**Afstudeerdossier**

Onderzoek software distributie

Welke software distributie tool sluit het beste aan op de software distributie binnen het Webapps team?



Auteur: Ruben Bouterse Eerste docent: H.P.Weenink

Opleiding: Informatica Tweede docent: G.M.Tuk

Bedrijf: Finalist Begeleider: R.Schellhorn

Opdrachtgever: M. van Breukelen

Datum: 04-08-2014

Versie: 3.0

# Referaat

Het doel van dit onderzoek is antwoord te krijgen op de hoofdvraag “Welke software distributie tool sluit het beste aan op de software distributie binnen het Webapps team?”. Om een duidelijk antwoord op deze hoofdvraag te krijgen, zijn er vier deelvragen beantwoord.

De eerste deelvraag begint met het beschrijven van de infrastructuur binnen Finalist. Vervolgens wordt er een duidelijk beeld van de huidige situatie van software distributie(1) binnen het Webapps team beschreven. Binnen de huidige situatie worden de processen die binnen de software distributie vallen in detail uitgeschreven.

De tweede deelvraag beschrijft de knelpunten die vastgesteld zijn. Daarnaast heb ik aan de hand van een interview het ideale toekomstbeeld van het Webapps team kunnen achterhalen en deze uitgeschreven in de deelvraag. Het ideale toekomstbeeld is beschreven om mogelijke oplossingen van de knelpunten vast te stellen. Tot slot heb ik requirements opgesteld en verwerkt in een requirementsrapport.

Binnen de derde deelvraag is een longlist opgesteld met als doel een tool te vinden welke de beste kandidaat is om toe te passen voor het Webapps team. De eerste stap die is ondernomen is het opstellen van criteria voor de longlist. De criteria zijn opgesteld conform (M)ust have requirements, requirements die verbonden zijn aan een knelpunt en extra factoren. Vervolgens worden er kandidaten beschreven voor de longlist. Er is een literatuuronderzoek gestart om de criteria te bevestigen voor de opgestelde kandidaten. Tot slot is er één tool uit de longlist gekozen welke als beste kandidaat voor het Webapps team zou kunnen functioneren.

Binnen de vierde en laatste deelvraag zijn de tools van de huidige situatie en de beste tool uit de longlist in praktijk getest. De testen bestaan uit testcases welke opgesteld zijn conform de opgestelde requirements.

Gezien de resultaten van de deelvragen, heb ik een verbeteradvies kunnen opstellen voor Finalist.

# 

# Voorwoord

Dit document dient als afstudeerdossier voor het uitgevoerde onderzoek naar software distributie in opdracht van Finalist. De opdracht is uitgevoerd door Ruben Bouterse, student aan De Haagse Hogeschool in Den Haag ter afsluiting van de opleiding Informatica.

Binnen het afstudeerdossier staat beschreven hoe ik het onderzoek heb uitgevoerd, welke keuzes ik heb gemaakt en de onderbouwing van de gemaakte keuzes.

De doelgroep van het afstudeerdossier is ten eerste de afstudeercommissie die aangesteld is door De Haagse Hogeschool om mijn opdracht te beoordelen. Ten tweede is het afstudeerdossier voor een ieder die geïnteresseerd is in het onderzoek. Van de lezer wordt verwacht dat deze beschikt over een kleine hoeveelheid kennis van Informatica.

# Dankwoord

Ik wil graag mijn dank betuigen aan een aantal personen die mij geholpen hebben met de uitvoering van dit onderzoek naar software distributie. Om te beginnen wil ik alle onderstaande personen bedanken voor het begrip en de omgang omtrent de ziekte van pfeiffer welke tijdens mijn afstudeerperiode geconstateerd werd.

Rob Schellhorn, software ontwikkelaar bij Finalist voor zijn begeleidende rol en het geven van feedback op mijn werk. Arne Timmerman, software ontwikkelaar bij Finalist voor zijn ideeën en gedachtes omtrent het onderzoeksonderwerp en het aanbieden van een afstudeerplaats. Iris Mouthaan, HR Manager bij Finalist voor het mogelijk maken van het werken van halve dagen en thuis werken.

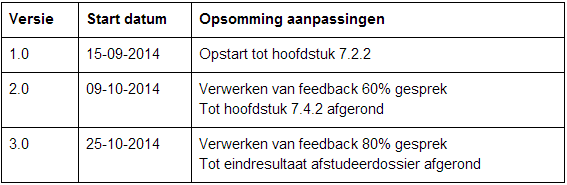
Van de Haagse Hogeschool wil ik mijn afstudeerbegeleiders Helene Weenink en Gerard Tuk bedanken voor hun begeleiding en het geven van feedback op mijn werk tijdens mijn afstudeerperiode.

Ook wil ik mijn vriendin Lianne Grijsen bedanken voor de interesse die ze toonde in de voortgang van mijn afstudeeropdracht.

In het bijzonder wil ik Jeroen Meijboom, mede afstudeerstudent bij Finalist bedanken voor de gezelligheid maar zeker ook de steun die we elkaar konden geven tijdens de uitvoering van onze opdrachten.

Tot slot wil ik alle werknemers binnen Finalist bedanken die betrokken zijn geweest bij mijn afstudeeropdracht.

# Versiebeheer



# Inhoudsopgave

Inhoud

[Referaat 2](#_Toc408222349)

[Voorwoord 3](#_Toc408222350)

[Dankwoord 3](#_Toc408222351)

[Versiebeheer 4](#_Toc408222352)

[Inhoudsopgave 5](#_Toc408222353)

[1. Inleiding 7](#_Toc408222354)

[2. Over het bedrijf 9](#_Toc408222355)

[3. De opdracht 10](#_Toc408222356)

[4. Probleemstelling 11](#_Toc408222357)

[5. Aanpak 12](#_Toc408222358)

[6. Voorbereiding onderzoek 13](#_Toc408222359)

[6.1 Planning V1.0 14](#_Toc408222360)

[6.2 Intern onderzoek 14](#_Toc408222361)

[6.3 Planning V2.0 17](#_Toc408222362)

[6.4 Vooronderzoek 18](#_Toc408222363)

[6.5 Onderzoeksplan 21](#_Toc408222364)

[6.6 Planning V3.0 25](#_Toc408222365)

[7. Uitvoering onderzoek 27](#_Toc408222366)

[7.1 Deelvraag 1 27](#_Toc408222367)

[7.1.1 Het bespreken van de infrastructuur binnen Finalist 27](#_Toc408222368)

[7.1.2 Gedetailleerde processen binnen software distributie van het Webapps team. 31](#_Toc408222369)

[7.2 Deelvraag 2 34](#_Toc408222370)

[7.2.1 Toelichting knelpunten Webapps team 34](#_Toc408222371)

[7.2.2 Beschrijven van ideaal toekomstbeeld Webapps team 36](#_Toc408222372)

[7.2.3 Requirementsrapport 40](#_Toc408222373)

[7.2.4 Planning V4.0 47](#_Toc408222374)

[7.3 Deelvraag 3 48](#_Toc408222375)

[7.3.1 Literatuuronderzoek & Opstellen longlist 49](#_Toc408222376)

[7.4 Deelvraag 4 54](#_Toc408222377)

[7.4.1 Architectuur Ansible & testproject (Fairwear) 54](#_Toc408222378)

[7.4.2 Opstellen testcases 57](#_Toc408222379)

[7.4.3 Uitvoering testcases 74](#_Toc408222380)

[7.5 Adviesrapport 76](#_Toc408222381)

[8. Conclusies 77](#_Toc408222382)

[8.1 Conclusie deelvraag 1 77](#_Toc408222383)

[8.2 Conclusie deelvraag 2 78](#_Toc408222384)

[8.3 Conclusie deelvraag 3 80](#_Toc408222385)

[8.4 Conclusie deelvraag 4 81](#_Toc408222386)

[8.5 Eindconclusie 83](#_Toc408222387)

[9. Evaluatie 84](#_Toc408222388)

[9.1 Hoe is het uitvoeringsproces verlopen 85](#_Toc408222389)

[9.2 Hoe zien de (tussen) producten eruit 88](#_Toc408222390)

[9.3 Behalen van de beroepstaken ( competenties ) 89](#_Toc408222391)

[10. Referenties 91](#_Toc408222392)

[10.1 Glossary 91](#_Toc408222393)

[10.2 Literatuurlijst 92](#_Toc408222394)

[10.3 bijlagen 93](#_Toc408222395)

[10.3.1 Vooronderzoek 93](#_Toc408222396)

[10.3.2 Onderzoeksplan 112](#_Toc408222397)

[10.3.3 Vragen half-gestructureerd interview 121](#_Toc408222398)

[10.3.4 Samenvatting interviews stakeholders 123](#_Toc408222399)

[10.3.5 Onderzoeksrapport 144](#_Toc408222400)

[10.3.6 Requirementsrapport 232](#_Toc408222401)

[10.3.7 Testrapport 242](#_Toc408222402)

[10.3.8 Adviesrapport 280](#_Toc408222403)

[10.3.9 Afstudeerplan 291](#_Toc408222404)

[10.3.10 Voortgangsverslag 297](#_Toc408222405)

[10.3.11 Resultaat bespreking concept afstudeerdossier 60% gesprek 300](#_Toc408222406)

[10.3.12 Resultaat bespreking TTA 80% gesprek 301](#_Toc408222407)

# 1. Inleiding

Graag wil ik de hoofdstukken toelichten die te lezen zijn in dit afstudeerdossier.

*Hoofdstuk 2*

Binnen dit hoofdstuk staat informatie beschreven over het bedrijf Finalist. Zo maakt het hoofdstuk duidelijk hoe groot de organisatie is en met welke passie Finalist te werk gaat.

*Hoofdstuk 3*

Binnen dit hoofdstuk staat een opdrachtomschrijving van de uitgevoerde opdracht vermeld.

*Hoofdstuk 4*

Binnen dit hoofdstuk wordt de probleemstelling beschreven. De centrale probleemstelling in dit onderzoek wordt samengevat in de hoofdvraag:

“Welke software distributie tool sluit het beste aan op de software distributie binnen het Webapps team?”

Er zijn vier deelvragen geformuleerd om de hoofdvraag te beantwoorden.

*Hoofdstuk 5*

Binnen dit hoofdstuk staat de aanpak beschreven welke ik gehanteerd heb om mijn onderzoek uit te voeren. De aanpak is verwerkt in een onderzoeksplan ( plan van aanpak ) waarin onderzoeksmethodes, de hoofd en deelvragen voor mijn onderzoek en de juiste uitvoeringsvolgorde van de deelvragen beschreven staan.

*Hoofdstuk 6*

Binnen dit hoofdstuk staat het proces beschreven van de voorbereiding van mijn onderzoek. Zo worden er opgestelde planningen besproken, het proces van het intern onderzoek en het proces van de uitvoering van het vooronderzoek.

*Hoofdstuk 7*

Binnen dit hoofdstuk staat de uitvoering van het onderzoek beschreven. Naast het uitgeschreven proces is te lezen welke keuzes ik heb gemaakt en onderbouwingen waarom ik deze keuzes gemaakt heb.

*Hoofdstuk 8*

Binnen dit hoofdstuk staan conclusies en aanbevelingen beschreven. Ook eventuele vervolgopdrachten die mogelijk zijn op mijn huidige onderzoek naar software distributie staan hier beschreven.

*Hoofdstuk 9*

Binnen dit hoofdstuk staat beschreven hoe het uitvoeringsproces van mijn project is verlopen. Dit bespreek ik door middel van een rode draad te beschrijven en deze toe te lichten.

Ook staat er beschreven hoe de opgestelde (tussen)producten eruit zien. Tot slot staat er een onderbouwing beschreven over het behalen van mijn beroepstaken binnen de opdracht.

*Hoofdstuk 10*

Binnen dit laatste hoofdstuk is een glossary te vinden waar de betekenis van moeilijke woorden worden toegelicht. Ook alle bronvermeldingen en bijlagen zijn terug te vinden in dit hoofdstuk.

In het document worden verwijzingen gedaan naar bronvermeldingen, de glossary en bijlagen. Binnen de tekst wordt dit als volgt weergegeven:

Bron vermelding: [Naam vermelding]

De totale literatuurlijst is te vinden in hoofdstuk 10.2

Glossary verwijzing: (nummer)

De totale glossary lijst is te vinden in hoofdstuk 10.1

Bijlage verwijzing: hoofdstuk 10.3.x

Binnen hoofdstuk 10.3 zijn de bijlagen van dit document te vinden. De bijlagen na hoofdstuk 10.3.8 zijn bijlagen waar niet naar verwezen wordt binnen het afstudeerdossier. Deze bijlagen bestaan uit documenten die opgesteld zijn tussen de student en school binnen het afstudeertraject.

# 

# 2. Over het bedrijf

Finalist is een organisatie die sinds 1988 actief is met het verzinnen, ontwerpen en bouwen van open IT oplossingen voor hun klanten. Momenteel beschikt Finalist over drie kantoren welke in Rotterdam, Eindhoven en Maarssen zijn gevestigd. Binnen deze drie organisaties werken +/- 80 werknemers. Binnen de vestiging Rotterdam zijn er 40 werknemers werkzaam.

De software ontwikkeling binnen Finalist wordt verricht door zeven verschillende teams. Deze teams zijn ieder verantwoordelijke voor specifieke projecten die gebaseerd zijn op hun team. Een toelichting over de zeven teams wordt nader behandeld binnen deelvraag 1 van mijn onderzoek.

Met hun passie voor softwareontwikkeling en beheersing van de nieuwste technologieën, bouwen de teams van Finalist snel en zorgvuldig aan hun opdrachten. Het inzetten van Open source technologieën staat bij Finalist centraal.

Finalist streeft ernaar om met hun open IT oplossingen organisaties intenser met elkaar samen te laten werken met hun belangengroepen. Door middel van advies, projecten en beheer realiseert Finalist intelligente websites, maatwerk applicaties en integratieoplossingen. De doelgroepen van Finalist zijn daarom primaire organisaties waarvoor een uitstekende informatievoorziening en intensieve samenwerking belangrijk zijn.

Ik mag mij verblijden met een plaats als afstudeerder binnen de vestiging Rotterdam van Finalist. Het is mijn taak om een onderzoek uit te voeren en hier een advies over uit te brengen.

# 

# 3. De opdracht

De afstudeeropdracht is opgesteld om onderzoek te doen naar een vervangende software distributie tool voor het Webapps team binnen Finalist. Deze vervangende tool zou oplossing moeten bieden aan de knelpunten die zich momenteel afspelen binnen de software distributie.

De opdracht start door de huidige processen die afspelen binnen de software distributie van het Webapps team in kaart te brengen. Wanneer deze processen in kaart zijn gebracht, kan ik de knelpunten die zich voordoen binnen de huidige processen beschrijven en toelichten. Het beschrijven van het ideale toekomstbeeld van het Webapps team is noodzakelijk om eventuele oplossingen voor de knelpunten vast te stellen.

Zodra de huidige situatie in kaart is gebracht dien ik een requirementsrapport op te stellen waarin de requirements beschreven worden waaraan de tool voor de software distributie processen moet voldoen.

Wanneer de requirements opgesteld zijn, dien ik criteria op te stellen voor een longlist. Als de criteria opgesteld is, wordt er een longlist opgesteld met tools welke goede kandidaten zijn voor het Webapps team. Uit de longlist dient één tool te komen welke het meest geschikt lijkt voor het Webapps team. Deze tool zal in praktijk getest worden.

Voor zowel de tools uit de huidige situatie als de meest geschikte tool uit de longlist, worden testcases opgesteld. De testcases zullen in praktijk worden getest.

Tot slot kan, conform de verzamelde gegevens en resultaten een adviesrapport opgesteld worden voor het Webapps team.

# 

# 4. Probleemstelling

De centrale probleemstelling in dit onderzoek wordt samengevat in de hoofdvraag:

*“Welke software distributie tool sluit het beste aan op de software distributie binnen het Webapps team?”*

Om deze hoofdvraag te kunnen beantwoorden zijn er vier deelvragen geformuleerd. Iedere deelvraag heeft zijn eigen bijdrage aan het antwoord op de hoofdvraag.

De deelvragen die beantwoord zullen worden luiden als volgt:

* *Hoe voert het Webapps team software distributie uit?*

Binnen deze deelvraag wil ik antwoord krijgen over de huidige uitvoering van software distributie van het Webapps team. Door de processen binnen de software distributie in detail uit te schrijven, kan er informatie over de knelpunten worden vast gesteld. De knelpunten vormen het huidige probleem van het Webapps team.

* *Welke knelpunten treden op binnen de software distributie van het Webapps team en hoe zijn deze op te lossen en hoe zijn deze op te lossen?*

Binnen deze deelvraag wil ik knelpunten toelichten die zich afspelen binnen de software distributie van het Webapps team. Vervolgens zal een beschrijving van een ideaal toekomstbeeld mogelijkheden moeten bieden om oplossingen te vinden voor de opgestelde knelpunten. Wanneer de oude en ideale toekomst situatie beschreven zijn, worden er requirements opgesteld welke verwerkt worden in een requirementsrapport.

* *Welke software distributie tools zijn geschikt voor het opstellen van de longlist en welke tool is het meest interessant om te testen in praktijk?*

Binnen deze deelvraag wil ik een aantal software distributie tools beschrijven die mogelijk de oplossing kunnen bieden voor de knelpunten uit de vorige deelvraag. Aan de hand van requirements wil ik criteria opstellen waaraan de tools moeten voldoen. Door een literatuuronderzoek te starten, wil ik informatie verzamelen over beschikbare tools en de opgestelde criteria om te bewijzen of de tools wel of niet in staat is hieraan te voldoen.

* *Welke resultaten leveren de beste tool uit de longlist op t.o.v.. Chef en Capistrano in praktijk?*

Binnen deze deelvraag wil ik de meest geschikte software distributie tool uit de vorige deelvraag testen in praktijk. Door middel van verschillende scenario’s die betrekking hebben tot de geselecteerde knelpunten uit deelvraag 2 te verwerken in testcases, hoop ik op een duidelijke manier aan te kunnen tonen welke tool de beste mogelijkheden biedt als oplossing voor het huidige probleem van het Webapps team.

# 5. Aanpak

Om de aanpak van mijn onderzoek te beschrijven heb ik een onderzoeksplan opgesteld dat als "plan van aanpak" kan worden gezien voor dit onderzoek. Zo wordt er beschreven welke onderzoeksmethodes er gebruikt worden om informatie te verzamelen, welke hoofd- en deelvragen ik wil beantwoorden en in welke volgorde ik de deelvragen wil beantwoorden.

Het onderzoeksplan is te vinden in bijlage 10.3.2

Om goed van start te gaan ben ik begonnen met het maken van een planning. Deze wordt nader toegelicht in hoofdstuk 6.1.

Om de juiste feedback te ontvangen en mijn begeleider goed bij mijn opdracht te betrekken, was er minimaal 1x per week een gesprek gepland met mijn begeleider. Binnen dit gesprek heb ik de voortgang, eventuele vragen en de nieuwe werkzaamheden voor een week besproken.

# 6. Voorbereiding onderzoek

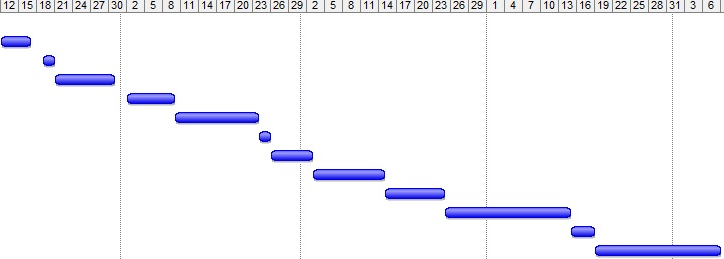
Binnen dit hoofdstuk beschrijf ik het proces van de voorbereiding op mijn onderzoek. Ik laat mijn keuzes binnen het proces naar voren komen en motiveer daarbij waarom ik deze keuzes gemaakt heb.

Omdat de opdracht vanuit het bedrijf nog zeer globaal was, ben ik gestart met het opstellen van vragen om duidelijkheid te verschaffen. De reden waarom de opdracht nog globaal was, kwam omdat de opdracht niet goed opgesteld was en er nog geen juiste afbakening afgesproken was.

Na het opstellen van de vragen, heb ik een gesprek met mijn begeleider gehad om de vragen door te nemen. Het gesprek heeft mij duidelijk gemaakt dat Finalist bestaat uit zeven verschillende teams die te maken hebben met software distributie. Elk team distribueert software op een andere manier en maakt zo gebruik van verschillende processen en tools om software distributie te realiseren. Binnen het gesprek is mij ook duidelijk gemaakt dat een intern onderzoek door middel van het afleggen van interviews met de teams mij duidelijkheid over de processen en gebruikte tools kon geven.

## 6.1 Planning V1.0

Voordat ik begon met het intern onderzoek, leek het mij verstandig om een globale planning te maken van de werkzaamheden om overzicht te krijgen. Ik heb deze planning opgebouwd conform het aantal dagen per activiteit zoals deze beschreven staat binnen mijn goedgekeurd afstudeerplan. De planning ziet er als volgt uit:



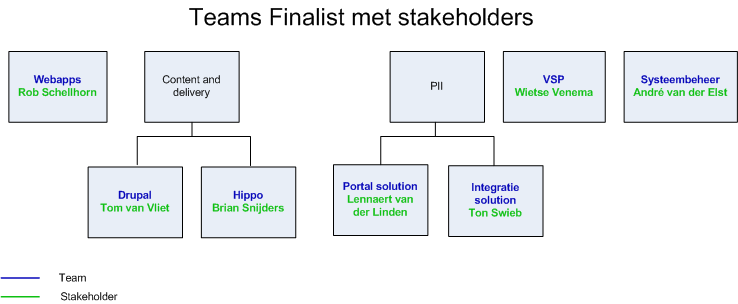
## 6.2 Intern onderzoek

Om stakeholders te verzamelen heb ik een gesprek aangevraagd met mijn begeleider om namen te vragen van werknemers die de beste stakeholders voor mijn intern onderzoek zijn. Hierbij heb ik bewust gevraagd om voor ieder team minimaal één stakeholder te kunnen spreken. Ik wou minimaal één stakeholder uit elk team omdat:

* Ieder team de processen binnen software distributie anders uitvoeren en verschilende tools gebruiken. Het verzamelen van deze gegevens voor elk team levert het meest actuele beeld van software distributie op.
* Ieder team verschillende software ontwikkelt wat voor variatie binnen de processen kan zorgen.
* Ieder team zijn eigen problemen en beperkingen heeft. Dit kan zeer interessant zijn wanneer we zoeken naar een oplossing tussen alle teams.

Mijn begeleider heeft een aantal mogelijke namen opgegeven. Daarbij is de keuze gevallen op de personen die eindverantwoordelijk zijn binnen de teams.

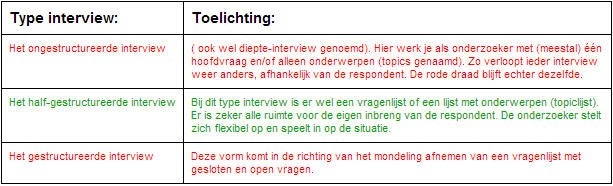
Na het verkrijgen van de namen heb ik contact opgenomen. Het is mij gelukt om met alle stakeholders een afspraak te maken voor een interview over de software distributie binnen het team. Een overzicht van de stakeholders en teams ziet er als volgt uit:



Na het maken van de afspraken moest ik mij voorbereiden op de interviews die ik de komende tijd zou afnemen. Graag wil ik toelichten waarom ik de keuze heb gemaakt om mijn informatie te winnen door middel van het afnemen van interviews bij mijn stakeholders.

Ik heb bewust gekozen voor het afnemen van interviews omdat deze methode van dataverzameling een open en gestructureerde inhoud naar de stakeholder biedt. Met “open” wordt bedoeld dat de stakeholder zelf ook invulling/aanvulling kan geven aan de interviewer. De “gestructureerde inhoud” wil zeggen dat het interview verdeeld is in drie delen namelijk begin, midden (de kern) en een einde [Nel Verhoeven].

Binnen het afnemen van interviews zijn drie mogelijkheden beschikbaar. Ik licht de drie mogelijkheden toe en bevestig mijn keuze:



[Nel Verhoeven]

Mijn keuze was gevallen op het uitvoeren van een half-gestructureerd interview. Zoals te lezen is in de tabel op de vorige bladzijde, is het voorbereiden van vragen voor de stakeholders van toepassing. Ter voorbereiding op de interviews heb ik een lijst met vragen opgesteld. Deze vragenlijst is te vinden als bijlage in hoofdstuk 10.3.3.

Bij het afnemen van de interviews, heb ik ervoor gekozen om geluidsopnames te maken. Uiteraard heb ik alle stakeholders eerst om toestemming gevraagd. Deze geluidsopnames heb ik gebruikt om het interview zo goed mogelijk samen te kunnen vatten. Na het maken van een samenvatting, heb ik deze laten doornemen door de stakeholder om te bevestigen of ik de informatie juist begrepen heb en ik geen belangrijke informatie vergeten ben.

De samenvattingen van alle stakeholder gesprekken vormden samen de informatie die ik nodig had om mijn vooronderzoek uit te voeren. De samenvattingen van de interviews zijn te vinden als bron in hoofdstuk 10.3.4

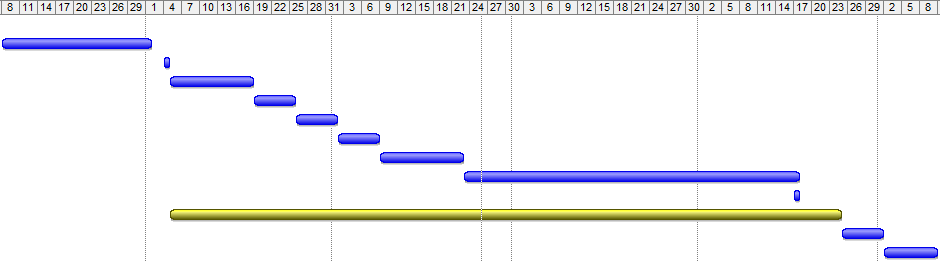
## 

## 6.3 Planning V2.0

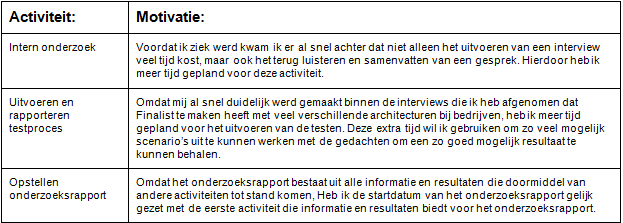
Helaas werd ik binnen mijn afstudeertraject ziek waardoor ik een uitstel van tien weken voor mijn onderzoek heb gekregen. Op verzoek van mijn begeleidend examinator H.P Weenink, heb ik een nieuwe planning opgesteld conform de nieuwe deadline. Deze planning ziet er als volgt uit:







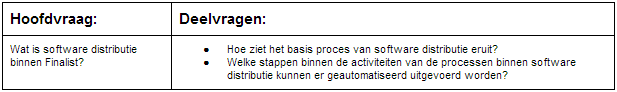
Ten opzichte van planning V1.0 zijn er een aantal activiteiten sterk toegenomen in het aantal dagen beschikbare tijd. Graag motiveer ik waarom ik dit heb gewijzigd:



## 6.4 Vooronderzoek

Direct na het verzamelen van de juiste informatie binnen mijn intern onderzoek, ben ik gestart met mijn vooronderzoek.

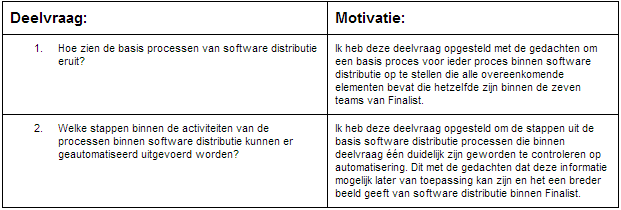
Het eerste wat ik uitgevoerd heb voor mijn vooronderzoek is het formuleren van een hoofdvraag met bijbehorende deelvragen. Echter omdat de opdracht vanuit het bedrijf nog erg onduidelijk was voor mij, was het zoeken naar de juiste hoofdvraag met bijbehorende deelvragen niet eenvoudig. De uiteindelijke hoofdvraag met bijbehorende deelvragen die ik geformuleerd heb zijn als volgt:



*Motivatie hoofdvraag:*

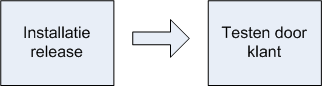
Binnen het vooronderzoek wil ik verduidelijking krijgen over wat software distributie nou binnen Finalist betekent. Uit de conclusie van de hoofdvraag wil ik een basis proces zien die opgesteld is op basis van de informatie die verkregen is uit het intern onderzoek. Daarnaast moet de conclusie mij antwoord kunnen geven op welke stappen binnen het basis proces automatisering uitgevoerd kan worden.

*Motivatie deelvragen:*



Na het bepalen van de hoofdvraag met bijbehorende deelvragen, ben ik gestart met het uitwerken van deelvraag 1. Binnen deze deelvraag heb ik de basis software distributie processen kunnen vaststellen en uitgeschreven. Als resultaat zijn er twee processen beschreven, namelijk voor software deployment en server provisioning(3). Het resultaat van deze basis processen uit deelvraag ziet er als volgt uit:

*Software deployment proces:*



De activiteiten hebben de volgende stappen:

*Activiteit installatie release:*

* De nieuwe release wordt gedownload door de server van de klant uit de stash git repository van Finalist
* Database wijzigingen doorvoeren
* Dependencies(2) updaten

*Activiteit testen door klant:*

* Uitvoeren van testen door de klant

*Server provisioning proces:*



De stappen die bij de activiteiten horen zijn als volgt:

Verzoek indienen:

* Verzoek indienen bij systeembeheerder voor server.

Installatie server:

* Systeembeheerder kopieert template van de server met applicaties (interne host).
* Systeembeheerder voert besturingssysteem installatie uit (externe host).

Installatie benodigde applicaties:

* Systeembeheerder installeert applicaties welke opgegeven zijn binnen het ingediende verzoek (externe host).
* Medewerkers team installeren zelf applicaties via een ssh-verbinding naar de server.

Na de beantwoording van de deelvragen heb ik een conclusie kunnen opstellen die antwoord geeft op de hoofdvraag “Wat is software distributie binnen Finalist?”. Om de volledige uitwerking van het vooronderzoek te bekijken verwijs ik u naar de bijlage in hoofdstuk 10.3.1.

## 6.5 Onderzoeksplan

Na het uitvoeren van het vooronderzoek, ben ik begonnen met het maken van een onderzoeksplan. Het onderzoeksplan kan worden gezien als “plan van aanpak” voor het daadwerkelijke onderzoek.

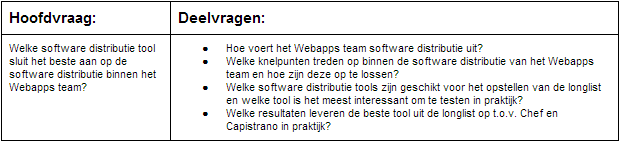
Binnen het onderzoeksplan ben ik begonnen met het uitschrijven van het onderzoekskader. Het onderzoekskader bestaat uit:

* Aanleiding
* Probleemstelling
* Doelstelling
* Hoofdvraag
* Deelvragen
* Samenhang deelvragen
* Aanpak
* Kwaliteit waarborging

Binnen de probleembeschrijving heb ik een aantal knelpunten opgesteld welke al duidelijk waren geworden binnen het half gestructureerde interview met het Webapps team.

Bij het opstellen van een hoofdvraag met bijbehorende deelvragen heb ik, samen met mijn begeleider, afgesproken om het onderzoek af te bakenen tot het Webapps team binnen Finalist. Ook het onderwerp “server provisioning” welke besproken staat in mijn afstudeerplan is gewijzigd tot “software distributie”. Deze wijziging heeft plaats gevonden omdat het onderzoek naar server provisioning te klein zou zijn voor mijn afstudeertraject.

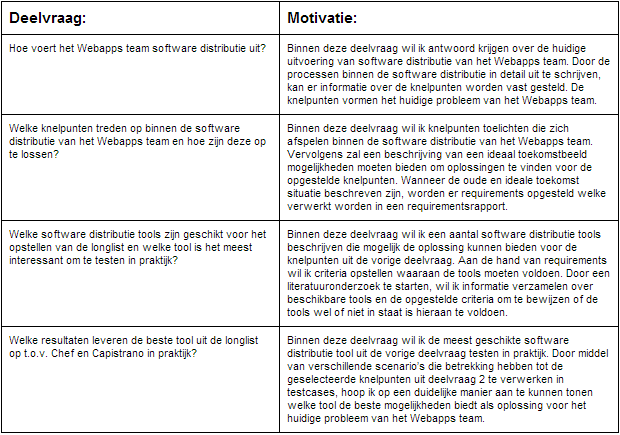
De hoofdvraag met bijbehorende deelvragen die ik opgesteld had voor het daadwerkelijke onderzoek zijn als volgt:



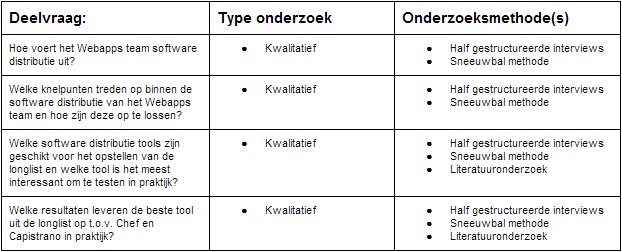
*Motivatie hoofdvraag:*

Binnen mijn onderzoek wil ik op zoek gaan naar een nieuwe software distributie tool die ingezet kan worden om het huidige probleem van het Webapps team te verbeteren. Mijn advies moet kunnen bevestigen of één / of een combinatie van de software distributie tool(s) oplossing bieden voor het huidige probleem van het Webapps team.

*Motivatie deelvragen:*



Binnen het onderzoeksplan heb ik ook de aanpak beschreven. Binnen de aanpak worden onderzoeksmethodes besproken. De onderzoekmethodes die ik gekozen heb zijn als volgt:



Over deze onderzoeksmethodes wil ik graag mijn motivatie uitleggen.

*Half gestructureerde Interview methode*

Ik heb bewust gekozen om gebruik te maken van de half gestructureerde interview methode voor het verzamelen van informatie. Deze keuze heb ik gemaakt omdat de methode de optie biedt om een open interview te houden waarbij zowel opgestelde vragen kunnen worden gesteld als eigen inbreng van de stakeholder een rol kunnen spelen.

*Sneeuwbal methode*

Ik heb bewust gekozen voor de sneeuwbal methode voor het verzamelen van stakeholders. Binnen de sneeuwbal methode spreek je één persoon of een kleine groep respondenten aan om aan meer stakeholders te komen. Mijn toepassing van de sneeuwbal methode is geweest naar mijn begeleider ( ook stakeholder ) die vervolgens namen gaf van mogelijke stakeholders voor mijn onderzoek. Deze keuze heb ik gemaakt omdat een stakeholder binnen het bedrijf weet wat zijn collega’s voor functie hebben en daarom beter kon beslissen voor mij welke personen mogelijk de beste stakeholders waren.

*Literatuuronderzoek*

Ik heb er bewust voor gekozen om gebruik te maken van een literatuuronderzoek. Door gebruik te maken van het literatuuronderzoek wil ik gegevens vinden over software distributie tools die momenteel gebruikt worden. Deze keuze heb ik gemaakt omdat het lastig is deze informatie te winnen binnen het bedrijf Finalist en een literatuuronderzoek een breed oppervlak biedt om onderzoek in te verrichten zoals internet en boeken.

Voor meer inhoudelijke informatie over de methodes verwijs ik naar bron [Nel Verhoeven].

Om de volledige uitwerking van het onderzoeksplan te bekijken verwijs ik u naar de bijlage in hoofdstuk 10.3.2.

Omdat de hoofdvraag met bijbehorende deelvragen in het onderzoeksrapport afwijken van het onderwerp die beschreven staat in het afstudeerplan, zijn er beroepstaken gewijzigd. De huidige beroepstaken die beschreven staan in mijn afstudeerplan zien er als volgt uit:

* 1.4 Uitvoeren analyse door definitie van requirements
* 3.4 Initiëren en plannen van het testproces
* 3.5 Uitvoeren van en rapporteren over het testproces
* 4.4 Beheren en distribueren van software

Omdat mijn hoofdvraag beantwoordt zal worden met een keuze voor een software distributie tool en deze gekozen zal worden uit een longlist, is de beroepstaak “1.3 selecteren van standaardsoftware” van toepassing.

Beroepstaak 3.4 zal afvallen omdat deze gebaseerd is op testen binnen software ontwikkeling. Omdat ik mij bezig zal houden met bestaande software is dit niet meer relevant.

Beroepstaak “4.4 Beheren en distribueren van software” komt ook meer tot zijn recht omdat mijn onderwerp is veranderd naar software distributie in plaats van server provisioning.

De beroepstaken binnen mijn onderzoek zien er nu als volgt uit:

* 1.3 selecteren van standaardsoftware
* 1.4 Uitvoeren analyse door definitie van requirements
* 3.5 Uitvoeren van en rapporteren over het testproces
* 4.4 Beheren en distribueren van software

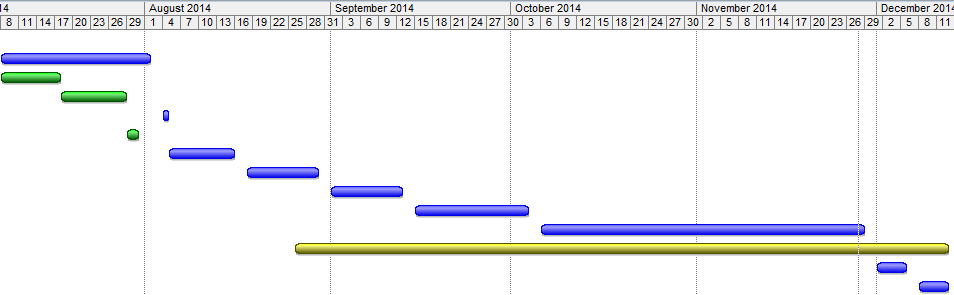
## 

## 6.6 Planning V3.0

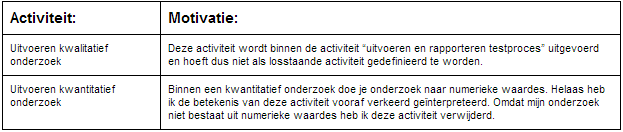
Na het maken van het onderzoeksplan, heb ik op advies van mijn begeleider een nieuwe planning gemaakt. De motivatie hiervoor was dat het echte onderzoek nu van start zou gaan en mijn begeleider goed inzicht wilde hebben in de geplande activiteiten en geplande einddatums. De planning ziet er als volgt uit:



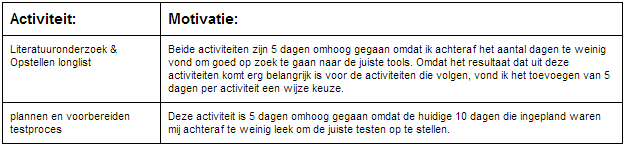




Ten opzichte van planning V2.0 zijn er twee activiteiten verwijderd. Graag motiveer ik waarom ik deze activiteiten heb verwijderd:



Ook zijn er een aantal activiteiten veranderd wat betreft het aantal dagen voor uitvoering. Hieronder mijn motivatie van de grootste veranderingen:



# 7. Uitvoering onderzoek

Binnen het onderzoeksrapport ben ik begonnen met het uitschrijven van de inleiding. De inleiding bestaat uit:

* Kennisgebied
* Voorbereiding
* Afbakening
* Doelstelling
* Hoofdvraag
* Deelvragen
* Aanpak

## 7.1 Deelvraag 1

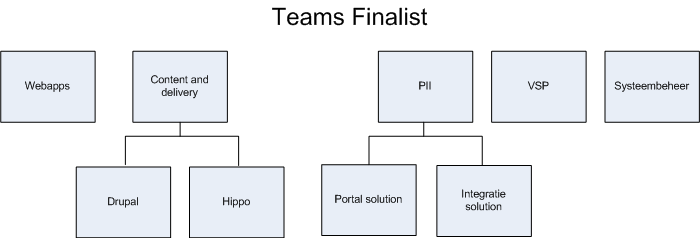
Vervolgens ben ik begonnen met het beantwoorden van de deelvragen. De eerste deelvraag luidt als volgt:

*“Hoe voert het Webapps team software distributie uit?”*

Om deze deelvraag te kunnen beantwoordde, was het noodzakelijk om goed na te denken over de opbouw van de deelvraag. Het proces over de uitvoering van deelvraag 1 wordt nu toegelicht.

### 7.1.1 Het bespreken van de infrastructuur binnen Finalist

Omdat de eerste deelvraag antwoord moest gaan geven over de software distributie processen binnen een team van Finalist, ben ik gestart met het uitleggen van de infrastructuur van Finalist. Ik ben begonnen met het moduleren van alle zeven teams binnen Finalist en heb deze vervolgens individueel toegelicht. Mijn motivatie was om een duidelijk beeld te verschaffen aan de lezer wat het Webapps team voor rol heeft binnen de organisatie. De zeven teams binnen Finalist zijn als volgt:



*Webapps team:*

Het Webapps team bestaat momenteel uit drie werknemers waaronder mijn begeleider Rob Schellhorn. Binnenkort zal het Webapps team mogelijk uitbreiden tot zes werknemers. Binnen het Webapps team worden voornamelijk Ruby on rails projecten ontwikkeld. Binnen mijn onderzoek wordt eerst vooronderzoek gedaan naar alle teams. Later vindt er een afbakening plaats waarbij het onderzoek zich richt tot alleen het Webapps team.

*Content and Delivery team:*

Het Content and Delivery team is onder te verdelen in twee subteams namelijk Hippo en Drupal. De namen van de subteams geven eigenlijk al aan wat voor projecten er ontwikkeld worden. Zo wordt er binnen het Hippo subteam Hippo CMS applicaties ontwikkeld en binnen het Drupal subteam Drupal CMS applicaties.

*Portals, Information & Identity (PII) team:*

Ook het Portals, information & Identity team is onder te verdelen in twee subteams namelijk Portal solution en Integratie solution. Binnen het portal solution team worden voornamelijk php maatwerk en Drupal projecten ontwikkeld.

Binnen het integratie solution team, worden ESB (Enterprise service bus) ontwikkeld. De ESB dient als koppelstuk tussen twee of meerdere applicaties. Het integratie team maakt voornamelijk gebruik van Jboss fuse applicatie servers ( is een platform voor ESB’s ). De meeste projecten worden ontwikkeld in Java.

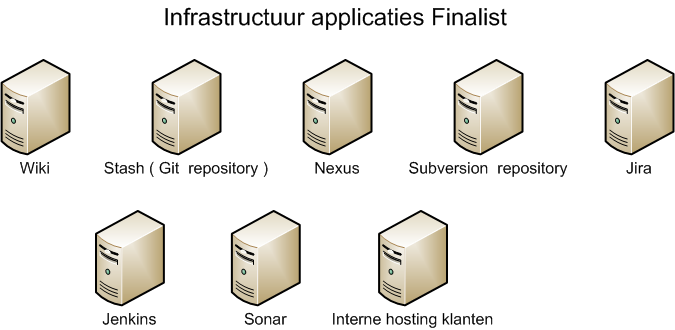
*VSP team:*

Het VSP team houdt zich voornamelijk bezig met het ontwikkelen van applicaties voor het bedrijf Netwerk VSP. Netwerk VSP is verantwoordelijk voor het verspreiden van folderpakketten naar alle huizen in Nederland. Momenteel vind er een onderzoek plaats naar optimalisatie van het produceerproces van de samengestelde folderpakketten. De meeste projecten worden ontwikkeld in Java.

*Systeembeheer team:*

De systeembeheerders houden zich voornamelijk bezig om alle werknemers binnen Finalist te kunnen laten werken. Zo zijn ze verantwoordelijk om de werknemers te voorzien van ontwikkelomgevingen en testomgevingen. Ook verzorgen de systeembeheerders de server provisioning van nieuwe servers voor zowel interne hosting als externe hosting. Kortom, de systeembeheerders verzorgen de OTAP omgevingen.

Na het beschrijven van de teams heb ik de infrastructuur van de servers van Finalist gemoduleerd. Bij de modulatie staat beschreven welke applicaties er beschikbaar zijn op een server en heb deze vervolgens toegelicht. Mijn motivatie om de infrastructuur van de servers van Finalist met applicaties op te nemen was om vooraf duidelijk te maken hoe sommige applicaties heten die gebruikt worden binnen het proces. Het resultaat van de infrastructuur van de servers met applicaties is als volgt:



*Wiki:*

De wiki applicatie wordt gebruikt voor het registreren van allerlei gegevens. Zo kun je de benodigde gegevens voor alle OTAP servers vinden maar ook procedures, roadmaps, documentaties en systeem installatie gegevens. Kortom alle benodigde gegevens voor een project kun je hier terug vinden.

*Stash git repository:*

Binnen de stash git repository staat de laatste broncode van alle projecten opgeslagen. Binnen stash kun je terug lezen welke commits er gedaan zijn en door wie ze uitgevoerd zijn. Binnen een commit kun je eenvoudig zien welke regels code er gewijzigd zijn.

*Nexus:*

Binnen Nexus staan alle projecten opgeslagen die gecompileerd zijn, succesvol getest zijn met integratie testen en voorzien zijn van de juiste dependencies. Nexus wordt gebruikt voor Java projecten.

*Subversion repository:*

De subversion repository werd voorheen gebruikt om de laatste broncode van projecten op te slaan. Tegenwoordig is de subversion repository vervangen door de stash git repository. De subversion repository wordt enkel nog gebruikt voor een paar projecten.

*Jira:*

Jira is een applicatie die gebruikt wordt voor het managen van projecten. Binnen Jira worden incidenten/issues aangemaakt welke gekoppeld kunnen worden aan een project. Een medewerker kan zo’n incident/issue oppakken en deze koppelen aan een release. Binnen Jira is het dus eenvoudig om te zien binnen welke release welke incidenten zijn opgelost en door wie ze zijn uitgevoerd. Jira wordt voornamelijk gebruikt door de projectmanagers.

*Jenkins:*

Jenkins is een “Continuous integration(4)” tool welke gebruikt wordt om geautomatiseerd nieuwe releases uit te brengen naar de klant. Binnen Jenkins kunnen verschillende stappen uitgevoerd worden zoals Integratietesten, dependencies installeren, compileren/builden software en de distributie van een software release. Binnen Jenkins is het mogelijk om een Jenkins taak te schrijven waarin een stapsgewijs script geschreven staat.

Binnen Jenkins is het ook mogelijk om “build triggers” in te stellen. Een build trigger is een tijdsduur die gehanteerd wordt door Jenkins om de taak geautomatiseerd uit te voeren. Aan een build trigger kun je voorwaarden instellen waaraan moet worden voldaan voordat de taak daadwerkelijk uitgevoerd wordt. Wanneer de tijdsduur van de build trigger verlopen is en de build trigger voldoet aan de ingestelde voorwaarden, voert Jenkins de taak geautomatiseerd uit.

*Sonar:*

Sonar is een applicatie die gebruikt wordt om “static code analysis(5)” uit te laten voeren. Binnen Sonar kun je eisen samenstellen waarop de applicaties moet controleren. Sonar wordt veel gebruikt om code die mogelijke problemen kunnen veroorzaken op te sporen en optimalisatie van de code te realiseren.

*Interne hosting klanten:*

Wanneer een klant ervoor kiest om de hosting van zijn applicatie door Finalist te laten verzorgen, zal deze gehost worden binnen de interne hosting van Finalist. Daarnaast wordt de interne hosting gebruikt om testservers te draaien.

### 7.1.2 Gedetailleerde processen binnen software distributie van het Webapps team.

Toen de infrastructuur van Finalist duidelijk omschreven was, ben ik begonnen met het uitschrijven van de processen binnen software distributie van het Webapps team. In totaal zijn dit twee processen geworden namelijk:

* Server provisioning
* Software deployment

Mijn motivatie om alle twee de processen uit te schrijven was om zo duidelijk mogelijk software distributie te beschrijven van het begin tot het einde.

*Server provisioning:*

Binnen het proces “server provisioning” heb ik als eerst het basis server provisioning uit het vooronderzoek herhaald. Mijn motivatie om dit te doen was om te kunnen vergelijken welke stappen er extra uitgevoerd worden binnen het Webapps team ten opzichte van het basis server provisioning proces.

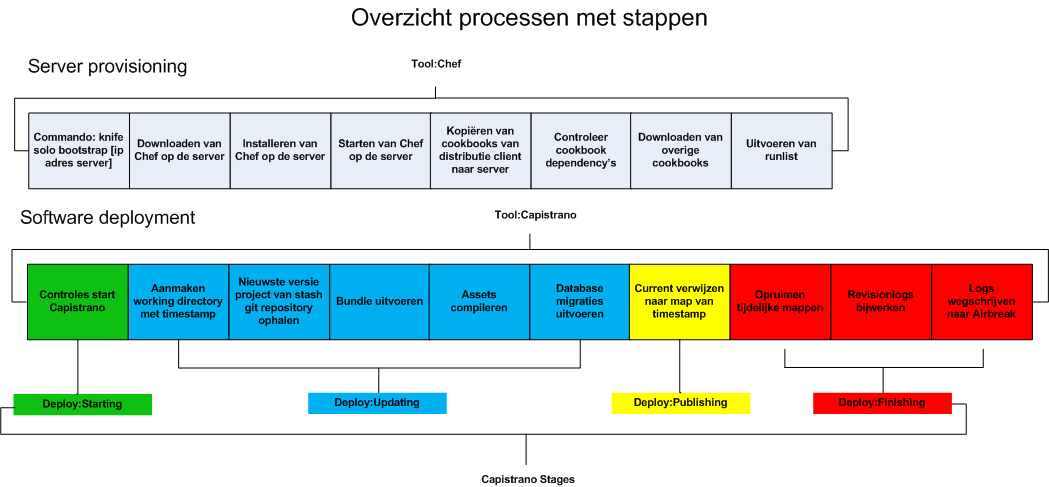
Het resultaat wees uit dat het Webapps team geen extra stappen onderneemt. Echter was de laatste activiteit niet in detail uitgeschreven in het vooronderzoek omdat deze stap gebruikt maakt van een server provisioning tool welke in ieder team anders is. Ik heb de structuur en de stappen welke de server provisioning tool “Chef” van het Webapps team doorloopt om server provisioning te realiseren in detail uitgeschreven. Mijn motivatie om dit te doen was hoofdzakelijk voor mijzelf om de server provisioning tool te leren gebruiken en te begrijpen maar ook om de lezer een duidelijk inzicht te geven in de structuur en werking van de server provisioning tool.

*Software deployment:*

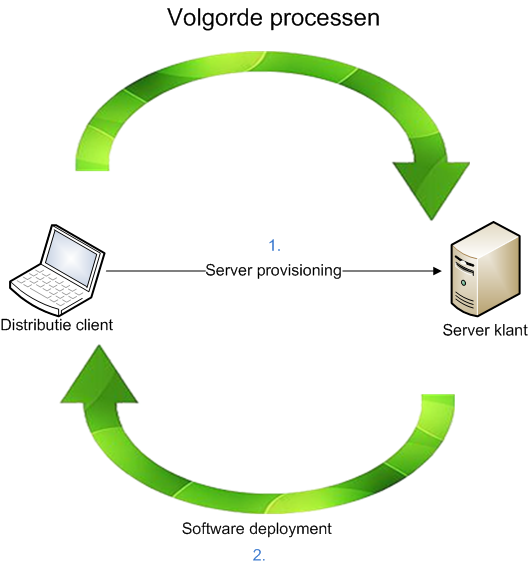
Binnen het proces “software deployment” heb ik eerst het basis software deployment proces uit het vooronderzoek herhaald. Mijn motivatie om dit te doen was om te kunnen vergelijken welke stappen er extra uitgevoerd worden binnen het Webapps team ten opzichte van het basis software deployment proces.

Het resultaat wees uit dat het Webapps team twee extra stappen toepast en één stap binnen een andere activiteit plaats vindt in vergelijking met het basis software deployment proces. Om het software deployment proces uit te voeren, wordt er gebruik gemaakt van de software deployment tool “Capistrano (V3.0)”. Ik heb de structuur en de stappen welke Capistrano V3.0 doorloopt om software deployment te realiseren in detail uitgeschreven. Mijn motivatie om dit te doen was hoofdzakelijk voor mijzelf om de software deployment tool te leren gebruiken en te begrijpen maar ook om de lezer een duidelijk inzicht te geven in de structuur en werking van de software deployment tool.

Tot slot laat ik de twee processen in één overzicht zien met de stappen die binnen het proces uitgevoerd worden en beschrijf ik de volgorde waarop de processen uitgevoerd worden. Mijn motivatie hiervoor is dat één overzicht duidelijk de opgeleverde resultaten laat zien en de volgorde van de processen net even wat extra duidelijkheid aan de lezer biedt. Het overzicht ziet er als volgt uit:



De volgorde van de processen zien er als volgt uit:



## 

*1. Server provisioning*

Dit proces wordt als eerste uitgevoerd en zal eenmalig plaatsvinden. De reden waarom dit proces eenmalig uitgevoerd wordt komt omdat de server eenmalig ingericht moet worden met de juiste provisioning dependencies.

*2. Software deployment*

Dit proces wordt als tweede uitgevoerd en zal herhalend plaatsvinden. De reden waarom dit proces herhalend uitgevoerd wordt komt omdat er steeds nieuwe releases gedistribueerd worden op de server.

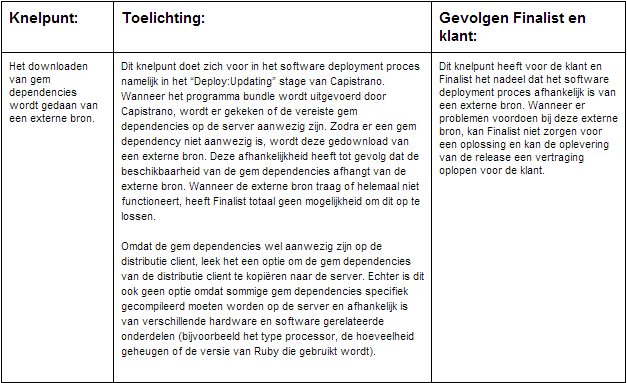
## 7.2 Deelvraag 2

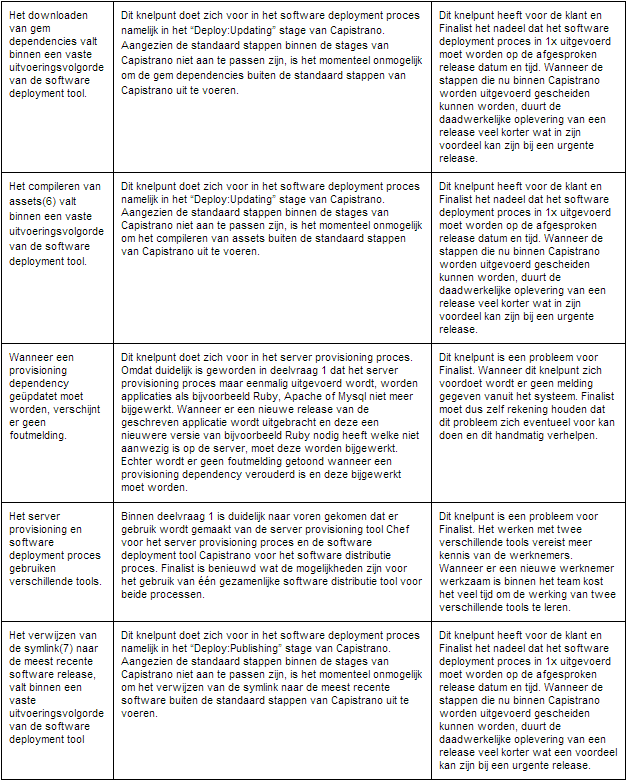
Na het uitwerken van deelvraag 1 ben ik gestart met het beantwoorden van deelvraag 2 in het onderzoeksrapport. Deze deelvraag luidt als volgt:

*“Welke knelpunten treden op binnen de software distributie van het Webapps team en hoe zijn deze op te lossen en hoe zijn deze op te lossen?”*

Om deze deelvraag te kunnen beantwoorden, was het noodzakelijk om goed na te denken over de opbouw van de deelvraag. Het proces over de uitvoering van deelvraag 2 wordt nu toegelicht.

### 7.2.1 Toelichting knelpunten Webapps team

Omdat de huidige processen in kaart waren gebracht binnen deelvraag 1, heb ik meer inhoudelijke informatie kunnen verschaffen over de knelpunten die vooraf opgesteld waren in het hoofdstuk “probleemstelling” binnen mijn onderzoeksplan. Hierdoor ben ik deelvraag 2 begonnen met het toelichten van de opgestelde knelpunten aan de hand van de nieuwe informatie uit deelvraag 1. Mijn motivatie om de knelpunten dieper toe te lichten was om het probleem van het Webapps team goed te begrijpen voor ik een toekomstige situatie zou beschrijven. Wanneer de knelpunten namelijk goed duidelijk zijn, is het makkelijker te begrijpen of een toekomstige situatie knelpunten oplost. De knelpunten zijn als volgt:  
  




### 7.2.2 Beschrijven van ideaal toekomstbeeld Webapps team

Na het toelichten van de knelpunten ben ik begonnen met het uitschrijven van het ideale toekomstbeeld van het Webapps team. Door middel van een interview ben ik erachter gekomen welk ideaal beeld het Webapps team in gedachten had om software distributie in de toekomst uit te kunnen voeren. Uit dit interview zijn drie concepten gekomen namelijk:

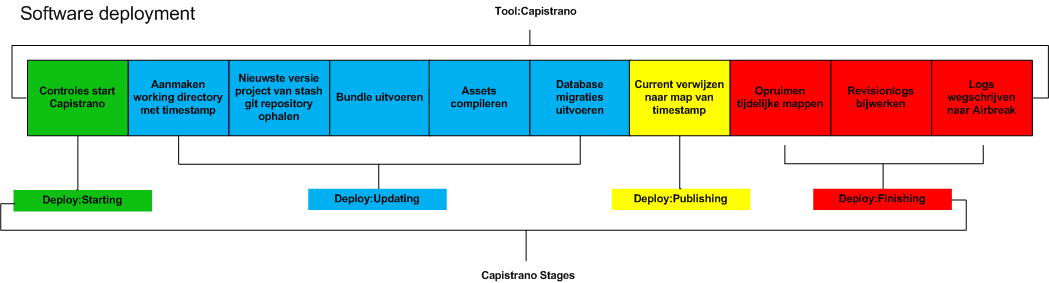
* Prepare deployment
* Foutmelding tonen bij het noodzakelijk updaten van provisioning dependencies
* Server provisioning en software deployment processen realiseerbaar binnen één software distributie tool

*Prepare deployment:*

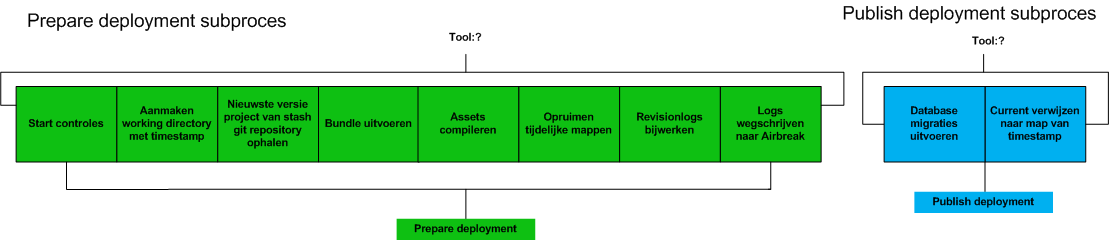
Het prepare deployment concept beschrijft een verandering in het software deployment proces waarbij een proces ontstaat met twee subprocessen. Mijn motivatie om het prepare deployment concept uit te werken in het onderzoeksrapport was omdat me al snel duidelijk werd dat dit concept meerdere knelpunten volledig zou kunnen oplossen en zeer interessant was om goed uit te schrijven.

Om het beeld wat duidelijker te maken laat ik graag het huidige software deployment proces en het prepare deployment concept naast elkaar zien:

Huidige software deployment proces:

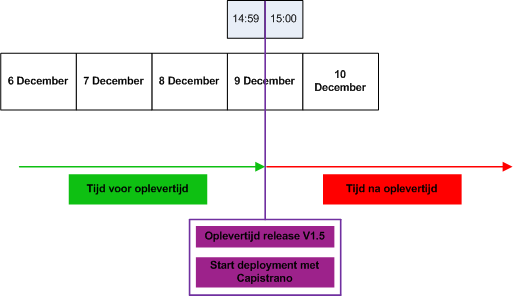


Prepare deployment concept:

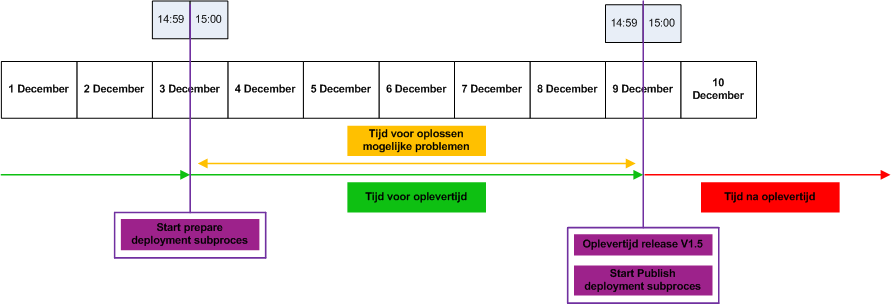


Binnen het huidige software deployment proces is te zien dat de tool Capistrano één proces doorloopt welke bestaat uit 4 stages. Het grote verschil binnen het prepare deployment concept is dat het proces bestaat uit twee subprocessen namelijk prepare deployment en publish deployment.

Binnen het onderzoeksrapport is een scenario beschreven om duidelijkheid te geven over de werking van het prepare deployment concept ten opzichte van het huidige software deployment proces. Binnen het huidige software deployment proces, wordt de hele oplevering van een release op de oplevertijd uitgevoerd. Wanneer er iets mis gaat binnen het proces, zal het probleem opgelost moeten worden na de geplande oplevertijd.



Bij het prepare deployment concept is het uitbrengen van een release verdeeld in twee subprocessen. Binnen het eerste subproces wordt de release uitgebracht op de server van de klant. Mochten er problemen optreden, dan is er nog tijd om deze problemen op te lossen voor de daadwerkelijke oplevertijd. Binnen het laatste subproces worden alleen nog de eventuele database migraties en de symlink verwezen naar de laatste release.



Om het hele scenario en de mogelijk opgeloste knelpunten te lezen verwijs ik naar het onderzoeksrapport in de bijlage in hoofdstuk 10.3.5.

*Foutmelding tonen bij het noodzakelijk updaten van provisioning dependencies:*

Binnen dit concept heb ik opties opgesteld die oplossing bieden voor één van de opgestelde knelpunten. Mijn motivatie was om de mogelijke opties goed uit te schrijven zodat ik duidelijk een keuze kon maken welke optie het beste was. De motivatie van mijn keuze is uitgewerkt in het onderzoeksrapport.

*Server provisioning en software deployment processen realiseerbaar binnen één software distributie tool:*

Binnen dit concept heb ik opties opgesteld die een oplossing bieden voor één van de opgestelde knelpunten. Mijn motivatie was om de mogelijke opties goed uit te schrijven zodat ik duidelijk een keuze kon maken welke optie het beste was. Helaas bleek dat ik nog geen keuze kon maken uit de mogelijke opties in verband met missende resultaten.

Ik heb de concepten opgenomen in het onderzoeksrapport omdat ze mogelijk een oplossing kunnen bieden voor de knelpunten en zo goede requirements vormen.

### 7.2.3 Requirementsrapport

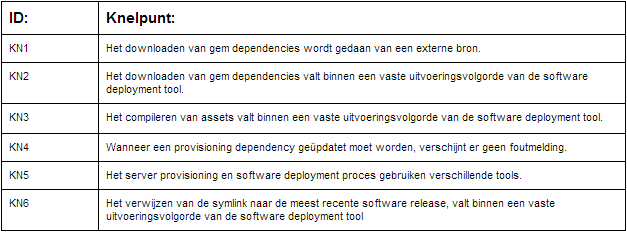
Nadat ik genoeg informatie had verzameld ben ik gestart met het opstellen van de requirements waaraan de toekomstige software distributie tool moet voldoen. De requirements die ik heb opgesteld zijn afkomstig uit drie verschillende bronnen namelijk:

* Stakeholder gesprekken
* Resultaten van deelvraag één uit het onderzoeksrapport
* Beschrijving van het ideale toekomstbeeld van het Webapps team uit het onderzoeksrapport

Mijn motivatie om de requirements uit verschillende bronnen te halen was omdat alle gebruikten bronnen verschillende informatie bevatten.

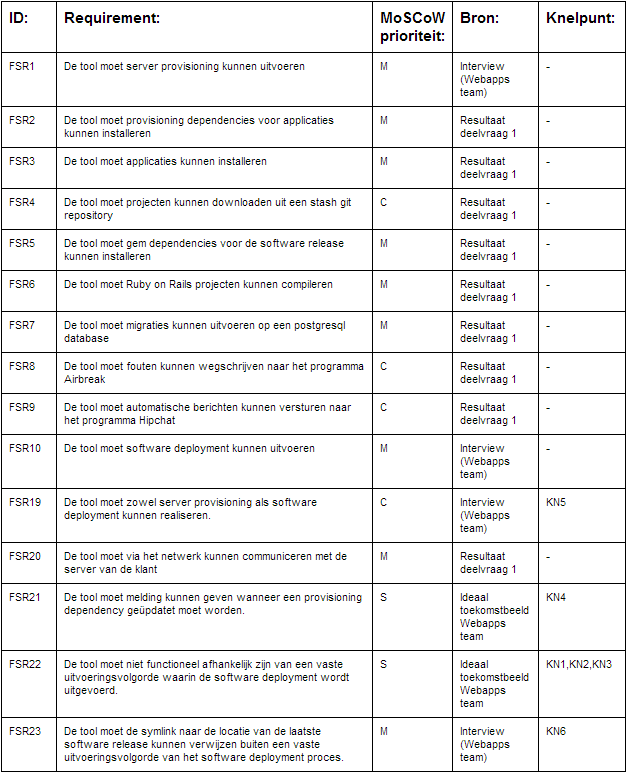
De opgestelde requirements zijn voorzien van een uniek ID waardoor ze te traceren zijn. Daarnaast staat bij iedere requirement vermeld of deze gekoppeld kan worden aan een knelpunt en zijn de requirements geprioriteerd volgens de MoSCoW() methode.

Om de knelpunten te kunnen koppelen aan requirements, zijn deze ook voorzien van een uniek ID. Het overzicht van de knelpunten met uniek ID zijn als volgt.

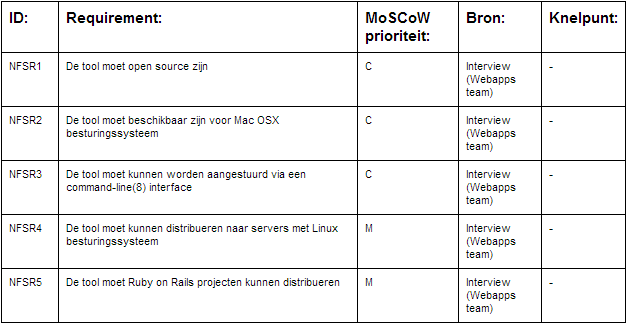


De requirements zijn opgedeeld in Functional software requirements, Non-functional software requirements en user requirements. De requirements zijn als volgt:

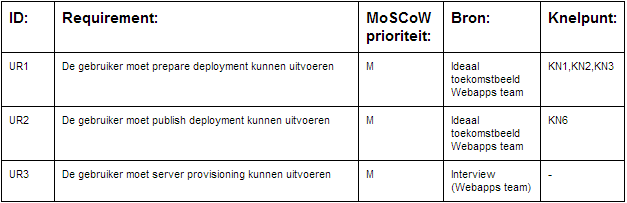
*Functional software requirement:*



*Non-functional software requirement:*

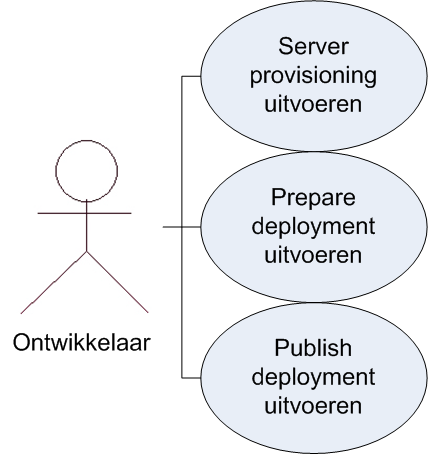


*User requirement:*

  
De prioriteit (M)ust is bij de functional software requirements toegewezen aan requirements die de noodzakelijke basis functionaliteiten beschrijven. Zonder één van deze requirements is het onmogelijk om het server provisioning of software deployment proces uit te voeren. Enkel alleen functional software requirement FSR23 is een requirement met de (M)ust have prioriteit zonder een noodzakelijke basis functionaliteit te zijn. Deze requirement heeft een knelpunt gekoppeld met een hoge prioriteit. Samen in overleg met mijn begeleider hebben we besloten om deze requirement ook een (M)ust have prioriteit te geven. Alle overige requirements hebben in overleg met mijn begeleider een prioriteit toegewezen gekregen.

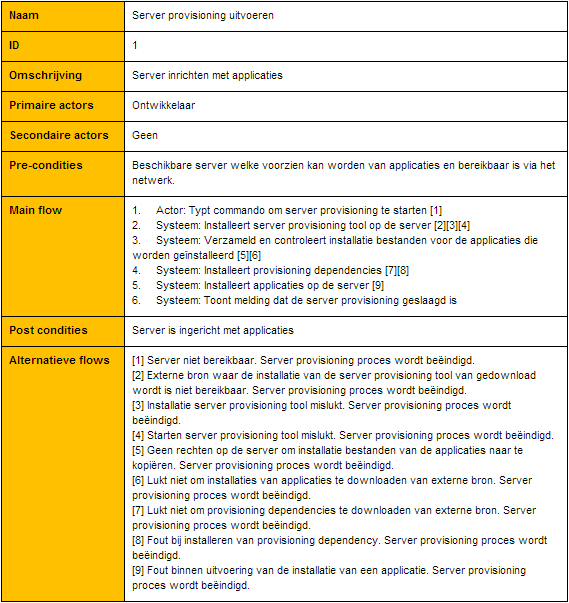
Voor volledige inzage van het requirementsrapport, verwijs ik naar de bijlage in hoofdstuk 10.3.6.

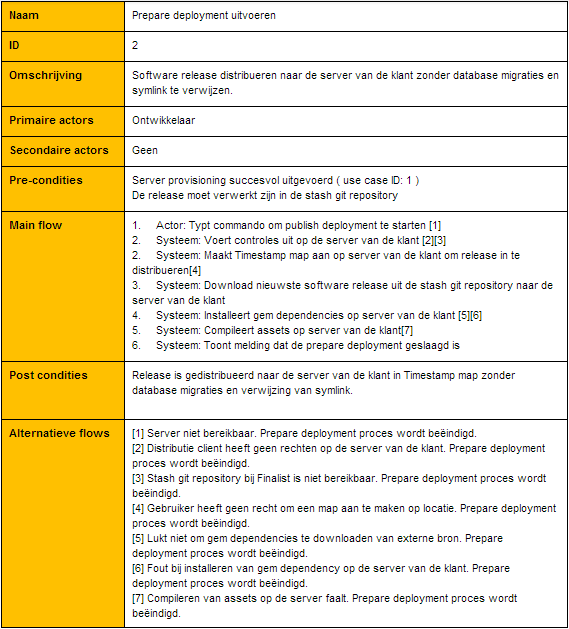
Na het opstellen van de requirements, ben ik gestart met het maken van een use case diagram. Het use case diagram ziet er als volgt uit:

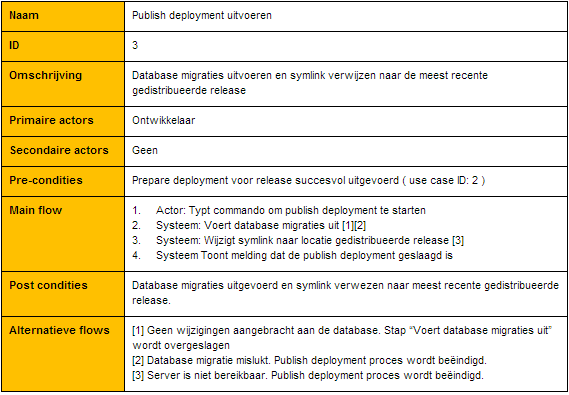


Tot slot heb ik voor iedere use case een beschrijving gemaakt. De use case beschrijvingen zijn als volgt:

### 





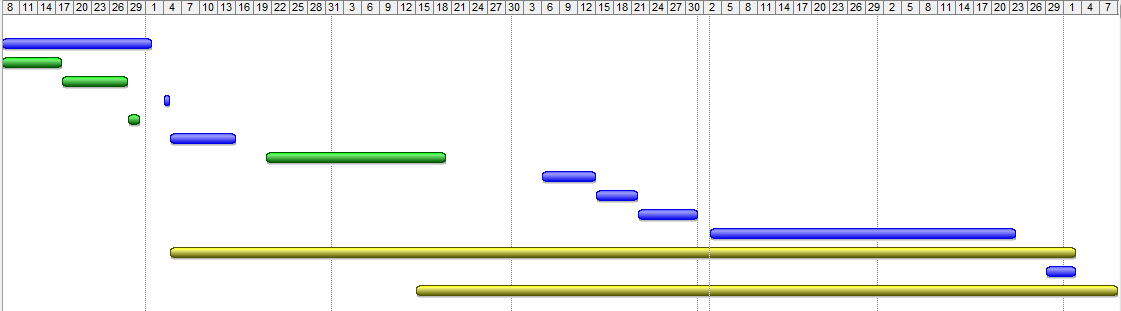


### 7.2.4 Planning V4.0

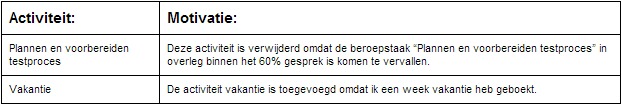
Op 8 oktober heb ik mijn 60% gesprek gehouden met mijn eerste afstudeerbegeleider H.P Weenink. Binnen dit gesprek is naar voren gekomen dat mijn uiteindelijke opleverdatum niet 15-12-2014 was maar 09-01-2015. Conform deze nieuwe data heb ik een nieuwe planning opgesteld.







Ten opzichte van planning V3.0 is er één activiteit verwijderd en één activiteit toegevoegd aan de planning. Graag motiveer ik waarom dit toegepast is:



Binnen de planning is duidelijk te zien dat de activiteit “Opstellen requirements” uitgelopen is ten opzichte van de geplande einddatum. Dit komt omdat het uitwerken van de deelvragen in het onderzoeksrapport ontzettend veel tijd heeft gekost.

## 7.3 Deelvraag 3

Na het uitwerken van deelvraag 2 ben ik door gegaan met het beantwoorden van deelvraag 3. Deze deelvraag luidt als volgt:

*“Welke software distributie tools zijn geschikt voor het opstellen van de longlist en welke tool is het meest interessant om te testen in praktijk?”*

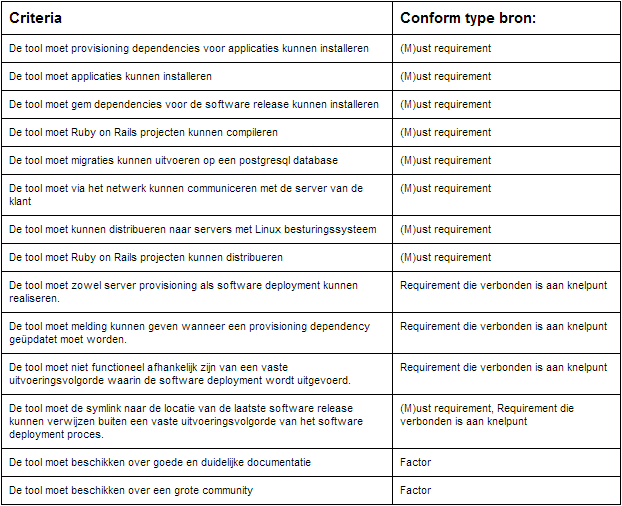
Om deze deelvraag te kunnen beantwoorden, was het noodzakelijk om goed na te denken over de opbouw van de deelvraag. Het proces over de uitvoering van deelvraag 3 wordt nu toegelicht.

### 7.3.1 Literatuuronderzoek & Opstellen longlist

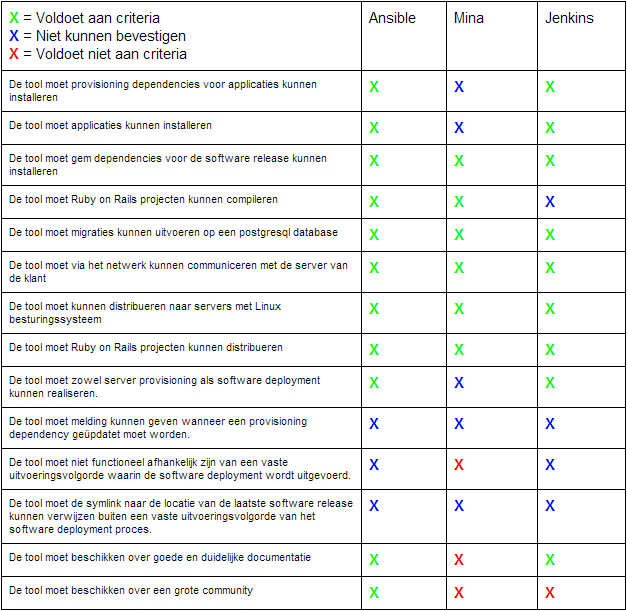
Nadat de requirements in kaart waren gebracht, was het mogelijk om criteria op te stellen voor de longlist. De criteria zijn opgesteld op basis van de volgende drie punten:

* Functional en Non functional requirements met de (M)ust have prioriteit
* Requirements die verbonden zijn aan een knelpunt
* Extra benoemde factoren

De criteria ziet er als volgt uit:



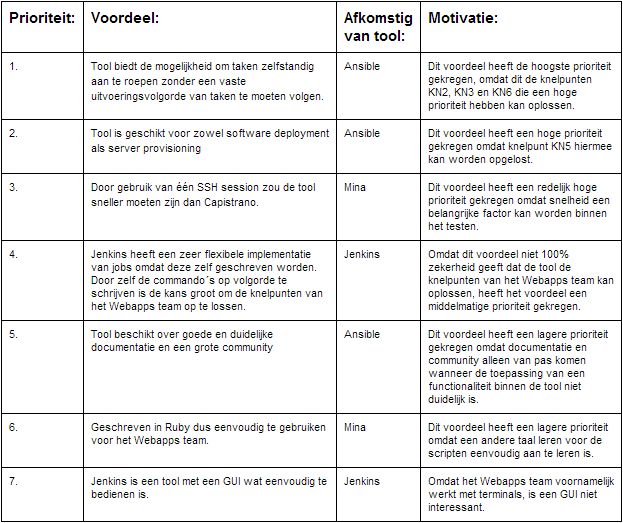
Na het opstellen van de criteria, ben ik begonnen met een literatuuronderzoek. Ik heb tijdens het literatuuronderzoek drie tools op de longlist opgesteld. Voor deze drie tools heb ik binnen het literatuuronderzoek literatuur opgezocht waarin de opgestelde criteria bevestigd kon worden. Het resultaat heb ik verwerkt in een tabel:



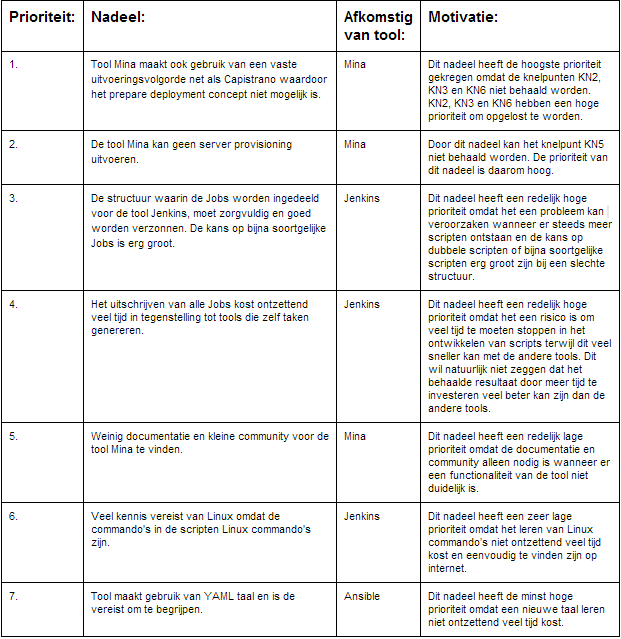
Binnen het overzicht lijkt Ansible te voldoen aan de meeste opgestelde criteria punten.

Voor elke tool heb ik een stuk achtergrond informatie uitgeschreven in het onderzoeksrapport. Naast wat algemene informatie, worden de bronnen genoemd voor de opgestelde criteria welke bevestigen dat de tool wel of niet voldoet aan de criteria. Ook beschrijf ik per tool de voor- en nadelen welke ik gevonden heb. Deze voor- en nadelen heb ik geprioriteerd en gemotiveerd waarom ik de prioriteit heb toegekend.

Het resultaat van de voordelen ziet er als volgt uit:



Het resultaat van de nadelen waarbij prioriteit 1 het “grootste” nadeel is ziet er als volgt uit:



Omdat er in totaal maar drie tools opgesteld waren in de longlist, was het maken van een shortlist overbodig geworden. De definitieve keuze voor de meest geschikte tool voor het Webapps team is direct uit de longlist gekozen.

Om de definitieve keuze te maken welke tool uit de longlist het meest geschikt leek voor het Webapps team, heb ik samen in overleg met mijn begeleider de keuze gebaseerd op de volgende punten:

* Resultaten uit literatuuronderzoek die criteria punten bevestigen
* Resultaten voordelen
* Resultaten nadelen

Uit de resultaten bleek dat Ansible de tool is die waarschijnlijk het beste aansluit op de software distributie binnen het Webapps team, omdat deze aan de meeste criteria punten voldoet, De beste voordelen en de minst erge nadelen heeft.

Voor alle informatie en resultaten van de tools verwijs ik naar het onderzoeksrapport welke als bijlage te vinden is in hoofdstuk 10.3.5.

## 

## 7.4 Deelvraag 4

Na het uitwerken van deelvraag 3 ben ik door gegaan met het beantwoorden van deelvraag 4. Deze deelvraag luidt als volgt:

*“Welke resultaten leveren de beste tool uit de longlist op t.o.v.. Chef en Capistrano in praktijk?”*

Om deze deelvraag te kunnen beantwoorden, was het noodzakelijk om goed na te denken over de opbouw van de deelvraag. Het proces over de uitvoering van deelvraag 4 wordt nu toegelicht.

### 7.4.1 Architectuur Ansible & testproject (Fairwear)

Nadat de resultaten hadden uitgewezen dat software distributie tool Ansible het beste uit de longlist is gekomen, ben ik gestart met het uitschrijven van de architectuur van Ansible. Dit heb ik gedaan door de mappenstructuur te illustreren en toe te lichten. Het resultaat van de mappenstructuur ziet er als volgt uit:



*Hostfile:*

Binnen de hostfile worden de ip adressen beschreven van de servers. Het is mogelijk om de ip adressen van de servers te verdelen in groepen. Deze groepen kun je vervolgens aanroepen om voor de gehele groep een provisioning of deployment te starten. Een voorbeeld is:

[webservers]

33.33.33.10

[appservers]

33.33.33.20

Binnen deze hostfile zijn er 2 groepen die je vervolgens weer apart kunt aanroepen in je playbooks of roles.

*Playbooks:*

Een playbook bestand is geschreven in YAML en bevat een opsomming van uit te voeren tasks. Tasks zijn niets anders dan een kleine taak waarin je een kleine uitvoering uitvoert bijvoorbeeld "apache herstarten" of "een map aanmaken". Binnen de taken kun je gebruik maken van Ansible modules. Door deze modules kun je eenvoudig voorgeprogrammeerde functionaliteiten uitvoeren. Voor de duidelijkheid laat ik een voorbeeld zien van een playbook:

**---**

**- hosts: webservers**

**tasks:**

**- name: Maak test map aan in op de server**

**file: path=/var/test state=directory**

**Bij de hosts wordt aangegeven dat alle servers die onder de groep webservers vallen de taak moeten uitvoeren.**

**File is een voorbeeld van een Ansible module. Er hoeft enkel alleen nog een path en state variabel beschreven te worden. Door de Ansible module "file" wordt nu automatisch een map aangemaakt op het aangegeven path op de server.**

Om de omvang van een playbook te beschrijven kun je denken aan bijvoorbeeld een playbook voor het installeren van Ruby, een playbook voor het installeren van Postgresql of een playbook voor het installeren van mysql.

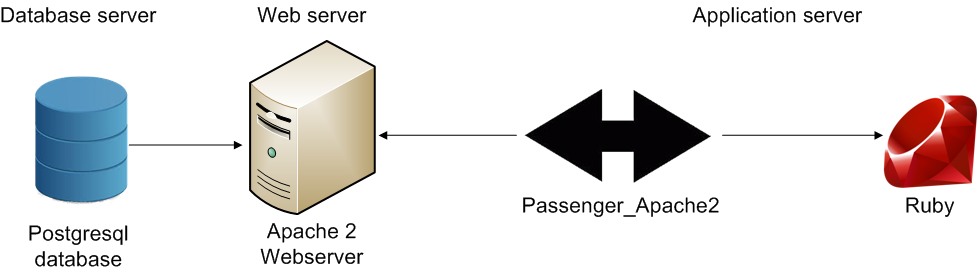
*Roles:*

Binnen een role is het mogelijk om een opsomming van playbooks te beschrijven. Elke rol heeft standaard zes mappen waarin het volgende mogelijk is:

* Files: Bestanden die je kunt uploaden naar de server.
* Meta: Dependencies over welke role als eerst uitgevoerd moet zijn voor je de volgende role kunt uitvoeren.
* Template: standaard configuratie bestand welke ingevuld wordt met variabelen en vervolgens naar de server geschreven wordt wanneer deze veranderd is.
* Handlers: Alle uitgeschreven handlers ( zijn taken die starten conform een conditie )
* Tasks: Alle uitgeschreven taken. ( Wanneer main.yml aanwezig is zal deze taak standaard uitgevoerd worden.
* Vars: Alle variabelen met uitgeschreven waardes.

Om de omvang van een role te beschrijven kun je denken aan bijvoorbeeld een role voor een complete web server voor rails projecten waar Ruby, Apache2 en Passenger op geïnstalleerd worden.

Vervolgens heb ik de architectuur beschreven van het project welke gebruikt zou worden voor het uitvoeren van de testcases. De architectuur van het project ziet er als volgt uit:



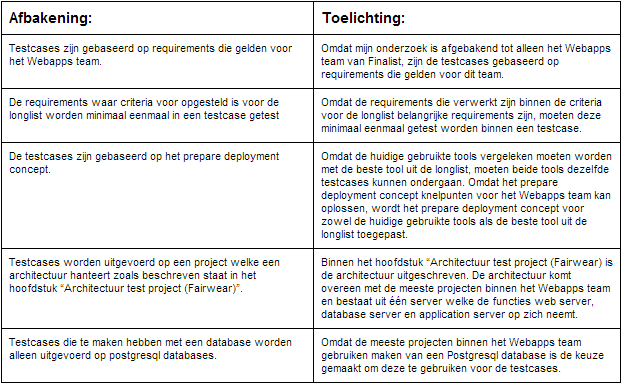
De architectuur laat duidelijk zien dat het om een project gaat waarbij één machine aanwezig is die de functies web server, database server en application server op zich neemt.

Toen de architectuur voor zowel Ansible als het test project duidelijk waren, ben ik gestart met het uitschrijven van de testcases. De testcases zijn uitgeschreven met de focus op het prepare deployment concept. Hierdoor zijn de testcases gebaseerd op het prepare deployment subproces en het publish deployment subproces en zijn de verwachte resultaten van een testcase hier ook op gebaseerd.

### 

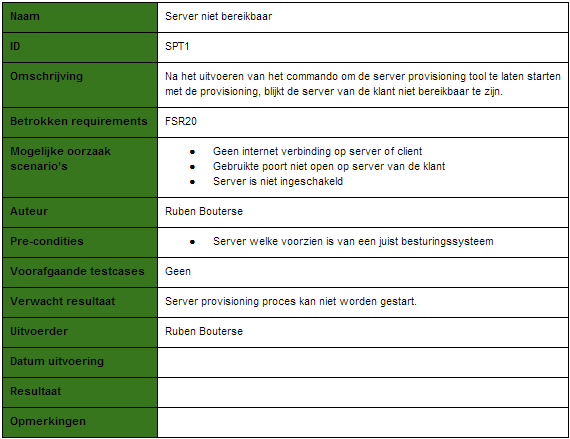
### 7.4.2 Opstellen testcases

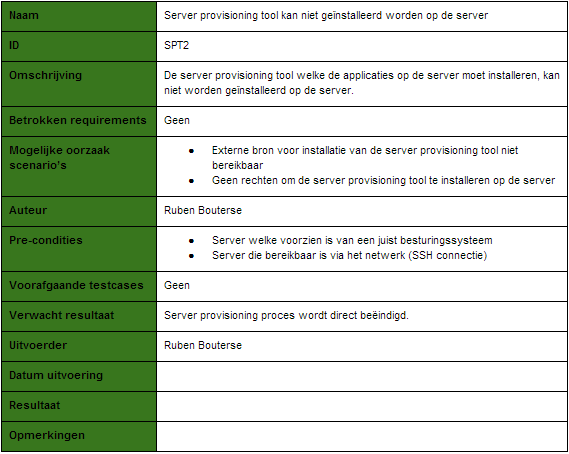
Om duidelijk aan te geven binnen welke scope de testcases uitgevoerd zouden worden, heb ik een tabel opgesteld. Deze ziet er als volgt uit:

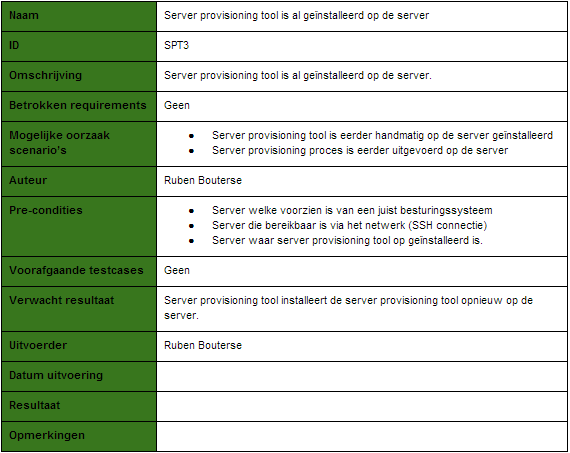


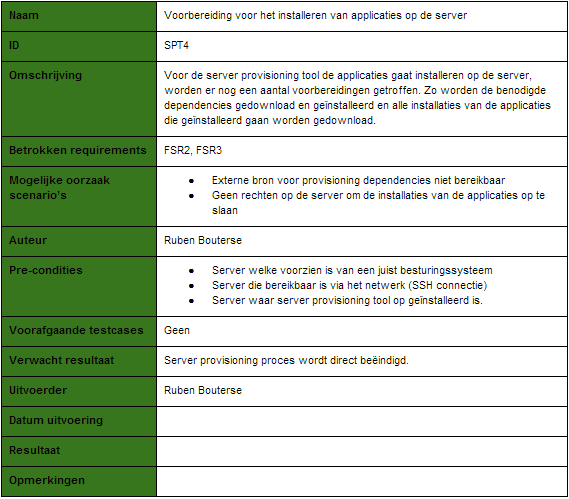
De testcases zijn als volgt:

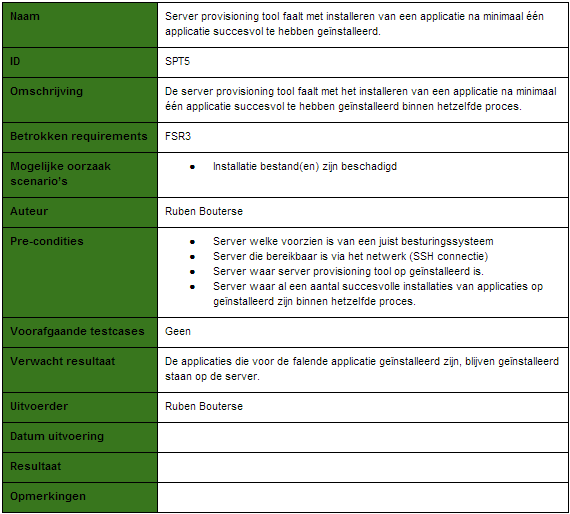
*Testcases server provisioning proces*

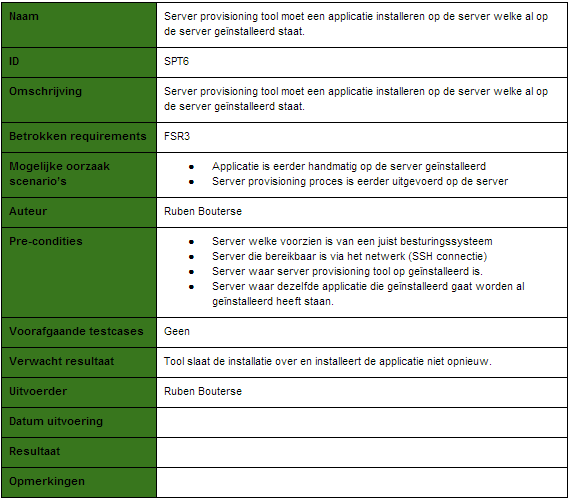


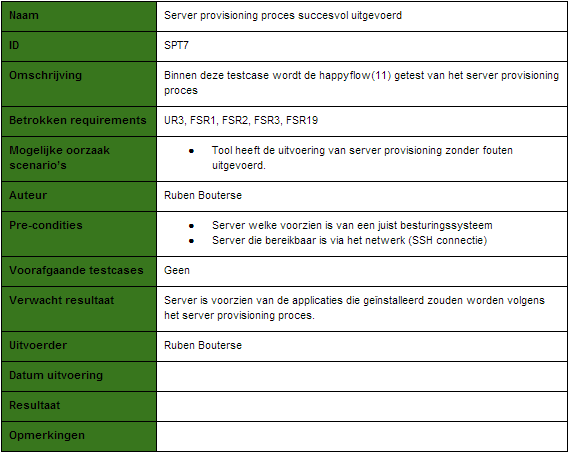




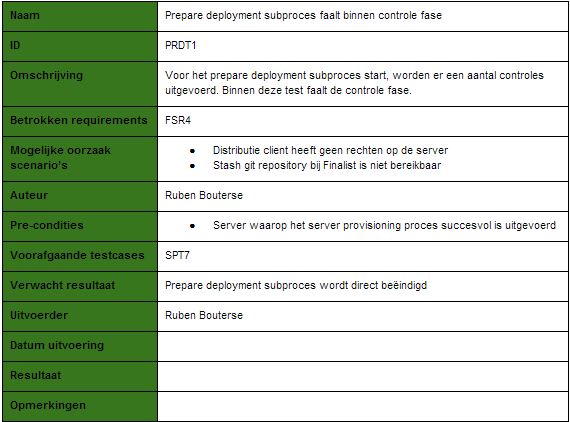


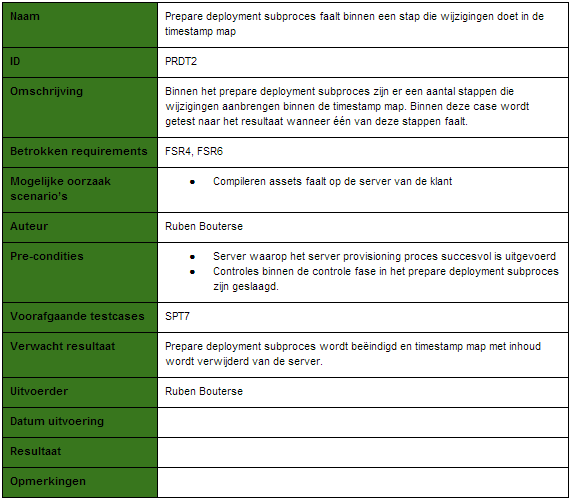


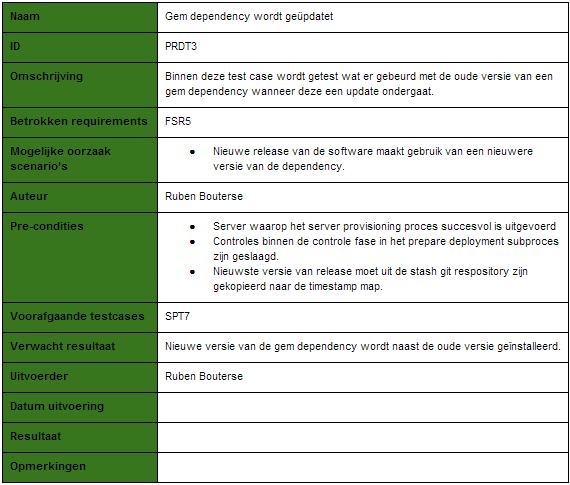




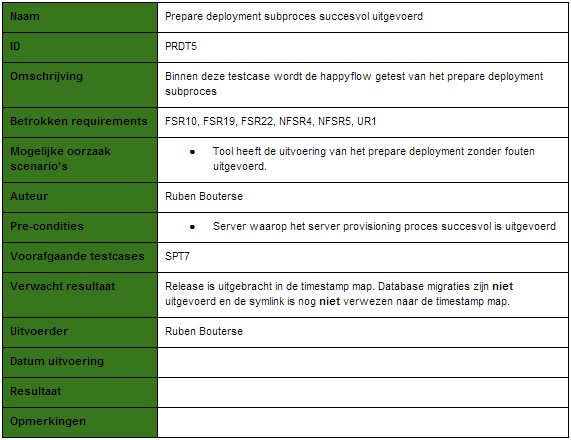
*Testcases prepare deployment subproces*





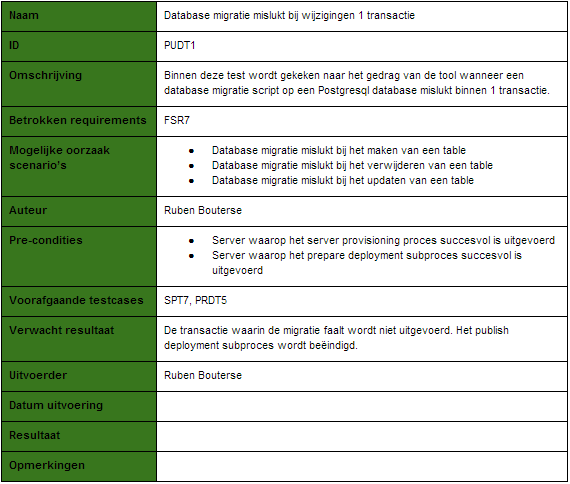


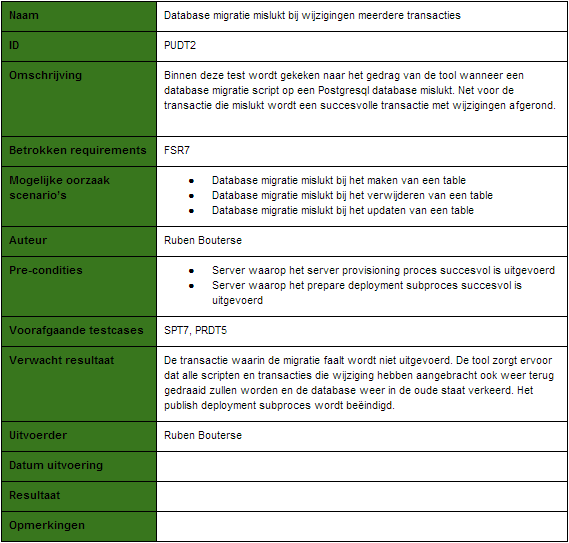


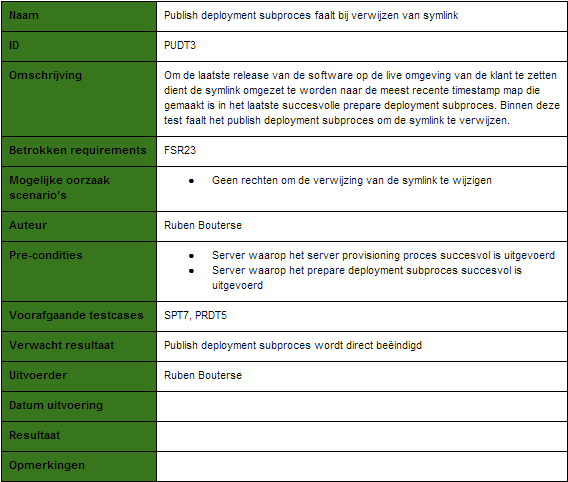


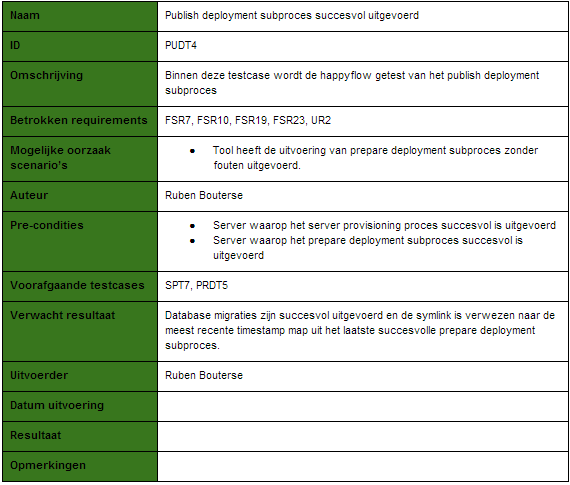
#### 

*Testcases publish deployment subproces*









### 7.4.3 Uitvoering testcases

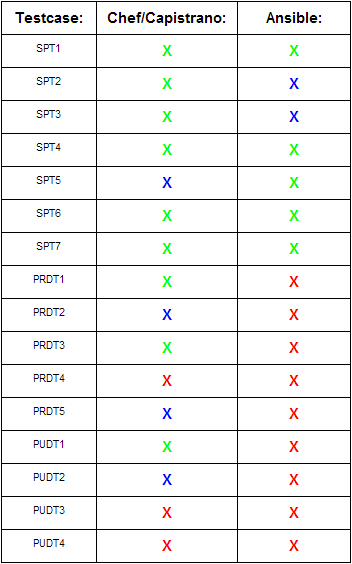
Tot slot heb ik de opgestelde testen uitgevoerd voor zowel de huidige situatie als de toekomstige situatie. Dit wil zeggen dat de tools Chef, Capistrano V3.0 en Ansible getest zijn. Helaas nam het testen van de tools in praktijk veel tijd in beslag en werd ik door mijn begeleider in een grote mate van zelfstandigheid gelaten. Hierdoor is het niet gelukt om alle testcases binnen mijn afstudeerperiode uit te voeren.

Om een beeld te geven welke testcases er uitgevoerd zijn en of deze afwijkend zijn van het verwacht resultaat laat ik een overzicht zien:

**X** = Testcase uitgevoerd met resultaat gelijk aan het verwacht resultaat

**X** = Testcase uitgevoerd met resultaat afwijkend dan het verwacht resultaat

**X** = Testcase niet uitgevoerd



Voor de volledig uitgewerkte testcases verwijs ik naar het testrapport welke als bijlage is te vinden in hoofdstuk 10.3.7.

## 7.5 Adviesrapport

Omdat het niet gelukt is om binnen mijn afstudeertijd alle gewenste resultaten te behalen,was het lastig om een eindconclusie te formuleren. Het geven van een advies was hierdoor moeilijk.

Ik heb de verzamelde conclusies herhaald en vervolgens een advies voor het Webapps team geformuleerd.

Omdat de missende resultaten enorm van belang zijn om een goede beslissing te kunnen nemen of Ansible of de huidige tools het meest geschikt zijn voor het Webapps team, heb ik als advies gegeven om de missende resultaten te verzamelen. Zolang deze resultaten niet bekend zijn, blijven Chef en Capistrano de meest geschikte tools voor de software distributie van het Webapps team.

# 

# 8. Conclusies

## 8.1 Conclusie deelvraag 1

De software distributie van het Webapps team wordt uitgevoerd in twee processen. Om deze twee processen uit te kunnen voeren, dient de distributie client voorzien te worden van een aantal software programma’s.

De twee processen die uitgevoerd worden zijn als volgt:

*Server provisioning:*  
Het server provisioning proces wordt uitgevoerd in drie verschillende activiteiten namelijk:

* *Verzoek indienen*

Binnen deze activiteit wordt er een verzoek ingediend bij de systeembeheerder voor het aanmaken van een nieuwe server.

* *Installatie server*  
  Binnen deze activiteit wordt de server voorzien van een besturingssysteem.
* *Installatie benodigde applicaties*  
  Binnen deze activiteit wordt de server voorzien van de provisioning dependencies voor de applicatie die in het software deployment proces zal worden gedistribueerd. Het installeren van de provisioning dependencies wordt met behulp van de tool “Chef” uitgevoerd.

Server provisioning wordt eenmalig uitgevoerd. De reden waarom dit proces eenmalig uitgevoerd wordt komt omdat de server eenmalig ingericht moet worden met de juiste provisioning dependencies.

*Software deployment:*Het software deployment proces wordt uitgevoerd in twee verschillende activiteiten namelijk:

* *Installatie release*

Binnen deze activiteit wordt de nieuwste release van de applicatie gedistribueerd naar de server van de klant. Het distribueren van de nieuwste release wordt met behulp van de tool “Capistrano V3.0” uitgevoerd. Capistrano V3.0 maakt gebruik van 4 “stages” waarin stappen worden uitgevoerd om de software te distribueren. Deze stages worden in een vaste uitvoeringsvolgorde uitgevoerd en zijn niet aanpasbaar.

* *Testen door klant*  
  Binnen deze activiteit worden testen uitgevoerd door de klant.

Software deployment wordt herhalend uitgevoerd. De reden waarom dit proces herhalend uitgevoerd wordt komt omdat er steeds nieuwe releases gedistribueerd worden op de server.

## 8.2 Conclusie deelvraag 2

De knelpunten die binnen de software distributie van het Webapps team optreden zijn als volgt:

|  |  |  |
| --- | --- | --- |
| **ID:** | **Knelpunt:** | **Prioriteit:** |
| KN1 | Het downloaden van gem dependencies wordt gedaan van een externe bron. | Middel |
| KN2 | Het downloaden van gem dependencies valt binnen een vaste uitvoeringsvolgorde van de software deployment tool. | Hoog |
| KN3 | Het compileren van assets valt binnen een vaste uitvoeringsvolgorde van de software deployment tool. | Hoog |
| KN4 | Wanneer een provisioning dependency geüpdatet moet worden, verschijnt er geen foutmelding. | Laag |
| KN5 | Het server provisioning en software deployment proces gebruiken verschillende tools. | Middel |
| KN6 | Het verwijzen van de symlink naar de meest recente software release, valt binnen een vaste uitvoeringsvolgorde van de software deployment tool | Hoog |

Om deze knelpunten op te lossen zijn er drie concepten opgesteld voor een ideaal toekomstbeeld van het Webapps team. De drie concepten zijn als volgt:

*Prepare deployment*Binnen het prepare deployment concept wordt gesproken over het opdelen van het software deployment proces in twee subprocessen. Door de opdeling van twee subprocessen is het mogelijk om een release vroegtijdig te distribueren naar de server van de klant en zo mogelijke complicaties op tijd op te kunnen lossen. Echter is de release nog niet doorgevoerd op de live omgeving van de klant.

Binnen het tweede subproces wordt enkel alleen nog de database migraties en de symlink verwezen naar de laatst gedistribueerde release. Hierdoor wordt de nieuwste release in de live omgeving doorgevoerd.

Met het prepare deployment concept worden de volgende knelpunten opgelost:

|  |  |  |
| --- | --- | --- |
| **ID:** | **Knelpunt:** | **Prioriteit:** |
| KN1 | Het downloaden van gem dependencies wordt gedaan van een externe bron. | Middel |
| KN2 | Het downloaden van gem dependencies valt binnen een vaste uitvoeringsvolgorde van de software deployment tool. | Hoog |
| KN3 | Het compileren van assets valt binnen een vaste uitvoeringsvolgorde van de software deployment tool. | Hoog |
| KN6 | Het verwijzen van de symlink naar de meest recente software release, valt binnen een vaste uitvoeringsvolgorde van de software deployment tool | Hoog |

*Geen foutmelding bij het noodzakelijk updaten van provisioning dependencies*Binnen dit concept zijn twee opties opgesteld namelijk:

* *Optie1: server provisioning proces starten voor iedere uitvoering van het software deployment proces*

Wanneer deze optie van toepassing is, zal voor het uitbrengen van een release elke keer het server provisioning proces worden gestart. Provisioning dependencies die geüpdatet moeten worden, zullen worden geïnstalleerd. Het opleveren van een release zal vors toenemen in tijdsduur omdat bij iedere release het server provisioning proces gestart wordt. Echter zal het server provisioning proces niet zo lang duren als de eerste keer dat server provisioning is uitgevoerd.

* *Optie2: tonen van een melding wanneer een provisioning dependency bijgewerkt moet worden*Wanneer deze optie van toepassing is zal dit niet toenemen in de tijdsduur van de oplevering van een release. Deze optie zorgt ervoor dat een werknemer direct weet dat er een update vereist is en om welke provisioning dependency het gaat. Het updaten van de provisioning dependency zal niet veel tijd in beslag nemen.

Met dit concept wordt het volgende knelpunt opgelost:

|  |  |  |
| --- | --- | --- |
| **ID:** | **Knelpunt:** | **Prioriteit:** |
| KN4 | Wanneer een provisioning dependency geüpdatet moet worden, verschijnt er geen foutmelding. | Laag |

*Server provisioning en software deployment processen realiseerbaar binnen één software distributie tool*Binnen dit concept zijn twee opties opgesteld namelijk:

* *Optie1: tool zoeken die zowel server provisioning als software deployment kan realiseren*

Binnen deze optie zal er één tool worden gebruikt voor zowel het server provisioning als het software deployment proces. Dit brengt als voordeel met zich mee dat een medewerker nog maar één tool hoeft te begrijpen om twee processen uit te kunnen voeren.

* *Optie 2: tools server provisioning en software deployment gescheiden houden*

Binnen deze optie zullen beide processen gebruik maken van een eigen tool. In de huidige situatie is dit ook gerealiseerd met Chef als tool voor server provisioning en Capistrano als tool voor software deployment.

Met dit concept wordt het volgende knelpunt opgelost:

|  |  |  |
| --- | --- | --- |
| **ID:** | **Knelpunt:** | **Prioriteit:** |
| KN5 | Het server provisioning en software deployment proces gebruiken verschillende tools. | Middel |

## 8.3 Conclusie deelvraag 3

Op basis van goede criteria, zijn er drie tools opgesteld binnen de longlist namelijk:

*Ansible:*

Ansible is momenteel een hype onder de automatisering tools. Ook binnen Finalist is de naam Ansible al meerdere keren gevallen bij verschillende teams en zelfs recent toegepast binnen het Netwerk VSP team. Ansible wordt momenteel gebruikt door vele grote bekende bedrijven en heeft al meer dan 1miljoen downloads.

*Mina:*

Mina is een tool die ik gevonden heb door te zoeken op Google. Mina lijkt de opvolger van Capistrano en geeft aan veel sneller software deployment te kunnen realiseren door het gebruik van één SSH connectie. Omdat Mina in de toepassing lijkt op Capistrano zou het leren van deze tool vrij eenvoudig kunnen zijn.

*Jenkins:*

Jenkins is een tool die ik het meeste ben tegen gekomen binnen de interviews met de zeven teams. Jenkins biedt een flexibele werkomgeving omdat de scripten helemaal zelf uitgeschreven moeten worden. Dit biedt de mogelijkheid om een taak (Job) helemaal te laten functioneren hoe je zelf wilt.

Om de definitieve keuze te maken welke tool uit de longlist het meest geschikt leek voor het Webapps team, heb ik samen in overleg met mijn begeleider de keuze gebaseerd op de volgende punten:

* Resultaten uit literatuuronderzoek die criteria punten bevestigen
* Resultaten voordelen
* Resultaten nadelen

Uit de resultaten bleek dat Ansible de tool is die waarschijnlijk het beste aansluit op de software distributie binnen het Webapps team, omdat deze aan de meeste criteria punten voldoet, De beste voordelen en de minst erge nadelen heeft. Ansible is dus de meest interessante tool om te testen in praktijk.

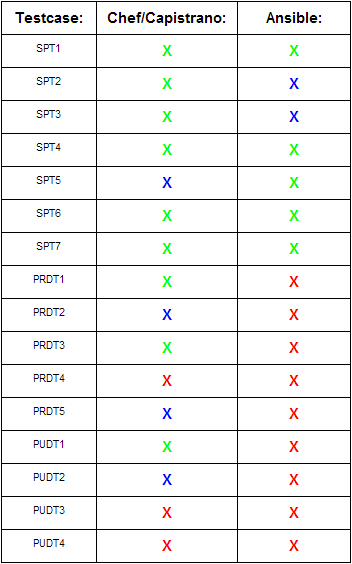
## 8.4 Conclusie deelvraag 4

Helaas nam het testen van de tools in praktijk veel tijd in beslag en is het niet gelukt om alle testcases binnen mijn afstudeerperiode uit te voeren. Het is daarom niet mogelijk om een volledige conclusie voor deze deelvraag op te stellen. Een overzicht met de resultaten is als volgt:

**X** = Testcase uitgevoerd met resultaat gelijk aan het verwacht resultaat

**X** = Testcase uitgevoerd met resultaat afwijkend dan het verwacht resultaat

**X** = Testcase niet uitgevoerd



We kunnen concluderen dat de testcases die beginnen met SPT (server provisioning testen) uitgevoerd zijn met zowel de tool Chef als de tool Ansible. De afwijkende resultaten van beide tools zijn als volgt:

*Chef:*

Wanneer we het afwijkende resultaat bekijken van testcase SPT5, valt er te concluderen dat het gaat om een positieve afwijking. Testcase SPT5 kan niet worden uitgevoerd omdat Chef de installatie bestanden al in een eerder stadium van het proces controleert en een foutmelding geeft bij een beschadigde installatie. Hierdoor is het dus onmogelijk dat er minimaal één applicatie succesvol geïnstalleerd is en vervolgens een beschadigde installatie faalt.

*Ansible:*

Wanneer we de afwijkende resultaten bekijken van de testcases SPT2 en SPT3, valt er te concluderen dat het gaat om positieve afwijkingen. Ansible wordt t.o.v. Chef **niet** geïnstalleerd op de server. Hierdoor zijn de testcases SPT2 en SPT3 niet relevant voor de Tool Ansible.

We kunnen concluderen dat het uitvoeren van server provisioning met zowel Chef als Ansible mogelijk is. Het afwijkende gedrag op de uitgevoerde testcases geven geen negatief gevoel over de gebruikte tool. Omdat beide tools in staat zijn om server provisioning uit te voeren zonder extreme positieve of negatieve afwijkingen, concludeer ik dat Chef niet vervangen hoeft te worden met Ansible.

Helaas zijn de testcases voor software deployment niet uitgevoerd met de tool Ansible. Hierdoor is het niet mogelijk een conclusie te kunnen trekken tussen beide software deployment tools.

## 8.5 Eindconclusie

Nu de resultaten en conclusies van de deelvragen grotendeels beantwoordt zijn, is het tijd om de hoofdvraag te beantwoorden. Een herhaling van de hoofdvraag is:

*“Welke software distributie tool sluit het beste aan op de software distributie binnen het Webapps team?”*

In deelvraag 1 hebben we kunnen concluderen dat de software distributie binnen het Webapps team bestaat uit twee processen die bestaan uit verschillende activiteiten namelijk:

* Server provisioning proces
  + *Verzoek indienen*
  + *Installatie server*
  + *Installatie benodigde installaties*
* Software deployment proces
  + *Installatie release*
  + *Testen door klant*

Omdat de processen in deelvraag 1 in detail uitgewerkt zijn, kunnen we een klein deel van de hoofdvraag verduidelijken namelijk:

*“Welke software distributie tool sluit het beste aan op de* ***software distributie*** *binnen* ***het Webapps team****?”*

Wat we verstaan onder de term “software distributie” binnen het onderzoek.  
Wat we verstaan onder de term “het Webapps team” binnen het onderzoek.

In deelvraag 2 hebben we kunnen concluderen dat er zich een aantal knelpunten voordoen in de huidige manier van software distributie binnen het Webapps team. Voor deze knelpunten zijn er opties beschreven voor een ideaal toekomstbeeld voor het Webapps team. Op basis van de verzamelde gegevens heb ik requirements kunnen opstellen. Door opstellen van de requirements, kunnen we een klein deel van de hoofdvraag verduidelijken namelijk:

*“Welke* ***software distributie tool*** *sluit het beste aan op de software distributie binnen het Webapps team?”*

Wat we verstaan onder de term “software distributie tool” binnen het onderzoek.

In deelvraag 3 hebben we kunnen concluderen dat Ansible de tool is die waarschijnlijk het beste aansluit op de software distributie binnen het Webapps team, omdat deze aan de meeste criteria punten voldoet, De beste voordelen en de minst erge nadelen heeft. Door deze conclusie kunnen we een klein deel van de hoofdvraag beantwoorden namelijk:

*“****Welke******software distributie tool*** *sluit het beste aan op de software distributie binnen het Webapps team?”*

Welke software distributie tool sluit **mogelijk** het beste aan op de software distributie binnen het Webapps team.

Helaas kunnen we geen volledige conclusie uit deelvraag 4 ophalen omdat niet alle resultaten behaald zijn. Hierdoor is het helaas niet mogelijk om een definitief antwoord op de hoofdvraag te geven.

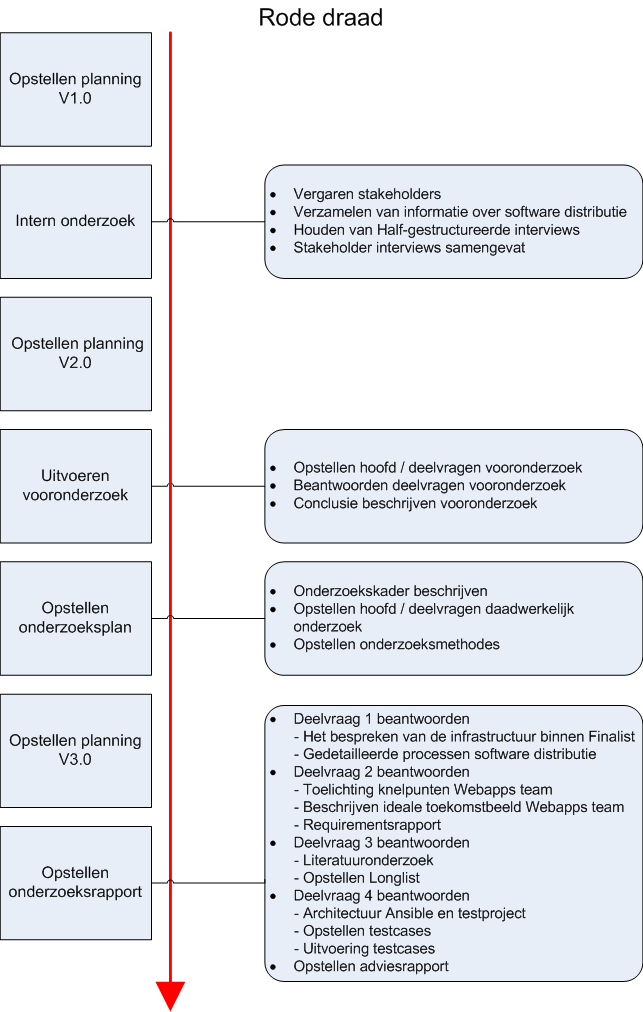
# 9. Evaluatie

Binnen dit hoofdstuk bespreek ik de evaluatie voor de uitgevoerde afstudeeropdracht. De evaluatie is opgedeeld in drie hoofdstukken. Het eerste hoofdstuk beschrijft hoe het uitvoeringsproces van de afstudeeropdracht is verlopen. Het tweede hoofdstuk beschrijft de evaluatie over de (tussen)producten welke gemaakt zijn. Het derde hoofdstuk beschrijft welke competenties binnen de opleiding van toepassing zijn geweest bij de uitvoering van de afstudeeropdracht.

## 

## 9.1 Hoe is het uitvoeringsproces verlopen

Om mijn afstudeeropdracht tot een goed eindresultaat te brengen, heb ik zorgvuldig gewerkt aan een goede en duidelijke rode draad binnen mijn onderzoek. De rode draad is als volgt:



Binnen dit hoofdstuk wil ik graag toelichten hoe het uitvoeringsproces van mijn onderzoek is verlopen.

*Opstellen planning V1.0*

Het proces is gestart met het opstellen van een planning. Het opstellen van de planning was eenvoudig te realiseren. Ik heb het aantal toegewezen dagen voor elke activiteit uit het goedgekeurd afstudeerplan gehaald en hiervan een Gantt chart(10) gemaakt.

*Intern onderzoek*

De uitvoering van mijn intern onderzoek is gestart met het vergaren van stakeholders. Om stakeholders te vergaren, heb ik een mail gestuurd naar alle medewerkers binnen Finalist. Echter kwam hier maar 1 reactie op binnen en was deze manier van stakeholders verzamelen niet zinvol. Achteraf was het versturen van een mail voor het vergaren van stakeholders een slecht idee. Wanneer ik namelijk wel veel reacties had gehad, kon het zijn dat dit niet de juiste stakeholders voor dit onderwerp waren. Volgende keer zou ik direct navragen welke stakeholders het meest geschikt zijn voor mijn onderzoek en deze aanspreken.

Over het houden van de half-gestructureerde interviews binnen het intern onderzoek ben ik zeer tevreden. De resultaten die ik verzameld heb waren goed te gebruiken binnen de rest van mijn onderzoek.

Het samenvatten van de stakeholder interviews heb ik efficiënt aangepakt door de gesprekken op te nemen met een geluidsopname. Naderhand kon ik door middel van deze geluidsopname eenvoudig een samenvatting maken.

*Opstellen planning V2.0*

Wegens ziekte was het noodzakelijk een nieuwe planning op te stellen. Ik heb de planning opgesteld over een periode van 30 weken met de intentie dat één dag bestond uit 4 uur. Ik heb de planning uitgewerkt in een gantt chart.

*Uitvoeren vooronderzoek*

Het starten aan het vooronderzoek is moeizaam verlopen. Omdat het onderzoeksonderwerp niet goed was beschreven door Finalist en hierdoor de opdracht niet duidelijk genoeg was, heb ik meerdere keren wijzigingen aangebracht aan de hoofd en deelvragen van het vooronderzoek.

Naarmate ik een aantal keer samen met mijn begeleider in gesprek ben geweest en we hadden besloten om in het vooronderzoek te focussen op de basis processen binnen software distributie, ben ik gestart met het beantwoorden van de laatste hoofd en deelvragen. Het beantwoorden van de hoofd en deelvragen is goed verlopen met de informatie uit het intern onderzoek.

Tot slot heb ik een conclusie kunnen beschrijven waarin de hoofdvraag van het vooronderzoek beantwoord werd.

*Opstellen onderzoeksplan*

Binnen het onderzoeksplan ben ik begonnen met het uitschrijven van het onderzoekskader. Door het onderzoekskader op te stellen, heb ik een duidelijk beeld gekregen van het doel en het probleem van de afstudeeropdracht. De afstudeeropdracht was in overleg met mijn begeleider van onderzoeksonderwerp veranderd omdat het onderzoek anders te klein zijn worden voor mijn afstudeertraject.

Door het duidelijke onderzoekskader, was het eenvoudig om hoofd / deelvragen op te stellen. Ik heb binnen het onderzoeksplan beschreven in welke volgorde ik de deelvragen wil beantwoorden en op welke manier ik informatie wil verzamelen voor mijn onderzoek. Het uitschrijven van het onderzoeksplan is goed verlopen.

*Opstellen planning V3.0*

Omdat er een aantal activiteiten zijn komen te vervallen door het veranderen van het onderzoeksonderwerp, is de planning aangepast. Ik heb de planning uitgewerkt in een gantt chart.

*Opstellen onderzoeksrapport*

Binnen het onderzoeksrapport heb ik 4 deelvragen beantwoord. Het beantwoorden van de deelvragen leek echter meer tijd in beslag te nemen dan verwacht. Hierdoor is na het beantwoorden van deelvraag 2 een achterstand van ongeveer twee maanden geconstateerd.

Gelukkig bleken echter de rest van de activiteiten mee te vallen die binnen deelvraag 3 en 4 verwerkt zijn en heb ik alle deelvragen binnen mijn afstudeertraject kunnen beantwoorden en een adviesrapport voor Finalist kunnen opstellen.

Tot slot wil ik zeggen dat ik binnen mijn afstudeertraject in een grote vorm van zelfstandigheid gelaten door mijn begeleider. Hierdoor heeft het begrijpen van informatie en technologieën extra inspanning gekost. Ik ben tevreden over de aanpak, procesverloop en resultaten van mijn onderzoek.

## 9.2 Hoe zien de (tussen) producten eruit

Binnen het afstudeerplan is een lijst opgesteld met (tussen)producten welke ontwikkeld zouden worden binnen het afstudeertraject. Op één van deze producten na, zijn alle (tussen)producten opgesteld en verwerkt binnen het afstudeerdossier. Het product “Testplan” is komen te vervallen door een wijziging binnen de opdracht. De (tussen)producten welke ontwikkeld zijn binnen mijn afstudeertraject zijn als volgt:

* Vooronderzoek
* Onderzoeksplan
* Onderzoeksrapport
* Requirementsrapport
* Testrapport
* Adviesrapport
* Afstudeerdossier

Alle (tussen)producten zijn volledig afgerond en bevatten de juiste richtlijnen zoals deze ook op school aangeleerd zijn. Zo is er gebruik gemaakt van enkele methodieken (Onderzoeksmethodieken, MoSCoW) maar ook op de juiste manier aan de opbouw van een product gewerkt.

## 

## 9.3 Behalen van de beroepstaken ( competenties )

Binnen het afstudeerplan zijn vier beroepstaken opgesteld welke binnen het afstudeertraject op een bepaald niveau door de student uitgevoerd moesten worden. Door een wijziging in de afstudeeropdracht, is één van deze opgestelde beroepstaken gewijzigd. De vier beroepstaken welke ik uitgevoerd heb binnen mijn afstudeertraject zijn als volgt:

* 1.3 selecteren van standaardsoftware
* 1.4 Uitvoeren analyse door definitie van requirements
* 3.5 Uitvoeren van en rapporteren over het testproces
* 4.4 Beheren en distribueren van software

Graag motiveer ik per beroepstaak mijn uitvoering binnen het afstudeertraject waarop ik baseer hoe ik de beroepstaak heb behaald:

*1.3 selecteren van standaardsoftware*

Binnen deelvraag 3 ben ik begonnen met het opstellen van criteria voor een longlist van software distributie tools welke mogelijk ter vervanging voor de huidige gebruikte tools binnen het Webapps team kunnen functioneren. Na het opstellen van deze criteria, heb ik 3 tools gevonden voor de longlist.

Om de beste tool te selecteren uit de longlist, heb ik een literatuuronderzoek gedaan waarin ik bronnen het gezocht over de haalbaarheid van de criteria punten per tool. Daarnaast heb ik per tool de voor- en nadelen op kunnen stellen. Ik heb uiteindelijk één tool geselecteerd op basis van de volgende punten:

* Tool met de meeste en belangrijkste haalbare criteria punten
* Tool met de minst erge nadelen
* Tool met de beste voordelen

Met deze onderbouwing bevestig ik te voldoen aan de uitvoering van beroepstaak 1.3.

*1.4 Uitvoeren analyse door definitie van requirements*

Voor het opstellen van de requirements ben ik begonnen met het verzamelen van informatie door half-gestructureerde interviews te houden met zeven stakeholders. Deze zeven interviews hebben mij informatie gegeven om deelvraag 1 van het onderzoek te beantwoorden. Binnen het interview met het Webapps team heb ik een aantal knelpunten vastgesteld en verwerkt tot requirements.

Binnen deelvraag 1 heb ik de processen van het server provisioning en software deployment in detail beschreven. Hierdoor heb ik kunnen bevestigen welke functionaliteiten de tool minimaal moet kunnen uitvoeren. Deze functionaliteiten heb ik verwerkt tot requirements.

Binnen deelvraag 2 heb ik het ideale toekomstbeeld van het Webapps team beschreven. Binnen het ideale toekomstbeeld zijn opties beschreven welke mogelijk de oplossing bieden voor de knelpunten van het Webapps team. Deze opties heb ik verwerkt tot requirements.

De requirements zijn verwerkt in een requirementsrapport welke te vinden is als bijlage in hoofdstuk 10.3.6. Het requirementsrapport beschrijft de bronnen waaruit de requirements zijn opgesteld, de scope waarvoor de requirements opgesteld zijn, de opgestelde requirements, use case diagram en use case beschrijvingen.

Met deze onderbouwing bevestig ik te voldoen aan de uitvoering van beroepstaak 1.4.

*3.5 Uitvoeren van en rapporteren over het testproces*

Binnen deelvraag 4 heb ik testcases opgesteld welke ik uitgevoerd heb met verschillende software distributie tools. De uitgevoerde testcases zijn verwerkt in een testrapport welke te vinden is als bijlage in hoofdstuk 10.3.7. Binnen het testrapport is de scope waarvoor de testen opgesteld zijn en de resultaten per tool te lezen.

Met deze onderbouwing bevestig ik te voldoen aan de uitvoering van beroepstaak 3.5.

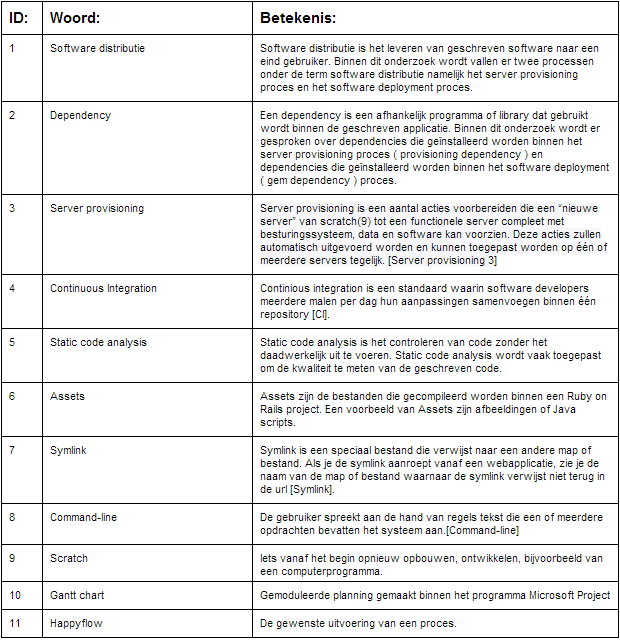
*4.4 Beheren en distribueren van software*

Het software deployment proces heeft als doel om software te distribueren naar de klant. Binnen testcase PRDT5 welke ik uitgevoerd heb met de tools Capistrano en Ansible, heb ik de happyflow van het software deployment getest. Met deze onderbouwing bevestig ik te voldoen aan de uitvoering van beroepstaak 4.4.

# 

# 10. Referenties

## 10.1 Glossary



## 

## 10.2 Literatuurlijst

Binnen dit hoofdstuk kunt u de bronvermeldingen vinden welke gebruikt zijn binnen dit document. Elke bron is voorzien van een [Naam] welke in de tekst op dezelfde wijze is terug te vinden. Achter iedere bronvermeldingen worden zoveel mogelijk gegevens over de bron beschreven. De volgorde van deze gegevens zijn als volgt:

Uniek nummer, [Naam bron], Jaartal waaruit bron afstamt, auteur, titel boek/artikel, Geraadpleegd op datum, Link bron.

1. [Nel Verhoeven] 2007, Nel Verhoeven, Wat is onderzoek?

Geraadpleegd op 20-05-2014

Bron: Boek

1. [Server Provisioning 3] Augustus 2010, Nix Craft, Top 5 Open Source Linux Server Provisioning Software  
   Geraadpleegd op 15-05-2014  
   Bron: <http://www.cyberciti.biz/tips/server-provisioning-software.html>
2. [CI] 2 December 2014, Continious integrations

Geraadpleegd op 25-05-2014

Bron: http://en.wikipedia.org/wiki/Continuous\_integration

1. [Symlink] 2013, Symlink

Geraadpleegd op 20-08-2014

<https://wiki.debian.org/SymLink>

1. [Command-line] 2013, Command-line-interface  
   Geraadpleegd op 18-08-2014   
   http://nl.wikipedia.org/wiki/Command-line-interface

## 10.3 bijlagen

### 10.3.1 Vooronderzoek

**Vooronderzoek**

Onderzoek software distributie

Wat is software distributie binnen Finalist?



Auteur: Ruben Bouterse Eerste docent: H.P.Weenink

Opleiding: Informatica Tweede docent: G.M.Tuk

Bedrijf: Finalist Begeleider: R.Schellhorn

Opdrachtgever: M. van Breukelen

Datum: 15-05-2014

Versie: 1.0

Inhoudsopgave



# 

Inleiding

Dit document is geschreven ter voorbereiding op mijn onderzoek naar de software distributie binnen Finalist. Het onderzoek zal verricht worden als afstudeeropdracht voor de Haagse Hogeschool.

De aanleiding van dit onderzoek is dat de eisen die klanten stellen voor software distributie vergeleken met vroeger steeds hoger worden. Zo zijn klanten meer gesteld op zo weinig mogelijk downtime van hun systemen en een kortere tijdsduur van software distributie doordat automatisering steeds meer een rol is gaan spelen. Finalist heeft aangegeven dat ze graag een onderzoek willen starten om te kijken op welke punten binnen het software distributie proces verbetering kan worden gebracht.

Het vooronderzoek dient als middel voor het verschaffen van informatie en het verduidelijken van het onderzoeksonderwerp. Afhankelijk van de gevonden informatie en de bruikbaarheid hiervan, kan worden besloten of het onderzoek naar de software distributie binnen Finalist uitgevoerd kan worden.

# 

1. Hoofdvraag

Binnen het vooronderzoek wordt de volgende hoofdvraag behandeld:

Wat is software distributie binnen Finalist?

2. Deelvragen

Om antwoord te kunnen geven op de hoofdvraag, is het noodzakelijk om eerst een aantal deelvragen te beantwoorden. De resultaten van de deelvragen vormen samen het antwoord op de hoofdvraag. De deelvragen voor de hoofdvraag zijn:

* Hoe zien de basis processen van software distributie eruit?
* Welke stappen binnen de activiteiten van het software distributie proces kunnen er geautomatiseerd uitgevoerd worden?

# 

3. Beantwoording deelvragen

3.1 Hoe zien de basis processen van software distributie eruit?

Zodra je als software ontwikkelingsbedrijf een release van je applicatie hebt geschreven en deze wilt opleveren naar de klant, kom je op het moment aan om software distributie toe te passen. Het software distributie proces zal bij iedere release van de geschreven software terug komen.

Software distributie komt niet alleen voor bij het leveren van zelf geschreven software naar de klant. Wanneer een nieuwe server moet worden ingericht zal deze moeten worden voorzien van een besturingssysteem en applicaties. Deze manier van software distributie valt onder de term server provisioning. In hoofdstuk 3.1.2 kunt u lezen wat er geschreven staat over server provisioning.

Voor zowel het distribueren van software als server provisioning zijn verschillende manieren mogelijk om het proces in te richten. Om toch antwoord op de deelvraag te kunnen geven, heb ik 7 interviews gehouden bij verschillende teams binnen Finalist [Stakeholder interviews]. De overeenkomsten en basiselementen die altijd terug kwamen binnen het software distributie proces wil ik graag bespreken in dit hoofdstuk om een beeld te krijgen welke dingen er minimaal plaats vinden.

3.1.1 Software distributie

Op internet is snel te lezen dat de betekenis van software distributie niet meer is dan het proces van het leveren van software naar een eind gebruiker. Dit klinkt allemaal nog vrij eenvoudig maar in werkelijkheid is software distributie vaak het meest stressvolle proces binnen software ontwikkeling bedrijven.

In het boek “Continuous Delivery” van Jez Humble en David Farley wordt ook een beschrijving gegeven van software distributie, namelijk:

*“In many software projects, release is a manually intensive process. The environments that host the software are often crafted individually, usually by an operations or IS team. Third-party software that the application relies on is installed. The software artifacts of the application itself are copied to the production host environments. Configuration information is copied or created through the admin consoles of web servers, application servers, or other third-party components of the system. Reference data is copied, and finally the application is started, piece by piece if it is a distributed or service-oriented application.”*

De bovenstaande omschrijving geeft aan dat software distributie vaak een “handmatig intensief” proces is en de activiteiten binnen het proces stap voor stap worden uitgevoerd om software te distribueren. Elke activiteit binnen het software distributie proces wordt vaak door een individu of team uitgevoerd.

Bij het uitbrengen van een release wordt vaak gedacht aan het updaten van alleen de code. Echter blijkt dit niet het enige onderdeel te zijn wat vervangen kan worden binnen het software distributie proces. Zo word het aanbrengen van wijzigingen in de database of het updaten van dependencies(3) die gebruikt worden door de geschreven software ook gezien als onderdeel van het software distributie proces. Ieder onderdeel noemen we vanaf nu een “subsysteem” en kan binnen het proces 3 acties ondervinden namelijk:

* Vervangen ( komt vaak voor bij het subsysteem van code )
* Wijzigen
* Niks doen

Om het proces van software distributie in kaart te brengen, is het handig om te kijken welke activiteiten er plaats vinden. Het software distributie proces begint met de activiteit “Voorbereiding”. Voordat er een release naar de klant kan worden uitgebracht is het noodzakelijk dat de release “goedgekeurd” is. Voorafgaande stappen om een release goedgekeurd te krijgen zijn:

* Het uitvoeren van integratie testen op ontwikkelomgeving.
* Nieuwe release verwerken in versiebeheer ( voorzien van een tag )
* Geautomatiseerde integratie testen op release in versiebeheer ( door Jenkins )

Zodra alle testen succesvolle resultaten geven, is het tijd om de release voor te bereiden om geüpload te worden naar de klant. Binnen mijn onderzoek ben ik er achter gekomen dat dit op 3 manieren gebeurd namelijk:

* De nieuwe release wordt gedownload door de server van de klant uit de repository van Finalist
* Van de nieuwe release wordt een artifact gemaakt ( zip / war ) en vervolgens geüpload via een deployment tool naar de klant
* Alle artifacts samenstellen in een boom en dit lokaal op klantlocatie uploaden

Nu de software goedgekeurd is kunnen we kijken naar de volgende activiteit. Het software distributie proces ziet er op dit moment als volgt uit:



Afbeelding 1: Proces software distributie stap 1

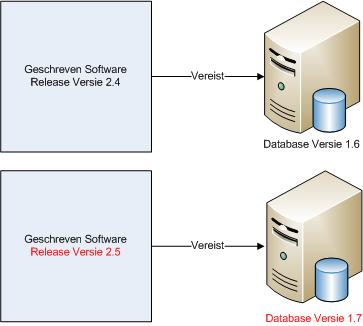
Zodra de nieuwe release op de server van de klant geüpload is via één van de bovenstaande mogelijkheden, is het tijd om de release in productie te laten nemen. Indien de code gedownload is door de server van de klant uit de repository van Finalist, zal deze code direct in de map komen waar de software draait.

Indien de release geüpload is via FTP of op klantlocatie, word de nieuwe release vaak eerst in een tijdelijke map gezet. De nieuwe release wordt dan geïnstalleerd door middel van één of meerdere scripts. Binnen deze scripts worden de acties van de andere subsystemen ook uitgevoerd.

Om een beter beeld te krijgen van andere subsystemen geef ik 2 voorbeelden:

*Voorbeeld 1:*

Wanneer de geschreven software gebruik maakt van een database, zal ook deze indien er wijzigingen zijn aangebracht moeten worden bijgewerkt binnen het software distributie proces.



Afbeelding 2: Voorbeeld subsysteem database wijziging bij uitbrengen release

Het uitvoeren van wijzigingen op de database kan op verschillende manieren gebeuren. Manieren die ik tegengekomen ben binnen mijn interviews zijn:

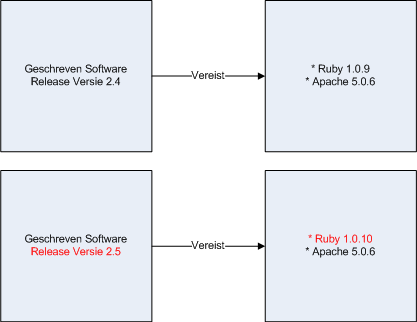
* Wijzigingen op de database via scripts op de server van de klant
* Wijzigingen op de database via tools ( bijv binnen het Ruby framework )

Het boek “Continuous Delivery” van Jez Humble en David Farley onderbouwt dat bij het uitbrengen van een release, wijzigingen aan de database ook behoren bij de oplevering:

*“we must be able to deploy any successful release candidate of our application, including the changes to the database, into production”*

*Voorbeeld 2 :*

Het tweede voorbeeld van een subsysteem is het bijwerken of installeren van dependencies die gebruikt worden door de geschreven software. Dit subsysteem zal alleen uitgevoerd worden wanneer de geschreven software een nieuwe versie nodig heeft van de dependency.

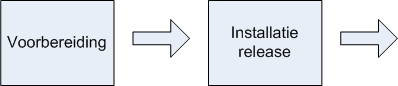


Afbeelding 3: Voorbeeld subsysteem dependencies wijziging bij uitbrengen release

Het boek “Continuous Delivery” van Jez Humble en David Farley geeft aan dat geschreven software in vrijwel alle gevallen gebruik maakt van dependencies:

*“A dependency occurs whenever one piece of software depends upon another in order to build or run. In any but the most trivial of applications, there will be some dependencies. Most software applications have, at a minimum, a dependency on their host operating environment. Java applications depend on the JVM which provides an implementation of the Java SE API, .NET applications on the CLR, Rails applications on Ruby and the Rails framework, C applications on the C standard Library, and so forth.”*

We noemen de activiteit voor het uitvoeren van de acties op de subsystemen “Installatie release”. Het software distributie proces ziet er nu als volgt uit:



Afbeelding 4: Proces software distributie stap 2

Nu de nieuwe release is geïnstalleerd op de server van de klant, komt het voor dat de klant zelf testen uitvoert om de nieuwe release te testen. Een aantal testen die uitgevoerd worden zijn:

* Acceptatie testen
* Performance testen

Echter is het de vraag of deze activiteit binnen het software distributie proces behoort omdat de activiteit door de klant zelf uitgevoerd wordt. Ik heb de keuze gemaakt om dit wel in het proces op te nemen omdat deze testen tot stand worden gebracht door het uitbrengen van een nieuwe release. De activiteit voor de uitvoering van de testen door de klant heet vanaf nu “Testen door klant”



Afbeelding 5: Proces software distributie stap 3

In afbeelding 5 is het totale proces van software distributie te zien.

3.1.2 Server provisioning

In het volgende artikel [Server Provisioning 1] staat een mooi citaat beschreven over de betekenis van server provisioning:

*“Server provisioning is a set of actions to prepare a server, taking it from bare metal to a functioning system complete with an operating system, data and software.”*

Het bovenstaande citaat geeft een eenvoudige maar heldere beschrijving van server provisioning. Server provisioning is een aantal acties voorbereiden die een “nieuwe server” van scratch(2) tot een functionele server compleet met besturingssysteem, data en software kan voorzien. Deze acties zullen automatisch uitgevoerd worden en kunnen toegepast worden op één of meerdere servers tegelijk [Server Provisioning 3] Het toepassen van server provisioning brengt een hoop complexiteit met zich mee. Zo staat er in het artikel [Server Provisioning 1] het volgende citaat geschreven over deze complexiteit:

*“ However, the difficulty is that operating systems have thousands of components with different interfaces for different components. This means that an automated provisioning tool is a complex beast especially if you consider the tasks that are involved in provisioning a server. These include the installation of an operating system, kernel modules, middleware and applications. Further, the organisation will want the system customized to their requirements such as deploying machines with specific roles such as web servers, email servers, with appropriate partitioning and packages. The server will also need to be appropriately configured for the network.”*

Het bovenstaande citaat geeft aan dat de moeilijkheid zit in de verschillende componenten met verschillende interfaces die besturingssystemen hebben. Dit betekend dat de server provisioning software die gebruikt wordt behoorlijk complex in elkaar zit. Naast het installeren van het besturingssysteem, data en software komt het ook vaak voor dat een organisatie de server willen inrichten volgens specifieke eisen ( bijvoorbeeld geïnstalleerde services als web server of email server ). Ook behoort het configureren van de juiste netwerk instellingen onder server provisioning.

### 

3.1.3 Het software distributie proces binnen server provisioning

Zoals te lezen is in hoofdstuk 3.1.2 vallen er naast de installatie van het besturingssysteem en de applicaties nog een hoop andere dingen onder server provisioning. Echter hebben deze dingen geen betrekking op software distributie. Vanaf nu is de term server provisioning afgebakend tot het installeren van de applicaties op de server.

Om het proces van server provisioning in kaart te brengen, is het handig om te kijken welke activiteiten er plaats vinden. Het server provisioning proces begint wanneer een team een verzoek indient voor een nieuwe server bij de systeembeheerder. Binnen dit verzoek geeft het team duidelijk aan welk besturingssysteem en welke applicaties er geïnstalleerd moeten worden en of de nieuwe server intern bij Finalist of extern bij een hostingspartij moet worden geplaatst. De eerste stap heet vanaf nu “Verzoek indienen”. Het proces van server provisioning ziet er nu als volgt uit:



Afbeelding 6: Proces server provisioning stap 1

Wanneer het verzoek binnen is gekomen bij de systeembeheerder, kijkt de systeembeheerder of de nieuwe server intern of extern moet worden geïnstalleerd. Het verschil tussen de installatie van de nieuwe server tussen intern en extern wordt nu beschreven.

*Intern:*

Omdat veel projecten overeenkomende applicaties gebruiken zoals Tomcat, Apache, Ruby etc worden er templates gemaakt. Deze templates zijn vaak gericht op een project uit een team waarbij de “basis” applicaties die vrijwel bij ieder project uit dat team voorkomen voor geïnstalleerd staan.

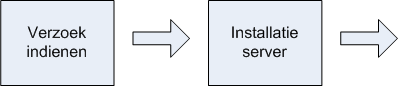
Om veel werk te besparen kopieert de systeembeheerder de template en wijzigt een aantal kleine configuraties zodat de server functioneert als nieuwe server.

*Extern:*

Wanneer de installatie bij een externe hostingspartij plaats vindt, is het niet mogelijk om een standaard template te gebruiken. De systeembeheerder dient een nieuwe installatie te starten van het besturingssysteem en alle gewenste applicaties.

Zowel een interne als externe server wordt voorzien van een ssh key zodat de teams naderhand nog zelf benodigde applicaties kunnen installeren op de server en de systeembeheerder hier niet meer voor nodig is.

De stap voor het installeren van de server noemen we vanaf nu “Installatie server”. Het server provisioning proces ziet er nu als volgt uit:



Afbeelding 7: Proces server provisioning stap 2

Wanneer de nieuwe server geïnstalleerd en geconfigureerd is, krijgt het team dat het verzoek heeft ingediend de gegevens van de nieuwe server door. Indien en nog andere applicaties op de server moeten worden geïnstalleerd kan het team dit zelf doen door middel van de ssh key die geïnstalleerd staat op de nieuwe server. De stap voor het installeren van andere benodigde applicaties noemen we vanaf nu “Installatie benodigde applicaties”. Het server provisioning proces ziet er nu als volgt uit:



Afbeelding 8: Proces server provisioning stap 3

“In afbeelding 8 is het totale proces van server provisioning te zien.”

3.2 Welke stappen binnen activiteiten van het software distributie proces kunnen geautomatiseerd uitgevoerd worden?

Nu we bij de vorige deelvraag antwoord hebben kunnen krijgen op de basis processen binnen software distributie, kunnen we gaan kijken welke stappen binnen de activiteiten geautomatiseerd uitgevoerd worden. Eventuele stappen die nog niet geautomatiseerd zijn, kunnen in het onderzoek worden opgenomen om eventuele verbetering in het proces te adviseren.

3.2.1 Welke stappen binnen een activiteit zijn geautomatiseerd?

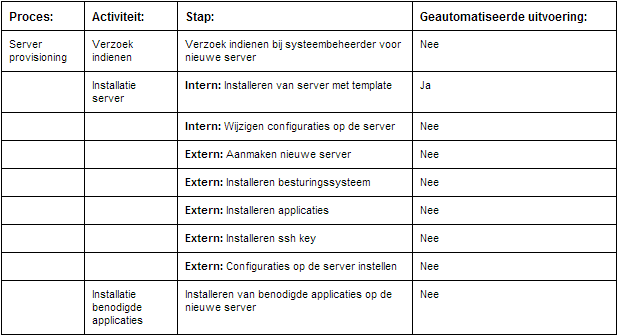
We gebruiken de activiteiten uit de processen die we bij de eerste deelvraag geconcludeerd hebben om duidelijk te maken welke stappen er binnen de activiteiten plaats vinden en of deze stappen geautomatiseerd uitgevoerd worden.

We beginnen met het proces van software distributie:



Note: de blauwe tekst zijn optionele stappen waarvan één uitgevoerd word binnen het proces.

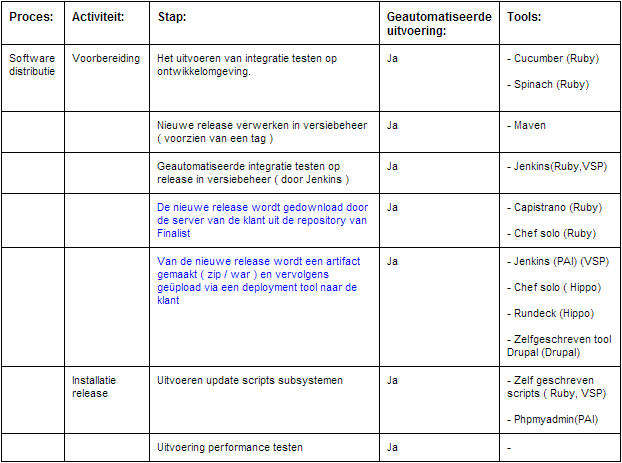
Het proces van server provisioning ziet er als volgt uit:



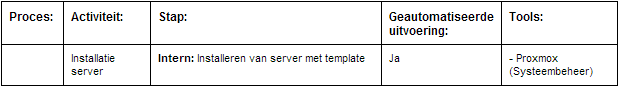
De resultaten of een stap geautomatiseerd uitgevoerd kan worden zijn afgeleid uit de zeven gehouden interviews binnen Finalist [Stakeholder interviews].

3.2.2 Welke tools worden gebruikt bij geautomatiseerde stappen binnen een activiteit?

In hoofdstuk 3.2.1 is duidelijk geworden welke stappen binnen de activiteiten automatisch uitgevoerd worden. Om stappen automatisch uit te voeren worden vaak tools gebruikt. Om duidelijker te maken met wat voor tools Finalist gebruikt maakt, laat ik in een overzicht zien van de geautomatiseerde stappen welke tools ik ben tegengekomen binnen mijn interviews.



Note: de blauwe tekst zijn optionele stappen waarvan één uitgevoerd word binnen het proces.



De benoemde tools zijn afgeleid uit de zeven gehouden interviews binnen Finalist [Stakeholder interviews].

4. Conclusie

De betekenis van software distributie is “het proces van het leveren van software naar een eind gebruiker”. Software distributie is (vaak) een “handmatig intensief” proces. De activiteiten binnen het proces worden stap voor stap uitgevoerd om software te distribueren.

Software distributie pas je toe wanneer je als software ontwikkelingsbedrijf een release van je applicatie hebt geschreven en deze wilt opleveren naar de klant. Ook server provisioning is onderdeel van software distributie waarin het installeren van een besturingssysteem en applicaties op een server uitgevoerd wordt.

Na het houden van verschillende interviews ben ik tot de conclusie gekomen dat binnen de procesbeschrijvingen overeenkomsten zaten. Deze overeenkomsten hebben geleidt tot de conclusie dat er altijd een basis proces nodig is om software te distribueren. Het basis proces bestaat uit 3 activiteiten namelijk:

* Voorbereiding
* Installatie release
* Testen door klant

Het basis proces voor server provisioning bestaat ook uit 3 activiteiten namelijk:

* Verzoek indienen
* Installatie server
* Installatie benodigde applicaties

De 3 activiteiten bestaan uit stappen welke handmatig of geautomatiseerd uitgevoerd worden. Na het houden van verschillende interviews ben ik tot de conclusie gekomen dat vrijwel alle stappen binnen de basis activiteiten geautomatiseerd zijn.

Tot slot heb ik gekeken naar tools die gebruikt worden bij de geautomatiseerde stappen binnen de activiteiten van het basis proces. Ik ben tot de conclusie gekomen dat de tools die gebruikt worden binnen de teams van Finalist verschillend zijn.

# 

5. Bronnen

1. [Server Provisioning 1] 2013, 6 of the best free Linux Server Provisioning tools Geraadpleegd op 15-05-2014

[http://www.Linuxlinks.com/article/20101024141930129/ServerProvisioning.html](http://www.linuxlinks.com/article/20101024141930129/ServerProvisioning.html)

2. [Server Provisioning 2] Mei 2014, Provisioning   
Geraadpleegd op 15-05-2014

<http://en.wikipedia.org/wiki/Provisioning#Server_provisioning>

3. [Server Provisioning 3] Augustus 2010, Nix Craft, Top 5 Open Source Linux Server Provisioning Software  
Geraadpleegd op 15-05-2014  
<http://www.cyberciti.biz/tips/server-provisioning-software.html>

4. [Software Distribution 1] April 2014, Software distribution

Geraadpleegd op 20-05-2014

<http://en.wikipedia.org/wiki/Software_distribution>

5. [Chef 1] Chef documents

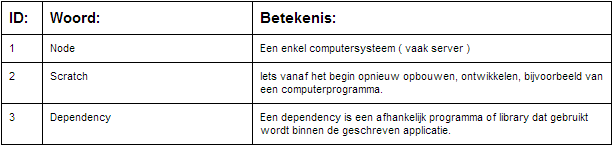
Geraadpleegd op 20-05-2014

<http://docs.opscode.com/chef_why.html>

6. [Stakeholder interviews] 2014, Interview samenvatting van 7 afdelingen binnen Finalist

Geraadpleegd op 20-07-2014

6 Glossary



### 10.3.2 Onderzoeksplan

**Onderzoeksplan**

Onderzoek software distributie

Welke software distributie tool sluit het beste aan op de software distributie binnen het Webapps team?



Auteur: Ruben Bouterse Eerste docent: H.P.Weenink

Opleiding: Informatica Tweede docent: G.M.Tuk

Bedrijf: Finalist Begeleider: R.Schellhorn

Opdrachtgever: M. van Breukelen

Datum: 30-07-2014

Versie: 2.0

Inhoudsopgave



Inleiding

Dit document is geschreven ter voorbereiding voor mijn onderzoek naar software distributie. Dit document kan worden gezien als “plan van aanpak”.

Inhoudelijk is te lezen welke aanleiding, probleemstelling en doelstelling relevant zijn op het onderzoek naar software distributie. Vervolgens zal de hoofdvraag geformuleerd worden en de bijbehorende deelvragen welke samen het antwoord op de hoofdvraag zullen concluderen.

Tot slot zal de aanpak van het onderzoek worden behandeld waarin de deelvragen te lezen zijn met bijbehorende onderzoeksmethode(n) en zal de kwaliteit waarborging besproken worden.

# 

# 

1. Onderzoekskader

1.1 Aanleiding

De aanleiding van dit onderzoek is dat de eisen die klanten stellen voor software distributie vergeleken met vroeger steeds hoger worden. Zo zijn klanten meer gesteld op zo weinig mogelijk downtime van hun systemen en een kortere tijdsduur van software distributie doordat automatisering steeds meer een rol is gaan spelen. Finalist heeft aangegeven dat ze graag een onderzoek willen starten naar verbetering binnen de software distributie. De focus zal hierbij liggen op het onderzoeken naar vervangende tools die gebruikt kunnen worden binnen het proces.

1.2 Probleemstelling

Binnen de software distributie van het Webapps team zijn een aantal knelpunten die voor belemmering zorgen. Deze knelpunten zijn:

* Het downloaden van gem dependencies(1) wordt gedaan van een externe bron.
* Het downloaden van gem dependencies valt binnen een vaste uitvoeringsvolgorde van de software deployment tool.
* Het compileren van assets(2) valt binnen een vaste uitvoeringsvolgorde van de software deployment tool.
* Wanneer een provisioning dependency geüpdatet moet worden, verschijnt er geen foutmelding.
* Het server provisioning en software deployment proces gebruiken verschillende tools.
* Het verwijzen van de symlink(3) naar de meest recente software release, valt binnen een vaste uitvoeringsvolgorde van de software deployment tool

Het gevolg van bovenstaande knelpunten is voornamelijk dat het veel tijd kost om software distributie uit te voeren. Bij de uitvoering van het onderzoek zullen de huidige processen van software distributie in detail worden uitgeschreven. Dit levert mogelijk nog een aantal nieuwe knelpunten op welke eventueel gebruikt kunnen worden in het onderzoek.

Omdat Finalist onderverdeeld is in meerdere teams, spelen er mogelijk nog vele andere knelpunten. Deze knelpunten van andere teams kunnen nader worden onderzocht of worden vergeleken met het opgestelde advies voor het Webapps team om te kijken of deze opgelost zijn.

De totale probleemstelling kan worden gezien als “Het huidige probleem van het Webapps team” zoals vermeld wordt in de hoofdvraag.

1.3 Doelstelling

Het doel van dit onderzoek is antwoord te krijgen op de hoofdvraag “Welke software distributie tool sluit het beste aan op de software distributie binnen het Webapps team?”. Met het antwoord op deze vraag kan ik gezien het resultaat verbeteradvies opstellen voor de software distributie binnen het Webapps team van Finalist.

1.4 Hoofdvraag

De hoofdvraag van mijn onderzoek:

* Welke software distributie tool sluit het beste aan op de software distributie binnen het Webapps team?

1.5 Deelvragen

* *Hoe voert het Webapps team software distributie uit?*

Binnen deze deelvraag wil ik antwoord krijgen over de huidige uitvoering van software distributie van het Webapps team. Door de processen binnen de software distributie in detail uit te schrijven, kan er informatie over de knelpunten worden vast gesteld. De knelpunten vormen het huidige probleem van het Webapps team.

* *Welke knelpunten treden op binnen de software distributie van het Webapps team en hoe zijn deze op te lossen en hoe zijn deze op te lossen?*

Binnen deze deelvraag wil ik knelpunten toelichten die zich afspelen binnen de software distributie van het Webapps team. Vervolgens zal een beschrijving van een ideaal toekomstbeeld mogelijkheden moeten bieden om oplossingen te vinden voor de opgestelde knelpunten. Wanneer de oude en ideale toekomst situatie beschreven zijn, worden er requirements opgesteld welke verwerkt worden in een requirementsrapport.

* *Welke software distributie tools zijn geschikt voor het opstellen van de longlist en welke tool is het meest interessant om te testen in praktijk?*

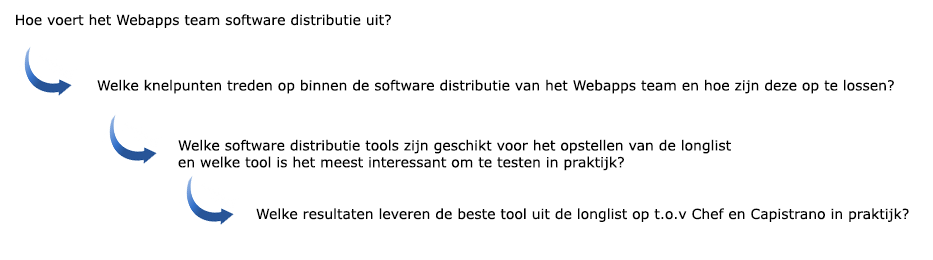
Binnen deze deelvraag wil ik een aantal software distributie tools beschrijven die mogelijk de oplossing kunnen bieden voor de knelpunten uit de vorige deelvraag. Aan de hand van requirements wil ik criteria opstellen waaraan de tools moeten voldoen. Door een literatuuronderzoek te starten, wil ik informatie verzamelen over beschikbare tools en de opgestelde criteria om te bewijzen of de tools wel of niet in staat is hieraan te voldoen.

* *Welke resultaten leveren de beste tool uit de longlist op t.o.v Chef en Capistrano in praktijk?*

Binnen deze deelvraag wil ik de meest geschikte software distributie tool uit de vorige deelvraag testen in praktijk. Door middel van verschillende scenario’s die betrekking hebben tot de geselecteerde knelpunten uit deelvraag 2 te verwerken in testcases, hoop ik op een duidelijke manier aan te kunnen tonen welke tool de beste mogelijkheden biedt als oplossing voor het huidige probleem van het Webapps team.

1.5.1 Samenhang deelvragen

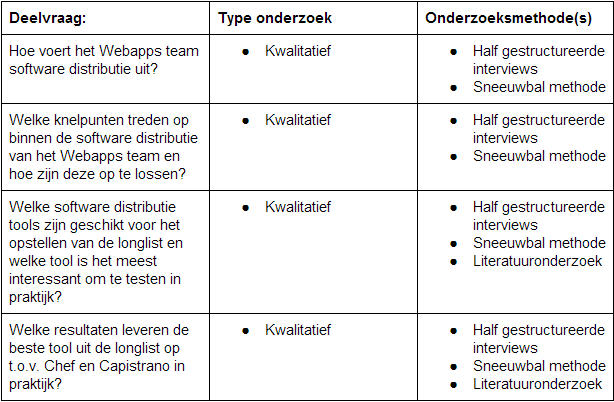
Sommige deelvragen zijn afhankelijke van voorgaande deelvragen. Om de samenhang van de deelvragen te verduidelijken, is hieronder een afbeelding weergegeven:



Zoals te zien is in de afbeelding dienen de deelvragen in volgorde beantwoord te worden.

1.6 Aanpak

Binnen het onderzoek zal kwalitatief onderzoek worden verricht. Hieronder een overzicht met de deelvragen en bijbehorende onderzoeksmethode welke gehanteerd zullen worden:



1.7 Kwaliteit waarborging

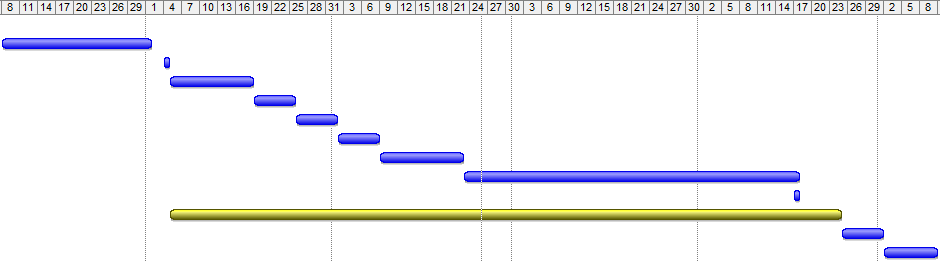
Om de kwaliteit van mijn onderzoek te kunnen waarborgen, zorg ik ervoor dat de conclusies die worden getrokken gebaseerd zijn op informatiegegevens die afkomstig zijn uit betrouwbare bronnen. Enkele bronnen zijn:

* Interviews binnen Finalist
* Wikipedia ( Engels )

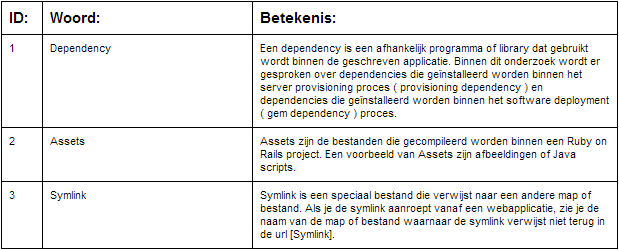
2. Planning

Helaas werd ik al snel binnen mijn afstudeertraject ernstig ziek waardoor ik een vertraging van tien weken heb opgelopen. Ik heb de planning opgesteld met als tijdsduur de nieuwe einddatum van mijn project. Deze planning ziet er als volgt uit:

## 



3. Glossary



### 10.3.3 Vragen half-gestructureerd interview

**Proces eerste keer software distribueren voor de klant:**

**Server Provisioning:**

Als ik het goed begrijp schrijven jullie binnen het team …..(specifiek voor team bijv: Drupal, Java) software. Stel jullie hebben een nieuw project en willen voor de eerste keer software leveren en de klant geeft aan dat Finalist het beheer mogen doen. Installeert/configureert jullie afdeling de server? of Andre misschien?

**Eventueel doorvragen:**

Kun je het proces van server provisioning van het begin tot het einde beschrijven en vertellen wat voor scenario’s/uitzonderingen er zoal voorkomen?

Is er documentatie aanwezig van dit proces?

Welke tools binnen het integratie team worden er gebruikt voor het inrichten van de server?

Welke services en besturingssystemen worden er geïnstalleerd op de server voor de klant?

Zijn er problemen waar jullie tegenaan lopen bij het inrichten van een server van de klant

**Proces nieuwe release/patch software distribueren voor de klant:**

**Software distribution:**

Kun je het proces van software distributie van het begin tot het einde beschrijven en vertellen wat voor scenario’s/uitzonderingen er zoal voorkomen?

**Doorvragen:**

Is er documentatie aanwezig van dit proces?

Welke tools gebruiken jullie om de software te distribueren?

Waar hebben jullie de geschreven software opgeslagen? (svn repo, websvn? )

Tegen wat voor problemen lopen jullie aan bij het distribueren van de software?

Als de klant jullie geen rechten of login gegevens verschaffen, hoe leveren jullie dan patches/releases op? ( download.finalist.com ?? )

Hoe testen jullie of de software release succesvol was? ( handmatig / automatisch? )

Hoe belangrijk is de uptime van de applicatie voor de klant bij het releasen van software? ( Is er een afgesproken downtime voor maintenance?

**Proces software release/patch herstellen:**

Wanneer het uitrollen van een nieuwe release/patch voor problemen zorgt bij de klanten, hoe voeren jullie een rollback van de software uit?

Hoe is het versiebeheer van de applicaties van het team geregeld.

Waar staan backups opgeslagen? ( Worden er releases bijgehouden op de server van de klant of worden deze vanaf Finalist weer geüpload ).

Hoe testen jullie of de software release succesvol was? ( handmatig / automatisch? )

**Algemeen:**

Zijn er nog andere dingen die misschien belangrijk kunnen zijn voor mijn onderzoek?

### 10.3.4 Samenvatting interviews stakeholders

#### 10.3.4.1 Webapps team ( Rob Schellhorn )

**Het proces van server provisioning:**

De server provisioning wordt door de ontwikkelaar zelf gedaan met gebruik van de tool Chef.

\* De tool Chef wordt geïnstalleerd op de server en vervolgens gestart

\* Cookbooks worden gekopieerd van de laptop van de ontwikkelaar naar de server  
\* Gecontroleerd op Cookbook dependencies en worden indien niet aanwezig op de server geïnstalleerd.  
\* Downloaden overige cookbooks welke niet aanwezig waren op de laptop van de ontwikkelaar  
\* Uitvoeren van Runlist

**Het proces van software deployment:**

\* Capistrano voert enkele controles uit  
\* Working directory aangemaakt met de naam van een timestamp

\* De nieuwe release wordt gedownload door de server van de klant uit de repository van Finalist.

\* Bundle wordt uitgevoerd voor het downloaden en installeren van gem dependencies.

\* De assets van het Ruby on rails project worden gecompileerd  
\* Database migraties worden uitgevoerd  
\* Current ( Symlink ) verwijzen naar map met de laatste timestamp

\* Opruimen van tijdelijke mappen  
\* Logs bijwerken  
\* Eventuele fouten wegschrijven naar het programma Airbreak

**Testen binnen releases:**

* Het Webapps team past integratie testen toe op de projecten die ze schrijven.
* De testen worden uitgevoerd via het Cucumber en Spinach framework in Ruby on Rails.

**Back-up plan:**

Het terugplaatsen van een back-up gaat eenvoudig door de symlink die gebruikt wordt te verwijzen naar de vorige uitgebrachte release.

**Back-up databases:**

* Andre maakt voor databases die gehost worden door Finalist back-ups
* Bij externe hostings partijen zijn de externe hostings partijen zelf verantwoordelijk voor het maken van database back-ups.

**Punten algemeen:**

* De huidige tools die momenteel gebruikt worden binnen het Webapps team zijn open source. Dit komt omdat Finalist zijn voorkeur heeft om zoveel mogelijk gebruikt te maken van Open Source oplossingen en tools.
* Binnen het Webapps team wordt gewerkt op een laptop met Mac OSX.
* Binnen het Webapps team wordt geprogrammeerd binnen een command-line interface.
* Het besturingssysteem van de servers is binnen de projecten van het Webapps team een Linux besturingssysteem.
* Binnen het Webapps team worden Ruby on Rails projecten gemaakt.

**Toekomst:**

* Het zou mooi zijn als de server provisioning en software deployment binnen één tool gerealiseerd zou kunnen worden.
* De tool Capistrano welke gebruikt wordt voor het software deployment proces maakt gebruik van een vaste uitvoeringsvolgorde waarin stap voor stap de release uitgebracht wordt op de server van de klant. Het Webapps team zou het proces graag willen opsplitsen in twee subprocessen. Het eerste proces bevat de oplevering van de release maar deze wordt echter alleen uitgebracht op de server van de klant op de niet live omgeving. Het tweede subproces zorgt ervoor dat de eerder opgeleverde release op de live omgeving wordt gebracht en eventuele database migraties uitgevoerd worden.
* Wanneer een provisioning dependency geüpdatet moet worden, wordt dit nergens duidelijk getoond in het software deployment proces. Het Webapps team zou het handig vinden als de toekomstige tool een melding kan tonen wanneer een provisioning dependency geüpdatet moet worden.

#### 10.3.4.2 Drupal team ( Tom van Vliet )

Binnen het drupal team is een voormalig systeembeheerder werkzaam die voor een project kan zien welke configuraties en software installaties er nodig zijn bij server provisioning. Wanneer het Drupal team een nieuwe server voor een klant wil laten inrichten laten ze dit doen door Andre. De voormalige systeembeheerder legt van te voren vast wat de server allemaal nodig heeft zodat dit als kant-en-klaar verzoek bij Andre neergelegd kan worden.

Het drupal team heeft zelf een tool geschreven voor het deployen van hun applicaties naar de klant ( tool is gebaseerd op drupal en puur voor deployen vanuit git repo’s). Hierbij maken ze gebruik van drush ( command line tool voor drupal ). Je kunt met drush omdat het command based is sneller zaken uitvoeren dan dat je dat met een UI doet. De zelfgeschreven tool maakt geautomatiseerd gebruik van het FTP protocol.

De zelfgeschreven tool heeft een interface waarin je eenvoudig kunt kiezen welke features je wilt uitrollen naar de klant of bijvoorbeeld aan kunt geven dat je een back-up wilt maken. De gemaakte keuzes roepen zo hun bijbehorende scripts aan welke uitgevoerd worden in de drush. Deze scripts bestaan ieder weer uit aparte commando’s.

**Proces software deployment:**

Release naar acceptatie omgeving:

\* Klant geeft formeel akkoord

\* Release plannen in samenwerking met de klant

\* Release uitrollen wat gedaan wordt volgens onderstaand proces

**Proces zelfgebouwde tool:**

Proces zelfgebouwde tool in globale stappen:

\* Je selecteert een tag uit de git repo ( stash ).

\* Je zet features aan

\* Updates aanzetten die relevant zijn

\* Klik knopje deploy om de software naar de klant te deployen

**Back-up:**

Het uitvoeren van een rollback is goed te doen met de zelfgeschreven tool. Het drupal team kan tamelijk eenvoudig een back-up maken. Binnen de zelfgemaakte tool kun je eenvoudig kiezen om terug te gaan naar een vorige release.

Alles wordt goed bijgehouden in versiebeheer. Zo ook welke features er uitgerold zijn. In feite kun je vanuit versiebeheer ook weer een rollback uitvoeren.

**Testen:**

Testen gebeuren geautomatiseerd bij grotere projecten. Dit is niet ingebouwd in de zelfgebouwde tool ( deze is echt puur geschreven voor deployment ).

.

Binnen het drupal team worden de volgende testen uitgevoerd:

* Integratietesten
* Selenium testen
* Simple test framework voor drupal ( soort van Unit test )

Geautomatiseerde testen worden alleen geschreven als daar budget voor is vanuit de klant.

**Uitzondering klant Universiteit Utrecht:**

Universiteit Utrecht heeft een eigen beheerpartij voor hun software. Finalist is echter alleen de ontwikkelpartij. Finalist dient de applicatie kant-en-klaar aan te leveren zodat de andere beheerpartij zelf de releases kunnen deployen.

De beheerpartij heeft toegang tot de git repo van Finalist. Zodra Finalist een tag/commit heeft gedaan, kan de beheerpartij de nieuwe release ophalen en releasen. De beheerpartij is in dit geval verantwoordelijk voor de features ( die moet Finalist dus aanleveren ). Features zijn algemeen voor drupal.

Finalist legt de focus voor Universiteit Utecht goed op hun tests en code review omdat het dus goed moet werken. Allemaal voordat het naar de beheerpartij gaat. De beheerpartij test ook de software nog een keer voor ze het echt releasen. De tag is eigenlijk de eindstap voor Finalist voor deze klant.

**Punten algemeen:**

* Bij drupal heb je geen builds die je klaarzet. In feite is het een code release en daarnaast configuratie die je over moet brengen ( bijvoorbeeld database ). Je zult scripts moeten draaien om ervoor te zorgen dat die configuratie mee gaat naar de klant ( niet allemaal met het handje ).
* feature = een module die algemeen bekend is binnen drupal. een feature bestaat uit configuraties. Bij het uitrollen van een feature worden de configuraties omgezet naar code en kun je zo eenvoudig toepassing binnen de hele OTAP straat.
* Downtime van servers is minimaal geworden door zelfgeschreven tool.
* Niet elke klant bij het drupal team heeft een acceptatie omgeving. Helaas komt het voor dat er verschillen zijn tussen de test en productie omgeving van een klant. Hierdoor is het vervelend wanneer er geen Acceptatie omgeving bestaat.
* Rundeck kun je drupal en andere programmeertalen deployen.
* Er zijn een aantal projecten binnen het drupal team waar de zelfgemaakte tool niet gebruikt kan worden. Hier wordt gewoon handmatig gebruik gemaakt van FTP.

**Punten bepalend hoe deployment kan worden uitgevoerd:**

Complexiteit opdrachtgever

SLA en contracten

Technische keuzes van een product zelf ( programmeertaal maar ook wat voor product je hebt (cms of niet cms?, hoe belangrijk is het cms in de wereld,)

Open source of niet open source? ( hangt er een bedrijf achter die ondersteuning biedt of bijvoorbeeld alleen een community? )

OTAP omgeving bij de klant? ( infrastructuur )

**Toekomst:**

Het drupal Team ziet er toekomst in om proactief bezig te zijn voor klanten. Hiermee bedoelen ze dat ze bij ieder project een voorstel kunnen aanbieden over hosting, beheer etc.

Finalist denkt erover dat het misschien mogelijk is dat hun zelfgemaakte tool ook voor klanten bruikbaar kan worden. Dan ga je naar betaalmodel en kunnen klanten zelf een aantal deploys op eigen initiatief doen.

#### 10.3.4.3 Hippo team ( Brian Snijders )

de server provisioning wordt door Andre van Elst gedaan door middel van een template VM met ssh key voor Chef solo.

**Het proces van server provisioning:**

\* Hippo team vraagt aan Andre van Elst om een kopie van een VM te maken die past bij het project.

\* Zodra deze omgeving klaar is neemt het Hippo team het over van Andre van Elst en installeert de applicaties die niet in de standaard VM zitten via Chef solo.

**Het proces van software distributie:**

Er worden twee manieren van software distributie gehanteerd binnen het Hippo team namelijk continiously build en release build.

Het proces voor continiously build is geautomatiseerd en ziet er als volgt uit:

\* Je schrijft een feature.

\* Je commit de feature.

\* Jenkins laat via triggers een continious deploy uitvoeren naar de productie omgeving van de klant.

De continiously deploy van Jenkins ziet er als volgt uit:

\* Jenkins maakt een snapshot build het project ( de broncode ) uit de git repository

\* Rundeck download de laatste snapshot build die gebuild is door Jenkins

\* De nieuwe release wordt geüpload naar de node(s) van de klant in een tijdelijke map.  
\* Tomcat word uitgezet op web applicatie server

\* Nieuwe code wordt geplaatst

\* Tomcat wordt aangezet en initialiseert de nieuwe release.

Bij een release build wordt te code getagged. De getaggede release wordt handmatig via de deployment tool Rundeck uitgevoerd. Releasejobs in Rundeck kunnen alleen release builds uitrollen, om te voorkomen dat een snapshot build per ongeluk in productie gezet wordt.

**Testen binnen releases:**

* Binnen het Hippo team zijn er projecten waar de testers veel betrokken zijn bij de sprintjes van scrum. In dit geval is continious integration toepasbaar.
* Binnen het Hippo team zijn er ook projecten die met een waterval methode verlopen waarbij de testers ergens op het einde van de waterval aan bod komen. In dit geval wordt er vaak een release uitgebracht via een getaggde release. Zodra de testers de release goedkeuren wordt de release via Jenkins deployed naar de klant.

**Back-up plan:**

back-ups worden op twee manieren gemaakt namelijk:

\* Door middel van het uitrollen op één VM. Zodra de oplevering succesvol is pakken ze de configuratie en de software van die VM en rollen ze uit over het hele cluster.

\* Het kan ook op het cluster zelf. Bijvoorbeeld 3 nodes met een database waarbij de database onderliggend nog een database draait en op het cluster een extra repo repository inleest waarmee je vervolgens in je configuratie van Tomcat switcht van database en repo repository locatie. Gaat het goed dan laat je het staan en ruim je de oude release op. Gaat het niet goed dan pas je de configuratiefile van Tomcat weer aan naar de oude locaties van de database en repo repository. Het nadeel van deze manier is dat je dubbel zoveel schijfruimte nodig hebt omdat je een dubbele database en repo repository hebt.

**Back-up databases:**

* Andre maakt voor databases die gehost worden door Finalist back-ups

Bij externe hostings partijen zijn de externe hostings partijen zelf verantwoordelijk voor het maken van database back-ups.

**Punten algemeen:**

* Rundeck is een workflow automation tool voor Devops teams ( je voert eigenlijk een paar bash scriptjes achter elkaar uit ).
* Er is documentatie aanwezig op de wiki ( wanneer is iets een continious build of release build?, systeemoverzicht van alle systemen die het Hippo team gebruikt ).
* Er zijn klanten waar eerst een vpn verbinding tot stand moet worden gebracht. Via de vpn tunnel moet een nieuwe release worden opgeleverd.
* Omdat het Hippo framework niet gebouwd is door Finalist, is het niet de verantwoordelijkheid van Finalist om deze te testen.
* Het Hippo team probeert zijn nieuwe klanten hosting af te laten nemen bij de cloud dienst van Hippo. Hierdoor kunnen ze zich zoveel mogelijk richten op één standaard oplossing/werkwijze voor de oplevering van hun software.
* Omdat de Hippo projecten gebruik maken van het Hippo framework, hoeft het Hippo team stukken minder code te typen dan een maatwerk project. De code die voornamelijk getypt wordt is integratie code.
* Uit ervaring van het verleden is gebleken dat het overrulen van standaard functionaliteiten binnen het Hippo framework behoorlijk wat problemen kan opleveren. Dit wordt dus ook niet meer toegepast binnen Hippo projecten.
* Op de server van de klant staat altijd maar één release van de software namelijk degene die in productie is.
* Het distribueren van de software gaat volgens Continious Integration.
* Binnen de git repository worden node configuraties bijgehouden omdat Rundeck zelf niet beschikt over de mogelijkheid om nodes toe te voegen. Jenkins maakt voor het uitvoeren van een release een build van de node configuratie en laat deze downloaden door Rundeck zodat de nodes bekend zijn.

**Toekomst:**

* Misschien is het wel mogelijk om de software distributie uit te besteden door middel van een Saas oplossing?
* Finalist gebruikt steeds vaker monitoring software die de servers van de klant onder toezicht houdt op fouten. Echter is dit een stap buiten het software distributie proces.

#### 10.3.4.4 Portal solution team ( Lennaert van der Linden )

Voor de grote projecten van het Portals and Integration team, ligt de verantwoordelijk voor de server provisioning bij de klanten zelf of bij hun shared host provider.

Binnen het Portals and Integration team worden voornamelijk Php maatwerk en Drupal projecten ontwikkeld.

**Proces per project:**

**Kennisnet:**

Het Portals and Integration team heeft geen toegang tot de daadwerkelijke productie en acceptatie omgeving van Kennisnet. Het proces eindigt daarom bij het opleveren van een release ( zip bestand ) met instructies ( oude bestanden vervangen voor nieuwe bestanden, lijstje met migratie scripts voor database) via FTP. Kennisnet zorgt vervolgens voor de deployment van de release.

Het Portals and Integration team heeft wel toegang tot een testomgeving van Kennisnet. Voor deze omgeving is een Jenkins taak geschreven waarin de software geautomatiseerd deployed wordt.

Omdat kennisnet zelf het deployen van de software op de acceptatie en productie omgeving doet, is Finalist niet verantwoordelijk voor de eventuele downtime. Kennisnet is verantwoordelijk voor eigen versiebeheer en heeft geen toegang tot het versiebeheer van Finalist.

**Teleblik:**

stappenplan Wiki release testomgeving kennisnet teleblik **( nog handmatig )**:

\* release nummer bijwerken in Netbeans project

\* script .export [versienummer] maakt een mapje aan en staan de bestanden in

\* script release [versienummer] kopieert bestanden naar de testserver

\* ssh -l ontwikkelteleblik.nl ( verbinding maken naar de testserver )

\* sudo install version.shell [versienummer] kopieert alle bestanden naar /var/www folder op de testserver

\* test.teleblik.nl

\* maak een tag aan op de kennisnet reversion repository

\* maak een tag aan op de kennisnet git repository

\* Email sturen naar de beheerder kennisnet

Finalist plaatst de releases in het versiebeheer systeem van de klant.

**Linked data:**

Voor Linked data zijn er geautomatiseerde Jenkins taken geschreven. Zo is er een Jenkins taak geschreven die de software klaar maakt tot een release. De Jenkins taak ziet er als volgt uit:

\* Het ophalen van de geschreven software uit het revisie systeem.

\* Het uitpakken van de geschreven software

\* Opschonen van Jenkins omgeving

\* Externe bibliotheken ophalen

\* Unit testen draaien ( bij het falen van unit testen wordt er niet gereleased )

\* Documentatie gegenereerd

\* Zip bestand gemaakt

\* Zip bestand gekopieerd naar de download.finalist.com server.

Een andere Jenkins taak is het distribueren van de release naar de klant. De Jenkins taak ziet er als volgt uit:

\* Het ophalen van de geschreven software.

\* Geautomatiseerde Unit testen draaien.

\* Custom deploy scriptje die applicatie via ftp protocol naar de web server van de klant upload.

Omdat het Portals and Integration team geen root toegang heeft naar zijn klanten, moeten database wijzigingen handmatig worden doorgevoerd. Dit wordt gedaan doormiddel van database migratie scripts via phpmyadmin.

**Punten Algemeen**

* Via FTP worden nieuwe releases geüpload naar de testomgeving van klanten die toegang geven tot hun testomgeving.
* Jira: word bijgehouden welke issues er in welke release behandeld zijn ( puur voor management )
* Jenkins: systeem dat gebruikt wordt voor geautomatiseerd bouwen, testen en deployment van software.
* Build triggers = de tijd die je instelt bij Jenkins (hoe vaak er gekeken moet worden of de Jenkins taak moet worden uitgevoerd).
* Portals and integration team hebben hun testen geautomatiseerd en proberen als het mogelijk is automatisch hun software te distribueren naar de klant.
* Op de wiki staan standaard release procedures ( stappenplan ) per project.
* Finalist houdt per release een bestand bij waarin de versienummers van de releases staan met bijbehorende database migratie scripts.

**Toekomst**

Na het plaatsen van de software via FTP bij de klant moet er nog een email worden verstuurd naar de beheerders bij de klant. Dit proces gebeurt momenteel nog handmatig en zou eventueel geautomatiseerd kunnen worden binnen de Jenkins taak.

#### 10.3.4.5 Integratie solution team ( Ton Swieb )

Het Integratie team is niet verantwoordelijk voor de server provisioning voor hun klanten. Wanneer een klant van het Intergratie team ervoor kiest om de server provisioning door Finalist te laten doen, verzorgt Andre van der Elst vaak de server provisioning.

Binnen het integratie team wordt ESB ( Enterprise Service Bus ) software ontwikkeld. Het distribueren van ESB software gaat volgens een deployment model. Het Integratie team heeft meerdere deployment modellen. Op basis van de organisatie infrastructuur van de klant, wordt het meest toepasbare deployment model toegewezen. Globale voorbeelden van deployment modellen zijn:

* Script schrijven waarin aangegeven wordt waar de juiste repository staat met de juiste artifacts in plaats van een assembly naar de server te kopiëren. Vervolgens gaat de applicatie in Nexus kijken of hij alle dingen kan vinden en gaat deze installeren.
* Deployment doen door alles direct op de server van de klant te plaatsen.
* Alle artifacts van te voren samenstellen in een boom en die lokaal bij de klant distribueren.

**Proces per klant:**

**Klant BNL:**

Het distributie proces voor deze klant begint met het opleveren van een script en Assembly via het Unix programma “Secure Copy”. De Assembly bevat de benodigde Artifacts die nodig zijn om de laatste versie van de software te kunnen draaien en met welke configuratie.

Wanneer het script en Assembly overgedragen zijn naar de server van de klant, wordt het script uitgevoerd. Het script verwijdert de oude artifacts en configuraties en plaatst de nieuwe artifacts en configuratie op de server.

Vervolgens wordt er na een aantal seconden een jmeter script gestart waarin geautomatiseerde integratie testen worden uitgevoerd. De resultaten van het jmeter script wordt terug gestuurd en als er testen falen wordt de logfile erbij gestuurd zodat je kunt analyseren wat er mis is.

Tot slot wordt er binnen het script een back-up gemaakt van de oude versie ( gearchiveerd ) en wordt er een back-up van de logfiles gemaakt. Van de oude configuratie en artifacts wordt geen back-up gemaakt. Het archief van assemblys en logfiles staat op datum en kan worden gebruikt om een rollback van een aantal versies uit te voeren. Tot slot start de applicatie op.

De documentatie over het deployment model is uitgeschreven ( staat op de wiki ).

**Klant Hogeschool Utrecht:**

Het distributie proces voor deze klant begint met het opleveren van een bundel en feature file waarin de dependencies staan. De oplevering wordt gedaan via het Unix programma “Secure copy”. Wanneer de bundel en feature files overgedragen zijn naar de server van de klant, voert Hogeschool Utrecht zelf de distributie uit ( ze plaatsen de artifacts in de juiste map en worden vervolgens automatisch geïnstalleerd ).

Hogeschool Utrecht is zelf verantwoordelijk voor het uitvoeren van testen omdat ze zelf de distributie van de release uitvoeren. Finalist test of de software werkt, vervolgens maken ze een release en levert Finalist het op. Soms bellen ze als het niet werkt om te vragen of Finalist bij kan schakelen maar in principe eigen verantwoordelijkheid om te testen.

Voor Hogeschool Utrecht is geen documentatie bij Finalist aanwezig over het distributie proces omdat ze zelf de deployment doen maar mogelijk wel bij Hogeschool Utrecht.

**Klant AZN:**

Het distributie proces van AZN is afgeleid van het distributie proces van BNL. Ook hier wordt gebruik gemaakt van een script.

De documentatie van het deployment model is uitgeschreven ( staat op de wiki ).

**Punten algemeen:**

* Bij iedere klant heeft Finalist rechten tot de applicatie folder van de klant voor oplevering van releases
* De grote klanten van het Integratie team zijn: BNL, Hogeschool Utrecht en AZN.
* Het Integratie team maakt geen gebruik van software distributie software maar levert de applicatie op via Secure copy bij de klant. Indien het Integratie team het deployment model gebruikt waarin ze gebruik maken van een script, worden er alleen enkele standaard Unix utilities gebruikt ( sturen van mailtje, zip / unzip etc ).
* In het SLA staat geplande downtime afgesproken voor het uitbrengen van releases.
* Git-Svn, Stash ( git repo ) (stash.finalist.com ) staan software repositories met de code van de projecten.
* Maven zorgt ervoor dat de dependecies worden opgehaald welke gebruikt worden voor compilatie ( zorgt er bijvoorbeeld voor dat er wat configuratie files goed gezet worden ). Dit komt allemaal in een jar file. in de svn repo wordt een tag gezet. De versie met de tag beland in Nexus als .jar bestand, waar deze vervolgens weer uit kan worden gedownload.
* Jenkins proces: het uitchecken van je code, het taggen van je code, het compileren van je code ( gebruik van Maven bij compileren ), het opslaan van de artifact in Nexus als 1 stap/scriptje. Jenkins wordt gebruikt om geautomatiseerd releases te builden en op te slaan binnen Nexus zodat deze gereleased kunnen worden naar de klant.
* Integratie team maakt gebruik van Applicatie server Jboss fuse ( is een platform voor ESB ).

**Toekomst:**

Zijn nog zoekende naar wat het beste voor de klant werkt. complete assembly? of incrementeel? per klant verschillend met verschillende rechten. Per klant zijn het aparte afspraken.

We zouden een standaard manier van releasen/deployment kunnen aanbieden en dan voorleggen aan de klant.

Het rollbacken van een release is niet handig om dat vanuit de Nexus repository te doen omdat het soms om 100mb gaat. Er is gesproken of het niet handig is om snapshots te maken van de virtual machine zodat je zeker weet dat er wat mist. ( maar dit is een applicatie beheer verantwoordelijkheid dus zal weer per klant verschillend gelden ). Per klant zouden er eigenlijk expliciete afspraken gemaakt moeten worden over wie back-ups maakt ( desnoods hostingspartij die snapshots maakt van de VM voor x aantal dagen). Is zeker nog een verbeterpuntje voor de toekomst.

#### 10.3.4.6 VSP team ( Wietse Venema )

Het Netwerk VSP team is niet verantwoordelijk voor de Server provisioning. Indien ze een nieuwe server willen inrichten, dienen ze contact op te nemen met de externe hostingspartij die Netwerk VSP gebruikt. Zodra de externe partij een server heeft aangemaakt krijgt Finalist beperkte rechten waarmee ze hun applicatie kunnen opleveren naar de server. Voor wijzigingen op de acceptatie en productie omgeving ( bijvoorbeeld het bereiken van een andere server of poorten die open gezet moeten worden ) moet het Netwerk VSP team mailen naar Netwerk VSP die vervolgens weer contact opneemt met de externe hostingspartij.

**Proces per klant:**

**Netwerk VSP:**

Binnen Netwerk VSP wordt gebruik gemaakt van JBoss applicatie servers. Het globale proces van Software Deployment ziet er als volgt uit:

**Jboss applicatie server stappen:**

\* Applicatie afsluiten

\* Nieuwe jar files in mapje zetten

\* Configuratie wijzigingen doorvoeren

\* Applicatie weer opstarten (Start niet op wanneer er iets mis gaat. bijvoorbeeld als er geen database connectie gemaakt kon worden start de applicatie niet meer op en dient Finalist de problemen te analyseren en te verhelpen)

Het Netwerk VSP team heeft hun gedeelte van het uitbrengen van een release geautomatiseerd door middel van bash scripts. Hieronder de stappen die uitgevoerd worden met bash scripts:

**Automatisering met bash scriptjes die nu gehanteerd word bij netwerk VSP:**  
\* Bij netwerk VSP staat een inlog server waar alle release data naartoe gekopieerd word

\* Vanaf de login server wordt de data naar alle productie servers gekopieerd

\* Alle productie omgeving servers gestopt

\* Nieuwe release geïnstalleerd

\* Database scriptjes met wijzigingen runnen ( ze gebruiken een Oracle database )

\* Productie servers weer gestart

Bij Netwerk VSP wordt een twee wekelijkse release schema gehanteerd. Met de automatisering die nu toegepast is kan een succesvolle release binnen een kwartier lukken.

**Liferay:**

Bij Netwerk VSP wordt ook gebruikt gemaakt van Liferay. Liferay is een cms systeem dat zeer gevoelig is in falen ( komt vaak voor bij opstarten ). Liferay is bij Netwerk VSP redundant opgesteld met verschillende instanties van Liferay. Netwerk VSP gebruikt twee verschillende servers waar Liferay op geïnstalleerd staat. Elke server heeft vier instanties en kunnen allemaal individueel falen.

Wanneer een nieuwe release van software wordt uitgebracht met Liferay dien je eerst alle Liferay instanties te stoppen. Na het deployen van de nieuwe release, start je alle Liferay instanties weer op en moet je bij alle Liferay instanties individueel controleren of de release succesvol was. Zoniet dan moet je weer opnieuw beginnen.

**Testen bij Netwerk VSP:**

Wanneer er een release wordt uitgebracht zit Finalist en Netwerk VSP gezamenlijk in een chatroom. Zodra Finalist aangeeft dat de release deployed is, klikt Netwerk VSP door alle applicaties heen ( mannen van functioneel beheer die klikken belangrijke schermen aan om te kijken of het werkt ). Testen uitvoeren valt dus onder beheer van Netwerk VSP.

**Algemeen Netwerk VSP:**

Omdat Finalist niet verantwoordelijk is voor het beheer van de servers, hebben ze geen SLA afspraken met Netwerk VSP over up en down time.

Finalist heeft wel een testomgeving in beheer voor Netwerk VSP. ( gebruiken ze Jenkins voor incl geautomatiseerde integratietesten ).

De acceptatie omgeving is handmatig ingericht door de externe hostingspartij van Netwerk VSP. Er zitten helaas verschillen tussen de acceptatie omgeving en de productie omgeving van Netwerk VSP. Finalist kan niet proefdraaien op de acceptatie omgeving voordat ze naar productie gaan. Het blijft dus altijd een verrassing of het releasen van software goed gaat op productie omgeving ook al is het goed gegaan op acceptatie omgeving. Kortom:

Het is niet mogelijk voor Finalist om de infrastructuur van de acceptatie en productie omgeving gelijk te laten zijn omdat externe partij ze te weinig rechten geeft op de server

**back-ups:**

Bij het releasen van software maakt het Netwerk VSP team gebruik van “Forward patching”.

Dit wil zeggen dat er altijd een versie van de oude release op de acceptatie en productie servers blijft staan zodat deze direct terug geplaatst kan worden indien het releasen van de software totaal mis dreigt te lopen.

Voor database back-ups zorgt de externe hostingspartij voor snapshots van de Oracle database ( Voor elke productie release maakt hij een snapshot ). Indien Finalist een release verpest kan er altijd een snapshot van de database terug gezet worden en Finalist de oude applicatie om te rollbacken naar de oude situatie.

**Punten algemeen**

* Wijzigen van configuratie is geautomatiseerd door Finalist omdat het niet goed ging toen netwerk VSP zelf de releases uitvoerde aan de hand van gedetailleerde release notes.
* Alle projecten met bijbehorende configuration files, properties en bash scripts staan op de stash van Finalist.
* In de stash van Finalist worden logs bijgehouden per release welke gebruikt kunnen worden om te kijken welke veranderingen er doorgevoerd worden en door wie.
* expliciet uit omloop nemen van applicatie server uit load balancer moet je handmatig doen omdat hij normaal alleen kijkt “kan de applicatie server alweer http requesten afhandelen” maar dat kan hij al voordat de release helemaal voltooid is dus krijg je steeds verschillende foutmeldingen bij de gebruiker tot de release voltooid is.
* feature toggles kun je gebruiken om functionaliteiten die op dat moment geüpdatet worden even een andere pagina of melding te laten tonen wanneer het updaten nog niet klaar is zodat de gebruiker geïnformeerd wordt dat er deployment plaats vind. ( Je kunt features van je applicatie uitzetten die voor problemen kunnen gaan zorgen wanneer je binnen je release die functionaliteit zou gebruiken omdat er bijv nog database wijzigingen moeten worden doorgevoerd. )
* Rolling upgrade = load balancer en meerdere applicatie servers waarbij je de applicatie servers om de beurt down brengt en laat updaten. De load balancer stuurt de verzoeken naar de overgebleven applicatie servers die nog niet down zijn gebracht. Wanneer het gaat om enkele applicatie servers (hoger dan twee) kun je bijvoorbeeld gebruik maken van Sticky session zodat een client verbonden is aan de applicatie server die toegewezen is door de load balancer. ( je kunt ook session binden met een cookie ).
* Canary release = op 10% van je applicatie servers de nieuwe release deployen en deze alleen voor bijvoorbeeld de interne medewerkers inzetten. Vervolgens kijk je naar de statistieken om te kijken of alles goed blijft draaien. Daarna voer je het eventueel de release uit naar de andere 90%.

**Toekomst**

Op het moment is de applicatie server down bij het uitbrengen van een release. De toekomst zou het beter zijn als je alleen de applicatie(s) die je gaat updaten down kunt brengen en niet de gehele applicatie server.

Database scripts worden op het moment nog handmatig uitgevoerd. Het zou mooi zijn als dit in de toekomst geautomatiseerd kan worden.

Wat mooi zou zijn in de toekomst is dat je nadat de Jboss applicatie server weer succesvol gestart is, je automatische testen uitvoert of pagina’s van de nieuwe release bereikbaar zijn.

Handmatige stappen die foutgevoelig zijn en voor problemen zorgen is waar Finalist nu erg tegenaan loopt. Automatisering is iets wat Finalist in de toekomst wil toepassen.

Dingen worden niet goed vastgelegd bij versiebeheer. ( gaat al wat beter met Jira )

Wat Finalist zou kunnen doen is voor het deployen van een release te kijken of alle applicaties die worden gebruikt binnen hun applicatie beschikken over de juiste versie ( bijv Apache versie ) en de configuraties ( bijvoorbeeld poorten ) juist zijn geconfigureerd op de server voordat ze überhaupt beginnen aan een release.

#### 10.3.4.7 Systeembeheer ( André van der Elst )

**Server provisioning intern:**

Twee opties voor het inrichten van een nieuwe server:

- Lege vm met ssh key voor chef ( gebruiken ze chef om de applicaties te installeren )

- Template waar al Drupal of Hippo op staat ( de rest doen ze zelf handmatig of met chef ) kopiëren.

Proces:

- Je bepaald een template VM waarvan je een kopie wilt maken

- Je zoekt de template op in de tool Proxmox en maakt de keuze om een kopie te maken

- Je start de VM op nadat hij gekopieerd is en installeert de applicaties die gewenst zijn op de server.

- Een email bericht wordt verstuurd naar de systeembeheerder met een overzicht van geïnstalleerde applicaties en gegenereerd wachtwoord voor de nieuwe server.

**Server provisioning extern:**

Installeren van een kale VM met ssh key welke vervolgens handmatig of met chef ( hangt er vanaf of chef te gebruiken is door de keuze of we zelf een hosting voor de klant geregeld hebben of dat ze dat zelf hebben gedaan ) ingericht wordt.

**Algemeen:**

* Puppet wordt meer gebruikt voor configuratie management
* Chef wordt meer gebruikt voor software distributie
* Automatische updates van applicaties staan standaard uit op een nieuwe server. Wanneer een applicatie namelijk automatisch update, kan het voorkomen dat de geschreven applicatie van Finalist niet meer goed functioneert.
* Proxmox is de tool voor de management van alle VM's binnen Finalist.

### 10.3.5 Onderzoeksrapport

**Onderzoeksrapport**

Onderzoek software distributie

Welke software distributie tool sluit het beste aan op de software distributie binnen het Webapps team?



Auteur: Ruben Bouterse Eerste docent: H.P.Weenink

Opleiding: Informatica Tweede docent: G.M.Tuk

Bedrijf: Finalist Begeleider: R.Schellhorn

Opdrachtgever: M. van Breukelen

Datum: 04-08-2014

Versie: 1.0

Voorwoord

Dit document dient als onderzoekrapport voor het uitgevoerde onderzoek software distributie in opdracht van Finalist. De opdracht is uitgevoerd door Ruben Bouterse, student aan De Haagse Hogeschool in Den-Haag ter afsluiting van de opleiding Informatica.

Binnen het onderzoeksrapport staat beschreven welke informatie, resultaten en conclusies ik heb kunnen verzamelen betreffende het onderzoeksonderwerp.

De doelgroep van het onderzoeksrapport is ten eerste Finalist die de start naar dit onderzoek heeft verzonnen. Ten tweede is het onderzoeksrapport voor een iedere die geïnteresseerd is in het onderzoek software distributie. Van de lezer wordt verwacht dat deze beschikt over een kleine hoeveelheid kennis van Informatica.

Dankwoord

Ik wil graag mijn dank betuigen aan een aantal personen die mij geholpen hebben met de uitvoering van dit onderzoek naar software distributie. Om te beginnen wil ik alle onderstaande personen bedanken voor het begrip en de omgang omtrent de ziekte van pfeiffer welke tijdens mijn afstudeerperiode geconstateerd werd.

Rob Schellhorn, software ontwikkelaar bij Finalist voor zijn begeleidende rol en het geven van feedback op mijn werk. Arne Timmerman, software ontwikkelaar bij Finalist voor zijn ideeën en gedachtes omtrent het onderzoeksonderwerp en het aanbieden van een afstudeerplaats. Iris Mouthaan, HR Manager bij Finalist voor het mogelijk maken van het werken van halve dagen en thuis werken.

Van de Haagse Hogeschool wil ik mijn afstudeerbegeleiders Helene Weenink en Gerard Tuk bedanken voor hun begeleiding en het geven van feedback op mijn werk tijdens mijn afstudeerperiode.

Ook wil ik mijn vriendin Lianne Grijsen bedanken voor de interesse die ze toonde in de voortgang van mijn afstudeeropdracht.

In het bijzonder wil ik Jeroen Meijboom, mede afstudeerstudent bij Finalist bedanken voor de gezelligheid maar zeker ook de steun die we elkaar konden geven tijdens de uitvoering van onze opdrachten.

Tot slot wil ik alle werknemers binnen Finalist bedanken die betrokken zijn geweest bij mijn afstudeeropdracht.

Inhoudsopgave

[Voorwoord](#h.q0gw7xiy6pl7)

[Dankwoord](#h.wsogzyz0456k)

[Inhoudsopgave](#h.hj86zywz1f8e)

[1. Inleiding](#h.jxe1c936zjy)

[1.1 Kennisgebied](#h.jwa3wvasqd9b)

[1.2 Voorbereiding](#h.ybhwbd3y3orx)

[1.3 Afbakening](#h.fxuci3tpg5zk)

[1.4 Doelstelling](#h.sbudanu1xx7z)

[1.5 Hoofdvraag](#h.i48t5t6fj7rs)

[1.6 Deelvragen](#h.5ayy1sr3zmp8)

[1.7 Aanpak](#h.e1x7cc3pe97i)

[2. Beantwoording deelvragen](#h.rd603ebfeiqq)

[2.1 Deelvraag 1: Hoe voert het Webapps team software distributie uit?](#h.o41h8heh4ypf)

[2.1.1 Infrastructuur Finalist](#h.f4ehydub96s4)

[2.1.1.1 Teams Finalist](#h.j9k31h3kalqa)

[2.1.1.2 Infrastructuur applicaties Finalist](#h.uechmn412stn)

2.1.2 Gedetailleerde processen binnen software distributie van het Webapps team

[2.1.2.1 Inrichten van de distributie client](#h.d5wzywaw09cn)

[2.1.2.2 Server provisioning proces](#h.yf9ird61764p)

[2.1.2.3 Software deployment proces](#h.nkyxi93ykn46)

[2.1.2.4 Overzicht processen met stappen](#h.wcnescmlrah6)

[2.1.2.5 Volgorde processen](#h.dl4dsmxmcnk5)

[2.2 Deelvraag 2: Welke knelpunten treden op binnen de software distributie van het Webapps team en hoe zijn deze op te lossen en hoe zijn deze op te lossen?](#h.kwtjvcpyibgt)

[2.2.1 Toelichting knelpunten met technische beperkingen, gevolgen Finalist en klant en prioriteit](#h.abcgtz941l14)

[2.2.2 Ideale toekomstbeeld Webapps team](#h.4vy3cdltsg5)

[2.2.2.1 Prepare deployment](#h.ry4xvi2a7ao9)

[2.2.2.2 Geen foutmelding bij het noodzakelijk updaten van provisioning dependencies](#h.tnfci1q8ptf3)

[2.2.2.3 Server provisioning en software deployment processen realiseerbaar binnen één software distributie tool](#h.wjcxwpz2nbe1)

[2.2.3 Requirements voor vervangende software distributie tool](#h.v60tnfct6o0q)

[2.3 Deelvraag 3: Welke software distributie tools zijn geschikt voor het opstellen van de longlist en welke tool is het meest interessant om te testen in praktijk?](#h.p03s1yakn1ly)

[2.3.1 Criteria voor het opstellen van een longlist](#h.lkn1f0qh40bm)

[2.3.2 De longlist](#h.frcgrfbfl6og)

[2.3.3 Informatie per tool](#h.qmk7dbjazphm)

[2.3.3.1 Ansible](#h.8piqtyqzslyz)

[2.3.3.2 Mina](#h.t08k2sd6kd0)

[2.3.3.3. Jenkins](#h.rkiahnt2hmci)

[2.3.4 Keuze tool uit longlist](#h.ho60khjtmwz)

[2.4 Deelvraag 4: Welke resultaten leveren de beste tool uit de longlist op t.o.v. Chef en Capistrano in praktijk?](#h.7cpxgwwbxrgv)

[2.4.1 Architectuur Ansible](#h.hn2wt394ptd4)

[2.4.2 Architectuur test project (Fairwear)](#h.3qywulck4z6f)

[2.4.3 Testcases](#h.fes2q5m21637)

[2.4.3.1 Scope](#h.sn4e2i96ti55)

[2.4.3.2 Testcases server provisioning proces](#h.rx9oe2k4w4lg)

[2.4.3.3 Testcases prepare deployment subproces](#h.5fhzoxnl757y)

[2.4.3.4 Testcases publish deployment subproces](#h.ioszpz63kl1g)

[2.4.4 Resultaten testcases](#h.83jjmj1nozrq)

[3. Evaluatie en bevindingen](#h.31854ofzf0nd)

[3.1 Samenvatting onderzoek](#h.rmi9ig2hs5y2)

[3.1.1 De rode draad](#h.mqjmrpiz1q59)

[3.2 Conclusie](#h.um83hpb30c7s)

[3.2.1 Conclusie deelvraag 1](#h.8qb4fhf04lt7)

[3.2.2 Conclusie deelvraag 2](#h.t7f4fgudq4ln)

[3.2.3 Conclusie deelvraag 3](#h.4q1vzuxjgv2)

[3.2.4 Conclusie deelvraag 4](#h.hf0k0o84nak6)

[3.2.5 Eindconclusie](#h.a454gxe196c)

[3.3 Vervolgonderzoek](#h.lsnh0bkhov41)

[4. Bijlage](#h.eow5a8a9hd3a)

[4.1 Vooronderzoek](#h.yoz8zssh816q)

[4.2 Onderzoeksplan](#h.rh67pxhor5in)

[4.3 Requirementsrapport](#h.n1czb2c2xaz0)

[4.4 Testrapport](#h.a1bbrbcjkleb)

[4.5 Afbakening](#h.qye6hs7iqiyf)

[5. Literatuurlijst](#h.wirxqh14cr5f)

[6. Glossary](#h.uzcw1d261yu7)

1. Inleiding

1.1 Kennisgebied

Om kennis te verschaffen over mijn onderzoeksonderwerp ben ik gestart met een vooronderzoek naar wat software distributie is binnen Finalist. Om inzicht te kunnen krijgen in de processen binnen software distributie, ben ik gestart met het interviewen van zeven stakeholders. Iedere stakeholders was werkzaam in een ander “team”. Alle teams ontwikkelen software voor de aangestelde projecten binnen het team.

Omdat ik binnen dit onderzoek testen in de praktijk wil uitvoeren, dien ik kennis op te doen voor de volgende onderwerpen:

* Linux Os
* Virtualbox
* Werken met SSH keys
* Ruby on rails
* Beste software distributie tool uit longlist ( Ansible )
* Knife/Chef
* Capistrano
* Capistrano DSL(1)

1.2 Voorbereiding

Ter voorbereiding van dit onderzoek ben ik gestart met het uitvoeren van een vooronderzoek. Het vooronderzoek heeft mij inzicht gegeven over de basis elementen waaruit de processen van software distributie bestaan. De gegevens voor dit vooronderzoek zijn verzameld door half gestructureerde interviews af te nemen met zeven werknemers binnen Finalist.

Het vooronderzoek is als bijlage te vinden in hoofdstuk 4.1.

Naast het uitvoeren van half gestructureerde interviews en het uitvoeren van een vooronderzoek, heb ik een onderzoeksplan opgesteld dat als "plan van aanpak" kan worden gezien. Zo wordt er beschreven welke onderzoeksmethodes er gebruikt worden om informatie te verzamelen, welke hoofd- en deelvragen ik wil beantwoorden en in welke volgorde ik de deelvragen wil beantwoorden.

## 

1.3 Afbakening

Ik ben ervan bewust dat in de loop van het onderzoek verschillende onderwerpen voorbij zullen komen die interessant zijn om op te nemen binnen mijn onderzoek. Helaas is het voor mij niet mogelijk om al deze onderwerpen binnen mijn onderzoek op te nemen in verband met tijd. De onderwerpen zijn interessant om te overdenken bij een eventueel vervolgonderzoek.

Binnen het onderzoek worden de afbakeningen aangeduid met een nummer ( bijv AF1 ). Deze verwijzen naar de bijlage in hoofdstuk 4.5 waar de afbakeningen beschreven staan.

1.4 Doelstelling

Het doel van dit onderzoek is antwoord krijgen op de hoofdvraag “Welke software distributie tool sluit het beste aan op de software distributie binnen het Webapps team?”. Met het antwoord op deze vraag kan ik een verbeteradvies opstellen voor de software distributie binnen het Webapps team van Finalist.

## 

1.5 Hoofdvraag

De hoofdvraag van mijn onderzoek luidt als volgt:

* Welke software distributie tool sluit het beste aan op de software distributie binnen het Webapps team?

1.6 Deelvragen

Om de hoofdvraag te kunnen beantwoorden stel ik de volgende deelvragen:

* *Hoe voert het Webapps team software distributie uit?*

Binnen deze deelvraag wil ik antwoord krijgen over de huidige uitvoering van software distributie van het Webapps team. Door de processen binnen de software distributie in detail uit te schrijven, kan er informatie over de knelpunten worden vast gesteld. De knelpunten vormen het huidige probleem van het Webapps team.

* *Welke knelpunten treden op binnen de software distributie van het Webapps team en hoe zijn deze op te lossen en hoe zijn deze op te lossen?*

Binnen deze deelvraag wil ik knelpunten toelichten die zich afspelen binnen de software distributie van het Webapps team. Vervolgens zal een beschrijving van een ideaal toekomstbeeld mogelijkheden moeten bieden om oplossingen te vinden voor de opgestelde knelpunten. Wanneer de oude en ideale toekomst situatie beschreven zijn, worden er requirements opgesteld welke verwerkt worden in een requirementsrapport.

* *Welke software distributie tools zijn geschikt voor het opstellen van de longlist en welke tool is het meest interessant om te testen in praktijk?*

Binnen deze deelvraag wil ik een aantal software distributie tools beschrijven die mogelijk de oplossing kunnen bieden voor de knelpunten uit de vorige deelvraag. Aan de hand van requirements wil ik criteria opstellen waaraan de tools moeten voldoen. Door een literatuuronderzoek te starten, wil ik informatie verzamelen over beschikbare tools en de opgestelde criteria om te bewijzen of de tools wel of niet in staat is hieraan te voldoen.

* *Welke resultaten leveren de beste tool uit de longlist op t.o.v. Chef en Capistrano in praktijk?*

Binnen deze deelvraag wil ik de meest geschikte software distributie tool uit de vorige deelvraag testen in praktijk. Door middel van verschillende scenario’s die betrekking hebben tot de geselecteerde knelpunten uit deelvraag 2 te verwerken in testcases, hoop ik op een duidelijke manier aan te kunnen tonen welke tool de beste mogelijkheden biedt als oplossing voor het huidige probleem van het Webapps team.

## 

1.7 Aanpak

Om antwoord te krijgen op de deelvragen, is het noodzakelijk om informatie te verzamelen. Voor het verzamelen van informatie maak ik gebruik van verschillende onderzoeksmethoden. Hieronder de deelvragen van mijn onderzoek met de bijbehorende onderzoeksmethoden:

* *Hoe voert het Webapps team software distributie uit?*
  + Half gestructureerde interviews
  + Sneeuwbal methode
* *Welke knelpunten treden op binnen de software distributie van het Webapps team en hoe zijn deze op te lossen en hoe zijn deze op te lossen?*
  + Half gestructureerde interviews
  + Sneeuwbal methode
* *Welke software distributie tools zijn geschikt voor het opstellen van de longlist en welke tool is het meest interessant om te testen in praktijk?*
  + Half gestructureerde interviews
  + Sneeuwbal methode
  + Literatuuronderzoek
* *Welke resultaten leveren de beste tool uit de longlist op t.o.v. Chef en Capistrano in praktijk?*
  + Half gestructureerde interviews
  + Sneeuwbal methode
  + Literatuuronderzoek

Over deze onderzoeksmethodes wil ik graag mijn motivatie uitleggen.

*Half gestructureerde Interview methode*

Ik heb bewust gekozen om gebruik te maken van de half gestructureerde interview methode voor het verzamelen van informatie. Deze keuze heb ik gemaakt omdat de methode de optie biedt om een open interview te houden waarbij zowel opgestelde vragen kunnen worden gesteld als eigen inbreng van de stakeholder een rol kunnen spelen.

*Sneeuwbal methode*

Ik heb bewust gekozen voor de sneeuwbal methode voor het verzamelen van stakeholders. Binnen de sneeuwbal methode spreek je één persoon of een kleine groep respondenten aan om aan meer stakeholders te komen. Mijn toepassing van de sneeuwbal methode is geweest naar mijn begeleider ( ook stakeholder ) die vervolgens namen gaf van mogelijke stakeholders voor mijn onderzoek. Deze keuze heb ik gemaakt omdat een stakeholder binnen het bedrijf weet wat zijn collega’s voor functie hebben en daarom beter kon beslissen voor mij welke personen mogelijk de beste stakeholders waren.

*Literatuuronderzoek*

Ik heb er bewust voor gekozen om gebruik te maken van een literatuuronderzoek. Door gebruik te maken van het literatuuronderzoek wil ik gegevens vinden over software distributie tools die momenteel gebruikt worden. Deze keuze heb ik gemaakt omdat het lastig is deze informatie te winnen binnen het bedrijf Finalist en een literatuuronderzoek een breed oppervalk biedt om onderzoek in te verrichten zoals internet en boeken.

Voor meer inhoudelijke informatie over de methodes verwijs ik naar bron [Nel Verhoeven].

2. Beantwoording deelvragen

Binnen dit hoofdstuk beschrijf ik mijn bevindingen over de deelvragen die behoren tot de hoofdvraag “Welke software distributie tool sluit het beste aan op de software distributie binnen het Webapps team?”. De resultaten van de deelvragen vormen samen het antwoord op de hoofdvraag. De resultaten worden gebruikt voor het opstellen van een adviesrapport.

## 

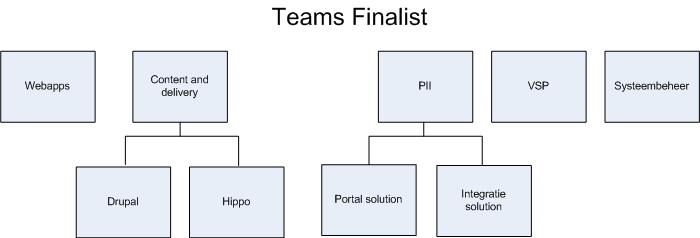
2.1 Deelvraag 1: Hoe voert het Webapps team software distributie uit?

Binnen deze deelvraag hoop ik antwoord te krijgen over hoe software distributie eruit ziet binnen het Webapps team. Om hier een duidelijk antwoord op te kunnen geven, wil ik eerst wat meer vertellen over de infrastructuur van Finalist om te begrijpen welke rol het Webapps team heeft binnen de organisatie. Vervolgens hoop ik antwoord te krijgen op de deelvraag door de processen binnen de software distributie van het Webapps team in detail uit te werken.

2.1.1 Infrastructuur Finalist

2.1.1.1 Teams Finalist

De software ontwikkeling binnen Finalist wordt verricht door zeven verschillende teams. Deze teams zijn ieder verantwoordelijke voor specifieke expertises die gebaseerd zijn op hun team. Om duidelijkheid te krijgen over de team verdeling en soorten projecten die ontwikkeld worden binnen de teams heb ik onderstaande afbeelding met toelichting toegevoegd:



*Webapps team:*

Het Webapps team bestaat momenteel uit drie werknemers waaronder mijn begeleider Rob Schellhorn. Binnenkort zal het Webapps team mogelijk uitbreiden tot zes werknemers. Binnen het Webapps team worden voornamelijk Ruby on rails projecten ontwikkeld. Deze projecten worden voornamelijk geschreven in terminal gebaseerde programma’s ( Bijvoorbeeld VIM ). Binnen mijn onderzoek wordt eerst vooronderzoek gedaan naar alle teams. Later vindt er een afbakening plaats waarbij het onderzoek zich richt tot alleen het Webapps team.

*Content and Delivery team:*

Het Content and Delivery team is onder te verdelen in twee subteams namelijk Hippo en Drupal. De namen van de subteams geven eigenlijk al aan wat voor projecten er ontwikkeld worden. Zo wordt er binnen het Hippo subteam Hippo CMS applicaties ontwikkeld en binnen het Drupal subteam Drupal CMS applicaties.

*Portals, Information & Identity (PII) team:*

Ook het Portals, information & Identity team is onder te verdelen in twee subteams namelijk Portal solution en Integratie solution. Binnen het portal solution team worden voornamelijk php maatwerk en Drupal projecten ontwikkeld.

Binnen het integratie solution team, worden ESB (Enterprise service bus) ontwikkeld. De ESB dienen als koppelstuk tussen twee of meerdere applicaties. Het integratie team maakt voornamelijk gebruik van Jboss fuse applicatie servers ( is een platform voor ESB ). De meeste projecten worden ontwikkeld in Java.

*VSP team:*

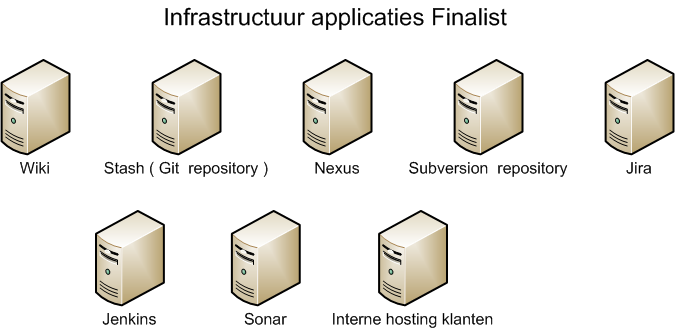
Het VSP team houdt zich voornamelijk bezig met het ontwikkelen van applicaties voor het bedrijf Netwerk VSP. Netwerk VSP is verantwoordelijk voor het verspreiden van folderpakketten naar alle huizen in Nederland. Momenteel vind er een onderzoek plaats naar optimalisatie van het produceerproces van de samengestelde folderpakketten. De meeste projecten worden ontwikkeld in Java.

*Systeembeheer team:*

De systeembeheerders houden zich voornamelijk bezig om alle werknemers binnen Finalist te kunnen laten werken. Zo zijn ze verantwoordelijk om de werknemers te voorzien van ontwikkelomgevingen en testomgevingen. Ook verzorgen de systeembeheerders de server provisioning van nieuwe servers voor zowel interne hosting als externe hosting. Kortom, de systeembeheerders verzorgen de OTAP omgevingen.

2.1.1.2 Infrastructuur applicaties Finalist

Binnen Finalist zijn een aantal servers met beschikbare applicaties opgesteld welke gebruikt kunnen worden door alle medewerkers. Binnen de interviews met de teams ben ik verschillende applicaties tegen gekomen. De volgende afbeelding bevat alle applicaties die ik verzameld heb:



*Wiki:*

De wiki applicatie wordt gebruikt voor het registreren van allerlei gegevens. Zo kun je de benodigde gegevens voor alle OTAP servers vinden maar ook procedures, roadmaps, documentaties en systeem installatie gegevens. Kortom alle benodigde gegevens voor een project kun je hier terug vinden.

*Stash git repository:*

Binnen de stash git repository staat de laatste broncode van alle projecten opgeslagen. Binnen stash kun je terug lezen welke commits er gedaan zijn en door wie ze uitgevoerd zijn. Binnen een commit kun je eenvoudig zien welke regels code er gewijzigd zijn.

*Nexus:*

Binnen Nexus staan alle projecten opgeslagen die gecompileerd zijn, succesvol getest zijn met integratie testen en voorzien zijn van de juiste dependencies. Nexus wordt gebruikt voor Java projecten.

*Subversion repository:*

De subversion repository werd voorheen gebruikt om de laatste broncode van projecten op te slaan. Tegenwoordig is de subversion repository vervangen door de stash git repository. De subversion repository wordt enkel nog gebruikt voor een paar projecten.

*Jira:*

Jira is een applicatie die gebruikt wordt voor het managen van projecten. Binnen Jira worden incidenten/issues aangemaakt welke gekoppeld kunnen worden aan een project. Een medewerker kan zo’n incident/issue oppakken en deze koppelen aan een release. Binnen Jira is het dus eenvoudig om te zien binnen welke release welke incidenten zijn opgelost en door wie ze zijn uitgevoerd. Jira wordt voornamelijk gebruikt door de projectmanagers.

*Jenkins:*

Jenkins is een “build” tool welke gebruikt wordt om broncode uit de stash git repository te builden en te plaatsen in het Nexus systeem. Binnen Jenkins is het mogelijk om een Jenkins taak te schrijven waarin een stapsgewijs script geschreven staat. De Jenkins taken die worden geschreven door de medewerkers van Finalist voeren vaak meer uit dan alleen het builden van de applicatie. Zo voeren de medewerkers ook geautomatiseerde integratietesten uit, halen eventuele dependencies op en soms zelfs voor het uploaden van het project naar de server van de klant.

Binnen Jenkins is het ook mogelijk om “build triggers” in te stellen. Een build trigger is een tijdsduur die gehanteerd wordt door Jenkins om de taak geautomatiseerd uit te voeren. Aan een build trigger kun je voorwaarden instellen waaraan moet worden voldaan voordat de taak daadwerkelijk uitgevoerd wordt. Wanneer de tijdsduur van de build trigger verlopen is en de build trigger voldoet aan de ingestelde voorwaarden, voert Jenkins de taak geautomatiseerd uit.

Jenkins wordt voornamelijk gebruikt bij projecten die “Continuous Integration(2)” hanteren.

*Sonar:*

Sonar is een applicatie die gebruikt wordt om “static code analysis(3)” uit te laten voeren. Binnen Sonar kun je eisen samenstellen waarop de applicaties moet controleren. Sonar wordt veel gebruikt om code die mogelijke problemen kunnen veroorzaken op te sporen en optimalisatie van de code te realiseren.

*Interne hosting klanten:*

Wanneer een klant ervoor kiest om de hosting van zijn applicatie door Finalist te laten verzorgen, zal deze gehost worden binnen de interne hosting van Finalist. Daarnaast wordt de interne hosting gebruikt om testservers te draaien.

2.1.2 Gedetailleerde processen binnen software distributie van het Webapps team

Nu we wat algemene kennis hebben over Finalist, kunnen we de focus leggen op afbakening AF1 welke beschreven staat binnen het hoofdstuk “Afbakening”. Deze afbakening houdt in dat er vanaf nu een afbakening geld tot software distributie binnen het Webapps team. Daarnaast geld vanaf nu ook afbakening AF2 welke inhoudt dat de term server provisioning een afgebakende betekenis heeft gekregen en vanaf nu onder de term “Software distributie” valt.

Binnen de software distributie van het Webapps team zijn twee processen te beschrijven namelijk:

* Server provisioning proces
* Software deployment proces

#### 

2.1.2.1 Inrichten van de distributie client

Voordat het software deployment en server provisioning proces vanaf de ontwikkelomgeving gerealiseerd kan worden binnen het Webapps team, dient de client van de ontwikkelaar te voorzien zijn van een aantal programma’s. De client uit de ontwikkelomgeving noemen we vanaf nu “de distributie client”.

De software die geïnstalleerd moet worden op de distributie client is als volgt:

* Het project dat gedistribueerd moet worden

Het project dat gedistribueerd moet worden beschikt over uitvoerbare scripts die nodig zijn om het server provisioning en software deployment proces te starten. De projecten beschikken over scripts voor alle gebruikte omgevingen van OTAP die binnen het project van toepassing zijn.

* Git

Het programma Git wordt gebruikt om de meest recente broncode van een project te kunnen downloaden uit de stash git repository.

* Ssh

Ssh moet worden geïnstalleerd om vanaf de distributie client verbinding te kunnen maken met de server van de klant. Via deze verbinding kunnen commando’s worden verzonden die vervolgens door de server van de klant worden uitgevoerd.

* RVM

RVM staat voor Ruby version manager en wordt op de distributie client gebruikt om per project de juist Ruby versie met juiste gem-set(4) te kunnen gebruiken.

* Bundle

Het programma Bundle moet worden geïnstalleerd om te controleren of de distributie client beschikt over alle gem dependencies die vereist zijn voor een project. Het verschilt namelijk per project welke gem dependencies er vereist zijn en welke versie van het programma gebruikt wordt. Bundle bevestigt of de distributie client beschikt over de dependency(5) en de juiste versie en installeert deze indien dit niet het geval is.

* Vagrant

Het programma Vagrant wordt gebruikt om test server aan te maken in Virtualbox op de distributie client. Deze testservers kun je gebruiken om te testen of het installeren van applicaties via Chef ( server provisioning ) of het distribueren van een project met Capistrano ( software deployment ) werkt.

#### 

2.1.2.2 Server provisioning proces

Nu we weten hoe onze distributie client ingericht moet worden, kunnen we beginnen met het server provisioning proces.

Omdat ik in een vooronderzoek de basis elementen uit het server provisioning proces heb opgesteld, wil ik de resultaten herhalen om te bevestigen dat de basis elementen aanwezig zijn binnen het proces van het Webapps team.

Het server provisioning proces uit het vooronderzoek bestaat uit drie activiteiten en zit er als volgt uit:



*Verzoek indienen:*

* Verzoek indienen bij systeembeheerder voor server.

*Installatie server:*

* Systeembeheerder kopieert template van de server met applicaties (interne host).
* Systeembeheerder voert besturingssysteem installatie uit (externe host).

*Installatie benodigde applicaties:*

* Systeembeheerder installeert applicaties welke opgegeven zijn binnen het ingediende verzoek (externe host).
* Medewerkers team installeren zelf applicaties via een ssh-verbinding naar de server.

De eerste twee activiteiten en stappen van het server provisioning proces zijn precies hetzelfde als het basis proces uit het vooronderzoek. De uitvoering van de stappen uit de eerste twee activiteiten zijn terug te lezen in het vooronderzoek.

De laatste stap “Medewerkers team installeren zelf applicaties via een ssh-verbinding naar de server” uit de laatste activiteit “installatie benodigde applicaties” is echter niet in detail beschreven in het vooronderzoek. Deze wil ik nu graag toelichten voor het Webapps team.

Het Webapps team maakt voor de server provisioning gebruik van het programma “Chef”. Binnen Chef zit een command-line(6) tool “Knife”, welke als interface kan worden gebruikt tussen de distributie client en de server[Knife]. Om de werking van Chef te verduidelijken, beschrijf ik de mappenarchitectuur die gebruikt wordt met beschrijving waarvoor ze dienen:



*Nodes:*

Binnen deze map kun je definiëren welke nodes er binnen een bedrijf ingericht zijn. Een goed voorbeeld zou bijvoorbeeld OTAP kunnen zijn. Voor iedere node is een bestand aangemaakt waarin een runlist(7) beschreven staat welke roles of recipes er uitgevoerd moeten worden.

*Roles:*

Binnen deze map kun je definiëren welke roles een server kan hebben. Een goed voorbeeld zou bijvoorbeeld een web server, applicatie server of database server kunnen zijn. Een role bestaat uit een runlist van één of meerdere recipes. Een recipe is een script met commando’s die op volgorde uitgevoerd worden.

*Cookbooks:*

Binnen de cookbooks map staan alle cookbooks opgeslagen die kunnen worden geïnstalleerd. Een cookbook kan gezien worden als een installatie van een applicatie. Een goed voorbeeld van een cookbook zou bijvoorbeeld Mysql kunnen zijn. Ook is het mogelijk om met een cookbook configuraties te wijzigen. Binnen een cookbook kun je meerdere recipes beschrijven. Voor het cookbook Mysql zouden bijvoorbeeld de recipes Mysql\_client.rb, Mysql\_server.rb en default.rb kunnen bestaan. Default.rb is altijd het standaard recept dat uitgevoerd wordt wanneer er niet specifiek aangegeven wordt welke recipe uit het cookbook geïnstalleerd moet worden. De cookbooks die in de cookbooks map staan zijn algemene installaties die je kunt downloaden van de Chef-community.

*Site-Cookbooks:*

Binnen deze map kun je dezelfde beschrijving hanteren als beschreven staat bij Cookbooks. Het verschil met deze map is, dat de cookbooks zelf geschreven zijn en niet afkomstig zijn van de Chef-community.

*Data-bags:*

Het kan voorkomen dat er binnen een recipe een aanroep wordt gedaan naar een specifieke gebruiker. Binnen de data-bags kun je deze gebruikers definiëren met daarbij het wachtwoord. De wachtwoorden zijn versleuteld in de data-bags verwerkt en zijn hierdoor veilig opgeslagen.

Om de server provisioning te starten vanaf de distributie client, dien je het volgende commando op te geven in een terminal:

**1. Knife solo bootstrap [ip adres server]**

De volgende stappen worden vervolgens uitgevoerd:

**2. Downloaden van Chef op de server**

Binnen deze stap wordt de installatie van Chef gedownload op de server.

**3. Installeren van Chef op de server**

Binnen deze stap wordt Chef geïnstalleerd op de server.

**4. Starten van Chef op de server**

Binnen deze stap wordt de zojuist geïnstalleerde Chef gestart op de server.

**5. Kopiëren van cookbooks van distributie client naar server**

Binnen deze stap worden de cookbooks van de distributie client naar de server gekopieerd.

**6. Controleer cookbook provisioning dependencies**

Binnen deze stap worden de provisioning dependencies die binnen de cookbooks beschreven staan gedownload, bijgewerkt en/of geïnstalleerd op de server.

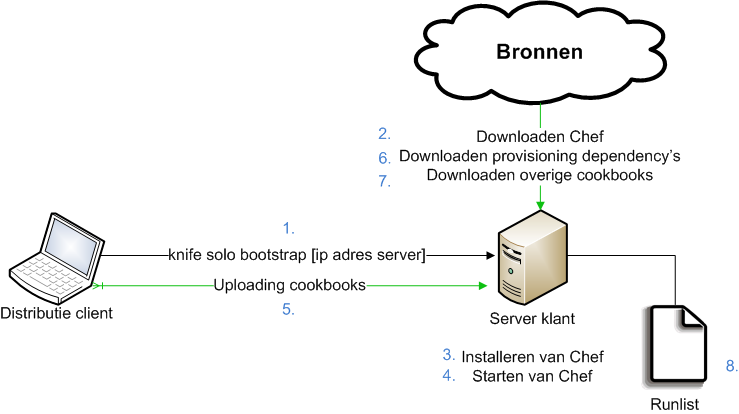
**7. Downloaden van overige cookbooks**

Binnen deze stap worden de overige cookbooks die niet gekopieerd zijn van de distributie client naar de server maar wel beschreven staan in de runlist gedownload van een externe bron.

**8. Uitvoeren van runlist**

Tot slot wordt de runlist uitgevoerd waarin alle cookbooks geïnstalleerd worden op de server die in de runlist beschreven staan.

Hieronder een afbeelding waarin de stappen van Chef weergegeven zijn in een overzichtelijk schema:

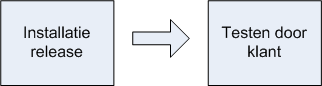


2.1.2.3 Software deployment proces

Na het server provisioning proces begint het software deployment proces.

Omdat ik in een vooronderzoek de basis elementen uit het software deployment proces heb opgesteld, wil ik de resultaten herhalen om te bevestigen dat de basis elementen aanwezig zijn binnen het proces van het Webapps team.

Het software deployment proces uit het vooronderzoek bestaat uit twee activiteiten en ziet er als volgt uit:



*Installatie release:*

* De nieuwe release wordt gedownload door de server van de klant uit de stash git repository van Finalist
* Database wijzigingen doorvoeren
* Dependencies updaten

*Testen door klant:*

* Uitvoeren van testen door de klant

Binnen het software deployment proces blijken de activiteiten precies hetzelfde te zijn als bij het basis proces uit het vooronderzoek. Echter blijken alleen een aantal extra stappen te missen. De activiteiten zien er als volgt uit:

*Activiteit installatie release:*

* De nieuwe release wordt gedownload door de server van de klant uit de stash git repository van Finalist
* Database wijzigingen doorvoeren
* Assets compileren
* Gem dependencies updaten
* Vastleggen van activiteiten in Logfiles

*Activiteit testen door klant:*

* Uitvoeren van testen door de klant

Note: de paarse tekst geeft een stap aan welke extra is toegevoegd binnen het Webapps team.

De uitvoering van de stappen binnen activiteit "testen door klant" zijn precies hetzelfde gebleven en zijn terug te lezen in het vooronderzoek. Ten opzichte van het basis proces zijn er enkel alleen in de activiteit "installatie release" extra stappen bijgekomen. De uitvoering van deze stappen zal ik nader toelichten.

De stappen binnen de activiteit "installatie release" worden door het Webapps team uitgevoerd door het programma "Capistrano (V3.0)". Capistrano is een programma die scripts uit kan voeren op één of meerdere servers. Het programma wordt voornamelijk gebruikt om web applicaties te distribueren [Capistrano].

Om de werking van Capistrano te verduidelijken, beschrijf ik de mappen en bestanden die gebruikt worden met beschrijving waarvoor ze dienen:



*Capfile:*

Binnen de Capfile kun je aangeven welke programma’s vereist zijn bij het uitvoeren van het software deployment proces. Zo wordt bijvoorbeeld eerst de DSL en standaard distributie taken voor Capistrano geladen. Vervolgens kun je eenvoudig met het commando “require” aangeven welke programma’s nog meer vereist zijn.

*Deploy.rb:*

Binnen het deploy.rb bestand worden een heleboel variabelen beschreven. Deze variabelen worden gebruikt wanneer Capistrano het software deployment proces start. Naast het uitschrijven van variabelen is het mogelijk om extra stappen toe te voegen aan het standaard software distributie proces van Capistrano. Het Deploy.rb bestand kan worden gezien als het script dat uiteindelijk uitgevoerd wordt bij de software deployment.

*Deploy:*

Binnen deze map kun je definiëren welke omgevingen er binnen een bedrijf ingericht zijn. Een goed voorbeeld zou bijvoorbeeld OTAP kunnen zijn. Voor iedere omgeving is een bestand aangemaakt waarin per role wordt aangegeven op welk ip deze te benaderen is. Binnen deze map bevinden zich ook de configuraties die worden gebruikt binnen het Deploy.rb bestand.

Het standaard software deployment proces van Capistrano bestaat uit verschillende stappen. De stappen worden “stages” genoemd. Capistrano maakt gebruik van de volgende stages:

**deploy:starting - start a deployment, make sure everything is ready  
deploy:started - started hook (for custom tasks)  
deploy:updating - update server(s) with a new release  
deploy:updated - updated hook  
deploy:publishing - publish the new release  
deploy:published - published hook  
deploy:finishing - finish the deployment, clean up everything  
deploy:finished - finished hook**

[Capistrano-Flow]

Het is mogelijk om voor of na een stage een eigen geschreven stap uit te voeren. Hierboven in het overzicht zijn deze stappen beschreven als “hook”. De eigen geschreven stappen kun je beschrijven in het deploy.rb bestand.

Omdat de standaard stages in het bovenstaande overzicht maar kort beschreven zijn, wil ik deze stages wat meer in detail uitwerken. Echter wil ik de stages wel afbakenen volgens afbakening AF3 waarbij de hook stages achterwegen worden gelaten.

**deploy:starting - start a deployment, make sure everything is ready**

* Controles start Capistrano

Controleren of Capistrano genoeg rechten heeft op de server om zijn script uit te kunnen voeren. Daarnaast worden er dingen bekeken als de bereikbaarheid van de Stash Git repository en de login gegevens hiervoor.

**deploy:updating - update server(s) with a new release**

* Aanmaken working directory met timestamp

Binnen deze stap wordt er een nieuwe map aangemaakt met als titelnaam de nummers van een timestamp. Binnen deze map zal de nieuwe release worden geplaatst.

* Nieuwste versie project van stash git repository ophalen

Binnen deze stap wordt de nieuwste versie van het project uit de stash git repository van Finalist gedownload naar de server.

* Bundle uitvoeren

Binnen deze stap worden de gem dependencies gecontroleerd op de server. Wanneer blijkt dat er een gem dependency mist, word deze gedownload en bijgewerkt.

* Assets compileren

Binnen deze stap vindt het compileren van het Ruby on Rails project plaatst. Het compileren van het Ruby on Rails project wordt uitgevoerd op de server.

* Database migraties uitvoeren

Binnen deze stap worden de wijzigingen aan de database uitgevoerd. Wanneer er geen wijzigingen in het database bestand zijn aangebracht, gaat het proces verder naar de volgende stap.

**deploy:publishing - publish the new release**

* Current verwijzen naar map van timestamp

Binnen deze stap wordt de symlink(8) verwezen naar de map met de nieuwe timestamp waarin de nieuwe release staat opgeslagen.

**deploy:finishing - finish the deployment, clean up everything**

* Opruimen tijdelijke mappen

Binnen deze stap worden alle tijdelijke mappen die binnen het software deployment proces worden aangemaakt verwijderd.

* Revisionslog bijwerken

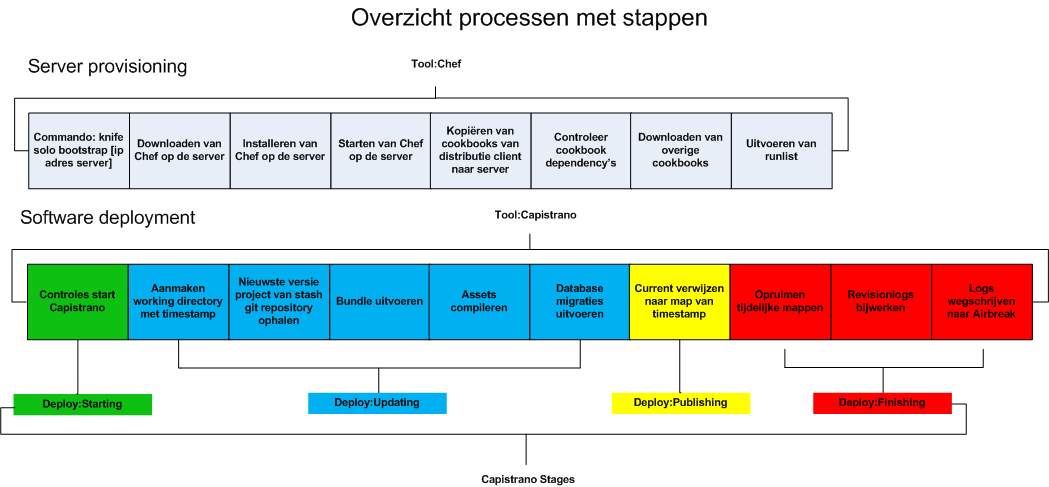
Binnen deze stap wordt binnen het revisions.log bestand welke aangemaakt is op de server binnen het server deployment proces beschreven welke release er is uitgebracht en door welke gebruiker deze is uitgevoerd.

* Logs wegschrijven naar Airbreak.

Binnen deze stap worden logs van de stappen met eventuele fouten verstuurd naar Airbreak. Airbreak is een tool voor error monitoring.

2.1.2.4 Overzicht processen met stappen

Binnen dit hoofdstuk wil ik een overzicht tonen van de twee processen met de inhoudelijke stappen die uitgevoerd worden door de tools.



2.1.2.5 Volgorde processen

De volgorde en toepassing van deze processen wil ik in dit hoofdstuk moduleren en toelichten. Hieronder een overzicht van de processen met volgorde van uitvoering en beschrijving hoe vaak de processen uitgevoerd worden:

## 

*1. Server provisioning*

Dit proces wordt als eerste uitgevoerd en zal eenmalig plaatsvinden. De reden waarom dit proces eenmalig uitgevoerd wordt komt omdat de server eenmalig ingericht moet worden met de juiste provisioning dependencies.

*2. Software deployment*

Dit proces wordt als tweede uitgevoerd en zal herhalend plaatsvinden. De reden waarom dit proces herhalend uitgevoerd wordt komt omdat er steeds nieuwe releases gedistribueerd worden op de server.

## 

2.2 Deelvraag 2: Welke knelpunten treden op binnen de software distributie van het Webapps team en hoe zijn deze op te lossen?

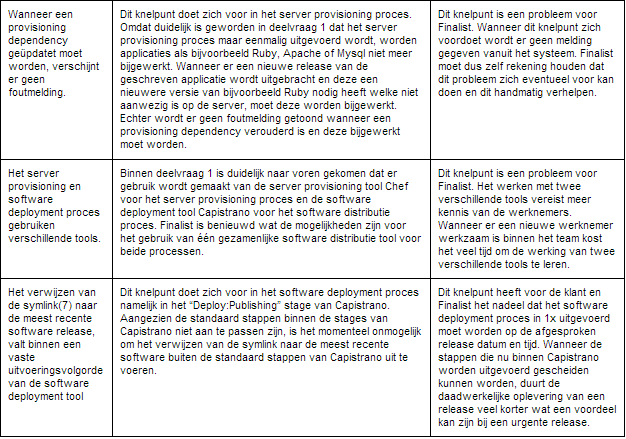
Binnen het onderzoeksplan heb ik bij het hoofdstuk “probleemstelling” een aantal knelpunten beschreven. Echter zijn deze knelpunten heel globaal geformuleerd omdat ik nog geen beschikking had over de kennis van deelvraag 1. Nu ik wel over deze kennis beschik kan ik een beter beeld geven van de opgestelde knelpunten uit het onderzoeksplan. Naast een toelichting wil ik de eventuele technische beperkingen bespreken.

Het onderzoeksplan is als bijlage te vinden in hoofdstuk 4.2.

2.2.1 Toelichting knelpunten met technische beperkingen, gevolgen Finalist en klant en prioriteit

Binnen dit hoofdstuk licht ik de knelpunten uit het onderzoeksplan toe en vul deze aan met de eventuele technische beperkingen. Daarnaast beschrijf ik welke gevolgen de knelpunten hebben voor Finalist en de klant.





Nu ik een beter beeld heb kunnen krijgen over de knelpunten die spelen binnen het Webapps team, wil ik de knelpunten voorzien van een uniek ID en prioriteit.

### 

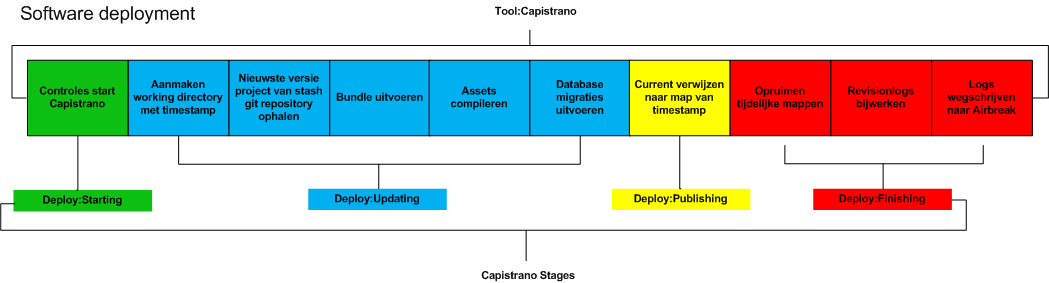
2.2.2 Ideale toekomstbeeld Webapps team

Binnen dit hoofdstuk wil ik het ideale toekomstbeeld van het Webapps team uitwerken. Het ideale toekomstbeeld biedt mogelijk oplossingen voor de knelpunten die spelen en vormen requirements voor de toekomstige software distributie tool.

Binnen een interview met het Webapps team is mij duidelijk gemaakt dat het Webapps team nagedacht heeft voor een oplossing voor de knelpunten. Zo werd mij verteld over een “prepare deployment concept”. Graag wil ik het prepare deployment concept beschrijven.

2.2.2.1 Prepare deployment

Eerder in deelvraag 1 hebben we een overzicht kunnen zien van de twee processen met de stappen die uitgevoerd worden door de tools Chef en Capistrano. Het prepare deployment concept heeft betrekking op de stappen van Capistrano. De huidige stappen van Capistrano zijn als volgt:



Binnen het prepare deployment concept worden de stappen opgesplitst over twee subprocessen namelijk:

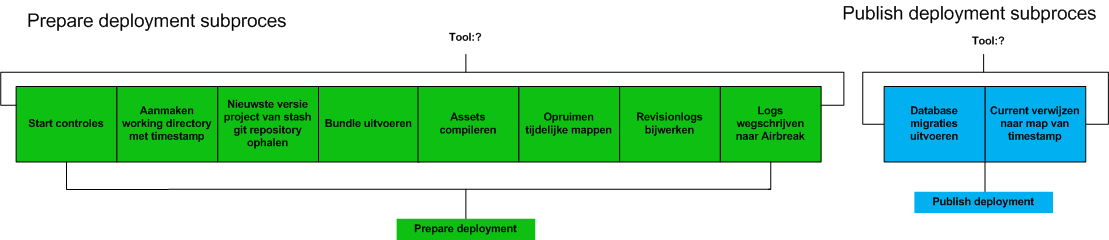
Prepare deployment subproces stappen:

* Start controles
* Aanmaken working directory met timestamp
* Nieuwste versie project van stash git repository ophalen
* Bundle uitvoeren
* Assets compileren
* Opruimen tijdelijke mappen
* Revisionlogs bijwerken
* Logs wegschrijven naar Airbreak

Publish deployment subproces stappen:

* Database migraties uitvoeren
* Current verwijzen naar map van timestamp

Een overzicht van de stappen binnen het prepare deployment concept ziet er als volgt uit:



Nu is natuurlijk de vraag wat deze verdeling van stappen te bieden heeft als oplossing voor de knelpunten van het Webapps team. Omdat de stappen binnen de Capistrano tool een vaste uitvoeringsvolgorde hanteren, is het niet mogelijk om op een eerder moment alvast te beginnen aan de software deployment. Binnen het prepare deployment concept is dit wel mogelijk.

Wanneer er een release van de software gepland staat, kan het prepare deployment subproces ruim voor de geplande datum worden uitgevoerd. Binnen het prepare distributie subproces kunnen veel complicaties zich voordoen. Stappen waar vaak complicaties in ontstaan zijn:

* Bundle uitvoeren
* Assets compileren

De medewerkers van Finalist hebben genoeg tijd om mogelijke complicaties op te lossen omdat het prepare deployment subproces ruim voor de geplande datum uitgevoerd wordt.

Wanneer het prepare deployment subproces succesvol is uitgevoerd, kan het publish deployment subproces worden uitgevoerd bij de daadwerkelijke oplevering van de release. Binnen het publish deployment subproces is nog enkel één stap welke gevoelig is voor complicaties namelijk:

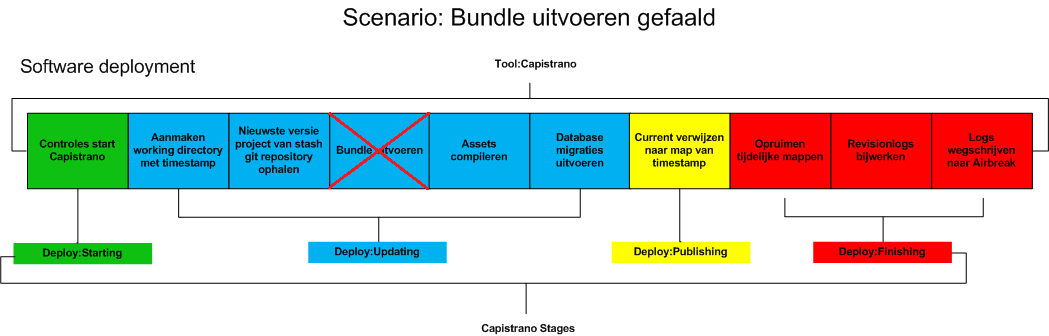
* Database migraties uitvoeren

Om het prepare deployment concept nog beter toe te lichten, toon ik nu een scenario voor zowel het software deployment proces met Capistrano als het gebruik van het prepare deployment concept:

*Scenario:*

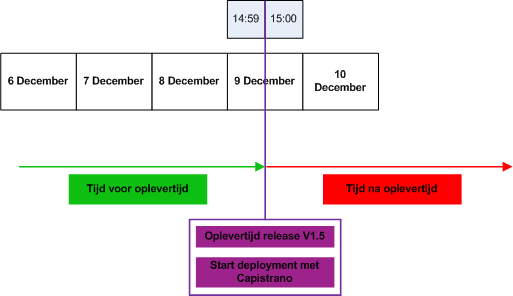
Dit scenario bevat fictieve data. Het is vandaag 1 december. Voor 9 December staat er gepland dat release V1.5 van de software gedistribueerd zal worden om 15:00.

Wanneer de release wordt gedistribueerd met Capistrano, zal de medewerker op 9 december om 15:00 het software deployment proces starten. Als alles goed verloopt, zal de distributie een aantal minuten duren. Binnen dit scenario gaat er iets mis binnen de stap “bundle uitvoeren”.

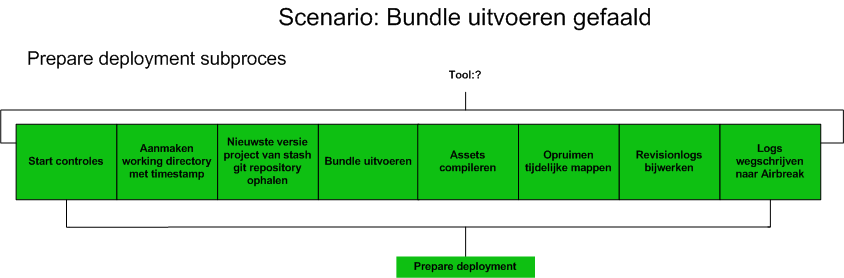


Het software deployment proces wordt geannuleerd en de server wordt weer in de oude staat gebracht zoals deze was voor het software deployment proces begon. Wat zijn nu de gevolgen voor de klant en Finalist?

Omdat Finalist niet de mogelijkheid had om eerder te beginnen aan het software deployment proces, was het niet mogelijk om de release V1.5 op tijd op te leveren op de geplande tijd. Dit heeft tot gevolg dat Finalist na de geplande oplevertijd het probleem moet oplossen.

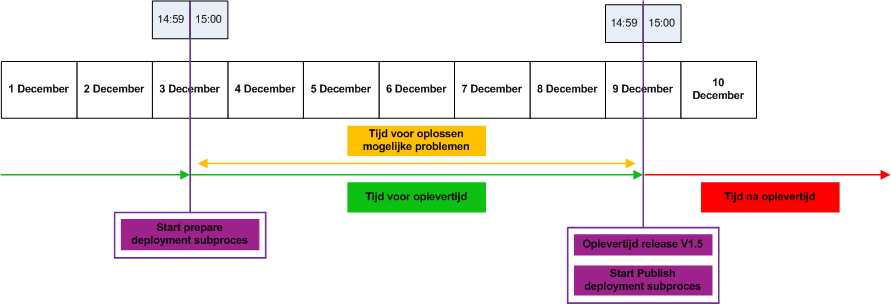


Wanneer de release wordt gedistribueerd met het prepare deployment concept, zal de medewerker op 3 december om 15:00 het prepare deployment proces starten. Als alles goed verloopt, zal de distributie een aantal minuten duren. Binnen dit scenario gaat er iets mis binnen de stap “bundle uitvoeren”.



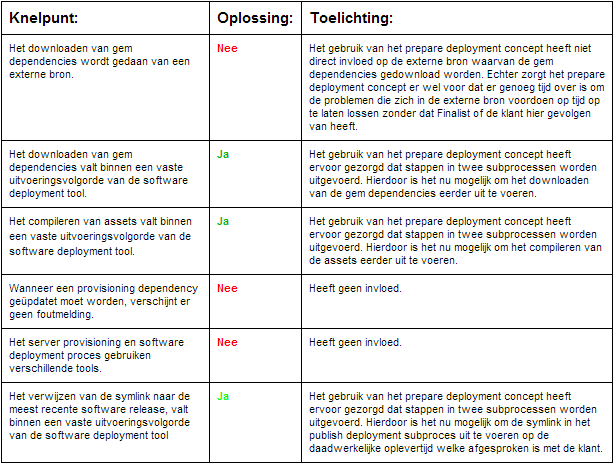
Het prepare deployment subproces wordt geannuleerd en de server wordt weer in de oude staat gebracht zoals deze was voor het prepare deployment subproces begon. Wat zijn nu de gevolgen voor de klant en Finalist?

Omdat Finalist de mogelijkheid had om een week eerder te beginnen aan het prepare deployment subproces, is er de mogelijkheid om binnen een week het probleem op te lossen en de release alsnog op tijd op te leveren. Zodra het prepare deployment subproces met succes is afgerond, zal het Publish deployment subproces op 9 december om 15:00 worden uitgevoerd.



Binnen dit scenario is duidelijk te zien dat het prepare deployment concept de mogelijkheid biedt om complicaties die zich voordoen op tijd op te kunnen lossen.

Nu ik het prepare deployment concept heb besproken en toegelicht met een scenario, wil ik kijken voor welke knelpunten dit concept een oplossing biedt.



2.2.2.2 Geen foutmelding bij het noodzakelijk updaten van provisioning dependencies

Binnen het interview met het Webapps team hebben we gesproken over het knelpunt “Wanneer een provisioning dependency geüpdatet moet worden, verschijnt er geen foutmelding”. Er werd mij verteld dat het updaten van de provisioning dependencies niet heel vaak voorkomt. De prioriteit die toegewezen is aan dit knelpunt is laag.

We zijn tot de conclusie gekomen dat er twee mogelijke oplossingen zijn waardoor het Webapps team dit knelpunt als verholpen zou kunnen beschouwen.

*Optie1* ( *server provisioning proces starten voor iedere uitvoering van het software deployment proces* )

Wanneer deze optie van toepassing is, zal voor het uitbrengen van een release elke keer het server provisioning proces worden gestart. Provisioning dependencies die geüpdatet moeten worden, zullen worden geïnstalleerd. Het opleveren van een release zal vors toenemen in tijdsduur omdat bij iedere release het server provisioning proces gestart wordt. Echter zal het server provisioning proces niet zo lang duren als de eerste keer dat server provisioning is uitgevoerd.

*Optie2 ( tonen van een melding wanneer een provisioning dependency bijgewerkt moet worden)*

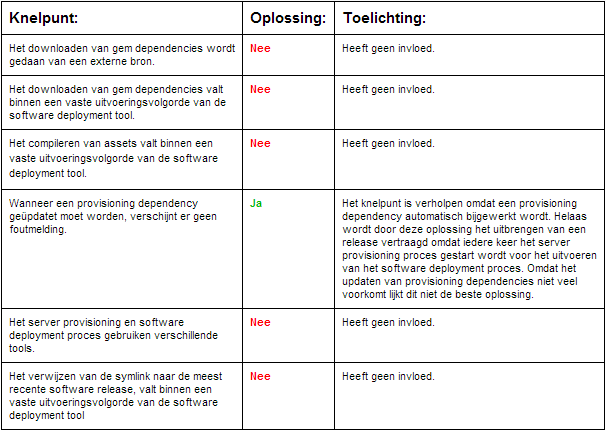
Wanneer deze optie van toepassing is zal dit niet toenemen in de tijdsduur van de oplevering van een release. Deze optie zorgt ervoor dat een werknemer direct weet dat er een update vereist is en om welke provisioning dependency het gaat. Het updaten van de provisioning dependency zal niet veel tijd in beslag nemen.

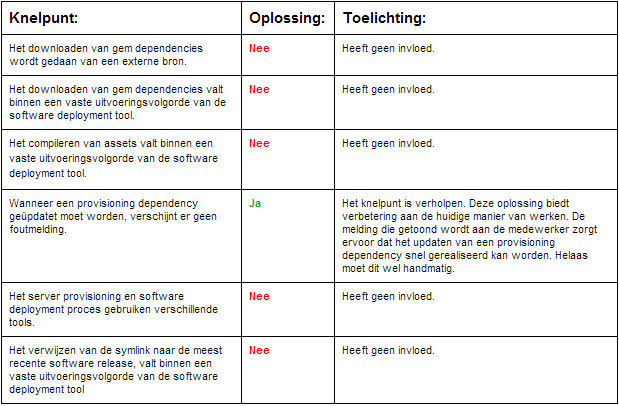
Wanneer we deze twee opties tegenover elkaar overwegen, gaat de voorkeur uit naar optie2. Deze keuze is gemaakt op basis van de volgende gedachten:

* Optie2 zorgt ervoor dat de tijdsduur van het opleveren van een release niet toeneemt.
* Het updaten van provisioning dependencies komt niet vaak voor. Hierdoor is het steeds opnieuw uitvoeren van server provisioning binnen optie1 vaak voor niets.
* Optie2 geeft informatie aan de werknemers waardoor het oplossen van het probleem binnen zeer korte tijd gerealiseerd kan worden ( misschien zelfs korter dan het opnieuw uitvoeren van server provisioning uit optie1 ).

Na het toelichten van de opties, wil ik kijken voor welke knelpunten dit als oplossing kan dienen.

*Optie1:*



*Optie2:*   
  


2.2.2.3 Server provisioning en software deployment processen realiseerbaar binnen één software distributie tool

Binnen het interview met het Webapps team hebben we gesproken over het knelpunt “Het server provisioning en software deployment proces gebruiken verschillende tools”. We zijn tot de conclusie gekomen dat er twee mogelijke opties zijn om dit knelpunt op te lossen namelijk:

*Optie1 ( tool zoeken die zowel server provisioning als software deployment kan realiseren ):*

Binnen deze optie zal er één tool worden gebruikt voor zowel het server provisioning als het software deployment proces. Dit brengt als voordeel met zich mee dat een medewerker nog maar één tool hoeft te begrijpen om twee processen uit te kunnen voeren.

*Optie2 ( tools server provisioning en software distributie gescheiden houden ):*

Binnen deze optie zullen beide processen gebruik maken van een eigen tool. In de huidige situatie is dit ook gerealiseerd met Chef als tool voor server provisioning en Capistrano als tool voor software deployment.

Nu is de vraag welke optie het meest geschikt lijkt om te hanteren. Echter durf ik hier vooraf geen uitspraak over te doen. Beide opties roepen vragen op welke nog niet beantwoord kunnen worden. U kunt denken aan vragen als:

* Hoe is de samenwerking tussen twee gescheiden tools? ( misschien wel zelfde uitgever? )
* Hoe is de ondersteuning van een tool?
* Hoe snel werkt een tool?

Kortom, zolang we nog geen gegevens hebben over welke tools het gaat kan er geen keuze worden gemaakt welke optie beter is.

2.2.3 Requirements voor vervangende software distributie tool

Naast de opgestelde knelpunten in het vorige hoofdstuk, is het belangrijk om requirements op te stellen waaraan de vervangende software distributie tool moet voldoen. De knelpunten worden omgezet tot requirements en worden aan elkaar gekoppeld in de tabel.

De overige requirements die niet aan de hand van knelpunten zijn opgesteld, zijn gebaseerd op de resultaten van de eerste deelvraag, het ideale toekomstbeeld van het Webapps team en het bijbehorende interview.

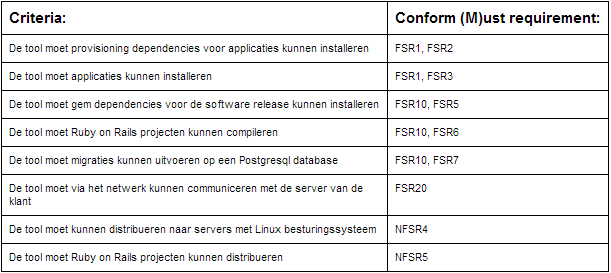
De opgestelde requirements zijn te vinden in de bijlage in hoofdstuk 4.3.

2.3 Deelvraag 3: Welke software distributie tools zijn geschikt voor het opstellen van de longlist en welke tool is het meest interessant om te testen in praktijk?

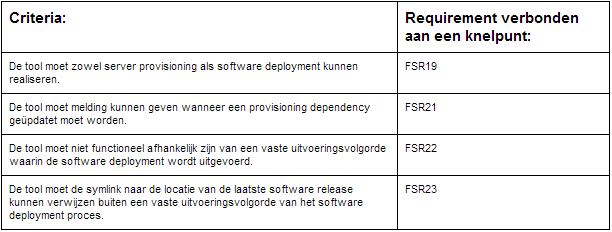
Nu de huidige situatie en het ideale toekomstbeeld beschreven zijn en de requirements duidelijk zijn geworden, wil ik gaan kijken naar geschikte tools die mogelijk de huidige tools Chef en Capistrano kunnen vervangen en knelpunten kunnen oplossen. Conform de requirements die opgesteld zijn in deelvraag 2, wil ik criteria opstellen waaraan een geschikte tool moet voldoen.

2.3.1 Criteria voor het opstellen van een longlist

één van de belangrijkste criteria waaraan de tool moet voldoen zijn de functional en non functional software requirements met de (M)ust prioriteit. De criteria ziet er als volgt uit:



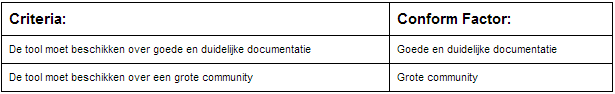
Naast de criteria die opgesteld is aan de hand van requirements met de (M)ust prioriteit, vormen de requirements die verbonden zijn aan een knelpunt ook goede criteria. De criteria ziet er als volgt uit:



Tot slot zijn er een aantal factoren die in de toekomst mogelijk belangrijk kunnen zijn bij het gebruik van de tool. Deze factoren zijn:

* Goede en duidelijke documentatie ( beschikbaar op originele site )
* Grote community ( vanaf 500.000 hits bij Google )

De criteria zien er als volgt uit:



De volledige lijst aan criteria ziet er nu als volgt uit:

### 

2.3.2 De longlist

Nu we de criteria opgesteld hebben waaraan een tool voor de longlist zich aan moet voldoen, is het tijd om te kijken welke tools er geschikt zijn. De tools die interessant genoeg lijken om meer informatie over te zoeken zijn:

* Ansible
* Mina
* Jenkins

Graag motiveer ik waarom ik deze tools heb opgesteld in mijn longlist.

*Ansible:*

Ansible is momenteel een hype onder de automatisering tools. Ook binnen Finalist is de naam Ansible al meerdere keren gevallen bij verschillende teams en zelfs recent toegepast binnen het Netwerk VSP team. Ansible wordt momenteel gebruikt door vele grote bekende bedrijven en heeft al meer dan 1miljoen downloads.

*Mina:*

Mina is een tool die ik gevonden heb door te zoeken op Google. Mina lijkt de opvolger van Capistrano en geeft aan veel sneller software deployment te kunnen realiseren door het gebruik van één SSH connectie. Omdat Mina in de toepassing lijkt op Capistrano zou het leren van deze tool vrij eenvoudig kunnen zijn.

*Jenkins:*

Jenkins is een tool die ik het meeste ben tegen gekomen binnen de interviews met de zeven teams. Jenkins biedt een flexibele werkomgeving omdat de scripten helemaal zelf uitgeschreven moeten worden. Dit biedt de mogelijkheid om een taak (Job) helemaal te laten functioneren hoe je zelf wilt.

### 

2.3.3 Informatie per tool

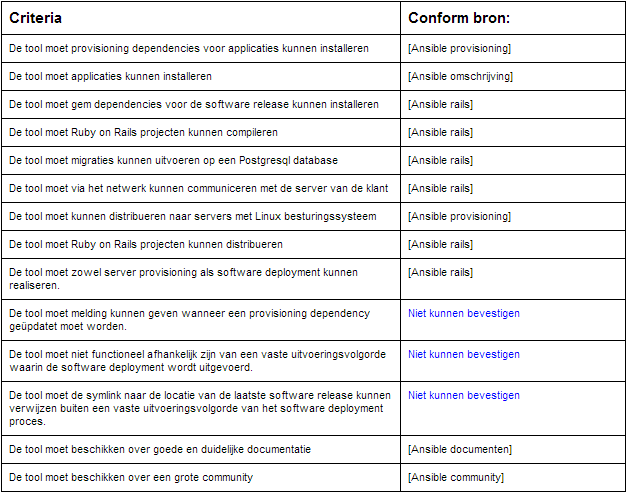
Nu we tools hebben opgesteld voor de longlist, is het interessant om te kijken wat er zoal geschreven is over de tools. Daarnaast is het interessant om te kijken welke bronnen bevestigen of de tool aan de opgestelde criteria voldoet of bevestigen welke criteria niet haalbaar blijken te zijn.

#### 

2.3.3.1 Ansible

Ansible is een tool waarmee je de installatie en configuratie van een server of applicatie kan automatiseren en daarmee kan herhalen. Bijvoorbeeld bij een herinstallatie, installatie van een nieuwe server of bij deployment van een nieuwe service. [Ansible omschrijving].

Aan de hand van informatie uit bronnen heb ik proberen te achterhalen of Ansible voldoet aan de opgestelde criteria.



*Voordelen:*

* Tool beschikt over goede en duidelijke documentatie en een grote community
* Tool is geschikt voor zowel software deployment als server provisioning
* Tool biedt de mogelijkheid om stappen uit te voeren in een zelf gemaakte uitvoeringsvolgorde.

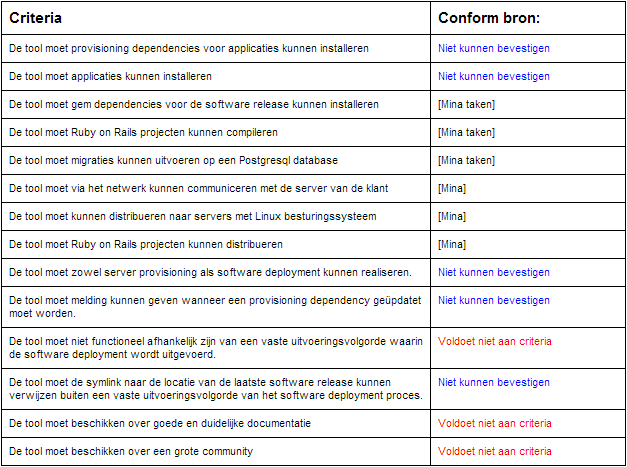
*Nadelen:*

* Tool maakt gebruik van YAML taal en is de vereist om te begrijpen.

2.3.3.2 Mina

Mini is een software deployment tool welke je scripten laat bouwen en draaien om je releases te beheren op een server via SSH. Mina maakt enkel maar gebruik van één SSH session per distributie. Met Mina is het mogelijk om elk soort project te distribueren welke via SSH te distribueren is. [Mina]

Aan de hand van informatie uit bronnen heb ik proberen te achterhalen of Mina voldoet aan de opgestelde criteria.



*Voordelen:*

* Door gebruik van één SSH session zou de tool sneller moeten zijn dan Capistrano.
* Geschreven in Ruby dus makkelijk te gebruiken voor het Webapps team.

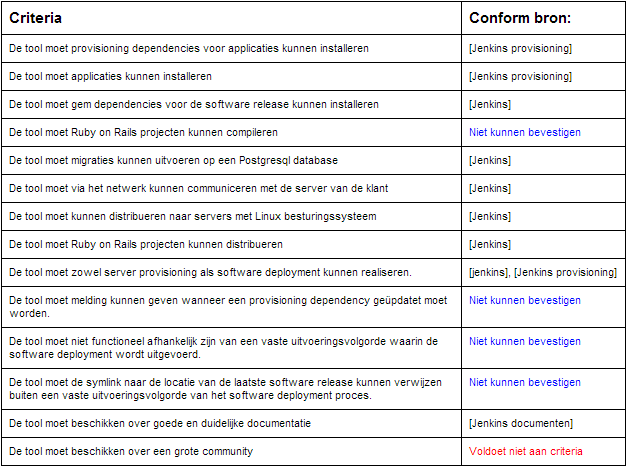
*Nadelen:*

* De tool mina kan geen server provisioning uitvoeren.
* Tool maakt gebruik van een vaste uitvoeringsvolgorde net als Capistrano waardoor het prepare deployment concept niet mogelijk is.
* Weinig documentatie en kleine community voor de tool Mina te vinden.

2.3.3.3. Jenkins

Jenkins is een open source Continious integration tool geschreven in Java. [Jenkins wiki]

Aan de hand van informatie uit bronnen heb ik proberen te achterhalen of Jenkins voldoet aan de opgestelde criteria.



*Voordelen:*

* Jenkins heeft een zeer flexibele implementatie van jobs omdat deze zelf geschreven worden. Door zelf de commando´s zelf op volgorde te schrijven is de kans groot om de knelpunten van het Webapps team op te lossen.
* Jenkins is een tool met een GUI wat eenvoudig te bedienen is.

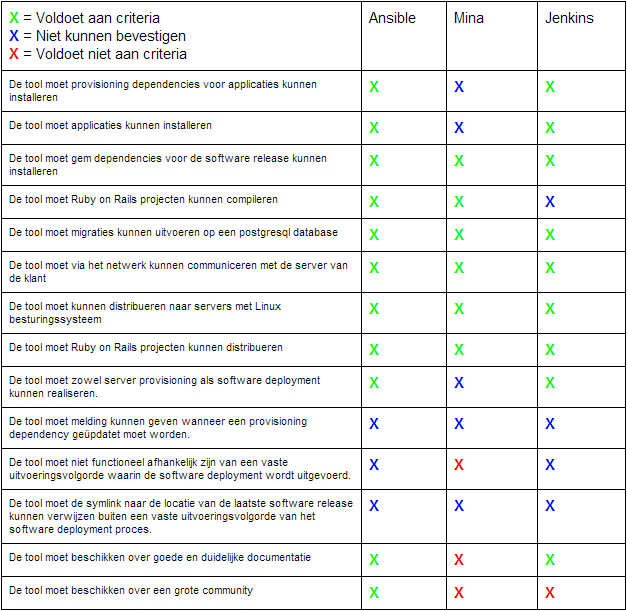
*Nadelen:*

* Het uitschrijven van alle Jobs kost ontzettend veel tijd in tegenstelling tot tools die zelf taken genereren.
* De structuur waarin de Jobs worden ingedeeld moet zorgvuldig en goed worden verzonnen. De kans op bijna soortgelijke Jobs is erg groot.
* Veel Linux kennis vereist omdat de commando´s in de scripten Linux commando´s zijn.

2.3.4 Keuze tool uit longlist

Omdat het testen van een software distributie tool ontzettend veel tijd in beslag neemt, maak ik één keuze uit de tools die opgesteld zijn in de longlist. Deze tool zal verder worden uitgewerkt binnen deelvraag 4. Vanaf nu is afbakening AF4 van toepassing.

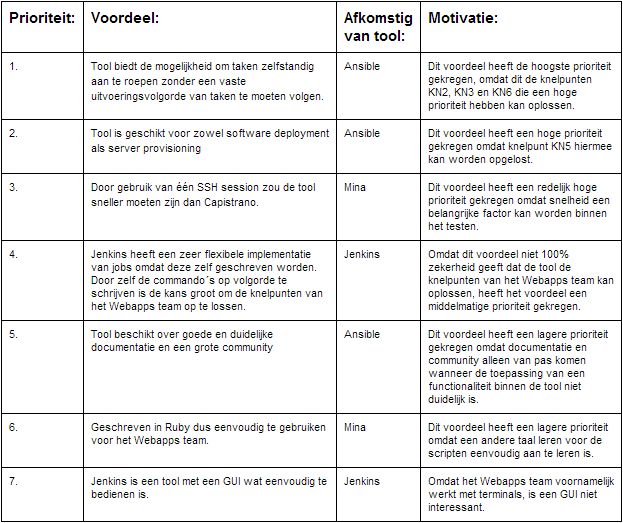
Om een goed besluit te kunnen nemen welke tool ik verder wil testen, stel ik een tabel op met alle criteria en tools om een overzicht te krijgen:



Binnen het overzicht lijkt Ansible te voldoen aan de meeste opgestelde criteria punten. Ook heeft Ansible als enige tool voor ieder criteria punt welke opgesteld is uit een (M)ust have requirement een bron waarin bevestigd wordt dat Ansible het criteria punt kan uitvoeren.

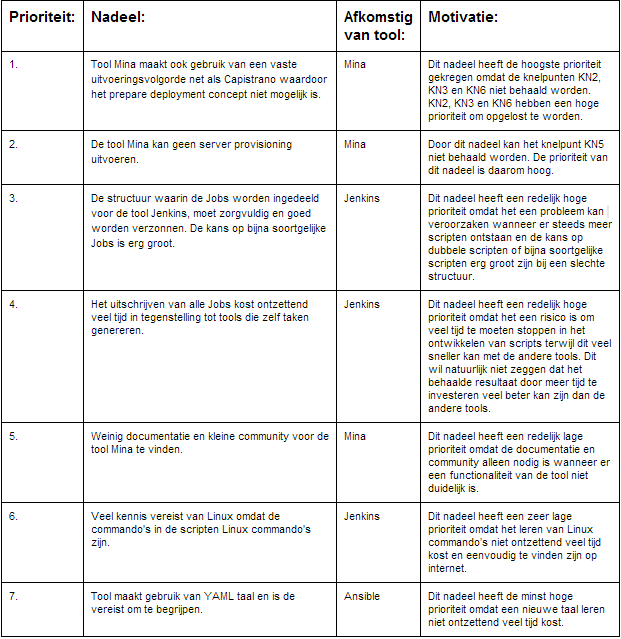
Omdat ik mijn keuze niet alleen wil baseren op de tool die aan de meeste criteria punten voldoet, wil ik de voor- en nadelen tegenover elkaar houden om af te wegen welke tool op basis hiervan het meest geschikt lijkt.

Hieronder een overzicht met alle voordelen van de tools. De voordelen staan op prioriteit en hebben allemaal een motivatie waarom het voordeel deze prioriteit gekregen heeft.



Zoals te zien is in de tabel, zijn de eerste twee voordelen afkomstig van de tool Ansible. Deze twee voordelen garanderen namelijk knelpunten op te kunnen lossen.

Hieronder een overzicht met alle nadelen van de tools. De nadelen staan op prioriteit en hebben allemaal een motivatie waarom het voordeel deze prioriteit gekregen heeft. Het nadeel op prioriteit 1 is het “grootste” nadeel.



In de tabel is te zien dat de tool Mina de grootste nadelen met zich mee brengt. Ansible heeft maar één nadeel en staat onderaan de lijst.

Op basis van alle vergelijkingen is de keuzevolgorde als volgt:

1. Ansible
2. Jenkins
3. Mina

Ansible staat op nummer 1 omdat de tool aan de meeste criteria punten voldoet, de beste voordelen en de minst erge nadelen heeft.

De tools Jenkins en Mina zouden eventueel in een vervolgonderzoek getest kunnen worden.

## 

2.4 Deelvraag 4: Welke resultaten leveren de beste tool uit de longlist op t.o.v. Chef en Capistrano in praktijk?

Binnen de laatste deelvraag wil ik Ansible en de huidige gebruikte tools Chef en Capistrano V3.0 testen in praktijk.

### 

2.4.1 Architectuur Ansible

Zoals eerder in deelvraag 1 de architectuur van Chef en Capistrano V3.0 zijn toegelicht, wil ik de architectuur van Ansible beschrijven. Dit wil ik doen door de mappenstructuur te illustreren en toe te lichten.



*Hostfile:*

Binnen de hostfile worden de ip adressen beschreven van de servers. Het is mogelijk om de ip adressen van de servers te verdelen in groepen. Deze groepen kun je vervolgens aanroepen om voor de gehele groep een provisioning of deployment te starten. Een voorbeeld is:

[webservers]

33.33.33.10

[appservers]

33.33.33.20

Binnen deze hostfile zijn er 2 groepen die je vervolgens weer apart kunt aanroepen in je playbooks of roles.

*Playbooks:*

Een playbook bestand is geschreven in YAML en bevat een opsomming van uit te voeren tasks. Tasks zijn niets anders dan een kleine taak waarin je een kleine uitvoering uitvoert bijvoorbeeld "apache herstarten" of "een map aanmaken". Binnen de taken kun je gebruik maken van Ansible modules. Door deze modules kun je eenvoudig voorgeprogrammeerde functionaliteiten uitvoeren. Voor de duidelijkheid laat ik een voorbeeld zien van een playbook:

**---**

**- hosts: webservers**

**tasks:**

**- name: Maak test map aan in op de server**

**file: path=/var/test state=directory**

**Bij de hosts wordt aangegeven dat alle servers die onder de groep webservers vallen de taak moeten uitvoeren.**

**File is een voorbeeld van een Ansible module. Er hoeft enkel alleen nog een path en state variabel beschreven te worden. Door de Ansible module "file" wordt nu automatisch een map aangemaakt op het aangegeven path op de server.**

Om de omvang van een playbook te beschrijven kun je denken aan bijvoorbeeld een playbook voor het installeren van Ruby, een playbook voor het installeren van Postgresql of een playbook voor het installeren van mysql.

*Roles:*

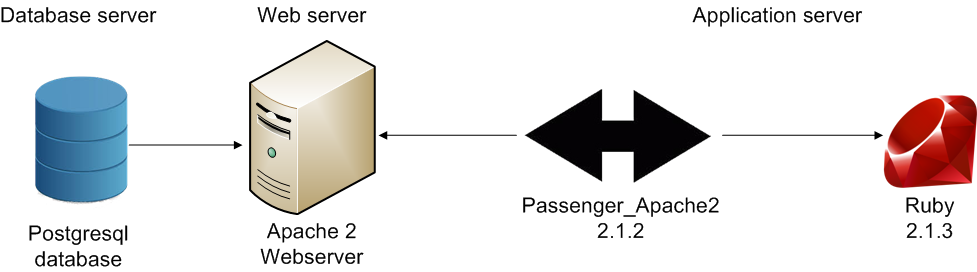
Binnen een role is het mogelijk om een opsomming van playbooks te beschrijven. Elke rol heeft standaard zes mappen waarin het volgende mogelijk is:

* Files: Bestanden die je kunt uploaden naar de server.
* Meta: Dependencies over welke role als eerst uitgevoerd moet zijn voor je de volgende role kunt uitvoeren.  
  Template: standaard configuratie bestand welke ingevuld wordt met variabelen en vervolgens naar de server geschreven wordt wanneer deze veranderd is.
* Handlers: Alle uitgeschreven handlers ( zijn taken die starten conform een conditie )
* Tasks: Alle uitgeschreven taken. ( Wanneer main.yml aanwezig is zal deze taak standaard uitgevoerd worden.
* Vars: Alle variabelen met uitgeschreven waardes.

Om de omvang van een role te beschrijven kun je denken aan bijvoorbeeld een role voor een complete web server voor rails projecten waar Ruby, Apache2 en Passenger op geïnstalleerd worden.

2.4.2 Architectuur test project (Fairwear)

Om de software distributie testen uit te kunnen voeren, is er een bestaand Ruby on rails project beschikbaar gesteld. Het project bevat geen extreem afwijkende punten binnen de architectuur en komt overeen met het meest voorkomende architectuur scenario. De architectuur ziet er als volgt uit:



Zoals te zien is binnen de architectuur is er maar één apache2 web server actief. Op dezelfde machine is één Postgresql database actief.

Voor het Fairwear project is Ruby 2.1.3 vereist. Een andere belangrijke module die vermeldt staat is Passenger\_Apache2. Deze module zorgt voor communicatie tussen Ruby en Apache2. Voor het Fairwear project is Passenger\_Apache2 2.1.2 vereist.

Kortom, er is één machine aanwezig die de functies web server, database server en application server op zich neemt.

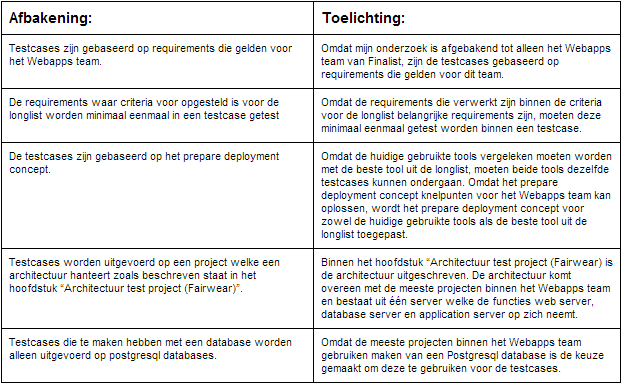
### 

2.4.3 Testcases

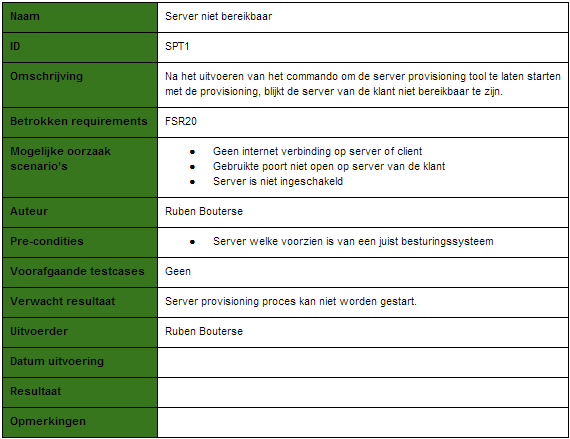
Om de huidige tools en Ansible te kunnen testen in praktijk dienen er testcases opgesteld te moeten worden. De resultaten die de testcases opleveren kunnen inzicht geven in de functionaliteit van de tools in praktijk. De resultaten kunnen van grote betekenis zijn bij het opstellen van het advies.

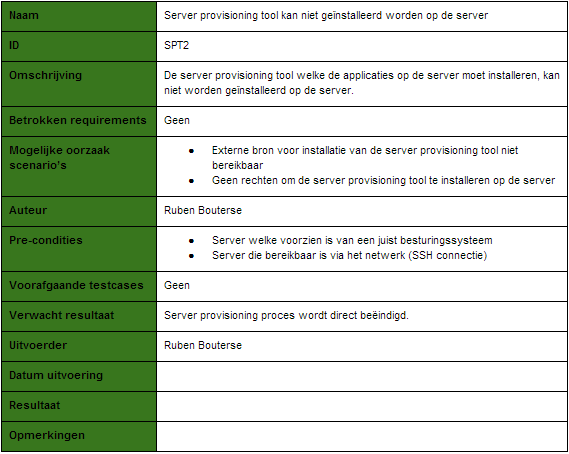
2.4.3.1 Scope

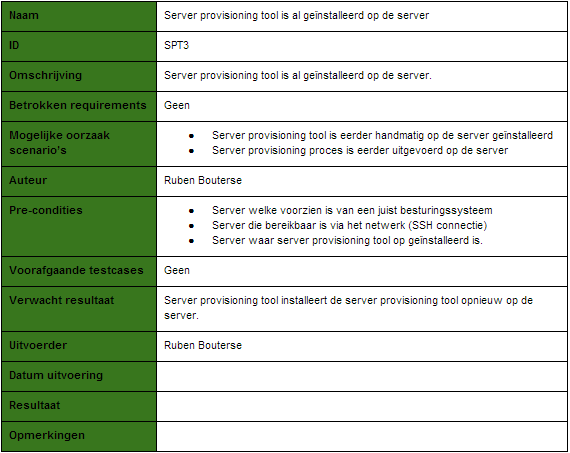
De testcases worden uitgevoerd binnen een omgeving waarin een aantal afbakeningen gelden. De afbakeningen die van toepassing zijn, zijn als volgt:

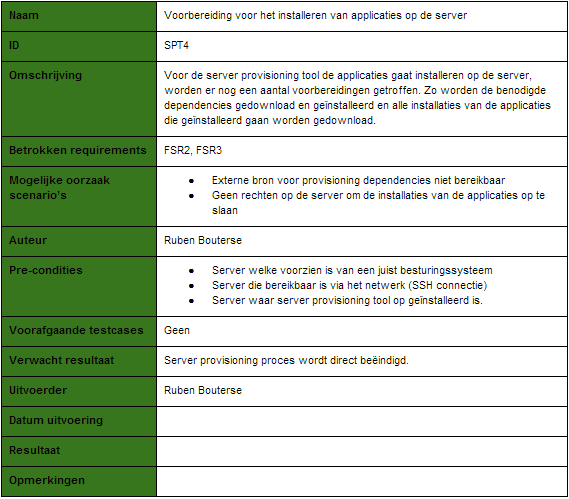


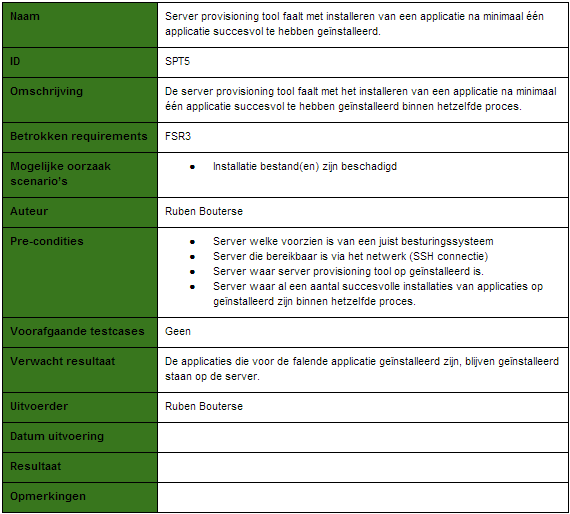
2.4.3.2 Testcases server provisioning proces

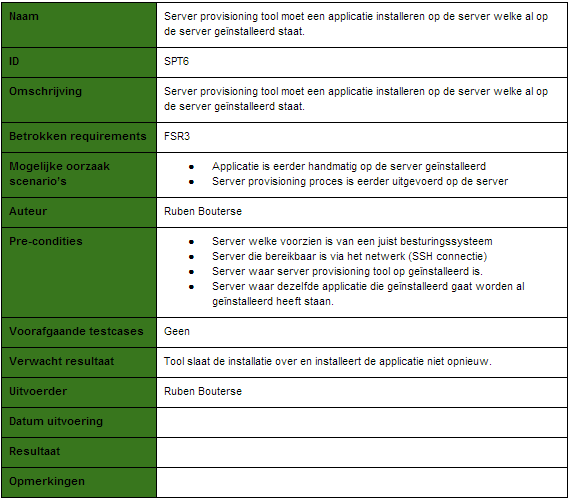


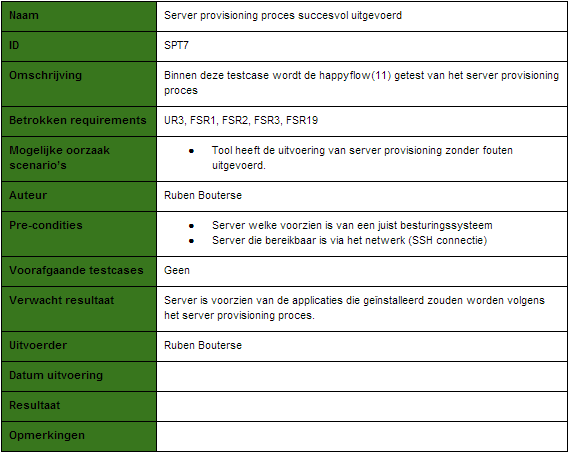




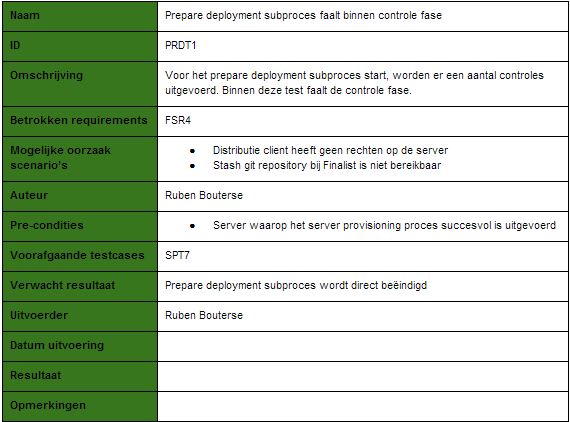


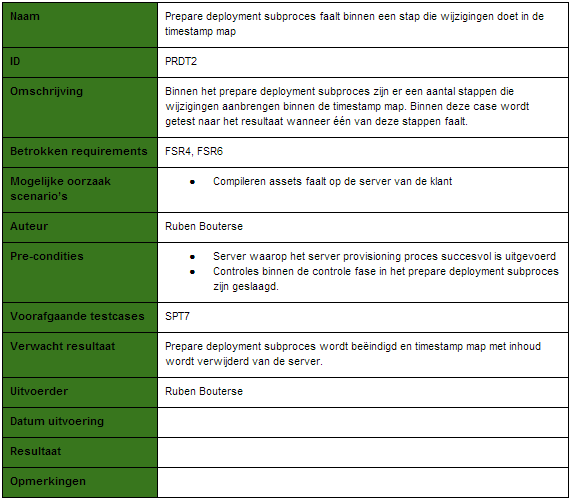


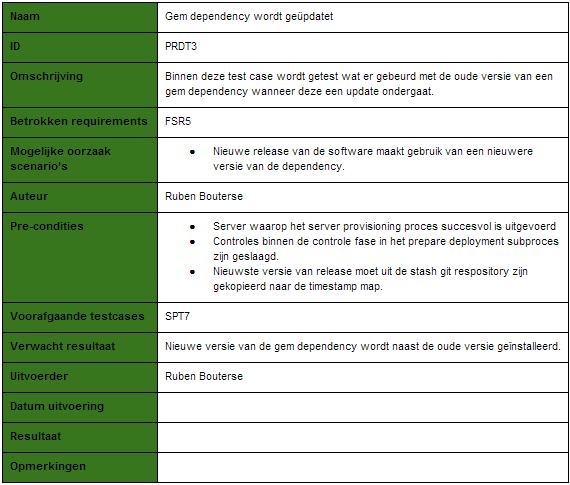




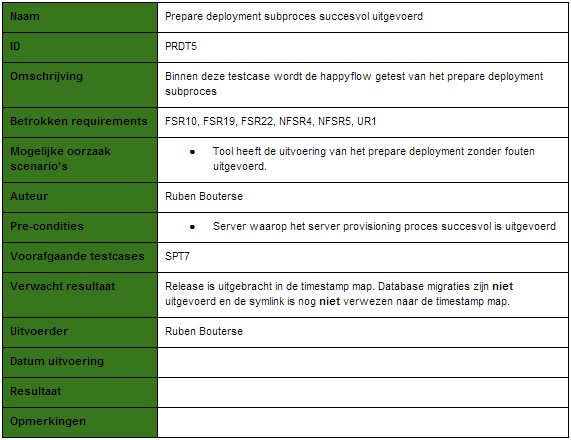
2.4.3.3 Testcases prepare deployment subproces



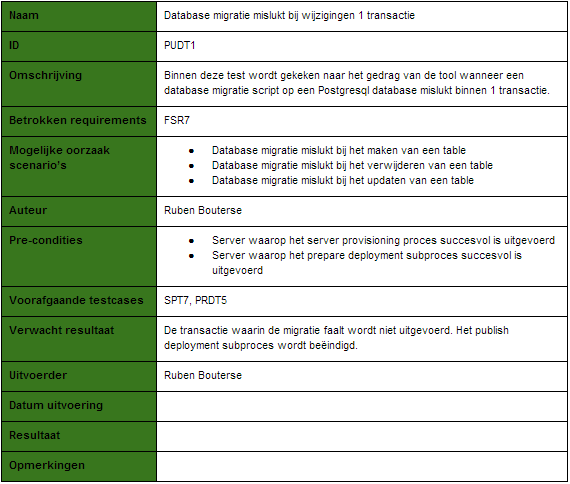


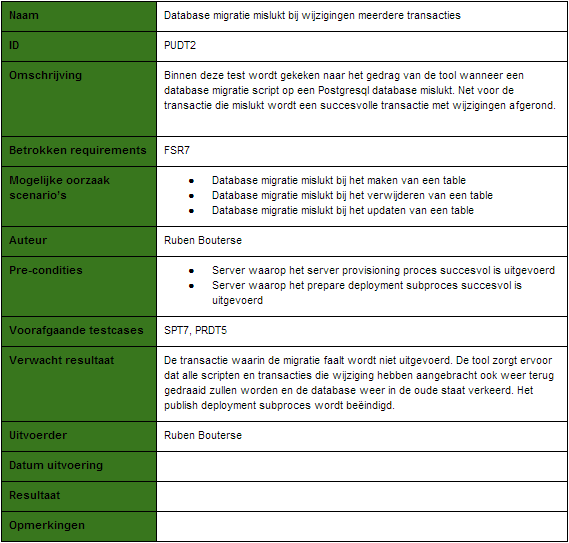


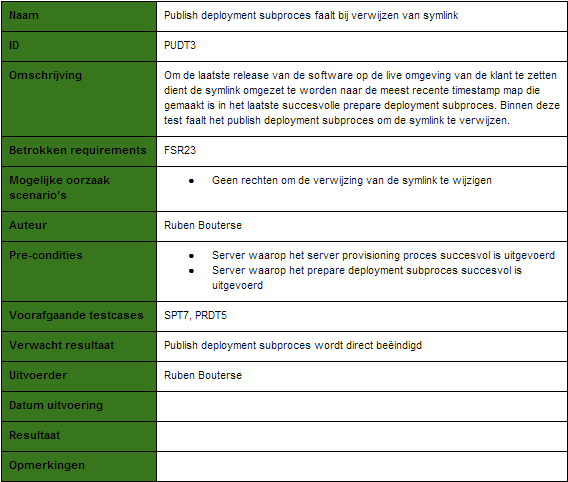


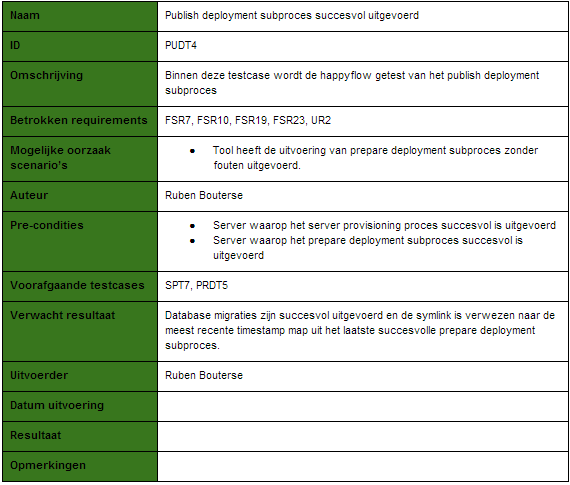


2.4.3.4 Testcases publish deployment subproces









2.4.4 Resultaten testcases

De opgestelde testcases zijn uitgevoerd voor zowel de huidige situatie als de toekomstige situatie. Dit wil zeggen dat de tools Chef, Capistrano V3.0 en Ansible getest zijn. Helaas nam het testen van de tools in praktijk veel tijd in beslag en is het niet gelukt om alle testcases binnen mijn afstudeerperiode uit te voeren.

De resultaten van de testcases zijn te vinden in het testrapport welke als bijlage te vinden is in hoofdstuk 4.4.

# 

3. Evaluatie en bevindingen

3.1 Samenvatting onderzoek

Het doel van dit onderzoek is antwoord te krijgen op de hoofdvraag “Welke software distributie tool sluit het beste aan op de software distributie binnen het Webapps team?”. Om een duidelijk antwoord op deze hoofdvraag te krijgen, zijn er vier deelvragen beantwoord.

De eerste deelvraag begint met het beschrijven van de infrastructuur binnen Finalist. Vervolgens wordt er een duidelijk beeld van de huidige situatie van software distributie binnen het Webapps team beschreven. Binnen de huidige situatie worden de processen die binnen de software distributie vallen in detail uitgeschreven.

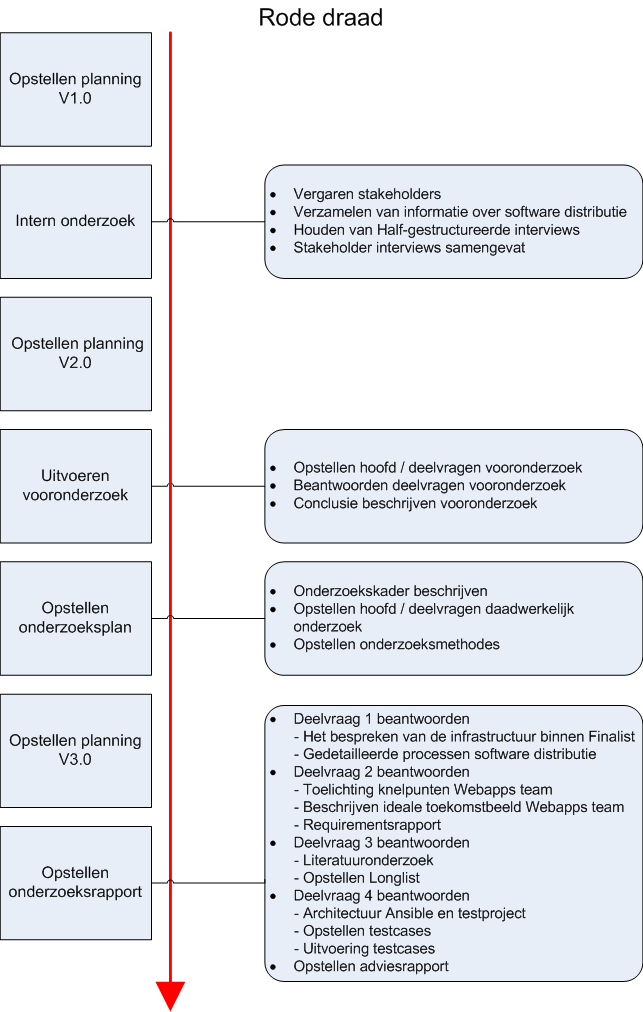
De tweede deelvraag beschrijft de knelpunten die vastgesteld zijn. Daarnaast heb ik aan de hand van een interview het ideale toekomstbeeld van het Webapps team kunnen achterhalen en deze uitgeschreven in de deelvraag. Het ideale toekomstbeeld is beschreven om mogelijke oplossingen van de knelpunten vast te stellen. Tot slot heb ik requirements opgesteld en verwerkt in een requirementsrapport.

Binnen de derde deelvraag is een longlist opgesteld met als doel een tool te vinden welke de beste kandidaat is om toe te passen voor het Webapps team. De eerste stap die is ondernomen is het opstellen van criteria voor de longlist. De criteria zijn opgesteld conform (M)ust have requirements, requirements die verbonden zijn aan een knelpunt en extra factoren. Vervolgens worden er kandidaten beschreven voor de longlist. Er is een literatuuronderzoek gestart om de criteria te bevestigen voor de opgestelde kandidaten. Tot slot is er één tool uit de longlist gekozen welke als beste kandidaat voor het Webapps team zou kunnen functioneren.

Binnen de vierde en laatste deelvraag zijn de tools van de huidige situatie en de beste tool uit de longlist in praktijk getest. De testen bestaan uit testcases welke opgesteld zijn conform de opgestelde requirements. Helaas nam het testen van de tools in praktijk veel tijd in beslag en is het niet gelukt om alle testcases binnen mijn afstudeerperiode uit te voeren.

3.1.1 De rode draad

Om mijn onderzoek tot een goed eindresultaat te brengen, heb ik zorgvuldig gewerkt aan een goede en duidelijke rode draad binnen mijn onderzoek. De rode draad is als volgt:



3.2 Conclusie

3.2.1 Conclusie deelvraag 1

De software distributie van het Webapps team wordt uitgevoerd in twee processen. Om deze twee processen uit te kunnen voeren, dient de distributie client voorzien te worden van een aantal software programma’s.

De twee processen die uitgevoerd worden zijn als volgt:

*Server provisioning:*  
Het server provisioning proces wordt uitgevoerd in drie verschillende activiteiten namelijk:

* *Verzoek indienen*

Binnen deze activiteit wordt er een verzoek ingediend bij de systeembeheerder voor het aanmaken van een nieuwe server.

* *Installatie server*  
  Binnen deze activiteit wordt de server voorzien van een besturingssysteem.
* *Installatie benodigde applicaties*  
  Binnen deze activiteit wordt de server voorzien van de provisioning dependencies voor de applicatie die in het software deployment proces zal worden gedistribueerd. Het installeren van de provisioning dependencies wordt met behulp van de tool “Chef” uitgevoerd.

Server provisioning wordt eenmalig uitgevoerd. De reden waarom dit proces eenmalig uitgevoerd wordt komt omdat de server eenmalig ingericht moet worden met de juiste provisioning dependencies.

*Software deployment:*Het software deployment proces wordt uitgevoerd in twee verschillende activiteiten namelijk:

* *Installatie release*

Binnen deze activiteit wordt de nieuwste release van de applicatie gedistribueerd naar de server van de klant. Het distribueren van de nieuwste release wordt met behulp van de tool “Capistrano V3.0” uitgevoerd. Capistrano V3.0 maakt gebruik van 4 “stages” waarin stappen worden uitgevoerd om de software te distribueren. Deze stages worden in een vaste uitvoeringsvolgorde uitgevoerd en zijn niet aanpasbaar.

* *Testen door klant*  
  Binnen deze activiteit worden testen uitgevoerd door de klant.

Software deployment wordt herhalend uitgevoerd. De reden waarom dit proces herhalend uitgevoerd wordt komt omdat er steeds nieuwe releases gedistribueerd worden op de server.

3.2.2 Conclusie deelvraag 2

De knelpunten die binnen de software distributie van het Webapps team optreden zijn als volgt:

|  |  |  |
| --- | --- | --- |
| **ID:** | **Knelpunt:** | **Prioriteit:** |
| KN1 | Het downloaden van gem dependencies wordt gedaan van een externe bron. | Middel |
| KN2 | Het downloaden van gem dependencies valt binnen een vaste uitvoeringsvolgorde van de software deployment tool. | Hoog |
| KN3 | Het compileren van assets valt binnen een vaste uitvoeringsvolgorde van de software deployment tool. | Hoog |
| KN4 | Wanneer een provisioning dependency geüpdatet moet worden, verschijnt er geen foutmelding. | Laag |
| KN5 | Het server provisioning en software deployment proces gebruiken verschillende tools. | Middel |
| KN6 | Het verwijzen van de symlink naar de meest recente software release, valt binnen een vaste uitvoeringsvolgorde van de software deployment tool | Hoog |

Om deze knelpunten op te lossen zijn er drie concepten opgesteld voor een ideaal toekomstbeeld van het Webapps team. De drie concepten zijn als volgt:

*Prepare deployment*Binnen het prepare deployment concept wordt gesproken over het opdelen van het software deployment proces in twee subprocessen. Door de opdeling van twee subprocessen is het mogelijk om een release vroegtijdig te distribueren naar de server van de klant en zo mogelijke complicaties op tijd op te kunnen lossen. Echter is de release nog niet doorgevoerd op de live omgeving van de klant.

Binnen het tweede subproces wordt enkel alleen nog de database migraties en de symlink verwezen naar de laatst gedistribueerde release. Hierdoor wordt de nieuwste release in de live omgeving doorgevoerd.

Met het prepare deployment concept worden de volgende knelpunten opgelost:

|  |  |  |
| --- | --- | --- |
| **ID:** | **Knelpunt:** | **Prioriteit:** |
| KN1 | Het downloaden van gem dependencies wordt gedaan van een externe bron. | Middel |
| KN2 | Het downloaden van gem dependencies valt binnen een vaste uitvoeringsvolgorde van de software deployment tool. | Hoog |
| KN3 | Het compileren van assets valt binnen een vaste uitvoeringsvolgorde van de software deployment tool. | Hoog |
| KN6 | Het verwijzen van de symlink naar de meest recente software release, valt binnen een vaste uitvoeringsvolgorde van de software deployment tool | Hoog |

*Geen foutmelding bij het noodzakelijk updaten van provisioning dependencies*Binnen dit concept zijn twee opties opgesteld namelijk:

* *Optie1: server provisioning proces starten voor iedere uitvoering van het software deployment proces*

Wanneer deze optie van toepassing is, zal voor het uitbrengen van een release elke keer het server provisioning proces worden gestart. Provisioning dependencies die geüpdatet moeten worden, zullen worden geïnstalleerd. Het opleveren van een release zal vors toenemen in tijdsduur omdat bij iedere release het server provisioning proces gestart wordt. Echter zal het server provisioning proces niet zo lang duren als de eerste keer dat server provisioning is uitgevoerd.

* *Optie2: tonen van een melding wanneer een provisioning dependency bijgewerkt moet worden*Wanneer deze optie van toepassing is zal dit niet toenemen in de tijdsduur van de oplevering van een release. Deze optie zorgt ervoor dat een werknemer direct weet dat er een update vereist is en om welke provisioning dependency het gaat. Het updaten van de provisioning dependency zal niet veel tijd in beslag nemen.

Met dit concept wordt de volgende knelpunt opgelost:

|  |  |  |
| --- | --- | --- |
| **ID:** | **Knelpunt:** | **Prioriteit:** |
| KN4 | Wanneer een provisioning dependency geüpdatet moet worden, verschijnt er geen foutmelding. | Laag |

*Server provisioning en software deployment processen realiseerbaar binnen één software distributie tool*Binnen dit concept zijn twee opties opgesteld namelijk:

* *Optie1: tool zoeken die zowel server provisioning als software deployment kan realiseren*

Binnen deze optie zal er één tool worden gebruikt voor zowel het server provisioning als het software deployment proces. Dit brengt als voordeel met zich mee dat een medewerker nog maar één tool hoeft te begrijpen om twee processen uit te kunnen voeren.

* *Optie 2: tools server provisioning en software deployment gescheiden houden*

Binnen deze optie zullen beide processen gebruik maken van een eigen tool. In de huidige situatie is dit ook gerealiseerd met Chef als tool voor server provisioning en Capistrano als tool voor software deployment.

Met dit concept wordt de volgende knelpunt opgelost:

|  |  |  |
| --- | --- | --- |
| **ID:** | **Knelpunt:** | **Prioriteit:** |
| KN5 | Het server provisioning en software deployment proces gebruiken verschillende tools. | Middel |

3.2.3 Conclusie deelvraag 3

Op basis van goede criteria, zijn er drie tools opgesteld binnen de longlist namelijk:

*Ansible:*

Ansible is momenteel een hype onder de automatisering tools. Ook binnen Finalist is de naam Ansible al meerdere keren gevallen bij verschillende teams en zelfs recent toegepast binnen het Netwerk VSP team. Ansible wordt momenteel gebruikt door vele grote bekende bedrijven en heeft al meer dan 1miljoen downloads.

*Mina:*

Mina is een tool die ik gevonden heb door te zoeken op Google. Mina lijkt de opvolger van Capistrano en geeft aan veel sneller software deployment te kunnen realiseren door het gebruik van één SSH connectie. Omdat Mina in de toepassing lijkt op Capistrano zou het leren van deze tool vrij eenvoudig kunnen zijn.

*Jenkins:*

Jenkins is een tool die ik het meeste ben tegen gekomen binnen de interviews met de zeven teams. Jenkins biedt een flexibele werkomgeving omdat de scripten helemaal zelf uitgeschreven moeten worden. Dit biedt de mogelijkheid om een taak (Job) helemaal te laten functioneren hoe je zelf wilt.

Om de definitieve keuze te maken welke tool uit de longlist het meest geschikt leek voor het Webapps team, heb ik samen in overleg met mijn begeleider de keuze gebaseerd op de volgende punten:

* Resultaten uit literatuuronderzoek die criteria punten bevestigen
* Resultaten voordelen
* Resultaten nadelen

Uit de resultaten bleek dat Ansible de tool is die waarschijnlijk het beste aansluit op de software distributie binnen het Webapps team, omdat deze aan de meeste criteria punten voldoet, De beste voordelen en de minst erge nadelen heeft. Ansible is dus de meest interessante tool om te testen in praktijk.

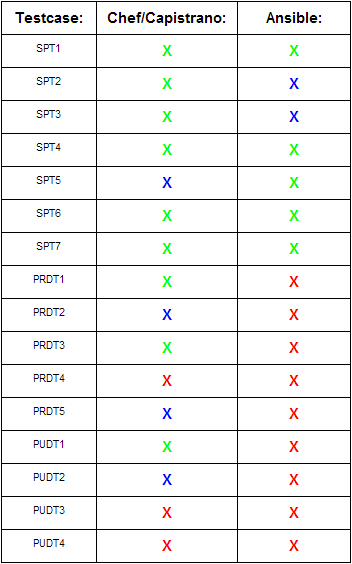
3.2.4 Conclusie deelvraag 4

Helaas nam het testen van de tools in praktijk veel tijd in beslag en is het niet gelukt om alle testcases binnen mijn afstudeerperiode uit te voeren. Het is daarom niet mogelijk om een volledige conclusie voor deze deelvraag op te stellen. Een overzicht met de resultaten is als volgt:

**X** = Testcase uitgevoerd met resultaat gelijk aan het verwacht resultaat

**X** = Testcase uitgevoerd met resultaat afwijkend dan het verwacht resultaat

**X** = Testcase niet uitgevoerd



We kunnen concluderen dat de testcases die beginnen met SPT (server provisioning testen) uitgevoerd zijn met zowel de tool Chef als de tool Ansible. De afwijkende resultaten van beide tools zijn als volgt:

*Chef:*

Wanneer we het afwijkende resultaat bekijken van testcase SPT5, valt er te concluderen dat het gaat om een positieve afwijking. Testcase SPT5 kan niet worden uitgevoerd omdat Chef de installatie bestanden al in een eerder stadium van het proces controleert en een foutmelding geeft bij een beschadigde installatie. Hierdoor is het dus onmogelijk dat er minimaal één applicatie succesvol geïnstalleerd is en vervolgens een beschadigde installatie faalt.

*Ansible:*

Wanneer we de afwijkende resultaten bekijken van de testcases SPT2 en SPT3, valt er te concluderen dat het gaat om een positieve afwijkingen. Ansible wordt t.o.v. Chef **niet** geïnstalleerd op de server. Hierdoor zijn de testcases SPT2 en SPT3 niet relevant voor de Tool Ansible.

We kunnen concluderen dat het uitvoeren van server provisioning met zowel Chef als Ansible mogelijk is. Het afwijkende gedrag op de uitgevoerde testcases geven geen negatief gevoel over de gebruikte tool. Omdat beide tools in staat zijn om server provisioning uit te voeren zonder extreme positieve of negatieve afwijkingen, concludeer ik dat Chef niet vervangen hoeft te worden met Ansible.

Helaas zijn de testcases voor software deployment niet uitgevoerd met de tool Ansible. Hierdoor is het niet mogelijk een conclusie te kunnen trekken tussen beide software deployment tools.

3.2.5 Eindconclusie

Nu de resultaten en conclusies van de deelvragen grotendeels beantwoordt zijn, is het tijd om de hoofdvraag te beantwoorden. Een herhaling van de hoofdvraag is:

*“Welke software distributie tool sluit het beste aan op de software distributie binnen het Webapps team?”*

In deelvraag 1 hebben we kunnen concluderen dat de software distributie binnen het Webapps team bestaat uit twee processen die bestaan uit verschillende activiteiten namelijk:

* Server provisioning proces
  + *Verzoek indienen*
  + *Installatie server*
  + *Installatie benodigde installaties*
* Software deployment proces
  + *Installatie release*
  + *Testen door klant*

Omdat de processen in deelvraag 1 in detail uitgewerkt zijn, kunnen we een klein deel van de hoofdvraag verduidelijken namelijk:

*“Welke software distributie tool sluit het beste aan op de* ***software distributie*** *binnen* ***het Webapps team****?”*

Wat we verstaan onder de term “software distributie” binnen het onderzoek.  
Wat we verstaan onder de term “het Webapps team” binnen het onderzoek.

In deelvraag 2 hebben we kunnen concluderen dat er zich een aantal knelpunten voordoen in de huidige manier van software distributie binnen het Webapps team. Voor deze knelpunten zijn er opties beschreven voor een ideaal toekomstbeeld voor het Webapps team. Op basis van de verzamelde gegevens heb ik requirements kunnen opstellen. Door opstellen van de requirements, kunnen we een klein deel van de hoofdvraag verduidelijken namelijk:

*“Welke* ***software distributie tool*** *sluit het beste aan op de software distributie binnen het Webapps team?”*

Wat we verstaan onder de term “software distributie tool” binnen het onderzoek.

In deelvraag 3 hebben we kunnen concluderen dat Ansible de tool is die waarschijnlijk het beste aansluit op de software distributie binnen het Webapps team, omdat deze aan de meeste criteria punten voldoet, De beste voordelen en de minst erge nadelen heeft. Door deze conclusie kunnen we een klein deel van de hoofdvraag beantwoorden namelijk:

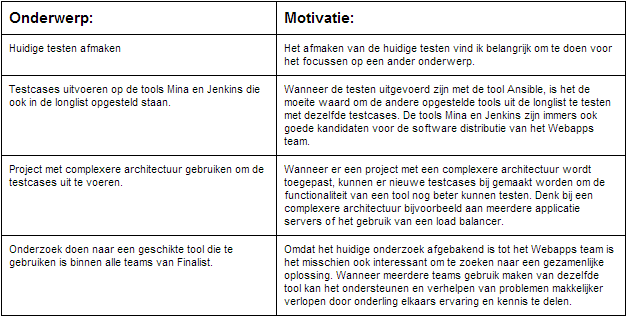
*“****Welke******software distributie tool*** *sluit het beste aan op de software distributie binnen het Webapps team?”*

Welke software distributie tool sluit **mogelijk** het beste aan op de software distributie binnen het Webapps team.

Helaas kunnen we geen volledige conclusie uit deelvraag 4 ophalen omdat niet alle resultaten behaald zijn. Hierdoor is het helaas niet mogelijk om een definitief antwoord op de hoofdvraag te geven.

3.3 Vervolgonderzoek

Ik heb een aantal onderwerpen binnen mijn onderzoek afgebakend of achterwegen gelaten. Deze onderwerpen zijn de moeite waard om in een vervolg onderzoek te onderzoeken. De belangrijkste onderwerpen zijn als volgt:



4. Bijlage

4.1 Vooronderzoek

Omdat dit rapport is toegevoegd als bijlage, verwijs ik voor deze bijlage naar hoofdstuk 10.3.1 van het afstudeerdossier.

4.2 Onderzoeksplan

Omdat dit rapport is toegevoegd als bijlage, verwijs ik voor deze bijlage naar hoofdstuk 10.3.2 van het afstudeerdossier.

4.3 Requirementsrapport

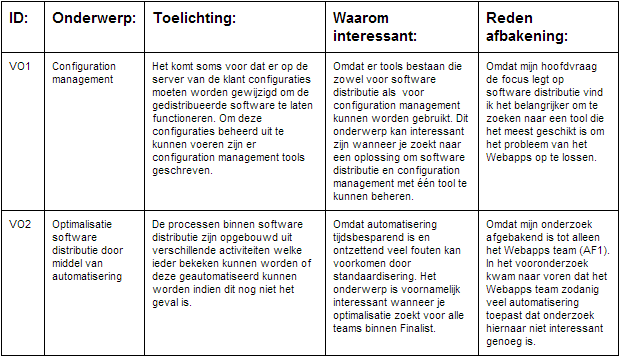
Omdat dit rapport is toegevoegd als bijlage, verwijs ik voor deze bijlage naar hoofdstuk 10.3.6 van het afstudeerdossier.

4.4 Testrapport

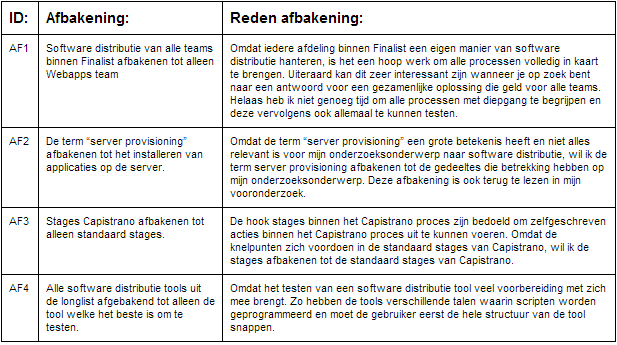
Omdat dit rapport is toegevoegd als bijlage, verwijs ik voor deze bijlage naar hoofdstuk 10.3.7 van het afstudeerdossier.

4.5 Afbakening

Hieronder een overzicht met de onderwerpen welke ik helaas niet in mijn onderzoek heb kunnen opnemen. Bij ieder onderwerp heb ik een motivatie gegeven waarom het onderwerp interessant is en wat de reden is waarom ik het onderwerp toch niet heb opgenomen in mijn onderzoek.



Ook zijn er onderwerpen die wel opgenomen zijn in het onderzoek maar afgebakend zijn tot een specifiek deel van het onderwerp. Hieronder een overzicht van de onderwerpen die afgebakend zijn:



5. Literatuurlijst

Binnen dit hoofdstuk kunt u de bronvermeldingen vinden welke gebruikt zijn binnen dit document. Elke bron is voorzien van een [Naam] welke in de tekst op dezelfde wijze is terug te vinden. Achter iedere bronvermeldingen worden zoveel mogelijk gegevens over de bron beschreven. De volgorde van deze gegevens zijn als volgt:

Uniek nummer, [Naam bron], Jaartal waaruit bron afstamt, auteur, titel boek/artikel, Geraadpleegd op datum, Link bron.

**1:** [Nel Verhoeven] 2007, Nel Verhoeven, Wat is onderzoek?

Geraadpleegd op 20-05-2014

Bron: Boek

**2:** [Command-line] 2013, Command-line-interface  
Geraadpleegd op 18-08-2014   
http://nl.wikipedia.org/wiki/Command-line-interface

**3:** [Knife] About Knife,

Geraadpleegd op 18-08-2014

<http://docs.getchef.com/knife.html>

**4:** [Gem-set] 2014, Ruby ( programming language )

Geraadpleegd op 19-08-2014

<http://en.wikipedia.org/wiki/Ruby_%28programming_language%29>

**5:** [Capistrano] 2014, Capistrano ( Software )

Geraadpleegd op 19-08-2014

<http://en.wikipedia.org/wiki/Capistrano_%28software%29>

**6:** [DSL] 2014, Domain-specific\_language

Geraadpleegd op 20-08-2014

<http://en.wikipedia.org/wiki/Domain-specific_language>

**7:** [Capistrano-flow] Deploy flow

Geraadpleegd op 20-08-2014

<http://capistranorb.com/documentation/getting-started/flow/>

**8:** [Symlink] 2013, Symlink

Geraadpleegd op 20-08-2014

<https://wiki.debian.org/SymLink>

**9:** [Ansible omschrijving] 2013, Introductie in Ansible

Geraadpleegd op 13-10-2014

<http://www.m3r.nl/blog/introductie-in-ansible-eenvoudig-geautomatiseerd-configuratie-en-serverbeheer/>

**10:** [Ansible rails] 2014, Maximiliun Gaß, Deploying rails with Ansible

Geraadpleegd op 13-10-2014

http://www.mxey.net/deploying-rails-with-ansible/

**11:** [Ansible provisioning] Deploying to Ubuntu using Ansible

Geraadpleegd op 13-10-2014

<http://guides.spreecommerce.com/developer/ansible-ubuntu.html>

**12:** [Ansible documenten]2014, Ansible Documentation

Geraadpleegd op 13-10-2014

http://docs.ansible.com/

**13:** [Ansible community] 2014, Ansible Google

Geraadpleegd op 13-10-2014

https://www.Google.nl/search?q=ansible&oq=ansible&aqs=chrome.0.69i59l2j69i65l2j69i59l2.819j0j8&sourceid=chrome&es\_sm=93&ie=UTF-8

**14:** [Mina] 2014, How to use Mina to Deploy a Ruby on Rails Application

Geraadpleegd op 14-10-2014

https://www.digitalocean.com/community/tutorials/how-to-use-mina-to-deploy-a-ruby-on-rails-application

**15:** [Mina taken] 2014, Tasks

Geraadpleegd op 14-10-2014

<http://nadarei.co/mina/tasks/>

**16:** [Jenkins]2014, Thomas Valent, Jenkins CI for Rails 4, RSpec, Cucumber, Selenium

Geraadpleegd op 15-10-2014

http://www.eq8.eu/blogs/6-jenkins-ci-for-rails-4-rspec-cucumber-selenium

**17:** [Jenkins provisioning]2012, Marcel Brikner, Continiously delivery in the cloud

Geraadpleegd op 15-10-2014

<https://blog.codecentric.de/en/2012/05/continuous-delivery-in-the-cloud-part-4-provisioning-your-test-staging-and-production-environments/>

**18:** [Jenkins documenten]2014, Use Jenkins

Geraadpleegd op 15-10-2014

<https://wiki.jenkins-ci.org/display/JENKINS/Use+Jenkins>

**19:** [Jenkins wiki] 5 December 2014, Jenkins (software)

Geraadpleegd op 15-10-2014  
http://en.wikipedia.org/wiki/Jenkins\_%28software%29

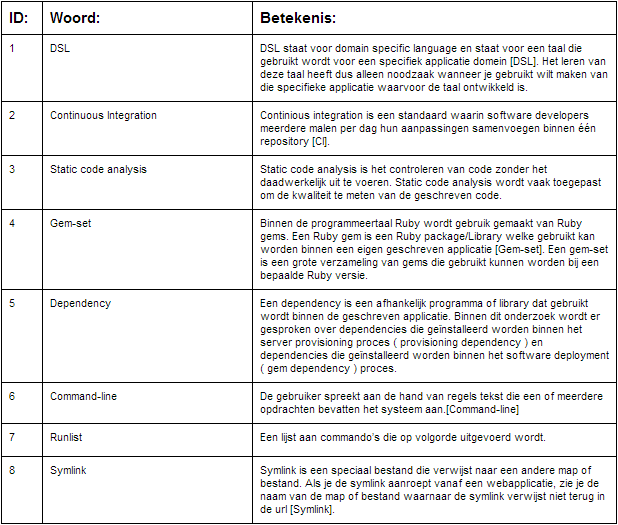
**20:** [CI] 2 December 2014, Continious integrations

Geraadpleegd op 25-05-2014

http://en.wikipedia.org/wiki/Continuous\_integration

# 

6. Glossary



### 10.3.6 Requirementsrapport

**Requirementsrapport**

Onderzoek software distributie

Welke software distributie tool sluit het beste aan op de software distributie binnen het Webapps team?



Auteur: Ruben Bouterse Eerste docent: H.P.Weenink

Opleiding: Informatica Tweede docent: G.M.Tuk

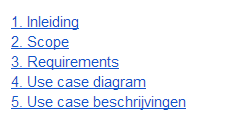
Bedrijf: Finalist Begeleider: R.Schellhorn

Opdrachtgever: M. van Breukelen

Datum: 22-09-2014

Versie: 1.0

Inhoudsopgave



1. Inleiding

Dit document beschrijft de requirements die opgesteld zijn voor het onderzoek “Software distributie” voor Finalist. De requirements zijn van toepassing voor software distributie tools die knelpunten binnen de software distributie processen van Finalist kunnen verbeteren.

De requirements zijn opgesteld aan de hand van verschillende bronnen namelijk:

* Stakeholder gesprekken ( Interviews )
* Resultaten van deelvraag één uit het onderzoeksrapport
* Beschrijving van het ideale toekomstbeeld van het Webapps team uit het onderzoeksrapport

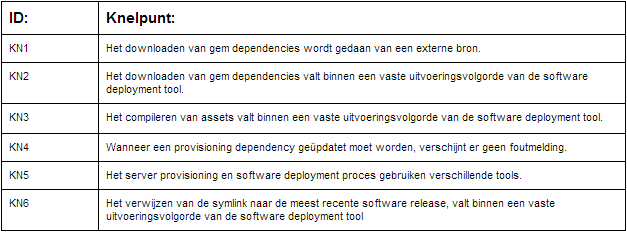
2. Scope

Vanwege een afbakening in het onderzoek “Software distributie” zijn de requirements alleen geldig voor software distributie tools binnen het Webapps team. Daarnaast zijn de opgestelde requirements gebaseerd op activiteiten en stappen die binnen de processen van software distributie binnen het Webapps team uitgevoerd worden.

# 

3. Requirements

De requirements zijn voorzien van een uniek ID zodat iedere requirement te traceren is. Daarnaast is bij iedere requirement te lezen vanuit welke bron deze tot stand is gekomen en aan welk knelpunt deze gekoppeld kan worden. Een overzicht van de knelpunten zijn:

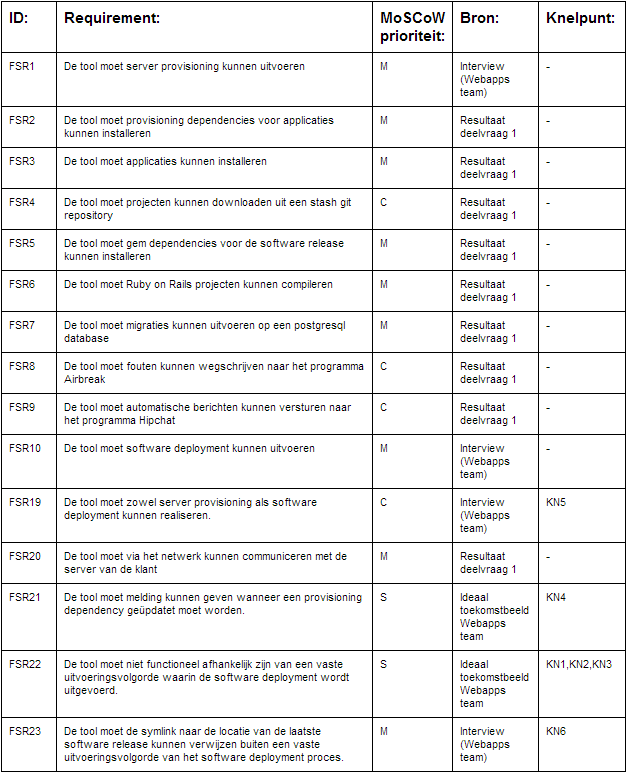


De requirements die niet gekoppeld zijn aan een knelpunt, zijn opgesteld aan de hand van de basis functionaliteiten die een software distributie tool dient te hebben op basis van de uitgeschreven processen in het onderzoeksrapport.

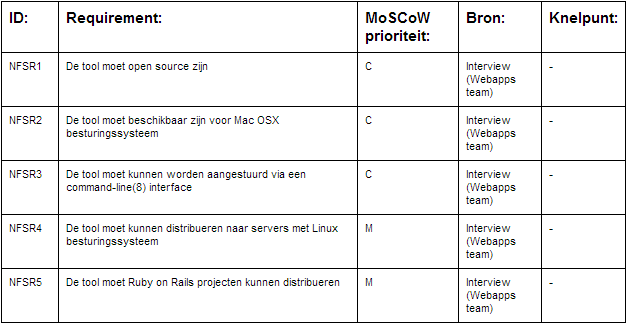
De requirements zijn voorzien van een prioriteit volgens de MoSCoW methode. Binnen de MoSCoW methode zijn vier verschillende prioriteiten beschikbaar. Een korte definitie van de vier prioriteiten zijn:



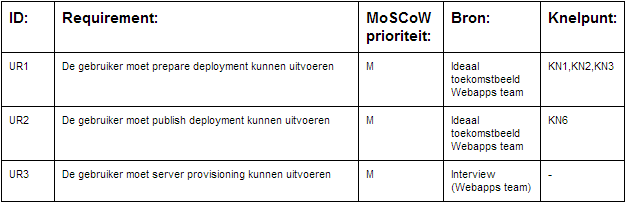
*Functional software requirement:*



*Non-functional software requirement:*



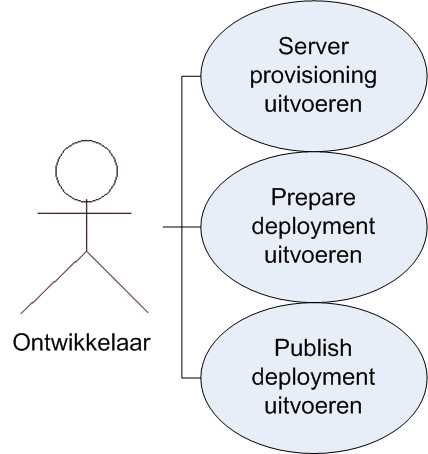
*User requirement:*



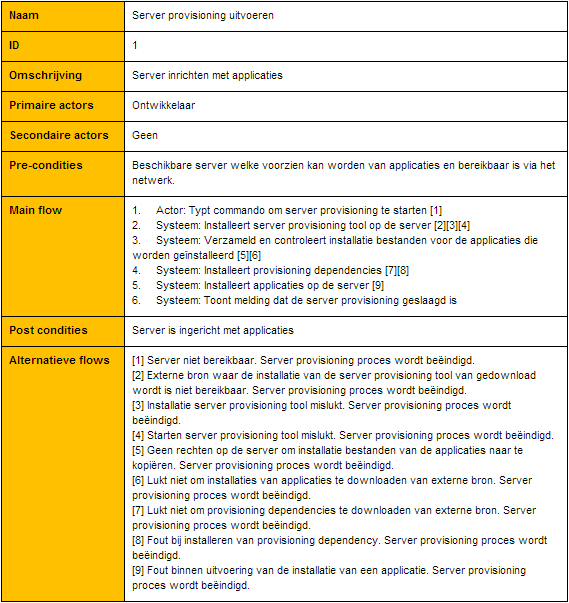
*Onderbouwing MoSCoW prioriteit:*

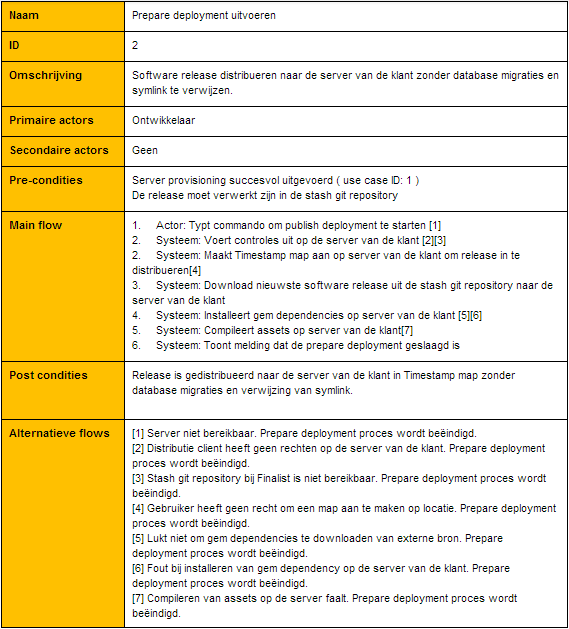
De prioriteit (M)ust is bij de functional software requirements toegewezen aan requirements die de noodzakelijke basis functionaliteiten beschrijven. Zonder één van deze requirements is het onmogelijk om het software distributie proces uit te voeren. Enkel alleen functional software requirement FSR23 is een requirement met de (M)ust have prioriteit zonder een noodzakelijke basis functionaliteit te zijn. Deze requirement heeft een knelpunt gekoppeld met een hoge prioriteit. Samen in overleg met mijn begeleider hebben we besloten om deze requirement ook een (M)ust have te geven. Alle overige requirements zijn in overleg met mijn begeleider toegewezen.

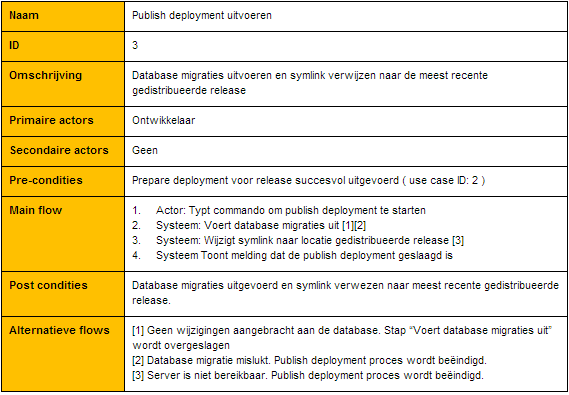
4. Use case diagram



# 

5. Use case beschrijvingen  
  






### 10.3.7 Testrapport

**Testrapport**

Onderzoek software distributie

Welke software distributie tool sluit het beste aan op de software distributie binnen het Webapps team?



Auteur: Ruben Bouterse Eerste docent: H.P.Weenink

Opleiding: Informatica Tweede docent: G.M.Tuk

Bedrijf: Finalist Begeleider: R.Schellhorn

Opdrachtgever: M. van Breukelen

Datum: 03-11-2014

Versie: 1.0

Inhoudsopgave

[Inhoudsopgave](#h.iqk84w22r8k4)

[Inleiding](#h.7pl5qah54xl4)

[1. Scope](#h.la3jq81d2onz)

[2. Resultaat testcases huidig gebruikte tools ( Chef + Capistrano )](#h.anrwyke9hpcp)

[2.1 Server provisioning testcases Chef](#h.4nxlkopmo6c6)

[2.2 Software deployment testcases Capistrano V3.0](#h.fybs6eahvefa)

[2.2.1 Prepare deployment subproces testcases](#h.73f49c1x3i9e)

[2.2.2 Publish deployment subproces testcases](#h.ir2a4hlyliwy)

[3. Resultaat testcases beste tool uit longlist ( Ansible )](#h.eohiuveycu2p)

[3.1 Server provisioning testcases Ansible](#h.5p4u7bbda671)

[3.2 Software deployment testcases Ansible](#h.ma1rlwobflz)

[3.2.1 Prepare deployment testcases](#h.mptmjw1caik1)

[3.2.2 Publish deployment testcases](#h.otqj8y2xffbm)

[4. Overzicht testcase resultaten](#h.mvez6wxy7o7d)

[5. Vergelijkingen en conclusies](https://docs.google.com/a/finalist.com/document/d/15eiieZIJSW7CigRFMLa0izK_SBYy9AYLbxlAnSMaGzk/edit#heading=h.hp8gp3wulycv)

[5.1 Server provisioning](https://docs.google.com/a/finalist.com/document/d/15eiieZIJSW7CigRFMLa0izK_SBYy9AYLbxlAnSMaGzk/edit#heading=h.c9bgqiom8we)

[5.2 Software deployment](https://docs.google.com/a/finalist.com/document/d/15eiieZIJSW7CigRFMLa0izK_SBYy9AYLbxlAnSMaGzk/edit#heading=h.m7gvrqbm5cb3)

Inleiding

Binnen het testrapport worden de resultaten verwerkt voor de opgestelde testcases. De testcases worden uitgevoerd op de tools uit de huidige situatie van het Webapps team en op de beste tool uit de opgestelde longlist. De resultaten van de testcases kunnen worden gebruikt voor het uitbrengen van een verbeteringsadvies.

De uitgevoerde testcases zijn ieder voorzien van kleur met de volgende betekenis:

**X** = Testcase uitgevoerd met resultaat gelijk aan het verwacht resultaat

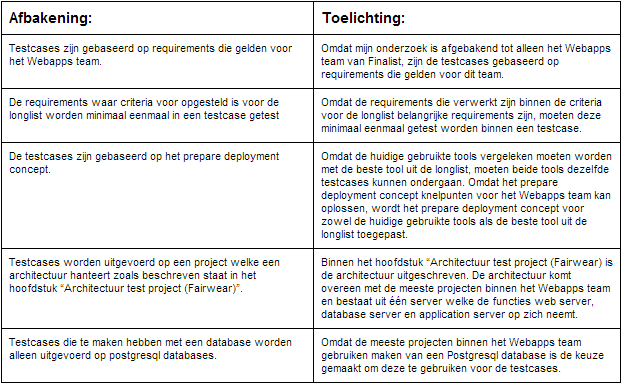
**X** = Testcase uitgevoerd met resultaat afwijkend dan het verwacht resultaat

**X** = Testcase niet uitgevoerd

Bij iedere uitgevoerde testcase wordt duidelijk aangegeven welk scenario gebruikt is om de testcase te testen.

1. Scope

De testcases zijn uitgevoerd binnen een omgeving waarin een aantal afbakeningen gelden. De afbakeningen die van toepassing zijn, zijn als volgt:

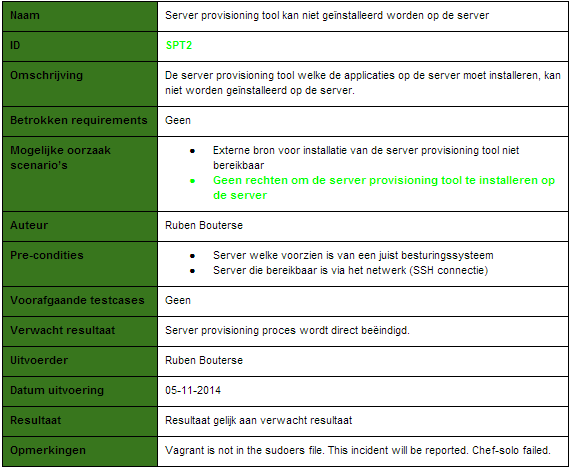


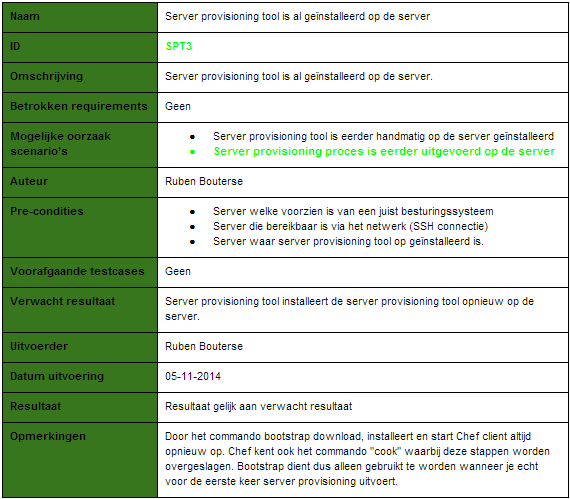
2. Resultaat testcases huidig gebruikte tools (Chef + Capistrano )

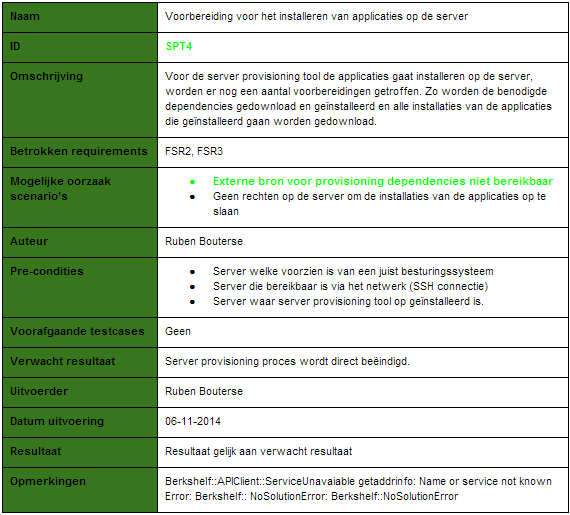
Binnen dit hoofdstuk zijn de resultaten te vinden van de testcases voor de tools die in de huidige situatie gebruikt worden. De testcases die opgesteld zijn voor het server provisioning proces zijn getest met de tool Chef. Voor het prepare deployment subproces en publish deployment subproces is getest met Capistrano V3.0.

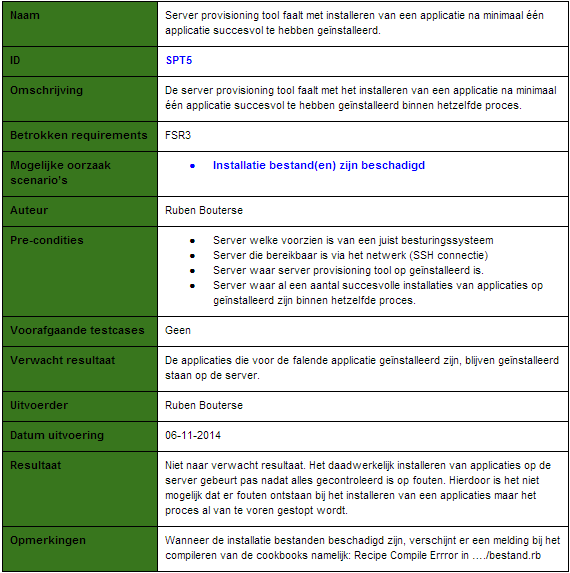
2.1 Server provisioning testcases Chef

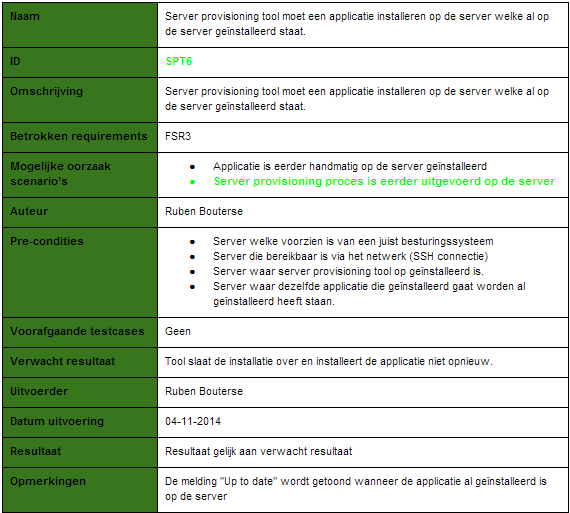


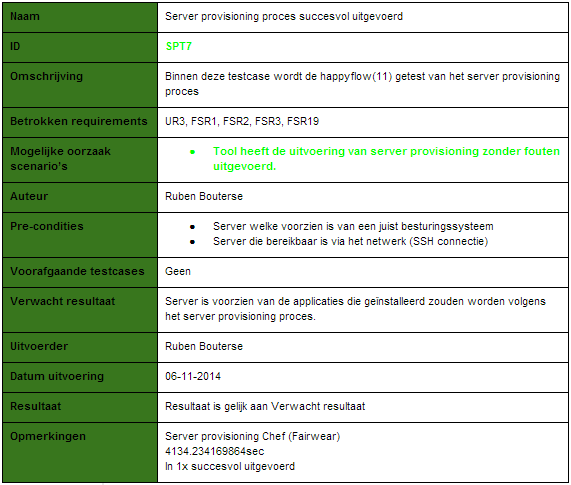






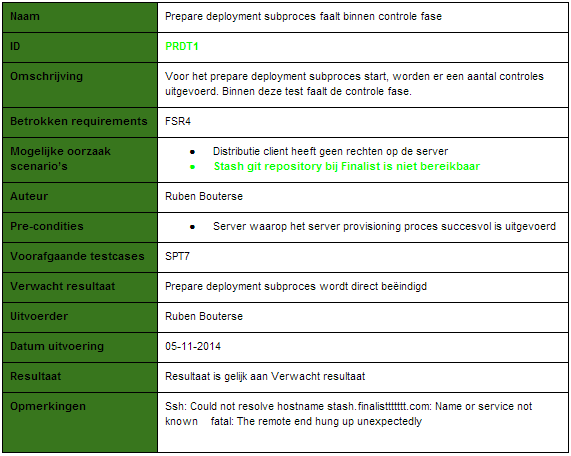


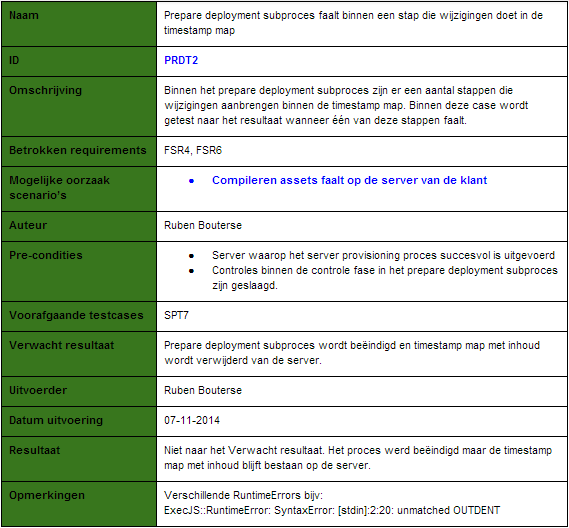


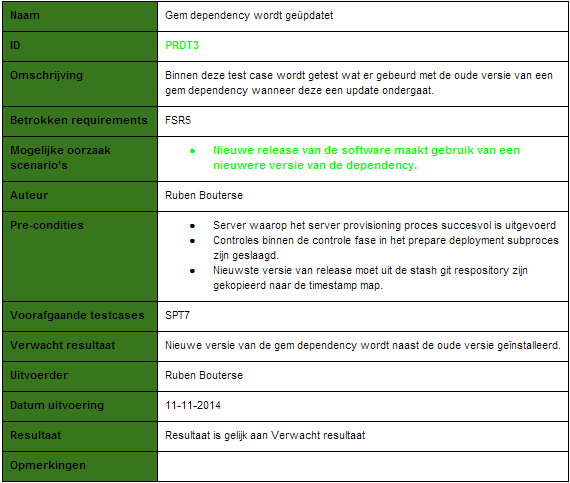


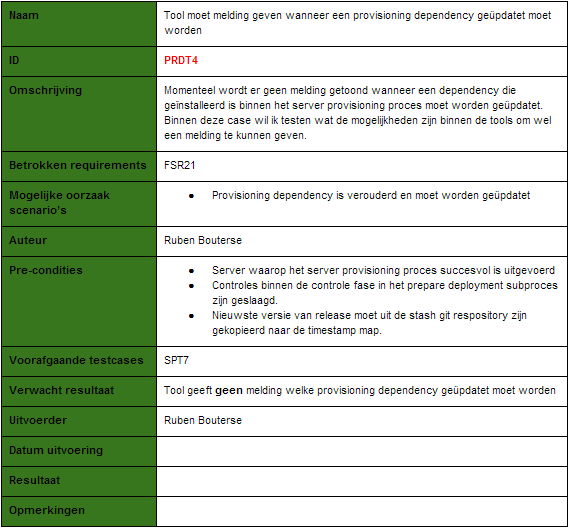
2.2 Software deployment testcases Capistrano V3.0

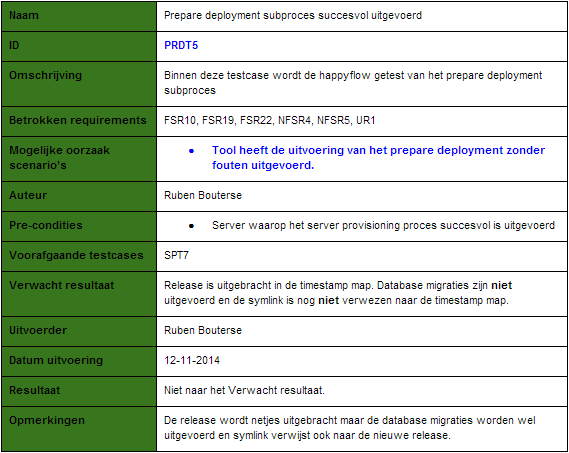
2.2.1 Prepare deployment subproces testcases





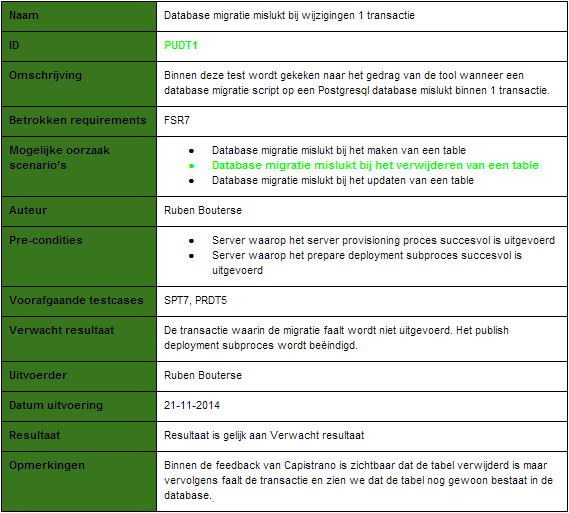


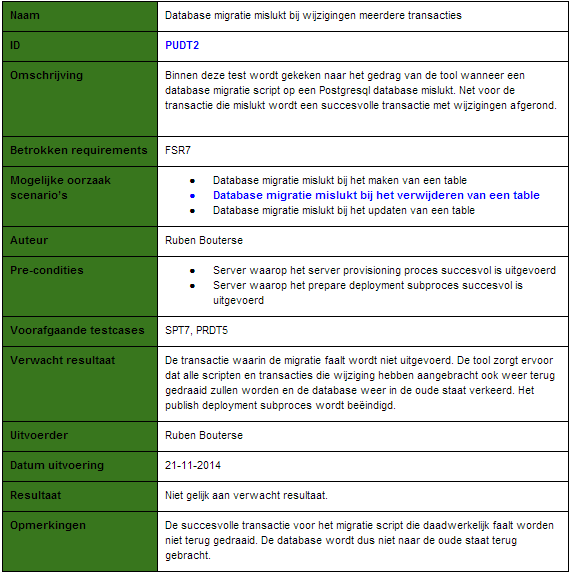


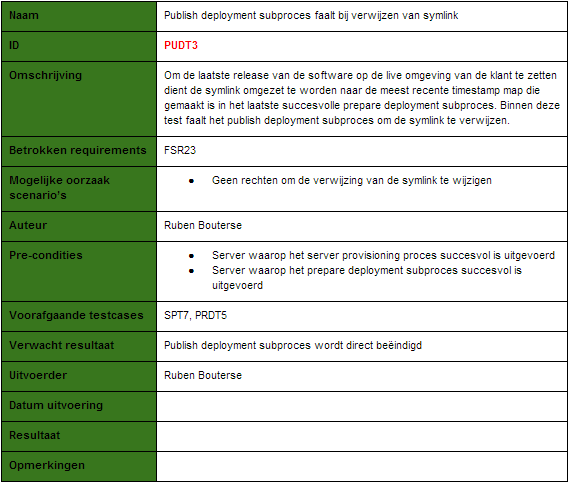


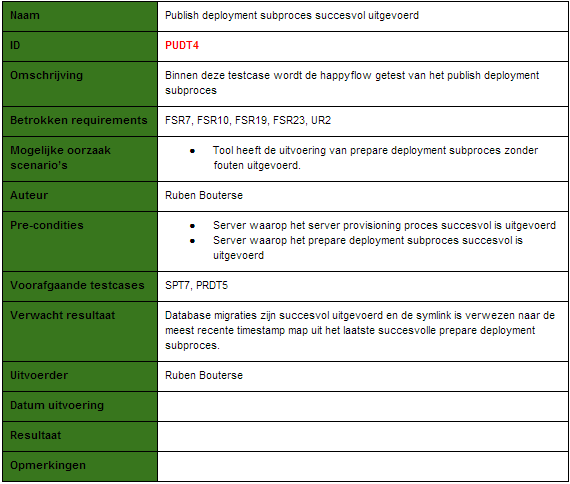
### 

2.2.2 Publish deployment subproces testcases





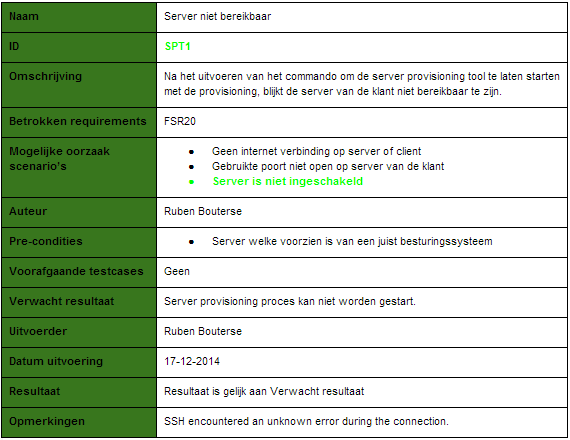




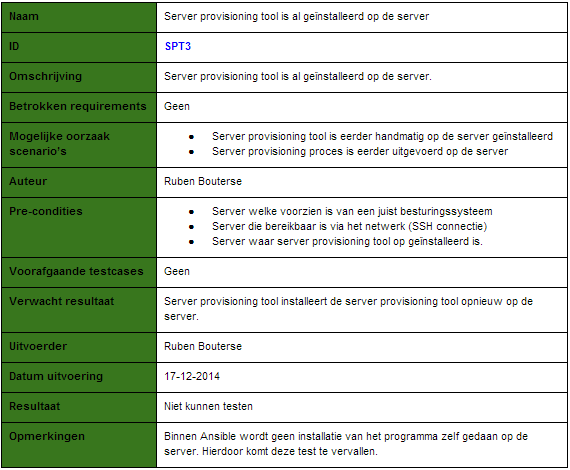
3. Resultaat testcases beste tool uit longlist ( Ansible )

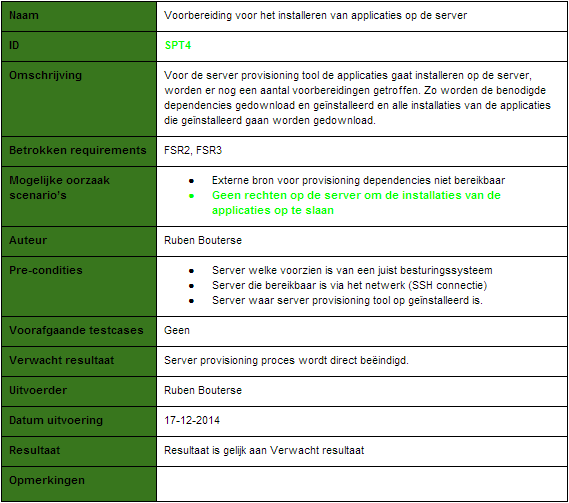
Binnen dit hoofdstuk zijn de resultaten te vinden van de testcases voor de beste tool uit de longlist. De testcases voor server provisioing en software deployment worden uitgevoerd door Ansible.

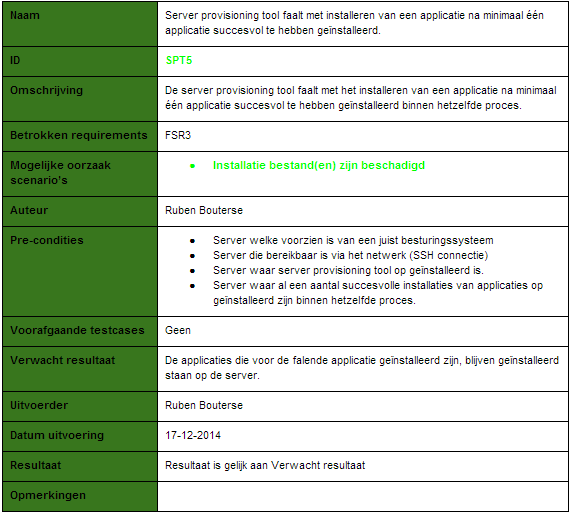
3.1 Server provisioning testcases Ansible

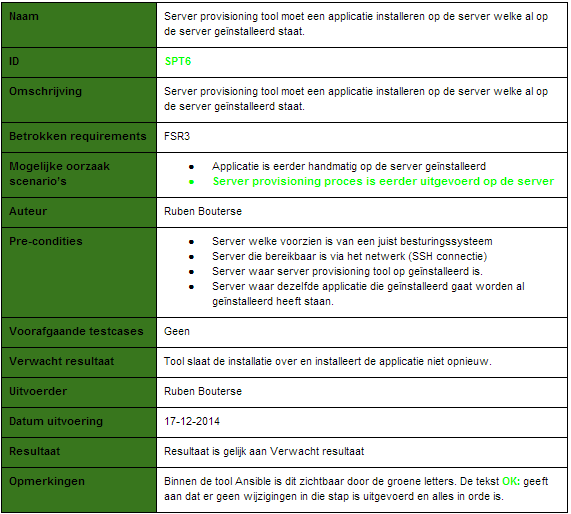


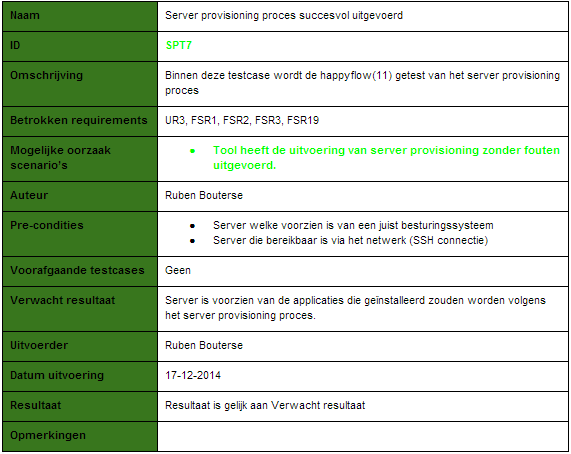








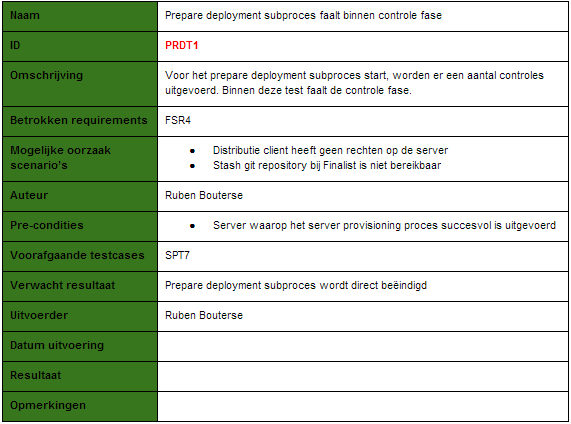


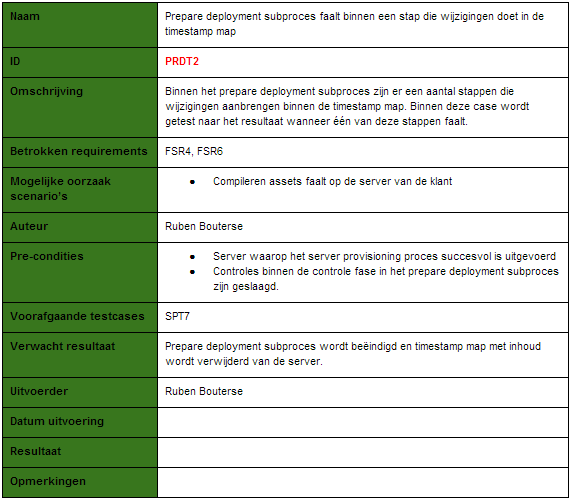


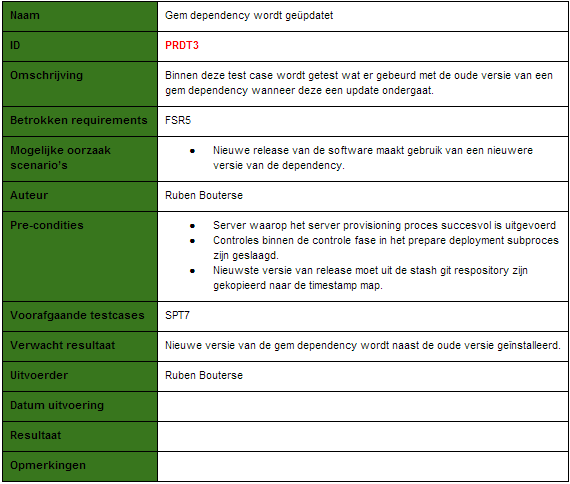
## 

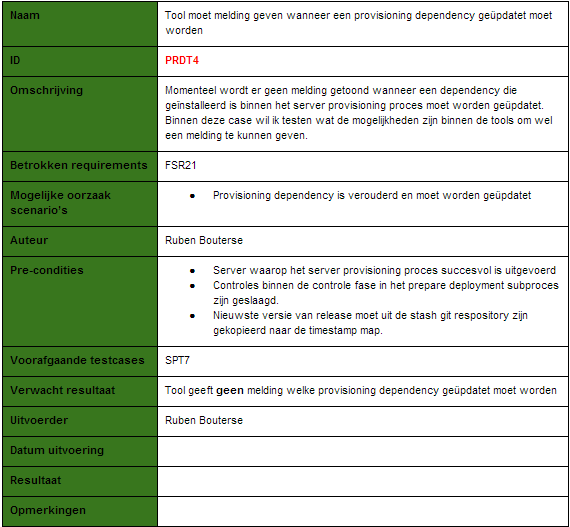
3.2 Software deployment testcases Ansible

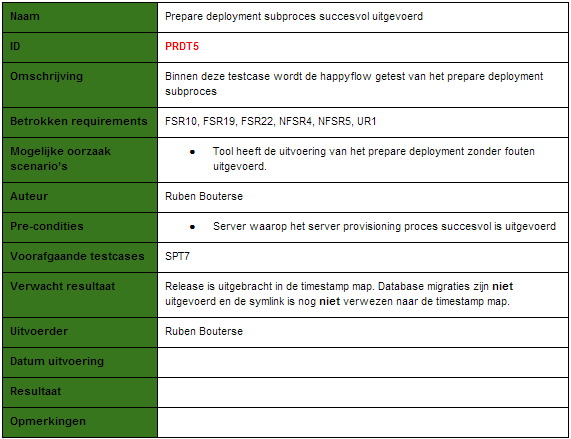
3.2.1 Prepare deployment testcases



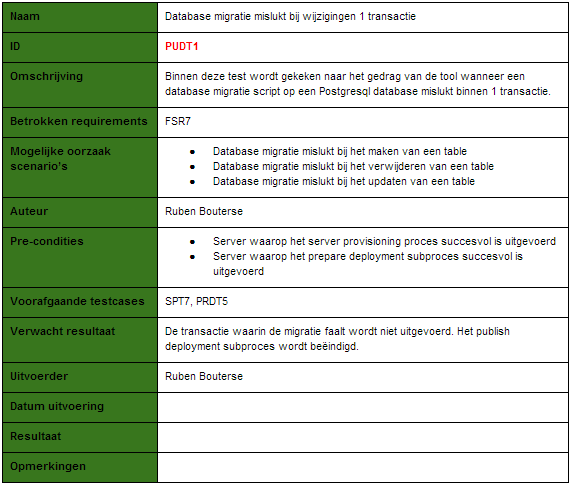


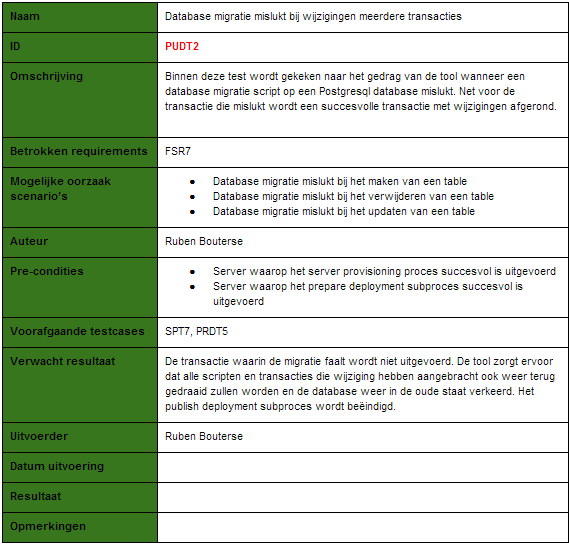


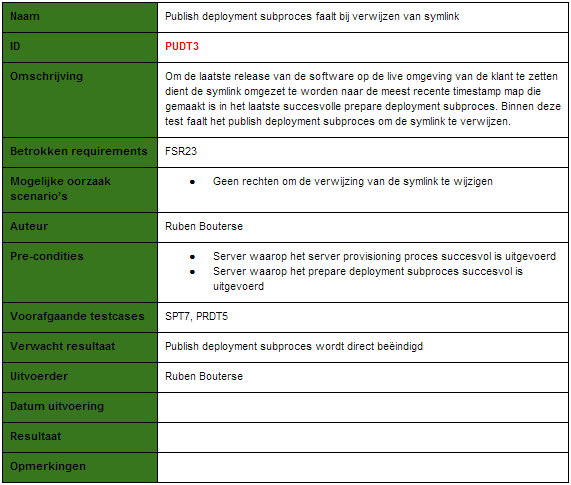


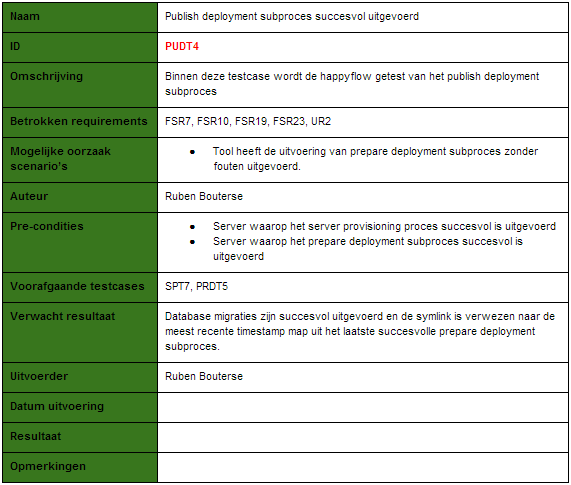


3.2.2 Publish deployment testcases









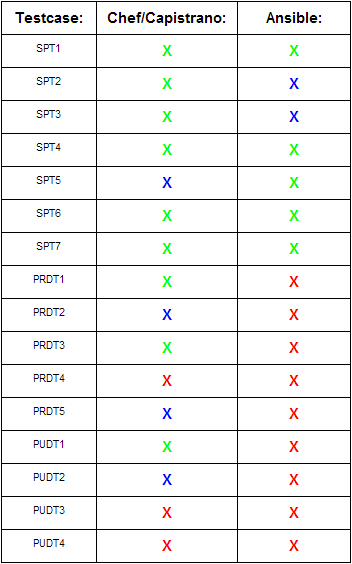
4. Overzicht testcase resultaten

Helaas nam het testen van de tools in praktijk veel tijd in beslag en is het niet gelukt om alle testcases binnen mijn afstudeerperiode uit te voeren. Een overzicht van alle testcases is als volgt:

**X** = Testcase uitgevoerd met resultaat gelijk aan het verwacht resultaat

**X** = Testcase uitgevoerd met resultaat afwijkend dan het verwacht resultaat

**X** = Testcase niet uitgevoerd



5. Vergelijkingen en conclusies

5.1 Server provisioning

We kunnen concluderen dat de testcases die beginnen met SPT (server provisioning testen) uitgevoerd zijn met zowel de tool Chef als de tool Ansible. Om de resultaten van Ansible t.o.v. Chef te vergelijken, wil de afwijkende resultaten doornemen.

*Chef:*

Wanneer we het afwijkende resultaat bekijken van testcase SPT5, valt er te concluderen dat het gaat om een positieve afwijking. Testcase SPT5 kan niet worden uitgevoerd omdat Chef de installatie bestanden al in een eerder stadium van het proces controleert en een foutmelding geeft bij een beschadigde installatie. Hierdoor is het dus onmogelijk dat er minimaal één applicatie succesvol geïnstalleerd is en vervolgens een beschadigde installatie faalt.

*Ansible:*

Wanneer we de afwijkende resultaten bekijken van de testcases SPT2 en SPT3, valt er te concluderen dat het gaat om een positieve afwijkingen. Ansible wordt t.o.v. Chef **niet** geïnstalleerd op de server. Hierdoor zijn de testcases SPT2 en SPT3 niet relevant voor de Tool Ansible.

We kunnen concluderen dat het uitvoeren van server provisioning met zowel Chef als Ansible mogelijk is. Het afwijkende gedrag op de uitgevoerde testcases geven geen negatief gevoel over de gebruikte tool. Omdat beide tools in staat zijn om server provisioning uit te voeren zonder extreme positieve of negatieve afwijkingen, concludeer ik dat Chef niet vervangen hoeft te worden met Ansible.

5.2 Software deployment

We kunnen concluderen dat de testcases voor software deployment helaas niet uitgevoerd zijn met de tool Ansible. Hierdoor is het niet mogelijk een conclusie te kunnen trekken tussen beide software deployemnt tools. Omdat de resultaten van Capistrano wel duidelijk zijn, wil ik de afwijkende resultaten en niet uitgevoerde testcases doornemen.

*Capistrano:*

Wanneer we het afwijkende resultaat bekijken van testcase PRDT2, valt er te concluderen dat het gaat om een negatieve afwijking. Omdat de timestamp map niet verwijderd wordt bij een foutieve poging van software deployment, verliest de server een goedwerkende backup. Het is namelijk zo dat er een aantal releases behouden blijven op de server. Wanneer de mislukte timestamp map blijft bestaan en er vervolgens nog een aantal foutieve pogingen gedaan worden en deze timestamp mappen ook behouden blijven, zullen de voorgaande goede timestamp mappen verwijderd worden op de server. Kortom, de afwijking zorgt voor vervuiling en verwijdering van voorgaande versies.

Wanneer we het afwijkende resultaat bekijken van testcase PRDT5, valt er te concluderen dat het gaat om een negatieve afwijking. De testcase bevestigd dat knelpunt KN6 optreed binnen de software deployment met Capistrano.  Daarnaast worden de database migraties ook uitgevoerd terwijl dit binnen het prepare publish subproces moet plaatsvinden.

Wanneer we het afwijkende resultaat bekijken van testcase PUDT2, valt er te concluderen dat het gaat om een negatieve afwijking. Omdat de database niet in de oude staat terug wordt gebracht kunnen er problemen met de applicatie ontstaan door missende gegevens.

Omdat testase PRDT4 niet uitgevoerd kon worden, kunnen we bevestigen dat knelpunt KN4 optreed binnen de software deployment met Capistrano.

Omdat testase PUDT3 niet uitgevoerd kon worden, kunnen we bevestigen dat knelpunt KN6 optreed binnen de software deployment met Capistrano.

Omdat testase PUDT4 niet uitgevoerd kon worden, kunnen we bevestigen dat knelpunt KN6 optreed binnen de software deployment met Capistrano.

### 10.3.8 Adviesrapport

**Adviesrapport**

Onderzoek software distributie

Welke software distributie tool sluit het beste aan op de software distributie binnen het Webapps team?



Auteur: Ruben Bouterse Eerste docent: H.P.Weenink

Opleiding: Informatica Tweede docent: G.M.Tuk

Bedrijf: Finalist Begeleider: R.Schellhorn

Opdrachtgever: M. van Breukelen

Datum: 02-01-2015

Versie: 1.0

Inhoudsopgave

[Inhoudsopgave](#h.iqk84w22r8k4)

[Inleiding](#h.7pl5qah54xl4)

[1. Conclusies deelvragen](#h.86qoepvmgu91)

[1.1 Conclusie deelvraag 1](#h.lr641e9uebbe)

[1.2 Conclusie deelvraag 2](#h.s8ujhu6cxgi8)

[1.3 Conclusie deelvraag 3](#h.9vh6tamgqqcl)

[1.4 Conclusie deelvraag 4](#h.5exs4xjub4k5)

[1.5 Eindconclusie](#h.dm092ok3hlka)

[2. Advies](#h.p78rczogs70r)

Inleiding

Binnen dit document is het advies te lezen die opgesteld is voor het onderzoek “Software distributie”. Het advies is opgesteld op basis van de behaalde resultaten en conclusies uit het onderzoeksrapport.

Binnen dit document worden als eerst de conclusies uit het onderzoeksrapport herhaald. Vervolgens beschrijf ik het advies dat opgesteld is voor het Webapps team van Finalist.

# 

1. Conclusies deelvragen

1.1 Conclusie deelvraag 1

De software distributie van het Webapps team wordt uitgevoerd in twee processen. Om deze twee processen uit te kunnen voeren, dient de distributie client voorzien te worden van een aantal software programma’s.

De twee processen die uitgevoerd worden zijn als volgt:

*Server provisioning:*  
Het server provisioning proces wordt uitgevoerd in drie verschillende activiteiten namelijk:

* *Verzoek indienen*

Binnen deze activiteit wordt er een verzoek ingediend bij de systeembeheerder voor het aanmaken van een nieuwe server.

* *Installatie server*  
  Binnen deze activiteit wordt de server voorzien van een besturingssysteem.
* *Installatie benodigde applicaties*  
  Binnen deze activiteit wordt de server voorzien van de provisioning dependencies voor de applicatie die in het software deployment proces zal worden gedistribueerd. Het installeren van de provisioning dependencies wordt met behulp van de tool “Chef” uitgevoerd.

Server provisioning wordt eenmalig uitgevoerd. De reden waarom dit proces eenmalig uitgevoerd wordt komt omdat de server eenmalig ingericht moet worden met de juiste provisioning dependencies.

*Software deployment:*Het software deployment proces wordt uitgevoerd in twee verschillende activiteiten namelijk:

* *Installatie release*

Binnen deze activiteit wordt de nieuwste release van de applicatie gedistribueerd naar de server van de klant. Het distribueren van de nieuwste release wordt met behulp van de tool “Capistrano V3.0” uitgevoerd. Capistrano V3.0 maakt gebruik van 4 “stages” waarin stappen worden uitgevoerd om de software te distribueren. Deze stages worden in een vaste uitvoeringsvolgorde uitgevoerd en zijn niet aanpasbaar.

* *Testen door klant*  
  Binnen deze activiteit worden testen uitgevoerd door de klant.

Software deployment wordt herhalend uitgevoerd. De reden waarom dit proces herhalend uitgevoerd wordt komt omdat er steeds nieuwe releases gedistribueerd worden op de server.

1.2 Conclusie deelvraag 2

De knelpunten die binnen de software distributie van het Webapps team optreden zijn als volgt:

|  |  |  |
| --- | --- | --- |
| **ID:** | **Knelpunt:** | **Prioriteit:** |
| KN1 | Het downloaden van gem dependencies wordt gedaan van een externe bron. | Middel |
| KN2 | Het downloaden van gem dependencies valt binnen een vaste uitvoeringsvolgorde van de software deployment tool. | Hoog |
| KN3 | Het compileren van assets valt binnen een vaste uitvoeringsvolgorde van de software deployment tool. | Hoog |
| KN4 | Wanneer een provisioning dependency geüpdatet moet worden, verschijnt er geen foutmelding. | Laag |
| KN5 | Het server provisioning en software deployment proces gebruiken verschillende tools. | Middel |
| KN6 | Het verwijzen van de symlink naar de meest recente software release, valt binnen een vaste uitvoeringsvolgorde van de software deployment tool | Hoog |

Om deze knelpunten op te lossen zijn er drie concepten opgesteld voor een ideaal toekomstbeeld van het Webapps team. De drie concepten zijn als volgt:

*Prepare deployment*Binnen het prepare deployment concept wordt gesproken over het opdelen van het software deployment proces in twee subprocessen. Door de opdeling van twee subprocessen is het mogelijk om een release vroegtijdig te distribueren naar de server van de klant en zo mogelijke complicaties op tijd op te kunnen lossen. Echter is de release nog niet doorgevoerd op de live omgeving van de klant.

Binnen het tweede subproces wordt enkel alleen nog de database migraties en de symlink verwezen naar de laatst gedistribueerde release. Hierdoor wordt de nieuwste release in de live omgeving doorgevoerd.

Met het prepare deployment concept worden de volgende knelpunten opgelost:

|  |  |  |
| --- | --- | --- |
| **ID:** | **Knelpunt:** | **Prioriteit:** |
| KN1 | Het downloaden van gem dependencies wordt gedaan van een externe bron. | Middel |
| KN2 | Het downloaden van gem dependencies valt binnen een vaste uitvoeringsvolgorde van de software deployment tool. | Hoog |
| KN3 | Het compileren van assets valt binnen een vaste uitvoeringsvolgorde van de software deployment tool. | Hoog |
| KN6 | Het verwijzen van de symlink naar de meest recente software release, valt binnen een vaste uitvoeringsvolgorde van de software deployment tool | Hoog |

*Geen foutmelding bij het noodzakelijk updaten van provisioning dependencies*Binnen dit concept zijn twee opties opgesteld namelijk:

* *Optie1: server provisioning proces starten voor iedere uitvoering van het software deployment proces*

Wanneer deze optie van toepassing is, zal voor het uitbrengen van een release elke keer het server provisioning proces worden gestart. Provisioning dependencies die geüpdatet moeten worden, zullen worden geïnstalleerd. Het opleveren van een release zal vors toenemen in tijdsduur omdat bij iedere release het server provisioning proces gestart wordt. Echter zal het server provisioning proces niet zo lang duren als de eerste keer dat server provisioning is uitgevoerd.

* *Optie2: tonen van een melding wanneer een provisioning dependency bijgewerkt moet worden*Wanneer deze optie van toepassing is zal dit niet toenemen in de tijdsduur van de oplevering van een release. Deze optie zorgt ervoor dat een werknemer direct weet dat er een update vereist is en om welke provisioning dependency het gaat. Het updaten van de provisioning dependency zal niet veel tijd in beslag nemen.

Met dit concept wordt het volgende knelpunt opgelost:

|  |  |  |
| --- | --- | --- |
| **ID:** | **Knelpunt:** | **Prioriteit:** |
| KN4 | Wanneer een provisioning dependency geüpdatet moet worden, verschijnt er geen foutmelding. | Laag |

*Server provisioning en software deployment processen realiseerbaar binnen één software distributie tool*Binnen dit concept zijn twee opties opgesteld namelijk:

* *Optie1: tool zoeken die zowel server provisioning als software deployment kan realiseren*

Binnen deze optie zal er één tool worden gebruikt voor zowel het server provisioning als het software deployment proces. Dit brengt als voordeel met zich mee dat een medewerker nog maar één tool hoeft te begrijpen om twee processen uit te kunnen voeren.

* *Optie 2: tools server provisioning en software deployment gescheiden houden*

Binnen deze optie zullen beide processen gebruik maken van een eigen tool. In de huidige situatie is dit ook gerealiseerd met Chef als tool voor server provisioning en Capistrano als tool voor software deployment.

Met dit concept wordt het volgende knelpunt opgelost:

|  |  |  |
| --- | --- | --- |
| **ID:** | **Knelpunt:** | **Prioriteit:** |
| KN5 | Het server provisioning en software deployment proces gebruiken verschillende tools. | Middel |

1.3 Conclusie deelvraag 3

Op basis van goede criteria, zijn er drie tools opgesteld binnen de longlist namelijk:

*Ansible:*

Ansible is momenteel een hype onder de automatisering tools. Ook binnen Finalist is de naam Ansible al meerdere keren gevallen bij verschillende teams en zelfs recent toegepast binnen het Netwerk VSP team. Ansible wordt momenteel gebruikt door vele grote bekende bedrijven en heeft al meer dan 1miljoen downloads.

*Mina:*

Mina is een tool die ik gevonden heb door te zoeken op Google. Mina lijkt de opvolger van Capistrano en geeft aan veel sneller software deployment te kunnen realiseren door het gebruik van één SSH connectie. Omdat Mina in de toepassing lijkt op Capistrano zou het leren van deze tool vrij eenvoudig kunnen zijn.

*Jenkins:*

Jenkins is een tool die ik het meeste ben tegen gekomen binnen de interviews met de zeven teams. Jenkins biedt een flexibele werkomgeving omdat de scripten helemaal zelf uitgeschreven moeten worden. Dit biedt de mogelijkheid om een taak (Job) helemaal te laten functioneren hoe je zelf wilt.

Om de definitieve keuze te maken welke tool uit de longlist het meest geschikt leek voor het Webapps team, heb ik samen in overleg met mijn begeleider de keuze gebaseerd op de volgende punten:

* Resultaten uit literatuuronderzoek die criteria punten bevestigen
* Resultaten voordelen
* Resultaten nadelen

Uit de resultaten bleek dat Ansible de tool is die waarschijnlijk het beste aansluit op de software distributie binnen het Webapps team, omdat deze aan de meeste criteria punten voldoet, De beste voordelen en de minst erge nadelen heeft. Ansible is dus de meest interessante tool om te testen in praktijk.

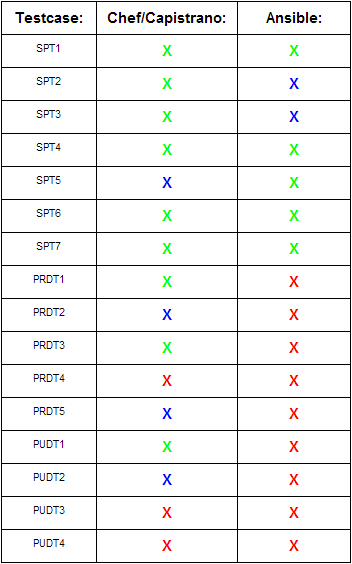
1.4 Conclusie deelvraag 4

Helaas nam het testen van de tools in praktijk veel tijd in beslag en is het niet gelukt om alle testcases binnen mijn afstudeerperiode uit te voeren. Het is daarom niet mogelijk om een volledige conclusie voor deze deelvraag op te stellen. Een overzicht met de resultaten is als volgt:

**X** = Testcase uitgevoerd met resultaat gelijk aan het verwacht resultaat

**X** = Testcase uitgevoerd met resultaat afwijkend dan het verwacht resultaat

**X** = Testcase niet uitgevoerd



We kunnen concluderen dat de testcases die beginnen met SPT (server provisioning testen) uitgevoerd zijn met zowel de tool Chef als de tool Ansible. De afwijkende resultaten van beide tools zijn als volgt:

*Chef:*

Wanneer we het afwijkende resultaat bekijken van testcase SPT5, valt er te concluderen dat het gaat om een positieve afwijking. Testcase SPT5 kan niet worden uitgevoerd omdat Chef de installatie bestanden al in een eerder stadium van het proces controleert en een foutmelding geeft bij een beschadigde installatie. Hierdoor is het dus onmogelijk dat er minimaal één applicatie succesvol geïnstalleerd is en vervolgens een beschadigde installatie faalt.

*Ansible:*

Wanneer we de afwijkende resultaten bekijken van de testcases SPT2 en SPT3, valt er te concluderen dat het gaat om positieve afwijkingen. Ansible wordt t.o.v. Chef **niet** geïnstalleerd op de server. Hierdoor zijn de testcases SPT2 en SPT3 niet relevant voor de Tool Ansible.

We kunnen concluderen dat het uitvoeren van server provisioning met zowel Chef als Ansible mogelijk is. Het afwijkende gedrag op de uitgevoerde testcases geven geen negatief gevoel over de gebruikte tool. Omdat beide tools in staat zijn om server provisioning uit te voeren zonder extreme positieve of negatieve afwijkingen, concludeer ik dat Chef niet vervangen hoeft te worden met Ansible.

Helaas zijn de testcases voor software deployment niet uitgevoerd met de tool Ansible. Hierdoor is het niet mogelijk een conclusie te kunnen trekken tussen beide software deployment tools.

1.5 Eindconclusie

Nu de resultaten en conclusies van de deelvragen grotendeels beantwoordt zijn, is het tijd om de hoofdvraag te beantwoorden. Een herhaling van de hoofdvraag is:

*“Welke software distributie tool sluit het beste aan op de software distributie binnen het Webapps team?”*

In deelvraag 1 hebben we kunnen concluderen dat de software distributie binnen het Webapps team bestaat uit twee processen die bestaan uit verschillende activiteiten namelijk:

* Server provisioning proces
  + *Verzoek indienen*
  + *Installatie server*
  + *Installatie benodigde installaties*
* Software deployment proces
  + *Installatie release*
  + *Testen door klant*

Omdat de processen in deelvraag 1 in detail uitgewerkt zijn, kunnen we een klein deel van de hoofdvraag verduidelijken namelijk:

*“Welke software distributie tool sluit het beste aan op de* ***software distributie*** *binnen* ***het Webapps team****?”*

Wat we verstaan onder de term “software distributie” binnen het onderzoek.  
Wat we verstaan onder de term “het Webapps team” binnen het onderzoek.

In deelvraag 2 hebben we kunnen concluderen dat er zich een aantal knelpunten voordoen in de huidige manier van software distributie binnen het Webapps team. Voor deze knelpunten zijn er opties beschreven voor een ideaal toekomstbeeld voor het Webapps team. Op basis van de verzamelde gegevens heb ik requirements kunnen opstellen. Door opstellen van de requirements, kunnen we een klein deel van de hoofdvraag verduidelijken namelijk:

*“Welke* ***software distributie tool*** *sluit het beste aan op de software distributie binnen het Webapps team?”*

Wat we verstaan onder de term “software distributie tool” binnen het onderzoek.

In deelvraag 3 hebben we kunnen concluderen dat Ansible de tool is die waarschijnlijk het beste aansluit op de software distributie binnen het Webapps team, omdat deze aan de meeste criteria punten voldoet, De beste voordelen en de minst erge nadelen heeft. Door deze conclusie kunnen we een klein deel van de hoofdvraag beantwoorden namelijk:

*“****Welke******software distributie tool*** *sluit het beste aan op de software distributie binnen het Webapps team?”*

Welke software distributie tool sluit **mogelijk** het beste aan op de software distributie binnen het Webapps team.

Helaas kunnen we geen volledige conclusie uit deelvraag 4 ophalen omdat niet alle resultaten behaald zijn. Hierdoor is het helaas niet mogelijk om een definitief antwoord op de hoofdvraag te geven.

2. Advies

Omdat het niet gelukt is om binnen mijn afstudeertijd alle gewenste resultaten te behalen,was het lastig om een eindconclusie te formuleren.

De resultaten over de huidige manier van werken hebben mij duidelijk gemaakt dat het vervangen van de huidige tools verbeteringen op kunnen leveren. Ansible lijkt hiervoor een geschikte kandidaat te zijn.

De praktijkgerelateerde werkzaamheden met Ansible heb ik als zeer nuttig en plezier ondervonden. Zo heb ik ook vertrouwen dat Ansible het ideaal toekomstbeeld van het Webapps team kan realiseren en zo de knelpunten die binnen het onderzoek opgesteld zijn op kan lossen.

Omdat de missende resultaten enorm van belang zijn om een goede beslissing te kunnen nemen of Ansible of de huidige tools het meest geschikt zijn voor het Webapps team, is het mijn advies om de missende resultaten te verzamelen. Zolang deze resultaten niet bekend zijn, blijven Chef en Capistrano de meest geschikte tools voor de software distributie van het Webapps team.

### 10.3.9 Afstudeerplan

**Afstudeerplan**

**Informatie afstudeerder en gastbedrijf (***structuur niet wijzigen***)**

**Afstudeerblok**: 2014-1.2 (start uiterlijk 12 mei 2014)

**Startdatum uitvoering afstudeeropdracht**:

**Inleverdatum afstudeerdossier volgens jaarrooster**: 6 oktober 2014

**Studentnummer**: 11020482

**Achternaam**: dhr. Bouterse

**Voorletters**: R.A

**Roepnaam**: Ruben

**Adres**: Weegbreestraat 1

**Postcode**: 2153 EA

**Woonplaats**: Nieuw-vennep

**Telefoonnummer**: 0655951672

**Mobiel nummer**: 0655951672

**Privé emailadres**: R.A.Bouterse@student.hhs.nl

**Opleiding**:

**Locatie**: Den Haag

**Variant**: voltijd

**Naam studieloopbaanbegeleider**: B.Derks

**Naam begeleidend examinator**: H.P.Weenink

**Naam tweede examinator**: G.M.Tuk

**Naam bedrijf**: Finalist

**Afdeling bedrijf**:

**Bezoekadres bedrijf**: Stationsplein 45, A4.205 ( Etage 4 )

**Postcode bezoekadres**: 3013 AK

**Postbusnummer**: 1354

**Postcode postbusnummer**: 3000 BJ

**Plaats**: Rotterdam

**Telefoon bedrijf**: 0882170800

**Telefax bedrijf**: +31 (0) 102809620

**Internetsite bedrijf**: http://www.finalist.nl

**Achternaam opdrachtgever**: van Breukelen  (dhr)

**Voorletters opdrachtgever**: M.

**Titulatuur opdrachtgever**: ir.

**Functie opdrachtgever**: Directeur interne it-zaken

**Doorkiesnummer opdrachtgever**: 0882170806

**Email opdrachtgever**: martijn@finalist.nl

**Achternaam bedrijfsmentor**: Schellhorn (dhr)

**Voorletters bedrijfsmentor**: R.

**Titulatuur bedrijfsmentor**: ir.

**Functie bedrijfsmentor**: Software engineer

**Doorkiesnummer bedrijfsmentor**: 0882170850

**Email bedrijfsmentor**: rob.schellhorn@finalist.nl

**Titel afstudeeropdracht**:

* Uitvoeren van onderzoek naar geschiktheid server provisioning programma voor de applicaties bij Finalist.

**Opdrachtomschrijving**

1. **Bedrijf**

Finalist is een organisatie die sinds 1988 actief is met het verzinnen, ontwerpen en bouwen van open IT oplossingen voor hun klanten. Momenteel beschikt Finalist over drie kantoren welke in Rotterdam, Eindhoven en Maarssen zijn gevestigd. Binnen deze drie organisaties werken +/- 80 werknemers. Binnen vestiging Rotterdam zijn er 40 werknemers werkzaam.

Met hun passie voor softwareontwikkeling en beheersing van de nieuwste technologieën, bouwen zij snel en zorgvuldig aan hun opdrachten. Het inzetten van Open source technologieën staat bij Finalist centraal.

Finalist streeft ernaar om met hun open IT oplossingen organisaties intenser met elkaar samen te laten werken met hun belangengroepen. Doormiddel van advies, projecten en beheer realiseert Finalist intelligente websites, maatwerk applicaties en integratieoplossingen. De doelgroepen van Finalist zijn daarom primaire organisaties waarvoor een uistekende informatievoorziening en intensieve samenwerking belangrijk zijn.

1. **Probleemstelling**

Om de installatie van een softwaresysteem geautomatiseerd te realiseren op een server, maakt Finalist gebruik van het programma "Chef". Het geautomatiseerd installeren van de software brengt echter een aantal problemen en beperkingen met zich mee. Problemen en beperkingen die momenteel spelen zijn:

* Per klant dient Finalist imperatieve scripts te schrijven voor het programma Chef. Deze worden snel groot en onoverzichtelijk ( moeilijk en veel werk om te onderhouden ).
* De servers die zich op klant locatie bevinden zijn niet altijd goed geconfigureerd door hun systeembeheerders. Zo komt het voor dat bepaalde poorten niet geopend zijn waardoor Finalist niet de juiste verbindingen tot stand kan brengen (bijv OpenSSL).
* Finalist ontwikkelt applicaties met gebruik van open source programmeertalen ( Ruby on rails, Php, Java ). Momenteel worden alleen de applicaties die geprogrammeerd worden in Ruby on rails geïnstalleerd via Chef. Finalist wil in de toekomst al hun applicaties kunnen installeren via server provisioning.

1. **Doelstelling van de afstudeeropdracht**

Door de problemen en beperkingen die momenteel optreden, wil Finalist onderzoek doen naar de geschiktheid van server provisioning programma’s die mogelijke oplossingen bieden en het configuration management van Finalist naar anno 2014 zal verbeteren.

Finalist heeft aangegeven dat er momenteel nog geen eerdere onderzoeken zijn geweest naar het verbeteren van de configuratie management binnen Finalist.

Met de resultaten van het onderzoek wenst Finalist voor de einddatum van 6 Oktober 2014 een duidelijk opgesteld adviesrapport te ontvangen. Met dit advies wil Finalist eenvoudig een besluit kunnen nemen of de realisatie van een ander server provisioning programma het configuration management binnen Finalist optimaliseert.

1. **Resultaat**

Het resultaat na het afronden van de afstudeeropdracht zal een advies opleveren. Dit zal echter na de afronding van mijn afstudeeropdracht niet de situatie binnen Finalist veranderen. Het resultaat zal namelijk een duidelijk opgesteld advies opleveren wat eventueel kan leiden tot een verandering van de situatie binnen Finalist.

Wanneer het advies aangeeft dat er een server provisioning programma geschikt genoeg blijkt te zijn en Finalist besluit om deze te realiseren, zal dit afhankelijk van de onderzoeksresultaten kunnen leiden tot de volgende verbeteringen:

* Verlaging van installatieduur softwaresystemen
* Vereenvoudiging van installatie softwaresystemen
* Vereenvoudiging van software updates
* Direct toegang tot servers van alle klanten
* Alle ontwikkelde applicaties installeren via server provisioning
* Geteste scripts

1. **Uit te voeren werkzaamheden, inclusief een globale fasering, mijlpalen en bijbehorende activiteiten**

|  |  |  |  |
| --- | --- | --- | --- |
| Nummer: | Activiteit | Omschrijving | Aantal dagen nodig |
| 1 | Opstellen onderzoeksplan | Het beschrijven van de aanleiding, probleemstelling en doelstelling van het onderzoek. Binnen het onderzoeksplan zal de hoofdvraag en de deelvragen beschreven worden en de aanpak van het onderzoek. | 2 dagen |
| 2 | Literatuuronderzoek | Uitzoeken wat er zoal geschreven is over Chef en andere server provisioning programma's en welke resultaten bestaande onderzoeken hebben opgeleverd. | 6 dagen |
| 3 | Intern onderzoek | Vaststellen welke problemen er ondervonden worden met het huidige programma Chef. Vaststellen welke resultaten er zijn van eventuele eerdere interne onderzoeken. | 5 dagen |
| 4 | Uitvoeren kwalitatief onderzoek | Binnen het kwalitatief onderzoek worden de gegevens die verzameld zijn ( uit o.a Literatuuronderzoek, Intern onderzoek ) geanalyseerd. | 10 dagen |
| 5 | Opstellen requirements | Het vaststellen en documenteren van de requirements waarop het onderzoek gebaseerd moet worden. | 8 dagen |
| 6 | Opstellen shortlist | Wanneer de requirements in kaart zijn gebracht is het noodzakelijk om een sortlist op te stellen met server provisioning programma's die geschikt zijn om te testen. | 2 dagen |
| 7 | Uitvoeren kwantitatief onderzoek | Binnen het kwantitatief onderzoek worden o.a verschillende soorten testen uitgevoerd ( bijvoorbeeld op snelheid ) die cijfermatige resultaten opleveren. | 8 dagen |
| 8 | Plannen en voorbereiden testproces | Binnen deze activiteit zal er een testplan worden opgesteld waarin de punten beschreven worden waarop getest zal worden en wanneer de test zal plaats vinden. Ook zal het voorbereiden van de testomgevingen behoren bij deze activiteit. | 5 dagen |
| 9 | Uitvoeren en rapporteren testproces | Binnen deze activiteit zal het testplan volgens de planning worden uitgevoerd en zullen de resultaten worden gerapporteerd. | 8 dagen |
| 10 | Opstellen onderzoeksrapport | Binnen het onderzoeksrapport worden de resultaten van het onderzoek beschreven. De deelvragen worden beantwoord en vormen samen het antwoord op de hoofdvraag. De inhoud van het onderzoeksrapport zal de onderbouwing van het adviesrapport vormen. | 15 dagen |
| 11 | Opstellen adviesrapport | Binnen het adviesrapport zal op basis van het resultaat uit het onderzoeksrapport beschreven worden, wat het uiteindelijke advies van het onderzoek zal zijn. | 2 dagen |
| 12 | Bouwen afstudeerdossier | Documenteren van uitvoering afstudeeropdracht. Onderbouwen van keuzes en verduidelijking geven. | 15 dagen |

1. **Op te leveren (tussen)producten**

De volgende (tussen)producten zullen worden opgeleverd binnen de afstudeeropdracht:

* Onderzoeksplan
* Onderzoeksrapport
* Requirementsrapport
* Testplan
* Testrapport
* Adviesrapport
* Afstudeerdossier

1. **Te demonstreren competenties en wijze waarop**

|  |  |  |
| --- | --- | --- |
| **Beroepstaak** | **Wijze uitvoering** | **Niveau** |
| 1.4 Uitvoeren analyse door definitie van requirements | Binnen deze beroepstaak dien ik zelfstandig een lijst met requirements op te stellen. De requirements zullen worden opgesteld doormiddel van intern onderzoek ( activiteit 3 ). Binnen dit interne onderzoek zullen diverse stakeholders worden geïnterviewd over het gebruik van Chef en zijn problemen / beperkingen. Om de kwaliteit van de requirements zo goed mogelijk te waarborgen, zal het toepassen van elicitatie bij de stakeholders worden toegepast. Tot slot worden de requirements vastgelegd en zullen deze worden voorzien van een uniek ID en prioriteit volgens de MoSCoW methode ( Activiteit 4 ). | 3 |
| 3.4 Initiëren en plannen van het testproces | Binnen deze beroepstaak dien ik zelfstandig een planning op te stellen voor de testen die uitgevoerd zullen worden. Deze testen zijn inhoudelijk afhankelijke van de opgestelde requirements uit beroepstaak 1.4 en zullen worden uitgevoerd op de server provisioning programma's die gekozen zijn bij het opstellen van de shortlist. | 3 |
| 3.5 Uitvoeren van en rapporteren over het testproces | Binnen deze beroepstaak dien ik zelfstandig de planning uit te voeren die opgesteld is bij beroepstaak 3.4. De testen dienen uitgevoerd te worden op test servers waar Linux Ubuntu Server als besturingssysteem functioneert ( vereist nieuwe kennis Linux ). De verschillende server provisioning programma's uit de opgestelde shortlist zullen ieder op hun eigen manier functioneren ( bijv scripten schrijven in taal x of y) en vergt dus veel nieuwe kennis om de testen met succes te kunnen afronden. De resultaten van de testen zullen worden gerapporteerd in het testrapport. | 4 |
| 4.4 Beheren en distribueren van software | Deze beroepstaak zal worden toegepast binnen beroepstaak 3.5. De testen die worden uitgevoerd zullen het correct distribueren van software als hoofddoel hebben. Daarnaast is momenteel het updaten van software ( beheren ) een probleem bij Finalist en zullen er ook testen plaats vinden die het beheren van software als hoofddoel hebben. Alle resultaten zullen in het onderzoeksplan verwerkt worden en meespelen aan het advies binnen het adviesrapport. | 3 |

### 10.3.10 Voortgangsverslag

Voorgangsverslag

Inmiddels zit ik in week 15 van mijn afstudeerperiode. De rede dat mijn voortgangsrapport zo laat opgesteld is, komt omdat bij mij de ziekte van Pfeiffer is vastgesteld in week 3 van mijn afstudeertraject. Hierdoor heb ik een enorme vertraging opgelopen omdat ik een aantal weken volledig niet of minder heb kunnen werken. Om precies te zijn gaat het om de volgende data:

Volledig niet gewerkt:  
2 Juni t/m 7 Juli

4 uur per dag gewerkt ( 1 dag per week naar kantoor ):  
7 Juli t/m 4 Augustus

6 uur per dag gewerkt ( 2 dagen per week naar kantoor )

4 Augustus tot heden

Ik ben sinds 4 Augustus gestart met 6 uur per dag werken omdat 4 uur per dag werken behoorlijk goed verliep. Ik had nog redelijk weinig rustmomenten overdag en voelde me een stuk beter. 6 uur per dag werken is wel weer een hele stap maar verloopt tot nu toe redelijk goed.

# 

Wat heb ik de afgelopen afstudeerperiode gedaan?

Ik ben begonnen met het verzamelen van gegevens voor mijn vooronderzoek. Dit heb ik gedaan door ongestructureerde interviews af te nemen bij 7 verschillende stakeholders. De interviews hadden als richtlijn om het server provisioning en software distributie proces van ieder team globaal in kaart te kunnen brengen.

Aan de hand van de gegevens die ik verzameld heb met de ongestructureerde interviews, heb ik een vooronderzoek uitgevoerd. Binnen dit vooronderzoek heb ik een basis proces beschreven die alle activiteiten en stappen bevat voor server provisioning en software distributie. De activiteiten en stappen in het basis proces zijn gebaseerd op overeenkomende gegevens uit de 7 interviews.

Na het uitvoeren van het vooronderzoek ben ik gestart met het onderzoeksplan. Binnen het onderzoeksplan heb ik beschreven welke aanleiding, probleemstelling en doelstelling relevant zijn op het onderzoek. Ook heb ik per deelvraag beschreven welke onderzoekmethode(s) ik toepas om gegevens te verzamelen.

Na het opstellen van het onderzoeksplan ben ik begonnen met het uitvoeren van het echte onderzoek. Samen met mijn begeleider hebben we het onderzoek afgebakend tot het Webapps team. Binnen dit team is mijn begeleider werkzaam en kan hij mij betere begeleiding geven. De hoofdvraag van mijn onderzoek is geworden:

“Welke software distributie tool sluit het beste aan op de software distributie binnen het Webapps team?”

Deze hoofdvraag wil ik gaan beantwoorden door 4 deelvragen te beantwoorden. Momenteel ben ik bezig met deelvraag 1 waarin ik het software distributie proces van het Webapps team in detail aan het beschrijven ben ( overige 3 deelvragen zijn terug te vinden in hoofdstuk “Waar ga ik nog aan werken”)..

* **Hoe ziet het software distributie proces van het Webapps team eruit?**

Binnen deze deelvraag wil ik antwoord krijgen over het huidige software distributie proces van het Webapps team. Ik wil het proces in detail uitschrijven om later zo veel mogelijk mogelijke knelpunten te kunnen vaststellen.

Voldoet het werk van afgelopen afstudeerderperiode ?

Mijn begeleider is tevreden met de voortgang van mijn project. Af en toe kijkt mijn begeleider mijn documenten door en vermeld hij commentaar of geeft hij verbeterpunten.

Lig ik op schema met mijn planning?

Op 31 Juli heb ik samen met mijn begeleider de afbakening van mijn project besproken. Mijn begeleider heeft mij toen gevraagd een nieuwe planning te maken. Deze planning is te vinden in mijn werkomgeving ( AfstuderenFinalistRuben/Algemeen/Planning/Planning V3.0 ).

Ik loop helaas een aantal dagen achter op mijn planning. Dit komt omdat het enorm veel tijd heeft gekost om het software distributie en server provisioning proces van het Webapps team in detail te begrijpen omdat hiervoor een heleboel kennis nodig is.

Waar ga ik nog aan werken?

Om mijn project succesvol af te ronden, dien ik de overige deelvragen in mijn onderzoeksrapport te beantwoorden. Deze deelvragen houden het volgende in:

* **Welke knelpunten treden op binnen het software distributie proces van het Webapps team?**

Binnen deze deelvraag wil ik antwoord krijgen over de knelpunten die vastgesteld kunnen worden uit het software distributie proces van het Webapps team. Zodra ik de knelpunten heb vastgeld wil ik de belangrijkste knelpunten selecteren om af te bakenen voor mijn verdere onderzoek. Daarnaast wil ik requirements opstellen waaraan een software distributie tool moet voldoen [Beroepstaak Requirements opstellen].

* **Welke software distributie tools bieden vermoedelijk oplossing voor het huidige probleem van het Webapps team?**

Binnen deze deelvraag wil ik een aantal software distributie tools beschrijven die vermoedelijk de oplossing kunnen bieden voor de afgebakende knelpunten uit de vorige deelvraag. De software distributie tools worden geselecteerd aan de hand van gevonden informatie die betrekking heeft over de geselecteerde knelpunten. De geselecteerde software distributie tool worden opgesteld in een shortlist aan de hand van de knelpunten en requirements uit de vorige deelvraag.

* **Welke resultaten leveren de vermoedelijke software distributie tools op in praktijk?**

Binnen deze deelvraag wil ik de software distributie tools uit de vorige deelvraag testen in praktijk [Beroepstaak Uitvoeren testen]. Door middel van verschillende scenario’s die betrekking hebben tot de geselecteerde knelpunten uit deelvraag 2 te beschrijven [Beroepstaak Opstellen testplan], hoop ik op een duidelijke manier aan te kunnen tonen welke tool de beste mogelijkheden biedt.

Na het beantwoorden van de deelvragen kan ik een conclusie trekken en mijn onderzoek afronden. Op basis van de conclusie kan ik een adviesrapport opstellen voor Finalist B.V.

Tot slot dien ik mijn afstudeerdossier af te ronden voor school.

### 10.3.11 Resultaat bespreking concept afstudeerdossier 60% gesprek

|  |  |  |
| --- | --- | --- |
| **Bespreking concept** | **Tussentijds assessment** | **Eerste beoordeling** |

**Formulier bespreking concept afstudeerdossier**

|  |  |  |  |
| --- | --- | --- | --- |
| **Student:** | Ruben Bouterse | **Studentnummer:** | 11020482 |
| **Datum:** | 8 oktober 2014 |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Tijdens de bespreking is het volgende geconstateerd:** | | **ja** | **nee** |
| A | Het voortgangsverslag is ontvangen | **x** |  |
| B | Het afstudeerdossier is beschikbaar op Blackboard (op Google drive) | **x** |  |
| C | Het afstudeerdossier is opgebouwd conform de richtlijnen |  | **x** |
| D | Het goedgekeurde afstudeerplan is aanwezig | **x** |  |
| E | Het plan van aanpak is aanwezig | **x** |  |
| F | Reeds geleverd commentaar is aanwezig, nvt |  |  |
| G | Het afstudeerdossier geeft voldoende inzicht in de stand van zaken | **x** |  |
| H | De afstudeeropdracht is tot nu toe naar behoren uitgevoerd | **x** |  |

**Verbeterpunten:**

De rode lijn in het verslag kan beter. De gemaakte rapporten moeten genoemd en geciteerd worden.

**Opmerkingen:**

Door een wijziging in de opdracht zijn ook de beroepstaken aangepast.

**Naam begeleider/examinator:** Helene Weenink

**Datum:** 8 oktober 2014

**Dit formulier wordt door de begeleider/examinator digitaal ingevuld en per email naar de student verstuurd met een cc naar de coördinator van ICT & Media @ Work (**[**A.M.Schipper@hhs.nl**](mailto:A.M.Schipper@hhs.nl)**). Het formulier dient door de student te worden opgenomen in het afstudeerdossier.**

### 10.3.12 Resultaat bespreking TTA 80% gesprek

|  |  |  |
| --- | --- | --- |
| **Bespreking concept** | **Tussentijds assessment** | **Eerste beoordeling** |

**Formulier tussentijds assessment**

|  |  |  |  |
| --- | --- | --- | --- |
| **Student:** | Ruben Bouterse | **Studentnummer:** | 11020482 |
| **Datum:** | 24 november 2014 | **tweede TTA:** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Tijdens het tussentijds assessment is het volgende geconstateerd:** | | **ja** | **nee** |
| A | *Het voortgangsverslag is ontvangen* | X |  |
| B | *Het afstudeerdossier is digitaal beschikbaar* | X |  |
| C | *Het afstudeerdossier is opgebouwd conform de richtlijnen* | X |  |
| D | *Het goedgekeurde afstudeerplan is aanwezig* | X |  |
| E | *Het plan van aanpak is aanwezig* | X |  |
| F | *Reeds geleverd commentaar is aanwezig* |  | X |
| G | *Het afstudeerdossier geeft voldoende inzicht in de stand van zaken* | X |  |
| H | *De afstudeeropdracht is tot nu toe naar behoren uitgevoerd* | X |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Aanpak** | **O** | **T** | **V** | **G** |
| *Passend* |  |  | X |  |
| *Theoretisch verantwoord* |  |  | X |  |
| *Samenhang uitvoering beroepstaken* |  |  | X |  |

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Beroepstaken op afgesproken niveau uitgevoerd?** | | | **O** | | **T** | | **V** | | **G** | |
| 1 | | 1.3 Selecteren van standaardsoftware |  | |  | | X | |  | |
| 2 | | 1.4 Uitvoeren analyse door definitie van requirements |  | | X | |  | |  | |
| 3 | | *3.4 Initiëren en plannen van het testproces* |  | |  | | X | |  | |
| 4 | | 4.4 Beheren en distribueren van software |  | |  | | X | |  | |
| **Producten** | | | **O** | | **T** | | **V** | | **G** | |
| *Tussenproducten* | | |  | |  | | X | |  | |
| *Eindproducten* | | | n.v.t. | | | | | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Effectief communiceren** | **O** | **T** | **V** | **G** |
| *Binnen afstudeerbedrijf* | n.v.t. | | | |
| *Afstudeerdossier* |  |  | X |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Reflectie** | **O** | **T** | **V** | **G** |
| *Inzicht in eigen functioneren* |  | X |  |  |
| *Inzicht in eigen leerproces* |  | X |  |  |

**Toelichting per beoordelingscriterium**

|  |
| --- |
| **Aanpak** |
| Onderzoek maakt een belangrijk deel uit van het verslag. De manier waarop dat is opgezet is traceerbaar en gemaakte keuzes worden voldoende onderbouwd. |

|  |
| --- |
| **Beroepstaken op afgesproken niveau uitgevoerd?** |
| **Genoemde ‘knelpunten’ komen niet duidelijk genoeg terug in de Must Haves die geformuleerd zijn.** |

|  |
| --- |
| **Producten** |
| De documenten die ten grondslag liggen aan het werk zijn in orde. Er zou meer ter illustratie terug mogen komen in het verslag. Het eindproduct is onvoldoende beschreven om beoordeeld te kunnen worden. |

|  |
| --- |
| **Effectief communiceren** |
| Taalkundig is het verslag voor verbetering vatbaar, vooral op het punt van werkwoordspelling. Tekstueel is het goed leesbaar. De opbouw en structuur van het verslag zijn goed genoeg, maar zouden beter tot hun recht kunnen komen in de hoofdstukindeling en onderverdeling in subkopjes. |

|  |
| --- |
| **Reflectie** |
| Tijdens het TTA-gesprek is gebleken dat de student in staat is tot een goede reflectie. In het verslag is daar nog nauwelijks sprake van. |

**Advies**

|  |  |
| --- | --- |
| X | Inleveren **(bindend advies)** |
|  | Verlengen **(vrijblijvend advies)** |
|  | Stoppen **(vrijblijvend advies)** |

**Besluit student**

Aankruisen welke beslissing de student heeft genomen (alleen na vrijblijvend advies)

|  |  |
| --- | --- |
|  | **Afstudeerdossier wordt op afgesproken datum ingeleverd** Inleverdatum: |
|  | **Afstudeerperiode wordt verlengd** Inleverdatum: |
|  | **Student stopt met afstudeeropdracht** |

**Naam begeleidend examinator:** H.P. Weenink

**Naam tweede examinator:** G.M. Tuk

**Datum:** 24 november 2014

**Dit formulier wordt door de tweede examinator digitaal ingevuld, waarna de begeleidend examinator het per email verstuurt naar de student met een cc naar de coördinator van ICT & Media @ Work (**[**A.M.Schipper@hhs.nl**](mailto:A.M.Schipper@hhs.nl)**). Het formulier dient door de student te worden opgenomen in het afstudeerdossier.**