



hogeschool
Leiden

13-1-2017

Onderzoeksrapport

Groep 4

Auteurs:

Dennis van Bohemen
Philip Wong
Roy Touw

Module:

IpsenH

Opleiding:

Klas: INF3B
Opleiding: Informatica
School: Hogeschool Leiden

Versie:

Versie 2.5 Final

Versie	Datum	Omschrijving	Opgeslagen door:
0.1 Ontwikkel	13-9-2016	Initiële versie	Dennis van Bohemen
0.2 Ontwikkel	14-9-2016	Hoofstukken toegevoegd	Roy Touw, Dennis van Bohemen, Philip Wong
1.1 Concept	15-9-2016	Feedback Roland verwerkt	Roy Touw, Dennis van Bohemen, Philip Wong
1.2 Concept	19-9-2016	Feedback Roland verwerkt	Roy Touw, Dennis van Bohemen, Philip Wong
1.3 Concept	20-9-2016	Spellingscontrole	Roy Touw, Dennis van Bohemen, Philip Wong
1.4 Concept	12-10-2016	Toevoeging deelvraag 3	Roy Touw
1.5 Concept	26-10-2016	Toevoeging deelvraag 2 + evaluatie	Philip Wong
1.6 Concept	06-10-2016	Literatuurlijst + Voorwoord + Inleiding	Philip Wong
1.7 Concept	10-1-2017	Deelvraag 1 overzicht aangepast	Dennis van Bohemen
2.1 Pre-final	12-1-2017	Samenvatting en Deelvraag 4 bijgevoegd	Dennis van Bohemen, Roy Touw
2.2 Pre-final	13-01-2017	Resultaten experiment verwerkt.	Roy Touw
2.3 Pre-final	13-01-2017	Openstaande paragrafen afgerond.	Roy Touw
2.4 Pre-final	13-01-2017	Opmaak afgemaakt.	Roy Touw
2.5 Final	13-01-2017	Document afgerond.	Roy Touw

INHOUDSOPGAVE

1	Samenvatting.....	2
2	Voorwoord	3
3	Inleiding.....	3
4	Literatuurverkenning	3
5	Probleemstelling	3
6	Theorie en hypothese	4
	Deelvraag 1: Welke tools en methodieken worden gebruikt binnen een ontwikkelstraat?.....	4
6.1	Deelvraag 2: Wat maakt een tool binnen onze ontwikkelstraat van toegevoegde waarde en welke tools sluiten hier bij aan?	4
6.2	Deelvraag 3: In welke mate verbetert een ontwikkelstraat de kwaliteit van code?	5
6.3	Deelvraag 4: Is het voor ons project het waard om een ontwikkelstraat op te zetten?	5
7	Resultaten(Deelvragen uitwerken)	6
7.1	Deelvraag 1: Welke tools en methodieken worden gebruikt binnen een ontwikkelstraat?.....	6
	7.1.1 Inleiding	6
	7.1.2 categorieën.....	6
	7.1.3 Categorie beschrijvingen	7
	7.1.4 tools per categorie	10
7.2	7.1.5 Conclusie	22
	Deelvraag 2: Wat maakt een tool binnen onze ontwikkelstraat van toegevoegde waarde en welke tools sluiten hier bij aan?	24
	7.2.1 Inleiding	24
	7.2.2 Wat is C#?.....	24
	7.2.3 Wat is AngularJS?	24
	7.2.4 Continuous Integration.....	25
	7.2.5 Source Code Management (SCM)	26
	7.2.6 Automated Unit Testen.....	28
	7.2.7 Build Automation.....	29
	7.2.8 Conclusie	30

	In welke mate verbetert een ontwikkelstraat de kwaliteit van code?	31
	7.3.1 Inleiding	31
	7.3.2 Definitie Code Kwaliteit	32
	7.3.3 Doelstelling Experiment	33
7.3	7.3.4 Opzet Experiment	35
	7.3.5 Uitvoering Experiment.....	36
	7.3.6 Analyse Resultaten	37
	7.3.7 Conclusie en Beantwoording.....	37
	Deelvraag 4: Is het voor ons project het waard om een ontwikkelstraat op te zetten?	38
7.4	7.4.1 Inleiding	38
	7.4.2 Beheerbaarheid	38
	7.4.3 Productiviteit	39
	7.4.4 Code Kwaliteit	39
	7.4.5 Conclusie	39
8	Conclusie en discussie (Deelvragen conclusies en een eindconclusie)	40
9	Evaluatie	40
10	Aanbevelingen	41
11	Suggesties voor verder onderzoek.....	42
12	Literatuur	43
13.1	13 Bijlagen	45
13.3	Configuration management software vergelijking.....	45
	13.2 Configuration management software vergelijking	47
	Version control systeem vergelijking	48

1 SAMENVATTING

In dit document wordt de vraag zoals in het onderzoeksplan gedefinieerd luidend: "Hoe kunnen wij als software ontwikkelteam zorgen voor een hogere code kwaliteit voor de opdrachtgever gebruikmakend van een ontwikkelstraat?" beantwoord aan de hand van de deelvragen.

De hoofdvraag en daarmee ook een groot deel van dit document is opgedeeld in vier deelvragen zijnde:

- Deelvraag 1: Welke tools en methodieken worden gebruikt binnen een ontwikkelstraat?
- Deelvraag 2: Wat maakt een tool binnen onze ontwikkelstraat van toegevoegde waarde en welke tools sluiten hier bij aan?
- Deelvraag 3: In welke mate verbetert een ontwikkelstraat de kwaliteit van code?
- Deelvraag 4: Is het voor ons project het waard om een ontwikkelstraat op te zetten?

Per deelvraag wordt er onderzoek gedaan en een conclusie gesteld, uiteindelijk wordt er een algehele conclusie gevormd om de hoofdvraag te beantwoorden. Het antwoord om de hoofdvraag is dat er nu duidelijk is hoe een ontwikkelteam een ontwikkelstraat kan ontwerpen, realiseren en implementeren ten behoeve van de code kwaliteit binnen een software project.

Naast het beantwoorden van de hoofdvraag wordt er in dit document gekeken naar het verloop van het onderzoek en de onopgeloste vragen. Aan de hand van de onopgeloste vragen wordt er advies gegeven voor verder onderzoek wat gedaan kan worden.

2 VOORWOORD

Voor u ligt het onderzoeksrapport wat is opgesteld voor het bedrijf FortyTwoWindmills en de Hogeschool Leiden. Dit rapport is geschreven in het kader van ons laatste groepsproject op de hogeschool. Dit project is verdeeld over twee semesters en heeft een waarde van 9 studiepunten. Na uitvoerig onderzoek te hebben gedaan naar de ontwikkelstraat en code kwaliteit hebben wij de hoofdvraag kunnen beantwoorden. Bij dezen willen wij graag onze begeleiders bedanken voor de begeleiding en hun ondersteuning tijdens dit traject.

3 INLEIDING

Zoals al in het onderzoeksplan beschreven is, zal tijdens dit project een software ontwikkelstraat moeten worden opgezet. Het doel van deze ontwikkelstraat is om de beheerbaarheid, productie en codekwaliteit van een bepaalde codebase te verbeteren. Dit onderzoek is gedaan in opdracht van de Hogeschool Leiden en FortyTwoWindmills. Voordat de ontwikkelstraat kan worden opgezet moet er uitvoerig onderzoek worden gedaan naar de voor- en nadelen van een ontwikkelstraat. In dit rapport is het onderzoek beschreven over de tools die beschikbaar zijn voor een ontwikkelstraat. Vervolgens is er selectie gemaakt van deze tools. Het volgende deel van dit rapport bestaat uit een onderzoek naar wat codekwaliteit precies is. En ten slotte wordt de hoofdvraag van dit onderzoek beantwoord.

4 LITERATUURVERKENNING

Op het moment van schrijven is er literatuur te vinden over de verschillende zoals in het onderzoeksplan gedefinieerde deelvragen evenals de verschillende componenten binnen een ontwikkelstraat, ook is er literatuur te vinden over het onderwerp ontwikkelstraten zelf. Wat nog niet is beschreven is de verhouding van de door ons gedefinieerde deelvragen en een ontwikkelstraat, in het bijzonder de implementatie hiervan in de praktijk. Wel is er literatuur te vinden per component over de toegevoegde waarde hiervan.

5 PROBLEEMSTELLING

De probleemstelling is dat het momenteel niet duidelijk is of de implementatie van een ontwikkelstraat een positieve invloed heeft op de code kwaliteit binnen een software project, dat het niet duidelijk is uit welke componenten deze ontwikkelstraat zou moeten bestaan en hoe deze geïmplementeerd zou moeten worden.

6 THEORIE EN HYPOTHESE

In dit hoofdstuk zetten wij de onderzoeksvraag uiteen.

De onderzoeksvraag luidt: Hoe kunnen wij als software ontwikkelteam zorgen voor een hogere code kwaliteit voor de opdrachtgever gebruikmakend van een ontwikkelstraat? Omdat deze hoofdvraag te breed en complex is om zo te kunnen beantwoorden hebben wij deze opgedeeld in de in dit hoofdstuk volgende deelvragen.

Deelvraag 1: Welke tools en methodieken worden gebruikt binnen een ontwikkelstraat?

- 6.1 Deze deelvraag zal onderzocht worden aan de hand van een literatuuronderzoek. Hierin zal naar online bronnen gezocht worden naar de verschillende tools die gebruikt worden binnen een ontwikkelstraat. Dit levert een lijst op met verschillende soorten tools binnen een ontwikkelstraat. De lijst zal over de informatie beschikken die benodigd is om uiteindelijk een keuze te maken uit de verschillende tools binnen deelvraag 2. Hiermee kan gedacht worden aan kosten, beschikbaarheid met bijbehorende beschrijving en het desbetreffende onderdeel waaronder de tool valt. Met deze lijst kan vervolgens deelvraag 5.2 beantwoordt worden zodat de tools worden uitgekozen die gebruikt worden binnen onze ontwikkelstraat.

6.2 Deelvraag 2: Wat maakt een tool binnen onze ontwikkelstraat van toegevoegde waarde en welke tools sluiten hier bij aan?

Om uiteindelijk een ontwikkelstraat op te kunnen zetten moeten wij uit de resultaten van deelvraag 1 een selectie maken met beste tools. Hiervoor zal er een literatuuronderzoek uitgevoerd worden. Er wordt dan gekeken naar zowel de gebruikerservaring als resultaten van wetenschappelijk onderzoek. Van alle potentiële tools wordt een opsomming gemaakt van de voor- en nadelen. Op basis van deze lijst worden de beste tools geselecteerd.

Deelvraag 3: In welke mate verbetert een ontwikkelstraat de kwaliteit van code?

6.3 Bij deze deelvraag wordt het verband van het implementeren van een ontwikkelstraat en de code kwaliteit onderzocht. De opzet is dat er een deel literatuur onderzoek is om vast te stellen wat code kwaliteit precies is, gevolgd door een door ons opgezet experiment om zo te pogen het verband vast te stellen. Bij deze deelvraag worden de conclusies van de eerdere deelvragen gebruikt om zo een simulatie ontwikkelstraat op te stellen welke gebruikt gaat worden in het experiment.

Met het experiment proberen we de volgende door ons opgestelde hypothese te bevestigen:

De hypothese bij dit experiment is dat er een meetbaar causaal verband is tussen het implementeren van een ontwikkelstraat bestaande uit de componenten zoals beschreven in de tweede deelvraag van ons onderzoeksplan binnen een bestaand software project, en de onderhoudbaarheid van code zoals vastgesteld in de ISO norm 25010:2011.

6.4 Deelvraag 4: Is het voor ons project het waard om een ontwikkelstraat op te zetten?

Deze deelvraag geeft antwoord op de hoofdvraag binnen ons eigen project. Deze deelvraag gaan we beantwoorden aan de hand van drie criteria zijnde:

- Efficiëntie
- Onderhoudbaarheid
- Code kwaliteit

Per criteria zal er onderzocht worden of de ontwikkelstraat binnen ons project een positieve invloed heeft gebracht. Aan de hand van deze drie criteria zal er geconcludeerd worden of de ontwikkelstraat het waard is geweest binnen ons project.

7 RESULTATEN(DEELVRAGEN UITWERKEN)

Deelvraag 1: Welke tools en methodieken worden gebruikt binnen een ontwikkelstraat?

7.1.1 Inleiding

- 7.1 Na diverse bronnen raad te plegen zijn de categorieën naar voren gekomen die in het volgende hoofdstuk worden behandeld. Deze categorieën worden vervolgens gebruikt om de diverse gevonden tools in te delen per categorie. Al deze informatie is ter beschikking gekomen door een literair onderzoek uit te voeren.

7.1.2 categorieën

We kunnen vaststellen dat er geen vaste categorie indeling wordt gemaakt in het grote aanbod van tools binnen de ontwikkelstraat. Dit komt omdat er na het onderzoek geen bronnen zijn gevonden die een vaste categorisering handhaven binnen de bron zijn informatiestructuur. Wel zijn er verschillende categorieën gevonden welke veel overeenkomsten bevatten of identiek overeenkomen. Doordat we deze overeenkomsten in de volgende bronnen hebben gevonden ([17] http://blogs.forrester.com/kurt_bittner/15-01-12-continuous_delivery_pipeline_tool_categories) en ([18] <http://thenewstack.io/exploding-infrastructure-automation-stack-ecosystem/>) hebben we er een aantal kunnen combineren en enkele laten vervallen omdat we deze in een andere categorie konden samenvoegen.

De volgende door ons gehanteerde categorieën zijn naar voren gekomen door de verschillende bronnen te combineren vanwege de verschillende overeenkomsten van categorieën.

- Continuous Delivery Management
- Continuous Integration tools
- Source Code Management
- Build Automation
- Unit Testing
- Service Virtualization
- Test (data) Management
- Automatisated Functional Testing
- Performance/Load Testing
- Deployment Automation
- BI Monitoring
- Cloud IAAS
- Configuration Management
- Release Management
- Databases
- Microservices

In het volgende hoofdstuk zullen we deze categorieën kort bespreken om duidelijkheid te scheppen waarom we op deze manier de tools hebben gecategoriseerd.

7.1.3 Categorie beschrijvingen

Continuous Delivery Management

Continuous Delivery Management tools managen de software oplevering van een ontwikkelstraat voor een applicatie, inclusief alle gerelateerde componenten en services, build-time en run-time afhankelijkheden. Ze organiseren de applicatie oplevering workflow van code check-in tot deployment van de applicatie in een productieomgeving. En weergeven de status van de applicatie in de flow van een ontwikkelstraat. Continuous Delivery is een verlenging van Continuous Integration.

Continuous Integration tools

Continuous Integration is het proces waarbij alle veranderingen van een software applicatie elke keer gebuild en getest wordt wanneer een code ingechecked is. Deze tools kunnen zorgen dat dit geautomatiseerd wordt.

Version Management Tools

Dit zijn versie management systemen voor de source code van een applicatie. Ze worden ook wel Source code management tools genoemd. We noemen het versie management tools omdat dit soort systemen vaak meer doen dan alleen source code beheren.

Build Automation

Build automation tools zijn software applicaties die het compileren en integreren van uitvoerbare softwarecomponenten van source code automatisch kunnen uitvoeren. Bij Continuous Delivery worden build automation tools betrokken bij het Continuous Delivery proces en tool.

Unit Testing Tools

Unit testen worden gebruikt om te bevestigen dat bepaalde onderdelen of functies van code goed werken.

Service virtualization

Deze tools simuleren het gedrag van een systeem of service door een alternatieve implementatie van dat systeem of service aan te bieden. De simulatie voorziet een exacte functionele gelijkheid en voorziet ook in de simulatie van de load en performance karakteristieken van het systeem of service.

Test (Data) Management

Test management tools beheren de software test processen zoals het beheren van test cases, tests automatiseren en het beheren van test resultaten. Hiermee kunnen enkele tools ook test data aanmaken en beheren zodat deze test data gebruikt kan worden voor het testen van de applicatie of service.

Automated Functional Testing

Net zoals Unit test tools gebruiken automated functional testing tools ook API's voor het testen van de geïntegreerde functionaliteit van een applicatie. Deze tools bevinden zich echter na unit testen en controleren of elke component werkt. Vaak wordt bij Automated functional testing tools ook geautomatiseerde UI-based tests inbegrepen.

Performance/Load Testing

Performance/Load Testing tools testen de schaalbaarheid en performance van een systeem door de workload en transacties te simuleren voor de te ontwikkelen applicatie.

Deployment Automation

Deployment Automation tools worden gebruikt om een applicatie automatisch uit te laten voeren in een bepaalde omgeving. De "applicatie" kan zijn een volledige stack, of alleen een set aan configuraties of een onderdeel van een applicatie.

([19]<https://www.youtube.com/watch?v=z3uwjGxmM6g> ,6:10, What is a deployment automation tool and how can it help me. 4 dec 2014).

Bi Monitoring tools

Bi Monitoring tools worden gebruikt om data te visualiseren zodat hier statistieken herleid kunnen worden. In een software ontwikkelstraat kan dit toegepast worden om bijvoorbeeld bepaalde metadata van het Version Management Systeem te visualiseren in een diagram. Zolang de data beschikbaar kan een Bi Monitoring tool de data ophalen uit een database en vervolgens visueel weergeven.

Cloud IaaS (Infrastructure as a Service)

IaaS is een vorm van cloud computing

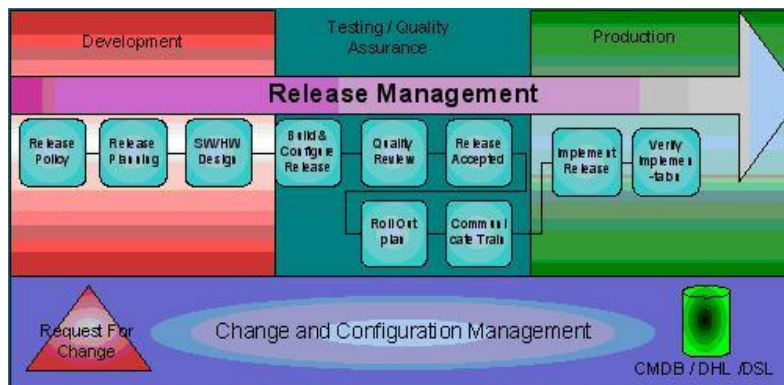
([20]https://nl.wikipedia.org/wiki/Infrastructure_as_a_service). Dit wil zeggen dat de infrastructuur virtueel wordt aangeboden en hardware waaronder de servers, netwerkkapparatuur en de werkstations eigendom zijn van de serviceprovider. De afnemer betaald alleen voor hetgeen daadwerkelijk gebruikt wordt. De kostprijs van een Cloud IaaS zal meegroeien met de toename van de benodigde capaciteit en omdat de diensten werken op basis van aanvragen zijn ze schaalbaar.

Configuration Management (CM)

Configuration Management is het systematisch hanteren van veranderingen zodat een systeem zijn integriteit behoudt. CM implementeert de toegang, procedures, technieken en tools die aangevraagde veranderingen beheren, en zorgt voor het onderhouden van een inventaris van een systeem en support documenten van de systeem veranderingen.

Release Management

Release Management beheert het opleveren van een applicatie tussen de verschillende fasen en omgevingen, inclusief het testen en de software deployen naar de productieomgeving. Het onderstaande schema geeft een visuele weergave van de verschillende fasen die doorlopen worden bij Release Management



Bron: <http://www.itsmwatch.com/itil/article.php/3680776/Release-Management-Where-to-Start.htm>

Databases

Een Database vormt een centrale opslagplaats voor de applicatie of service. Onder de database verstaan we ook het beheersysteem voor de toegankelijkheid van de database. De databases worden vaak verkregen met een beheertool van dezelfde uitgever.

Microservices

Microservices kan vergeleken worden met SOA en worden gezien als een aftakking van SOA.

De term Microservices beschrijft een manier van designen van software applicaties welke passen bij onafhankelijk inzetbare services ([21]<http://www.martinfowler.com/articles/microservices.html>). De Microservices zijn vaak klein ontworpen services en worden steeds populair voor Continuously Deployment systemen.

Deze Microservices kunnen vergeleken worden met kleine applicaties welke makkelijk zijn om snel te deployen en te vervangen. Hierdoor is Continuously Deployment vaak gebruikt in samenhang met Microservices en vanwege het feit dat Microservices van nature een modulaire opbouw van de applicatie afdwingt. Microservices kunnen beheert worden door speciaal ontwikkelde zogeheten Container tools of door een PaaS (Platform as a service) Tool zoals de Google Cloud Service.

7.1.4 tools per categorie

Nu we de verschillende categorieën hebben opgesteld, gaan we de tools inventariseren per categorie. Dit doen we door een literatuuronderzoek te doen. We produceren hiermee een lijst met verschillende tools per categorie waarbij beschikbare informatie is gevonden over de kosten en een beknopte beschrijving van de betreffende tool. Tools worden meerdere malen genoemd omdat er overlap bestaat bij enkele tools. Wanneer de tool is genoemd zal er niet nog een beschrijving plaatsvinden om overzicht te bewaren. Deze tools bieden meerdere functionaliteiten die onder meerdere categorieën vallen. We specificeren niet iedere tool die gevonden wordt maar benoemen de tools die in het literatuuronderzoek het meest naar voren kwamen uit verschillende bronnen en het best aansluiten bij de behoefte van de opdrachtgever. Vervolgens wordt na de gegeven lijst deelvraag 2 verder uitgewerkt met de geproduceerde lijst van deze deelvraag.

7.1.4.1 Continuous Delivery (CD) Management

Jenkins

Kosten: 14 dagen gratis trial versie. Kosten op aanvraag.

Website: www.jenkins.io

Beknopte beschrijving: Jenkins is een cross-platform, continuous integration en continuous delivery applicatie die de productiviteit verhoogd. Jenkins kan gebruikt worden om software te builden en testen volgens de Continuous Integration en Continuous Delivery filosofieën.

Features:

Easy installation: Just run `java -jar jenkins.war`, deploy it in a servlet container. No additional install, no database. Prefer an installer or native package? We have those as well.

- Easy configuration: Jenkins can be configured entirely from its friendly web GUI with extensive on-the-fly error checks and inline help.
- Rich plugin ecosystem: Jenkins integrates with virtually every SCM or build tool that exists. View plugins.
- Extensibility: Most parts of Jenkins can be extended and modified, and it's easy to create new Jenkins plugins. This allows you to customize Jenkins to your needs.
- Distributed builds: Jenkins can distribute build/test loads to multiple computers with different operating systems. Building software for OS X, Linux, and Windows? No problem.

GoCD

Kosten: Open Source

Website: www.Go.cd

Beknpte beschrijving: GoCD is een Open Source Continuous delivery server, gespecialiseerd in een complexe workflow modellering en visualisatie.

Buildbot

Kosten: Neem contact op voor meer informatie

Website: www.buildbot.net

Beknpte beschrijving: Buildbot is een open source framework voor het automatiseren van software builds, tests en release processen.

Circleci

Kosten: Eerste container is gratis, meerdere containers kosten \$50 per maand.

Website: www.circleci.com

Beknpte beschrijving: Flexibel te gebruiken cross-platform CI en CD tool. De flexibiliteit komt voort uit de verschillende plugins die gebruikt kunnen worden.

- Veel notificatie mogelijkheden zoals Slack, Hipchat, Mail
- Onafhankelijke builds.
- Overzichtelijke UI.
- Goede Github integratie
- Android support

7.1.4.2 Continuous Integration (CI) tools

Jenkins

Eerder beschreven.

Bamboo

Kosten: Gratis versie voor Open Source projecten. Commerciële organisaties betalen per build agent.

Website: www.atlassian.com

Beknorte beschrijving: Bamboo is een Cross-platform server voor CI. Het wordt gebruikt voor het bouwen, testen en deployen van applicaties automatisch per requirement. Hierdoor helpt Bamboo het release proces.

Buildbot

Eerder beschreven.

Circleci

Eerder beschreven

CloudBees

Kosten: Gebaseerd op Jenkins kosten

Website: www.CloudBees.com

Beknorte beschrijving: CloudBees is een platform als een service gemaakt op het Jenkins platform. Een cloud provider die zich focust op open source CI. Verschillende Jenkins plugins kunnen gedownload worden voor het optimaliseren van de beheersbaarheid van grote operaties en het optimaliseren van bestaande services en beveiliging.

Snap

Kosten: Open Source

Website: www.snap-ci.com

Beknorte beschrijving: Snelle CI tool met debug, Pull Request + Branche Tracking functionaliteit en Automatic Deployment functionaliteiten. De tool maakt gebruik van pipelines en tests die parallel naast elkaar lopen.

- integratie met GitHub
- Krachtige API

TeamCity (Jetbrains)

Kosten: Gratis versie. Enterprise editie op aanvraag

Website: www.jetbrains.com/teamcity

Beknorte beschrijving: Een Java gebaseerde build management en CI tool van JetBrains. Java, .NET en Ruby platformen worden ondersteund.

Gated commits. Hierdoor kunnen developers een source code niet kapot maken door het build script remote uit te voeren. Meerdere builds en testen kunnen worden uitgevoerd op verschillende platformen en omgevingen. Tegelijkertijd geïntegreerde code coverage, inspecties en dubbele gegevens zoek functionaliteit. Integratie met Eclipse, IntelliJ, Visual Studio

Shippable

Kosten: Gratis versie waarbij open source projecten maximaal 150 builds per maand mogen maken. Betaalde versie \$75.

Website: www.shippable.com

Beknorte beschrijving: Shippable is een gehoste cloud platform die CI, deployment en testen levert aan GitHub en Bitbucket repositories. De ondersteunde talen zijn Ruby, Python, Java, Scala, Node.js en PHP.

Uitgebreide informatie over CI tools.

(https://en.wikipedia.org/wiki/Comparison_of_continuous_integration_software)

7.1.4.3 Source Code Management (SCM)**GIT**

Kosten: Gratis

Website: www.git-scm.com

Beknorte beschrijving: Git is een versie control systeem. GIT valt onder een distributed revision control system en is gericht op snelheid, data integriteit en ondersteuning voor non-lineaire workflows. Linus Torvalds heeft in 2005 Git ontwikkeld. GIT wordt vaak door andere SCM systemen als basis gebruikt.

Mercurial

Kosten: Gratis

Website: www.mercurial.com

Beknorte beschrijving: Een cross-platform gedistribueerd SCM systeem. Het ondersteund Microsoft Windows en UNIX systemen zoals FreeBSD, MAC OS en Linux. Mercurial is primair een command line programma.

Bitbucket

Kosten: Gratis private repository met 5 gebruikers. Voor 10 gebruikers \$10 per maand, voor 25 gebruikers \$25 aan kosten.

Website: www.bitbucket.com

Beknorte beschrijving: Bitbucket is een online server voor het onderhouden van een code source door middel van een SCM systeem. Bitbucket ondersteund GIT en Mercurial als SCM systeem voor de versioning van de source code. Bitbucket biedt hierdoor alleen de opslag van de source code aan.

Subversion (SVN)

Kosten: Gratis

Website: www.subversion.apache.org

Beknorte beschrijving: SVN is een versioning systeem en revisie control systeem. SVN werkt in tegenstelling tot GIT volgens een client-server repository model.

Gitlab

Kosten: Gratis open repository.

Website: www.gitlab.com

Beknorte beschrijving: Een gratis repository manager. Evenals Bitbucket biedt Gitlab de mogelijkheid om code op te slaan en kan Gitlab gebruik maken van verschillende versioning control systemen.

Uitgebreide informatie over SCM systemen.

(https://en.wikipedia.org/wiki/Comparison_of_version_control_software)

7.1.4.4 Build Automation

TeamCity (Jetbrains)

Eerder beschreven.

Apache Ant

Kosten: Open Source

Website: www.ant.apache.org

Beknorte beschrijving: Een software tool voor automatisch software building processen. Het vereist het Java platform en kan het beste gebruikt worden voor het maken van Java projecten. Ant gebruikt XML voor de opbouw van projecten.

Apache Maven

Kosten: Open Source

Website: www.maven.apache.org

Beknorte beschrijving: Maven is een cross-platform Automatic Build tool gemaakt voor vooral Java projecten. Maven gebruikt een plugin-based architectuur. Hierdoor kan elke applicatie bestuurd worden door de standaard input. Naast Java wordt het .NET framework en de C/C++ taal minimaal ondersteund. Maven gebruikt XML voor de opbouw van projecten. Maven gebruikt in tegenstelling tot Ant een Groovy-based domain-specific language.

Gradle

Kosten: Open Source

Website: www.gradle.org

Beknorte beschrijving: Build System gebaseerd op de concepten van Apache Maven en Apache Ant. In tegenstelling tot Maven wordt het XML formaat niet gebruikt om het project te definiëren. Hier wordt een directed acyclic graph voor gebruikt. Gradle richt zich vooral op Java, Groovy en Scala development.

MSBuild

Kosten: Gratis

Website: www.github.com/microsoft/msbuild

Beknorte beschrijving: Microsoft Build Engine ook wel MSBuild genoemd is een build tool gemaakt voor het managen van code geschreven in C#, C++ en andere programmeertalen die gebruikmaken van het .NET framework. MSBuild kan gebruikt worden voor het compilen van de source code, packaging, testing en deployment. MSBuild werkt samen Visual Studio projecten en wordt dan ook geleverd bij de IDE Visual Studio. MSBuild werkt met project bestanden op dezelfde XML wijze en syntax als Apache Ant.

Bamboo

Eerder beschreven.

Jenkins

Eerder beschreven.

*7.1.4.5 Unit Testing***JUnit**

Kosten: Gratis

Website: www.junit.org

Beknpte beschrijving: Test framework voor Java Platformen.

Cucumber

Kosten: Gratis

Website: www.atlassian.com

Beknpte beschrijving: Behavior-driven development test tool.

Mocha

Kosten: Gratis

Website: www.mochajs.org

Beknpte beschrijving: Javascript test framework dat draait op Node.js

Unit JS

Kosten: Open Source

Website: www.unitjs.com

Beknpte beschrijving: Cross-platform unit test framework voor Javascript. Het volgt het principe van Behavior-driven development

CSUnit

Kosten: Open Source

Website: www.csunit.org

Beknpte beschrijving: C# Unit test tool met GUI en command line besturing. CSUnit ondersteund .NET 3.5 en eerder.

MSTest (Visual Studio Test Framework)

Kosten: Bij aanschaf van Visual Studio

Website: www.visualstudio.com

Beknorte beschrijving: De IDE Visual Studio heeft een geïntegreerd test framework voor unit testen genaamd MSTest. Hierdoor kan bij het gebruik van Visual Studio deze tool als Unit test tool voor C# gebruikt worden.

NUnit

Kosten: Gratis versie voor Open Source projecten. Commerciële organisaties betalen per build agent.

Website: www.nunit.com

Beknorte beschrijving: Komt met een GUI en een command line besturing. Het is een open source unit test framework voor het Microsoft .NET framework. Het heeft dezelfde functionaliteit als JUnit bij Java. NUnit is geïntegreerd in Visual studio met Resharper

*7.1.4.6 Service Virtualization***MockServer**

Kosten: Gratis versie voor Open Source projecten. Commerciële organisaties betalen per build agent.

Website: www.mockserver.com

Beknorte beschrijving: Mockserver is een open source mocking framework voor HTTP en HTTPS. MockServer is gemaakt om integratie testen te vergemakkelijken door HTTP en HTTPS systemen zoals een webservice of website testen.

*7.1.4.7 Test (data) Management***Extension Cord**

Kosten: Open Source

Website: <https://github.com/DeemOpen/excord>

Beknorte beschrijving: Open Source test case management tool voor basis use cases.

Meer info zie (https://en.wikipedia.org/wiki/Test_management_tools)

7.1.4.8 Automated Functional Testing

Selenium

Kosten: Open Source

Website: www.seleniumhq.org

Beknorte beschrijving: Ondersteund Functioneel testen voor web applicaties en voor desktop applicaties. Je kan browser events automatiseren en het opslaan en opnieuw uitvoeren van scripts. Werken met Selenium is makkelijk volgens (<http://testingfreak.com/top-10-free-open-source-functional-testing-tools/>)

Cucumber

Eerder beschreven.

Windmill

Kosten: Open Source

Website: <https://github.com/windmill>

Beknorte beschrijving: Open Source web testing tool. Tests kunnen geschreven worden voor Javascript, Python en Java

Ranorex

Kosten: € 1990,- voor de goedkoopste versie

Website: <http://www.ranorex.com>

Beknorte beschrijving: Ranorex kan je webapplicatie en GUI tests automatiseren (naast nog andere dingen). Ranorex valt onder de meer populaire test tools.

JMeter

Kosten: Open Source

Website: <http://www.jmeter.apache.org>

Beknorte beschrijving: Open Source Java platform applicatie voor het testen van performance tests en functionele tests.

7.1.4.9 Performance/Load Testing

WebLoad

Kosten: Gratis tot 50 virtuele gebruikers

Website: <http://www.radview.com/>

Beknorte beschrijving: Load en performance test tool voor webapplicaties. Elke internetapplicatie die gebruik maakt van Ajax, Adobe Flex, .Net, Oracle forms, HTML5 en meer kan gebruik maken van deze test tool. Applicatie draait alleen op Windows en Linux.

JMeter

Eerder beschreven.

7.1.4.10 *Deployment Automation***Snap**

Kosten: Open Source

Website: www.snap-ci.com

Beknorte beschrijving: Snelle CI tool met debug, Pull Request + Branche Tracking functionaliteit en Automatic Deployment functionaliteiten. De tool maakt gebruik van pipelines en tests die parallel naast elkaar lopen.

- integratie met GitHub
- Krachtige API

Jenkins

Eerder beschreven.

Bamboo

Eerder beschreven.

TeamCity (Jetbrains)

Eerder beschreven

Visual Studio (Deployment Automation)

Kosten: Bij aanschaf van Visual Studio

Website: www.visualstudio.com

Beknorte beschrijving: De IDE Visual Studio heeft een geïntegreerd framework voor deployment automation.

Buildbot

Eerder beschreven.

7.1.4.11 *Bi Monitoring***Nagios**

Kosten: Laagste versie is gratis.

Website: www.nagios.org

Beknorte beschrijving: Een Open Source BI monitoring tool

7.1.4.12 *Cloud IAAS*

Jenkins

Eerder beschreven.

Puppet

Kosten: gratis tot 10 nodes

Website: www.puppet.com

Beknpte beschrijving: Puppet zorgt voor een standaard manier van opleveren en operationaliseren van software, ongeacht waar de software staat. Puppet werkt met de programmeertaal Ruby.

Microsoft Azure

Kosten: Een gratis versie maar meer functionaliteit vereist betaling

Website: www.azure.windows.com

Beknpte beschrijving: Een Cloud-computing platform van Microsoft waarmee internet diensten aangeboden kunnen worden via het internet of binnen de omgeving van het bedrijf.

Google Cloud Platform

Kosten: Betaling per request

Website: www.googleappengine.com

Beknpte beschrijving: Een Cloud-computing platform van Googlee waarmee internet diensten aangeboden kunnen worden via het internet of binnen de omgeving van het bedrijf. Het wordt ook wel het Google App Engine genoemd en is beschikbaar voor Python en Java.

7.1.4.13 *Configuration Management*

Puppet

Eerder beschreven.

Chef

Kosten: gratis versie met beperkte functionaliteit

Website: www.chef.io

Beknpte beschrijving: Chef beschrijft "recepten" die beschrijven hoe Chef de server applicaties managed en de bijkomende server applicaties zoals bijvoorbeeld Apache HTTP Server, MySQL of Hadoop zijn geconfigureerd. Deze recepten kunnen als "kookboek" worden gekoppeld. Chef kan ook integreren met de meeste Cloud Based Platforms zoals Microsoft Azure of Google Cloud Platform

7.1.4.14 *Release Management*

Buildbot

Eerder beschreven.

Gradle

Eerder beschreven.

Bmc

Kosten: Gratis trial versie. Contact opnemen voor meer informatie

Website: www.bmc.com

Beknpte beschrijving: Speciaal gericht op het managen van het releasen van software producten.

7.1.4.15 *Databases*

MySQL

Kosten: Gratis community editie

Website: www.mysql.com

Beknpte beschrijving: Relationale SQL database met een overzichtelijk web-interface database management systeem.

PostgreSQL

Kosten: Gratis

Website: www.postgresql.org

Beknpte beschrijving: Open Source relationele SQL Database met bijbehorend database management systeem met veel functionaliteit.

MongoDB

Kosten: Gratis en Open Source

Website: www.mongodb.com

Beknpte beschrijving: Open Source NoSQL database met een database management systeem.

7.1.4.16 *Microservices*

Kubernetes

Kosten : Open Source

Website: www.kubernetes.io

Beknpte beschrijving: Open Source applicatie die gecontaineriseerde applicaties beheert.

Mesos

Kosten: Open Source

Website: <http://mesos.apache.org/>

Beknpte beschrijving: Gericht op het beheren applicaties en is cross-platform.

7.1.5 Conclusie

7.1.5.1 *Eindresultaat*

De verschillende categorieën geven een duidelijk overzicht van de verschillende soorten tools. Weliswaar is de hoeveelheid tools die beschikbaar zijn per categorie zodanig groot dat er maar een beperkt aantal in dit onderzoek zijn omschreven. De tools zijn vervolgens gekozen op basis van de eigenschappen die het beste bij de opdrachtgever passen.

In de categorisering van de tools worden Integrated Development Environments buitenschoot gelaten. In het onderstaande hoofdstuk zal beargumenteerd worden waarom deze niet binnen een ontwikkelstraat vallen en hierdoor ook niet als categorie is meegenomen binnen dit onderzoek.

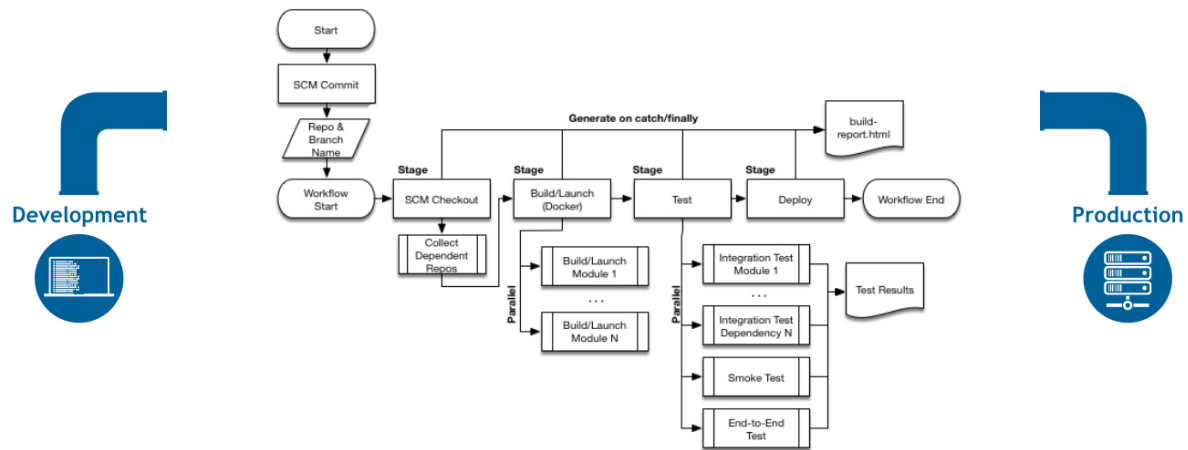
7.1.5.2 *Integrated Development Environments (IDE ' s)*

IDE's worden bij de verschillende door ons onderzochte bronnen altijd buiten de categorisering van ontwikkelstraten gehouden, waardoor dit kopje is bijgevoegd om op een beknopte wijze te beargumenteren waarom dit onderzoek de IDE niet als tool beschouwd voor ons onderzoek naar een ontwikkelstraat.

Integrated Development Environments zijn uitgebreide text-editors voor het maken, aanpassen, debuggen en managen van code. Daarnaast bieden de meeste IDE's een compiler of interpreter waarmee de code lokaal uitgevoerd kan worden. Het gebruik van een IDE vergemakkelijkt het ontwikkelen van code voor een software ontwikkelaar (<http://searchsoftwarequality.techtarget.com/definition/integrated-development-environment>).

IDE's vallen onder de ontwikkelaar zijn programmeeromgeving (1.8, blz 52, concepts of programming languages) en bevat de eerste stap voor het aanmaken, uitvoeren en testen van de code. IDE's zijn onafhankelijk van de ontwikkelstraat. Dit komt omdat een IDE geen verband heeft met het deployment proces van stukken code. Ongeacht welke IDE gebruikt wordt, zal de IDE het deployment proces niet beïnvloeden. Dit wordt

bevestigd wanneer we kijken naar een Continuous Integration tool genaamd Jenkins. Het onderstaande schema weergeeft de workflow van het integratieproces van een stukje code met behulp van Jenkins.



bron: <https://jenkins.io/doc/pipeline/>

Het schema weergeeft een start en een SCM Commit in de workflow waarbij code wordt gecreëerd en geupload naar een Repository. In dit schema is geen IDE functionaliteit benoemd en wanneer we de IDE zullen veranderen naar bijvoorbeeld een ander merk, wordt het gebruikersproces van Jenkins niet aangepast in het schema en kan exact de zelfde flow voor het Continuous Integration proces gehandhaafd worden.

Deelvraag 2: Wat maakt een tool binnen onze ontwikkelstraat van toegevoegde waarde en welke tools sluiten hier bij aan?

7.2 7.2.1 Inleiding

In deze deelvraag gaan wij alle tools selecteren die wij nodig zullen hebben in onze ontwikkelstraat. Op het moment dat dit onderzoek is uitgevoerd stond het vast de back-end programmeertaal die gebruikt zal worden C# is. Daarnaast is het bekend dat we gebruik zullen maken van AngularJS 2.0 voor de front-end. Tijdens het onderzoek naar deelvraag 1 was dit nog niet duidelijk, en zijn er dus een aantal tools opgesteld die niet kunnen worden gebruikt met C# en AngularJS 2.0.

7.2.2 Wat is C#?

C# is een door Microsoft gebouwde programmeertaal. Deze taal is multi-paradigma, wat inhoudt dat meerdere programmeerconcepten en technieken door elkaar kunnen worden gebruikt. Het is een van de grootste talen binnen het .NET framework. (Wikipedia: C# [6])

7.2.3 Wat is AngularJS?

AngularJS is een door Google opgezet Javascript framework wat gebruikt wordt voor het ontwikkelen van grote webapplicaties. Merk op dat wij gebruik maken van AngularJS 2.0, deze versie heeft namelijk veel verschillen met de 1.x versies. Angular 2.0 biedt een betere performance, en het is component-based. Dit houdt in dat er verschillende componenten worden ontwikkeld die kunnen worden hergebruikt in de applicatie.

(Wikipedia : AngularJS [7])

In dit onderzoek heb ik een selectie van de categorieën uit deelvraag 1 gemaakt, en hiervan de voor ons beste tools geselecteerd. Met behulp van informatie van een aantal blogpost over ontwikkelstraten heb ik de een lijst met de kernelementen van een ontwikkelstraat kunnen vormen (Matt Kopala, 2012 [8]). Deze is als volgt:

Continuous Integration
Source Code Management
Unit Testing
Build Automation

7.2.4 Continuous Integration

“Trust is the essential reason we need continuous integration.” via @settermjd

CLICK TO TWEET 


(Matthew Setter, 2016 [16])

Deze tweet is van de welbekende developer Robert Martin, ook wel genoemd als Uncle Bob. Uncle Bob zegt dat een van de meest voorkomende fouten bij software development is het verval van vertrouwen tussen de developers en de rest van de business.

Doordat bepaalde functies niet geïmplementeerd kunnen worden vanwege een aantal bugs kunnen deadlines niet behaald worden en brokkelt het vertrouwen in de developer langzaam af. En daarom is er Continuous Integration, om ervoor te zorgen de alle developers hun wijzigingen regelmatig (op zijn minst dagelijks) kunnen integreren in het systeem. Dit zorgt ervoor de software productie zijn momentum houdt en dat developers niet worden weerhouden van bugs in het systeem.

Op de website stackoverflow wordt meerdere malen verwezen naar Jenkins en TeamCity. In figuur 1 hieronder is een afbeelding weergegeven van een vergelijking tussen Jenkins en TeamCity. Wat voor ons vooral van belang is, is dat het platform gratis in gebruik te nemen is. Wij hebben geen budget, en kunnen TeamCity, wat begint bij €300, niet aanschaffen.

Hoewel Jenkins gratis is, wordt het momenteel door veel bedrijven gebruikt, waaronder Netflix, Dell en Facebook. (Larry Shatzer, 2016 [9]).

	 Jenkins	 TeamCity
usage:		
Free and Open Source	✓	✗
Widely Used	✓	✗
Well Documented	✓	✓
Easy to setup, use, and configure	✗	✓
Security (by default)	✗	✓
Email notification	✓	✓
Logging	✓	✓
Running builds for multiple branches dynamically	✓	✗
Individual validation	✗	✓
Inter-branch merges	✗	✗
Advanced Features	✓	✗
Port Flexibility	✗	✓
Overall	✓	✓

7.2.4.1 Conclusie

Continuous Integration zal ons een hoop tijd besparen. CI dwingt af om regelmatig nieuwe builds te maken en de wijzigingen van alle developers in het systeem te integreren.

Zonder CI zou een project al snel vastlopen vanwege bugs die in het systeem zitten.

De CI tool die het beste naar voren kwam was Jenkins. Jenkins is een tool die door veel grote bedrijven in gebruik wordt genomen en is daarnaast ook gratis.

7.2.5 Source Code Management (SCM)

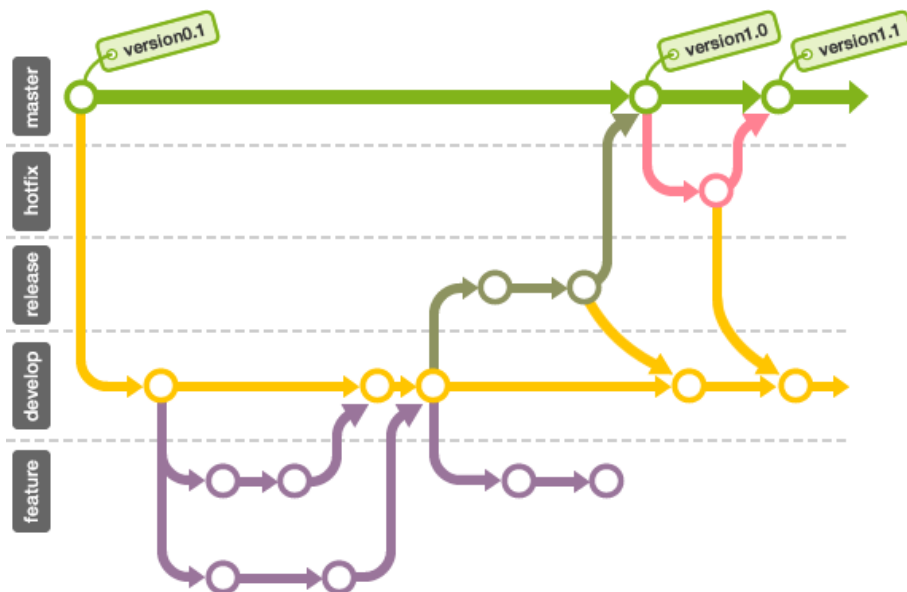
Wanneer je als software developer of als team aan een applicatie werkt wil je altijd ergens een back-up hebben staan van de code. Daarnaast geeft SCM de gebruiker meer inzicht in de code die andere gebruikers hebben gemaakt. Op deze manier kunnen developers elkaars code reviewen om eventuele bugs er al uit te kunnen halen.

In Source Code Management wordt onderscheid gemaakt tussen DVCS (distributed version control system) en CVS (Central Version System). De standaard workflow van een CVS is:

- Changes ophalen van de server die zijn gemaakt door andere gebruikers
- Changes toepassen en ervoor zorgen dat deze veranderingen goed werken.
- De changes 'committen' naar de centrale server, zodat andere gebruikers deze kunnen zien.

"These workflows help you to develop many parallel ideas, and to refactor your work quickly and easily and then get it into a state where you can share it with your team to get it ready for game time."

Van de website RhodeCode.com (Brian, 2014 [11]) heb ik dit citaat gehaald wat een goede beschrijving geeft van de flexibiliteit die DVCS de gebruiker biedt. Elke gebruiker kan zijn eigen branch aanmaken en daarin aanpassingen maken. Deze kan hij later weer samenvoegen met de centrale versie. Figuur 2, die hieronder is weergegeven, laat zien hoe verschillende branches worden gemaakt door alle gebruikers, en deze later weer worden samengevoegd met de centrale versie, de master.



Git Beginners Guide For Dummies [12]

CVS gaat ervan uit dat er een plek is waar alle source code moet komen te staan. DVCS daarentegen, maakt het mogelijk om alle gebruikers een eigen plek in te repository te geven om de source code te bewaren. Dit wordt gedaan met behulp van 'branches'. De gebruikers kunnen de code van de repository ophalen door de repository te 'clonen'. Hiermee wordt de repository met alle branches van de desbetreffende server opgehaald.

Omdat elke gebruiker over de branches van de andere gebruikers beschikt, is het makkelijk om bepaalde changes tussen deze branches toe te passen.

Daarnaast werkt DVCS ook offline. Dit maakt DVCS significant sneller dan CVS, omdat alle handelingen behalve pullen en pushen gebruik hoeven te maken van de remote server. DVCS heeft dan alleen toegang nodig tot de hard drive van de gebruiker.

De opdrachtgever, 42Windmills, maakt momenteel gebruik van de IDE Visual Studio. Hoewel er in Visual Studio een Team Foundation Server (TFS) applicatie zit geïntegreerd, maakt 42Windmills gebruik van de DVCS tool Git. Dit omdat Git een DVCS is en TFS een gecentraliseerd versie systeem.

Conclusie

DVCS biedt veel voordelen over CVS. DVCS maakt het committen van changes veel sneller vanwege zijn offline modus. Doordat er gebruik kan worden gemaakt van branches is de gebruiker minder afhankelijk van de centrale source code en dit maakt de workflow van DVCS een stuk prettiger dan CVS.

7.2.6 Automated Unit Testen

Voordat ik verder inga op de tools in deze categorie, ga ik antwoord geven op de vraag:

Welke bijdrage levert Automated Unit Testen aan de code kwaliteit?

The objective of automated testing is to simplify as much of the testing effort as possible with a minimum set of scripts. If unit testing consumes a large percentage of a quality assurance (QA) team's resources, for example, then this process might be a good candidate for automation. (Margaret Rouse, 2014 [13])

Het bovenstaande citaat geeft naar mijn mening een goede beschrijving van automated testing. Wanneer testen van de code een groot deel van de product kwaliteit waarborgt, is het misschien verstandig om gebruik te maken van automated testing.

Ik wil verder onderzoeken of automated testing daadwerkelijk de code kwaliteit verhoogt ten opzichte van handmatig testen.

First, of two major differences was that while many academic sources provide evidence that test automation increases fault detection, still 58% of the practitioners do not agree with this. This was further explained by the open questions where a practitioner pointed out that *it is the tester that facilitates high defect detection*, and that one can get high defect detection with both manual and automated testing depending how they are used. (Lund University, 2012 [14])

In dit onderzoek, van Lund University uit Zweden, naar de voor- en nadelen van automated testing, werd er een vergelijking gemaakt tussen wetenschappelijke artikelen over automated testing en de ervaringen van mensen in de praktijk. Dit onderzoek is gebaseerd op een selectie van 25 papers uit een verzameling van 24.706 papers, en 115 ervaringsdeskundigen.

Hoewel de wetenschappelijke artikelen aangeven dat test automation meer bug detectie verhoogd, 58% mensen met praktijkervaring het hier niet mee eens is. De ervaringsdeskundigen stelde dat het aan de tester zelf is om de tests veel bugs te laten detecteren, en dat dit bij zowel automated als handmatig testen kan worden gedaan. Dit is afhankelijk van hoe de tools worden gebruikt. Wel gaven de ervaringsdeskundigen aan, dat de herbruikbaarheid van de tests het testen een stuk makkelijker maakte. Ook kwam er naar voren dat de test coverage (de hoeveelheid code die getest wordt) toeneemt.

Ik kan uit dit onderzoek concluderen dat automated testing de code kwaliteit verhoogt, mits de tests zorgvuldig worden geschreven om zoveel mogelijk bugs te detecteren.

Binnen C# zijn er twee test frameworks die duidelijk boven de rest uitspringen. Dat zijn NUnit en MS Test. Op Stack Overflow heb ik de meningen van de gebruikers van beide tools bekeken (Stackoverflow : NUnit vs MSTest [15]). Het volgende kwam naar voren:

- NUnit heeft een mocking framework
Mocken is een ander woord voor imiteren. Een mockingframework imiteert het gedrag van een bepaalde module en dit kan gebruikt worden om tests op uit te voeren. Mocking maakt het mogelijk om modules te testen zonder afhankelijk te zijn van andere dependencies.
- De gebruikers hebben ervaren dat MS Test relatief langzaam is vergeleken met NUnit. Bij het runnen van 3800 tests neemt MS Test 1GB aan ram in beslag, wat vaak regelmatig leidt tot OutOfMemoryExceptions.
- NUnit heeft veel updates, MS Test wordt alleen geupdate wanneer Visual Studio wordt geupdate.

Voor AngularJS zijn er twee test frameworks beschikbaar. Deze twee tools hebben allebei een verschillend doel. Dit zijn Jasmine en Protractor. Jasmine wordt gebruikt om unit tests uit te voeren, en Protractor wordt gebruikt voor end to end testing. End to end testing houdt in dat de test worden uitgevoerd vanuit het perspectief van de eindgebruiker.

7.2.6.1 Conclusie

Met deze gegevens kan ik tot de conclusie komen dat NUnit een hoop voordelen biedt tegenover MS Test. Wat vooral doorslaggevend is, is de goede performance van NUnit, ofwel de slechte performance van MS Test. Wanneer de code zullen gaan testen, willen wij niet dat het test framework de helft van ons RAM geheugen in beslag neemt, wat ook nog eens leidt tot exceptions. Daarom zullen wij voor de ontwikkelstraat NUnit kiezen boven MS Test. Voor front-end testing zullen wij gebruik maken van Jasmine en Protractor.

Jasmine is gericht op het unit testen, en Protector op end to end testen.

7.2.7 Build Automation

Voor het Build proces zijn er in C# twee bekende tools, NAnt en MSBuild.

Een gebruiker op stackoverflow die met zowel NAnt als MSBuild gewerkt heeft meldde het volgende:

“ NAnt is quite straightforward and you-get-what-you-see. MSBuild on the other hand requires a bit more thought. The learning curve is steeper. But once you "get it", you can do some amazing things with it.”

Een citaat van een andere gebruiker:

“NAnt has more features out of the box, but MSBuild has a much better fundamental structure (item metadata rocks) which makes it much easier to build reusable MSBuild scripts.”

Uit deze citaten van mensen met praktijkervaring is te zeggen dat MSBuild meer tijd nodig heeft om goed in gebruik te kunnen nemen dan NAnt. De tijd die je in MSBuild investeert zal uiteindelijk lonen. Waarom dit zo is wordt niet vermeld.

Met deze informatie kon uiteindelijk weinig doen en heb ik nog geen tool kunnen vaststellen voor het gebruik van build automation.

MSBuild is al geïntegreerd in de IDE die wij gebruiken (Visual Studio), waardoor het makkelijker is om MSBuild in gebruik te nemen. Echter is volgens Google Trends meer informatie te vinden over Nant dan over MSBuild. Hier hebben wij uiteindelijk op beoordeeld om gebruik te maken van de build tool Nant.

7.2.8 Conclusie

Wat maakt een tool binnen onze ontwikkelstraat van toegevoegde waarde en welke tools sluiten zich hierbij aan?

Uit het onderzoek wat voor deze deelvraag 2 is uitgevoerd, kan ik concluderen dat wij gebruik zullen gaan maken van:

- Continuous Integration (CI). Dit is een must om de productie van het te ontwikkelen systeem op gang te brengen. Voor CI zal de tool Jenkins gebruikt worden;
- Source Code Management. Dit is nodig om backups te hebben van de code en om code reviews uit te voeren. Hiervoor zal Git worden gebruikt.
- Automated Testing. Dit zal de projectgroep een hoop tijd besparen boven handmatig testen. Hiervoor zullen wij NUnit gebruiken.
- Build Automation: Voor Build Automation zullen wij gebruik maken van Nant.

In welke mate verbetert een ontwikkelstraat de kwaliteit van code?

7.3.1 Inleiding

Deze deelvraag gaan we proberen te beantwoorden gebruikmakend van een door ons opgezet experiment. Voordat we de deelvraag kunnen beantwoorden moet eerst de definitie van code kwaliteit vastgesteld worden, dit gebeurt in hoofdstuk 2. Hierna gaan we behandelen wat we met ons experiment willen vaststellen en wordt er een hypothese geformuleerd in hoofdstuk 3.

In hoofdstuk 4 gaan we vaststellen welke waarden we willen meten in het experiment, welke wij vervolgens in hoofdstuk 5 daadwerkelijk gaan meten door het uitvoeren van het experiment.

Uiteindelijk gaan wij de resultaten in hoofdstuk 6 analyseren om in hoofdstuk 7 af te sluiten met een conclusie en door de deelvraag te beantwoorden.

7.3.2 Definitie Code Kwaliteit

A good way to treat an elusive concept, such as the quality of software, is to decompose it into finergrained attributes and decompose them again, until we get tired of the process, or, preferably, we reach a level at which we can base our discussion on meaningful examples that share common

characteristics. The ISO/IEC 9126 standard, which defines the quality model we will use in our discussion, categorizes internal and external software quality characteristics into six major areas: functionality, reliability, usability, efficiency, maintainability, and portability. (Spinellis, Code Quality: The Open Source Perspective [1])

Volgens Spinellis zou het een goede methode zijn om een ongrijpbaar begrip als code kwaliteit te behandelen door het op te delen in kleinere componenten. In 2011 heft de ISO norm 25010 de ISO norm 9216 vervangen als de standaard voor de kwaliteitskenmerken van software. (ISO 25010:2011 [2]) De ISO norm 25010 verdeelt de componenten van code kwaliteit over acht categorieën, die verder zijn verdeelt over 31 kwaliteitseigenschappen zoals te zien valt in onderstaande afbeelding:



7.3.3 Doelstelling Experiment

De doelstelling van het experiment is om een experiment te bedenken, op te zetten en uit te voeren om de hieronder geformuleerde hypothese te bevestigen.

De hypothese bij dit experiment is dat er een meetbaar causaal verband is tussen het implementeren van een ontwikkelstraat bestaande uit de componenten zoals beschreven in de tweede deelvraag van ons onderzoeksplan binnen een bestaand software project, en de onderhoudbaarheid van code zoals vastgesteld in de ISO norm 25010:2011.

We hebben ons bij dit experiment beperkt tot het analyseren van de onderhoudbaarheid zoals beschreven in de ISO norm 25010 omdat het anders niet in het tijdsbestek van dit onderzoek past. We hebben gekozen voor de kwaliteitscomponent onderhoudbaarheid omdat deze het best aansluit op het onderzoek naar de samenhang tussen een ontwikkelstraat en code kwaliteit; de opdrachtgever heeft ook duidelijk gemaakt dat er vraag is naar verbetering in testbaarheid van de code, wat een kwaliteitseigenschap is van de component onderhoudbaarheid.

Zoals eerder beschreven heeft de ISO norm 25010 de kwaliteitscomponenten onderverdeelt over meerdere kwaliteitseigenschappen. De kwaliteitscomponent onderhoudbaarheid is opgesplitst tot de onderstaande vijf kwaliteitseigenschappen:

- 1) Modulariteit
- 2) Herbruikbaarheid
- 3) Analyseerbaarheid
- 4) Wijzigbaarheid
- 5) Testbaarheid

Per kwaliteitseigenschap wordt bekeken of deze voor ons binnen het experiment meetbaar is, en of deze meetresultaten een meerwaarde hebben voor ons onderzoek; mocht de kwaliteitseigenschap meetbaar blijken wordt er bekeken hoe dit gemeten gaat worden.

7.3.3.1 Modulariteit

Modularity is the property of a system that has been decomposed into a set of cohesive and loosely coupled modules. Thus, the principles of abstraction, encapsulation, and modularity are An object provides a crisp boundary around a single abstraction, and both encapsulation and modularity provide barriers around this abstraction. (Booch, Object-Oriented Analysis and Design [3])

Modulariteit blijkt een redelijk abstract wederom een abstract begrip, als we dichterbij de opdrachtgever zijn wens kijken, het implementeren van een ontwikkelstraat voor applicaties merendeel geschreven in C# komen we tot de volgende, wat concretere definitie van modulariteit:

A modular application is divided into functional units named modules that can be integrated into a larger application. Each module provides a portion of the application's

overall functionality and represents a set of related concerns.(MSDN, Chapter 5 Modularity [4])

De mate van modulariteit binnen een C# applicatie blijkt te zitten in de hoeveelheid modules binnen een project.

7.3.3.2 Herbruikbaarheid

Reusability is the degree to which a component can be reused, and reduces the software development cost by enabling less writing and more assembly.(Washizaki, A Metrics Suite for Measuring Reusability of Software Components [5])

De definitie van herbruikbaarheid lijkt minder abstract en minder complex te zijn dan de kwaliteitseigenschap modulariteit, niks is minder waar als we kijken naar de elementen waaraan gedacht moet worden om de mate van herbruikbaarheid aan te kunnen duiden.

According to the above-mentioned reusability model, we define five metrics: existence of meta information, rate of component observability, rate of component customizability, self-completeness of component's return value and self completeness of component's parameter.(Washizaki, A Metrics Suite for Measuring Reusability of Software Components[5])

Door de mate van complexiteit sluiten we de kwaliteitseigenschap herbruikbaarheid omwille van de scope van het onderzoek uit van het experiment. Om de mate van herbruikbaarheid vast te stellen van een stuk software adviseren wij het onderzoek van Washizaki te lezen of een specifiek onderzoek op te zetten.

7.3.3.3 Analyseerbaarheid

De mate waarin het mogelijk is om effectief en efficiënt de impact, van een geplande verandering van één of meer onderdelen, op een product of systeem te beoordelen, om afwijkingen en/of foutoorzaken van een product vast te stellen of om onderdelen te identificeren die gewijzigd moeten worden.(ISO 25010:2011, [2])

De mate van analyseerbaarheid zullen wij vaststellen door de benodigde tijd om een fout of wijziging in een stuk code op te sporen te meten.

7.3.3.4 Wijzigbaarheid

De mate waarin een product of systeem effectief en efficiënt gewijzigd kan worden zonder fouten of kwaliteitsvermindering tot gevolg.(ISO 25010:2011, [2])

De mate van wijzigbaarheid zullen wij vast stellen door de benodigde tijd om een nieuwe functionaliteit aan de code toe te voegen te meten. De tijd die wordt gemeten is het punt vanaf het implementeren van de module tot het punt dat het systeem weer succesvol draait inclusief de nieuwe module. Het ontwikkelen van de module valt niet onder de gemeten tijd nog onder de mate van wijzigbaarheid van de code.

7.3.3.5 Testbaarheid

De mate waarin een stuk software getest kan worden, afhankelijk van de modulariteit en de grootte van methoden.

7.3.4 Opzet Experiment

7.3.4.1 Opzet

Nu er in kaart is gebracht wat we willen gaan meten om de hypothese te bevestigen gaan we uiteenzetten hoe dit precies gedaan gaat worden.

Wij nemen een code base die gemaakt is zonder ontwikkelstraat welke wij dupliceren, bij het duplicaat sluiten wij een selectie van tools uit deelvraag twee aan. Vervolgens gaan wij bij beide code bases op verschillende manieren nieuwe stukken code toevoegen, om vervolgens zoals in hoofdstuk drie gedefinieerd de code onderhoudbaarheid van beide code bases te meten.

Na het meten zullen de resultaten hiervan vergeleken worden en geanalyseerd worden. Naar verwachting zal er verschil zijn tussen de metingen van de code base met een aangesloten ontwikkelstraat en de code base zonder aangesloten ontwikkelstraat.

7.3.4.2 De Gebruikte Code Base

De code base die wij gebruiken bij dit experiment is die van het IPSEN 4 project aangezien dit een project is die lijkt op dat van de opdrachtgever. Zo bestaat het uit een client server architectuur, met een REST server. De server is geschreven in een object-georiënteerde taal en bestaat de front-end uit een AngularJS applicatie; de hele applicatie gemaakt zonder ontwikkelstraat. De code base is gemaakt door Anton Steenvoorden en mij, Roy Touw.

7.3.4.3 Uitvoering Metingen

De metingen zullen gefaseerd plaatsvinden omdat de verwachting is dat kwaliteitseigenschappen per type uitgevoerde mutatie op de code base anders reageren. Er zijn twee fasen, de eerste fase is het toevoegen van een nieuwe werkende reeds gemaakte functionaliteit, het gaat in deze fase puur om de implementatie van nieuwe code. De tweede fase bestaat uit het opzettelijk implementeren van bugs en niet werkende code.

7.3.4.4 Eerste Fase

De wijzigbaarheid zal gemeten worden door te meten hoeveel tijd het kost om de nieuwe functionaliteiten te implementeren. Er worden een aantal nieuwe functionaliteiten toegevoegd, per toevoeging komt er een tijd uit in minuten. De tijd die gemeten wordt is de start van de implementatie tot de applicatie inclusief de nieuwe functionaliteit weer werkt, het ontwikkelen van de nieuwe functionaliteit staat buiten dit experiment en zal dus niet bij de meting meegenomen worden.

De modulariteit zal gemeten worden door na het implementeren van de nieuwe functionaliteiten het aantal modules in de code bases te tellen.

7.3.4.5 Tweede Fase

In fase twee zullen er een aantal bugs en niet werkende stukken code geïmplementeerd worden; de kwaliteitseigenschappen die gemeten gaan worden in deze fase zijn analyseerbaarheid en de testbaarheid. Er zijn drie groepen personen die meewerken aan deze fase, een groep die zich bezighoudt met het implementeren van foutieve code, een groep die zich bezighoudt met het oplossen binnen de code base die

aangesloten is op een ontwikkelstraat en een groep die zich bezighoudt met het oplossen binnen de code base die aangesloten is op een ontwikkelstraat. Deze groepen zijn gemaakt zodat de oplossende persoon niet bij aanvang weet waar het stuk foutieve code zit.

De analyseerbaarheid zal gemeten worden door de benodigde tijd om het foutief werkende stuk code te achterhalen en op te lossen.

7.3.4.6 De Ontwikkelstraat

7.3.5 Uitvoering Experiment

7.3.5.1 Opzet Code Base en Implementatie Ontwikkelstraat

Voordat de mutaties en metingen kunnen beginnen moeten eerste de code bases gereed worden gemaakt. De gekozen code base staat op een git repository, deze wordt naar twee ontwikkelomgevingen gepulled. Hierna wordt er in een ontwikkelomgeving de gekozen ontwikkelstraat aangesloten op de code base.

7.3.5.2 Uitvoering Eerste Fase

De functionaliteit die is toegevoegd is het laten zien van de boekingen die een gebruiker heeft gedaan. De module is reeds gemaakt en het gaat louter om een module reeds ontwikkeld is te implementeren.

Uitvoering bij Dennis kwam afgerond tot 6 minuten gebruikmakend van een ontwikkelstraat. Uitvoering van Roy zonder ontwikkelstraat kwam afgerond tot 11 minuten. Het tijdsverschil kwam voornamelijk door een foutmelding die moeilijk te achterhalen was zonder de uitgevoerde unit tests.

7.3.5.3 Uitvoering Tweede Fase

Bij de tweede fase wordt er een module toegevoegd met het registreren tijdens het boeken, hierin zit expres een paar fouten. Het gaat hierom dat deze fouten opgelost worden zodat de functionaliteit gebruikt kan worden binnen de TravelPlanner.

Bij Roy gebruikmakend van de ontwikkelstraat was er 7 minuten nodig om de fouten op te sporen. Bij Dennis zonder ontwikkelstraat duurde het ruim een kwartier om de fouten op te lossen, dit kwam voornamelijk doordat de foutmeldingen moeilijk thuis te brengen waren en er hulp nodig was van Stack Overflow.

7.3.6 Analyse Resultaten

Uit het uitgevoerde experiment komt het vermoeden naar voren dat een ontwikkelstraat zorgt voor een hogere code kwaliteit en een snellere ontwikkeling. Ook blijkt dat de codestijl hoger is gebruikmakend van de ontwikkelstraat door de feedback van SonarCube, waar dingen als een leeg commentaar zonder ontwikkelstraat onopgemerkt is gebleven.

7.3.7 Conclusie en Beantwoording

De conclusie is dat het vermoeden dat de code kwaliteit omhoog gaat door het gebruik van een ontwikkelstraat bevestigd lijkt te zijn, echter hebben wij niet genoeg personen tot ons beschikking om het tot een goed bewijs te vormen.

Deelvraag 4: Is het voor ons project het waard om een ontwikkelstraat op te zetten?

7.4.1 Inleiding

In dit hoofdstuk wordt de vierde deelvraag behandeld. Voordat wij deze deelvraag 7.4 kunnen beantwoorden moeten wij eerste weten wat voor ons de criteria is om een ontwikkelstraat het waard te laten zijn. Voor ons heeft een ontwikkelstraat een meerwaarde wanneer deze de efficiëntie van manuren vergroot, de code kwaliteit vergroot of de beheerbaarheid vergroot. Om een antwoord te geven op deze deelvraag wordt deze opgedeeld over de drie criteria en worden deze los beantwoord, vervolgens wordt er naar deze drie antwoorden gekeken om een conclusie bij deze deelvraag te vormen.

7.4.2 Beheerbaarheid

Hieronder heb ik een drietal punten beschreven die van toepassing zijn op de beheerbaarheid.

Deze punten komen ook voor in een whitepaper over code manageability van National Instruments. (National Instruments, What is manageability?, 2015)[17] Aangezien het travelplanner project nog in een vroege fase zit, is het moeilijk om nu al te zeggen of de ontwikkelstraat de beheerbaarheid zal verbeteren. Maar wanneer de applicatie in de toekomst zal uitbreiden aan functionaliteiten, dan zal het volgens het onderzoek een positief effect hebben op de beheerbaarheid.

Configuratie

De configuratie van de ontwikkelstraat is van essentieel belang voor optimale productiviteit en availability. De configuratie van de ontwikkelstraat heeft onze projectgroep een hoop tijd gekost, maar wanneer de ontwikkelstraat eenmaal juist geconfigureerd is, zal er in de toekomst niet of nauwelijks tijd geïnvesteerd hoeven te worden in de configuratie. Afhankelijk van de hoeveelheid tools die gebruikt worden spelen rekenkracht en opslaggeheugen spelen een steeds belangrijkere rol in de configuratie.

Health Monitoring en Loggen

Door middel van het gebruik van Sonar Qube is het een stuk kunnen loggegevens en codeanalyses weergegeven worden. Dit maakt het voor de developers een stuk makkelijker om bepaalde bugs op te lossen of te voorkomen.

Deployment en Updates

De projectgroep heeft de applicatie nog nooit daadwerkelijk gedeployed, dus hier kunnen wij geen oordeel over geven.

7.4.3 Productiviteit

De productiviteit is een belangrijk onderwerp in het gebruik van een ontwikkelstraat. Wanneer de productiviteit van een ontwikkelstraat achteruit gaat zal de toegevoegde waarde van een ontwikkelstraat nihil zijn. Aan de hand van het berekenen van de hoeveelheid tijd die we er aan kwijt zijn kunnen we bepalen of de productiviteit in ons geval toeneemt bij het gebruik van een ontwikkelstraat.

We verwachten ongeveer 30 uren op te leveren bij het gebruik van een ontwikkelstraat binnen een periode van 8 weken. We zijn bij de ontwikkeling van de ontwikkelstraat 70 uur kwijt geweest aan het ontwikkelen. Het project duurt ongeveer 8 weken waarbij er 150 uur aan ontwikkeltijd besteed kan worden per persoon. We zijn met 3 teamleden en hebben dus een totaal van $150 * 3 = 450$ uren aan ontwikkeltijd. Dit betekent dat hier van 70 uur verloren gaat aan ontwikkeling van de ontwikkelstraat gedurende dit project, hier houden we $450 * 70 = 380$ uren over om te ontwikkelen. Omdat de investering van 70 uur en de opbrengst van 30 uur over een periode van 10 weken niets oplevert, kunnen we stellen dat de productiviteit in ons geval niet verbeterd maar verslechterd. Het gebruiken van een ontwikkelstraat die nog niet opgezet is heeft voor dit project van ongeveer 8 weken dus geen toegevoegde waarde in het opzicht van productiviteit.

Mochten er verder gewerkt worden met de TravelPlanner zou het op den duur meer tijd kunnen gaan besparen dan dat het gekost heeft.

7.4.4 Code Kwaliteit

Om te bepalen of de code kwaliteit van de code omhoog gaat na de implementatie van een ontwikkelstraat zoals vastgesteld in de eerste twee deelvragen te vinden in respectievelijk §7.1 en §7.2 wordt de definitie code kwaliteit gebruikt zoals beschreven in deelvraag drie te vinden in §7.3.2.

Deze criteria van de concluderende laatste deelvraag bestaat uit het antwoord op de derde deelvraag.

Na aanvang van het uitgevoerde experiment en de analyse van de resultaten is het aannemelijk te stellen dat de implementatie van een ontwikkelstraat zorgt voor een grotere code kwaliteit binnen dit project.

7.4.5 Conclusie

Kijkende naar de criteria behandeld in dit hoofdstuk kunnen we concluderen dat de ontwikkelstraat binnen ons project meer tijd heeft gekost dan dat het opgeleverd heeft; gaan we er echter van uit dat er nog meer functionaliteiten bij de TravelPlanner komen, wat zeer goed mogelijk is, dan zou de ontwikkelstraat zeker een meerwaarde zijn vooral door de verhoogde code kwaliteit en de tijd die met de beheerbaarheid wordt bespaard. Ook is er aangetoond dat de code kwaliteit wel daadwerkelijk omhoog is gegaan, de afweging is nu dan ook of de verhoogde code kwaliteit zwaarder weegt dan de extra tijd die het heeft gekost voor het project om de ontwikkelstraat te ontwerpen en implementeren. Naar onze mening zou er bij een in gebruik name van de applicatie de code kwaliteit zwaarder wegen dan de verloren uren, en daarmee zou de ontwikkelstraat binnen dit project voor een meerwaarde zorgen.

8 CONCLUSIE EN DISCUSSIE (DEELVRAGEN CONCLUSIES EN EEN EINDCONCLUSIE)

De hoofdvraag was hoe kunnen wij als software ontwikkelteam zorgen voor een hogere code kwaliteit voor de opdrachtgever gebruikmakend van een ontwikkelstraat? Aan de hand van de eerste twee deelvragen zijn we gekomen tot wat een ontwikkelstraat nou is, en welke componenten wij zouden kunnen gebruiken. Uit de derde deelvraag kwam wat de definitie van software kwaliteit nou precies is en werd er een causaal verband aangetoond tussen de software kwaliteit en het gebruik van een ontwikkelstraat.

In de vierde deelvraag kwam naar voren dat er naast een eventuele tijdsbesparing na een hoeveelheid gespendeerde uren aan een project er ook een verhoging van de beheerbaarheid en de code kwaliteit aangetoond is.

Deze vier deelvragen komen dan ook tot de conclusie dat met het implementeren van een ontwikkelstraat bestaande uit de componenten zoals in dit document beschreven een manier is om als ontwikkelteam te zorgen voor een verhoogde code kwaliteit binnen een applicatie. Hiermee hebben wij de hoofdvraag kunnen beantwoorden, wel moet er in overweging genomen worden dat wij een relatief klein project hebben gebruikt, en dat er bij verschillende deelvragen naar voren is gekomen dat verder onderzoek gewenst is.

9 EVALUATIE

Het onderzoek kwam vrij langzaam van de grond omdat er aan het begin nog veel onduidelijkheden waren. Een groot deel van de tijd zijn wij bezig met het opstellen en aanpassen van de deelvragen. Het grootste deel van dit onderzoek was literatuuronderzoek, en er is een experiment uitgevoerd. Het selecteren van de tools was af en toe wat lastig. Vaak moesten wij het dan hebben van meningen van andere gebruikers op stackoverflow. Wij baseren onze argumenten liever op informatie uit wetenschappelijke artikelen, maar deze waren vaak niet te vinden over specifieke tools.

10 AANBEVELINGEN

Omdat het experiment uitgevoerd is met een relatief kleine code base met eveneens weinig testgevallen voortkomend doordat wij deze met twee mensen uitgevoerd hebben is het aannemelijk dat de code kwaliteit vergroot is, maar zeker niet bewezen.

Verder hebben wij het onderzoek uitgevoerd met de tools en componenten die wij nu tot handen hebben, aangezien de ontwikkelingen binnen deze sector snel gaan zal er bij nieuwe technieken en nieuwe tools uitbreidend onderzoek nodig zijn om deze wijzigingen mee te nemen.

11 SUGGESTIES VOOR VERDER ONDERZOEK

Gedurende het onderzoek naar de code kwaliteit is er naar voren gekomen dat de code kwaliteit een te breed begrip is voor de scope van dit project, om deze reden is er gekozen om een deel van het begrip code kwaliteit te onderzoeken, zijnde de onderhoudbaarheid van de code. Verder onderzoek kan kijken naar de overige deelgebieden van code kwaliteit en de relatie tussen deze deelgebieden en een ontwikkelstraat.

Omdat wij er van uitgaan dat de technologische ontwikkeling binnen de informatica niet zal stoppen verwachten wij dat er nieuwe componenten voor de ontwikkelstraat gemaakt zullen worden, daarom zou er met de tijd nieuw onderzoek moeten komen naar deze nieuwe componenten om te zien of de conclusie van de eerste twee deelvragen nog steeds volstaan.

12 LITERATUUR

- [1] Code Quality: The Open Source Perspective, Diomidis Spinellis, isbn 0-321-16607-8, §1.1.2.
- [2] ISO 25010:2011
- [3] Object-Oriented Analysis and Design, Grady Booch, isbn 0-8053-5340-2, §2.2.
- [4] MSDN, Chapter 5 Modularity, 06-10-2016, <https://msdn.microsoft.com/en-us/library/hh404079.aspx>.
- [5] A Metrics Suite for Measuring Reuasability for Software Components, Proceedings of the Ninth International Software Metrics Symposium (METRICS'03) 1530-1435/03.
- [6] https://nl.wikipedia.org/wiki/C_#
- [7] <https://nl.wikipedia.org/wiki/AngularJS>
- [8] Building a delopment infrastructure
<http://mattkopala.com/blog/2012/01/08/building-a-development-infrastructure/>
- [9] Lijst met bedrijven die jenkins gebruiken.
<https://wiki.jenkins-ci.org/pages/viewpage.action?pageId=58001258>
- [10] Link naar een pagina Jenkins vs TeamCity
<https://www.linkedin.com/pulse/jenkins-vs-teamcity-amir-najjar>
- [11] DVCS
<https://rhodecode.com/blog/62/why-embrace-distributed-version-control-systems>
- [12] Plaatje branches
https://backlogtool.com/git-guide/en/stepup/stepup1_5.html
- [13] What is automated testing
<http://searchsoftwarequality.techtarget.com/definition/automated-software-testing>
- [14] Onderzoek automated testing
http://mikamantyla.eu/ast_2012_bare_conf.pdf
- [15] NUnit vs MS Test
<http://stackoverflow.com/questions/1554018/unit-test-nunit-or-visual-studio>
- [16] Continuous Integration
<https://blog.codeship.com/continuous-integration-important/>

- [17] What is Manageability? – Whitepaper
<http://www.ni.com/white-paper/14415/en/>
- [18] Continue Delivery catogires – Kurt Bittner
http://blogs.forrester.com/kurt_bittner/15-01-12-continuous_delivery_pipeline_tool_categories
- [19] Infrastructure automatation
<http://thenewstack.io/exploding-infrastructure-automation-stack-ecosystem/>
- [20] What is a deployment automation tool
<https://www.youtube.com/watch?v=z3uwjGxmM6g> ,6:10, What is a deployment automation tool and how can it help me
- [21] IaaS
https://nl.wikipedia.org/wiki/Infrastructure_as_a_service
- [22] Microservices
<http://www.martinfowler.com/articles/microservices.html>
- [23] Go Ci
www.Go.cd
- [24] Buildbot
www.buildbot.net
- [25] Circle Ci
www.circleci.com
- [26] CloudBees
www.CloudBees.com
- [27] Snap Ci
www.snap-ci.com
- [28] JetBrains
www.jetbrains.com/teamcity
- [29] Shippable
www.shippable.com
- [30] Git
www.git-scm.com
- [31] Mercurial
www.mercurial.com
- [32] Bitbucket
www.bitbucket.com

- [33] Subversion
www.subversion.apache.org
- [34] Maven
www.maven.apache.org
- [35] NUnit
www.nunit.com
- [36] Chef
www.chef.io

13 BIJLAGEN

Stukken die niet in je onderzoeksrapport past kan je in de bijlagen plaatsen. Denk hierbij aan vragenlijsten, observatieschema's, codeerschema's, brieven aan respondenten, etc.

Configuration management software vergelijking

13.1

	Language ↕	License ↕	Mutual auth ↕	Encrypts ↕	Verify mode ↕	Agent-less ↕	Have a GUI ↕	First release ↕	Latest stable release ↕
Ansible	Python	GPLv3+	Yes ^[1]	Yes ^[2]	Yes	Yes	Yes ^[3] (Free 30-day Trial)	2012-03-08	2016-11-01 2.2.0.0 ^{[4][5]}
Bcfg2	Python	BSD 2-clause ^[6]	Yes ^[7]	Yes ^[8]	Yes ^[9]	No	Yes ^[10]	2004-08-11 ^[11]	2014-09-05 1.3.5 ^[11]
Capistrano	Ruby	MIT License		Yes ^[2]				2005	2015-03-02 3.4.0
cdist	Python	GPLv3+	Yes ^[1]	Yes ^[2]		Yes		2010	2015-03-19 3.1.12
CFEngine	C	GPLv3 ^[18]	Yes ^[1]	Yes ^[19]	Yes ^{[20][21]}	No			
Chef	Ruby, Erlang	Apache 2.0	Yes ^[12]	Yes ^[13]	Yes ^{[14][15]}	No	Yes	2009-01-15 0.5.0	2016-03-07 12.8.1 (client), ^[16] 2016-02-04 12.4.1 (server) ^[17]
ISconf	Python	GPL ^[22]	Yes ^[23]	No ^[24]				1998	2006-08-13 4.2.8.233
Juju	Python, Go ^[25]	Affero General Public License	Yes ^[1]	Yes ^[8]	No	No	Yes ^[26]	2010-09-17 ^[27]	2015-06-17 1.24.0 ^[28]
Local ConFiGuration system (LCFG)	Perl	GPL	Partial ^[29]	Partial ^[30]				1994	Weekly Releases
NOC	Python	BSD	Yes ^[1]	Yes ^[2]	Yes	Yes	Yes	2012-03-08	2015-05-20 15.05.1 ^[31]

Open pc server integration (Ops)	Python, Java	GPL	No	Yes ^[8]		No		2004	2013-03-01 4.0.3
PIKT	C	GPLv2+ ^[34]	Yes ^[35]	Yes ^[36]		No		1998 ^[37]	2007-09-10 1.19.0
Puppet	Ruby	Apache from 2.7.0, GPL before then	Yes ^[38]	Yes ^[8]	Yes ^{[39][40]}	No	Yes ^[41]	2005-08-30 ^[42]	2016-03-16 4.4.0 ^[43]
Quattor	Perl, Python	Apache 2.0 ^{[44][45]}	Yes ^[46]	Yes ^[47]				2005-04-01 ^[48]	2017-01-04 16.12.0 ^[49]
Radmind	C	BSD ^[50]	Yes ^[51]	Yes ^[52]		No		2002-03-26 ^[53]	2008-10-08 1.13.0 ^[54]
Rex	Perl	Apache	Yes ^[1]	Yes ^[2]		Yes		2010-11-05 0.9.0 ^[55]	2015-09-04 1.3.3 ^[56]
Rudder	C and Scala	GPL and Affero GPL	Yes ^[1]	Yes ^[8]	Partial	No	Yes	2011-10-31	2015-08-19 3.1.1 ^[57]
Salt^[61]	Python ^[62]	Apache 2.0 ^[63]	Yes ^[64]	Yes ^[64]	Yes	Both ^{[65][66]}	Yes ^{[67][68]}	2011-03-17 0.6.0 ^[69]	2016-06-08 2016.3.1 ^[70]
SmartFrog	Java	LGPL	Yes ^[58]	Yes ^[58]		No		2004-02-11	2009-01-26 3.16.004 ^{[59][60]}
Spacewalk	Java (C, Perl, Python, PL/SQL)	GPLv2	Yes	Yes		No		2008-06 ^[71]	2016-11-29 2.6 ^[72]

https://en.wikipedia.org/wiki/Comparison_of_open-source_configuration_management_software

13.2 Configuration management software vergelijking

Name	Platform	License	Builders: Windows	Builders: Java	Builders: other	Notification	Integration, IDEs	Integration, other
AnthillPro	Cross-platform	Proprietary	MSBuild, NAnt, Visual Studio	Ant, Maven 1-2-3	Shell script, batch script, cross-platform command-line, Groovy, Make, RTC Jazz, TFS Build, Custom Script Interpreter	Email, XMPP, RSS, Systray	Eclipse, Visual Studio	Many
Apache Continuum	JDK, web container	Apache 2.0	Unknown	Maven 1-2-3	Shell script ^[2]	Mail, XMPP and Google Talk, MSN, IRC, report deployment with wagon	Unknown	Unknown
Apache Gump	Python	Apache 2.0	Unknown	Ant, Maven 1	Unknown	Email	Unknown	Unknown
AppVeyor	Hosted	Proprietary	Visual Studio, MSBuild, Psake	No	Custom Script, PowerShell	Email, HipChat, Slack, Catlight	No	GitHub, Bitbucket, Kln, Windows Azure
Assertible	Hosted	Proprietary	Unknown	Unknown	Unknown	Email, Slack, Web, Webhooks	No	Many
Bamboo	Web container	Proprietary	MSBuild, ^[3] NAnt, ^[4] Visual Studio ^[5]	Ant, ^[6] Maven 1-2-3 ^[7]	Custom script, command-line tool, Bash, Xcode, ^[8] Phing, ^[9] Grunt	XMPP, Google Talk, Email, RSS, Remote API, HipChat	IntelliJ IDEA, Eclipse, Visual Studio	FishEye, Jira, Clover, Bitbucket, GitHub
BuildBot	Python	GPL	Command-line	Command-line	Command-line	Email, Web, GUI, IRC	Unknown	Unknown
Buddy	Cross-platform, Hosted	Proprietary	Yes	Ant, Maven, Gradle, Android	C, C++, Clojure, Dart, Elixir, Erlang, Go, Groovy, Haskell, Node.js, PHP, Python, Rake, Ruby, Scala, Shell Script, Command-Line	Email, GUI, Slack, SMS, Webhook	No	AWS, Bitbucket, DigitalOcean, Docker, Elastic Beanstalk, GitHub, GitLab, GCE, Heroku, Microsoft Azure, Modulus, Shopify
BuildMaster	Cross-platform	Proprietary	Yes	Yes	Cross-platform command-line	Email, custom	No	Many
CABIE	LAMP	GPL2	Unknown	Unknown	Unknown	Web	Unknown	Unknown
CircleCI	Hosted-enterprise	Proprietary	No	Yes	Go, Ruby, Python, Node.js, PHP, Java	Email, Slack, Campfire, HipChat, CClray	Unknown	AWS, Heroku, GitHub, Slack
Codship	Hosted	Proprietary	No	Yes	Go, Java, Node.js, PHP, Python, Ruby	Email, Flowdock, Grove, HipChat, Slack, web	No	CloudControl, Engine Yard, GitHub, Heroku, Amazon Web Services, Microsoft Azure

Jenkins-Hudson	Web container	Creative Commons and MIT	MSBuild, NAnt	Ant, Maven 2, Kundo	Cmake, Gant, Gradle, Grails, Phing, Rake, Ruby, SCons, Python, shell script, command-line	Android, Email, Google Calendar, IRC, XMPP, RSS, Twitter, Slack, Catlight, CCMenu, CCTray	Eclipse, IntelliJ IDEA, NetBeans	Bugzilla, Google Code, Jira, Bitbucket, Redmine, FindBugs, Checkstyle, PMD and Mantis, Trac, HP ALM
LuniBuild	Web container	Apache 2.0	Unknown	Ant, Maven 1-2	Shell script, Rake	Unknown	Unknown	Unknown
Lordui	Windows	Proprietary	Yes	Yes	Command-line, everything with a user interface	Email, possibly any way with user interface	No	Java, command-line
NCI	Node.js	MIT	No	No	Command-line	Email, XMPP	No	GitHub, Bitbucket
OpenMake Software Meister	Cross-platform	Proprietary	MSBuild, NAnt, Visual Studio	Ant, Maven 1-2-3	Shell script, batch script, cross-platform command-line, Groovy, Make, RTC Jazz, TFS Build, Custom Script Interpreter	Email, XMPP, RSS, Systray	Eclipse, Visual Studio	Bugzilla, Google Code, Jira, Bitbucket, Redmine, FindBugs, Checkstyle, PMD and Mantis, Trac
Probo CI	Node.js	Apache 2.0	No	No	Node.js	Email, HipChat, Slack (all coming soon)	No	GitHub, Bitbucket, Stash
QuickBuild	Cross-platform	Proprietary	MSBuild, NAnt, Visual Studio	Ant, Maven 1, Maven 2	Rake, shell script, command-line	Email, XMPP, RSS, Google Talk, Remote API	Eclipse, IntelliJ IDEA, Visual Studio	Jira
Semaphore	Hosted	Proprietary	Unknown	Maven 3	Ruby, Java, Python, Node.js, PHP, Clojure, Rake, command-line	Email, Campfire, HipChat, Slack, Flowdock, Webhooks, Remote API	Unknown	GitHub, Bitbucket, Heroku, Code Climate, Cloud 66
Shippable	Hosted	Proprietary	No	Ant, Maven, Gradle	Ruby, Python, Node.js, Java, Scala, PHP, Go	Email, Slack, HipChat, Webhooks, irc, Campfire	No	GitHub, Bitbucket, AWS, Kubernetes, Azure, Heroku
Snap CI	Hosted	Proprietary	No	Ant, Maven, Gradle, Android	C, C++, Clojure, Erlang, Go, Groovy, Haskell, Java, Node.js, Perl, PHP, Python, Ruby, Scala	Email, Campfire, HipChat, Webhook	No	GitHub, Heroku, AWS
Solano CI	Hosted, cross-platform, private cloud	Proprietary	No	Ant, Maven, Gradle, Android	C, C++, Clojure, Go, Java, Javascript, Node.js, PHP, Python, R, Ruby, Scala, command-line	Email, Campfire, HipChat, Flowdock, Slack, Webhook, CCMenu	No	GitHub, GitHub Enterprise, Bitbucket, Heroku, AWS, Azure, Git, Mercurial, Docker
Strider	Node.js	BSD	No	No	C, C++, Clojure, Erlang, Go, Groovy, Haskell, Java, Node.js, Perl, PHP, Python, Ruby, Scala	Email, Slack, Web, Webhook	No	GitHub, Bitbucket, Heroku, GitHub Enterprise, Git

https://en.wikipedia.org/wiki/Comparison_of_continuous_integration_software

Version control systeem vergelijking

13.

Software ↕	Maintainer ↕	Development status ↕	Repository model ↕	Concurrency model ↕	License ↕	Platforms supported ↕	Cost ↕
AccuRev SCM	Micro Focus International	Active	Client-server	Merge or lock	Proprietary	Most Java Platforms (Unix-like, Windows, OS X)	Non-free Quoted on an individual basis. \$350 /seat
GNU Bazaar	Canonical Ltd.	Active	Distributed and Client-server	Merge	GNU GPL	Unix-like, Windows, OS X	Free
BitKeeper	BitMover Inc.	Active	Distributed	Merge	Apache	Unix-like, Windows, OS X	Free
ClearCase	IBM Rational	Active	Client-server	Merge or lock ^[nb 1]	Proprietary	Linux, Windows, AIX, Solaris, HP UX, i5/OS, OS/390, z/OS,	Non-free \$4600 per floating license (held automatically for 30-minutes minimum per user, can be surrendered manually)
Code Co-op	Reliable Software	Active	Distributed	Merge	Proprietary	Windows	Non-free \$150 per seat
Codeville	Ross Cohen	official site offline; latest release July 13, 2007	Distributed	precise codeville merge	BSD	Unix-like, Windows, OS X	Free
CVS	The CVS Team ^[1]	maintained but new features not added	Client-server	Merge	GNU GPL	Unix-like, Windows, OS X	Free
CVSNT	March Hare Software ^[2] and community members	maintained and new features under development	Client-server	Merge or lock	GPL or proprietary	Unix-like, Windows, OS X, i5/OS	Free (with £425 distribution fee) for older version or £85 commercial license for latest version of CVS Suite or Change Management Server
darcs	The Darcs team	Active	Distributed	Merge	GNU GPL	Unix-like, Windows, OS X	Free
Dimensions CM	Serena Software	Active	Client-server	Merge or lock	Proprietary	Windows, Linux, Solaris, AIX, HP UX, z/OS	Non-free
Endevor	CA Technologies ^[3]	Active	Client-server	Merge or Lock	Proprietary	z/OS	Non-free
Fossil	D. Richard Hipp	Active	Distributed	Merge	BSD	POSIX, Windows, OS X, Other	Free
Git	Junio Hamano	Active	Distributed	Merge	GNU GPL	POSIX, Windows, OS X	Free

GNU arch	Andy Tai	unmaintained	Distributed	Merge	GNU GPL	Unix-like, Windows, OS X	Free
IC Manage	IC Manage Inc.	Active	Client–server	Merge or lock	Proprietary	Unix-like, Windows, OS X	Non-free Commercial
MKS Integrity	Integrity, a PTC Company	Active	Client–server	Merge or lock	Proprietary	Unix-like, Windows	Non-free
Mercurial	Matt Mackall	Active	Distributed	Merge	GNU GPL	Unix-like, Windows, OS X	Free
Monotone	Nathaniel Smith, Graydon Hoare	Active	Distributed	Merge	GNU GPL	Unix-like, Windows, OS X	Free
Perforce	Perforce Software Inc.	Active	Client–server	Merge or lock	Proprietary	Unix-like, Windows, OS X	Cost free license, available on application, for OSS or educational use; Also free for up to 20 users, 20 workspaces, and unlimited files; ^[4] Or free for unlimited users and up to 1,000 files; Else \$740–\$900 per seat in perpetuity, or \$144–\$300 per seat per year on a subscription model, both with volume discounts ^[5]
Plastic SCM	Codice Software	Active	Client–server	Merge or lock	Proprietary	Linux, Windows, OS X	Free for up to 15 users; else starting at \$595 per seat, or \$3,500 per 25 developers per year ^[6]
PVCS	Serena Software	Active	Client–server	Lock	Proprietary	Windows, Unix-like	Non-free

Rational Team Concert	IBM Rational	Active	Client–server ^{[nb 2][7][8]}	Merge or lock	Proprietary	Linux, Windows, AIX, Solaris, HP UX, i5/OS, OS/390, z/OS, OS X	Free for up to 10 users; else non-free
Revision Control System	Thien-Thi Nguyen	Active	local	Merge or lock	GNU GPL	Unix-like	Free
SCM Anywhere	Dynamsoft Corporation	Active	Client–server	Merge or Lock	Proprietary	Unix-like, Windows, OS X	Non-free Single user free; \$299 per user, bulk discount available
Source Code Control System	Jörg Schilling ^[nb 3]	Active	local	lock ^[nb 4]	CDDL / proprietary ^[nb 5]	Unix-like, Windows, OS X	While SCCS has traditionally been bundled in commercial UNIX distributions, free CDDL-licensed versions exist
StarTeam	Borland (Micro Focus)	Active	Client–server	Merge or lock	Proprietary	Windows and Cross-platform via Java based client	Non-free Quoted on an individual basis.
Subversion (SVN)	Apache Software Foundation ^[9]	Active	Client–server ^[nb 6]	Merge or lock ^[nb 7]	Apache	Unix-like, Windows, OS X	Free

Vault	SourceGear LLC	Active	Client-server	Merge or lock	Proprietary	Unix-like, Linux, Windows	Non-free \$300 per user
Veracity	SourceGear LLC	web site appears unmaintained; latest release March 25, 2013	Distributed	Merge or lock	Apache	Unix-like, Linux, Windows	Free
Vesta	Kenneth Schaik; Tim Mann, ^{[11][12]}	web site not updated since 2006; latest release February 15, 2009	Distributed NFS-protocol-emulation choice to optionally confederate clients and/or servers	lock on branch; merge branch-to-branch	LGPL	Tru64, Linux	Free
Visual SourceSafe (VSS)	Microsoft	serious bug fixes only	Shared Folder	Merge or lock	Proprietary	Windows	Non-free ~\$500 per license or single license included with each MSDN subscription.
Visual Studio Team Services	Microsoft	Active	Client-server, Distributed	Merge or lock	Proprietary	Windows, cross-platform via Visual Studio Team Services	Free for up to 5 users in the Visual Studio Team Services or for open source projects through codeplex.com; else non-free, licensed through MSDN subscription or direct buy.
Software	Maintainer	Development status	Repository model	Concurrency model	License	Platforms supported	Cost

https://en.wikipedia.org/wiki/Comparison_of_version_control_software