KU Leuven

1 Ma Ingenieurswetenschappen:
Computerwetenschappen

Bedrijfservaring: Computerwetenschappen

Stageverslag bedrijfservaring Alcatel-Lucent

Author:
Helena Brekalo

July 30, 2015

Helena Brekalo
1e master Computerwetenschappen (Veilige software)

Gegevens bedrijf:
Alcatel-Lucent
Copernicuslaan 50
2018 Antwerpen

Gegevens stagebegeleider:
Bart Hemmeryckx-Deleersnijder
E-mail: bart.hemmeryckx-deleersnijder@alcatel-lucent.be
GSM-nummer: +32 472 95 00 65
Vast nummer binnen het bedrijf: +32 3 240 8525

Stageperiode:
1 juli - 7 augustus
7 september - 11 september

Abstract

Contents

1 Het bedrijf: Alcatel-Lucent

Alcatel-Lucent is een Frans bedrijf met de hoofdzetel in Frankrijk en vestigingen in Amerika, Azië, Europa (met dus onder andere een vestiging in Antwerpen), het Midden-Oosten
todohoofdletters? en Afrika. Alcatel kent een lange historie van overnames, met de meest recente overname die door Nokia, begin 2015.
De eerste funderingen van Alcatel werden gelegd in 1869, met de opstart van Gray and Barton in Cleveland, Ohio. Een tien jaar later worden ze overgenomen door American Bell, wat in 1925 leidt tot het ontstaan van Bell Telephone Laboratories. In deze "Bell Labs" worden verschillende grote ontwikkelingen verwezenlijkt, waaronder de eerste lange afstandstelevisietransmissie en de uitvinding van de batterij op zonne-energie. In 1984 worden ze overgenomen door Câbles de Lyon en worden ze een Frans bedrijf. In 2006 mergen Alcatel en het Amerikaanse Lucent Technologies, waarop de naam verandert naar Alcatel-Lucent. In 2015 werd Alcatel-Lucent overgenomen door Nokia, wat effectief zal worden in april 2016. Door deze overname zal Nokia het tweede grootste bedrijf in wireless zijn, op Ericsson na. Op deze manier wordt de concurrentie met het Chinese Huawei vergroot op het gebied van draadloze communicatietechnologie.
Alcatel-Lucent heeft verschillende afdelingen, zijnde het *core networking segment*, wat IP routing, transport en platforms omvat en het *access segment*, wat wireless en fixed access omvat, alsook licensing en *managed services*. In Antwerpen wordt er aan al deze segmenten gewerkt; zelf werd ik tewerkgesteld in de fixed access afdeling, namelijk Motive Network Analyzer-Copper (Motive NA-C). Zij staan ervoor in om het gebruik van het kopernetwerk te optimaliseren en fouten te detecteren en verhelpen.

beter schrijven!

nog tekst toevoegen?

2 Stage

2.1 Afdeling: Motive NA

De stage ging door op de afdeling Motive Network Analyzer - Copper (NA-C) bij het team dat ook onder begeleiding staat van mijn stagebegeleider Bart Hemmeryckx-Deleersnijder. Motive NA-C heeft een team in Antwerpen, Chennai en Bangalor, waarbij de development voornamelijk in Antwerpen gebeurt en het testen in Chennai en Bangalor. Er zijn altijd ook enkele testers van Chennai en Bangalor in Antwerpen, om zo de communicatie tussen de teams te verbeteren en de samenwerking te vergemakkelijken.

2.2 Werkmethoden

2.2.1 Scrum

Het team van Bart werkt volgens de scrum-methode, zoals kort was aangehaald tijdens de hoorcollege's van Software-Ontwerp. Scrum is een voorbeeld van een agile werkmethode, waarbij men ervan uitgaat dat de vereisten voor het project kunnen en zullen veranderen gedurende het project. Bij het klassieke watervalmodel stelt men eerst de vereisten op, om deze vervolgens te implementeren,

dit te evalueren en dan te onderhouden. Het probleem hierbij is dat de communicatie tussen de klant en het bedrijf niet optimaal verloopt, gezien de klant nog van gedachte kan veranderen over de requirements die hij wil. Als deze pas in de maintenance fase ontdekt en aangekaart worden, is het heel kostelijk om (grote) veranderingen opnieuw door te voeren.
Bij een agile werkmethode, waar scrum een onderdeel van is, gaat men er van uit dat de klant op voorhand niet perfect kan weten wat hij wil en dat hij dus de requirements zal aanpassen.
De term "scrum" is afkomstig vanuit rugby, waarbij de twee teams vooroverge-bogen tegen elkaar leunen, met de hoofden bij elkaar. De bal wordt dan in deze groep gegooid, waarop ze proberen om de bal als eerste in hun bezit te krijgen. De gelijkenis met de software-ontwikkeling-scrum is dat het team heel nauw samenwerkt, inspeelt op veranderde omstandigheden en zo samen tot een doel komt. De samenwerking uit zich in dagelijkse stand-up meetings, waarbij iedereen vertelt wat hij/zij de dag ervoor heeft gedaan, welke problemen er eventueel zijn opgetreden en of er daar hulp bij nodig is en wat de planning voor de komende dag is. Op deze manier is iedereen op de hoogte van wie waarmee bezig is en kunnen problemen snel opgelost worden. Bij het NA-C team in Antwerpen maakt men hierbij ook gebruik van het agile dashboard van JIRA, waarbij gevisualiseerd wordt wie waarmee bezig is. Indien iemand ergens problemen mee heeft, doet men aan pair-programming, waarbij men bij elkaar gaat zitten en samen nadenken over hoe het desbetreffende probleem opgelost kan worden en dit ook samen uit te werken.

Foto toevoegen van JIRA dashboard

In het NA-C team heeft men sprints van twee weken, wat wil zeggen dat men elke twee weken een evaluatie doet van het werk dat in de afgelopen twee weken verzet is en waarbij eventueel demo's gegeven worden van wat er gerealiseerd is (sprint review). Samen met het einde van de sprint is er de sprint retrospective, waarbij men gaat oplijsten wat er goed en minder goed is gegaan, welke problemen er (binnen het team) waren en hoe deze opgelost kunnen worden. Aan het begin van een sprint is er dan de sprint planning, waarin er bekeken wordt wat er moet gebeuren met welke prioriteit en wie wat gaat doen (heel algemeen). Men maakt hierbij gebruik van user-stories, die een bepaalde taak omschrijven en de subtaken ervan. Het beschrijft wat de gebruiker nodig heeft bij het uitvoeren van zijn taak en bepalen zodus wat er moet ontwikkeld worden om de gebruiker hierbij te helpen. Het omschrijft het "wie", "wat" en "waarom" van deze vereisten op hoog niveau.

2.2.2 Continuous integration

Aangezien er regelmatig releases van software zijn, doet men aan continuous integration, wat inhoudt dat wanneer iemand klaar is met het schrijven van een stuk code en deze commit, alle builds en testen hierrond automatisch zullen runnen zodat fouten in een vroeg stadium kunnen gevonden en opgelost worden. In Antwerpen maakt men gebruik van Jenkins[1], een open-source continuous integration systeem. Jenkins heeft een dashboard dat een overzicht toont van alle builds, maar je kan ook filteren op zelfgemaakte criteria, zodat je enkel de status van bepaalde builds ziet. Het overzicht toont een gekleurde bol om aan te geven of de laatste build geslaagt (groen/blauw) of gefaald is. Als er testen

[1] https://jenkins-ci.org/

gefaald zijn, dan zal de bol geel kleuren.
Er is ook een weerbericht dat toont hoe stabiel de build is, waarbij slecht weer duidt op een instabiele build en goed weer op een satbiele build.
Het probleem met dit dashboard is dat het niet echt overzichtelijk is. De kleur van de bollen trekt wel de aandacht, maar als je wil zien over welke build het gaat, is dit niet leesbaar, tenzij je je vlak voor het scherm bevindt. Het Jenkins dashboard wordt op de werkvloer weergegeven op een grote televisie, maar men heeft niet de neiging hiernaar te kijken tijdens de uren, waardoor gefaalde builds enkel maar zichtbaar worden voor het team wanneer ze zelf Jenkins openen op hun PC of tijdens de stand-up.

foto van toevoegen

-scrum -agile -¿ jenkins: continuous integration

-zelf ook: jira en conlfuence -¿ link ernaartoe! -erg betrokken: standup meetings, sprint meeting, planning meeting, demo (Zelf een moeten geven) - behulpzaam team (onderling, ook voor mij)

Foto invoegen, getrokken op 29072015 1418

2.3 Stage-opdracht

Mijn stage-opdracht bestond erin om Jenkins awareness te creëren aan de hand van een dashboard dat een beter overzicht toont. Als fouten meer in het oog springen, kunnen ze nog sneller verholpen worden en is er meer sprake van continuous integration. Gedurende heel mijn stage werd ik enorm betrokken bij alles wat het team deed; zo deed ik dagelijks mee aan de stand-up meetings en werd ik meegevraagd in de sprint planning, sprint review en de retrospective, om zo een idee te krijgen hoe het er in een bedrijf aan toe gaat. Ook al kon ik niet meteen volgen waar ze het over hadden, het heeft me enorm veel bijgeleerd over hoe deze manier van werken het team sterker kan maken gezien iedereen (bijna) dagelijks updates krijgt over wie waarmee bezig is en de korte evaluatieperiodes ervoor zorgen dat er kort op de bal gespeeld kan worden.

lame zin, maak beter!

Er werd voor gezorgd dat ik mijn eigen project kreeg, waarbij er dan ook tweewekelijkse sprints gepland werden, zodat ik een idee kreeg van hoe het eraan toe gaat.
Mijn eerste sprint bestond uit het maken van een literatuurstudie, op aanraden van Bart, om zo te zien waarop ik moet letten bij het maken van een (goed) dashboard en wat er verbeterd kan worden aan het originele Jenkins dashboard. Eens ik me een idee gevormd had van hoe ik wou dat het dashboard eruit zou zien, heb ik enkele voorstellen opgemaakt en een kleine enquête gemaakt binnen het team, om te kijken welke voorkeuren zij hadden. Hierop wordt dieper ingegaan in sectie 2.3.1.
Vervolgens ben ik begonnen met het implementeren van het dashboard, dit was mijn tweede sprint. Het proces van hoe ik tot het resultaat ben gekomen, is te vinden in sectie 2.3.2.
De derde en laatste sprint bestond uit het voorstellen van de eerste versie van het dashboard en hierover feedback krijgen, om het dashboard vervolgens te verbeteren. Tijdens de eindfase van deze sprint werd het dashboard in gebruik genomen op de Jenkins server van Alcatel-Lucent. Dit wordt besproken in sectie 2.3.3.

2.3.1 Sprint 1: Onderzoek en literatuurstudie

Literatuurstudie Tijdens de eerste sprint heb ik onderzoek gevoerd naar hoe goede dashboards eruit zien en hoe je ze overzichtelijk kan houden. Hierrond heb ik een literatuurstudie gemaakt, die gevonden kan worden in de annex. Uit deze literatuurstudie bleek inderdaad dat het voorziene Jenkins dashboard niet voldeed aan veel van de vereisten, aangezien je niet in één oogopslag de belangrijkste info kan vergaren.

anders verwoorden?

Naast de literatuurstudie heb ik ook gezocht naar tools om dashboards te maken, waarbij ik van een collega op de afdeling de aanbeveling kreeg om Dashing[2] te gebruiken, waarbij hij ook een demo gaf. Na het bekijken van nog enkele andere tools om dashboards te ontwikkelen, heb ik uiteindelijk voor Dashing gekozen, aangezien het enorm veel flexibiliteit biedt (je kan het dashboard helemaal inrichten zoals je wilt), zodat ik het dashboard helemaal zou kunnen maken zoals ik het in gedachten had.

laatste deel zin weg?

deel tussen haakjes weg?

anders verwoorden?

Enquête Na het onderzoek en de literatuurstudie, heb ik een voorstel voor dashboardontwerpen gemaakt en deze voorgelegd aan de mensen bij NA-C. Ik liet hen hierbij een kleine enquête invullen waarbij ze konden kiezen voor verschillende designs en ook opmerkingen geven. Uit die opmerkingen zijn nog nuttige commentaren gekomen, bijvoorbeeld dat het tijdstip van de laatste build ook best getoond kan worden. De enquête en de resultatentabel zijn ook te vinden in de annex.

referentie ernaar

2.3.2 Sprint 2: Ontwikkeling dashboard

Op het einde van de eerste sprint heb ik met Bart een sprint review en retrospective gehouden, om te zien waar ik stond. We hebben deze meeting gecombineerd met de sprint planning, waarbij ik mocht aangeven hoe ik van plan was het dashboard uit te werken en met welke tools. Ik werd hierin heel vrij gelaten, zodat ik zelf kon ontdekken wat me lag en wat minder.

beter verwoorden

Gebruikte tools Dashing is een dashboard framework dat geschreven is in Sinatra (meer hierover in de volgende paragraaf), een open source software web applicatie geschreven in Ruby. De ontwikkeling gebeurde op Linux, waarbij ik gebruik maakte van Ubuntu 14.04. Op deze PC is dan een lokale Jenkinsomgeving opgezet, zodat ik het dashboard kon testen met mijn eigen projectjes. De code schrijven gebeurde in gedit zelf en debuggen werd gedaan via de terminal.

nog zaken toevoegen?

De "tegels" op het dashboard worden omschreven als "widgets" waarbij je verschillende soorten widgets kan hebben. Ik heb me toegespitst op een widget om informatie over Jenkins te tonen. Deze widget kan dan gebruikt worden voor verschillende builds, het is niet de bedoeling om voor elke build een nieuw widget te ontwikkelen (tenzij je andere informatie wilt tonen).

Gebruikte talen Het schrijven van het dashboard werd gedaan in verschillende talen, hieronder een opsomming welke taal wat voorzag .

anders zeggen?

[2] http://dashing.io/

HTML HTML wordt gebruikt om te bepalen hoe de inhoud getoond wordt op de widget, dus welk deel als titel zal getoond worden en waar de elementen op de widget staan. De informatie die getoond wordt, wordt via Batman Bindings[3] uit de JSON data van de Jenkins server geparset.

klopt dit helemaal? Denk het wel.

(S)CSS CSS wordt gebruikt om de tekst uit de HTML-file op te maken (grootte van de tekst, stijl, kleur,...), waarbij SCCS (Sassy CSS) ervoor zorgt dat je ook variabelen kan gebruiken, wat niet mogelijk is in standaard CSS. Op deze manier kan je dan (kleur)code hergebruiken via variabelen.

beter zeggen

Javascript Javascript werd gebruikt in de Batman Bindings, die ervoor zorgt dat de data die van de website geparset wordt, dynamisch gebruikt kan worden in de HTML-files (via data-bind).Op zich is de Batman Bindings-library klaar om te gebruiken, maar om een tijdstip te parsen (dat werd weergegeven als Unix timestamp), heb ik nog enkele lijnen Javascript-code moeten toevoegen, alsook de moment.js-library.

is dit te diep erop ingegaan?

Coffeescript Coffeescript is een taal die transcompileert naar Javascript. De .coffee-file voor het widgetspecifiëert hoe de data, die via de HTML-file opgevraagd wordt, weergegeven wordt. Deze file was van belang om bijvoorbeeld het juiste weerbericht te tonen. Het coffeescript parsetde data die via de Batman Bindings werd opgehaald en geeft aan hoe deze doorgegeven wordt aan de HTML-file. Zonder de .coffee-file zou er enkel *plain text* te zien zijn op de widget.

of de widget?

juist vervoegd?

Ruby Dashing maakt gebruik van Embedded Ruby om te bepalen welke tegels getoond worden op het dashboard en van plain Ruby om de data op te halen en te parsen vanuit de juiste plaats (nl. de Jenkins server) via de JSON-data die werd voortgebracht hierdoor. Ook wordt hier de link gelegd tussen de build jobs die getoond *kunnen* (niet per sé zullen[4]) worden op het dashboard.

JQuery JQuery wordt gebruikt in de Coffeescript file om CSS-gewijs klassen of ID's te gebruiken van elementen waarop je een bepaalde actie wil uitvoeren.

JSON JSON werd gebruikt om de data uit van de Jenkins server te parsen, zodat deze opgevraagd kon worden in de Ruby-file.

Omdat geen van deze talen me helemaal eigen was, heb ik enkele dagen genomen om de basics te leren via Codecademy[5]. Dit was voldoende om me op weg te zetten om zelf verder te experimenteren. Een widget wordt opgesteld door middel van een .HTML-file, een .SCSS-file en een .coffee-file.html, (s)css, coffeescript, Sinatra(ruby), javascript (batman bindings), jquery

dat laatste mss weglaten?

waar dit zetten?

2.3.3 Sprint 3: Feedback en verbetering

vermelden dat demo gedaan tijdens sprint 2 (naar einde toe), zo (positieve) reacties gekregen en over gepraat. tijdens sprint 3 dan demo gegeven op tv,

[3] http://batmanjs.org/
[4] Dat wordt bepaald door de Embedded Ruby file.
[5] https://www.codecademy.com/

bevraging gedaan: wat goed/slecht Derde: feedback krijgen (Demo! zowel voor antwerpen, bangalor als chennai, maar ook binnen het team zelf feedback vragen)

2.4 Resultaat

screenshot tonen

2.5 Initiatieven voor interns

bijeenkomsten: samen lunchen, infosessie over Alcatel, gesprek met Joeri

3 Relatie stage en opleiding

-scrum: swop -pair programming: swop -zelf project plannen en uitwerken, met tussentijdse evaluaties P&O -documentatie: ogp

4 Krititsche reflectie competenties

5 Conclusie

6 Annex