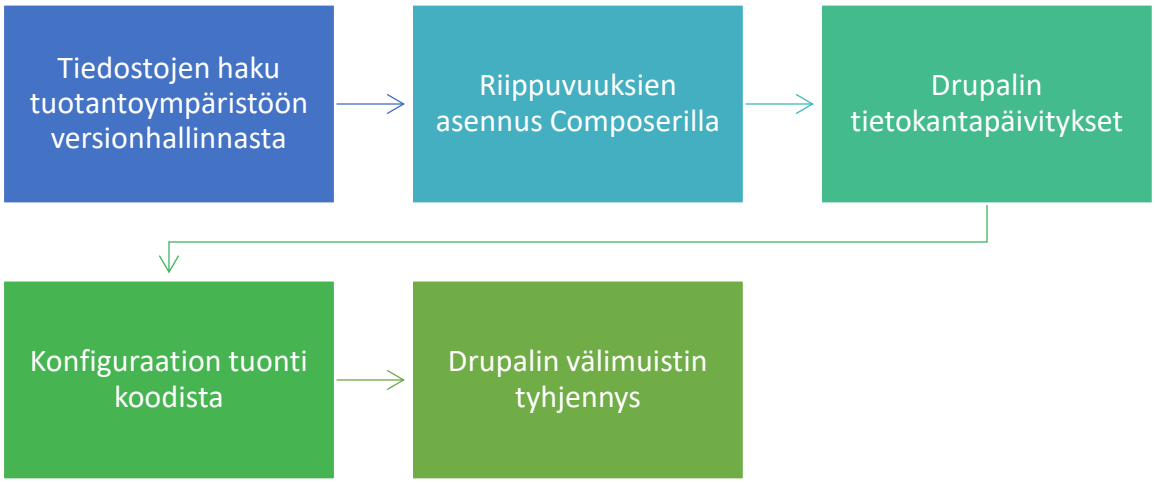
Kohdeprojektin laajuutta on vaikea määritellä yleisesti, koska se on aina hyvin tapauskohtaista. Koska tässä tutkimuksessa kohdeprojektina ei ole oikeasti käytössä olevaa Drupal-sivustoa, vaan pelkkä Drupalin perusasennus, voisi kohdeprojektin määritellä laajuudeltaan pieneksi.

Koska pieneksi määritelty projekti ei kuitenkaan antaisi todellista kuvaa esimerkiksi Drupalilla toteutetun yrityssivuston laajuudesta, päätettiin kohdeprojektit määritellä suuremmiksi. Yrityssivustoilla voi olla paljonkin käyttäjiä ja sivustojen luotettava toiminta voi olla tärkeää yritysten liiketoiminnalle. Lisäksi Drupalia itsessään voidaan pitää jo melko laajana järjestelmänä, sillä se koostuu yli 1,8 miljoonasta koodirivistä (Black Duck Software, Inc. 2018). Kuten luvussa 3.5 todettiin, etenkin suuremmissa projekteissa käyttöönottoprosessi tulisi suunnitella huolellisesti ja on syytä miettiä etukäteen, kuinka mahdolliset ongelmatilanteet ratkaistaan.

5.5.2 Suoritettavat vaiheet

Luvussa 4.3.2 käytiin läpi esimerkkiä Drupalin käyttöönottoprosessista ja todettiin, ettei Drupalin asennusta suoriteta uudestaan muutostöiden yhteydessä. Koska asennus ei ole usein toistuva työvaihe, eikä näin ollen hyödy niin paljoa automatisoinnista, päätettiin se jättää pois toteutettavasta ratkaisusta.

Kuten tutkimuskysymyksen esittelyssä luvussa 2.2 todettiin, on käyttöönottoprosessi rajattu käsittelemään niitä vaiheita, joiden avulla kehitysympäristössä tehty työ viedään tuotantoympäristöön. Käyttöönottoprosessi voidaan siis luvun 4.3.2 perusteella toteuttaa suorittamalla kuviossa 5 kuvatut työvaiheet.



Kuvio 5. Ratkaisussa suoritettavat työvaiheet

Näitä vaiheita ennen täytyy myös suorittaa erilaisia kehitystyön vaiheita, kuten konfiguraation vienti koodiksi ja tiedostojen lähetys versionhallintajärjestelmään. Vaikka nämä vaiheet ovat yhteydessä käyttöönottoprosessiin, kuuluvat ne enemmän kehitystyöhön, eikä niitä siksi sisällytetä toteutettavaan ratkaisuun.

5.5.3 Ongelmatilanteisiin varautuminen

Kuten luvussa 3.5 todettiin, käyttöönottoprosessia suorittaessa voi tulla vastaan monenlaisia ongelmia. Kaikkia ongelmia ei voida ennustaa, ja siksi niihin on aiheellista varautua jo etukäteen. Luvuissa 4.1 ja 4.2 käytiin läpi Drupalin perusteita ja toimintaa, ja todettiin Drupalin olevan PHP-ohjelmointikielellä toteutettu verkkosovellus. Sen toiminta perustuu tiedostoihin määritettyyn logiikkaan (lähdekoodi) ja tietokannassa sijaitsevaan dataan. Tämän perusteella voidaan toteutettavassa ratkaisussa varautua ainakin kahdentyyppisiin ongelmiin:

- Tiedostoihin liittyvät ongelmat.
- Tietokantaan liittyvät ongelmat.

Tiedostojen siirto voi syystä tai toisesta keskeytyä, jolloin sovellus ei välttämättä enää toimi, koska osa tiedostoista on uusia ja osa vanhoja. Voi myös esiintyä tilanteita, joissa kehitysympäristössä tehdyt muutokset eivät toimi samalla tavalla tuotantoympäristössä tai niitä ei ole testattu tarpeeksi hyvin. Kuten luvussa 3.4 todetaan, voidaan tiedostoihin liittyvistä ongelmista palautua tuomalla versionhallintajärjestelmästä aiempi versio tiedostoista.

Myös tietokantaan tehtävät päivitykset ja muutokset voivat epäonnistua ja estää sovelluksen normaalin toiminnan. Koska tietokantaa ei ole käytännöllistä sisällyttää versionhallintajärjestelmään tiedostojen kanssa, täytyy siitä tehdä varmuuskopio ennen muutoksien tekoa. Tietokantamuutoksia ei välttämättä tehdä jokaisen käyttöönoton yhteydessä, mutta selkein tapa varautua ongelmiin on ottaa varmuuskopiot aina. Tietokantaa ei kuitenkaan välttämättä haluta palauttaa automaattisesti virhetilanteissa, koska virheitä ei aina huomata heti. Jos sivustoa on esimerkiksi käytetty päiviä tai viikkoja ennen vian huomaamista, on sinne mahdollisesti ehditty luomaan uutta sisältöä, joka menetettäisiin, jos tietokanta palautettaisiin varmuuskopiosta. Tietokannan palauttaminen vaatii siis usein tapauskohtaista harkintaa ja siksi sen automatisointi on haasteellista.

5.5.4 Suunnitelman muodostus

Kuten luvussa 2.3.2 määriteltiin, tavoitteena on suunnitella automaatiota hyödyntävä käyttöönottoprosessi, joka ratkaisee luvussa 5.4 asetetut tavoitteet mahdollisimman hyvin. Automatisoimalla luvussa 5.5.2 määritetyt käyttöönottoprosessin vaiheet voidaan todennäköisesti ratkaista ainakin hitaus, virhealttius ja avainhenkilöihin sidonnaisuus. Seurannan puutteellisuus voidaan puolestaan ratkaista käyttämällä prosessin hallintaan automaatiopalvelinta, josta kerrottiin luvussa 3.4.

Ennen luvussa 5.5.2 määriteltyjä käyttöönottoprosessin vaiheita tulisi myös suorittaa ongelmatilanteiden varalta tietokannan varmuuskopiointi. Tietokannan palautusta ei kuitenkaan automatisoida osaksi prosessia luvussa 5.5.3 mainituista syistä. Tiedostojen osalta ongelmatilanteisiin varaudutaan käyttämällä versionhallintajärjestelmää.

Versionhallintajärjestelmistä löytyy luvussa 3.4 esitelty ominaisuus, jonka avulla versionhallintaan voidaan luoda päätyöhaarasta eriäviä työhaaroja. Käyttöönottoprosessin suorittamisen ja hallinnan selkeyttämiseksi näitä työhaaroja hyödynnetään niin, että vain muutokset päätyöhaaraan viedään automaattisesti tuotantoympäristöön. Näin kehitystyötä varten voidaan perustaa erillisiä työhaaroja, joiden käyttöönotto tapahtuu yhdistämällä työhaara takaisin päätyöhaaraan. Tämän menetelmän avulla käyttöönoton voi suorittaa kuka tahansa, kenellä on oikeus yhdistää kehitystyöhaara päätyöhaaraan.

Käyttöönottoprosessin hallintaan käytettävä automaatiopalvelin määritellään seuraamaan muutoksia versionhallintajärjestelmässä sijaitsevan projektin päätyöhaarassa. Muutoksien tapahtuessa automaatiopalvelin suoritta sille määritetyt toimenpiteet, eli tässä tapauksessa käynnistää käyttöönottoprosessin suorituksen. Käyttöönottoprosessin vaiheet itsessään suoritetaan erillisellä käyttöönottotyökalulla, jolloin ne voidaan määritellä helpommin

ja niiden toiminta on mahdollisesti luotettavampaa. Kokonaisuudessaan käyttöönottoprosessin tulisi tapahtua siis seuraavasti:

1. Versionhallintajärjestelmän päätyöhaaraan lähetetään muutos.
2. Automaatiopalvelin saa tiedon muutoksesta.
3. Automaatiopalvelin käynnistää käyttöönottotyökaluun määritetyn prosessin.
4. Käyttöönottotyökalu suorittaa luvussa 5.5.2 määritetyt vaiheet, joita ennen suoritetaan myös tietokannan varmuuskopiointi ongelmatilanteiden varalta.

5.5.5 Työkalujen valinta

Käyttöönottoprosessin toteutuksessa voidaan käyttää avuksi monentyyppisiä työkaluja, joita käytiin läpi luvussa 3.4. Koska työkaluja on hyvin paljon ja monilla niistä voidaan päästä samankaltaiseen lopputulokseen, ei yksittäisten työkalujen valinnan perustelu ole kovin oleellista tämän tutkimuksen kannalta, vaan tärkeämpää on niiden tuomat ominaisuudet ja mahdollisuudet.

Versionhallintaan käytetään Git-pohjaista Atlassian yrityksen tarjoamaa Bitbucket-palvelua. Yhtä hyvin voitaisiin käyttää jotain muutakin ratkaisua, oleellista tämän osalta on, että järjestelmään voidaan ottaa yhteys internetin välityksellä.

Automaatiopalvelimena käytetään Jenkinsiä, joka on ohjelmiston verkkosivuilta löytyvän dokumentaation (Jenkins 2018a) mukaan avoimen lähdekoodin automaatiopalvelin, jota voidaan käyttää monenlaisten tehtävien automatisointiin.

Käyttöönottoprosessin vaiheiden automatisointiin käytetään avoimen lähdekoodin Ansible-komentorivityökalua. Ansiblella voidaan automatisoida mm. verkkopalvelimen määrittely tai ohjelmiston käyttöönottoprosessi. Yksinkertaisuus, helppokäyttöisyys, tietoturva ja luotettavuus kuuluvat Ansiblen periaatteisiin. (Red Hat, Inc., 2017.)

5.6 Ratkaisun toteutus

Ratkaisun käytännön toteutus tehtiin luvussa 5.5 luodun suunnitelman perusteella ja avuksi käytettiin myös verkosta löytyviä ohjeistuksia ja ohjelmistojen dokumentaatioita. Toteutus aloitettiin pystyttämällä kehitys- ja tuotantoympäristöt ja samalla myös luotiin tutkimuksen kohdeprojektina toimiva Drupal-projekti, jotta ratkaisua voitaisiin helpommin kehittää. Tämän jälkeen asennettiin käyttöönottoprosessin vaiheiden automatisointiin käytetty Ansible ja määriteltiin sillä suoritettavat työvaiheet. Lopuksi asennettiin ja määriteltiin

vielä Jenkins-automaatiopalvelin, joka sitoi käytetyt työkalut toisiinsa ja toimi ratkaisun keskitettynä hallintapisteenä.

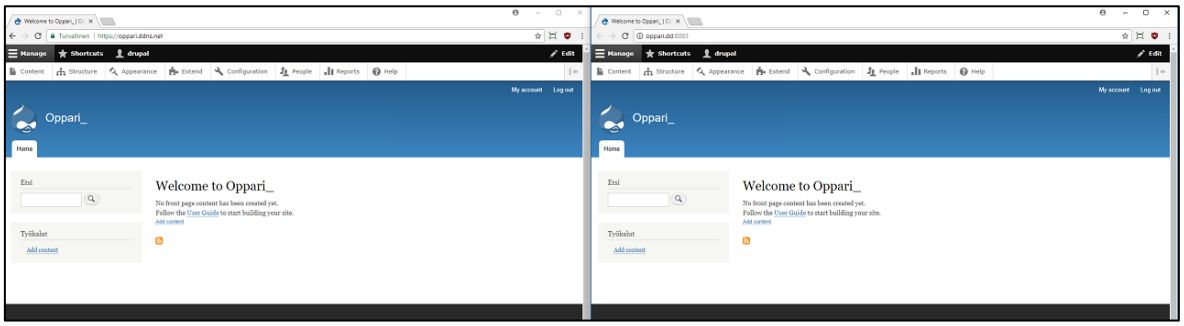
5.6.1 Ympäristöjen pystytys ja kohdeprojekti

Käyttöönottoprosessin toteutusta ja arviointia varten aluksi pystytettiin kehitys- ja tuotantoympäristöt. Kehitysympäristönä käytettiin Acquia Dev Desktop ohjelmistopakettia, joka on tehty Drupal-sivustojen paikalliseen kehittämiseen. Se pitää sisällään mm. Apache-verkkopalvelimen, MySQL-tietokannan ja useita PHP-versioita (Acquia 2018).

Jotta tuotantoympäristöstä saatiin todenmukaisempi, päätettiin käyttää pilvipalvelussa toimivaa virtuaalipalvelinta. Palvelimelle asennettiin LAMP-ohjelmistopino (Linux, Apache, PHP, MySQL), joka on hyvin yleinen Drupal-sivustojen ylläpitoon käytettävillä palvelimilla. Palvelinohjelmistojen asennukseen ja määrittelyyn käytettiin Ansible-komentorivityökalua, mutta koska prosessi on hyvin pitkä ja monimutkainen, ei sitä käsitellä sen tarkemmin tässä. Lukijalle voi kuitenkin olla mielenkiintoista tietää, että työkalulla voidaan suorittaa myös palvelinten määrittely kokonaisuudessaan. Lisäksi määritettiin verkko-osoite ”oppari.ddns.net” osoittamaan palvelimen IP-osoitteeseen yhteyden ottamisen helpottamiseksi.

Tutkimuksessa käytettävän Drupal-projektin pohjana käytettiin Drupal.org-sivustolta ladattua puhdasta Drupal 8.5 versiota. Projektikansioon luotiin Git-versionhallinta varasto (engl. repository), johon projektin tiedostot sisällytettiin. Varasto lähetettiin Bitbucket-palveluun, josta se haettiin myös tuotantoympäristöön.

Lopuksi suoritettiin vielä Drupal-projektin alustava asennus kehitysympäristöön, josta se vietiin myös tuotantoympäristöön. Tässä vaiheessa oltiin siis tilanteessa, jossa sama Drupal-sivusto oli saatavilla kehitys- ja tuotantoympäristöissä (kuva 3).



Kuva 3. Drupal-sivustot kehitys- ja tuotantoympäristöissä

5.6.2 Ansiblen asennus

Käyttöönottoprosessin toteutus tapahtui Windows 10-käyttöjärjestelmää käyttävällä tietokoneella. Koska Ansiblesta ei ole Windows-versiota, käyttöjärjestelmästä otettiin käyttöön Windows Subsystem for Linux-ominaisuus Windowsin dokumentaatiosta (Microsoft 2017) löytyvien ohjeiden mukaan. Tämän ominaisuuden avulla on mahdollista ajaa Linux-pohjaisia sovelluksia Windowsissa.

Ansiblen asennus suoritettiin ohjelmiston dokumentaatiosta (Red Hat, Inc. 2018b) löytyvien ohjeiden mukaan Ubuntu Linuxiin (Windows Subsystem) suorittamalla seuraavat käskyt komentoriviltä:

```
sudo apt-get update

sudo apt-get install software-properties-common

sudo apt-add-repository ppa:ansible/ansible

sudo apt-get update

sudo apt-get install ansible
```

Käskyjen suorittamisen jälkeen Ansible todettiin toimintavalmiiksi suorittamalla komentoriviltä käsky:

```
ansible --version
```

Tämä käsky tulostaa mm. käytössä olevan ohjelmistoversion (kuva 4).

```
jaakko@DESKTOP-NGEPNS9:/mnt/c/Users/Jaakko/projects/oppari/deploy-drupal$ ansible --version
ansible 2.5.0
  config file = /mnt/c/Users/Jaakko/projects/oppari/deploy-drupal/ansible.cfg
  configured module search path = [u'/home/jaakko/.ansible/plugins/modules', u'/usr/share/ansible/plugins/modules']
  ansible python module location = /usr/lib/python2.7/dist-packages/ansible
  executable location = /usr/bin/ansible
  python version = 2.7.12 (default, Dec  4 2017, 14:50:18) [GCC 5.4.0 20160609]
```

Kuva 4. Ansiblen versio

5.6.3 Tuotantoympäristöön yhdistäminen Ansiblella

Jotta Ansible voisi suorittaa sille määriteltyjä tehtäviä kohdekoneella, täytyy sinne olla jonkinlainen yhteys. Yhteyden luontiin Ansible käyttää SSH-yhteyttä ja oletuksena on, että yhteyden turvaamiseen käytetään SSH-avaimia. (Red Hat, Inc. 2018a.) SSH avainparit koostuvat julkisesta ja yksityisestä avaimesta. Yksityinen avain säilytetään asiakasko-

neella ja julkinen avain sijoitetaan kohdekoneelle, jonka jälkeen sinne voidaan ottaa yhteys ilman salasanoja yksityisen avaimen avulla. SSH-avaimen luonti voidaan suorittaa antamalla komentoriviltä käsky:

```
ssh-keygen
```

Käskyn suorittamisen jälkeen vastataan ohjelman esittämiin kysymyksiin, joiden perusteella ohjelma luo SSH-avainparin. (Ellingwood, 2014.)

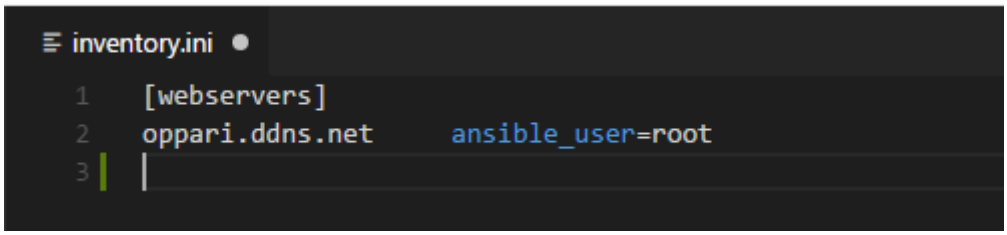
Käyttöönottoprosessin automatisointi aloitettiin siis luomalla SSH-avainpari. Yksityinen avain jätettiin koneelle, johon Ansible asennettiin ja julkinen avain sijoitettiin tuotantoympäristöön. Yhteyden toiminta varmistettiin vielä antamalla komentoriviltä käsky:

```
ssh root@oppari.ddns.net
```

Tämä käsky siis avaa SSH-yhteyden kohdekoneeseen pääkäyttäjälle (root).

Ansiblella voidaan ajaa sille määriteltyjä tehtäviä useissa kohteissa. Kohteet valitaan Ansiblen inventaariosta. Inventaario voidaan määritellä oletustiedostoon, tai luomalla oma tiedosto ja antamalla komentorivillä parametri ”`-i <tiedostopolku>`”. Inventaariotiedostoon määritellään kohdekoneet ja se voidaan luoda käyttäen esimerkiksi INI- tai YAML-tiedostomuotoa. (Red Hat, Inc. 2018c.)

Tässä ratkaisussa päätettiin käyttää erillistä inventaariotiedostoa, joka luotiin INI-tiedostomuodossa. Käytännössä tiedoston sisältö on hyvin yksinkertainen, sillä käytössä on vain yksi kohdekone. Tiedostoon voidaan määritellä hakasulkuihin kohderyhmiä, joiden alle listataan riveittäin kohdekoneiden osoitteet ja mahdolliset muuttujat. Toteutuksessa käytetyn inventaariotiedoston sisältö näkyy kuvassa 5.



Kuva 5. Ansiblen inventaario

Tässä vaiheessa Ansiblen pitäisi olla valmis ottamaan yhteys kohdekoneeseen ja toiminta varmistettiin antamalla komentorivillä käsky:

```
ansible -i inventory.ini webservers -m ping
```

Tämä käsky suorittaa koeyhteyden inventaariossa määritetyille kohteille. Yhteys todettiin toimivaksi ja käskyn tulos näkyy kuvassa 6.

```
jaakko@DESKTOP-NGEPNS9:/mnt/c/Users/Jaakko/projects/oppari/deploy-drupal$ ansible -i inventory.ini webservers -m ping
ansible -i inventory.ini webservers -m pingoppari.ddns.net | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
```

Kuva 6. Ansible yhteyden kokeilu ping-ohjelmalla

5.6.4 Käyttöönottoprosessin automatisointi Ansiblella

Ansiblella voidaan suorittaa yksinkertaisia tehtäviä suoraan komentoriviltä, mutta monimutkaisemmat kokonaisuudet voidaan määrittää myös erillisissä tiedostoissa. Näitä määrittelytiedostoja kutsutaan Ansiblessa pelikirjoiksi (engl. playbook). Pelikirjoissa käytettävä kieli on selkokielistä ja se on suunniteltu ihmisten luettavaksi. (Red Hat, Inc. 2018d.) Pelikirjat kirjoitetaan YAML-tiedostomuodossa (Red Hat, Inc. 2018e).

Pelikirjoihin voidaan koota useita tehtäviä, jotka voidaan ajaa yhdellä komennolla. Ansible käyttää tehtävien suorittamiseen moduuleja, joiden avulla voidaan esimerkiksi asentaa ohjelmia tai kopioida tiedostoja. (Fideloper LLC 2018.) Saatavilla olevat moduulit ja ohjeet niiden käyttöön löytyvät Ansiblen dokumentaatiosta (Red Hat, Inc. 2018f),

Pelikirjoilla voidaan rakentaa hyvin monimutkaisiakin kokonaisuuksia, mutta tässä työssä toteutettava ratkaisu koostuu suhteellisen yksinkertaisista tehtävistä. Ensimmäisenä suoritettava vaiheen, eli tietokannan varmuuskopioinnin, määrittely voidaan pelikirjassa tehdä kuvassa 7 näkyvällä tavalla.

```
! deploy.yml ×
1   - hosts: "{{ hostname }}"
2
3     remote_user: root
4
5     vars_files:
6       - vars/main.yml
7
8     tasks:
9       - name: Database backup
10        mysql_db:
11          state: dump
12          name: "{{ db_name }}"
13          target: ~/drupal-db-backups/{{ hostname }}-{{ ansible_date_time.iso8601 }}.sql
14          login_user: "{{ db_user }}"
15          login_password: "{{ db_password }}"
```

Kuva 7. Vaihe 1: Tietokannan varmuuskopiointi

Kuvassa 7 näkyy osa toteutettuun ratkaisuun käytetyn pelikirjan sisällöstä. Pääkohdat tiedostossa ovat:

- **`hosts`**: Määrittää kohdekoneet, eli tässä viitataan inventaariosta löytyviin kohteisiin. Aaltosulkujen sisällä olevat teksti ”`hostname`” on muuttuja, eli se viittaa muualla määritettyyn sisältöön.
- **`vars_files`**: Tässä kohdassa voidaan määritellä erillisiä tiedostoja muuttujia varten. Esimerkki tässä määritellyn main.yml-tiedoston sisällöstä on tämän tutkimuksen liitteenä (LIITE 1).
- **`tasks`**: Tämän alle voidaan määrittää suoritettavat tehtävät. Tehtävälle annetaan nimi ja määritellään käytettävä moduuli sekä moduulille annettavat parametrit.

Tietokannan varmuuskopiontiin käytetään tässä tapauksessa mysql_db-moduulia, jolle on annettu seuraavat parametrit:

- **`state:`** Määrittää mikä tila tietokannassa halutaan saavuttaa, ”`dump`”-tilalla viitataan tässä tietokannan vientiin.
- **`name:`** Määrittää mihin tietokantaan suoritettava tehtävä kohdistetaan.
- **`target:`** Määrittää kohdetiedoston polun (kohdekoneella).
- **`login_user:`** Tietokannan käyttäjänimi.
- **`login_password:`** Tietokannan salasana.

Seuraavaksi vaiheeksi suunnitelmassa määriteltiin tiedostojen nouto versionhallintajärjestelmästä tuotantoympäristöön. Tämä tehtävä voidaan pelikirjassa määritellä seuraavasti (kuva 8):

```
22    - name: Pull from Git
23      git:
24        repo: "{{ repo_address }}"
25        dest: "{{ project_root }}"
26        accept_hostkey: yes
27        force: yes
```

Kuva 8. Vaihe 2: tiedostojen nouto versionhallinnasta

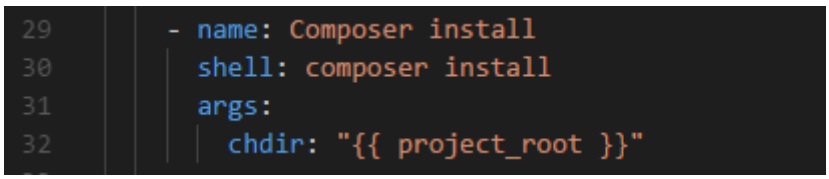
Tässä tehtävässä käytetään siis git-moduulia, joka hakee muuttujalla määritetyn versionhallinnan varaston ja sijoittaa sen muuttujalla määritettyyn sijaintiin kohdekoneella. Parametri ”`force`” suorittaa tehtävän, vaikka kohteessa havaittaisiin paikallisia muutoksia. Tässä on myös syytä huomioida, että kohdekoneella on oltava käyttöoikeus versionhallintajärjestelmään. Tässä tutkimuksessa käyttäjän todentamista varten luotiin tuotantoympä-

ristöön SSH-avainpari, jonka avulla annettiin tälle koneelle lukuoikeus käytettyyn version-hallinnan varastoon. Bitbucket-palvelussa tämä voidaan suorittaa syöttämällä julkinen SSH-avain verkkosivujen hallintapaneelin kautta kyseisen varaston asetuksiin.

Kolmantena vaiheena oli riippuvuuksien asentaminen Composerin avulla, joka voidaan suorittaa komentoriviltä käskyllä:

```
composer install
```

Ansiblessa tähän voidaan käyttää shell-moduulia, jonka avulla voidaan suorittaa komentorivikäskyjä. Alla olevassa kuvassa (kuva 9) näkyy, miten tämä voidaan määritellä pelikirjaan:



Kuva 9. Vaihe 3: Composer pakettien asennus

Shell-moduulille annetaan arvoksi suoritettava käsky ja tarvittavat parametrit. Tässä tapauksessa tarvitaan vain yksi parametri, eli `”chdir”,` jonka avulla voidaan siirtyä määrättyyn kansioon ennen käskyn suorittamista.

Loputkin käyttöönottoprosessin vaiheet määriteltiin vastaavalla tavalla shell-moduulin avulla. Koska tehtävämäärittelyt niissä ovat hyvin samankaltaisia, ei niitä ole tarpeellista käydä yksitellen läpi. Ratkaisussa käytetty pelikirjatiedosto on kokonaisuudessaan tämän opinnäytetyön liitteenä (LIITE 2).

Pelikirjan määrittelyn valmistuttua käyttöönottoprosessin ydin oli jo käyttövalmis ja se olisi mahdollista suorittaa suoraan komentoriviltä käskyllä:

```
ansible-playbook -i inventory.ini deploy.yml
```

Tällä käskyllä Ansible siis suorittaa deploy.yml-tiedostoon määritellyn pelikirjan ja käyttää inventaariona inventory.ini-tiedostoa.

5.6.5 Jenkinsin asennus

Toisin kuin Ansible, Jenkins on mahdollista asentaa myös Windowsiin. Tässä tutkimuksessa Jenkins päätettiin kuitenkin asentaa Ansiblen tavoin Linux ympäristöön. Asennus suoritettiin Jenkinsin verkkosivuilta löytyvän dokumentaation (Jenkins 2018b) mukaan suorittamalla alla olevat käskyt komentoriviltä:

```
wget -q -O - https://pkg.jenkins.io/debian/jenkins.io.key |
sudo apt-key add –

sudo sh -c 'echo deb http://pkg.jenkins.io/debian-stable
binary/ > /etc/apt/sources.list.d/jenkins.list'

sudo apt-get update

sudo apt-get install jenkins
```

Käskyjen suorittamisen jälkeen Jenkins käynnistettiin suorittamalla komentoriviltä käsky:

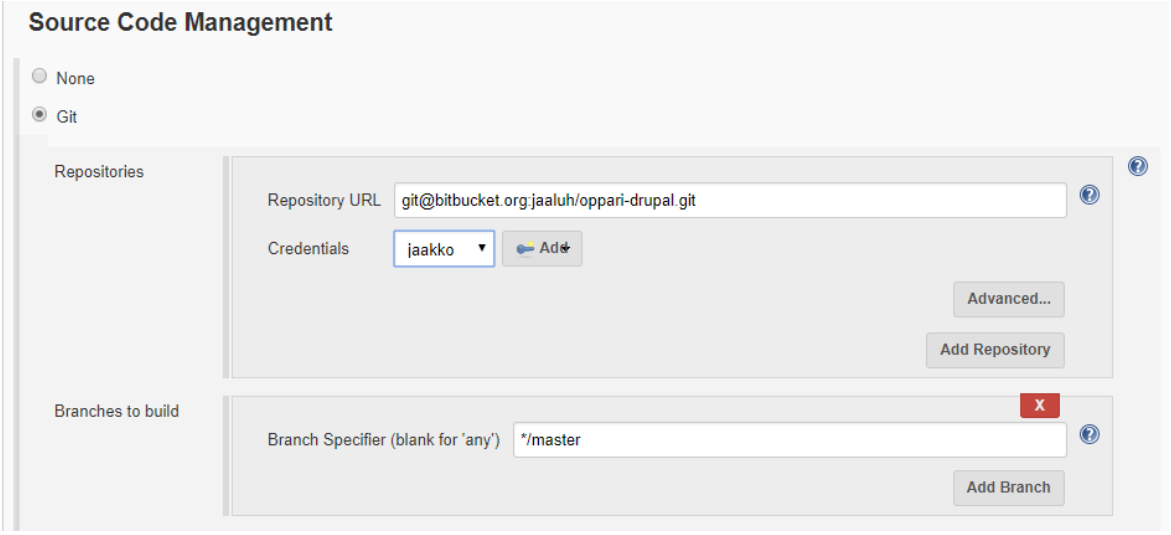
```
sudo service jenkins start
```

Jenkinsin käynnistyttyä asennus viimeisteltiin verkkoselaimen kautta (oletuksena osoite on http://localhost:8080) käyttöliittymän ohjeistuksen mukaan. Lopuksi asennettiin vielä Ansible ja Bitbucket liitännäiset käyttöliittymän liitännäisvalikon kautta, joilla Jenkins voidaan helposti yhdistää kyseisiin ohjelmistoihin/palveluihin.

5.6.6 Jenkinsin määrittely

Jenkinsiin voidaan tallentaa erilaisia tilitietoja kuten käyttäjätunnuksia, salasanoja ja SSH-avaimia. Tallennettuja tilitietoja voidaan käyttää kaikkialla Jenkinsin asetuksissa, jos ne määritellään globaaleiksi. (Jenkins 2018c.) Koska Bitbucket ja Ansible liitännäiset molemmat käyttävät SSH-avainta tunnistautumiseen, tallennettiin luvussa 5.6.3 luotu yksityinen SSH-avain Jenkinsin tilitietoihin.

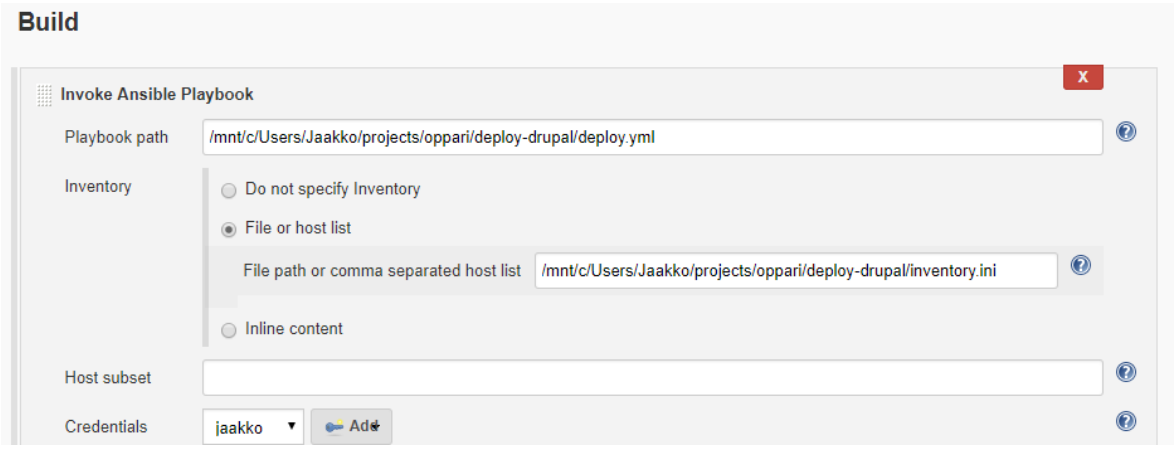
Tämän jälkeen luotiin uusi Jenkins työ, jolle annettiin nimeksi ”Oppari” ja sen tyypiksi valittiin vapaatyylinen projekti. Työn asetuksista valittiin versionhallintajärjestelmäksi Git, jolle määritettiin varaston osoite ja tunnistautumiseen käytettävä tilitieto. Seurattaviin työhaaroihin valittiin vain varaston päätyöhaara (kuva 10).



Kuva 10. Versionhallinnan määrittely Jenkins työlle

Versionhallintaan liittyen työn asetuksista valittiin myös Bitbucket-liitännäisen tarjoama valinta, joka laukaisee työn suorittamisen, kun yllä määritetyn varaston työhaarassa tapahtuu muutoksia. Jotta tämä toimisi, täytyy Jenkins-palvelimeen olla pääsy internetin välityksellä ja lisäksi Bitbucket-palveluun on määritettävä kytkentä, joka lähettää tiedon Jenkinsiin muutoksien tapahtuessa. (Belzunce 2018.)

Viimeisenä työn asetuksista lisättiin vaihe, jossa suoritetaan luvussa 5.6.4 määritelty Ansible pelikirja. Suoritettava pelikirja ja inventaariotiedosto valittiin syöttämällä niiden tiedostopolut. Myös tälle vaiheelle valittiin käytettäväksi tilitiedoksi aiemmin tallennettu SSH-avain. Pelikirjan määrittely Jenkins työlle näkyy kuvassa 11.



Kuva 11. Ansible pelikirjan määritys Jenkins työlle

Työn tallentamisen jälkeen oli Jenkinsin määrittely ja käyttöönottoprosessi kokonaisuudessaan valmis.

5.7 Ratkaisun demonstrointi

Luvussa 2.3.2 määriteltyä suunnitelmaa tutkimusmenetelmän soveltamisesta seuraten tuotosten kyvykkyyttä demonstroitiin ratkaisemalla tutkittavan ongelman ilmentymä suorittamalla kolme muutostyötä luvussa 5.6.1 luotuun Drupal-projektiin. Kuten luvussa 2.3.2 määriteltiin, tehdyistä muutostöistä kerättiin havainnoimalla aineistoa, jotta ratkaisun toimivuutta pystyttiin arvioimaan myöhemmin luvussa 5.8. Havainnoissa keskityttiin etenkin luvussa 5.4 määriteltyihin ratkaisun tavoitteisiin, mutta myös muut huomiot kirjattiin ylös.

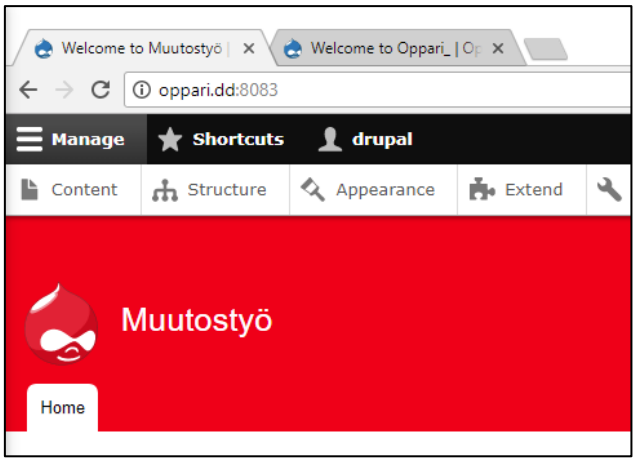
Ensimmäisenä suoritettiin hyvin yksinkertainen kehitystyö, jonka tarkoituksena oli todistaa ratkaisun kyky viedä kehitysympäristössä tehty työ tuotantoympäristöön. Tämän jälkeen suoritettiin virheen sisältävä muutostyö, joka siis sisälsi tahallisen virheen sivuston lähdekoodissa. Lopuksi ratkaisun kyky palautua virhetilanteesta osoitettiin palauttamalla sivusto virhettä edeltävään tilaan.

Tämän luvun aliluvut etenevät niin, että ensin kuvaillaan, kuinka muutostyö tehtiin ja sitä seuraavassa luvussa on kirjattu tämän työn kirjoittajan näkemyksen mukaan tärkeimmät havainnot käyttöönottoprosessin toiminnasta kyseisessä muutostyössä.

5.7.1 Kehitystyön suoritus

Lähtötilanne tässä vaiheessa kohdeprojektin osalta oli sama, kuin luvun 5.6.1 lopussa, eli kehitys- ja tuotantoympäristöissä oli täysin samanlaiset Drupal-sivustot. Kehitystöihin voi sisältyä ainakin kahdenlaisia toimenpiteitä, eli muutoksia voidaan tehdä Drupalin lähdekoodiin tai konfiguraatioon. Lähdekoodin muutoksia mallinnetaan tässä muuttamalla sivuston taustaväri punaiseksi muokkaamalla aktiivisena olevan teeman tyylitiedostoa. Konfiguraatiomuutoksia puolestaan mallinnetaan muuttamalla sivuston nimeksi ”Muutostyö” Drupalin hallintapaneelin kautta.

Ennen muutoksien tekoa luotiin uusi ”muutostyo”-niminen työhaara versionhallintajärjestelmään. Tämän jälkeen tehtiin yllä kuvatut muutokset kehitysympäristössä ja vietiin Drupalin konfiguraatio koodiksi. Lopuksi muutokset tallennettiin versionhallintajärjestelmään. Tässä vaiheessa kehitysympäristössä oli siis punaisella taustalla oleva Drupal-sivusto, jonka nimi oli Muutostyö (kuva 12).



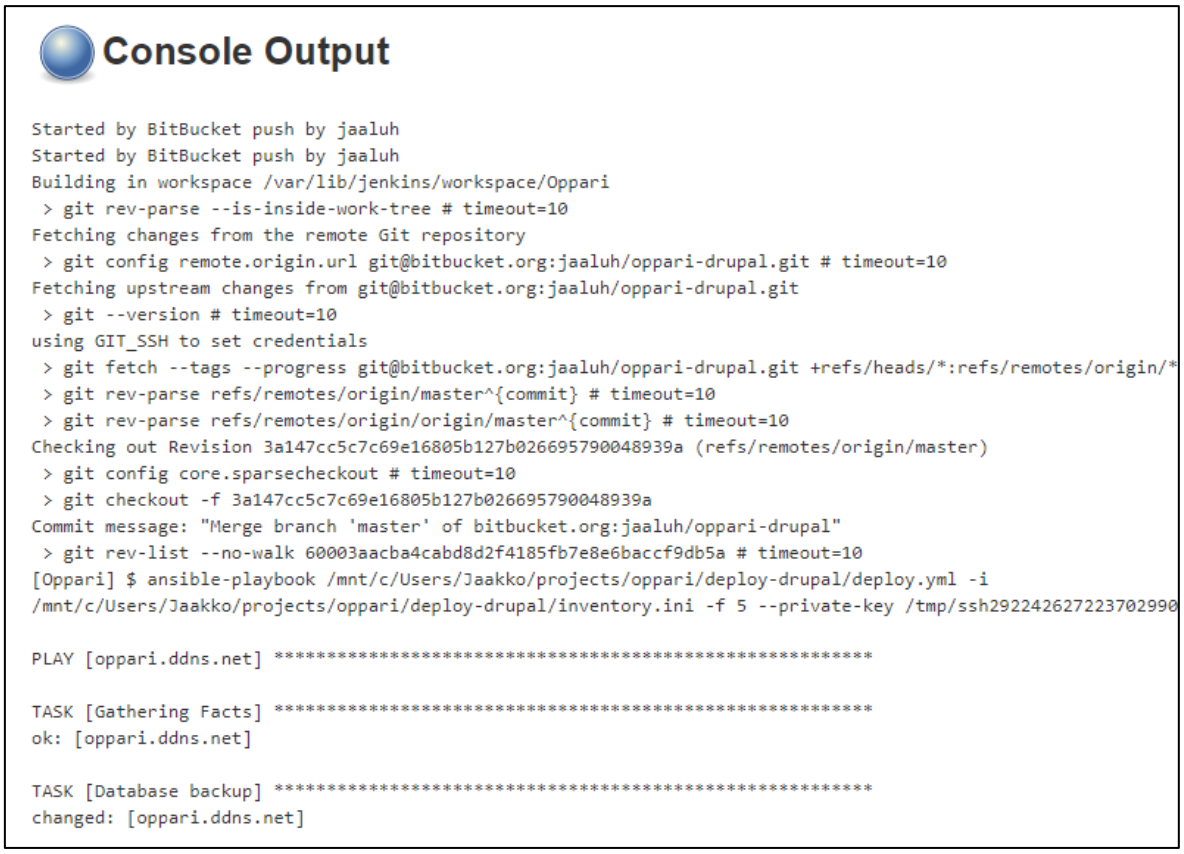
Kuva 12. Muutostyö kehitysympäristössä

Muutostyön käyttöönotto suoritettiin lopuksi yhdistämällä ”muutostyo”-työhaara versionhallinnan päätyöhaaraan Bitbucket-palvelussa. Tämä on käytännössä hyvin yksinkertainen toimenpide, joka voidaan toteuttaa palvelun käyttöliittymän kautta selaimen välityksellä.

5.7.2 Havainnot kehitystyöstä

Kehitystyön käyttöönottoprosessin suoritus ei vaatinut tekijältään aktiivista osallistumista muuten, kuin kehitystyöhaaran yhdistämisessä versionhallinnan päätyöhaaraan. Kun tämä toimenpide suoritettiin, havaittiin toteutetun ratkaisun toimivan suunnitellulla tavalla, eli Jenkins sai tiedon muutoksesta ja käynnisti Ansbilella määritellyn käyttöönottoprosessin. Käyttöönottoprosessin suoritus sujui ongelmitta ja suorituksen jälkeen tarkistettiin tilanne tuotantoympäristössä, ja siellä todettiin olevan samanlainen sivusto, kuin kehitysympäristössä (kuva 12). Työ suoritettiin lopuksi viisi kertaa Jenkinsin käyttöliittymän kautta käynnistämällä ja lopputulos pysyi joka suorituskerran jälkeen samanlaisena.

Jenkinsin kautta voitiin tarkkailla työn etenemistä terminaalinäkymän kautta (kuva 13).



Kuva 13. Jenkins työn seuranta terminaalinäkymässä

Terminaalinäkymästä pystyttiin seuraaman työssä tehtyjä vaiheita ja Ansible-pelikirjan suorittamista samaan tapaan kuin jos työ olisi suoritettu manuaalisesti terminaalikäskyllä. Merkintöjä työn suorituksesta oli mahdollista tarkastella myös jälkikäteen (kuva 14).

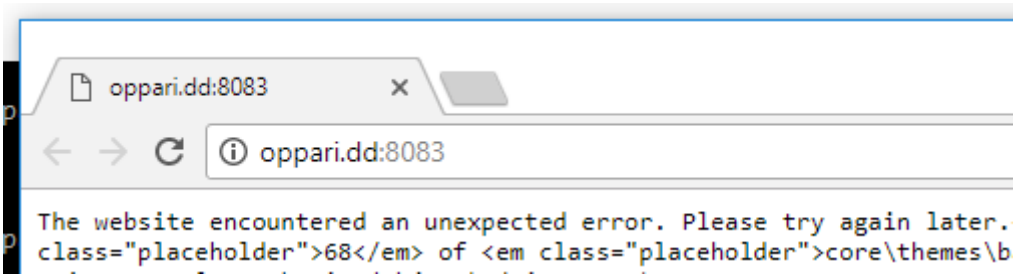
Kuva 14. Jenkins työn lopputulos

Työn tuloksista (kuva 14) voidaan nähdä mm. seuraavat asiat:

- Sininen pallo vasemmalla ylhäällä kertoo suorituksen onnistuneen (pallo olisi punainen, jos suorituksessa olisi tullut jokin virhe).
- Teksti sinisen pallon vieressä kertoo, monesko työn suorituskerta tämä oli sekä myös sen ajankohdan.
- ”Started”-teksti oikealla ylhäällä kertoo, koska Jenkins-työ aloitettiin. Tämän alla oleva ”Took”-teksti puolestaan kertoo, kuinka kauan työn suoritus kesti.
- ”Changes”-kohdassa on listattu muutokset, jotka versionhallintajärjestelmään lähetetylle työlle oli listattu sekä myös linkit, joiden kautta voitaisiin tutkia muutoksia tarkemmin.
- ”Revision”-kohdasta voidaan nähdä versionhallinnasta haettuun työhön viittaava yksilöivä merkkijono, jonka avulla voidaan selvittää, mikä versio sovelluksesta on käytössä.

5.7.3 Virheellisen muutostyön suoritus

Ohjelmistokehityksessä virheitä syntyy helposti ja aina ei voida välttyä virheellisen koodin lähettämiseltä versionhallintajärjestelmään. Jo yhdenkin merkin puuttuminen voi aiheuttaa koko sivuston kaatumisen ja tämänkaltaista tilannetta mallinnettiin poistamalla tarkoituksella yksi %-merkki aktiivisen teeman sivupohjasta. Kun sivusto tämän jälkeen avattiin kehitysympäristössä, todettiin se toimimattomaksi (kuva 15).



Kuva 15. Tahallinen virhetilanne

Virheen sisältävä muutostyö tallennettiin lopuksi versionhallintaan. Sen käyttöönotto suoritettiin kuten aiemmassa muutostyössä, eli yhdistämällä ”muutostyo”-työhaara versionhallinnan päätyöhaaraan Bitbucket-palvelussa.

5.7.4 Havainnot virheellisestä muutostyöstä

Virheen sisältänyt muutos yhdistettiin versionhallinnan päätyöhaaraan ja lähetettiin Bitbucket-palveluun, joka puolestaan jälleen käynnisti Jenkins työn. Jenkins työn suoritus

eteni samalla tavalla kuin kehitystyön kohdalla tehdyissä havainnoissa luvussa 5.7.2 ja sen suorituksessa ei siis ilmennyt ongelmia.

Lopuksi tarkistettiin tilanne tuotantoympäristössä avaamalla sivusto selaimessa ja myös siellä törmättiin samaan virheilmoitukseen (kuva 15). Tärkeimpänä havaintona tästä muutostyöstä voidaan siis mainita, ettei toteutettu ratkaisu estänyt virheen etenemistä tuotantoympäristöön. Käytännössä ratkaisu ei siis sisällä minkäänlaista tarkistusta sovelluksen koodin virheettömyyden osalta ja myös virheen sisältävä työ merkitään onnistuneeksi Jenkinsissä, sillä työ sujui määritetyllä tavalla suunnitelman mukaan.

5.7.5 Ongelmatilanteesta palautuminen

Tässä vaiheessa tuotantoympäristössä oli siis toimimaton verkkosivusto (kuva 15), joka täytyisi pystyä palauttamaan takaisin toimintakuntoon. Luvussa 5.7.3 aiheutettuun virheeseen ei sisältynyt muutoksia tietokantaan, vaan ainoastaan sovelluksen lähdekoodiin, joten ongelmasta voitiin siis palautua palauttamalla tiedostot virhettä edeltävään tilaan.

Käytännössä palautus tehtiin palauttamalla kehitysympäristön sivusto aiempaan versioon versionhallintajärjestelmän avulla. Tämän jälkeen muutos tallennettiin versionhallintajärjestelmään ja käyttöönotto suoritettiin taas yhdistämällä ”muutostyo”-työhaara versionhallinnan päätyöhaaraan Bitbucket-palvelussa.

5.7.6 Havainnot ongelmatilanteesta palautumisesta

Jenkins työ eteni kuten aiemmissa tapauksissa ja työn suorituksen jälkeen jälleen tarkistettiin tilanne tuotantoympäristössä, jossa havaittiin sivuston palautuneen toimintakuntoon. Myös tämä työ suoritettiin viidesti Jenkinsin käyttöliittymän kautta käynnistämällä ja lopputulos pysyi joka suorituskerran jälkeen samanlaisena.

Ongelmasta palautumiseen ei tässä tapauksessa liittynyt tietokannan palautusta, koska sitä ei automatisoitu osaksi prosessia luvussa 5.5.3 mainituista syistä. Jenkins työn suorituksesta voitiin kuitenkin havaita, että tietokannasta otettiin varmuuskopio ennen muutosten tekoa (kuva 16).

```
PLAY [oppari.ddns.net] *********************************************************

TASK [Gathering Facts] *********************************************************
ok: [oppari.ddns.net]

TASK [Database backup] *********************************************************
changed: [oppari.ddns.net]
```

Kuva 16. Tietokannan varmuuskopiointivaiheen suoritus

Tietokantavarmuuskopioiden olemassaolo varmistettiin myös tuotantoympäristössä tarkistamalla varmuuskopioiden sijainniksi määritetyn kansion sisältö. Kansiossa havaittiin olevan tietokantavarmuuskopiot, jotka oli Ansiblessa määritetyllä tavalla nimetty sivuston nimellä ja varmuuskopion kellonajalla (kuva 17).

```
root@oppari:~/drupal-db-backups# ls
oppari.ddns.net-2018-04-08T19:05:04Z.sql
oppari.ddns.net-2018-04-08T19:06:16Z.sql
oppari.ddns.net-2018-04-08T19:09:48Z.sql
```

Kuva 17. Tietokantavarmuuskopiot tuotantoympäristössä

Näiden tiedostojen avulla tietokanta ja sitä myöden Drupalin tila olisi siis ongelmatilanteissa mahdollista palauttaa tiedostonimestä selviävään ajankohtaan.

5.8 Ratkaisun arviointi

Kuten luvussa 2.3.1 todettiin, suunnittelutieteellisessä tutkimuksessa tuotoksen toimivuutta arvioidaan vertaamalla tehtyjä havaintoja määritettyihin ratkaisun tavoitteisiin. Tämän tutkimuksen osalta ratkaisun arviointi suoritettiin siis vertaamalla luvussa 5.7 tehtyjä havaintoja ratkaisun toimivuudesta luvussa 5.4 määritettyihin ratkaisun tavoitteisiin. Luvussa 2.3.2 ratkaisun arviointitapaa tarkennettiin siten, että ratkaisun tavoitteet tullaan käymään yksitellen läpi ja jokaisen kohdalla arvioidaan induktiivista päättelyä käyttäen, kuinka hyvin kyseinen tavoite pystyttiin ratkaisemaan.

Ensimmäinen tavoite oli, ettei käyttöönottoprosessin suorittamisen tulisi vaatia tekijältään aktiivista osallistumista. Luvussa 5.7 tehdyissä muutostöissä käyttöönotto pystyttiin jokaisen tehdyn muutoksen kohdalla suorittamaan yhdistämällä kehitystyöhaara versionhallinnan päätyöhaaraan Bitbucket-palvelun käyttöliittymän kautta. Koska tämä yhdistäminen on hyvin yksinkertainen toimenpide, voidaan tehtyjen havaintojen perusteella todeta, että toteutetussa ratkaisussa käyttäjältä ei vaadita aktiivista osallistumista käyttöönottoprosessin suorittamiseen.

Toinen tavoite oli, että käyttöönottoprosessiin määritetyt työvaiheet suoritetaan toistettavasti ja luotettavasti. Lukujen 5.7.1 ja 5.7.5 muutostyöt suoritettiin kuusi kertaa (kerran automaattisesti versionhallinnan kytkennästä ja viidesti Jenkinsin käyttöliittymän kautta) ja jokaisella kerralla lopputulos havaittiin oikeanlaiseksi. Tehdyissä muutostöissä ei myöskään havaittu mitään epämääräistä, vaan jokaisessa työssä vaiheet tapahtuivat juuri määritellyllä tavalla. Tehtyjen havaintojen perusteella voidaan siis tehdä päätelmä, että käyttöönottoprosessiin määritetyt työvaiheet suoritetaan toistettavasti ja luotettavasti.

Kolmanneksi tavoitteeksi ratkaisulle asetettiin, että käyttöönottoprosessin suorittamisesta ja lopputuloksista tulisi pitää kirjaa. Luvussa 5.7.2 tehdyistä havainnoista nähdään, että käyttöönottoprosessin suoritusta oli mahdollista seurata terminaalinäkymästä, josta nähtiin työhön kuuluneiden toimenpiteiden suoritus vaiheittain. Tämän lisäksi samassa luvussa todettiin, että työn tuloksia oli mahdollista tarkastella myös jälkikäteen ja näin pystyttiin näkemään mm. työn suoritusaika, työn kesto, työhön sisältyneet muutokset ja työssä käyttöönotetun sovellukset versio (tai siihen viittaava merkkijono). Näiden havaintojen perusteella voidaan siis tehdä päätelmä, että toteutettu ratkaisu pitää kirjaa käyttöönottoprosessin suorittamisesta ja lopputuloksista.

Neljäntenä tavoitteena oli, ettei käyttöönottoprosessin suorittaminen vaatisi käyttäjältä tapauskohtaista tietoa. Luvussa 5.7 suoritettiin kolme muutostyötä ja jokaisen työn kohdalla käyttöönoton suoritus tapahtui samalla tavalla, eli versionhallinnan välityksellä. Tehtyjen havaintojen perusteella käyttöönoton suorittajalta ei missään vaiheessa kysytty lisätietoja, kuten esimerkiksi salasanaa tai tuotantopalvelimen osoitetta. Kaikki käyttöönottoprosessin työvaiheet suoritettiin siis automaattisesti, joten tästä voidaan tehdä päätelmä, ettei käyttäjältä vaadita käyttöönottoprosessin suorittamiseksi tapauskohtaista tietoa.

Alla olevaan taulukkoon (taulukko 3) on vielä koottu yhteenvetona yllä käsitellyt ratkaisun tavoitteet ja havainnot, joiden mukaan tavoite katsottiin saavutetuksi.

Taulukko 3. Ratkaisun toimivuuden arviointi

Ratkaisun tavoite	**Havainnot, jonka perusteella tavoite todettiin saavutetuksi**
Ei vaadi tekijältä aktiivista osallistumista.	Luvussa 5.7 tehdyissä muutostöissä käyttöönottoprosessi käynnistettiin yhdistämällä kehitystyöhaara päätyöhaaraan versionhallintajärjestelmän kautta.
Toteuttaa määritetyt työvaiheet toistettavasti ja luotettavasti.	Lukujen 5.7.1 ja 5.7.5 muutostyöt suoritettiin kuusi kertaa ja lopputulos pysyi joka kerta samanlaisena.
Pitää kirjaa käyttöönottoprosessin suorittamisesta ja lopputuloksista.	Luvussa 5.7.2 käytiin läpi havaintoja Jenkins-työn suorituksesta, joiden perusteella prosessin etenemistä ja tuloksia on mahdollista seurata yksityiskohtaisesti.
Ei vaadi tekijältä tapauskohtaista tietoa.	Koska käyttäjältä ei luvun 5.7 muutostöissä kysytty lisätietoja, voidaan katsoa, ettei käyttäjältä vaadita tapauskohtaista tietoa prosessin suorittamiseksi.

Yllä käsiteltyjen havaintojen ja niistä muodostettujen päätelmien perusteella voidaan tässä työssä toteutettu ratkaisu automaatiota hyödyntävästä Drupal-julkaisujärjestelmän käyttöönottoprosessista todeta onnistuneeksi, sillä se saavutti kaikki sille asetetut tavoitteet.

Vaikka toteutettu ratkaisu saavuttikin kaikki sille asetetut tavoitteet, havaittiin siinä myös yksi puute. Luvussa 5.7.3 toteutettu tahallisen virheen sisältänyt muutostyö vietiin myös automaattisesti tuotantoympäristöön ja muutostyöstä tehdyissä havainnoissa todettiin, ettei ratkaisu sisällä minkäänlaista virheentarkistusta sovelluksen lähdekoodin osalta. Tämä puute oli työn kirjoittajan tiedossa jo ratkaisua suunnitellessa, mutta on aiheellista tuoda se selkeästi esille, vaikkei virheentarkistusta ratkaisun tavoitteisiin määriteltykään.

6 JOHTOPÄÄTÖKSET

Toteutettu ratkaisu automaatiota hyödyntävästä Drupal-julkaisujärjestelmän käyttöönottoprosessista rakennettiin pääasiassa kolmen työkalun avulla, jotka olivat Git-versionhallintajärjestelmä, Ansible-komentorivityökalu ja Jenkins-automaatiopalvelin. Git toi prosessiin mahdollisuuden palautua tiedostoihin liittyvistä virhetilanteista ja sitä hyödynnettiin myös tiedostojen siirrossa ympäristöjen välillä. Ansible mahdollisti käyttöönottoprosessin vaiheiden automatisoinnin luotettavasti ja Jenkins kytki työkalut toisiinsa, minkä lisäksi se toimi myös prosessin keskitettynä hallintapisteenä. Jokainen näistä työkaluista toi mukanaan oleellisia ominaisuuksia, joita ilman ratkaisua ei olisi voitu toteuttaa.

Toteutetun ratkaisun toimivuus osoitettiin suorittamalla kolme erityyppistä muutostyötä Drupal-sivustolle. Nämä muutostyöt osoittivat, että ratkaisulla voidaan viedä kehitysympäristössä tehty työ hyvin pienellä vaivalla tuotantoympäristöön. Samalla myös todettiin puutteita ratkaisussa, sillä myös virheellinen muutostyö vietiin tuotantoympäristöön samalla tavalla kuin mikä tahansa muutos. Tämä on selkeä puute ratkaisussa, joka voitiin kuitenkin paikata palauttamalla sivusto virhettä edeltävään tilanteeseen. Käytännössä muutostöiden laatu ja virheettömyys tässä ratkaisussa on siis kokonaan työntekijän vastuulla, aivan kuten lähtötilanteena olleessa manuaalisessa käyttöönottoprosessissa.

Vaikka tehdyn työn virheettömyys on toteutetussa ratkaisussa edelleen työntekijän vastuulla, on vastuu käyttöönottoprosessin suorituksesta siirtynyt tietokoneille. Tämän ansiosta käyttöönoton voi toteuttaa yrityksessä lähes kuka vain, kenellä on riittävät oikeudet versionhallintajärjestelmään. Käyttöönotto voidaan näin suorittaa joustavasti ja esimerkiksi ajoittaa se asiakkaalle sopivaan ajankohtaan.

Kun toteutettua ratkaisua verrattaan lähtötilanteena olleeseen manuaaliseen käyttöönottoprosessiin, voidaan yllä mainittujen asioiden lisäksi todeta useita muitakin parannuksia. Koska tilitiedot ja salasanat on määritetty osaksi käyttöönottoprosessia, ei työntekijän tarvitse kaivella niitä ja näin ollen työnteko on nopeampaa ja miellyttävämpää sekä myös tietoturvan osalta tilanne on parempi. Myös projektien tilanteen selvittäminen helpottuu, sillä automaatiopalvelimelta voidaan helposti selvittää mm. milloin sovellukseen on tehty muutoksia, mikä versio sovelluksesta on käytössä ja ongelmatilanteissa voidaan myös jäljittää, mitkä toimenpiteet tilanteeseen johtivat. Käyttöönottoprosessiin kuluvaa työaikaa ei tutkimuksessa mitattu, mutta hyvin todennäköisesti automaattisen prosessin suorittamiseen kuluva aktiivinen työaika on huomattavasti pienempi kuin lähtötilanteen manuaalisessa prosessissa.

Vaikka automaattinen käyttöönottoprosessi toi huomattavia etuja, menetetään siinä myös manuaalisen prosessin joustavuus. Kokenut työntekijä voi käyttöönottoa suorittaessa mm. ratkaista monia eteen tulevia odottamattomia ongelmia, joihin automaattinen prosessi kaatuisi. Joissain tapauksissa manuaalisen käyttöönottoprosessin joustavauus voi siis olla automaation tuomia etuja tärkeämpi.

Kokonaisuudessaan toteutettu ratkaisu tuo kuitenkin huomattavia etuja lähtötilanteen manuaaliseen prosessiin verrattuna. Tutkimuskysymykseen voidaan siis vastata, että automaatiota voidaan hyödyntää Drupal-julkaisujärjestelmän käyttöönottoprosessin kehityksessä esimerkiksi ratkaisun toteutukseen käytetyillä työkaluilla ja menetelmillä.

7 YHTEENVETO

Tätä opinnäytetyötä lähdettiin toteuttamaan, koska toimeksiantajayrityksessä oltiin havaittu tarve kehittää käytössä olevien avoimen lähdekoodin julkaisujärjestelmien käyttöönottoprosesseja. Työn päätarkoituksena toimeksiantajan kannalta oli kartoittaa mahdollisia keinoja, joilla käyttöönottoprosessia voitaisiin automatisoida.

Opinnäytetyön tutkimusosuus toteutettiin suunnittelutieteellistä tutkimusmenetelmää käyttäen. Tavoitteena tutkimuksessa oli selvittää, kuinka automaatiota voidaan käytännössä hyödyntää Drupal-julkaisujärjestelmän käyttöönottoprosessin kehityksessä. Tutkimuksessa toteutettiin ratkaisu automaatiota hyödyntävästä käyttöönottoprosessista, jonka kykyä ratkaista tutkittava ongelma arvioitiin suorittamalla muutostöitä Drupal-sivustolle. Ratkaisu saavutti kaikki sille asetetut tavoitteet ja toi mukanaan etuja, joiden perusteella käyttöönottoprosessin kehityksen katsottiin toteutuneen. Suurin havaittu ongelma ratkaisussa oli virheentarkistuksen puute.

Tutkimuksessa käytetty aineisto kerättiin havainnoimalla toteutettua ratkaisua käytännön tilanteessa ja tehdyt havainnot ovat toistettavissa seuraamalla tutkimukseen dokumentoituja vaiheita, joten siltä osin tutkimusta voidaan pitää luotettavana. Tutkimuksen yleistettävyyttä arvioidessa kiinnittyy huomio etenkin testiympäristön rajallisiin puitteisiin. Rakennetulla testiympäristöllä pyrittiin mallintamaan todellista työelämän tilannetta, mutta käytännössä ympäristö oli kuitenkin hyvin yksinkertainen. Myös käytössä ollut kohdeprojekti oli huomattavasti yksinkertaisempi, kuin esimerkiksi Drupalilla toteutettu yrityssivusto. Tulokset eivät siis välttämättä ole yleistettävissä työelämän projekteihin ja toteutettua ratkaisua tulisikin pitää lähinnä viitteellisenä ratkaisuna, jota voidaan käyttää pohjana käyttöönottoprosessien kehityksessä. Tutkimuksen avulla pystyttiin kuitenkin keräämään paljon aiheen kannalta merkityksellistä käytännönläheistä tietoa sekä vastaamaan myös asetettuun tutkimuskysymykseen, joten tutkimus voidaan katsoa päteväksi, kunhan otetaan huomioon yllä mainitut asiat.

Opinnäytetyön aihe vaikutti ennen työn aloittamista melko pieneltä ja helposti hallittavalta, mutta toteutuksen aikana ilmeni, että monet osa-alueet olivat yllättävän laajoja ja voisivat toimia itsenäisinä tutkimuksen kohteina. Työssä keskityttiin käyttöönottoprosessin automatisointiin käytettyihin tekniikoihin, mutta työtä tehdessä tuli myös esille, että verkkosovellusten käyttöönottoon liittyy monia muitakin osa-alueita, kuten henkilöstön koulutus ja ohjeistus. Työssä toteutettu ratkaisu on siis vain yksi osa käyttöönottoa ja sen sisällyttäminen yrityksen toimintatapoihin voisi olla hyvä aihe jatkotutkimuksille. Myös jatkuvan integraation työmenetelmän ja siihen sisältyvän automaattisen testauksen soveltamista julkai-

sujärjestelmiin olisi hyödyllistä selvittää. Automaattisen testauksen avulla voitaisiin mahdollisesti korjata tässä työssä toteutetun ratkaisun ongelmaksi havaittu virheentarkistuksen puute.

Opinnäytetyössä on tavoitteena myös oppia uusia asioita ja tämän opinnäytetyön tekijälle kertyi työtä tehdessä paljon uutta tietoa Drupal-julkaisujärjestelmästä, verkkosovellusten käyttöönottoprosessista sekä erilaisista käyttöönottoprosessin automatisointiin soveltuvista työkaluista ja menetelmistä. Työssä toteutetun ratkaisun rakentamiseen tarvitun tiedon lisäksi myös käytetyt työtavat ja tutkimusmenetelmät olivat työn kirjoittajalle uusia asioita, joista voi mahdollisesti olla hyötyä myös työelämässä. Kaikkiaan opinnäytetyöprojekti oli siis onnistunut, sillä sen avulla pystyttiin keräämään hyödyllistä tietoa työn kirjoittajalle ja sen toimeksiantajalle.

LÄHTEET

Acquia. 2018. Acquia Dev Desktop [viitattu 26.3.2018]. Saatavissa: https://www.acquia.com/products-services/dev-desktop

Belzunce, F. 2018. Bitbucket plugin [viitattu 8.4.2018]. Saatavissa: https://plugins.jenkins.io/bitbucket

Black Duck Software, Inc. 2018. Open Hub [viitattu 17.4.2018]. Saatavissa: https://www.openhub.net/p/drupal

Bucher, S. G., Dunwoodie, B., & Shreves, R. 2011. Bible : Drupal 7 Bible. [Verkkokirja]. New Jersey: John Wiley & Sons. [Viitattu 27.3.2018]. Saatavissa: https://masto.finna.fi/Record/nelli16_phkk.2550000000041236

Coyier, C. 2013. Website Deployment: Let Us Count The Ways! [viitattu 16.3.2018]. Saatavissa: https://css-tricks.com/deployment/

Drupal.org. 2016. Directory Structure [viitattu 17.3.2018]. Saatavissa: https://www.drupal.org/docs/8/understanding-drupal/directory-structure

Drupal.org. 2018a. Development tools overview [viitattu 17.3.2018]. Saatavissa: https://www.drupal.org/docs/develop/development-tools/development-tools-overview

Drupal.org. 2018b. Drupal API [viitattu 17.3.2018]. Saatavissa: https://api.drupal.org/api/drupal

Drupal.org. 2018c. Installing Drupal 8 [viitattu 14.3.2018]. Saatavissa: https://www.drupal.org/docs/8/install

Drupal.org. 2018d. Managing your site's configuration [viitattu 14.3.2018]. Saatavissa: https://www.drupal.org/docs/8/configuration-management/managing-your-sites-configuration

Ellingwood, J. 2014. How To Configure SSH Key-Based Authentication on a Linux Server [viitattu 7.4.2018]. Saatavissa: https://www.digitalocean.com/community/tutorials/how-to-configure-ssh-key-based-authentication-on-a-linux-server

Ellingwood, J. 2017. An Introduction to Continuous Integration, Delivery, and Deployment [viitattu 16.4.2018]. Saatavissa: https://www.digitalocean.com/community/tutorials/an-introduction-to-continuous-integration-delivery-and-deployment

Fideloper LLC. 2018. An Ansible2 Tutorial [viitattu 7.4.2018]. Saatavissa: https://serversforhackers.com/c/an-ansible2-tutorial

Git. 2018a. Getting Started - About Version Control [viitattu 16.3.2018]. Saatavissa: https://git-scm.com/book/en/v1/Getting-Started-About-Version-Control

Git. 2018b. Git Branching - Branches in a Nutshell [viitattu 24.3.2018]. Saatavissa: https://git-scm.com/book/en/v2/Git-Branching-Branches-in-a-Nutshell

Jenkins. 2018a. Jenkins User Documentation [viitattu 24.3.2018]. Saatavissa: https://jenkins.io/doc/

Jenkins. 2018b. Installing Jenkins [viitattu 8.4.2018]. Saatavissa: https://jenkins.io/doc/book/installing/#debian-ubuntu

Jenkins. 2018c. Using credentials [viitattu 8.4.2018]. Saatavissa: https://jenkins.io/doc/book/using/using-credentials/

Lullabot Education. 2018. Drupalize.me - Deployment Workflows [viitattu 17.3.2018]. Saatavissa: https://drupalize.me/topic/deployment-workflows

March, S. T. & Smith, G. F. 1995. Design and natural science research on information technology. [Verkkodokumentti]. Minneapolis: Decision Support Systems(15), 251-266. [Viitattu 5.2.2018]. Saatavissa: https://pdfs.semanticscholar.org/d93f/fe572b15a163e2ec1336a4e507b0b7a766f0.pdf

Microsoft. 2017. Install the Windows Subsystem for Linux [viitattu 25.3.2018]. Saatavissa: https://docs.microsoft.com/en-us/windows/wsl/install-win10

Octopus Deploy. 2014. The Benefits of Deploymen Automation. [Verkkodokumentti]. Octopus Deploy. [viitattu 10.2.2018]. Saatavissa: http://download.octopusdeploy.com/files/whitepaper-automated-deployment-octopus-deploy.pdf

Peda.net. 2018. Tieteellinen tutkimusmenetelmä [viitattu 17.4.2018]. Saatavissa: https://peda.net/iin-kunta/iin-lukio/oppiaineet2/filosofia/e1atel2/filosofia1-1001152/4-tieteenfilosofia/tutkimus

Peffers, K.,Tuunanen, T.,Rothenberger, M. A. & Chatterjee, S. 2008. A Design Science Research Methodology. [Verkkodokumentti]. Journal of Management Information Systems, 24(3), 45-78. [Viitattu 5.2.2018]. Saatavissa: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.535.7773&rep=rep1&type=pdf

Pittet, S. 2018. Continuous integration vs. continuous delivery vs. continuous deployment [viitattu 22.2.2018]. Saatavissa: https://www.atlassian.com/continuous-delivery/ci-vs-ci-vs-cd

Ravichandran, A.,Taylor, K. & Waterhouse, P. 2016. DevOps for Digital Leaders. [Verkkokirja]. Berkeley: Apress. [Viitattu 13.2.2018]. Saatavissa: https://link.springer.com/10.1007/978-1-4842-1842-6

Red Hat, Inc. 2017. Ansible Docs - About Ansible [viitattu 24.3.2018]. Saatavissa: http://docs.ansible.com/ansible/latest/index.html

Red Hat, Inc. 2018a. Ansible Docs - Getting Started [viitattu 7.4.2018]. Saatavissa: http://docs.ansible.com/ansible/latest/user_guide/intro_getting_started.html

Red Hat, Inc. 2018b. Ansible Docs - Installation Guide [viitattu 27.3.2018]. Saatavissa: http://docs.ansible.com/ansible/latest/installation_guide/intro_installation.html

Red Hat, Inc. 2018c. Ansible Docs - Working with Inventory [viitattu 7.4.2018]. Saatavissa: http://docs.ansible.com/ansible/devel/user_guide/intro_inventory.html

Red Hat, Inc. 2018d. Ansible Docs - Working With Playbooks [viitattu 7.4.2018]. Saatavissa: http://docs.ansible.com/ansible/devel/user_guide/playbooks.html

Red Hat, Inc. 2018e. Ansible Docs - Intro to Playbooks [viitattu 7.4.2018]. Saatavissa: http://docs.ansible.com/ansible/devel/user_guide/playbooks_intro.html

Red Hat, Inc. 2018f. Ansible Docs - Module Index [viitattu 7.4.2018]. Saatavissa: http://docs.ansible.com/ansible/latest/modules/modules_by_category.html

Stephens, R. 2015. Beginning Software Engineering. [Verkkokirja]. Hoboken: Wiley. [Viitattu 25.3.2018]. Saatavissa: https://masto.finna.fi/Record/nelli16_phkk.3710000000370116

Taylor, A. 2016. Managing Site Configuration in Code [viitattu 14.3.2018]. Saatavissa: https://pantheon.io/blog/managing-site-configuration-code

W3Techs. 2018. Usage of content management systems for websites [viitattu 23.3.2018]. Saatavissa: https://w3techs.com/technologies/overview/content_management/all

LIITTEET

LIITE 1. Ansible-pelikirjassa muuttujien määrittelyyn käytetty main.yml-tiedosto

Tiedoston sisältö (muuttujien arvot ovat esimerkkejä):

```
hostname: 'someaddress.net'
repo_address: 'git@bitbucket.org:user/repo.git'
project_root: '/path/to/webroot'
db_name: 'dbname'
db_user: 'dbuser'
db_password: 'dbpass'
```

LIITE 2. Ansible-pelikirjan määrittelyyn käytetty deploy.yml-tiedosto

Tiedoston sisältö:

```
- hosts: "{{ hostname }}"

  remote_user: root

  vars_files:
    - vars/main.yml

  tasks:
    - name: Database backup
      mysql_db:
        state: dump
        name: "{{ db_name }}"
        target: ~/drupal-db-backups/{{ hostname }}-{{
ansible_date_time.iso8601 }}.sql
        login_user: "{{ db_user }}"
        login_password: "{{ db_password }}"


    - name: Pull from Git
      git:
        repo: "{{ repo_address }}"
        dest: "{{ project_root }}"
        accept_hostkey: yes
        force: yes

    - name: Composer install
      shell: composer install
      args:
        chdir: "{{ project_root }}"


    - name: Drush configuration import
      shell: drush config-import -y
      args:
        chdir: "{{ project_root }}"

    - name: Drush reset temp dir
      shell: drush config-set system.file path.temporary
/tmp -y
      args:
        chdir: "{{ project_root }}"

    - name: Drush database updates
      shell: drush updatedb -y
      args:
        chdir: "{{ project_root }}"

    - name: Drush cache rebuild
      shell: drush cr
```

```
args:
  chdir: "{{ project_root }}"
```