KU Leuven

1 Ma Ingenieurswetenschappen:
Computerwetenschappen

Bedrijfservaring: Computerwetenschappen

Stageverslag bedrijfservaring Alcatel-Lucent

Author:
Helena Brekalo

August 3, 2015

Gegevens student: 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

Samenvatting

Dit verslag beschrijft de opgedane kennis en ervaring tijdens mijn 6 weken durende stage bij Alcatel-Lucent in Antwerpen. Het doel tijdens de stage was om een dashboard te ontwerpen en implementeren voor het continuous integration systeem Jenkins en op deze manier "Jenkins-awareness" te creëren.
Het dashboard is ontwikkeld met Dashing en toont het overzicht van een reeks zelfgekozen builds. De ontwikkeling is gebeurd in Javascript, Coffeescript, SCSS, Ruby en HTML op Linux.
Gedurende de stage werd ik erg betrokken bij alle onderdelen van het scrum-proces, zodat ik me een beeld kon vormen van hoe dit er in het bedrijfsleven aan toe gaat.

Contents

1 Inleiding

De probleemstelling, waarvoor ik tijdens mijn stage een oplossing ontworpen heb, gaat als volgt. Op de afdeling Motive Network-Analyzer Copper (NA-C) bij Alcatel-Lucent in Antwerpen doet men aan continuous integration, waarbij gebruik gemaakt wordt van de tool Jenkins. Deze biedt een dashboard om het overzicht van de build statussen weer te geven, maar de software-ontwikkelaars hebben niet de neiging deze te bekijken op de televisie die aanwezig is. Het doel van mijn stage was om "Jenkins-awareness" te creëren en een dashboard te ontwikkelen dat snel een overzicht toont van de builds waar aandacht aan besteed moeten worden. Op de afdeling NA-C werkt men volgens het agile-principe van scrum, waar ik zelf ook gebruik van maakte. Zo kreeg ik mijn eigen project toegewezen en moest ik tweewekelijkse sprints inplannen en evaluaties uitvoeren.

Om dit te realiseren, heb ik tijdens de eerste sprint een literatuurstudie uitgevoerd, om mij zo een idee te vormen van hoe het dashboard eruit moet zien. Tijdens de tweede sprint is het dashboard ontwikkeld, met behulp van Dashing. Aan het einde van de tweede sprint is deze geëvalueerd door middel van een demonstratie voor de teams in Antwerpen, Chennai en Bangalor. Op basis van de feedback die hierop ontvangen werd, zijn veranderingen doorgevoerd gedurende de derde sprint. In het tweede deel van de derde sprint is het dashboard in gebruik genomen op de Jenkins server van NA-C.

De voornaamste resultaten zijn het dashboard dat succesvol ontwikkeld is, door te werken via de scrum-methode. De talen die ik hierbij heb geleerd zijn Javascript, Coffeescript, (S)CSS, HTML en Ruby. Het dashboard is nu ook voorzien van documentatie, om hergebruik te vergemakkelijken. Doordat ik overal in betrokken werd, heb ik nu ook een idee van hoe grote projecten verlopen in een bedrijfsomgeving.

Dit verslag is als volgt ingedeeld. In Sectie 2 wordt het bedrijf Alcatel-Lucent voorgesteld. De derde sectie behandelt de stage (Sectie 3), waarin de afdeling waar ik gewerkt heb wordt voorgesteld (Subsectie 3.1), er wordt uitgelegd volens welke methoden er wordt gewerkt (Subsectie 3.2). Mijn stageopdracht wordt uitgelegd in Subsectie 3.3. Het resultaat wordt besproken in Subsectie 3.4. Gedurende mijn stage waren er ook initiatieven voor interns, wat besproken wordt in Subsectie 3.5.
Sectie 4 legt de link tussen de stage en mijn opleiding. De kritische reflectie van de competenties die ik trachtte te verbeteren is te vinden in Sectie 5. Sectie 6 bevat het besluit en de appendices zijn te vinden in Appendix A en B.

ref naar andere appendices toevoegen

2 Het bedrijf: Alcatel-Lucent

Alcatel-Lucent is een Frans bedrijf met de hoofdzetel in Frankrijk en vestigingen in Amerika, Azië, het Midden-Oosten, Afrika en Europa, waaronder een vestiging in Antwerpen. Alcatel kent een lange historie van overnames, met de meest recente overname die door Nokia, begin 2015.

is dit ok?

De eerste funderingen van Alcatel werden gelegd in 1869, met de opstart van Gray and Barton in Cleveland, Ohio. Een tien jaar later zijn ze overgenomen

door American Bell, wat in 1925 leidde tot het ontstaan van Bell Telephone Laboratories. In deze "Bell Labs" zijn verschillende grote ontwikkelingen verwezenlijkt, waaronder de eerste lange-afstands televisietransmissie en de uitvinding van de batterij op zonne-energie. In 1984 werd Bell Labs overgenomen door Câbles de Lyon en werden ze een Frans bedrijf. In 2006 zijn Alcatel en het Amerikaanse Lucent Technologies gemerged, waarop de naam veranderd werd 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 het Zweedse Ericsson na. Op deze manier wordt de concurrentie met het Chinese Huawei vergroot op het gebied van draadloze communicatietechnologie.

beter schrijven!

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.

nagaan, infodag was anders verteld

nog tekst toevoegen?

3 Stage

3.1 Afdeling: Motive Network Analyzer-Copper

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.

Motive NA-C staat in voor het analyseren en onderhouden van de koperlijnen. Hun software zorgt ervoor dat de bandbreedte zo optimaal mogelijk gebruikt wordt en verandert dit gebruik ook dynamisch naargelang de beschikbare bandbreedte variëert. Indien er fouten optreden op de lijn, kan hun software aanduiden waar op de lijn deze zich waarschijnlijk bevindt en waaraan de fout mogelijks te wijten is. Een manier om tot een oplossing te komen wordt ook meegegeven.

is dit ok?

3.2 Werkmethoden

3.2.1 Scrum

Het team van Bart werkt volgens de scrum-methode, zoals kort werd 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. Deze werkmethode staat tegenover de "klassieke" watervalmethode, waarbij men eerst de vereisten opstelt, om deze vervolgens te implementeren, dit te evalueren en het

systeem vervolgens 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 en deze meestal niet tijdens het ontwikkelingsproces kunnen doorgevoerd worden. Als deze pas in de onderhoudsfase ontdekt en aangekaart worden, is het heel kostelijk om (grote) veranderingen 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. Hierop wordt dan ingespeeld tijdens het ontwikkelingsproces, door dit niet zoals het watervalmodel te zien, maar te werken in ontwikkelingsfases van korte perioden, die gewoonlijk één tot vier weken duren. Op deze manier kunnen veranderingen in de requirements snel opgepikt en doorgevoerd worden, wat het ontwikkelingsproces minder kostelijk maakt.
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 veranderende omstandigheden en zo samen tot een doel komt. De samenwerking uit zich onder andere in dagelijkse stand-up meetings, die telkens rond hetzelfde uur plaatsvinden, ongeacht of iedereen aanwezig is of niet. Tijdens deze meetings vertelt iedereen wat hij/zij de dag ervoor heeft gedaan, welke problemen er 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 (zie Figuur 1). Indien iemand ergens problemen mee heeft, doet men aan pair-programming, waarbij men bij elkaar gaat zitten en samen nadenkt over hoe het desbetreffende probleem opgelost kan worden en dit ook samen uit te werken.

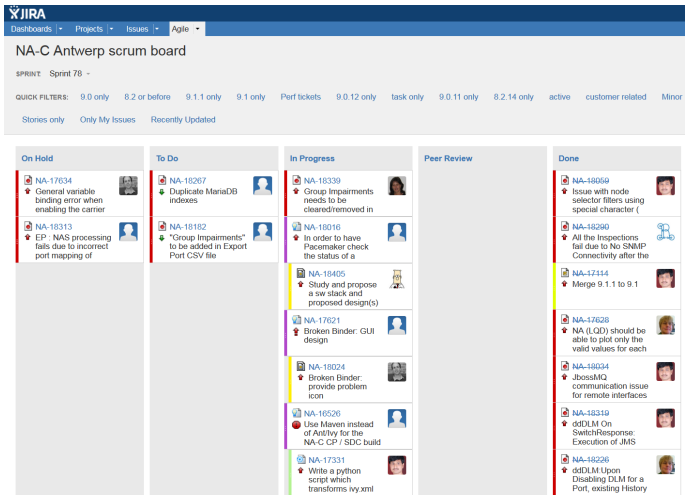

Figure 1: Het JIRA dashboard van het NA-C team in Antwerpen.

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, op een heel algemeen niveau. 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. Een voorbeeld van een user-story is te zien in Figuur 2.

in het begin zetten?

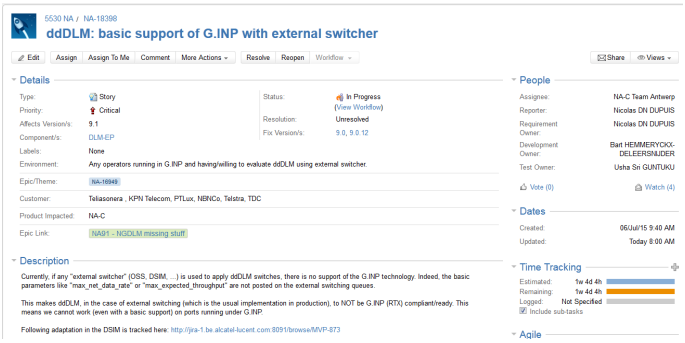

Figure 2: Een van de user stories van het team van NA-C in Antwerpen.

3.2.2 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. Een screenshot van dit dashboard is te zien in Figuur 8. Het overzicht van elke build toont een gekleurde bol om aan te geven of de laatste build geslaagd (groen/blauw) of gefaald is. Als er testen 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 stabiele 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,

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

Figure 3: Het Jenkins dashboard, zoals oorspronkelijk gebruikt op de afdeling NA-C in Antwerpen.

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 meeting. Zoals op Figuur 4 te zien is, moet je zelfs als je voor het dashboard staat goed kijken waar de build staat waarin je geïnteresseerd bent.

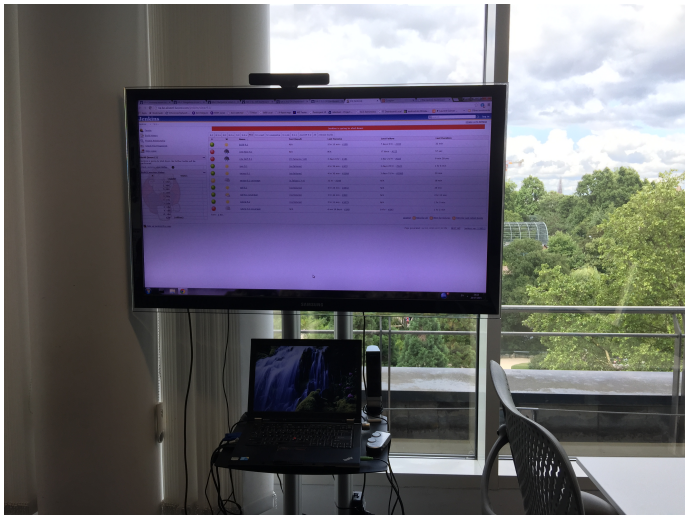

Figure 4: Het originele Jenkins dashboard weergegeven op de televisie op de afdeling van NA-C in Antwerpen.

3.3 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 wordt het continuous integration principe nog meer benut. 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

lame zin, maak beter!

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 hoe de korte evaluatieperiodes ervoor zorgen dat er kort op de bal gespeeld kan worden.
Er werd voor gezorgd dat ik mijn eigen project kreeg, waarbij er dan ook tweewekelijkse sprints gepland werden, zodat ik zelf kon ervaren hoe deze manier van agile werken in elkaar zit. Er werd voor mij een JIRA-pagina aangemaakt, waarop ik mijn vooruitgang kon bijhouden door taken aan te maken die ik wou doen en problemen kon rapporteren. Een screenshot van mijn JIRA-dashboard is te zien in Figuur 5. Mijn eerste sprint bestond uit het maken van een literatuurstudie, te vinden in Appendix A, 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 3.3.1.
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 3.3.2.
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 3.3.3.

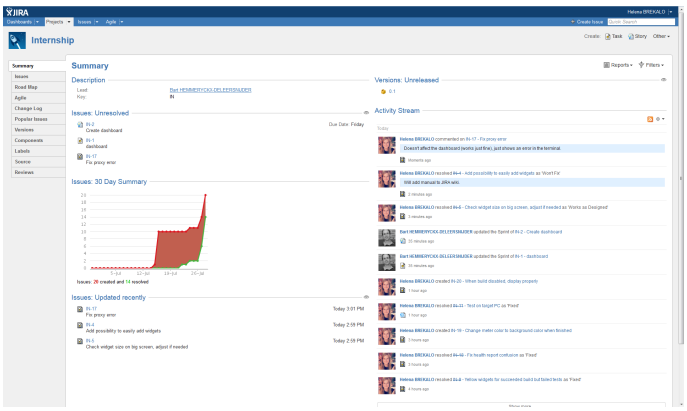

Figure 5: Het JIRA-dashboard voor mijn stage.

3.3.1 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 Appendix A. 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.
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. Na het bekijken van nog enkele andere tools om dashboards te ontwikkelen, heb ik uiteindelijk voor Dashing gekozen, aangezien het enorm veel flexibiliteit biedt.

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 waarbij opmerkingen gegeven konden worden. Uit die opmerkingen zijn nog nuttige commentaren gekomen, waarbij enkele van die ideeën ook in het dashboard verwerkt zijn. De enquête en de resultatentabel zijn ook te vinden in Appendix B.

3.3.2 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 bepalen wat binnen mijn mogelijkheden lag en wat niet.

beter verwoorden

Gebruikte tools Dashing is een dashboard framework dat geschreven is in Sinatra, 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 Jenkins-omgeving opgezet, zodat het dashboard tijdens de ontwikkeling hiermee getest kon worden. Het schrijven van de code gebeurde in gedit 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 één 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 verschillende informatie wilt tonen op de verschillende Jenkins-widgets).

indien RSS ook: erbij zetten!

Gebruikte talen Het schrijven van het dashboard is gebeurd in verschillende talen, hieronder een opsomming welke taal gebruikt werden.

HTML HTML wordt gebruikt om te bepalen hoe de inhoud getoond wordt op de widget, dus bijvoorbeeld welk deel als titel zal getoond worden en welke onderdelen als paragrafen weergegeven worden. HTML bepaalt ook 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 code van een bepaalde opmaak hergebruiken via variabelen.

[2] http://dashing.io/
[3] http://batmanjs.org/

Javascript Javascript werd gebruikt in de Batman Bindings, die ervoor zorgen 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 een widget specifiëert 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 parset de 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* (mits wat opmaak door CSS) te zien zijn op de widget.

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], wat voldoende was om me op weg te zetten.
De verschillende bestanden worden als volgt gebruikt:

- Een widget wordt opgesteld door middel van een .HTML-file, een .scss-file en een .coffee-file.
- Een Embedded Ruby file geeft via HTML-blocks aan welke build jobs getoond worden op het dashboard.
- Een plain Ruby file doet de mapping van de JSON data op de variabelen in het dashboard en definiëert de namen van de build jobs die getoond kunnen worden.

Sprint Review Op het einde van de tweede sprint werd mij gevraagd om de eerste versie van het dashboard te demonstreren voor het team in Antwerpen, Bangalor en Chennai. Ik kon een werkend dashboard tonen dat informatie toonde over de Jenkins build jobs die op mijn lokale server draaiden. De feedback hierop was positief en enkele vragen van mensen in de verschillende teams konden beantwoord worden.

goed gezegd?

[4] Welke build jobs getoond worden, wordt bepaald door de Embedded Ruby file.
[5] https://www.codecademy.com/

3.3.3 Sprint 3: Feedback en verbetering

Na de sprint review heb ik de Alcatel-Lucent Jenkins-server geïntegreerd in het dashboard, wat nog anderhalve dag van debugging vroeg, gezien de Jenkins-server nu niet lokaal runde, maar op een andere server. Ook gebruikt men bij Alcatel-Lucent een andere versie van Jenkins, waardoor bepaalde sleutelwoorden waarop geparset wordt anders heten.
Na de sprint retrospective van sprint 2 en de sprint planning van sprint 3 heb ik nog een enquête gedaan om positieve en negatieve puntjes te verzamelen waar men bij de voorstelling van het dashboard tijdens de demo misschien niet was opgekomen. De resultaten van deze enqête zijn te vinden in de appendix.

verwijzing toevoegen

Ingebruikname

over vertellen

3.4 Resultaat

De eerste versie van het dashboard is te zien in Figuur 6. Het dashboard in zijn huidige vorm is gemaakt om getoond te worden op een HD-scherm met 16:9-ratio. Het is de bedoeling dat er zes tegels naast elkaar kunnen staan en dat er drie rijen zijn, wat maakt dat de status van 18 build jobs gelijktijdig getoond kunnen worden. Dit bleek ruim voldoende voor het NA-C team, aangezien er altijd maar op enkele builds tegelijkertijd gefocust wordt.

Indien er een definitieve (dwz veranderde) versie is, praat erover!

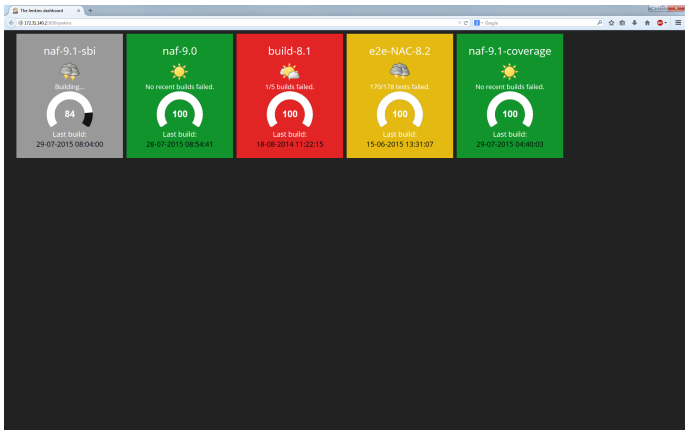

Figure 6: Eerste versie van het dashboard.

Figuur 7 toont de tegels in meer detail. De titel is de naam van de build waarover de tegel informatie verschaft. Het weerbericht toont de stabiliteit van de build, of, indien er testen gefaald zijn, de stabiliteit van de testen. De tekst eronder geeft aan hoe stabiel de builds/testen zijn, de meter geeft aan hoe ver de build is in zijn vooruitgang en de timestamp daaronder toont wanneer de laatste build uitgevoerd is. Er zijn verschillende achtergrondkleuren gebruikt, die de status van de laatste build weergeven:

- Grijs: de build is nog bezig, het weerbericht toont de stabiliteit van de builds voor degene die nu bezig is.
- Rood: de laatste build is gefaald.

- Geel: één of meerdere testen zijn gefaald.
- Groen: de laatste build is geslaagd en alle testen zijn geslaagd.

Als er gekeken wordt naar het voorstel opgemaakt in de literatuurstudie, is de achtergrondkleur daar niet aanwezig (de build status wordt er weergegeven met een gekleurde bol), maar de rondvraag op de afdeling toonde dat iedereen te vinden was voor de gekleurde achtergrond.

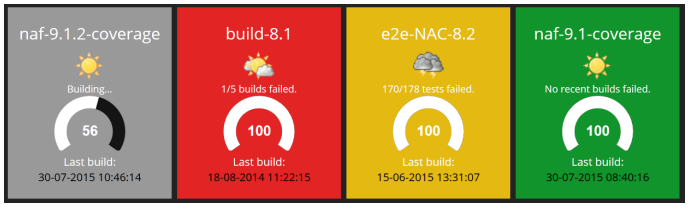

Figure 7: Gedetailleerd overzicht dashboard.

3.5 Initiatieven voor interns

Alcatel-Lucent organiseerde deze zomer een initiatief om alle interns met elkaar in contact te brengen tijdens speciale info-events en meetings. Op deze manier konden interns van verschillende afdelingen met elkaar bespreken hoe de stage op hun afdeling verliep en zo ervaringen met elkaar uitwisselen. Er zijn verschillende contactmomenten georganiseerd, meestal tijdens de middagpauze. Er waren verschillende bijeenkomsten, waaronder samen lunchen om kennis te maken, een gesprek met Joeri Veldeman, begonnen als doctor in de *electrical engineering* bij Alcatel en nu hoofd van de afdeling van human resources. Zijn ongewone carrièrepad werd ons uitgelegd en hij heeft informeel met ons gepraat over ons verdere studietrajecten. Verder is er ook een infosessie geweest over ALU Labs, waarin enkele van de zaken aan bod kwamen waarin Alcatel-Lucent investeert en research bij uitvoert.
Dit alles maakte dat de stage niet alleen een verrijking van mijn kennis binnen de afdeling werd, maar ook daarbuiten, waarbij contact met andere studenten gelegd werd.

4 Relatie stage en opleiding

Het doel van de stage was om een dashboard te maken zodat de build status van de verschillende onderdelen van projecten overzichtelijk getoond konden worden. Binnen deze stage werd ik bovendien ook betrokken bij alle bijeenkomsten met het team, om zo te ervaren hoe het er in een werkomgeving aan toe gaat. Op deze manier heb ik extra facetten gezien die tijdens de opleiding aan bod zijn gekomen, die ik niet had gezien moest ik niet zo betrokken geweest zijn bij alles. Zo werkt men hier volgens de agile-methode scrum, met elke dag standup meetings en tweewekelijkse sprints. Dit is iets dat in de lessen van Software-Ontwerp kort aan bod was gekomen, waarvan ik tijdens de stage zag dat het echt zijn nut heeft. De korte sprints en regelmatige reviews zorgen ervoor dat problemen niet te lang kunnen blijven bestaan en omdat er bijna dagelijks gepraat wordt over

wie waarmee bezig is, kunnen mensen elkaar vlot helpen en weet iedereen wie waaarmee bezig is. Doordat men gebruik maakt van continuous integration zijn (merge)fouten snel opgespoord en opgelost. Zelf kreeg ik ook eigen scrumproject, waardoor ik ook met sprints werkte en zo "kleine" doelen voor ogen had. Deze manier van werken is vergelijkbaar met die tijdens de P&O-sessies tijdens de bachelorproef, waarbij we om de paar weken een bepaalde functionaliteit moesten kunnen demonstreren. Op de afdeling NA-C doet men ook regelmatig aan pair-programming, iets wat ook aangeraden was in de sessies van Software-Ontwerp. Aangezien ik aan een eigen project moest werken, heb ik dit niet zo vaak gedaan, maar als ik met vragen zat, kwamen collega's met plezier samen met mij zoeken naar de oplossing en me uitleg geven.
Het dashboard dat ik moest maken moest niet een kant-en-klaar dashboard zijn waar ik dan zelf info aan toevoegde, maar één waarbij ik zelf moest nadenken over het design en de implementatie van de code om het dashboard te realiseren. Dit werd behandeld in de lessen van Software-Ontwerp.
Ik maakte gebruik van een bestaand framework, maar heb zelf nog heel veel van de code moeten schrijven. Dit heeft me er meer dan eens attent op gemaakt hoe belangrijk goede documentatie is, gezien de bestaande code op zich nauwelijks/niet gedocumenteerd was en ik soms moeite had om te begrijpen wat een bepaalde functie deed. Het schrijven van (goede) documentatie is iets waar tijdens de lessen van Objectgericht Programmeren altijd op gehamerd is. De documentatie op de websites van het framework en de plugins was vaak ook erg kort door de bocht of onvolledig, dus ik heb vaak lang gezocht naar bepaalde zaken. Omdat ik het zelf zo erg miste, heb ik zelf documentatie geschreven, in de hoop dat het voor een eventuele opvolger makkelijker is om alles te begrijpen.
Binnen de opleiding wordt er altijd gewerkt in kleine groepjes, meestal van twee mensen, maximaal in groepen van zes mensen. Op de afdeling NA-C werkt men in een team van ongeveer 10 mensen, wat het managen van het team nog moeilijker maakt dan bij een groep van slechts enkele mensen. Toch is er getoond dat dit perfect mogelijk is indien je de juiste tools gebruikt (bij NA-C maakt men gebruik van Confluence en JIRA) en indien er genoeg gecommuniceerd wordt.
Tijdens de lessen van Informatica werktuigen zijn Ruby, Linux en LaTeX aan bod gekomen. Ruby bleek nuttig om de (Embedded) Ruby voor het dashboard te schrijven, waar de kennis van Linux ervoor zorgde dat ik iets vlotter mijn weg vond bij het installeren van programma's en het werken met Linux in het algemeen, gezien het dashboard in Linux is ontwikkeld. LaTeX is gebruikt om de literatuurstudie te maken en het verslag op te stellen.
Doordat er in de bachelor informatica al een heel aantal programmeertalen gezien werden, was het aanleren van andere (opmaak)talen veel makkelijker.
Het schrijven van (wetenschappelijke) verslagen was een groot onderdeel tijdens Wetenschapscommunicatie en tijdens P&O, wat me hielp bij het schrijven van de literatuurstudie en het stageverslag.
Waar versiecontrole tijdens het project van Software-Ontwerp aangeraden, maar niet verplicht was, werd het bij P&O wel verplicht. Doordat ik hiermee had kennis gemaakt, heb ik hiervan gebruik kunnen maken (zij het aan de late kant, zie de kritische reflectie van de competenties in Sectie 5) tijdens een groot deel van de stage.

evt. tekst bijvoegen

5 Krititsche reflectie competenties

- Technische competenties die ik hoopte te verbeteren:

  Bijleren over version control Er wordt op de afdeling NA-C gewerkt met Git, waar ik ook tijdens mijn opleiding al mee in aanraking ben gekomen. Tijdens de ontwikkeling van het dashboard heb ik geen gebruik gemaakt van Git, hoewel dit zeker nuttig geweest zou zijn bij het schrijven van mijn dashboard. Eens de eerste versie van het dashboard klaar was, is het wel op Git gezet[6], maar dit had dus veel eerder moeten gebeuren. Er is gebruik van gemaakt bij het toevoegen van documentatie en bij het maken van veranderingen tijdens de derde sprint.
  Ondanks het te laat beginnen gebruiken van version control, zijn mijn competenties hierrond wel verbeterd, gezien ik Git nu ook via de Linux command line heb leren gebruiken en niet alleen in de grafische user interface op Windows.

  dit beter vertellen?

  Leren werken met Jenkins Op de afdeling NA-C werkt men met het continuous integration systeem Jenkins, dat builds automatisch runt nadat er een nieuwe versie van de software gecommit is. Tijdens de stage was het duidelijk hoe nuttig dit is: er werd op een gegeven moment een fout gemaakt, die ook nog eens onvoldoende getest bleek, maar men heeft deze heel snel ontdekt en binnen de 2 dagen kunnen oplossen. Moest er gewacht zijn met mergen tot vlak voor de release, was men waarschijnlijk veel meer problemen tegengekomen. Continuous integration is vooral handig in teams met meerdere mensen, het was dus van pas gekomen tijdens P&O of Software-Ontwerp.

  dat laatste weglaten?

  Mij inwerken in een reeds lopend project Ik moest niet meewerken aan de code van het NA-C team zelf, maar heb mij wel moeten inwerken in de code van het dashboard. Zoals hierboven vermeld, was dit niet altijd makkelijk, maar het heeft er wel voor gezorgd dat ik enorm veel heb bijgeleerd, net omdat ik heel veel zelf moest uitzoeken. Eens de eerste versie van het dashboard klaar was, kende ik de code enorm goed, dus nieuwe veranderingen waren zeer snel geïmplementeerd.

- Persoonlijke competenties die ik hoopte te verbeteren:

  Leren werken in grote teams In het NA-C team in Antwerpen werken er een tiental mensen, in Bangalor en Chennai zijn er teams van gelijkaardige grootte. Doordat er dagelijks een standup meeting was met de mensen van het team in Antwerpen, was iedereen goed op de hoogte van wie wat deed en doordat men met sprints van twee weken werkte, kwamen problemen snel boven. Dit is iets dat ik zeker meeneem naar projecten in de toekomst toe, ik heb ervaren hoe al deze zaken kunnen helpen in het beter doen verlopen van een project.

  Plannen op lange termijn Aan het begin van mijn stage werd me meteen gezegd dat ik redelijk vrij gelaten ging worden in mijn planning, maar er werd wel gevraagd om er een op te stellen, zodat ik ook in

[6] De code is te vinden op https://github.com/HelenaCat/jenkins-dashboard

sprints kon werken. Ik heb zelf voorstellen mogen doen en aan mijn eigen tempo (maar toch binnen de termijnen van de sprints) kunnen werken. Als er grote dingen te doen stonden, plande ik daar altijd genoeg tijd voor in, zodat ik uiteindelijk "te vroeg" klaar was en aan verbeteringen kon werken. Dit is een van de competenties die ik zowel in het tweede semester van mijn laatste bachelorjaar heb aangescherpt, als nu op de stage.

6 Besluit

Mijn stage ging voor 6 weken door bij Alcatel-Lucent in Antwerpen. Er werd mij gevraagd een dashboard te ontwerpen om de build status van het continuous integration systeem Jenkins te tonen.
Het dashboard is succesvol gemaakt binnen de opgelegde tijd met behulp van Dashing, draaiende op een Linux server.
Tijdens de stage heb ik enkele (opmaak)talen bijgeleerd, waaronder Javascript, JQuery, HTML en (S)CSS. Mijn kennis van Ruby kwam hierbij ook van pas.
Onder de goede begeleiding van Bart heb ik grotendeels zelfstandig mijn doelen en het project uitgewerkt en geleerd hoe het er in een bedrijf aan toe gaat.

Appendix A Literatuurstudie

General A dashboard is a tool to quickly show information at a glance. Its purpose is to have the "reader" gather information without having to look things up; everything should be clear by having a quick look at the dashboard. At Alcatel-Lucent, they make use of the Jenkins dashboard, but it doesn't comply with the given description: in order to see what builds failed/succeeded, you need to find the build you're looking for and then check its status. It is clear from the -lack of- its use that this way of displaying the information does not suffice, since the TV dashboard is not/barely used but every employer looks up the dashboard on their own PC every once in a while.

The purpose of this literature study is to create Jenkins awareness, so the developers can take a look at the TV dashboard a few times a day and have them see how the project is proceeding.
One of the most important things is that the information displayed on the dashboard can be easily interpreted. This means that you don't want a cluttered dashboard with too much information so it becomes unclear. Stephen Few has written an excellent article on the topic, with the main conclusions being:

- don't show too much information,
- keep the information simple,
- colors should be used sparingly or they will mean nothing at all,
- visuals should be useful; not overwhelming or distracting and
- the information represented should be accurate, so you should make sure the information is displayed correctly and with enough context.

This last remark is also made by Matthew Skelton[7], who shows that if you only show the failed builds, people will start to ignore the 'alarm signals' the dashboard displays and perform less well compared to when a context is shown, also showing the successful builds.
Few's theory is backed up by the people at Geckoboard, who have made a blog consisting of 6 parts, referencing to Few, but also adding some rules of thumb[8]. Every article listed in the appendix refers to Few's article and some make additional remarks. One of them is that it's very important to keep your audience in mind, the board of directors will want to see different metrics represented in the dashboard than, say, the people of human resources. In the case of software developers, they'll want to see metrics about the build status of the different jobs. Related to this is that not every dashboard will need the same layout and look.
In the case of the Jenkins dashboard, charts won't tell you much, since it doesn't really tell anything about the status of the last build. Below, the example of the Jenkins dashboard is further explained and a proposal is made that conforms to the guidelines of a good dashboard.

[7] http://blog.matthewskelton.net/2013/03/11/what-makes-an-effective-build-and-deployment-radiator-screen/

[8] https://www.geckoboard.com/blog/building-great-dashboards-6-golden-rules-to-successful-dashboard-design/

Improving the Jenkins dashboard The Jenkins dashboard, as seen in Figure 8, displays a list of the builds of a project, together with a colored dot and a "weather report". The dot displays information about the last build, the color indicating whether it failed or succeeded and a flashing dot meaning the build is still in progress. The color of the dot at that time then refers to the last finished build. The weather report tells you something about the stability of the builds. If the weather looks good, then the build is stable. The worse the weather gets, the less stable the build is. This way, you get an overall view about the stability of the different build jobs.

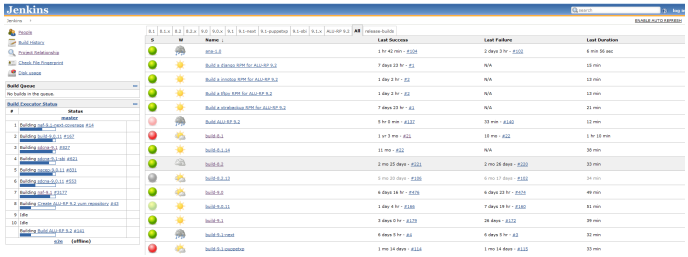

Figure 8: The Jenkins dashboard.

These two things give you a fairly good overview, but the downside of the dashboard is that you have to be right in front of the screen to see what job the colored dot and the weather report are referring to. The details of the build job are in plain text, so if you want to project it on a TV screen, people will have a hard time knowing what builds are doing well and which ones aren't. There are plug-ins available that show a dashboard, but they none of them comply with the most important rules described above: they're either too crowded or use too much color, distracting the viewer from what's really important.

The proposed images, as seen in Figure 9 and 10 comply with the design problems to be avoided as proposed by Stephen Few:

- Too much complexity
- Too many alert conditions
- Alerts that cannot be differentiated
- Overwhelming visuals
- Distracting visuals
- Inappropriate visual salience
- Mismatch between information and its visual representation
- Indirect expression of measures
- Not enough context

The images show the build stability together with the success or failure of the last build and show the name of the build clearly. It's also possible to add a timestamp of when the last success and failure were and the last duration at

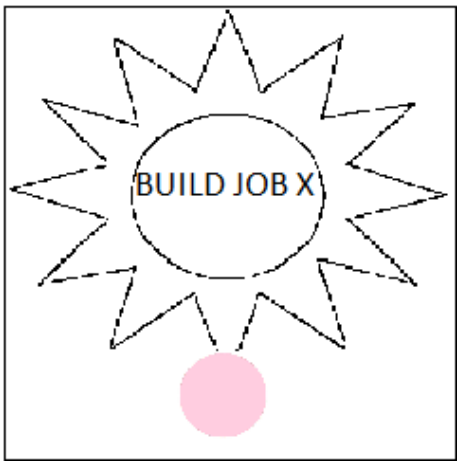

Figure 9: Proposal 1.

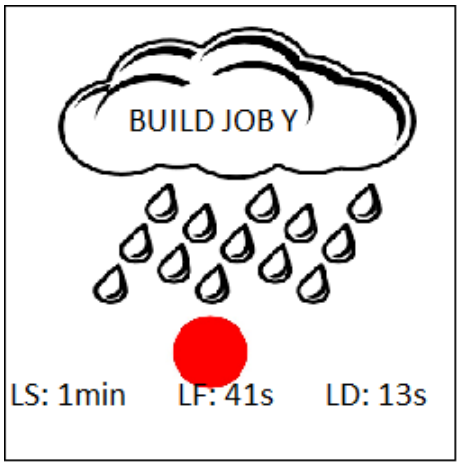

Figure 10: Proposal 2.

the bottom of the tiles, as shown in Figure 10. If you have color blind people in your team (as Few pointed out might happen), you could replace the green dot by a very light red dot, indicating it doesn't need attention, as shown in Figure 9. This could be realized using existing Jenkins plug-ins and changing them. The Radiator View[9] seems like a good basis. An alternative could be to implement already existing software like Dashing[10], which provides the tools to generate dashboards very easily. Instead of showing the weather as proposed, color codes may be used for the tile backgrounds, either representing the status of the last build or the status of the past builds. A combination is also possible where the color represents the status of the last build and an opaque weather report image as the background. An example is given in Figure 11:

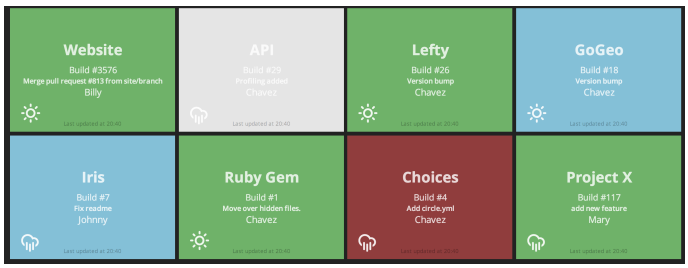

Figure 11: Example dashboard for Jenkins.

The colors aren't too overwhelming and they're meaningful, since people are used to the colored dots in the Jenkins dashboard. A third, less tile-oriented dashboard is Team Dashboard[11]. It also supports Jenkins plug-ins.

[9] http://blog.matthewskelton.net/2013/03/11/what-makes-an-effective-build-and-deployment-radiator-screen/

[10] dashing.io

[11] http://fdietz.github.io/team_dashboard/

References

[1] Stephen Few, *Dashboard Design For Real-Time Situation Awareness*, http://www.perceptualedge.com/articles/Whitepapers/Dashboard_Design.pdf, consulted on 09/07/2015

[2] Matthew Skelton, *What Makes an Effective Build and Deployment Radiator Screen?*, http://blog.matthewskelton.net/2013/03/11/what-makes-an-effective-build-and-deployment-radiator-screen/, consulted on 09/07/2015

[3] Nick Smith, *Designing and Building Great Dashboards - 6 Golden Rules to Successful Dashboard Design*, https://www.geckoboard.com/blog/building-great-dashboards-6-golden-rules-to-successful-dashboard-design/, consulted on 09/07/2015

[4] Zach Gemignani, *A Dashboard Alerts Checklist*, http://www.juiceanalytics.com/writing/dashboard-alerts-checklist/, consulted on 09/07/2015

[5] No author specified, *A Guide to Creating Dashboards People Love to Use*, http://www.cpoc.org/assets/Data/guide_to_dashboard_design1.pdf, consulted on 10/07/2015

Images:

[6] *Sun*, http://azcoloring.com/coloring-page/112445

[7] *Rain cloud*, http://www.clipartpanda.com/categories/animated-rain-clouds

[8] *Red dot*, http://www.thepointless.com/reddot

[9] *Pink dot*, http://www.create-a-mural.com/dry-erase-11-soft-pink-dot-decal.html

[10] *Example dashboard*, https://camo.githubusercontent.com/1509a366bd27fad24ba09c45793308a85bcbf8bf/687474703a2f2f662e636c2e6c792f6974656d732f336330743076335a31313145306f3436333033322f53637265656e25323053686f742532303230313132d31302d303325323061742532303825323e34362e3437253230504d2e706e67

Appendix B Enquête 1

Enquête De enquête bestond uit een reeks voorbeeldtegels met bepaalde kenmerken (zie Figuur 12) die ook in de enquête zelf uitgelegd werden (het deel boven "Preferences"). De enquête bestond uit volgende vragen, met ruimte om commentaar te voorzien:

"Hello" == name of build

1. No progress meter for the currently running build. Background color indicating the success/failure of the last completed build, big weather report about the last builds.
2. Small meter showing the progress of the currently running build, weather report at the bottom, background color indicating success/failure of last completed build.
3. Small meter with weather report on top, background color indicating success/failure.
4. Circular meter with weather report on top, background color indicating success/failure of last build.

Preferences (circle the preference)

- Meter/no meter; in case of meter:
  - Large/small
  - On top/at bottom
  - Circular/dashboard-like
- Weather report/no weather report
- Background color/no background color (colored dot is better)

Resultaten De resultaten zijn te zien in Tabel 1. Let op, niet iedereen heeft op elk onderdeel geantwoord, daarom zijn er niet op elke vraag 9 antwoorden.

	Yes	No
Meter	8	0
-Large	4	4
-On top	1	6
-Circular	3	5
Weather report	8	1
Background color	9	0

Table 1: Resultaten van de eerste enquête

Bijkomende opmerkingen zijn de volgende:

- "Would combine 1 and 3. Big weather report with small meter at the bottom"
- "It would be nice if we can show the failure count for failure builds"
- "I would replace 'last updated' with lastest finished build and building since (if in progress). Only show meter when busy building"

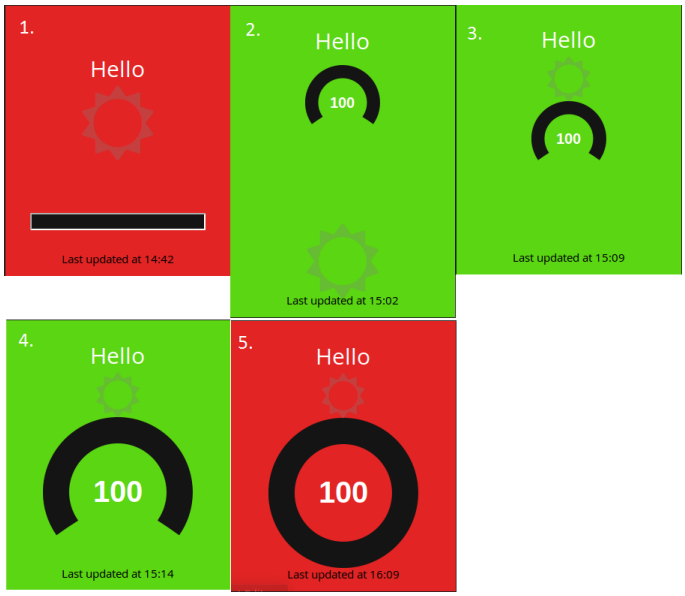

Figure 12: Voorbeeldwidgets.

- "Overview of last builds = list or circle with % of failed/success"
- "I'd put more visibility for the weather report/more important than current build progress. Maybe also add the date of the last build/number of test or failing tests?"