# Scriptie: Project Automatische Testcase Generatie


| | |
|---|---|
| Naam: | Neijts, Jelle J |
| Studentnummer : | 2150906 |
| Afstudeerperiode: | Van 9-2-2015 Tot 2-7-2015 |
| Aantal dagen: | 98 |
| Naam bedrijf: | PinkRoccade Local Government |
| Business unit: | Publiekszaken |
| Plaats: | ‘s Hertogenbosch |
| Naam begeleider: | Jos van der Velde |
| Titel verslag: | Scriptie Project Automatische Testcase Generatie |
| Datum uitgifte: | 4-7-2015 |

# Voorwoord

Voor u ligt de scriptie van het Project Automatische Testcase Generatie. De scriptie is geschreven ten behoeven van de afronding mijn opleiding ICT & Business, aan de Fontys Hogeschool.

Dit project is uitgevoerd in opdracht van PinkRoccade Local Government te 's Hertogenbosch. Het doel van het project is om PinkRoccade te adviseren in de mogelijkheden van automatische testcase generatie.

Met begeleiding van dhr. J.D van der Velde en dhr. L. Koning is dit project succesvol afgerond. Hiervoor wil ik hen hartelijk bedanken.

Tevens wil ik het testteam en de ontwikkelteams 1 en 2 van iBurgerzaken bedanken.

Ik heb erg genoten van mijn stageperiode bij PinkRoccade Local Government. Ik hoop dat dit enthousiasme terug te lezen is in mijn scriptie!

4 juli 2015

Jelle Neijts

# Verklarende woordenlijst

| Begrip | Afkorting | Uitleg |
|---|---|---|
| Agile[i] | | Herbergt verschillende software ontwikkelmethodes. De volgende kenmerken zijn belangrijk voor een agile software ontwikkelmethode:<br>• Opleveren van kleine werkende sub producten (iteraties).<br>• De communicatie is persoonlijk.<br>• Zo min mogelijk documenten. |
| Apache subversion [ii] | SVN | Repository. |
| Automated regression testing | ART | Het automatisch uitvoeren van testcases van de regressietest. |
| Black box testing | | Een software test techniek die de input en output test van een software applicatie en daarbij de werking van binnenuit niet meeneemt. |
| "brown paper"sessie | | Aan de hand van een model wordt feedback gegeven door stakeholders. Deze informatie geeft degene die een Brown paper sessie organiseert relatief snel en accuraat, een beeld van het desbetreffende proces. |
| Capture and replay | CAR | Dekkingsvorm. |
| Code completion | | Door middel van sleutelwoorden, regels codes en fragmenten van een code, worden door de "code completion tool " overeenkomstige codes gezocht. Deze worden ter suggestie aangeboden. Door deze te selecteren bespaard dit tijd en worden potentiële fouten voorkomen. |
| Compiling | | Omzetten van een machinetaal naar een andere programmeertaal. De machinetaal is vaak op een lager niveau zodat machines deze kunnen lezen. Daarom pas je compiling toe. |
| Cucumber [iv] | | Applicatie dat gebruikt wordt voor de uitvoer van testcases. |
| Dekkingsgraad | | Zie paragraaf 5.3 testdekking. |
| Dekkingsvorm | | Zie paragraaf 5.3 testdekking. |
| Deploying | | Opzetten van een applicatie in een omgeving. |
| Domain object model | DOM | HTML- structuur van een staat in de iBurgerzaken applicatie. |
| Domain specific language. | DSL | Een zelf ontwikkelde metataal voor specifiek een domein. Deze DSL is leesbaar voor zowel mens als machine. |
| Dumb Monkey | | Een geautomatiseerde testapplicatie [iii] |
| Executable | | Een bestand wat uitgevoerd kan worden door de computer. |
| Feature (file) | | Een feature is een functie die door middel van Gherkin wordt omschreven. Een feature bestaat uit 1 of meerdere scenario's binnen Gherkin. |
| Gherkin[iv] | | DSL van Cucumber. |
| Grenswaarde | | Een dekkingsvorm waarbij de testdata op of over de grens is van de parameter. Bijvoorbeeld als de business rule voorschrijft dat een datum niet kleiner mag zijn dan 01-01- 2015, dan kun je de waarde groter dan, kleiner dan, gelijk , of een belachelijke waarde , invullen. |
| Java Archive | JAR | Een datacompressie standaard. |
| Jenkins[v] | | Tool dat bouwen en testen van software makkelijker maakt door het automatiseren van kleine taken. |
| Library | | Gedragsstappen die zijn opgeslagen in een bestand die aan te roepen zijn door een applicatie. |
| Maven[vi] | | Tool wat projecten opbouwt. |
| Open source | | Software zonder licentie. |
| Orthogonale array | | Een dekkingsvorm waarbij in een tabel testcases worden gevormd door combinaties van parameters. |
| PictMaster | | Een op Excel gebaseerde applicatie, waarmee orthogonale arrays en "pairwise" toegepast kan worden. |
| Project object model | POM | Hierin wordt de Maven projectinformatie opgenomen. |
| "proof of concept" | POC | Het bewijzen van een theorie of concept door middel van een prototype. |

| | |
|---|---|
| Regressietesten | Het testen van een bestaande applicatie waarbij gecontroleerd wordt of de niet gewijzigde “content” onveranderd is gebleven na het maken van wijzigingen. |
| Repository | Opslag van (Meta) data. |
| Risico | Kans maal bedreiging. |
| Ruby[vii] | Programmeertaal. |
| Scenario | Een scenario geeft de interactie weer van de gebruiker en het systeem. Een scenario is ook een stap binnen Gherkin wat een gedrag van het systeem weergeeft. |
| Scrum | Een software ontwikkelmethode die valt onder agile. Kenmerken van Scrum zijn o.a. :<br>• Sprints dit is de periode waarin iteraties worden gedaan.<br>• Dagelijkse meetings over voortgang. |
| Selenium webdriver | Stuurt en haalt informatie op van een webbrowser. |
| Step definition | Binnen Cucumber wordt een stap gekoppeld aan JAVA of Ruby code. Deze code is de step definition. |
| Systeem test | Bij deze test wordt de werking getest van de nieuwe componenten(clusters) in de totale omgeving. Hierbij worden alleen de nieuwe functionaliteiten meegenomen. |
| Test suite | Een verzameling van test cases. |
| Testcase/testgeval/scenario | Een test waarmee de werking van een software applicatie of deel daarvan, wordt getoetst. |
| Testdekking | Het totale verwachte gedrag wat getest wordt met specifieke dekkingsvorm(en). Dit is geen absoluut getal, maar een benadering. *Dit komt doordat het totale verwachte gedrag niet meetbaar is.* |
| Unit test | Is een test van een klein deel van de code. En test de werking losstaand. |
| User story | Beschrijft in een paar zinnen het te implementeren gedrag (bij behavior driven development). |

# Inhoudsopgave

# Managementsamenvatting

PinkRoccade Local Government's (PRLG) testteam had behoefte aan het behalen van een hogere dekkingsgraad en het verlagen van tijd, die nodig is voor het testen en / of creëren van testgevallen. Het Project Automatische Testcase Generatie (PATG) stond in het teken van deze behoefte.

PATG heeft zich gericht op het zoeken, creëren en beoordelen van methodieken, technieken en tools, waarmee black box testen mogelijk is.

PATG gebruikt hierbij een variëteit aan methodieken als: "deskresearch", interviews, Brown paper sessies en "proof of concepts".

Naast project lid dhr. J. Neijts bestond PATG uit dhr. J.D van der Velde, dhr. L. Koning en het testteam dat betrokken is bij iBurgerzaken.

PATG presenteert vijf varianten die een bijdragen leveren aan de automatisering van testcases. Na de beoordeling van deze varianten is een advies uitgebracht.

De aanbevelingen zijn als volgt[viii]:

- Voer de "code completion library" CucumberPeople per direct in.
- Voer de PictMaster "pairwise" tool in.
- Voer "capture and replay" niet in, dit omdat deze methode geen verbeteringen oplevert.
- Test de Dumb Monkey door middel van "proof of concept".
- Bij invoering van PictMaster en de Dumb Monkey is het belangrijk dat er een "productowner" komt en dat er documentatie bestaat. De "productowner" moet komen uit het testteam.
- Test Sogeti' s Cover om te beoordelen of deze de Dumb Monkey kan verbeteren

Door het opvolgen van deze aanbevelingen zal tijd bespaard worden bij het testproces en zal de dekking van de tests verbeteren. Hierdoor neemt het risico van fouten na "deployment" af en neemt de kwaliteit van iBurgerzaken toe.

# Management summary

PinkRoccade Local Government 's test team had the need to test with a higher test coverage and to reduce the test case creation time.

Project automatic test case generation (PATCG ) as a graduate internship is created to provide in those needs.

PATCG is mainly focused on searching, creating and assessing techniques, methods and tools which made black box testing possible.

PATCG used a variety of methods: desk researching, interviewing, brown paper sessions and proof of concepts.

Members of PATCG's project were sir J Neijts, sir J.D van der Velde, sir L. Koning and the test team of iBurgerzaken.

PATCG presents six different improvements after using varieties of methods, techniques and professional opinions.

The recommendations are [x]:

- Add a "code completion library" CucumberPeople to the IDE as soon as possible.
- Introduce the PictMaster pairwise test method.
- Do not introduce researching capture and replay, because of the small compatibility with the current test set up and because of the small or none improvements.
- Improve the Dumb Monkey, the more intelligent the Monkey becomes the more improvements it will bring.
- Set up proper documentation for the recommended improvements and make one of the members of the test team, the product owner.
- Test Sogeti's Cover to asses possible improvements to the Dumb Monkey.

By following these recommendations the testing time will be reduced a test coverage will rise. Thereby the risk of faults after "deployment" will be reduced and the quality of iBurgerzaken quality will almost certainly approve.

# 1. Inleiding

Het testteam van PinkRoccade Local Government (PRLG) Publiekszaken, heeft de professionele opinie dat sommige fouten niet gedetecteerd door het testproces komen[ix]. Dit heeft te maken met de grote hoeveelheid tests die uitgevoerd dienen te worden. Door deze grote hoeveelheid is het, volgens het testteam van iBurgerzaken, onmogelijk om alle "content" mee te nemen in het testproces.

Hierop volgt de vraag: kan de huidige testdekking of test tijd binnen PinkRoccade Local Government's afdeling Publiekszaken, verbeterd worden?

Om het antwoord te kunnen geven op deze vraag zijn de volgende hoofd- en deelvragen opgesteld:

Hoofdvraag:

*"In hoeverre zijn er mogelijkheden om de testdekking van het regressie en systeemtestproces te vergroten door een vorm van testcase automatisering?"*

Deelvragen:

- Welke methodes en tools (intern en extern [1]) worden bij PRLG ingezet om een test scenario te genereren en uit te voeren in de huidige situatie?
- Welke tools ( intern en extern) kunnen bijdragen aan de automatisering van de generatie van test scenario's?
- Welke methodes zijn beschreven in de literatuur en welke kunnen worden geraadpleegd om test scenario's deels of volledig automatisch te genereren?
- Wat zijn de mogelijke voor- en / of nadelen van gedeeltelijke of volledige geautomatiseerde generatie van test scenario's ten opzichte van de huidige situatie?

Deze scriptie is bedoeld om de processen, activiteiten en beslissingen te beschrijven welke ten grondslag hebben gelegen aan het Project Automatische Testcase Generatie PATG. Het doel van dit project is inzichtelijk te krijgen op welke manieren testcases sneller vervaardigd kunnen worden en tevens voor een hogere testdekking kunnen zorgen.

De stagiair is Jelle Neijts, hij studeert af in de richting van ICT & Business aan de Fontys Hogeschool te Eindhoven.

Dit document geeft in het eerste hoofdstuk weer in welk bedrijf de stagiair heeft stage gelopen en welke rol het project speelt binnen de organisatie. Het hoofdstuk "Huidige situatie", beschrijft

[1] Interne tool: door PinkRoccade reeds in gebruik. Externe tool: tool afkomstig uit de buitenwereld.

**welke methodieken, technieken momenteel worden gebruikt bij het testen van de iBurgerzaken applicatie.**

**De hoofdstukken " Stageonderzoek, Projectuitvoer en Conclusies", beschrijven de belangrijkste producten en activiteiten van het project.**

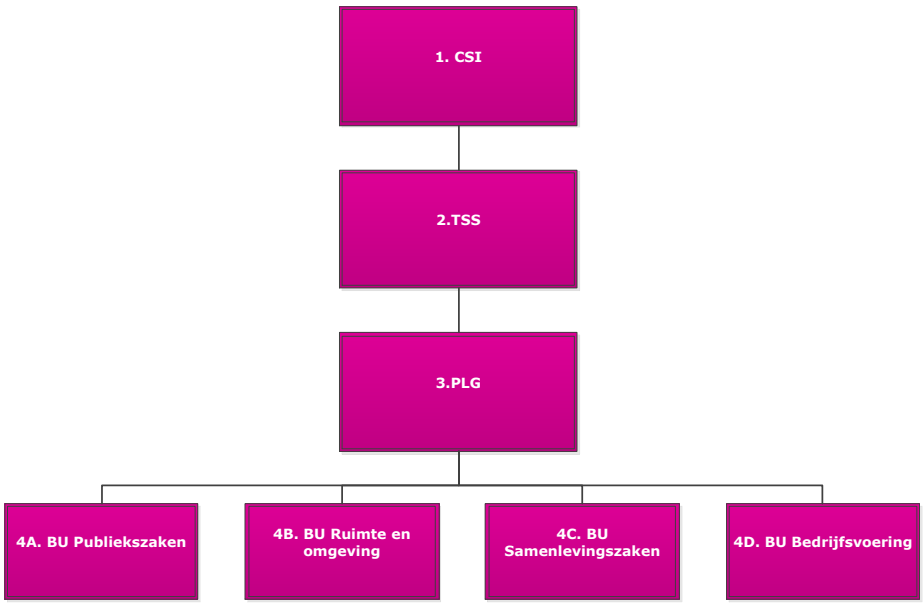
**Tenslotte zal in het hoofdstuk "Persoonlijke evaluatie", de stagiair een terugkoppeling geven van hetgeen hij heeft geleerd in relatie tot de vooraf gestelde leerdoelen.**

# 2. Beschrijving PinkRoccade Local Government

Dit hoofdstuk beschrijft PinkRoccade Local Government en de business unit Publiekszaken binnen PinkRoccade Local Government en de bedrijven waarvan zij deel uitmaakt. Deze beschrijving geeft de belangrijkste bedrijfsactiviteiten, de cultuur en de plaats en functie van de stagiair weer.

## 2.1 PinkRoccade Local Government

Figuur 1 geeft weer hoe de business unit Publiekszaken is gepositioneerd binnen PinkRoccade Local Government en bovenliggende partijen.




FIGUUR 1 ORGANOGRAM PRLG[x]

Het figuur 1 is als volgt opgebouwd:

1. Constellation Software Inc. Dit is de overkoepelende organisatie waaronder PinkRoccade Local Government valt.
2. Total Specific Solutions (TSS). Dit is het moeder bedrijf van PinkRoccade Local Government.
3. PinkRoccade Local Government (PRLG). Dit is het bedrijf waaronder de afstudeerstage valt.

4A. Business unit (BU). Dit is de business unit die zich voornamelijk richt op de iBurgerzaken applicatie. Hierbinnen vindt het PATG plaats.

4B t/m 4D. Andere business units binnen PinkRoccade Local Government, worden niet verder beschreven.

## 2.2 Business unit publiekszaken

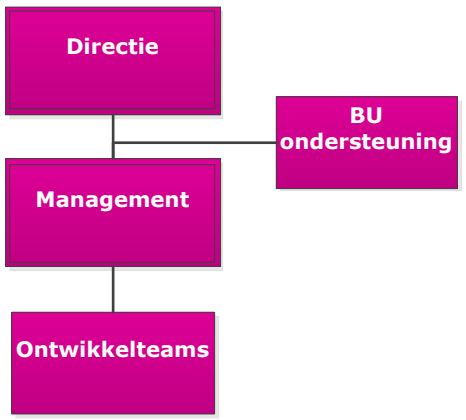
Figuur 2 geeft beknopt de onderdelen van de business unit Publiekszaken weer:

**Directie:** De directie bestaat uit de directeur van de business unit.

**BU ondersteuning:** Deze ondersteunt de directie in haar taken.

**Management:** Dit management bestaat uit product managers, een development manager, een sales manager en een HR manager.

**Ontwikkelteams:** Er zijn vijf ontwikkelteams voor de iBurgerzaken applicatie. Tevens zijn er teams die bezig houden met Cloud, Cipers en Zaak.




FIGUUR 2 ORGANOGRAM BU PUBLIEKSZAKEN

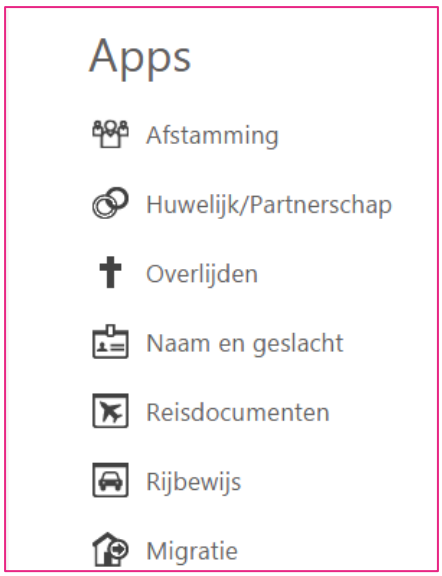
## 2.3 Bedrijfsactiviteiten

Momenteel vindt het programma modernisering gemeentelijke basisadministratie (MGBA)[xi] plaats bij de lokale overheden. Dit houdt in dat lokale overheden een voorbereiding maken om hun gegevens te koppelen aan de landelijke database (Basis Registratie Personen BRP). De business unit publiekszaken biedt bij deze verandering een oplossing in de vorm van iBurgerzaken. Dit is een softwarepakket dat bestaat uit veertien standaard applicaties en enkele specifieke applicaties. Ambtenaren en burgers kunnen met deze web applicatie een geboorte melden, een trouwerij regelen of een rijbewijs laten maken enzovoort.

De applicatie wordt, zoals is vermeld in figuur 1, door vijf ontwikkelteams gebouwd. Hierbij wordt de agile ontwikkelmethode Scrum gebruikt. Dit houdt in dat ze door middel van sprints incrementele (oplopende) werkende delen opleveren. Deze stukken worden in diverse stadia getest. Dit zijn de ontwikkel-systeem[xii]-en acceptatietests. Binnen deze tests worden diverse testvormen gebruikt zoals regressietesten.

## 2.4 iBurgerzaken

iBurgerzaken is de applicatie waar het PATG betrekking op heeft. Zoals in figuur 3 is geïllustreerd bestaat iBurgerzaken uit diverse applicaties. De applicatie heeft een versie voor de ambtenaar en een versie voor de burger. In figuur 3 is een screenshot genomen uit de versie van de ambtenaar.




FIGUUR 3 IBURGERZAKEN APPLICATIES[xiii]

## 2.5 Cultuur

De cultuur binnen de business unit Publiekszaken is aan de hand van Daniel Denison's model[xiv] beschreven. Het model beschrijft de cultuur in vier algemene categorieën en drie subcategorieën. De vier algemene categorieën zijn: Missie, Consistentie, Betrokkenheid en Aanpassingsvermogen. Deze vier categorieën vormen samen de cultuur van de organisatie. In de volgende paragrafen worden de categorieën aan de hand van de opinie / bevindingen van de stagiair uitgewerkt.

### *Missie*

De missie van PinkRoccade Local Government geeft richting aan dat wat zij willen bereiken in de toekomst en hoe zij dit willen realiseren. Logischerwijs geeft dit kopje missie invulling aan de missie, de visie en strategie.

De missie en visie zijn opgenomen in het Merkkompas van PinkRoccade Local Government. Deze zijn weergegeven in figuur 4.


FIGUUR 4 MISSIE EN VISIE PRLG[xv]

**Strategie en doelstellingen: PinkRoccade Local Government richt zich op het hervormen van processen zodat de burger meer taken over kan nemen van de ambtenaar.**

### *Aanpassingsvermogen* [xv]

**PinkRoccade Local Government speelt op hoog niveau, en zal om moeten gaan met veranderingen. Deze veranderingen komen zowel uit de organisatie als buiten de organisatie. De veranderingen zijn op te delen in veranderen, in klantfocus en in organisatie. Hoe om wordt gegaan met deze veranderingen bepaalt het aanpassingsvermogen.**

**Veranderen: Met iBurgerzaken heeft PinkRoccade de procesveranderingen in gang gezet met als doel de ambtelijke processen te verbeteren.**

**Klant Focus: Met de ontwikkeling van iBurgerzaken worden de procesveranderingen voor de ambtenaar en burger gerealiseerd, PinkRoccade speelt hiermee in op veranderingen in wetgeving en behoeften van overheden.**

**Organisatie: Door het gebruiken van Scrum wordt kennis dagelijks gedeeld. Problemen worden samen opgelost. Hiervoor is ruimte voor ideeën en inzichten van iedereen welkom.**

### *Betrokkenheid*

**De betrokkenheid van de werknemers bij de projecten is erg belangrijk. Betrokkenheid kan getoond worden in drie vormen: zelfstandigheid van de werknemer, de kijk van het team en coaching/training.**

**Zelfstandigheid: De werknemers binnen de BU Publiekszaken hebben de vrijheid om hun eigen keuzes te maken binnen hun werk. Het resultaat is het belangrijkste.**

De kijk van het team: Teamwork is erg belangrijk. Scrum maakt het mogelijk om gecoördineerd samen aan taken te werken. Taakverdeling wordt overlegd en men helpt elkaar bij lastigere taken.

Coaching/training: Kennis en kunde is essentieel om te ontwikkelen. Om deze op peil te houden is er voldoende budget om vaardigheden bij te spijkeren.

*Consistentie*

PinkRoccade Local Government zorgt voor consistentie binnen de organisatie door hun kernkwaliteiten uit te dragen. Zij zorgt voor overeenstemming en creëert een basis waarin coördinatie en integratie als belangrijk wordt geacht.

*Kernkwaliteiten:* Deze worden uitgedragen door het management. Tot deze kernkwaliteiten behoren: de klant centraal stellen, veranderingsgezindheid, kennis en ervaring binnen de publieke sector en investeringskracht.

*Overeenstemming:* PinkRoccade Local Government zorgt middels periodiek informele overlegmomenten, Scrum meetings en persoonlijk contact, dat veranderingen besproken worden.

*Coördinatie en integratie:* De werknemers van PinkRoccade Local Government hebben een overzicht van hun eigen taken en verantwoordelijkheden en hebben een overzicht van taken en verantwoordelijkheden van anderen. Dit wordt dagelijks besproken binnen de “daily” Scrum meetings.

## 2.6 Plaats en functie

De stagiair voert het Project Automatische Testcase Generatie, uit onder de business unit Publiekszaken.

# 3. Huidige situatie

Dit hoofdstuk beschrijft hoe de producten, gebaseerd op achtergrond, probleemstelling, en de huidige testmethodieken, tot stand zijn gekomen.

## 3.1 Achtergrond en probleemstelling

Het testteam van de afdeling publiekzaken heeft behoefte aan een hogere mate van testverbreding naast de bestaande uitvoer van de regressie en systeemtests [xvi]. Deze vraag komt voort uit de constatering dat in de productomgeving incidenten zijn geweest die met testverbreding voorkomen hadden kunnen worden [xvi]. Een incident was bijvoorbeeld dat er "bugs" gerapporteerd werden die volgens dhr. L. Koning[2] wel door de ART [xvii] gevonden hadden moeten worden. De oorzaak was, volgens hem, dat niet alle testen meegenomen konden worden door gebrek aan tijd.

Hieruit is de behoefte ontstaan om test scenario's automatisch te genereren. De vraag bestaat of dit een mogelijkheid is, en als dit zo is, op welke manieren dit zou kunnen[xviii].

### "Exploratory" testen

De wens om (semi) automatische "exploratory" testing[xix] te kunnen toepassen is bij de start van dit project uitgesproken. Bij "exploratory" testen wordt niet volgens een herhalend patroon getest, de tester kan deze tests niet automatiseren. Feitelijk worden hier de situaties getest die niet gewoon zijn. Deze vorm kost zeer veel tijd om uit te voeren door de tester. Een automatiseringsslag bij deze test zou volgens dhr. L. Koning een verbetering zijn.

### Belangrijkste bedrijfsactiviteiten

Door middel van interviews met o.a. het testteam en ontwikkelaars van iBurgerzaken, zijn belangrijkste bedrijfsactiviteiten[xx] duidelijk geworden. De meest relevante bedrijfsactiviteit van PRLG is het testproces van iBurgerzaken. Naast interviews werd ook "meegelopen" door stagiair. Hierdoor werden de dagelijkse activiteiten van het testproces duidelijk.

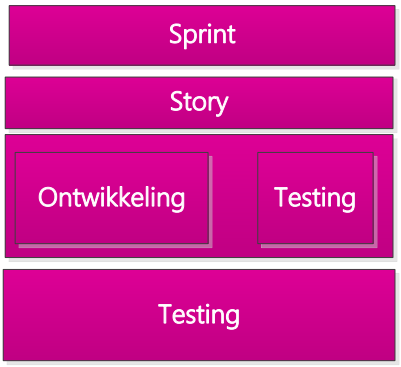
### Ontwikkeling iBurgerzaken

Het algemene ontwikkelproces van de iBurgerzaken applicatie werd duidelijk door het bijwonen van een aantal "daily" Scrums van ontwikkelteam 1. Om deze bij te kunnen wonen is vooraf door de stagiair het Agile manifest bestudeerd. Daarbij werden, modellen gemaakt door de stagiair van de testprocessen om inzicht te krijgen.

[2] Tester iBurgerzaken en tweede stagebegeleider

### Testproces

Het testproces[xxi] werd duidelijk door eerdergenoemde interviews met de heren L. Koning en. J. Ueberbach en A Sandu en met mevr. C. Bunea en mevr A. Pascariu. Zij zijn de leden van het iBurgerzaken testteam. Het testproces is met de hulp van modellen en aan de hand van Brown paper sessies gedocumenteerd. In figuur 5 wordt het globale testproces binnen iBurgerzaken geïllustreerd.




FIGUUR 5 TESTPROCES M.B.V. SCRUM [xxii]

Het testteam werd op de hoogte gebracht van de ontwikkelingen van het project door de stagiair. Tijdens wekelijkse Skype meetings kreeg de stagiair een beter beeld van de wekelijkse werkzaamheden van het iBurgerzaken testteam.

## 3.2 Huidige testmethodieken

In deze paragraaf van het adviesdocument wordt specifiek ingegaan op de testcase creatie en automatisch testen. Hier is voor gekozen omdat deze onderwerpen in detail bekend moeten zijn bij de stagiair. Deze kennis dient dan ook als uitgangspunt voor de gewenste situatie.

### Testcase creatie

De opbouw van een testcase bij systeem en regressietesten, dient geanalyseerd te worden om te beoordelen welke elementen hierin verbeterd of geautomatiseerd konden worden.

Aan de hand van “deskresearch” waarbij onder andere de testmethodiek Tmap Next van Sogeti is gebruikt, is de theorie van het maken van testcases bestudeerd.

De omschrijving van de testcase creatie binnen Publiekszaken is gedaan aan de hand van interviews met L. Koning. De omschrijving diende als input voor een “proof of concept” van de huidige testsituatie[xxiii] (basisopstelling).

### Automatisch testen

De automatische regressietest (ART) is opgenomen in de basisopstelling om de huidige situatie in detail weer te geven. De ART is een omgeving waarin automatische black box tests worden uitgevoerd, met als doel regressie en systeemtesten uit te voeren.

Bij de vervaardiging van de basisopstelling is o.a. een Brown paper sessie gebruikt. De Brown paper sessie is gehouden samen met J.D van der Velde[3] en L. Koning. Naast Brown paper sessies werd de ART in detail duidelijk aan de hand van interviews met L. Koning en J.D van der Velde. De modellen die resulteerde uit de Brown paper sessies[xxiv] zijn aan de hand van feedback van L. Koning en J.D van der Velde geperfectioneerd en aangepast m.b.v. modeleersoftware.

### "business case"

De business case[xxv] is opgezet om de opdrachtgever onderbouwd te kunnen tonen welke voordelen het project met zich meebrengt. Samenvattend geeft deze weer dat een hogere dekkingsgraad meer risico's afdekt. Bovendien wordt de vergelijking gemaakt dat meer testcases resulteren in een hogere testdekking. Hoe meer testdekking je bereikt hoe meer risico's je afdicht voor de iBurgerzaken applicatie. Dit komt uiteindelijk ten goede aan de kwaliteit van de applicatie.

[3] Eerste stagebegeleider.

# 4. Stageonderzoek

In dit hoofdstuk worden de hoofd- en deelvragen, doelstellingen, belangrijke resultaten, aanpak en fasering, scope, planning en gebruikte methoden, uiteengezet.

## 4.1 Doelstellingen

Zoals in de inleiding is beschreven heeft het testteam van iBurgerzaken de behoefte om de testdekking te verhogen binnen haar testproces. Zij willen dit graag voor elkaar krijgen door een automatiseringsslag te maken op het genereren van testcases. De doelstellingen van PATG worden als volgt benoemd [xxvi]:

- Voor juli 2015, middels een literatuuronderzoek aantonen welke technieken kunnen helpen bij de automatisering van testcases binnen PinkRoccade Local Government.
- Voor juli 2015, via een "proof of concept" aantonen of een black box techniek of tool, de testdekking van de regressie en systeemtests binnen de Business Unit, (BU) Publiekszaken van PinkRoccade Local Government kan verhogen.
- Voor juli 2015, een advies geven of een black box techniek en / of tool, het meest geschikt is voor de verhoging van de testdekking van regressie en systeemtests binnen PinkRoccade Local Government.

## 4.2 Belangrijkste resultaten

De belangrijkste resultaten van PATG zijn:

### "Project initiation document" (PID)

Dit is het projectplan voor het Project Automatische Testcase Generatie. In bijlage A is het PID bijgevoegd.

### Adviesdocument (AD)

Het adviesdocument geeft de opdrachtgever van PATG antwoord op de hoofd- en deelvragen en voorziet hem van concrete aanbevelingen[xxvii]. Dit document is het belangrijkste product van de stage en is in Bijlage B bijgevoegd.

### "proof of concepts"

Deze techniek is toegepast op de basisopstelling en op de vijf onderzochte varianten[xxviii].

## 4.3 Aanpak en fasering en mijlpalen

Het Project Automatische Testcase Generatie is opgebouwd in drie fases: oriëntatiefase, onderzoeksfase en afrondingsfase. Iedere fase bevat een of meerdere mijlpalen. Deze mijlpalen gaven de voortgang weer van het project. In hoofdstuk 5 Conclusies, aanbevelingen en opdrachtevaluatie, wordt een reflectie op de mijlpalen beschreven.

*fasering*[xxix]

Door de fasering is het project beheersbaar gebleven en hield de stagiair de grote lijnen van het project, zie bijlage A.

| Fase | Mijlpaal |
|---|---|
| Oriëntatiefase:<br>In deze fase wordt de scope van het project, de achtergrond en de voorwaarden gedefinieerd.<br><br>Definitiefase:<br>Hier worden de eisen ten behoeven van de gewenste situatie benoemd. | Mijlpaal 1: Het huidige testproces is gedocumenteerd: Deze mijlpaal is cruciaal omdat dit input geeft voor zowel het project initation document (PID) maar ook het Adviesdocument (AD), de scriptie maar ook basis is voor de basisopstelling. Als het huidige proces niet volledig in kaart is gebracht, bestaat het gevaar dat er onvoldoende kennisoverdracht is geweest, zo kan het project buiten scope raken en haar doelstellingen niet haalt. |
| Onderzoeksfase: | Mijlpaal 2: Het POC basis- test-opstelling is gereed: De basisopstelling was het "proof of concept" van het huidige testproces. Hiermee verkreeg de stagiair inzicht in de interne tools en methodieken.<br><br>Mijlpaal 3: De keuze in oplosrichting is gemaakt.<br>Hier werd de keuze gemaakt van de oplosrichting. Belangrijke methodes als "model based"en "capture and replay" legde de basis van de oplosrichting. Bekende dekkingsvormen werden ook meegenomen als oplosrichting. Aan de hand van deze richtingen zijn vijf varianten geformuleerd[4]. |

[4] Paragraaf 5.4

| | |
|---|---|
| | Mijlpaal 4: De geschikte methodes en tools zijn geselecteerd. Het samenstellen van een long en short list is een essentiële stap en vormt de basis van de aanbevelingen en conclusies. |
| Afrondingsfase:<br>Alle resultaten zijn binnen, deze zijn verwerkt in conclusies en adviezen. | Mijlpaal 5: De conclusie is getrokken. Aan de hand van de resultaten uit het "proof of concept", de interviews en de mening van het testteam zijn conclusies getrokken met betrekking tot de juiste tools. Hierop zijn de aanbevelingen gebaseerd. |

## 4.4 Scope

In de meetings met L. Koning en J.D van der Velde is gekozen voor een Black box test variant(en) voor de toekomstige situatie[xxx].

In het adviesdocument worden uitgebreid de voor en nadelen van black en White box testen beschreven [xxxi]. Bovendien worden hier belangrijke argumenten genoemd waarom juist gekozen is voor black box testtechnieken.

## 4.5 Planning

Deze is in het PID opgenomen als masterplanning[xxxii], de grove lijnen van het project zijn hier uitgezet. Wekelijks werden met J.D van der Velde de lijnen uitgezet voor de komende week. Ook werd besproken waar de aandacht kwam te liggen en waar eventuele problemen lagen. Voor de zeer korte termijnplanning werden dagelijks "todo" lijstjes gemaakt in de plannings-tool Trello[xxxiii]. Deze lijstjes waren altijd zichtbaar voor J.D van der Velde in de communicatietool Slack. Zo kon hij het dagelijks proces volgen.

## 4.6 Gebruikte methoden en technieken

De volgende methoden en technieken waren belangrijk bij het voltooien van PATG:

### Programmeren

De stagiair is veel bezig geweest met het programmeren in Java om de Dumb Monkey applicatie te ontwikkelen[xxxiv]. De Dumb Monkey applicatie is een door PATG ontwikkelde applicatie en de vierde variant in het adviesdocument[xxxv]. Bij het opdoen van de nodige programmeerkennis kreeg de stagiair ondersteuning van J.D van der Velde, Java experts binnen PinkRoccade en het internet. Buiten Java zijn ook Cucumber, Jsoup[xxxvi], Selenium webdriver[xxxvii] en Javascript gebruikt. Jsoup en Selenium zijn beide bibliotheken. Jsoup wordt gebruikt om HTML te op te halen uit webpagina's

en op te slaan in elementen zoals input, button, koppen etc. Selenium werd gebruikt als web driver.

### Analyseren

Bij de huidige en toekomstige situaties zijn modellen gemaakt. Deze modellen komen deels uit analyses waarbij een Brown paper sessie is toegepast. Aan het ontwikkelen van de Dumb Monkey is veel analyse vooraf gegaan.

### Interviewen

De interviews[xxxviii] met L. van der Aalst en B Visser werden vooraf goed voorbereid middels een inventarisatie te maken van vragen. Beide gesprekken werden opgenomen en later uitgewerkt. De samenvattingen zijn meegenomen in het adviesdocument.

### Brown paper sessies

De Brown paper sessies waren georganiseerd met L. Koning en J.D van der Velde. Tijdens deze meetings werden modellen ontwikkeld van de bestaande en huidige situatie. In vervolgsessies werden deze de modellen middels feedback verder verrijkt.

### Testmethode

Als leidraad bij de stage is Tmap Next[xxxix] aangehouden als methode voor o.a. de terminologie. Bovendien is de basale kennis van softwaretesten binnen Scrum gebruikt om de huidige en gewenste situatie te analyseren.

### “deskresearch”

Door middel van het gebruik van een zoekmachine werden wetenschappelijke papers, relevante websites, professionele opinies gevonden. Hieruit ontstond een beeld van de richtingen die bij konden dragen aan een verbetering. Door middel van “long en short lists” zijn vergelijkbare tools, technieken en methodes vergeleken en geselecteerd.

# 5. Projectuitvoering

Dit hoofdstuk beschrijft de totstandkoming van de in het adviesdocument benoemde vijf varianten [xxviii]. De vijf varianten zijn tot stand gekomen door globaal onderzoek naar het automatiseren van testcases, maar ook door het onderzoek naar de bestaande dekkingsvormen en testmethodieken[5]. De varianten zijn besproken met het testteam van iBurgerzaken. De resultaten van het onderzoek naar de varianten zijn getoetst aan gestelde criteria.

## 5.1 Opzet

Het uitgangspunt van de oplossingsrichtingen waren de achtergrond en probleemstelling[xl].

Door het afnemen van interviews, Brown paper sessies en "deskresearch" oplossingsrichtingen is inzichtelijk geworden welke methodieken een bijdrage kunnen leveren aan de automatisering van testcase generatie.

Eerst is onderzocht of er tools of oplossingen binnenshuis[6] voor handen waren. Dit bleek niet het geval. Daarna heeft het onderzoek zich op externe methodieken en tools gericht.

Wanneer er voor de oplossingsrichting geen complete tools bestonden, werd geanalyseerd welke deelcomponenten nodig zijn om te komen tot (een deel van) de beoogde oplossingsrichting. Hierbij werd in overweging genomen of er derde partij tools gebruikt werden of de tool binnenshuis ontwikkeld zou worden.

Uit dit onderzoek volgde vijf varianten die nader onderzocht werden.

De onderzoeken zijn uitgevoerd door middel van "proof of concepts" en door middel van conclusies die zijn gedaan op basis van reviews over de variant.

In het geval dat meerdere oplosrichtingen voor dezelfde variant gevonden waren, is een opensource versie gekozen bij vergelijkbare werking.

Aan de hand van de resultaten zijn per variant conclusies getrokken op basis van de criteria[xli].

## 5.2 Onderzoek

Iedere variant kent (soms meerdere) specifieke opzet(ten) waarin de tool(s) binnen de variant worden vergeleken en getest. Zo zijn sommige varianten beoordeeld middels een "proof of concepts", andere zijn beoordeeld door middel van expert opinies of interviews. Vaak was dit een combinatie.

[5] Dit vormde de theoretische kennis voor PATG

[6] Reeds gebruikte tools door business units binnen PRLG

### Beoordeling varianten

De tools binnen de varianten zijn aan de hand van gedefinieerde kwaliteitscriteria beoordeeld. Hieronder volgen deze criteria:

### Criteria

- Tijdsbesparing: gekozen om tijd vrij te maken om meer te kunnen testen
- Dekking: meer testdekking resulteert in lagere kans op falen in productie.
- "feasibility": Hoe goed past de tool in het huidige test "framework".
- Gebruiksvriendelijkheid: Hoe eenvoudig is het gebruik van de oplossing.

Toelichting: Deze criteria zijn in samenwerking met het testteam en de stagiair samengesteld. Ze zijn ontworpen om een onderscheid te kunnen maken op basis van kwalitatieve eigenschappen. De criteria zijn niet altijd meetbaar te maken. Dit omdat een testcase altijd verschilt, hierdoor kan geen ontwikkeltijd worden gehangen[7] aan een testcase, omdat niet iedere testcase dezelfde dekkingsvorm heeft. Dit maakt dat geen kwantitatief oordeel kan worden gegeven over de dekkingsgraad. "feasibility", de mate waarin de variant (tool) past binnen de huidige situatie en gebruiksvriendelijkheid is eveneens kwalitatief.

Naast de individuele beoordeling per tool per variant werden de varianten ook ten opzichte van elkaar beoordeeld. Eveneens is onderzocht of de varianten elkaar uit konden sluiten in de gewenste situatie of elkaar konden complementeren.

De beoordeling werd door het testteam, de externe expert en de stagiair gedaan. Zij hebben aangegeven welke variant of varianten het beste antwoord geeft op de onderzoeksvraag.

## 5.3 Theoretische kennis

Een basis van theoretische kennis over testen was noodzakelijk om juist "deskresearch" te verrichten en de huidige situatie te begrijpen. Belangrijke theorie is in deze paragraaf opgenomen en uitgelegd.

### Testsoort en testbasis

Om een algemeen beeld te krijgen waar, in het software ontwikkelproces het testen plaats vindt en welke input ten grondslag hieraan ligt, is gekozen om het V model van Tmap Next aan te houden. Aan de hand van dit model wordt de scope van het project ook een stuk duidelijker.

[7] Met uitzondering van variant 1.

### Dekkingsvormen

De dekkingsvormen zijn beschreven in het adviesdocument omdat dit de bouwstenen zijn voor de varianten in de oplossingsrichting. In het adviesdocument onder de paragraaf dekkingsvormen zijn de belangrijkste dekkingsvormen voor het project uitgelegd. De overige dekkingsvormen zijn meegenomen in de bijlage[xlii]. Er zijn verschillende bronnen gebruikt voor deze paragraaf zoals: survey, websites, scripties (de bronnen zijn vermeld in het adviesdocument).

### Testdekking

Dit is een belangrijk begrip omdat de probleemstelling en de "business case" hieraan is opgehangen. Het is dus belangrijk dat de opdrachtgever en de andere "stakeholders" weten wat dit begrip inhoudt. Tmap Next heeft hier als basis gediend.

### Testgevallen en testscripts

Deze begrippen zijn fundamenteel om te begrijpen waar het project over gaat. Wederom is gekozen voor de Tmap Next methode.

### Zoekalgoritmes

Hierin worden twee belangrijke algoritmes uitgelegd namelijk die van "breadth first" en "depth first"[xliii]. Deze algoritmes zijn bedoeld om te implementeren in de Monkey tests. Deze algoritmes zijn specifiek onderzocht om "exploratory"testen met de Dumb Monkey effectiever te maken. Door afhankelijkheden binnen iBurgerzaken en de beperkte tijd om de Dumb Monkey nog verder te kunnen ontwikkelen zijn deze algoritmes niet toegepast.

## 5.4 Vijf varianten

In het adviesdocument zijn vijf varianten [xxiii] benoemd en beoordeeld. Deze varianten vormen individueel of in combinaties een verbetering op de huidige situatie.

Deze varianten zijn:

1. Semiautomatische testcase generatie door *"code completion library" (CCL).*
2. Semi en automatische testcase generatie door *"capture and replay"*. (CAR)
3. "pairwise" testgeval generatie.
4. Dumb Monkey testen.
5. Brilliant Monkey testen.

### *Variant 1 Semi automatische testcase generatie door CCL*

Semi automatische testcase generatie door CCL[xliv] heeft als doel de testcase creatie te versnellen, door een library toe te voegen aan de testcase ontwikkelomgeving. Deze library heeft kennis van bestaande testgevallen en geeft, op basis van sleutelwoorden, suggesties van bestaande testgevallen, zodat deze semi automatisch aangevuld kunnen worden. Hierdoor wordt tevens het hergebruik van testgevallen vergroot en wordt de kans van herhaling van bestaande testgevallen, en delen daarvan kleiner.

#### Opzet

Deze variant is geselecteerd d.m.v. "de long list short list methode" en getoetst door middel van een "proof of concept". De geselecteerde tool, CucumberPeople, werd lokaal in de ontwikkelomgeving van de stagiair getest op zijn werking. Hiervoor heeft de stagiair Cucumber gebruikt en Java om testgevallen te maken. Later is deze tool geëvalueerd door L. Koning en gepresenteerd aan de rest van het testteam.

#### Resultaten en discussie

De tool werkt prima. De tool scoort goed op gebruiksvriendelijkheid en tijdsbesparing. Testdekking voegt deze niet toe, maar door de controlerende werking wordt testcase creatie accurater.

### *Variant 2 Semi en automatische testcase generatie door "capture and replay"*

Deze variant[xlv] heeft als kenmerk dat de gebruiker zelf door de applicatie heen klikt en aanpassingen doet en dat deze sessies worden opgenomen als testgevallen. Deze testgevallen kunnen later worden afgespeeld. Afwijkingen in de applicatie worden gevonden als het testgeval niet meer volledig afgespeeld kan worden.

#### Opzet

Deze variant is onderzocht door zoeken naar bestaande tools. Aan de hand van het gebruiken van diverse van deze tools is Selenium IDE gekozen om uitgebreider te testen. Deze tool is door middel van een "proof of concept" onderzocht en beproefd. Dit hield in dat de stagiair lokaal de iBurgerzaken applicatie gebruikte om testgevallen op los te laten. De resultaten van deze test zijn aan J.D van der Velde voorgelegd, hij kon deze resultaten reproduceren.

#### Resultaten en discussie

Selenium IDE werkt niet voldoende doordat testcases niet correct afspelen, bovendien heeft de variant beperkingen bij hergebruik van testcases. De testdekking is hierdoor geen toevoeging. De gebruiksvriendelijkheid is daarentegen wel goed.

### *Variant 3 "pairwise" test data generatie*

"Pairwise"[xlvi] is gebaseerd op de theorie dat fouten (bugs) gevonden worden door combinaties van parameters. Doordat er combinaties worden gemaakt van parameters worden testgevallen feitelijk samengevoegd, dit reduceert de oorspronkelijke hoeveelheid testgevallen. "Pairwise" kan worden toegepast op paden en testdata. De tester gebruikt deze techniek als hij weet dat een story teveel permutaties in de parameters oplevert en hierdoor teveel testcases worden afgeleid om te testen.

#### Opzet

Diverse tools zijn gezocht en diverse tools zijn lokaal getest door de stagiair als "proof of concept". PictMaster is hieruit geselecteerd. De tool is middels een "proof of concept" aan de hand van een praktijksituatie die L. Koning hiervoor verschafte. Deze tool is daarna voorgelegd aan het testteam.

#### Interviews

Onder andere bij deze variant is de expertise gevraagd van een externe expert op het gebied van test tooling.

Het ging hier om L. van der Aalst, hij is een testmanager bij Sogeti en expert op het gebied van testen binnen en buiten Nederland en co auteur van Tmap Next.

Uit beide interviews bleek dat "pairwise" testen een erg positieve invloed heeft op de dekkingsgraad. Dit komt doordat het overgrote deel van de fouten niet gevonden wordt door het testen van de normale paden, maar door combinaties hiervan.

Beide heren maakte de kanttekening dat in veel gevallen "pairwise" onvoldoende gebruikt wordt. Volgens L. van der Aalst is dit te wijten aan het feit dat de meeste testers de situatie waarin ze "pairwise" kunnen inzetten niet herkennen of begrijpen.

#### Resultaten en discussie

Door het gebruik van PictMaster (dit is een "pairwise" tool) wordt het aantal testcases bij veel combinaties gereduceerd. De tester krijgt een overzicht van de dekkingsgraad welke behaald wordt bij het uitvoeren van een bepaald aantal testcases. Dit is een grote verbetering op de testdekking.

Het is moeilijk voor het testteam om praktijkvoorbeelden te vinden bij deze vorm. Goede documentatie en een "testtool owner"[8] zijn hierin helpend.

[8] Een tester uit het testteam: voortrekkersrol in documenteren van situaties waarin de tool gebruikt wordt.

### *Variant 4 Dumb Monkey testing*

Dumb Monkey[xlvii] is een "model based"testalgoritme. Deze test de applicatie zonder volledig besef te hebben waar deze is in de applicatie.

De Monkey maakt een model van een staat[9] waar de iBurgerzaken applicatie zich bevindt door het overnemen van de domain object model (DOM). Aan de hand van de elementen[10] die deze ophaalt uit de DOM selecteert hij zoveel mogelijk random inputwaarden. De Monkey vergelijkt bij iedere staatverandering de elementen op invulling. Hierdoor weet deze wat reeds is ingevuld en wat niet.

Doordat de Monkey bijna volledig willekeurig invult en klikt, (met uitzondering van waardes zoals burger service nummers, data kunnen bijvoorbeeld niet volledig random worden gebruikt) komt deze in situaties waar de tester zelf misschien niet op zou komen. De Monkey maakt dus zelf unieke combinaties van invoer. Als de Monkey een foutmelding of crash constateert dan rapporteert hij deze door middel van een tekstbestand met daarin de betrokken parameters en het desbetreffende hoofdstuk.

De Dumb Monkey is gemaakt om "exploratory" tests uit te voeren. Deze kan tevens ingezet worden om systeemtests uit te voeren. Het verschil bij beide tests is dat de "exploratory" test over een langere periode wordt gedraaid en dat deze door veel "content" heen loopt. Bij de systeemtest wordt de Dumb Monkey specifieker ingezet op het deel waar de nieuwe "content" is toegevoegd. Het is dus mogelijk om deze tool in handen te geven van de "developer"en de tester.

#### Opzet

Voordat besloten was om de tool zelf te bouwen is onderzocht of er bestaande tools zijn die Dumb Monkey testen uit kunnen voeren. Uit diverse onderzoeken bleek dat deze wel bestonden. Echter geen enkele tool was beschikbaar om getest te worden. De oorzaak is hiervan onduidelijk. B. Vissers bevestigt dat er vóór de "model based test tool Cover" van Sogeti nog weinig goede tools waren die breed gebruikt werden en die aan de omschrijving voldeden van de Dumb Monkey.

#### Interviews

Uit het interview met B. Visser bleek dat hij de dekkingsvorm grenswaarde erg geschikt vindt voor de Dumb Monkey. Hij gaf hierbij aan dat door het toepassen van grenswaarde over het algemeen veel fouten worden gevonden. Beide geïnterviewde vonden dat "model based testen" een erg toekomst gerichte methodiek is.

[9] Doordat de Monkey op knoppen drukt of velden invult worden staat-veranderingen getriggerd in de applicatie. Dit valt te vergelijken met uiterlijke verandering van de pagina en haar input mogelijkheden.

[10] Tekstvelden, dropdown-velden, datavelden, koppen, tekst, etc.

Uit een meeting met L. Koning bleek dat hij de toepassing van de Dumb Monkey ziet voor de ambtenarenzijde en de burgerzijde van de iBurgerzaken applicatie. Een belangrijk argument om de Dumb Monkey te gebruiken voor de burgerzijde is dat de deze door meer gebruikers benaderd wordt, hierdoor ontstaan ook meer situaties. Deze unieke situaties worden niet allemaal meegenomen in de diverse tests.

### Ontwikkeling

Met J.D van der Velde is overlegd wat de Dumb Monkey zou moeten kunnen. In eerste instantie is ingezet om een pagina te kunnen inladen. Daarna is de functionaliteit aangebracht om input velden in te vullen. Vervolgens is ingezet om op knoppen te klikken. Daarna volgde het invullen van drop down velden enz. Na het invullen van een random waarde is ingezet om specifiekere waardes in te vullen. Hierbij is gekeken of het veld verplicht was en wat voor soort waarde er in moet; date, string, int, boolean.

De volgende stap was om de Monkey door verschillende pagina's te laten lopen binnen de Afstamming applicatie in de iBurgerzaken applicatie. De Monkey heeft later de mogelijkheid gekregen om testcases op te slaan en de functie het kiezen van een startpunt.

Ten behoeven van de Monkey is de applicatie onderzocht met betrekking tot het "breadth first" en "depth first" logaritme.

### Resultaten en discussie

De Dumb Monkey is een makkelijk te gebruiken tool doordat de gebruiker weinig keuzes hoeft te maken bij het gebruik. De tool is bij diverse testen inzetbaar. Bij een "exploratory" test kan zowel de burger als de ambtenarenzijde meegenomen worden. De Dumb Monkey kan verbeterd worden door de toevoeging van de dekkingsvorm grenswaarden.

Door een combinatie van tijdsgebrek en de complexiteit van de implementatie van "breadth" en "depth first" is dit niet gerealiseerd.

Bij Dumb Monkey is documentatie van testgevallen minder belangrijk omdat deze elke keer opnieuw worden gebouwd.

Systeemtesten met de Dumb Monkey gaat relatief snel en is zeer eenvoudig.

De effectiviteit op langere termijn van de tool moet nog worden bewezen door middel van een "proof of concept".

### *Variant 5 Brilliant Monkey testing*

Brilliant Monkey[xlviii] test is voor een groot deel gelijk aan Dumb Monkey testen. Het grote verschil tussen beide vormen is dat de Brilliant Monkey meer intelligentie heeft. Die intelligentie uit zich in de kennis van de staten waarin iBurgerzaken zich bevind. Hierdoor kan de Monkey gericht keuzes maken en intelligente paden doorlopen.

#### Opzet

Door de gelijkenis met de Dumb Monkey is voor deze variant gelijksoortig "deskresearch" gedaan. De ontwikkeling van de Dumb Monkey is de opzet voor de Brilliant Monkey.

#### Interviews

Deze variant is meegenomen in de interviews met L. van der Aalst en B. Visser. Hier werd wederom de conclusie getrokken dat deze variant niet beschikbaar is als tool. Echter een groot deel van de intelligentie is beschikbaar in diverse tools.

Sogeti heeft een tool die deze intelligentie zou kunnen toevoegen aan de Dumb Monkey variant. Deze tool combineert diverse dekkingsvormen als "pairwise", grenswaarde en "model based"testen.

#### Ontwikkeling

Door gebrek aan tijd binnen de stage en de te verwachten hoeveelheid programmeerwerk, is besloten om de Brilliant Monkey niet binnen de stageperiode te ontwikkelen.

#### Resultaten en discussie

De Brilliant Monkey is feitelijk een verbeterde versie van de Dumb Monkey.
Om te bewijzen dat tool van Sogeti (Cover) de intelligentie kan leveren om van de Dumb Monkey een Brilliant Monkey te maken dient deze onderzocht te worden.
Er is meer onderzoek nodig om uit te wijzen of deze tool ontwikkeld dient te worden.

# 6. Conclusies, aanbevelingen en opdrachtevaluatie

In dit hoofdstuk zijn de conclusies getrokken op PATG, het behalen de mijlpalen en doelstellingen. Tevens worden de aanbevelingen uit het adviesdocument benoemd en wordt de stage opdracht geëvalueerd door de ogen van de stagiair.

## 6.1 Mijlpalen

De volgende conclusies over de mijlpalen zijn getrokken:

### Mijlpaal 1 huidige testproces gedocumenteerd.

Deze is behaald, met uitzondering van sommige details. Het documenteren van bijvoorbeeld de ART duurde langer om te beschrijven. Dit heeft te maken met de complexiteit van de ART en de hoeveelheid details. Dit was echter geen probleem voor de voortgang, omdat de focus in de oriëntatiefase niet lag op een detailstudie van de huidige situatie, maar op het grote geheel.

### Mijlpaal 2 basistestopstelling gereed.

Deze is binnen het gestelde termijn behaald, namelijk bijna tegelijk met mijlpaal 1. Achteraf had de stagiair deze wellicht samen kunnen voegen met mijlpaal 1.

### Mijlpaal 3 keuze in "oplosrichting" gemaakt.

Als je kijkt naar de oplosrichting dan is deze behaald in grote lijnen. Alle varianten waren benoemd, met uitzondering van de Monkey tests. Het model waarop de Dumb Monkey gebaseerd is, was toen nog niet compleet. De Brilliant Monkey is wel benoemd, maar nog minder uitgewerkt dan de Dumb Monkey. Dit kwam doordat het "deskresearch" naar de Dumb Monkey langer duurde dan verwacht.

### Mijlpaal 4 Geschikte methodes en tools geselecteerd.

Het model van de Dumb Monkey is voldoende uitgewerkt.
Variant 1 ("code completion library") is hier voldoende uitgewerkt.
Variant 2 ("capture and replay") bestaat nog uit een "longlist", hier is nog geen tool uit geselecteerd. Dit komt omdat achteraf teveel tools met "capture and replay" zijn meegenomen, hier hadden al eerder keuzes gemaakt kunnen worden.
Variant 3 ("pairwise") bestaat nog uit 2 tools waaruit gekozen moest worden.

Voor variant 4 (Dumb Monkey) en 5 (Brilliant Monkey) was al een eerder besloten om deze zelf te ontwikkelen. Besloten was dat variant 5 gezien werd als uitbreiding op variant 4.

### Mijlpaal 5 conclusie of de methode past binnen het huidige testproces.

Van variant 1 t/m 3 is het bewezen dat deze passen binnen het huidige testproces. Variant 4 past theoretisch goed binnen het testproces. Dit is op dat moment nog niet bewezen middels een "proof of concept". Voor variant 5 zijn op dit moment geen resultaten.

## 6.2 Doelstellingen

De volgende conclusies zijn getrokken over de doelstellingen:

**Voor juli 2015 middels een literatuuronderzoek aantonen welke technieken kunnen helpen bij de automatisering van testcases binnen PinkRoccade Local Government.**
Deze doelstelling is behaald. PinkRoccade heeft nu inzicht in "model based"testen, "capture and replay" en "pairwise" testen. Deze technieken zijn meegenomen in het adviesdocument als theoretisch kader en verwerkt in de varianten.

**Voor juli 2015 via een "proof of concept" aantonen of een black box techniek of tool de testdekking van de regressie en systeemtests binnen de business unit Publiekszaken van PinkRoccade Local Government kan verhogen.**
Deze doelstelling is behaald. Bij variant 1 is middels een "proof of concept" aangetoond dat deze een gunstige invloed heeft op de testgeval creatie tijd. Bij variant 2 is dit ook door middel van een "proof of concept" aangetoond. Bij variant 3 hebben een "proof of concept" en interviews dit beide zaken aangetoond. Variant 4 is aangetoond met een "proof of concept", er zijn nog geen langer termijn resultaten. Variant 5 is impliciet bewezen, echter hier moet wel meer onderzoek naar worden gedaan.

**Voor juli 2015 een advies geven of een black box techniek en/of tool het meest geschikt is voor de verhoging van de testdekking van regressie en systeemtests binnen PinkRoccade Local Government.**
Deze doelstelling is behaald. Het adviesdocument concludeerde dat variant 1 en 3 een bijdrage leveren aan de dekking en dat variant 4 dit theoretisch doet.

## 6.3 Aanbevelingen

De volgende aanbevelingen zijn aan PinkRoccade gepresenteerd:

Door de goede resultaten van variant 1 en 3 is het aan te raden deze varianten per direct in te voeren. Variant 3 behoeft goede documentatie en een "testtool owner", door zijn complexiteit in de uitvoering.

Variant 2 is geen geschikte variant doordat testcases zonder veel aanpassingen niet werken. Daarbij is hergebruik van de testcases niet mogelijk.

De Dumb Monkey variant 4 heeft een brede inzetbaarheid en een hoge potentie. De tool kan ingezet worden bij "exploratory"- en systeemtests. Het toevoegen van grenswaarden maakt deze effectiever. Een "proof of concept" van deze tool is aan te raden. Hiermee kan geconcludeerd worden, in te voeren, aan te passen of te stoppen met variant.

Variant 5 is niet ontwikkeld. Een testtool van Sogeti (Cover) kan wellicht de brug slaan tussen de Dumb Monkey en de Brilliant Monkey. Het is daarom nog verstandig deze tool te testen op haar werken en "feasibility" met de Dumb Monkey.

## 6.4 Opdracht evaluatie

De hoofdvraag en de deelvragen zijn goed beantwoord. PinkRoccade heeft nu inzicht in mogelijke methodieken en tools die mogelijkheden bieden het huidige testproces in meerdere mate te automatiseren. Hierbij zijn zowel externe als interne bronnen geraadpleegd. Het adviesdocument geeft bovendien inzicht in de effecten bij het gebruik van de genoemde methodieken en tools.

De probleemstelling is naar mening van de stagiair goed aangepakt. Er zijn concrete oplossingen gerealiseerd die ook als erg waardevol ervaren zijn door PinkRoccade.

Door de veelzijdigheid van het gebruik van methodes, interviews, Brown paper sessies, "proof of concepts" is de onderzoeksvraag uit meerdere hoeken bekeken. Hierdoor is de beantwoording effectief en accuraat.

# 7. Persoonlijke evaluatie

Dit hoofdstuk beschrijft de persoonlijke evaluatie van de student op het PATG. Tevens wordt een terugkoppeling gemaakt op het persoonlijk ontwikkelingsplan wat vóór de stageperiode is opgesteld.

Ik ben erg tevreden over het geleverde product, de hoeveelheid opgedane kennis en de manier waarop het project verlopen is. Ondanks minimale Java programmeervaardigheid en de vele kleine problemen die ik heb ondervonden tijdens de ontwikkeling van de Dumb Monkey applicatie, ben ik erg tevreden over het resultaat van de Dumb Monkey.

De ontwikkelsnelheid van de Dumb Monkey applicatie was in het begin traag, maar werd steeds sneller naarmate de vaardigheid toenam. Het was vooraf moeilijk in te schatten hoeveel tijd hieraan besteed moest worden.

De planning die ik heb gemaakt was reëel en goed uitvoerbaar. Ik had de idee dat ik de controle had over de taken en activiteiten binnen de gestelde tijd.

De doelstellingen vind ik ook goed behaald. In het geval van variant 1 en 3 heb ik nog de stappen naar implementatie gezet d.mv. het maken van handleidingen.

Zelf heb ik er voor gekozen om meer dagen te besteden aan de stage dan de minimale 85 dagen. Dit is geen noodzaak geweest, maar een combinatie van plezier hebben in de opdracht en het maximale willen overdragen aan het einde van de stage periode.

## *Reflectie op POP*

**Mijn persoonlijk doel van mijn stage was het vaardiger worden in programmeren en technisch ontwerpen.**

**De volgende leerdoelen heb ik mijzelf gesteld:**

1. **Programmeervaardigheden vergroten.**
2. **Mijn product moet gemaakt worden aan de hand van een goed onderzoek.**
3. **Vaardiger worden in het doen van onderzoek naar testmethodes, test technieken.**
4. **Kennis over testmethodes vergroten.**

**Naar mijn idee heb ik mijn leerdoelen behaald. Door het programmeren van de Dumb Monkey heb ik veel geleerd op het gebied van Java, Selenium Webdriver en Jsoup.**

**Door mijn onderzoek naar testen in het algemeen, de ART en de interviews heb ik een goed beeld van testprocessen testvormen en testmethodieken.**

# BIBLIOGRAFIE


[i] (agilealliance, 2015)
[ii] (https://subversion.apache.org/, 2015)
[iii] (Neijts, Advies document project automatische testcase generatie, 2015, pp. 25, 39, 43)
[iv] (https://cucumber.io/, 2015)
[v] (https://jenkins-ci.org/, 2015)
[vi] (https://maven.apache.org/, 2015)
[vii] (https://www.ruby-lang.org/en/, 2015)
[viii] (Neijts, Advies document project automatische testcase generatie, 2015)
[ix] (Neijts, Project initiation document: Project automatische testcase generatie, 2015)
[x] (Neijts, Advies document project automatische testcase generatie, 2015, pp. 44,45)
[xi] (vereniging-van-Nederlands-gemeenten, 2015)
[xii] (Neijts, Advies document project automatische testcase generatie, 2015, pp. 8,9)
[xiii] (Government, PinkRoccade Local, iBurgerzaken, 2015)
[xiv] (DenisonConsulting, Organizational-culture, 2015)
[xv] (Government, PinkRoccade Local, Merkkompas, 2012)
[xvi] (Neijts, Project initiation document: Project automatische testcase generatie, 2015, pp. 2,6)
[xvii] (Neijts, Advies document project automatische testcase generatie, 2015, p. 21)
[xviii] (Neijts, Project initiation document: Project automatische testcase generatie, 2015, p. 6)
[xix] (Neijts, Advies document project automatische testcase generatie, 2015, p. 11)
[xx] (Neijts, Project initiation document: Project automatische testcase generatie, 2015, pp. 5,6), (Neijts, Advies document project automatische testcase generatie, 2015, p. 18 t/m 21)
[xxi] (Neijts, Advies document project automatische testcase generatie, 2015, p. 18 t/m 21)
[xxii] (Neijts, Scriptie Project automatische testcase generatie, 2015, p. 19)
[xxiii] (Neijts, Advies document project automatische testcase generatie, 2015, p. 16 t/m 19)
[xxiv] (Neijts, Advies document project automatische testcase generatie, 2015, pp. 15,17,19,23,24,26,28)
[xxv] (Neijts, Advies document project automatische testcase generatie, 2015, pp. 5,6,7)
[xxvi] (Neijts, Project initiation document: Project automatische testcase generatie, 2015, p. 7)
[xxvii] (Neijts, Advies document project automatische testcase generatie, 2015, pp. 46,47)
[xxviii] (Neijts, Advies document project automatische testcase generatie, 2015, p. 21 t/m27)
[xxix] (Neijts, Project initiation document: Project automatische testcase generatie, 2015, pp. 9,10)
[xxx] (Neijts, Project initiation document: Project automatische testcase generatie, 2015, p. 11)
[xxxi] (Neijts, Advies document project automatische testcase generatie, 2015, p. 9 t/m 11)
[xxxii] (Neijts, Project initiation document: Project automatische testcase generatie, 2015, pp. 9,16)
[xxxiii] (Trello, StageJelle, 2015)
[xxxiv] (Neijts, Advies document project automatische testcase generatie, 2015, p. 25)
[xxxv] (Neijts, Advies document project automatische testcase generatie, 2015, pp. 24,25)
[xxxvi] (http://jsoup.org/, 2015)
[xxxvii] (http://www.seleniumhq.org/projects/webdriver/, 2015)
[xxxviii] (Neijts, Advies document project automatische testcase generatie, 2015, p. 49 t/m 52)
[xxxix] (Koomen, Aalst van der, Broekman, & Vroon, Tmap Next voor resultaat gericht testen, 2006)
[xl] (Neijts, Advies document project automatische testcase generatie, 2015, p. 5 t/m 7)
[xli] (Neijts, Advies document project automatische testcase generatie, 2015, p. 29)
[xlii] (Neijts, Advies document project automatische testcase generatie, 2015, p. 48)
[xliii] (Neijts, Advies document project automatische testcase generatie, 2015, pp. 15,16)
[xliv] (Neijts, Advies document project automatische testcase generatie, 2015, pp. 20,21,34 t/m 36, 40, 41, 45, 47)
[xlv] (Neijts, Advies document project automatische testcase generatie, 2015, pp. 21,22,37,41,42, 45, 47)
[xlvi] (Neijts, Advies document project automatische testcase generatie, 2015, pp. 22,23,38,42,43,45,47)
[xlvii] (Neijts, Advies document project automatische testcase generatie, 2015, pp. 24,25,38,39,43,44, 46,47)
[xlviii] (Neijts, Advies document project automatische testcase generatie, 2015, pp. 26,40, 44, 46, 47)

# BIJLAGE A: Project initiation document

# Project automatische testcase generatie


**PLAATS** 's Hertogenbosch

**AUTEUR** Dhr. J Neijts

**FUNCTIE** Stagiair


**VERSIE** 1.1

# Documenthistorie

## Revisies

| Documentbeheer | | | | |
|---|---|---|---|---|
| Versie | Status | Auteur | Datum | Opmerking |
| 0.1 | Concept | J Neijts | 09-02-2015 | Aanleiding geschreven |
| 0.2 | Concept | J Neijts | 11-02-2015 | Doelstellingen beschreven, Aanpak beschreven, Projectstrategie beschreven, planning gemaakt, scope bepaald, producten/eindresultaten |
| 0.3 | Concept | J Neijts | 12-02-2015 | Projectorganisatiestructuur, afhankelijkheden, aannames, projectbeheersing. |
| 0.4 | Concept | J Neijts | 13-2-2015 | Aanpassen achtergrond, feedback van Dhr. L van Gorp toevoegen, feedback Dhr. L Koning achtergrond. |
| 0.5 | Concept | J Neijts | 13-2-2015 | Aanpassingen gedaan aan de hand van Dhr. J van der Velde zijn feedback. |
| 0.6 | Concept | J Neijts | 19-2-2015 | Aanpassingen gedaan aan de hand van Dhr. L Koning zijn feedback. Inleiding en managementsamenvatting geschreven. |
| 0.7 | Concept | J Neijts | 23-2-2015 | Producten uitleg uitgebreid. |
| 0.8 | Concept | J Neijts | 27-02-2015 | Toevoegen bijlage C en D |
| 0.9 | Concept | J Neijts | 27-02-2015 | Toevoegen feedback Dhr J van der Velde. |
| 1.0 | Concept | J Neijts | 6-3-2015 | Toevoegen feedback Dhr L van Gorp. Verwijderen bijlage C en D. |
| 1.1 | Concept | J Neijts | 10-3-2015 | Feedback Dhr L Koning |

## Distributie

Dit document is verstuurd aan:

| Distributielijst | | | |
|---|---|---|---|
| **Naam** | **Organisatie / Afdeling** | **Datum** | **functie** |
| 0.3 | 12-2-2015 | Dhr. L van Gorp | Assessor 1 |
| 0.4 | 13-2-2015 | Dhr. J van der Velde | Projectmanager |
| 0.5 | 13-2-2015 | Dhr. L Koning | Projectsupport |
| 0.8 | 27-02-2015 | Dhr. J van der Velde | Projectmanager |
| 0.9 | 2-3-2015 | Dhr L van Gorp | Assessor 1 |
| 1.0 | 9-3-2015 | Dhr. L Koning | Projectsupport |

# Managementsamenvatting


### Aanleiding

De projectteams van de iBurgerzaken applicatie hebben de behoefte om een hogere testdekking te behalen. Momenteel zijn er bevindingen die mogelijk voorkomen hadden kunnen worden als de testdekking groter was.

Deze verhoging van de testdekking is gewenst tijdens de regressie en systeemtesten van de iBurgerzaken applicatie.

### Globale doelstelling

Het project automatische testcase generatie (PATG) adviseert de opdrachtgever in hoeverre de testdekking verhoogd kan worden door volledige of gedeeltelijke automatisering van testcases.

### Globale business case

Door een vorm van automatisering toe te passen op de generatie van testcases kunnen in dezelfde tijd meer testcases worden gemaakt. Aangenomen is dat een toename aan testcases een toename aan testdekking veroorzaakt. Bij een hogere testdekking zullen meer bugs verdwijnen uit de iBurgerzaken applicatie waardoor de kwaliteit van de applicatie en voor de gebruiker toeneemt.

### Globale aanpak

De huidige testsituatie binnen PinkRoccade Local Government's business unit Publiekszaken wordt in kaart gebracht op proces, applicatie en test technisch niveau. Hiervoor worden vooral interviews en literatuur gebruikt die testmethodieken beschrijft.

PATG zal door middel van deskresearch uitwijzen welke testmethoden en testtechnieken bij kunnen dragen aan het verhogen van de testdekking van de regressie en systeemtest van de iBurgerzaken applicatie. Daarbij wordt in specifiek gezocht naar testvormen en testtechnieken die de testcases van deze tests volledig of deels kunnen automatiseren.

Tevens zal het deskresearch zich richten op het vinden van (mogelijke) testtools die zijn gebaseerd op de testmethodes en testtechnieken die als geschikt of als veelbelovend zijn gecategoriseerd.

Een proof of concept zal uitwijzen welke mogelijke oplossingen het meest geschikt zijn voor de verhoging van de testdekking.

### Globale doorlooptijd

Het project zal tussen februari 2015 en juli 2015 plaats vinden.
Het project zal drie fases doorlopen: oriëntatie, onderzoek en afronding. Tijdens de oriëntatiefase wordt de probleemstelling helder gemaakt. In de onderzoeksfase wordt inhoudelijk onderzoek gedaan naar oplossingen die de hoofd en deelvragen beantwoorden. In de afrondingsfase worden de conclusies getrokken van het onderzoek en worden de aanbevelingen gedaan. Door middel van mijlpalen wordt de status ven de fasering gewaarborgd.

# Inhoudsopgave

# 1 Inleiding

Dit document geeft de projectinformatie weer van het project automatische testcase generatie. Aan de hand hiervan wordt het project gemonitord.

Als dit project voltooid is heeft de ontwikkelafdeling Publiekszaken van PinkRoccade Local Government antwoord op de hoofdvraag:

*"In hoeverre zijn er mogelijkheden om de testdekking van het regressie en systeemtestproces te vergroten door een vorm van testcases automatisering?"*

Met deze informatie kan een volgende stap gemaakt worden om het regressie testproces te verbeteren.

**Doelstellingen**

- Voor juli 2015 middels een literatuuronderzoek aantonen welke technieken kunnen helpen bij de automatisering van testcases binnen PinkRoccade Local Government.

- Voor juli 2015 aantonen via een proof of concept of een black box techniek of tool de testdekking van de regressie en systeemtest binnen de business unit Publiekszaken van PinkRoccade Local Government kan verhogen.

- Voor juli 2015 een advies geven of een black box techniek en/of tool het meest geschikt is voor de verhoging van de testdekking van regressie en systeemtest binnen PinkRoccade Local Government.

Door een projectmatige aanpak zullen de doelstellingen worden beantwoord met de volgende producten die worden opgeleverd binnen de periode van februari 2015 t/m juni 2015.

**Producten:**
- Projectinitation document (PID)
- Adviesdocument met business case (AD)
- Proof of concept (POC)
- Dagelijkse rapportages
- Scriptie

Het project zal worden geleid door J. van der Velde die zal toezien op de voortgang en doorgang van het project. L. Koning is projectsupport, hij zal zijn kennis delen over testprocessen en support geven op technisch gebied.
Het onderzoek en de vervaardiging van de producten zal door J. Neijts worden gerealiseerd (projectlid).

# 2 Achtergrond

PinkRoccade local government is een dienstverlener van informatietechnologische oplossingen voor lokale overheden. Deze oplossingen worden aangeboden in de vorm van bedrijfssoftware waarmee de lokale overheid haar dienstverlenende zaken kan regelen. PinkRoccade local government is onderdeel van Total Specific Solutions (TSS). Binnen PinkRoccade local government zijn vier business units: Publiekszaken, Bedrijfsvoering, Ruimte en omgeving en Samenlevingszaken. Het project automatische testcase generatie (PATG) vindt plaats onder de business unit Publiekszaken te 's Herthogenbosch. De business unit Publiekzaken houdt zich bezig met een viertal bedrijfssoftware oplosrichtingen, namelijk in de richting van documentmanagement, Zaaksuite, WebNext Zaaksysteem en burgerzaken.

Documentmanagement houdt zich bezig met de ontwikkeling van iDocumenten. Dit is een softwaresysteem wat functionarissen bij de lokale overheden in staat stelt om gezamenlijk te werken aan zaken. Zaaksuite is een verzameling van applicaties die het mogelijk maken om zaakgericht te werken. Dit zijn onder andere Makelaarsuite CiVision Zaakafhandeling en CiVision Klant Contact Centrum. WebNext Zaaksysteem is een oplossing in de mid en frontoffice dat de communicatie tussen de burger en de lokale overheid regelt.

PATG richt zich op de applicatie iBurgerzaken. Momenteel vindt het programma modernisering gemeentelijke basisadministratie (mGBA) plaats bij de lokale overheden. Dit houdt in dat lokale overheden een voorbereiding maken om hun gegevens te koppelen aan de landelijke database (basisregistratie personen BRP). PinkRoccade Local Government biedt bij deze verandering een oplossing in de vorm van iBurgerzaken. Dit is een softwarepakket dat bestaat uit veertien standaard apps en enkele specifieke apps om deze verandering te kunnen realiseren.

Op dit moment werken er vijf ontwikkelteams aan de iBurgerzaken applicatie.
De iBurgerzaken applicatie wordt met de Scrum software ontwikkelmethode ontwikkeld. Scrum is een agile software ontwikkelmethode waarbij in teams gewerkt wordt aan het vervaardigen van functionele delen van de applicatie. Functionele eisen vanuit de product back log worden opgedeeld in kleinere taken. Deze worden opgenomen in de sprint back log. Een sprint is een periode waarin de sprint back log van iBurgerzaken wordt ontwikkeld.

Het testen van de iBurgerzaken applicatie is een belangrijk onderdeel van de ontwikkeling. De applicatie en de delen daarvan worden getest om te zorgen dat: de functionaliteiten naar behoren werken, de applicatie werkt bij grootschalig gebruik, de applicatie werkt op verschillende platformen, alle bekende browsers en de applicatie aansluit bij de behoeften van de eindklanten. Er zijn verschillende testvormen die de ontwikkelteams van iBurgerzaken gebruiken om de diverse aspecten van de applicatie te testen. PATG richt zich uitsluitend op het regressietesten en systeemtesten.

Bij regressietesten wordt getest of het bestaande systeem nog functioneert zoals verwacht bij toevoeging van nieuwe functionaliteit. De acceptatietest test of de nieuwe functionaliteit binnen de totale iBurgerzaken applicatie.

Het gedrag van het systeem wordt getest door middel van testcases. De uitvoer van testcases zijn deels geautomatiseerd en deels handmatig. Door een user story te beschrijven in de vorm van een feature, kunnen de use cases, feitelijk testcases, worden gekoppeld.

Als er een release is samengesteld na een bepaald aantal sprints vindt er een regressietest plaats. Deze regressietest wordt deels gedaan met automated regression tests (ART's) en deels met de hand. Dit houdt in dat uit een lijst met scenario's (testsuite), de scenario's worden geselecteerd die worden gebruikt voor de test. Deze selectie wordt gedaan om de belangrijkste scenario's mee te nemen in de test. Hierbij wordt gewogen welke scenario's het meeste risico dragen en, hoe complex de scenario's zijn. Bovendien worden eerdere ervaringen meegenomen. In de uitvoer van de scenario's is geen vaste volgorde. Het vervaardigen van test scenario's is een tijdsintensieve taak wanneer alle userstory's getest moeten worden, daarom worden meestal alleen de test scenario's uitgevoerd waarbij het risico het hoogst is.

Er is zeer veel reeds bestaande functionaliteit. Het is onmogelijk om dat in een kort tijdsbestek (enkele dagen) te testen. Door deze methode is er een zekerheid dat je als tester niet alle functionaliteit meeneemt maar grofweg 80%[1]. Elk scherm bevat een grote hoeveelheid 'basisfunctionaliteit' welke door het uitvoeren van bestaande geautomatiseerde en handmatige test scenario's niet of nauwelijks geraakt wordt.

Het testteam van de afdeling publiekzaken heeft behoefte aan een hogere mate van testverbreding naast de bestaande uitvoer van de regressie en systeemtests. Deze vraag komt voort uit de constatering[2] dat in de productomgeving incidenten zijn geweest die met testverbreding voorkomen hadden kunnen worden. Hieruit is de behoefte ontstaan om test scenario's automatisch te genereren. De vraag bestaat of dit een mogelijkheid is en als dit zo is op welke manieren dit zou kunnen. Met name het exploratory testing wat theoretisch lastig te automatiseren is zal een grote bijdrage leveren aan de verbreding van het testen.

Welke zoekalgoritmes geschikt kunnen zijn om te gebruiken voor het automatiseren van het testproces moet worden onderzocht omdat hier momenteel binnen de organisatie te weinig kennis over is. Bovendien is de vraag er of er bestaande tools zijn die kunnen bijdragen aan deze automatiseringsslag.

Als blijkt uit onderzoeksresultaten dat deze algoritmes en tools bruikbaar zijn dan roept dit de vraag op: hoe deze kunnen worden ingezet binnen de huidige situatie? Momenteel zijn er geen projectgroepen of individuen die onderzoek doen naar deze mogelijkheden. vereist. Het beantwoorden van de hoofdvraag zal kennis vragen van diverse functies binnen PinkRoccade Local Government. De belangen van dit project zijn in potentie hoog. Door deze factoren in combinatie met de complexe de vraagstukken is een projectmatige aanpak nodig.

[1] Deze schatting is gedaan aan de hand van een interview met Dhr L Koning.
[2] Dit argument komt voort uit een interview met dhr. Laurens Koning (tester).

# 3 Projectdefinitie

## 3.1 Projectdoelstellingen

Het project is in het leven geroepen om te beoordelen of het automatisch genereren van test scenario's van een regressietest en systeemtest met behulp van black box testing mogelijk is. Naast de vaststelling of dit mogelijk is wil PinkRoccade Local Government een beeld krijgen op welke manieren dit mogelijk gemaakt kan worden. PinkRoccade wil inzicht krijgen in de mogelijke effecten van de toepassing van project automatische testcase generatie binnen de organisatie.

Na de uitvoering van dit project heeft PinkRoccade een inzicht in de mogelijkheden van automatische testcase generatie en kan PinkRoccade aan de hand van aanbevelingen en een business case een besluit nemen of automatische testcase generatie volledig of gedeeltelijk toegepast zal worden.

Concreet zijn de doelstellingen als volgt benoemd:

- Voor juli 2015 middels een literatuuronderzoek aantonen welke technieken kunnen helpen bij de automatisering van testcases binnen PinkRoccade Local Government.

- Voor juli 2015 via een proof of concept aantonen of een black box techniek of tool de testdekking van de regressie en systeemtests binnen de business unit Publiekszaken van PinkRoccade Local Government kan verhogen.

- Voor juli 2015 een advies geven of een black box techniek en/of tool het meest geschikt is voor de verhoging van de testdekking van regressie en systeemtests binnen PinkRoccade Local Government.

## 3.2 Gekozen oplossing of aanpak

Om de betrokkenen van dit onderzoek goed met elkaar samen te laten werken en de resultaten tijdig op te kunnen leveren is een projectmatige aanpak vereist.
Er wordt een projectteam[3] opgezet binnen de business unit publiekzaken van PinkRoccade local government. Het projectteam is een verzameling van actief betrokkenen van het project welke zijn geselecteerd in overleg met de projectmanager en de stagiair.

Om deze doelstellingen en deliverables te kunnen realiseren is een hoofdvraag met daarbij horende deelvragen geformuleerd.

*"In hoeverre zijn er mogelijkheden om de testdekking van het regressie en systeemtestproces te vergroten door een vorm van testcases automatisering?"*

**Deelvragen:**

- Welke methodes en tools (intern[4] en extern) worden bij PinkRoccade local government ingezet om een test scenario te genereren en uit te voeren in de huidige situatie?
- Welke tools(extern en intern) kunnen bijdragen aan de automatisering van de generatie van test scenario's?
- Welke methodes zijn beschreven in de literatuur die kunnen worden geraadpleegd om test scenario's deels of volledig automatisch te genereren?
- Wat zijn de mogelijke voor-en-of nadelen van gedeeltelijke of volledige geautomatiseerde generatie van test scenario's ten opzichte van de huidige situatie?

[3] Zie 4 projectorganisatiestructuur voor een uitgebreide beschrijving van het projectteam.
[4] Interne tool: door PinkRoccade reeds in gebruik, externe tool: tool afkomstig uit de buitenwereld.

*Onderzoeksmethodes*
De volgende methodes en technieken worden ingezet om antwoord te krijgen op de hoofd en deelvragen:

- *Deskresearch*
  In het begin richt de stagiair zich op mogelijke bestaande tools en applicaties op de markt die een bijdrage kunnen leveren aan de automatiseren van testcases. Daarnaast zoekt de stagiair ook naar de bestaande methodieken die inzicht geven in het huidige testproces en wellicht toe gepast kunnen worden in de toekomstige situatie.

  Door het gebruik van zoekmachines en wetenschappelijke databases worden bestaande onderzoeken naar black box testing, automatisch testen, en onderzoek naar regressietest en systeem test methodes geïnventariseerd. Door het lezen van literatuur over bestaande methodieken, technieken en tools krijgt de stagiair een beeld van de nieuwe ontwikkelingen en bestaande testprocessen en methodieken. Het deskresearch zal starten door het maken van een lijst met keywords die betrekking hebben op het onderzoek. Deze lijst zal worden aangevuld door voortschrijdend inzicht en door input in de vorm van bedrijfsstukken en interviews en bovendien input uit deel 2 van het proof of concept.

  De mogelijke methodieken en tools die een bijdrage kunnen leveren aan de automatisering van testcases zullen in een longlist worden opgenomen. Na beoordeling van de bronnen zal een shortlist ontstaan wat input geeft voor het proof of concept. Deze shortlist wordt aan de hand van keuzes gedaan die samen met het projectmanagement worden gemaakt. Deze keuzes gaan over richtingen van methodieken en technieken.

- *Interviewen*
  Door de juiste vragen te stellen aan de juiste personen ontstaat een holistisch beeld van de huidige situatie. Deze interviews vinden op de volgende manieren plaats: meelopen met specifieke testwerkzaamheden, bijwonen van een daily start-up, testmeetings en feedback van het projectsupport of indirect betrokkenen.

  Om een beter algemeen beeld te krijgen van de toekomst van automatische testcase generatie wordt door de stagiair contact opgenomen met een autoriteit op dit gebied. Deze autoriteit is op hoog niveau bezig met testen en heeft veel inzicht in de mogelijke manieren van testcase generatie. De selectie van een autoriteit zal in overleg worden gedaan met J. van der Velde.

- *Proof of concept*
  De doelstellingen van het proof of concept zijn: beter begrijpen van de huidige situatie, testen van tools en methodieken en als hoofddoel de bewijsvorming voor de business case en bewijsvorming voor de conclusie in het adviesdocument.

  Het proof of concept wordt met behulp van de literatuur, het internet en de kennis van Dhr. J van der Velde en Dhr. L Koning vervaardigd. Deze kennis zal gebruikt worden wanneer de stagiair vast loopt, maar kan ook extra inzichten verschaffen. Eveneens kan een beroep worden gedaan op Java experts uit de ontwikkelteams van iBurgerzaken.

  Het proof of concept is een product dat uit drie onderdelen bestaat. Het eerste deel (basisopstelling) geeft de regressie/ systeem testopstelling weer zoals die op dit moment is. Hiermee wordt inzicht verkregen in de huidige tools en

methodieken. Dit wordt bereikt door een eenvoudig zelfgemaakte webapplicatie te testen met de tools en methodieken zoals die in de huidige situatie worden gebruikt. Dit inzicht geeft input voor de toekomstige situatie en dient als uitgangspunt voor de tools en methodieken die mogelijk worden gevonden.

Deel twee test de tools of methodes die zijn gevonden met behulp van het deskresearch en zijn gedocumenteerd in de shortlist in een simpele testopstelling.

Hiermee wordt inzicht verkregen in de werking van de tool of methode en wordt deze beoordeeld op zijn werking en eigenschappen. De eigenschappen zijn gedocumenteerde eigenschappen die terug moeten komen in de toekomstige situatie. Deze worden in overleg met de projectmanager benoemd. De mate waarin wordt getest wordt overlegd met het projectmanagement.

Het verkrijgen van inzichten over de tools wordt bereikt door de tool toe te passen in de basisopstelling en deze te vergelijken aan de eigenschappen. Als er meerdere tools gevonden worden met grotendeels dezelfde functionaliteit dan kunnen in dit deel ook benchmark tests worden gedaan om te besluiten met welke tool verder te gaan. De tools die gebruikt worden komen uit het deskresearch.

Deel drie is een aaneenschakeling van tools die samen een geautomatiseerde testcase generatie tool vormen en hiermee het volledige regressietest/systeem proces automatiseren. Deze automatisering is een combinatie van het genereren van testcases en het geautomatiseerd uitvoeren hiervan.

Deze aaneenschakeling van tools wordt toegepast op de basisopstelling met 1 kleine aanpassing namelijk nu worden de schakeling van tools toegepast op een pagina van de iBurgerzaken applicatie. Dit product vormt het prototype van de toekomstige situatie.

Het proof of concept wordt getoetst aan de hoeveelheid tests die in de beperkte testtijd gebouwd kunnen worden. Aangenomen is dat een verhoging van testcases de testdekking vergroot, mits deze niet te veel overlappen. Deze overlapping kan theoretisch worden tegengegaan door de toepassing van algoritmes. Deze algoritmes moeten nog nader worden bepaald.

*Planning, fasering en voortgang*

Om de projectvoortgang te waarborgen bestaat er een masterplanning[5] in de vorm van een Excel Gantt chart waarin de deliverables van het project zijn uitgezet tegen de tijd. Tevens geeft deze chart een inzicht in de mijlpalen binnen het project. Deze mijlpalen zijn meetpunten om de voortgang van het proces te bewaken en zijn tevens de ijkpunten van de fasering. Aan de hand van deze mijlpalen kan het project waar nodig worden bijgestuurd. De fasering met mijlpalen van het project zijn als volgt:

[5] Zie bijlage A

De volgende mijlpalen zijn opgesteld:

| Mijlpaal | Fase |
|---|---|
| ***Mijlpaal 1 huidige testproces gedocumenteerd:*** Deze mijlpaal is cruciaal omdat dit input geeft voor zowel het PID maar ook het AD, de scriptie maar ook basis is voor de POC. Als het huidige proces niet volledig in kaart is bestaat het gevaar er onvoldoende kennisoverdracht is geweest, zo kan het project buiten scope raken en haar doelstellingen niet haalt. | Oriëntatiefase: In deze fase wordt de scope van het project, de achtergrond en de voorwaarden gedefinieerd. Definitiefase: Hier worden de eisen ten behoeven van de soll situatie benoemd. |
| ***Mijlpaal 2 POC basistestopstelling gereed:*** Om een bepaalde vaardigheid te krijgen in het gebruik van diverse ontwikkeltools is het van belang hier voldoende kennis over te ontwikkelen om zo binnen de gestelde tijd te komen tot het draaien van de huidige testopstelling op de lokale pc van de stagiair. | Onderzoeksfase: In deze fase wordt het deskresearch en het proof of concept deel één en twee en drie vervaardigd. |
| ***Mijlpaal 3 keuze in oplosrichting gemaakt*** Als het deskresearch wijst op 2 of meerdere uiteenlopende onderzoeksrichtingen zijn dan wordt voor dit moment en na mijlpaal twee besloten welke oplosrichting mee door wordt gegaan | |
| ***Mijlpaal 4 Geschikte methodes en tools geselecteerd:*** Een long en short list is een essentiële stap en vormt de basis van de aanbevelingen en conclusies. | |
| ***Mijlpaal 5 conclusie of de methode past binnen het huidige test proces:*** Er moeten genoeg aanknopingspunten en sub conclusies zijn gedaan om een goed advies document op te leveren. Hier komt alles samen. | Afrondingsfase: Alle resultaten zijn binnen, op dit moment worden deze verwerkt tot conclusies en adviezen |

*Planning*
De masterplanning is het overzicht van het project en haar deliverables echter de wekelijkse planning wordt bijgehouden met de tool Trello. Het projectmanagement kan op deze manier op on demand monitoren waar de stagiair mee bezig is en wat hij heeft gedaan.

*Dagelijkse voortgang*
De communicatie over de dagelijkse voortgang met projectmanagement en project lid vindt plaats met de tool Slack. Slack is een tool om topics aan te maken en feeds te kunnen ontvangen uit Trello en Bitbucket. Bitbucket is een repository om code mee te delen die lokaal verzonden wordt via GIT(lokale repository). Deze repository's worden gebruikt om de code van de proof of concept mee te delen.

De verzameling van dagelijkse rapportages gericht aan de assessor van de Fontys Hogeschool worden wekelijks uitgebracht via de tool Slack.

### 3.3 Scope van het project

Het project zal plaatsvinden bij PinkRoccade Local Government onder de business unit Publiekszaken. Binnen de business unit zijn ontwikkelteam 1 en 2 betrokken bij het project. Een mogelijke uitrol kan worden gedaan naar andere business units binnen PinkRoccade Local Government, maar hier wordt initieel geen rekening mee gehouden. Het onderzoek richt zich daar niet op.

De automatisering van de test scenario's wordt uitgevoerd op het regressietestproces, en het systeemtesten van de applicatie iBurgerzaken. Er zal uitsluitend onderzoek worden gedaan naar black box testmethodes.

Hier is voor gekozen omdat de iBurgerzaken applicatie in zijn geheel uitsluitend getest kan worden met een black box methode door de onderliggende gelaagdheid van ontwikkeltalen. Bovendien wordt met de unit test de code individueel al goed getest.

Het project zal zich in eerste instantie richten op het zoeken van bestaande tools die bijdragen aan de automatisering van testcases. Wanneer dit niet het geval is dan zal het onderzoek zich richten op de ontwikkeling van een dergelijke tool.

### 3.4 Producten c.q. eindresultaat

Het project zal als eindresultaat de volgende producten opleveren ten behoeven van de opdrachtgever. Elk product is opgebouwd uit diverse deliverables per deelproduct en heeft een relatie met een andere product. Hieruit volgt de volgende productdecompositiestructuur[6]. De hoofdproducten zijn als volgt:

- Projectinitation document (PID)
- Adviesdocument met business case (AD)
- Proof of concept (POC)
- Dagelijkse rapportages

Bovendien zullen de volgende voor studie relevante producten worden opgeleverd:

- Scriptie

*Project initiation document:*
In het PID wordt de probleemstelling en de oplossing van het project automatische testcase generatie verklaard. Bovendien geeft dit document weer welke producten, deliverables en milstones moeten worden gehaald in het bepaalde tijdsbestek door wie en met wie. Dit document is de leidraad van het project en is tevens een formele afspraak over de inhoud van de afstudeerstage.

*Adviesdocument met business case*
Dit document geeft antwoord op de hoofd en deelvragen van het project. Dit document levert de conclusie en aanbevelingen voor de opdrachtgever van het project. De business case zal de vergelijking maken tussen de testdekking in de huidige situatie en de testdekking van de toekomstige situatie.

Proof of concept
Dit product geeft input voor het adviesdocument en de scriptie. De POC is een tool om methodieken en externe tools te testen en te boordelen binnen een gecontroleerde omgeving. Aan de hand van deze tests worden conclusies getrokken over de toepasbaarheid en kwaliteits toe of afname van het regressie/systeem test proces. Het proof of concept wordt samengesteld uit bestaande tools en of methodieken. Er zullen geen nieuwe algoritmes, methodieken of complete tools worden geproduceerd.

[6] Zie bijlage B productdecompositiestrucruur

*Scriptie*
Dit product is bedoeld voor de stagiair en diens studie. Dit document geeft de totstandkoming weer van de stage en geeft uitleg over de opgeleverde producten.

### 3.5 Afhankelijkheden

De testers werken met het programma Eclipse, zij gebruiken dit als een JAVA ontwikkelplatform. Bij vragen over de werking van het POC zal er een betere aansluiting zijn als de stagiair de Integrated development environment (IDE) Eclipse gebruikt. Het vervaardigen van software wordt ook in de taal JAVA gedaan omdat hier de expertise voor is.

Het gebruik van de DSL Gherkin waarin de testcases worden gemaakt is niet noodzakelijk. Dit geld ook voor de Tool Jenkins.

### 3.6 Randvoorwaarden

Om het project doorgang te kunnen laten vinden is het gebruik van de volgende zaken cruciaal:

- Noodzakelijke software: MS office, Eclipse, trial versies van tooling die bij kan dragen aan de automatisering van testcase.
- Toegang tot internet
- Toegang tot een image van de iBurgerzaken applicatie
- Commitment van het projectteam

# 4 Projectorganisatiestructuur

De projectorganisatiestructuur is beschreven in Figuur 1 Projectorganisatiestructuur. De taken en verantwoordelijkheden van de betrokkenen bij het project als volgt geformuleerd:

## 4.1 Opdrachtgever

De opdrachtgever is eigenaar van het project en zal de eindproducten in ontvangst nemen. De opdrachtgever wordt geïnformeerd door projectmanager. Dhr. B. Gelsing is de opdrachtgever.

## 4.2 Projectmanager

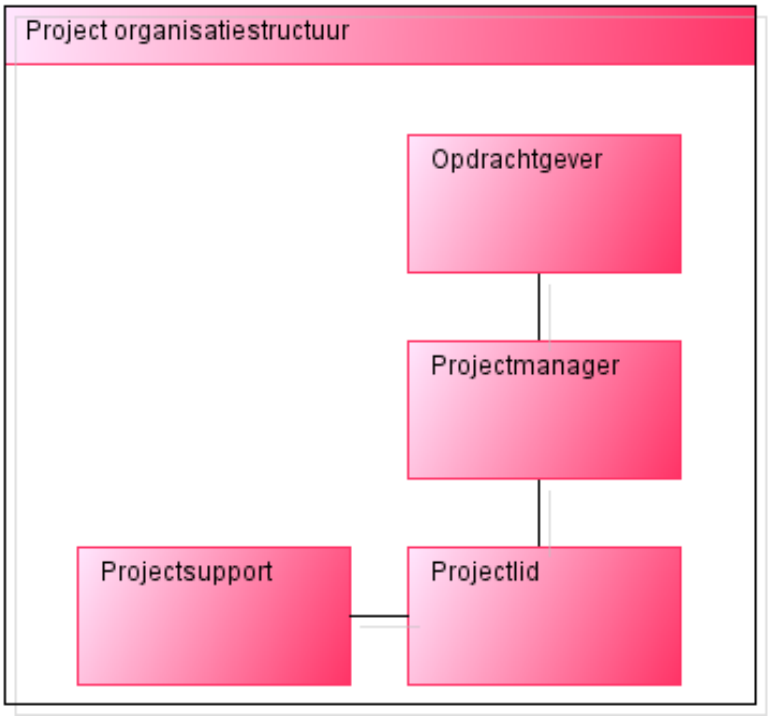
De project manager is verantwoordelijk voor de voortgangsbewaking, scope bewaking en communicatie met de opdrachtgever. Dhr. J. van der Velde is Projectmanager

## 4.3 Projectsupport

Het projectsupport biedt ondersteuning bij het project in de vorm van kennis deling op gebied van het testproces. Tevens biedt het projectsupport ondersteuning bij het gebruiken van de iBurgerzaken image en tools als Maven en Jenkins die worden gebruikt in de huidige situatie. Dhr. L. Koning is projectsupport.

## 4.4 Project lid

Het project lid doet de uitvoering van het project. Hij is verantwoordelijk voor de tijdige oplevering van de deliverables en eindproducten. Dhr. J. Neijts is project lid.




Figuur 1 Projectorganisatiestructuur

# 5 Projectbeheersing

## 5.1 Voortgangsbewaking

De voorgang van het project wordt vastgesteld door het gebruik van meetings en beoordelingspunten. *Tabel 1 Voortgangsbewaking* geeft een weergave van de voortgangsbewaking.

| Voortgangsbewaking | | | | | |
|---|---|---|---|---|---|
| **Overleg** | **Aanwezig** | **Frequentie** | **Tijdstip** | **Doel** | **Onderwerpen** |
| **Overleg met bedrijfsbegeleider** | Projectmanagement | Wekelijks | Nader te bepalen | Feedbacksessie en algemene voortgangsgesprek | Wat heeft de stagiair gedaan, wat gaat hij doen, waar zitten de problemen |
| **Rapportage dagelijkse werkzaamheden** | Projectmanagement | Dagelijks via Trello, Slack | On demand | Sturing van het project. | |
| **Beoordeling PID** | Projectmanagement<br><br>Assessoren Fontys | | Week 5 | Goedkeuring van het PID, Go van het project. | Project doorgang. |
| **Mijlpalen** | Projectmanagement | 5 momenten | De mijlpalen worden besproken tijdens het wekelijkse overleg | Zijn de mijlpalen behaald. Zijn er moeilijkheden, verwacht de stagiair moeilijkheden, is de het project binnen scope. | mijlpalen |

**Tabel 1 Voortgangsbewaking**

## 5.2 Risicomanagement

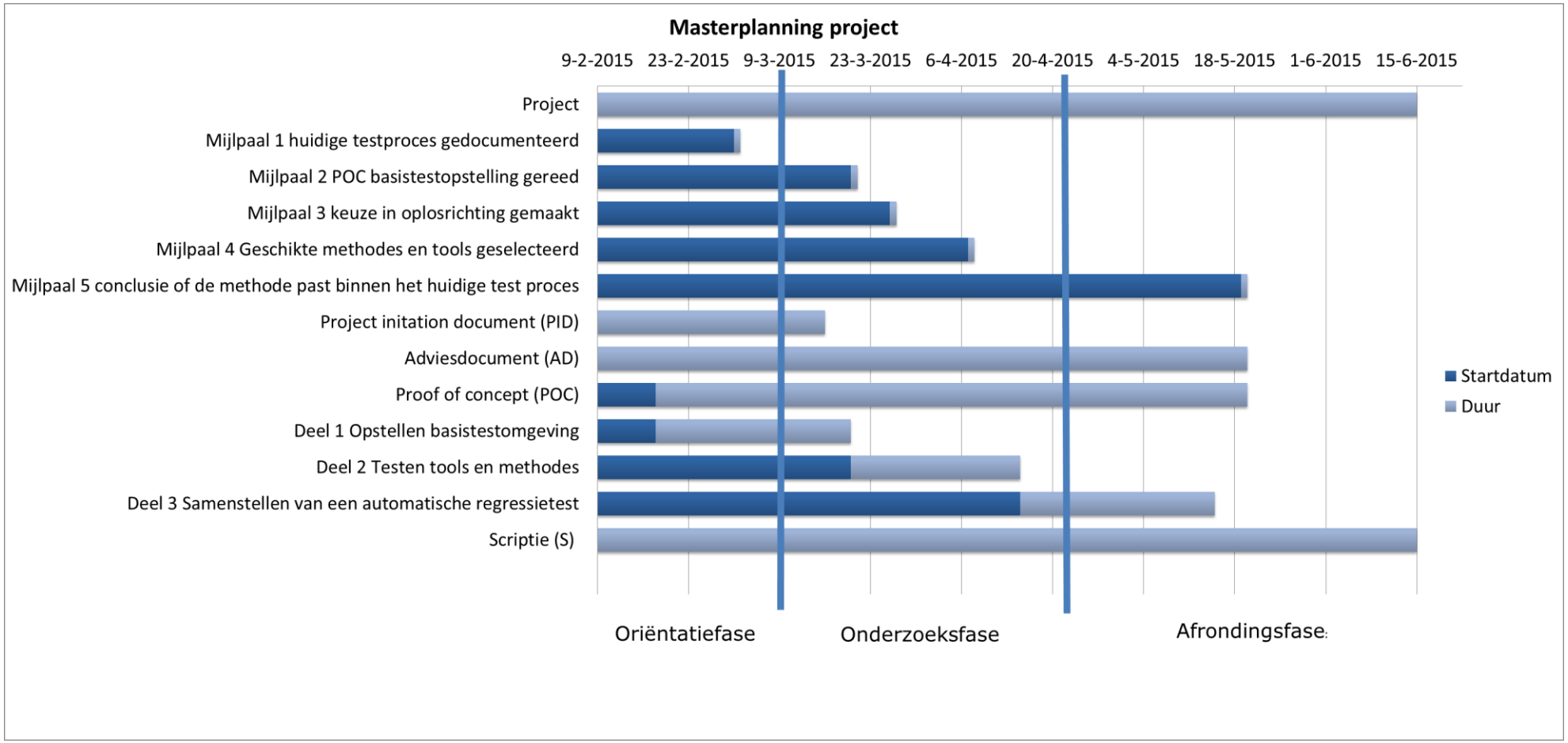
*De hoofdvraag is voortijdig beantwoord:*

Wanneer al in een vroeg stadium de hoofdvraag wordt beantwoord worden de hoofd en deelvragen van het project aangepast. De scope zal dan worden verbreed met de toevoeging van een of meerdere deelvragen. Deze deelvragen zullen antwoord geven op de vraag hoe het testproces verder verbeterd kan worden.

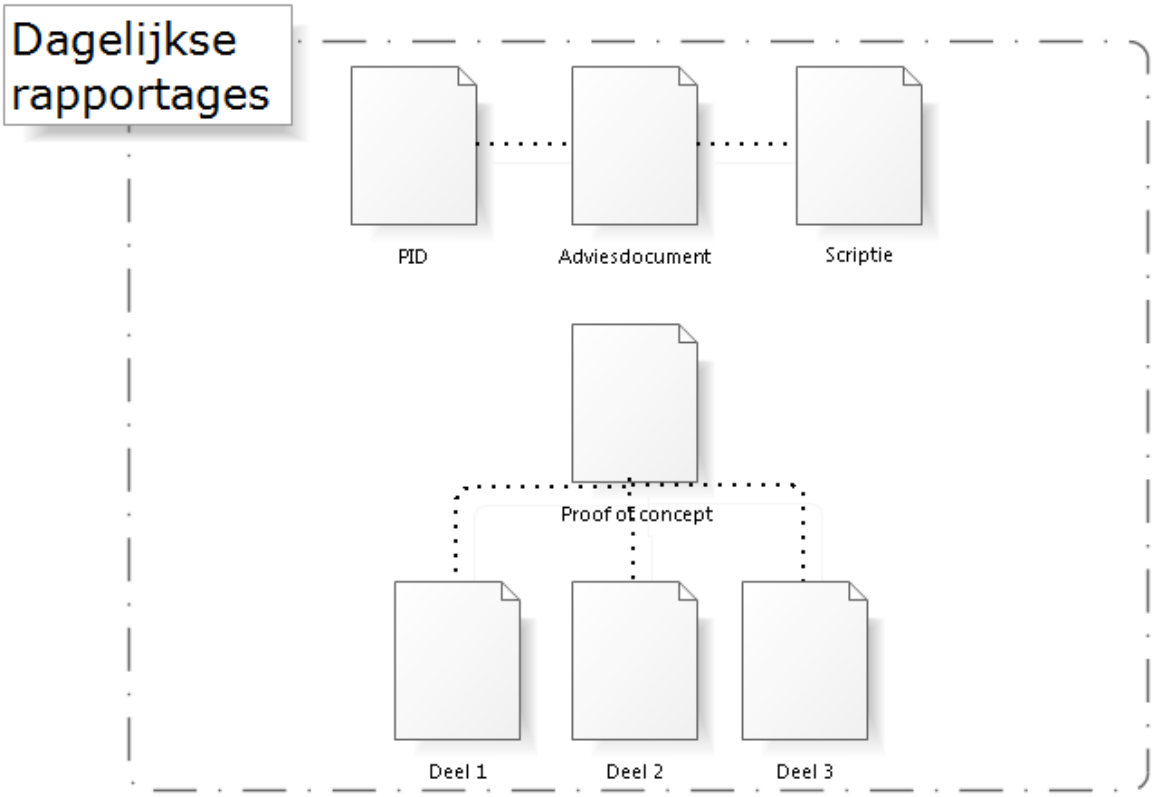
*De koppeling van tools is te complex om binnen het project af te ronden:*

Het deel drie van het proof of concept kan te gecompliceerd worden door een eventueel moeten programmeren van een application programming interface (API) of API's. Deze API zou geschreven moeten worden in het geval dat een bestaande tool gekoppeld dient te worden aan de door de stagiair gebouwde oplossing. Deze situatie is nice to have en zal als het een probleem wordt buiten de scope vallen.

# 6 Bijlage A: Projectplanning



# 7 Bijlage B: Productdecompositiestructuur



# 8 Bijlage C: Business case

De business case geeft inzicht in de kwalitatieve waarde van het project automatische testcase generatie. Met behulp van deze business case heeft de opdrachtgever inzicht in de investeringen en opbrengsten. De aanleiding van het project wordt gekwalificeerd met behulp van een SWOT-analyse.

## 8.1 Aanleiding

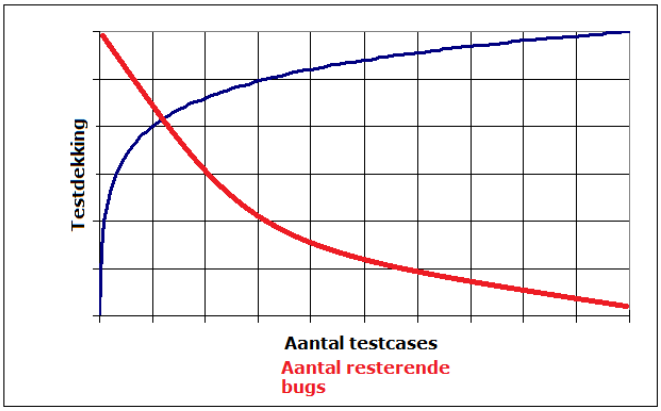
Door middel van een SWOT analyse zijn de belangrijkste punten van de aanleiding van het project weergegeven.

| SWOT analyse | |
|---|---|
| **Strong** | Goede kennis over de applicatie waardoor er veel inzicht is in het maken en inzetten van testcases. |
| **Weak** | iBurgerzaken wordt niet volledig getest omdat testers niet alle scenario's kunnen meenemen in de testcasesdoor gebrek aan tijd.<br><br>Testcases maken neemt veel tijd in beslag |
| **Opportunities** | Het automatiseren van de regressietest en systeemtestcases.<br><br>Uitrol naar andere business units binnen PinkRoccade Local Government. |
| **Threats** | Bugs blijven in de applicatie nadat deze in productie is gegaan. |

## 8.2 Kwaliteitstoename

De insteek van het project is het verhogen van de testdekking door middel van het automatiseren van testcases voor de regressie en de systeem test. Door het automatiseren van deze testcases kunnen in kortere tijd meerdere testcases worden gemaakt. Aangenomen is dat de toename van testcases bijdraagt aan de verhoging van de testdekking zie Figuur 2 Testdekking vs aantal testcases.

Aangenomen is dat het aantal bugs die na het testen in het systeem zit ongeveer omgekeerd evenredig is aan de testdekking.




Figuur 2 Testdekking vs aantal testcases

## 8.3 Conclusie

Als het project automatische testcase generatie doorgang krijgt heeft dit positieve invloed op de testdekking. Deze wordt sterk verhoogd door de toename van testcases. Hierdoor blijven minder bugs in het systeem en daardoor verbeterd te kwaliteit van de iBurgerzaken applicatie. Bovendien zou het project kunnen worden overgenomen door andere business units zodat zij kwaliteits verbetering ervaren.

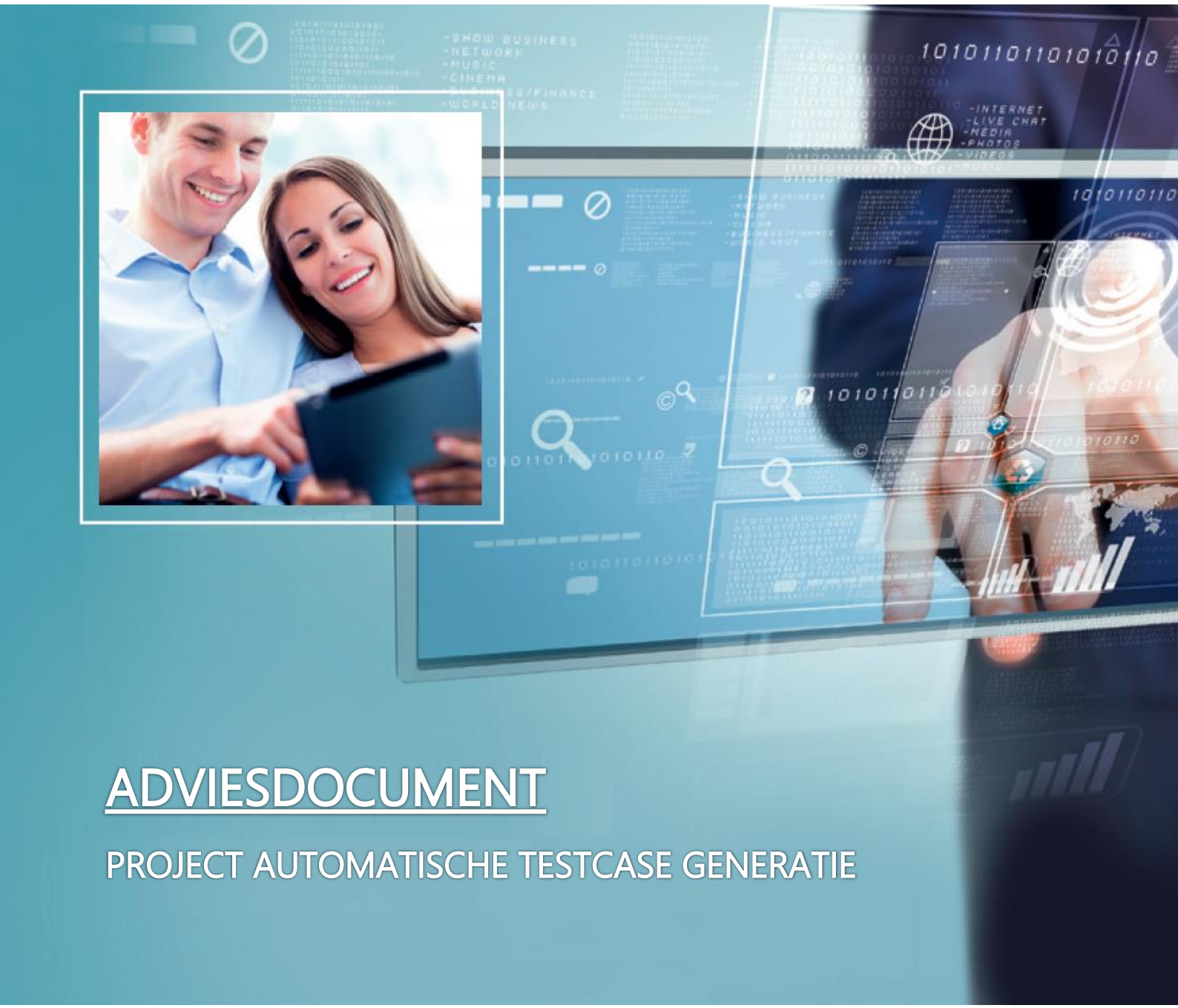
# 9 Bijlage D: Begrippenlijst

| Begrip | Afkorting | Uitleg |
|---|---|---|
| Agile | | Software ontwikkelmethode waarbij opleveren van kleine werkende sub producten (iteraties).De communicatie is persoonlijk, zo min mogelijk documenten. |
| Automated regression testing | ART | Het automatisch uitvoeren van testcases van de regressietest. |
| Black box testing | | Een software test techniek die de input en output test van een software applicatie maar daarbij de werking van binnenuit niet meeneemt. (bron: www.redstonesoftware.com) |
| Cucumber | | Applicatie dat gebruikt wordt voor de uitvoer van testcases. |
| Domain specific language. | DSL | Een software taal voor specifiek voor een domein. |
| Feature | | Een feature is een functie die door middel van Gherkin wordt omschreven. Een feature bestaat uit 1 of meerdere scenario's binnen Gherkin. |
| Gherkin | | DSL van Cucumber. Deze DSL is leesbaar voor zowel mens als machine. |
| Jenkins | | Tool dat bouwen en testen van software makkelijker maakt door het automatiseren van kleine taken. |
| Proof of concept | POC | Bewijs van de werking door een demonstratie. |
| Regressie testen | | Het testen van een bestaande applicatie met toegevoegde functionaliteiten. |
| Risico | | Kans x bedreiging |
| Scenario | | Een scenario geeft de interactie weer van de gebruiker en het systeem.<br>Maar een scenario is ook een stap binnen Gherkin wat een gedrag van het systeem weergeeft. |
| Scrum | | Een agile software ontwikkelingsmethode |
| Systeem test | | Bij deze test wordt de werking getest van de nieuwe componenten(clusters) in de totale omgeving. Hierbij worden alleen de nieuwe functionaliteiten meegenomen. |
| Testcase | | Een test waarmee de werking van een |

| | |
|---|---|
| | software applicatie of deel daarvan, wordt getoetst. |
| Testdekking | Het totale verwachte gedrag wat getest wordt met de testcases. Dit is geen absoluut getal, maar een benadering. Dit komt doordat het totale verwachte gedrag niet meetbaar is. |
| Unit test | Is een test van een klein deel van de code. En test de werking losstaand. |
| User story | Beschrijft in een paar zinnen wat de gebruiker moet/wil doen. |

# BIJLAGE B: Adviesdocument

pinkroccade
TSS LOCAL GOVERNMENT



# ADVIESDOCUMENT

## PROJECT AUTOMATISCHE TESTCASE GENERATIE

| | |
|---|---|
| DATUM | 4-7-2015 |
| PLAATS | ‘s Hertogenbosch |
| AUTEUR | Neijts, J |
| FUNCTIE | Stagiair |
| VERSIE | 1.2 |

pinkroccade TSS LOCAL GOVERNMENT

# INHOUDSOPGAVE

# 1 INTRO

## 1.1 Aanleiding

Het testen van de iBurgerzaken applicatie is een integraal onderdeel van het ontwikkelproces. De applicatie en de delen daarvan worden getest om te zorgen dat: functionaliteiten naar behoren werken, de applicatie werkt bij grootschalig gebruik, de applicatie werkt op verschillende platforms, alle bekende browsers en de applicatie aansluiten bij de behoeften van de eindklanten. Er zijn verschillende testvormen die de ontwikkelteams van iBurgerzaken gebruiken om de diverse aspecten van de applicatie te testen.

Er is zeer veel al bestaande functionaliteit, elk scherm bevat een grote hoeveelheid basisfunctionaliteit. Naast functionaliteiten bevat de IBurgerzaken applicatie ook veel kwaliteitsattributen. Het is onmogelijk om deze in een kort tijdsbestek te testen. Binnen de Scrum methode betekent dit een periode van ongeveer twee weken.

De huidige "behavior driven development" aanpak maakt dat de test scenario's zijn gebaseerd op het verwachte gedrag van de applicatie. Het creëren van deze test scenario's is arbeidsintensief. Hierdoor is er een zekerheid dat je als tester niet alle scenario's meeneemt maar grofweg 80%[1]. De tester voert naast de uitvoer van deze "testcases" ook "exploratory tests" uit om de testdekking te vergroten. Toch heeft het testteam van iBurgerzaken het beeld dat de testdekking niet optimaal is.

Door het automatisch genereren van "testcases" kunnen meer "testcases" in dezelfde tijd gerealiseerd worden. Het testteam van iBurgerzaken heeft er behoefte aan dat hier onderzoek naar wordt gedaan.

## 1.2 Business case

De business case geeft inzicht in de kwalitatieve waarde van het project automatische testcase generatie. Met behulp van deze business case heeft de opdrachtgever inzicht in de investeringen en opbrengsten. De aanleiding van het project wordt gekwalificeerd met behulp van een SWOT-analyse (tabel 1 SWOT analyse).

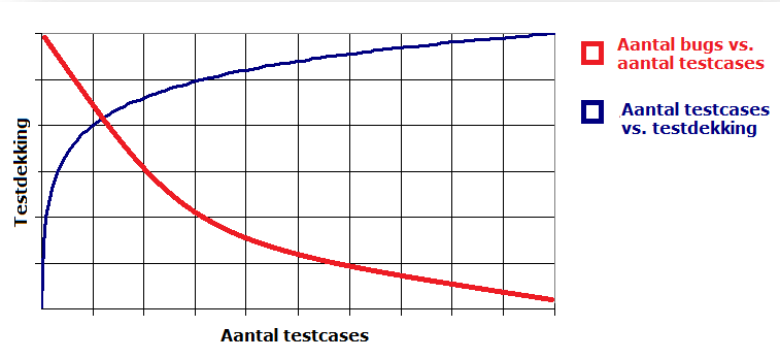
[1] Deze schatting is gedaan aan de hand van een interview met L Koning.

| SWOT analyse | |
|---|---|
| Strong | Goede kennis over de applicatie waardoor er veel inzicht is in het maken en inzetten van testscenario's. |
| Weak | iBurgerzaken wordt niet volledig getest omdat testers niet alle scenario's kunnen meenemen. Door de korte ontwikkeltijd bij de iteraties blijft het maken van scenario's een aandachtspunt. |
| Opportunities | Het automatiseren van de regressietest en systeemtests.<br>Uitrol naar andere business units binnen PinkRoccade Local Government. |
| Threats | Fouten blijven in de applicatie nadat deze in productie is gegaan. |

Tabel 1 SWOT business case

### Kwaliteitstoename

De insteek van het project is het verhogen van de testdekking door middel van het automatiseren van "testcases" voor de regressie en de systeem test. Door het automatiseren van deze "testcases" kunnen in kortere tijd meerdere "testcases" worden gemaakt. Aangenomen is dat de toename van "testcases" bijdraagt aan de verhoging van de testdekking zie Figuur 1 Testdekking versus aantal "testcases"[2].




Figuur 1 Testdekking versus aantal "testcases" vi

[2] Een hogere dekkingsgraad per dekkingsvorm resulteert in een hogere testdekking. Aangenomen is dat het aantal fouten die na het testen in het systeem zit ongeveer omgekeerd evenredig is aan de testdekking.

Als het project automatische testcase generatie doorgang krijgt heeft dit positieve invloed op de testdekking. Deze wordt sterk verhoogd door de toename van "testcases". Hierdoor blijven minder fouten in het systeem en daardoor verbeterd de kwaliteit van de iBurgerzaken applicatie. Bovendien zou het project kunnen worden overgenomen door andere business units zodat zij kwaliteitsverbetering ervaren.

## 1.3 Onderzoeksvragen

### Hoofdvraag:

*"In hoeverre zijn er mogelijkheden om de testdekking van het regressie en systeemtestproces te vergroten door een vorm van testcase generatie automatisering?"*

**Deelvragen:**

- Welke methodes en tools (intern[3] en extern) worden bij PinkRoccade local Government ingezet om een testscenario te genereren en uit te voeren in de huidige situatie?
- Welke tools(extern en intern) kunnen bijdragen aan de automatisering van de generatie van test scenario's?
- Welke methodes zijn beschreven in de literatuur die kunnen worden geraadpleegd om test scenario's deels of volledig automatisch te genereren?
- Wat zijn de mogelijke voor-en-of nadelen van gedeeltelijke of volledige geautomatiseerde generatie van test scenario's ten opzichte van de huidige situatie?

## 1.4 Opbouw document

Het adviesdocument bestaat uit de volgende hoofdstukken:

### Gerelateerd

Hier worden gerateerde onderzoeken behandeld.

### Deskresearch

Hier is de relevante theorie voor het onderzoek beschreven. Deze heeft betrekking op de huidige en/of op de gewenste situatie.

### Situatie

Hier is het huidige testproces, testcase creatie en het ART testproces beschreven. Tevens zijn hier de toekomst varianten beschreven.

### Onderzoek

Hier is het onderzoek naar de toekomst varianten beschreven. De opzet en de resultaten zijn per variant en per tool uitgeschreven.

[3] Interne tool: door PinkRoccade reeds in gebruik, externe tool: tool afkomstig uit de buitenwereld.

### Conclusies/Aanbevelingen

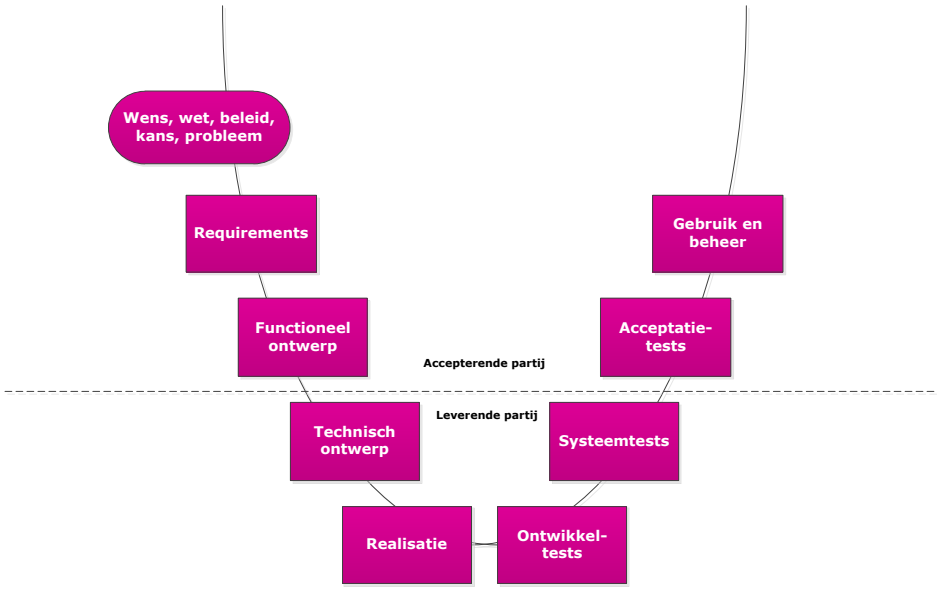
Aan de hand van de resultaten zijn conclusies getrokken en aanbevelingen gedaan, deze zijn hier uiteengezet.

# 2 DESKRESEARCH

In dit hoofdstuk worden de technieken, begrippen en werkwijzen die voorkomen in de huidige en de mogelijke toekomstige situaties uitgelegd.

## 2.1 Testsoort en testbasis

Software wordt getest aan de hand van gedefinieerde kwaliteitsattributen als: functionaliteit, continuïteit en performance. De referentie van de attributen komt uit de linkerkant van Figuur 2 V-model Tmap Next. De gehele linkerkant van wens tot technisch ontwerp vormt de testbasis. De testsoort bevat een aantal activiteiten om op deze attributen te testen. In Figuur 2 V-model Tmap Next zijn dit aan de rechterkant: acceptatietest, systeemtest en ontwikkeltest.




Figuur 2 V-model Tmap Next[i]

*Testsoorten:*

### Ontwikkeltest

Tijdens de ontwikkeltest worden kleine stukken van de applicatie getest, dit zijn classes of regels code. Dit is een unittest. Als deze unittests zijn uitgevoerd dan wordt de unitintegratietest uitgevoerd.

### Systeemtest

Bij deze test worden de niet functionele en functionele requirements van het systeem of delen daarvan getest. De systeemtest wordt uitgevoerd na de ontwikkeltests.

### Acceptatietest

Deze wordt aan de klantzijde uitgevoerd. Hier wordt getest of de applicatie is conform de klant eisen en wensen.

### Systeem en acceptatietest bij Scrum

Door het toepassen van Scrum worden de acceptatie en systeemtest min of meer gecombineerd. Deze lopen in elkaar over.

## 2.2 Testvormen

In grote lijnen zijn er twee vormen van testen: "black box testing" en "white box testing". De definitie van "white" en "black box testing" volgens (Association, 1990):

*"black box testing" wordt ook wel functioneel testen genoemd, en negeert de interne mechanismes en werking van het systeem en componenten en richt zich volledig op de uitput die gegenereerd wordt als antwoord op de input en de context.*

"white box testing" wordt ook wel structureel testen genoemd en "glass box testing". "white box testing" houdt rekening met de interne werking van het systeem of component.

"white box testing" kan worden gebruikt voor de volgende test vormen[ii]

- Unittest
- Integratietest
- Regressietest

"white box testing" brengt de volgende voor-en nadelen met zich mee[iii]:

### Voordelen

- De inwendige code van de applicatie wordt getest, dit is een voordeel wanneer de GUI nog niet volledig af is, of als deze vaak verandert.
- De "testcases" kunnen vaker herhaald worden als de objectnamen niet of nauwelijks wijzigen.
- De testdekking is hoger omdat men op meerdere niveaus test.

### Nadelen

- Een hogere mate van kennis over de applicatie is nodig om deze te kunnen testen. De achterliggende code kan vaak erg complex zijn, dit maakt dat de tester tevens een hoge skillset nodig heeft.
- Net als bij "black box testing" kan de testcase bij veranderingen van objecten falen.

- De tools om mee te testen moeten geïntegreerd zijn met de applicatie, dit kan bij de uitvoer van de applicatie zorgen voor complicaties.

Black box testen of wel "behavioral testing" genoemd richt zich op de werking van de applicatie

zoals die beschreven is in de functionele requirements. "black box testing" zoekt de fouten in het externe gedrag van de applicatie in de volgende categorieën [iv]:

- Incorrecte of ontbrekende functionaliteit
- Interface fouten
- Fouten in datastructuren
- Fouten in gedrag of prestaties
- Fouten bij openen en sluiten

"black box testing" kan worden gebruikt voor de volgende testen:

- Integratietesten
- Functioneel testen
- Systeemtesten
- Acceptatietesten
- Betatesten
- Regressietesten

"black box testing" heeft enkele voor- en nadelen [iii] :

### Voordelen

- De "testcases" worden gemiddeld sneller gecreëerd doordat de tester de inwendige werking niet hoeft mee te nemen in de testcase.
- Relatief makkelijke methode voor de tester, deze houdt zich alleen bezig met de GUI van de applicatie niet met de inwendige werking. De tester houdt zich alleen bezig met de input en output van de applicatie

### Nadelen

- Men test niet de volledige werking van de applicatie
- Bij grote en frequente veranderingen van de GUI (of een laag als HTTP) wordt het onderhouden van de "testcases" moeilijker.
- "testcases" werken niet altijd, dit komt doordat de GUI vaak kan veranderen wanneer men test op een andere omgeving.

In de huidige regressie en systeemtest procedure wordt de iBurgerzaken applicatie getest met black box test methodieken. De afzonderlijke componenten worden getest aan de hand van "white box testing". Voor dit project is gekozen voor Black-box oplossingen omdat:

- De problematiek ligt bij het automatische testproces en dat gebeurt door middel van "black box testing".

- De testcase creatie over het algemeen sneller verloopt.
- De kennis van het testteam overwegend ligt bij de huidige manier van automatisch testen, daardoor zal de kennis overdracht naar alle waarschijnlijkheid moeizamer verlopen.
- De stagiair naar waarschijnlijkheid te weinig software kennis heeft om white-box test tools goed te kunnen beoordelen. Bovendien heeft J.D van der Velde aangegeven dat met zijn
- Testcase creatie bij regressietesten verloopt trager.

### 2.2.1 Regressietest [v]

"Een regressietest is erop gericht om te controleren dat alle ongewijzigde onderdelen van een systeem nog correct functioneren na het doorvoeren van een wijziging." [v]

Bij het samenvoegen van nieuwe "content" en bestaande "content" kan het voorkomen dat de applicatie fouten gaat vertonen.

Het testen van de regressie kan met behulp van diverse dekkingsvormen. Regressietesten worden door middel van de ART uitgevoerd.

### 2.2.2 "Exploratory test"

Het uitvoeren van tests die niet vooraf zijn bedacht. De tester bedenkt tijdens het uitvoeren van de "exploratory test" de stappen aan de hand van de informatie die vrijkomt bij de uitvoering hiervan. De tester gaat af op zijn ervaring en zijn bevindingen.

### 2.2.3 Dekkingsvormen

Een dekkingsvorm wordt afgeleid aan de hand van een gekozen testbasis en test vorm. Er zijn verschillende dekkingsvormen. De belangrijkste dekkingsvormen voor dit project worden in deze paragraaf uitgelegd. De overige dekkingsvormen zijn opgenomen in BIJLAGE A: .

#### "Capture and replay"

Opnemen van gebruikers gedrag voor referentie en input. Hierbij worden de input gegevens in de GUI opgeslagen. Deze opgenomen handelingen en input worden later afgespeeld bij bijvoorbeeld een regressietest. Reden van gebruik in het onderzoek is omdat deze dekkingsvorm eenvoudig is in gebruik en een testcase relatief snel gemaakt kan worden.

#### Orthogonale arrays/ "pairwise" testing

Dekkingsvorm door middel van orthogonale arrays paren van parameters, hierdoor reduceert het aantal combinaties. Door het vormen van paren is er een grote kans van het vinden van fouten.

Met behulp van de "combinatorial optimization" techniek "pairwise" testen worden het aantal combinaties verkleind door het toepassen van testen in paren( paren van 2, of 3 of meer). Voorbeeld: Bij het aanvragen van een reisdocument in de iBurgerzaken applicatie worden de gegevens van een adellijk persoon opgenomen. Deze titels kunnen worden gebruikt bij de aanhef van het reisdocument. De tester heeft ingeschat dat hij te veel tijd nodig heeft om alle vormen te testen. Daarom besluit hij door het maken van paren in de testdata het aantal "testcases" te verkleinen.

Voorbeeld:

| Geslacht | M | V | O | | | |
|---|---|---|---|---|---|---|
| Adellijke titel | Baron | Graaf | hertog | Jonkheer | Markies prins | Ridder |
| Aanschrijven met titel | Ja | nee | | | | |

In het voorbeeld zijn 3*7*2= 42 combinaties mogelijk. Als je "pairwise" combineren toepast verlaag je dit naar 21 combinaties. In kleur zijn een aantal van de mogelijke paren weergegeven.

| | | | |
|---|---|---|---|
| 1 | M | Baron | Ja |
| 2 | M | Graaf | Nee |
| 3 | V | Baron | Nee |
| 4 | V | Graaf | Ja |
| 5 | M | Hertog | Ja |
| 6 | M | Jonkheer | Ja |
| 7 | M | Markies | Ja |
| 8 | M | Prins | Ja |
| 9 | M | Ridder | Ja |
| 10 | O | Baron | Ja |
| 11 | O | Graaf | Nee |
| 12 | V | Hertog | Nee |
| 13 | V | Jonkheer | Nee |
| 14 | V | Markies | Nee |
| 15 | V | Prins | Nee |
| 16 | V | Ridder | Nee |
| 17 | O | Hertog | Nee |
| 18 | O | Jonkheer | Ja |
| 19 | O | Markies | Nee |
| 20 | O | Prins | Ja |
| 21 | O | Ridder | Nee |

Figuur 3 "pairwise" voorbeeld [vi]

Rede van gebruik in het onderzoek:

Er zijn veel situaties waar de tester de inschatting heeft gemaakt dat het testen van het gewenste gedrag (te) veel "testcases" oplevert. Bovendien heeft de tester niet het overzicht wat de relatie is van de dekkingsgraad met het aantal geteste "testcases".[4]

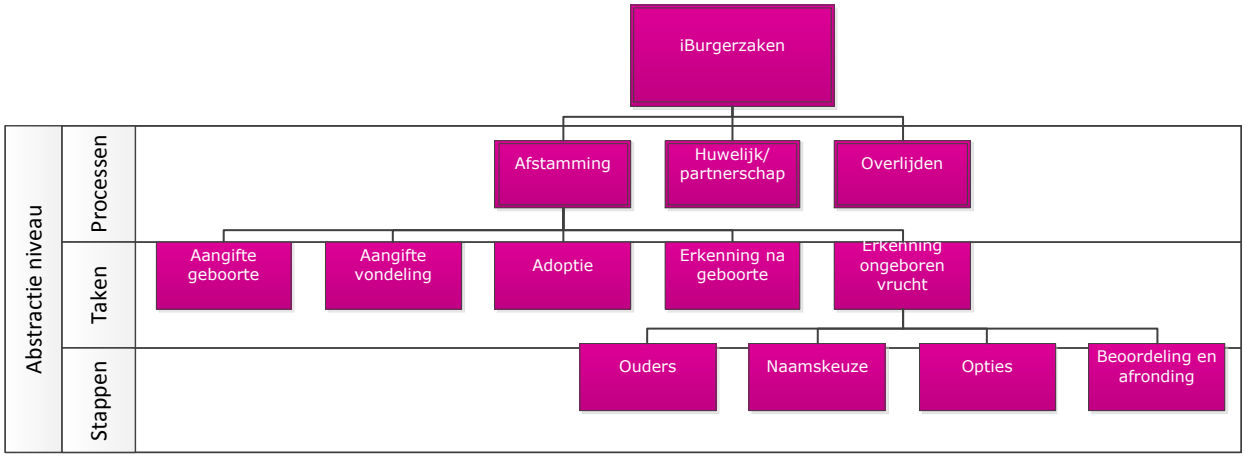
Een manual is geschreven om meer uitleg te geven over het gebruik van "pairwise", deze is opgenomen in BIJLAGE D: .

"model based testing:"

Hierbij wordt een model gebruikt om het gewenste gedrag van de applicatie te testen. De applicatie heeft een interne status die wijzigt als het systeem input accepteert en output genereert. Het model beschrijft de mogelijke in en output volgordes.

[4] Deze redenen zijn naar voren gekomen uit een interview met Dhr. L. Koning, maart 2015.

Dit kan op verschillende niveaus, zie Figuur 4 Abstractie niveau iBurgerzaken.




Figuur 4 Abstractie niveau iBurgerzaken vi De iBurgerzaken heeft een hoofdmenu, daaruit kunnen diverse richtingen worden gekozen als: afstamming, huwelijk/partnerschap, overlijden etc. Bij afstamming kun je weer kiezen uit Aangifte geboorte, aangifte vondeling, adoptie, erkenning na geboorte en erkenning ongeboren vrucht. Als je kiest voor Erkenning ongeboren vrucht zijn er 4 stappen die gevolgd worden: Ouders, Naamkeuze, Opties, Beoordeling en Afronding.

Een testselectie algoritme maakt van het model "testcases". Hierbij wordt uit veel mogelijke inputwaarden een selectie gemaakt. Er zijn drie richtingen in "model based testing:"[vii] om de situatie van de geteste applicatie te beschrijven. Zo heb je de axiomatic aproach[viii], labeled transition systems en finite statemachine (FSM). PATG heeft gekozen voor FSM omdat iBurgerzaken een eindigende hoeveelheid staten heeft. In een "Finite state machine" worden de mogelijke staten van een systeem beschreven, en van iedere staat de mogelijke transities naar een volgende staat (mogelijk zichzelf). In iBurgerzaken kan een staat gezien worden als een pagina, en een transitie als de pagina-overgang na het drukken op een knop.

### Monkey testing

Een Monkey test is een automatische "model based"test tool die weinig energie van de tester vergt. Deze test wordt lang gerund, vaak tot fouten uiteindelijk zijn gevonden. De term Monkey testen komt van het "infinite Monkey theorem".[ix] De theorie houdt in dat als je maar genoeg Monkeys en typemachines hebt, de Monkeys uiteindelijk het complete werk van Shakespeare typen. Dit komt neer op het maken van genoeg random combinaties totdat je uiteindelijk de fouten in het systeem vindt. De Monkey test kan bij verschillende soorten testen uit worden gevoerd. De Monkey testen zijn verschillend in intelligentie zijn er drie vormen van lage intelligentie tot hoge:

### Dumb Monkey:

De Dumb Monkey variant is de variant met de laagste intelligentie[x]. De Dumb Monkey voert input in zonder rekening te houden met al vorige input waardes. Deze vult vooral random waardes in. De Dumb Monkey is vooral geschikt voor "exploratory testing".

### Smart Monkey:

Bij deze test weet de Monkey welke individuele waarde gebruikt moeten worden in de test en hoe de relatie is tussen deze waardes. De smart Monkey creëert test scenario's voor deze input waardes gaat invullen [x]. Deze vorm biedt meer dekkingsvormen dan de Dumb Monkey variant, hierdoor wordt deze effectiever in het vinden van fouten.

### Brilliant Monkey

De Brilliant Monkey simuleert het gebruik nauwkeurig. Hierbij genereerd deze test scenario's waarbij de verschillende staten meegenomen worden in het pad wat deze genereerd. De Brilliant Monkey kan voor meerdere testen ingezet worden, bijvoorbeeld regressietesten.

## 2.3 Testdekking

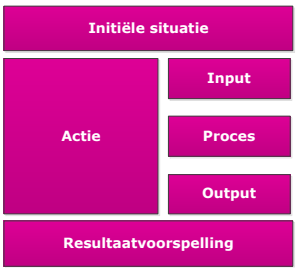
Tmap Next maakt het onderscheid tussen testvormen, testdekking en dekkingsgraad (Koomen, Aalst van der, Broekman, & Vroon, Tmap Next voor resultaatgericht testen, 2006, pp. 600-602). Testdekking is een relatief begrip omdat je niet kunt spreken over de testdekking van de totale applicatie, je weet namelijk niet welke combinaties en testen er totaal kunnen zijn. Daarom bepaal je de testdekking uit de dekkingsvormen en de bijbehorende dekkingsgraad. Een dekkingsvorm geeft aan welke mogelijkheden worden getest. In het testvoorbeeld is test vorm x "pairwise" testen zijn test vorm y CRUD, beide zijn testvormen (grote roze vierkanten).

De dekkingsgraad (kleine rode vierkanten) zegt iets over het percentage geteste mogelijkheden van een dekkingsvorm. De dekkingsgraad van het totaal aantal vormen is de dekking. De testdekking in het (fictieve) voorbeeld is dus (1/4+1/4 )/2 = ¼ dus 25%. Wat betekend dat 25% van de "content" wordt afgedekt met een testvorm.


Figuur 5 Voorbeeld testdekking vi

## 2.4 Testgevallen en testscripts

Met een testgeval test men het gewenste gedrag van het systeem aldus Tmap Next[xi]. Een testgeval bestaat uit een initiële situatie, acties en een resultaatvoorspelling. De initiële situatie schetst de uitgangspositie van het systeem en beschrijft de input van het testgeval. De acties beschrijven de processen die gedaan moeten worden. De resultaatvoorspelling geeft aan welke output verwacht wordt.




Figuur 6 Generieke opbouw testgeval [xii]

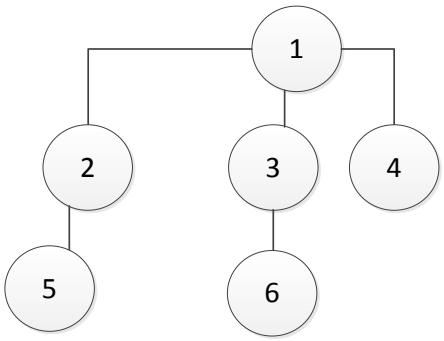
Een testscript is een verzameling van testgevallen. Deze testgevallen worden bij elkaar gevoegd door overeenkomsten als: initiële situatie, doelsituatie, volgorde.

## 2.5 Zoekalgoritmes

Bij het testen met een Monkey applicatie dient een volgorde aangehouden te worden in het vinden en doorlopen van staten. Belangrijke algoritmes die ingezet kunnen worden hiervoor zijn “breadth first” en “depth first”.

### Breadth-first search

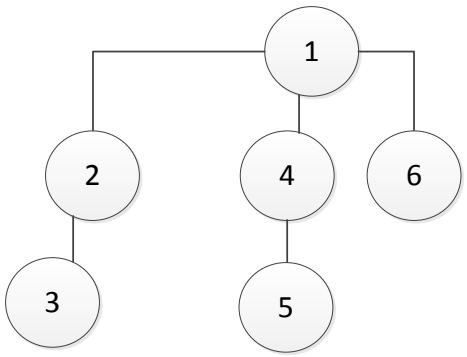
Dit zoekalgoritme volgt een piramidevorm bij het zoeken. Het algoritme begint op het hoogste niveau en zakt pas verder als alle punten zijn geraakt op 1 niveau zie figuur 7. De Dumb Monkey applicatie gebruikt “breadth first” om de applicatie te vinden binnen iBurgerzaken om getest te worden. Hierbij gaat de Monkey alleen op het hoogste niveau door de pagina’s tot hij op de beoogde pagina is. Op dat moment gebruikt deze “depth first”.




Figuur 7 “breadth first” vi

### Depth-first

Bij "depth first" gaat het algoritme eerst de diepte in alvorens de breedte. Als geïllustreerd in figuur 8.




Figuur 8 "depth first"

# 3 SITUATIE

Dit hoofdstuk beschrijft de huidige situatie en de gewenste situatie. De huidige situatie beschrijft het algemene testproces, de testcase creatie en het automatische testen zoals dit nu gebeurt binnen PinkRoccade Local Government. In de gewenste situatie worden de varianten besproken die bij kunnen dragen aan een gewenste situatie.

## 3.1 Huidige situatie

Binnen de business unit Publiekszaken van PinkRoccade Local Government wordt ontwikkelt aan de iBurgerzaken applicatie. De ontwikkelteams werken met de agile methode Scrum.

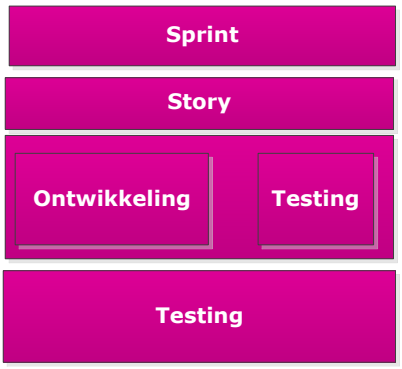
Het opleveren van kleine werkende stukjes software in korte periodes kenmerkt de agile methode. Bij scrum heten deze periodes sprints, de "content" wordt bepaald door user story's. Dit zijn abstracte kleine verhaaltjes die vertellen wat de user met het systeem wil bereiken.

Een "product owner" bepaalt welke user story's iedere sprint in de sprint "backlog" komen te staan. Deze story's komen uit een product "backlog", waarin de user story's zijn geprioriteerd. Deze user story's zijn functionaliteiten, waarin aangegeven wordt wat de gebruiker kan doen met het systeem.

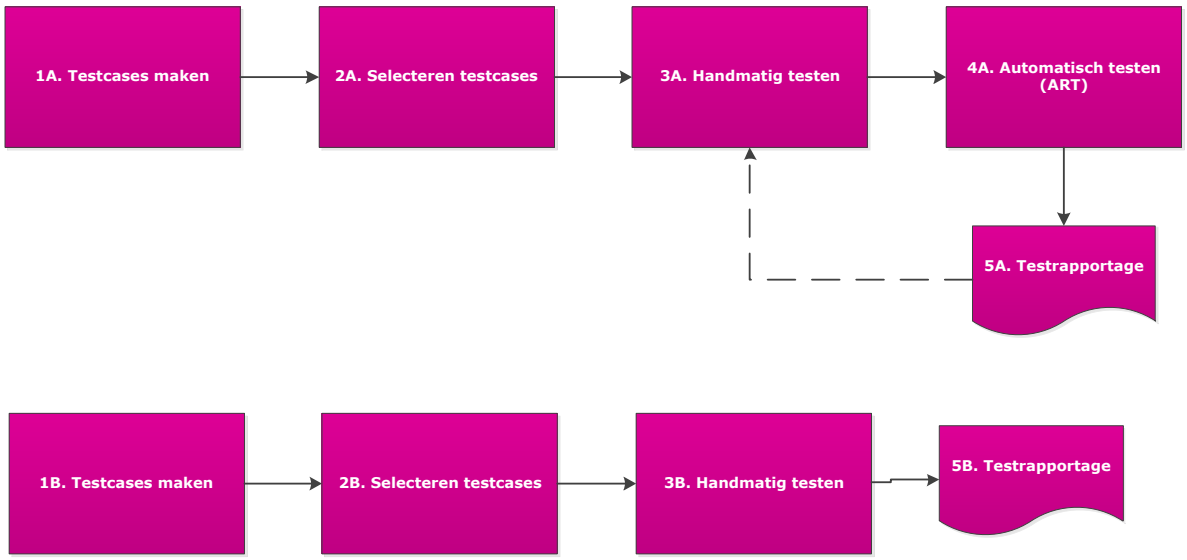
Figuur 9 geeft een sprint binnen het GAAS (Gemeente as a service) project weer. Iedere user story wordt tijdens de ontwikkeling "reviewed" en getest. De ontwikkelaar of tester maakt een testcase die getest moet worden.

Tijdens de ontwikkeling wordt de ontwikkeltest (unittesten) uitgevoerd, daar wordt de story getest op zijn werking op kleinschalig niveau (classes en regels code). De werking van de software wordt getest op input en output maar ook impliciet de verwachting van de gebruiker.

Aan het einde van een aantal sprints voor oplevering naar de preproductieomgeving vindt een regressietest plaats. Hier wordt getest hoe de "stories" zich binnen de totale applicatie (iBurgerzaken) gedragen.




Figuur 9 Testmomenten vi




Figuur 10 testproces vi

Figuur 10 testproces illustreert het automatische en handmatige testproces van de iBurgerzaken applicatie.

Stap 1A is het maken van een testcase aan de hand van een story. Dit wordt beschreven in Testcase creatie3.1.1. Bij stap 2A worden de juiste "testcases" geselecteert om de user story te testen. De selectie hangt af van hoe de story is geprioriteerd en wat de risico's zijn. In het algemeen wordt er zoveel meegenomen als mogelijk is. Stap 3A, de tester gaat nu zijn testcase handmatig uitvoeren om te controleren of deze werkt, daarbij neemt hij het gedrag waar van de iBurgerzaken applicatie. Stap 4A hier worden de tests automatisch afgespeeld, zie 0 voor een gedetailleerde beschrijving. Stap 5A is het rapporteren van de testresultaten. Dit gebeurt grotendeels handmatig momenteel, maar er zijn ontwikkelingen gaande waarin dit meer wordt geautomatiseerd.

Stappen 1B t/m 3B zijn gelijk aan de stappen 1Atm 3A echter hier houdt het handmatige testen op er volgt een rapportage, deze is niet automatische gegenereerd als bij 5A, maar wordt handmatig ingevoerd in het rapportage systeem.

### 3.1.1 Testcase creatie

"testcases" worden gebruikt voor semi automatische tests en voor ART tests. Een testcase wordt gebaseerd op de user story's functionele requirements. De "testcases" worden gegroepeerd op basis van componenten en proces flow. Elke testcase wordt minimaal twee keer uitgevoerd, positief en negatief. Een testcase bestaat uit features en scenario's. Deze komen uit het Gherkin(domain specific language) domein. Gherkin is een DSL die leesbaar is voor mens en voor machine. Deze beschrijft het gewenste gedrag van de applicatie zonder in te gaan op de details hoe dit moet gebeuren. De Gherkin grammatica van de taal bestaat uit 27 talen (Engels, Duits etc.).

Een feature file is opgebouwd uit step die zijn gedefinieerd met "Given", "When", "Then", And en But. "Given" is de gegeven status van het systeem, "When" beschrijft een actie, "Then" beschrijft het resultaat van de actie. And en But zijn aanvullingen op de "When", "Then" en "Given"s.

Een voorbeeld van een testcase:

Feature: registration of a birth

As a user,

I want to register a birth

So I want to test its functionality.

Scenario: Register a birth positive flow

"Given" I open Birth application

And I start Birth process

"Then" I should see Ouders page

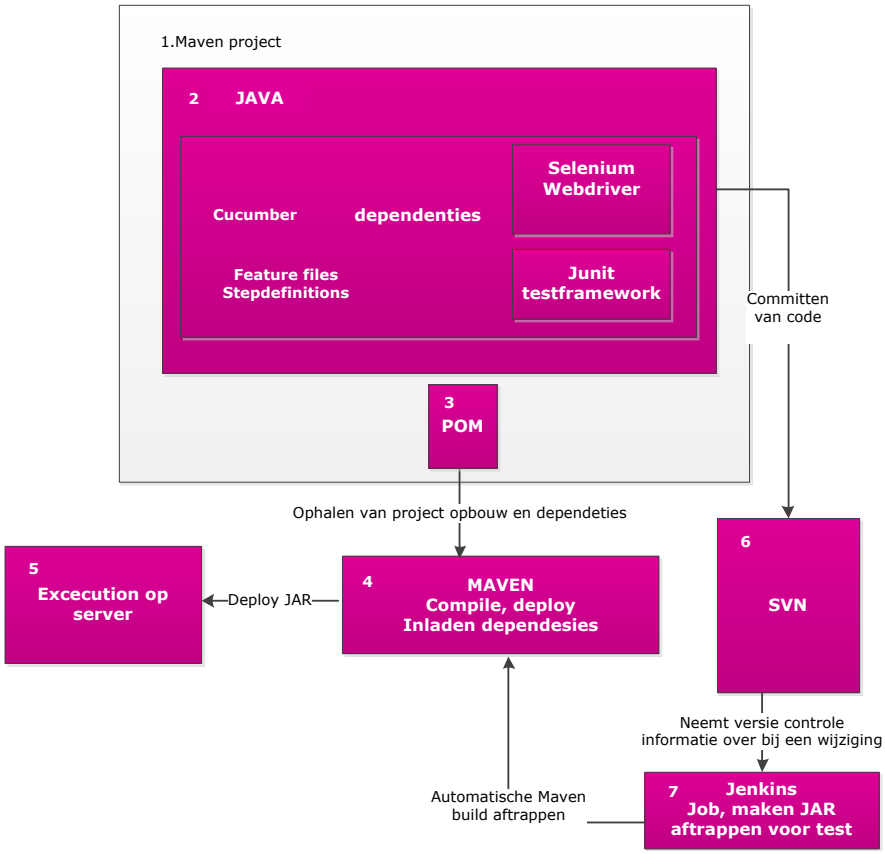
Een scenario hoeft bij het testen van verschillende input niet opnieuw gekopieerd en geplakt te worden. Hiervoor worden scenario Outlines gebruikt. Het scenario wordt gedocumenteerd om later afgespeeld te worden met behulp van een tag. Deze zijn van hoog naar laag in hiërarchie deze zijn: module tag, app tag, scenario functional discription tag en component tag.

Bijvoorbeeld bij een business rule in Verhuizing:

@RegressionTest @Migratie @CivilianVerhuizing @BusinessRules @Calender

### 3.1.2 Automatisch testen

In Figuur 11 ART schematisch is het automatische testsysteem schematisch weergegeven.




Figuur 11 ART schematisch vi

1. De automatische testcase is een onderdeel van een Maven project. Maven is een tool waarmee JAVA projecten worden gemanaged.
2. Het Cucumber project is opgebouwd uit diverse dependenties:
   De Cucumber library zorgt dat de Gherkin feature files en stepdefinities worden gekoppeld aan de onderliggende JAVA functies.
   Junit test framework dat o.a. de testvolgorde bepaald.
   Selenium webdriver is een library die browser interactie automatiseert.
3. POM is een bestand wat het project definieert, hierdoor kent Maven de opbouw en dependenties.
4. Maven “compiled” het project met de dependenties en maakt hier een JAR file van.
5. JAR file is een executabel van het project wat geëxecuteerd wordt op de server.
6. SVN is een code “repository” waarmee het project gedeeld door “developers”.
7. Jenkins is een Tool dat bouwen en testen van software makkelijker maakt door het automatiseren van kleine taken. Jenkins haalt zijn informatie uit de SVN dat een code is gewijzigd en initieert automatisch een Maven “build” af. Handmatig kunnen ook “deploys” worden afgeroepen in Jenkins. De JAR die Maven genereerd wordt door Jenkins geëxecuteerd op de juiste plek.

## 3.2 Gewenste situatie

In paragraaf 3.2.1 tot en met 3.2.6 worden de varianten (oplosrichtingen) beschreven. Deze oplosrichtingen zijn beschreven in de volgende structuur: uitgangspositie, verbetering, realisatie en onderzoek. De uitgangspositie geeft weer wat de aanleiding in de huidige situatie is, waarop de oplosrichting van toepassing is. De verbetering komt overeen met de verwachte berekening. In de realisatie wordt beschreven hoe deze oplossing wordt gerealiseerd. Het onderzoek bestaat uit vragen die beantwoord worden door verder theorie onderzoek en door middel van het *"proof of concept"*.

Uit deskresearch is gebleken dat er momenteel geen realiseerbare manieren zijn om scenario's volledig te automatiseren. Daarom is onderzoek gedaan naar de stappen die gedaan worden in de huidige test frame met als doel deze te verbeteren. Naar aanleiding van de resultaten zijn vijf varianten omschreven.

### 3.2.1 Variant 1 Semi automatische testcase generatie door "code completion library"

Een "code completion library"(CCL) is een plug-in die kennis heeft van de bestaande code en bij het invoeren van nieuwe code referenties maakt naar deze bestaande delen. Bij deze referentie heeft de gebruiker de keuze om de suggesties tijdens het maken van nieuwe code over te nemen.

#### Uitgangspositie

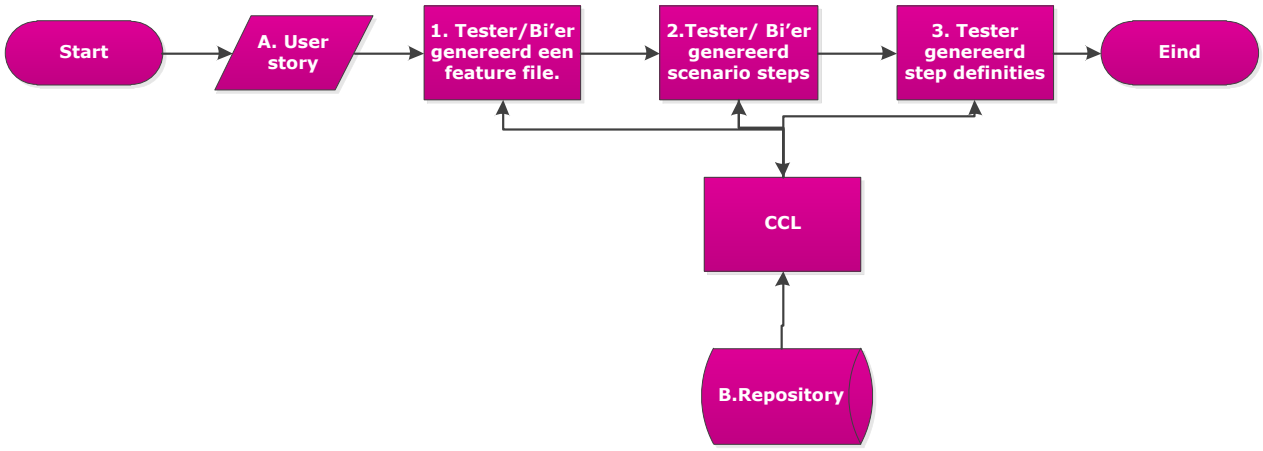
De tester of de business analist wil aan de hand van een user story een testcase opbouwen in de huidige situatie. Hiervoor moet hij nagaan welke stappen worden gedaan binnen de user story om deze te kunnen testen. Nu moet hij controleren of deze stappen al bestaan. Dit controleren wordt niet altijd gedaan. Als hij dit wel doet dan checkt hij of deze bestaan. Als dit het geval is dan zal hij deze samenvoegen of aanpassen. Als deze niet bestaan dan zal hij deze moeten creëren.

#### Verbetering

In Figuur 12 wordt het verbeterde proces van het creëren van "testcases" weergegeven. Deze bestaat uit de volgende stappen:

1. De Tester/ BE'er (Business engineer) bouwt een Gherkin feature file op aan de hand van een user story (Figuur 7 A). Een "code completion library" (CCL) verifieert of de naamgeving of *keywords* binnen een step van deze feature al voorkomt in de "repository" (B). Als deze voorkomt of overeenkomt meldt de CCL aan de tester/BE'er. Hier voorkomt de CCL dat features dubbel worden gecreëerd. Als de feature file klaar is dan wordt deze in de "repository" opgeslagen.
2. De tester/BE'er vult de feature in met scenario's en scenario steps. Wederom controleert de CCL of scenario's of steps voorkomen in de "repository". Als de tester/BE'er zijn naamgeving overeenkomt met opgeslagen scenario steps dan meldt de CCL dit.
   Hiermee wordt de "reusability" vergroot. Als het scenario gecreëerd is wordt deze in de "repository" opgeslagen.
3. De tester genereerd met de CCL stepdefinities in JAVA. Wederom meldt de CCL welke bestaande step definities al voorkomen in de "repository" en meldt dit als dit het geval is.

Door het gebruik van een CCL worden scenario steps en stepdefinities consistent hergebruikt. Door CCL wordt voorkomen dat "testcases" of elementen daarvan dubbel worden gecreëerd. Door CCL is een testcase eenvoudiger en sneller samen te stellen.




Figuur 12 variant 1 verbetering vi

### Realisatie

De gewenste situatie wordt bereikt door het schrijven of vinden van een CCL. De CCL wordt binnen het ontwikkelprogramma Eclipse geïntegreerd of naast Eclipse gebruikt. De CCL wordt gekoppeld aan de testcase "repository". De code die de tester heeft opgesteld, wordt herkend door de CCL. Als de tester nieuwe test scenario's wil creëren dan signaleert hij bij het typen van de eerste woorden van het scenario automatisch de CCL, waarop deze suggesties weergeeft van de te creëren scenario's. Als de scenario's zijn afgerond kan de tester door de CCL automatisch "step definitions" laten genereren. Bovendien kan de CCL overeenkomende stepdifinities herkennen door de steps te matchen aan bestaande steps in de "repository".

### Onderzoek

- In welke mate zou CCL dubbele aanmaak van "testcases" of steps kunnen voorkomen?
- In welke mate zou testcase creatie eenvoudiger met CCL?
- In welke mate zou een testcase sneller samengesteld worden met CCL?

## 3.2.2 Variant 2 Semi automatische testcase generatie door capture en replay

### Uitgangspositie

De tester/BE'er wil een user story testen. Hiervoor wil hij een testcase opstellen. "capture and replay" (CAR) houdt in dat de tester/developer de stappen in de applicatie opneemt die betrekking hebben op het gedeelte van de applicatie die veranderd is naar aanleiding van de toegevoegde "content". Later speelt hij deze stappen af, iedere stap wordt geverifieerd door de "capture and replay" applicatie.

### Verbetering

In Figuur 13 wordt variant 2 weergegeven. Deze bestaat uit de volgende stappen:

1. De tester/BE'er maakt aan de hand van (A) user story een pad waarmee hij het systeem doorloopt. Als hij dit pad uitvoert in het systeem dan neemt de CAR tool op wat de tester invoert en welke handelingen hij binnen het systeem uitvoert. CAR slaat deze testcase op in de "repository" (B).
2. De tester/BE'er muteert de testcase wanneer nodig. Dit is bijvoorbeeld het geval als hij niet de hele testcase uit wil voeren, of hij wil sommige attributen niet meenemen in de test. Stap 2 kan de eerste stap zijn als de tester een bestaande testcase wil muteren.

Het maken van een testcase voor de eerste keer kost minder tijd ten opzichte van de huidige manier van het opstellen van een testcase. Er is weinig vaardigheid nodig om deze testcase uit te maken, dit maakt deze tool in theorie gebruiksvriendelijk.


Figuur 13 variant 2 vi

### Realisatie

Door het inzetten van een capture en replay tool kan een testcase worden gemaakt. Deze tool kan een integratie zijn binnen de huidige setup of een apart systeem zijn.

### Onderzoek

- In welke mate kun je "testcases" aanpassen die met CAR zijn gemaakt?
- In welke mate kun je "testcases" hergebruiken.
- In welke mate kost het maken van "testcases" in CAR minder tijd dan de huidige manier van testen?

## 3.2.3 Variant 3 "pairwise" testgeval generatie

### Uitgangspositie

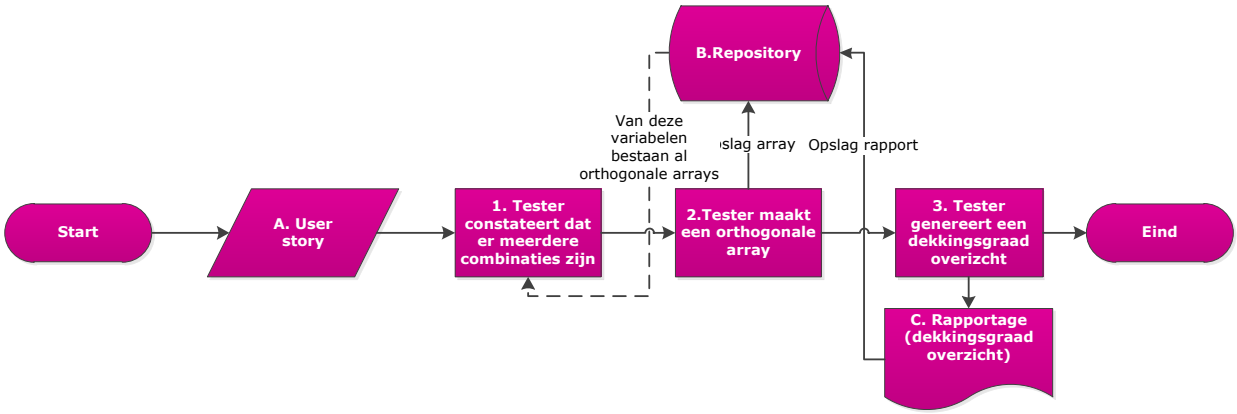
De tester constateert dat een story resulteert in een veelvoud van "testcases". De tester heeft niet de tijd om al deze "testcases" te maken en uit te voeren. Bovendien heeft de tester niet het overzicht welke dekkingsgraad hij haalt bij welke hoeveelheid uitgevoerde "testcases".

Door het gebruik van "pairwise" worden de combinaties van test data gereduceerd. Door een overzicht te maken voor de tester hoeveel tests hij moet uitvoeren om een dekkingsgraad te halen kan deze beter sturen. Door het opslaan van combinaties kunnen deze hergebruikt worden. Het genereren van de dekkingsgraad in een rapportage creëert een kwaliteitsborging.

### Verbetering

In Figuur 14 wordt variant 3 weergegeven. Deze bestaat uit de volgende stappen:

1. De tester constateert uit de user story (A) dat het geen wat hij wil testen resulteert in veel "testcases". De tester haalt de "repository" op om te checken of er al "pairwise" combinatie tabellen bestaan van de gezochte sleutelwoorden. Als deze al bestaan haalt hij de rapportage die erbij hoort op om zijn dekkingsgraad te achterhalen. (in dit geval gaat hij niet verder met stap 2 en 3)
2. De tester vult met behulp van een tool in welke parameters hij heeft in combinatie met andere variabelen en genereerd een orthogonale array waarin hij "pairwise" toepast. Hiermee reduceert de tester het totaal aantal aan combinaties zonder een grote reductie van de dekkingsgraad. De tester slaat de orthogonale array op in een repository (B).
3. De tester genereerd een dekkingsgraad overzicht C. Dit overzicht geeft de tester inzicht in hoeveel verschillende "testcases" moet maken om tot een bepaalde dekkingsgraad te komen.




Figuur 14 variant 3 vi

### Realisatie

De tester gaat op inzicht en ervaring af bij het gebruik van de orthogonale array tool. Deze tool draait lokaal op de pc van de tester. De "repository" is toegankelijk voor alle testers.

### Onderzoek

- Hoe kunnen combinaties worden opgeslagen en gedeeld worden?
- Kan de desbetreffende tool "pairwise" combinaties maken?
- Kan de desbetreffende tool de dekkingsgraad weergeven?

### 3.2.4 Variant 4 Dumb Monkey testen

De Dumb Monkey is een volledig automatische manier van testen. De tool is een vervanging en een aanvulling op de huidige manier van testen.

#### Uitgangspositie

De tester wil de iBurgerzaken applicatie exploratory of systeem testen. Door de Dumb Monkey random input te laten invoeren ontstaan input combinaties waar de tester zelf niet aan toe komt door het volgen van de "normale" testen. Hierdoor ontstaat een vorm van "exploratory testing". De duur van deze test zal dan rond de 24 uur liggen. Dit komt omdat de Monkey genoeg tijd nodig heeft om voldoende combinaties gemaakt te kunnen hebben van parameters. Het overgrote deel van combinaties zal geen fouten opleveren, daarom kost de test veel tijd. De tester krijgt resultaten terug van foutmeldingen.

De tester kan systeemtesten uitvoeren op specifieke gebieden van de iBurgerzaken applicatie.

#### Verbetering

Het systeem doet automatische exploratory tests naast de tester. Dit kost geen tijd van de tester. De tester hoeft voor systeemtesten geen Cucumber scenario's te maken, maar selecteert alleen het gebied wat getest dient te worden. Bovendien is het gebruik van de Dumb Monkey eenvoudig door compacte functionaliteit.

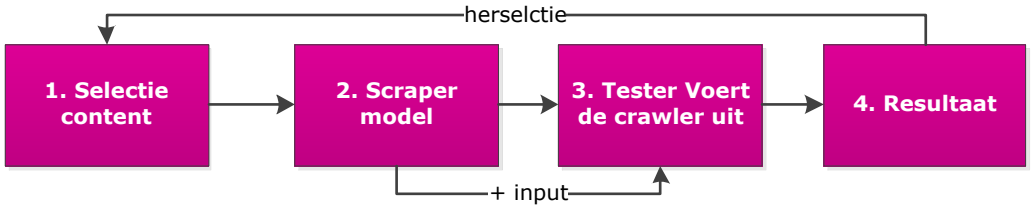
#### Realisatie

*Stappen*

1. De tester/ ontwikkelaar selecteert de paginas binnen iBurgerzaken. Hij selecteert tevens of de Dumb Monkey BSN's[5] gebruikt of handmatig door de applicatie gaat.
2. De Scraper[6] gaat de DOM[7] van de iBurgerzaken applicatie. Aan de hand van de elementen wordt de juiste input gekozen. Hiervoor is een lijst met mogelijkheden hard gecodeerd voor data, textvelden, dropdownvelden etc.
3. De Crawler vult aan de hand van de elementen die verkregen zijn door het scrapen in de juiste velden de juiste input in.
4. Een tool maakt op dat moment dat een errorpagina (onvoorziene fout) zich voordoet een tekst bestandje wordt aangemaakt met daarin de aanleiding en locatie van de fout. Een errorpagina is makkelijk te herkennen in de iBurgerzaken applicatie.

De Crawler reset zichzelf of gaat naar een andere pagina. Hierna wordt de test herhaald of gestopt. Als de Crawler alle pagina's die zijn opgegeven heeft gehad stopt hij.

[5] Burger service nummer
[6] Scraped het domain object model en haar elementen
[7] Domain object model




Figuur 15 variant 4 vi

### Onderzoek

- Hoe gaat de Crawler van pagina naar pagina?
- Welke input heeft de Crawler nodig? (kan een Crawler op zichzelf staan?)

## 3.2.5 Variant 5 Brilliant Monkey test

De Brilliant Monkey is een intelligentere versie van de Dumb Monkey tool. De belangrijkste toevoeging is het maken van "testcases" waarbij rekening wordt gehouden met de volgorde van input en de mogelijk combinaties van staten die de Monkey kan doorlopen. De Monkey maakt combinaties van staten hierdoor ontstaan krachtigere "testcases".

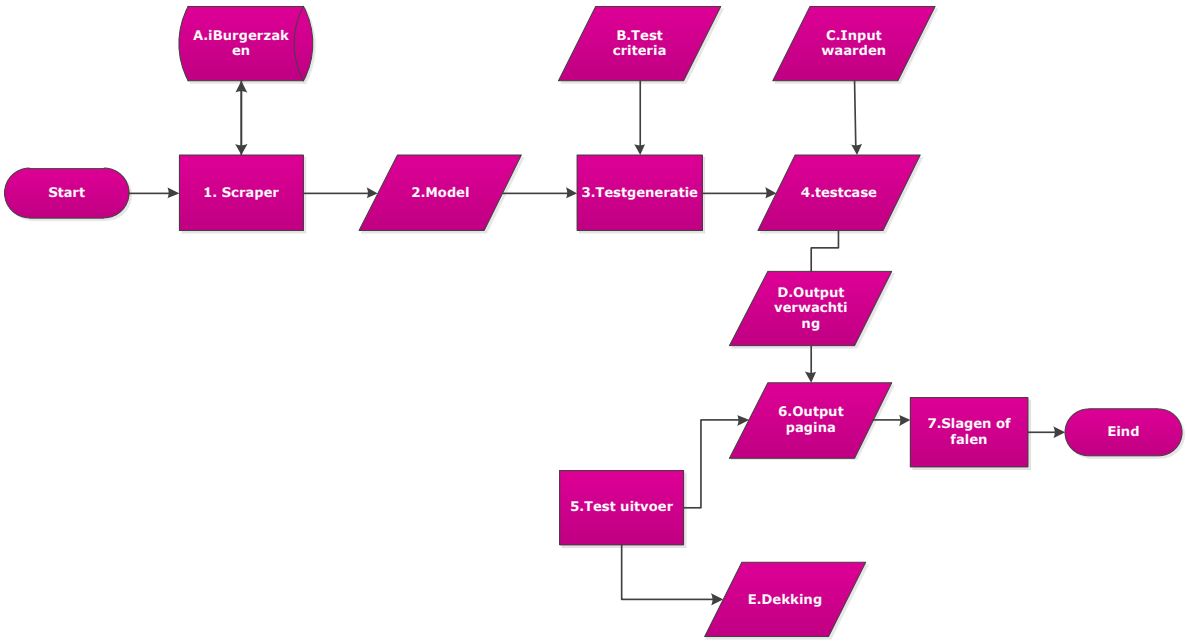
### Uitgangspositie

De tester wil een regressie, systeem of exploratory test uitvoeren op de iBurgerzaken applicatie. Hiervoor wil hij aanvullende dekkingsvormen gebruiken waar de Dumb Monkey niet over beschikt. Deze dekkingsvormen kunnen "pairwise" zijn of een andere vorm van optimalisatie van volgorde waarin de staten worden doorlopen.

### Verbetering

Naast de voordelen van die de Dumb Monkey biedt is de Brilliant Monkey breder inzetbaar door een toevoeging van dekkingsvormen. Door input verwachting zijn er meerdere scenario's waar de applicatie een testcase voor kan maken.

## Realisatie




Figuur 16 Variant 5 vi

Figuur 16 illustreert de architectuur van de Brilliant Monkey test. Deze is opgebouwd uit de volgende stappen:

1. Een model wordt gebouwd aan de hand van de opgeslagen pagina door het scrapen van iBurgerzaken (A).
2. Het model van iBurgerzaken zijn de diverse staten waarin deze kan verkeren.
3. Door het toepassen van combinatorische algoritmes worden aan de hand van de overgenomen staten “testcases” gegenereerd. Hierbij wordt rekening gehouden met de volgende criteria (B): de testdiepte, dekkingsvormen en test duur.
4. Per testcase worden inputwaarden (C) toegekend. Deze inputwaarden zijn afhankelijk van de staten die doorlopen worden tijdens de testcase, maar ook de dekkingsvorm. Bij het toepassen grenswaardes kan deze input verschillen. De output verwachting wordt hier eveneens opgesteld (D). Dit zijn de verwachte staten waarin iBurgerzaken verkeerd tijdens de uitvoer van de testcase.
5. De Monkey voert de testcase uit op de iBurgerzaken applicatie. Hierbij wordt bijgehouden welke staten doorlopen zijn en welke input is toegekend. Dit resulteert in de outputpagina. De dekking (E)geeft de dekkingsgraad weer van de testcase.
6. De outputpagina wordt vergeleken met de verwachte output.
7. Als de verwachte en werkelijke output overeenkomen resulteert dit in een pass, anders wordt dit een fout.

Onderzoek

- Welke architectuur is het meest efficiënt en effectief?
- Hoe kan een model gegenereerd worden met een Crawler?
- Welke tools kunnen een testcase genereren aan de hand van een model?
- Welke tools kunnen deze testcase "interpreteren"?

# 4 ONDERZOEK

Het onderzoek heeft als doel de gewenste situaties te toetsen. Deze situaties worden getoetst door middel van een *"proof of concept"* met als input theorie onderzoeken, feedback van stakeholders, feedback van externe en interne experts (interviews). Het theorieonderzoek heeft als doel om geschreven bronnen zoals "papers", "survey" etc. en interne documenten te gebruiken om tot een beeld te komen van een situatie of tool.

De opzet beschrijft op welke manier het onderzoek gedaan wordt. Hierin worden de criteria die gesteld worden aan de oplosrichtingen opgenomen. Bovendien worden de testopstellingen per variant beschreven.

De resultaten van het onderzoek worden gedocumenteerd in Resultaten. Deze worden per variant uitgewerkt. In Discussie worden de resultaten van de varianten geëvalueerd.

## 4.1 Opzet

Aan de hand van de varianten die zijn beschreven in de gewenste situaties zijn via zoekmachines en advies van derden (experts, collegae) tools gezocht. Deze tools zijn in een long list opgenomen. De tools in de long list zijn beoordeeld op vorm die gedefinieerd is in paragraaf 5.2 en hierdoor ontstond een short list.

Wanneer er voor de oplosrichting geen complete tools wordt geanalyseerd welke deelcomponenten nodig zijn om te komen tot (een deel van) de beoogde oplosrichting. Deze componenten konden een bestaande component zijn of een zelf gefabriceerd component.

De componenten en tools zijn geanalyseerd op compliance, de houdt in dat getest is of deze componenten en tools samen konden werken. Deze tests zijn uitgevoerd door middel van *"proof of concepts"* en door middel van aannames die zijn gedaan op basis van reviews.

In het geval dat meerdere oplosrichtingen voor dezelfde variant gevonden waren zijn er benchmark tests uitgevoerd om hun werking te vergelijken. Deze criteria zijn in samenwerking met J. van der Velde en Dhr. L. Koning samengesteld.

De resultaten van deze tests zijn per variant opgenomen in het hoofdstuk Resultaten. Aan de hand van de resultaten zijn per variant conclusies getrokken over "feasibility", kosten, dekkingsgraad bijdrage en gebruiksvriendelijkheid.

### 4.1.1 Criteria

De rating van de tools per variant wordt gekwalificeerd door de tools per variant tegen elkaar af te wegen op tijdsbesparing, dekking, "feasibility", gebruiksvriendelijkheid. Hierdoor ontstaat een ranking.

Tijdsbesparing: Hierbij wordt gekeken welke tool het meest tijd bespaard ten opzichte van het huidige test proces.

Dekking: : Hierbij wordt gekeken welke tool de beste dekking oplevert ten opzichte van het huidige test proces.

"feasibility": Hierbij wordt gewogen wat de impact is op de huidige situatie als de tool in gebruik genomen zou worden.

Gebruiksvriendelijkheid: Welke tool is makkelijk te gebruiken door de stagiair.

Om de knowhow en de behoeften van het testteam mee te nemen zijn deze meegenomen in de beoordelingen van de varianten. De input hiervoor is: test sessies met Dhr. L. Koning van de specifieke varianten, meningen van andere testers op de presentatie van de varianten.

De resultaten van alle varianten zijn tegen elkaar afwogen in een eindconclusie (discussie), deze zijn opgenomen in 6.3.

### 4.1.2 Testopstelling

Hier volgen de omschrijvingen van de testopstellingen van de diverse varianten.

#### *Variant 1 – CucumberPeople*

Omschrijving: CucumberPeople (CP) is een gratis Eclipse plug-in die gebaseerd op RubyMine[xiii8]. CP is een tool die helpt bij het maken en bewerken van Cucumber-code.

De belangrijkste functies van deze plug-in in zijn:

- Cucumber scenario step completion.
- Vergelijken van scenario stappen met step definities.
- Stap definities aanmaken.

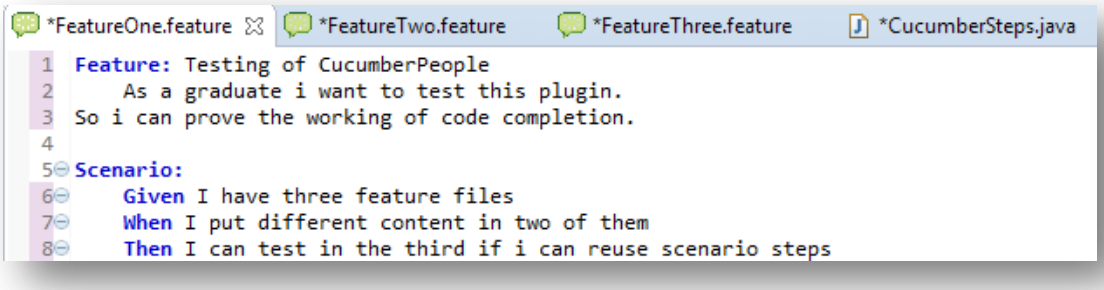
Opstelling:

Alvorens CucumberPeople geïnstalleerd te hebben is Aptana 3 geïnstalleerd. Deze plug-in is noodzakelijk voor de werking van CucumberPeople. De CucumberPeople Library is toegevoegd aan Eclipse. Binnen Eclipse is er een Cucumber project aangemaakt. Binnen dit project zijn 2 *"class"es* aangemaakt: een feature file klasse en een stapdefinitie *"class"*. De feature file klasse bestaat uit drie

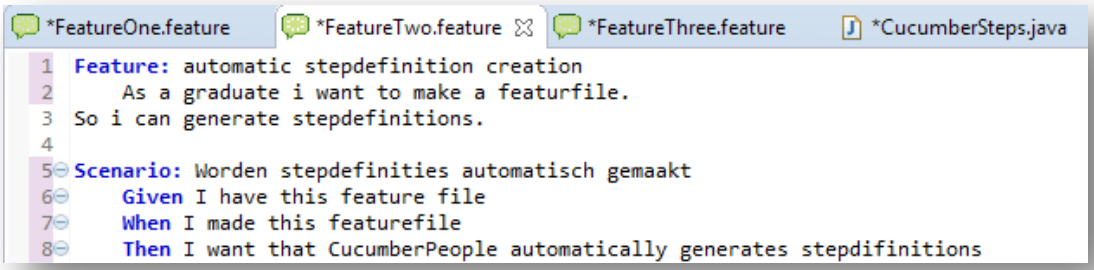
[8] RubyMine is een editor voor de integratie van Rails in Ruby.

feature files: Naast deze simpele basis opstelling is de tool eveneens getest op de huidige testframe waar alle "testcases" zijn gedocumenteerd.

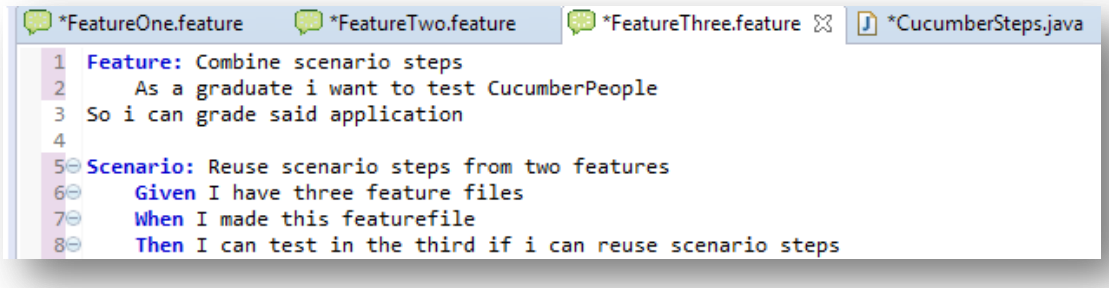
FeatureOne [xiv]:




FeatureTwo xiv




FeatureThree: xiv



### *Variant 2-Selenium IDE*

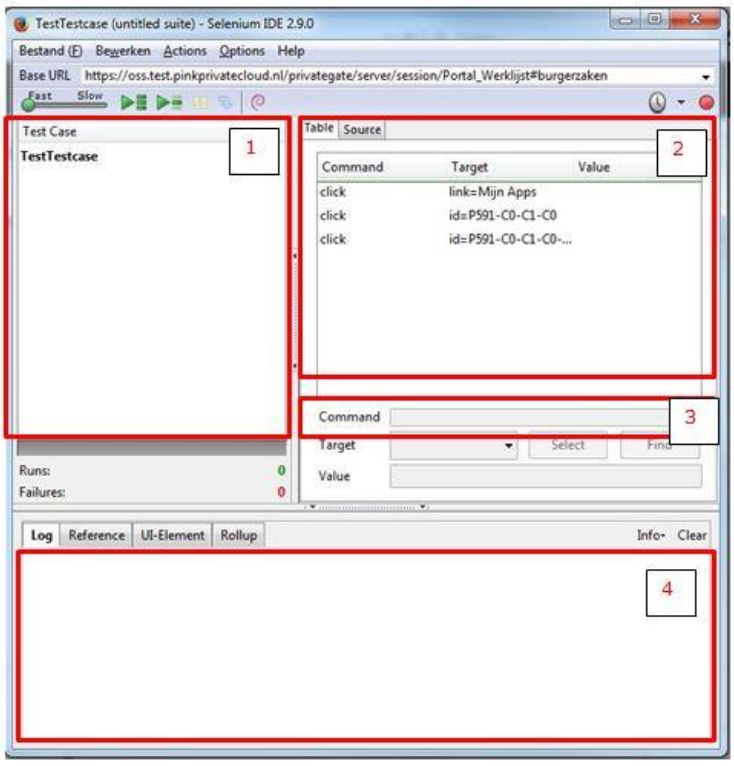
Omschrijving:

Selenium IDE is een plug-in voor Mozilla Firefox. Deze plug-in wordt hoofdzakelijk gebruikt voor "capture and replay" testen.

Opstelling:

Lokaal draait een simpel zelfgebouwde website met 3 pagina's. daarna wordt de iBurgerzaken applicatie getest. De opstelling bewijst of de tool testen op kan nemen en af kan spelen.

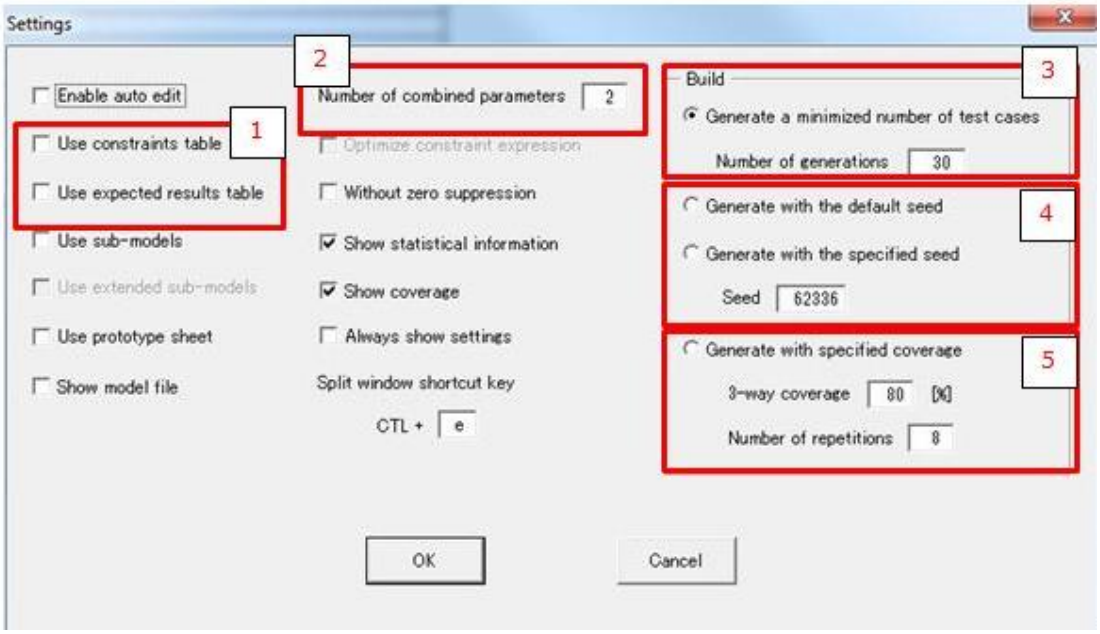
Eveneens wordt de mening van dhr. Dhr. L. Koning en van het testteam meegenomen. Bovendien wordt deze variant waar mogelijk besproken in de geplande expert interviews.




Figuur 17 Selenium IDE [xv]

In figuur 17 is een weergave van de plug-in Selenium IDE. In de rode rechthoeken is met cijfers aangegeven in welke stappen de testcase wordt gecreëerd. Deze stappen zijn:

1. Maken van een testsuite en testcase (s).
2. Dit zijn de stappen die gedaan zijn binnen de iBurgerzaken applicatie en vormen de testcase.
3. Per stap kun je commando's toevoegen, bijvoorbeeld wachttijd.
4. Hier wordt weergegeven of de stappen uitgevoerd zijn en welke mogelijke fouten (fouten) tegen zijn gekomen.




Figuur 18 PictMaster settings xvi

1. Je kunt paren toewijzen die niet bij elkaar passen (constraints) en invullen welke verwachte resultaten uit de "testcases" komen. Het aantal parameters die gecombineerd worden is ingesteld op 2. Zo kun je bepalen hoeveel combinaties je wil maken en welke dekking je wilt bereiken.
2. Dit is hoeveel "testcases" hij genereerd, 30 is volgens de handleiding voldoende en default.
3. Seed (random condities) heeft een kleine invloed op het aantal combinaties. Default is dit 0.
4. Om een efficiënte dekkingsgraad te halen per hoeveelheid "testcases" kun je deze instellen.

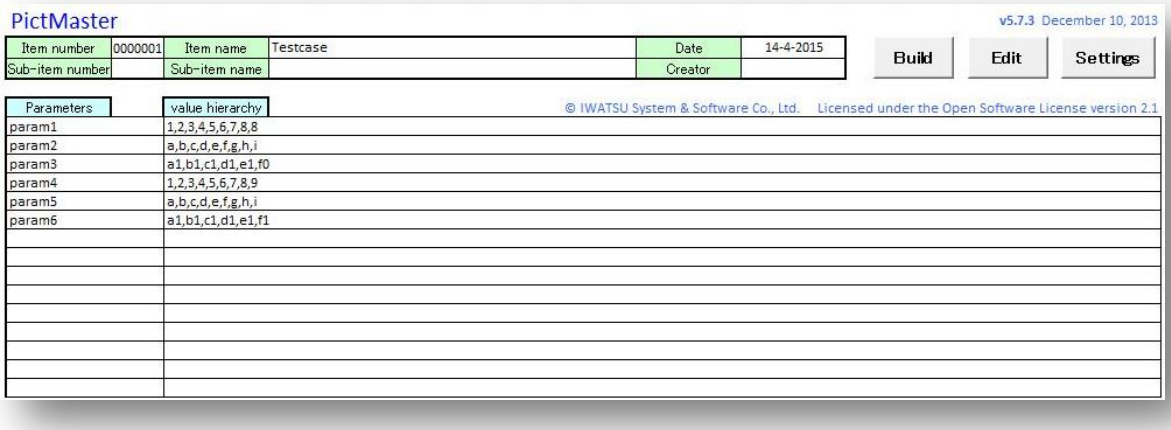
### *Variant 3- PictMaster*

Omschrijving: PictMaster is een spreadsheet based tool voor "pairwise" testen. Met deze tools kunnen combinaties van parameters gemaakt worden door de "pairwise" techniek. Er kunnen combinaties worden gemaakt met 2 of 3 paren, eveneens kan de dekkingsgraad worden bepaald die de tester voor ogen heeft. Door het maken van paren wordt het initieel maximale aantal permutaties gereduceerd.

Opstelling: Aan de hand van een praktijk voorbeeld waar veel permutaties in voorkomen wordt "pairwise" toegepast. Bewezen wordt dat "pairwise" voor een reductie zorgt in het aantal mogelijke permutaties met behoudt van dekkingsgraad.

Eveneens wordt de mening van Dhr. L. Koning en het testteam meegenomen. Bovendien wordt deze variant waar mogelijk besproken in de geplande expert interviews.

Figuur 19 geeft weer waar de parameters en de volgorde hiervan wordt bepaald. Er zijn zes parameters beschreven met elk x waarde.




Figuur 19 PictMaster screenshot [xvi]

### *Variant 4 Dumb Monkey*

Momenteel zijn er geen kant en klare tools die over de beoogde Dumb Monkey functionaliteiten beschikken. Daarom is besloten deze tool binnenshuis te ontwikkelen. De tool is opgebouwd met behulp van Jsoup[xvii], Selenium webdriver en Java.

Jsoup is een library die geschikt is op de DOM van de website op te slaan in web elementen.

Selenium webdriver[xviii] is een web driver die de website benaderd alsof hij een menselijke gebruiker is. Op deze manier heeft de web driver dezelfde voor en nadelen als een echte gebruiker, dit maakt de Dumb Monkey realistisch. De webdriver kan diverse browsers aansturen. Bij de Dumb Monkey is gekozen voor een Firefox driver, deze wordt ook gebruikt in de ART.

De Dumb Monkey is opgebouwd uit diverse "classes". De hoofdclasses zijn: de Crawler "class", de Page *"class"*, de Scraper "class" en de Dumb Monkey "class". De Crawler *"class"* bevat alle methoden die gebruikt worden bij de aansturing van de web driver. Methodes als klikken, invullen van velden en het selecteren van elementen zijn hier geschreven.

De Scraper "class" maakt gebruik van Jsoup en heeft als belangrijkste functie om alle gewenste elementen uit de web pagina te parsen. Deze elementen worden in de Page *"class"* opgeslagen. Er kunnen meerdere versies van de page "classes" worden gebruikt.

De Dumb Monkey "class" brengt de Crawler, Scraper en Page klasse bij elkaar. Dit is het brein van de Dumb Monkey applicatie. Hier wordt bepaald welke pagina's van de iBurgerzaken applicatie worden doorlopen, welke pagina's doorlopen zijn en welke pagina's doorlopen gaan worden.

De Dumb Monkey kan momenteel geen zoekalgoritmes als “breadth” en “depth first” gebruiken omdat door afhankelijkheden de Monkey niet altijd de keuze heeft om een staat terug te gaan. Door het gebrek aan deze vrijheid kan de Monkey niet bewust een “breadth” of “depth first” pad kiezen.

De Dumb Monkey geeft input in de applicatie op basis van verplichte velden, en op basis van een random input methode.

### *Variant 5 Brilliant Monkey*

Deze variant is een uitbreiding op de Dumb Monkey applicatie. Er zijn voor deze variant net als de Dumb Monkey geen commerciële of open source tools gevonden. Het verschil tussen beide vormen is dat de Dumb Monkey applicatie minder dekkingsvormen tot zijn beschikking heeft en de Dumb Monkey heeft een minder complex zoekalgoritme.

Om van de Dumb Monkey een Brilliant Monkey te maken zijn nog veel implementaties nodig. Dit is binnen de afstudeerperiode niet haalbaar meer. Mogelijk kunnen bestaande tools toegevoegde waarde bieden om de Dumb Monkey applicatie in de richting van Brilliant Monkey op te waarderen.

Uit het interview met B. Visser bleek dat Sogeti een tool ontwikkeld heeft die veel overeenkomsten heeft met de beoogde functies voor de Brilliant Monkey applicatie Cover[xix].

Tevens heeft B. Visser gewezen op de tool: Tricentis Tosca. Dit is een test tool die gebruik maakt van Selenium. De tool is ontworpen om het probleem van het breken van testgevallen door een veranderende applicatie op te lossen.

In geval van extra tijd zal het onderzoek naar deze tools zich richten op de vraag:

Kunnen Cover en Tosoca de Dumb Monkey applicatie verheffen tot Brilliant Monkey? Daarbij wordt gekeken naar de input en output van cover en de dekkingsvormen en intelligentie deze bevat. Bovendien wordt de compatibiliteit tussen de huidige Dumb Monkey en Cover meegenomen.

## 4.2 Resultaten

In dit kopje volgen de resultaten uit de vijf varianten. Dit zijn de resultaten die volgen uit de “proof of concepts”, interviews en mening van Dhr. L. Koning, J. van der Velde en de testteams van iBurgerzaken.

### 4.2.1 Resultaten Variant 1 CucumberPeople

In deze paragrafen zijn de resultaten opgenomen van het onderzoek naar CucumberPeople. De belangrijkste functies van CucumberPeople zijn weergegeven in figuur 20 en 21 waar de scenario en step definitie compleet is geïllustreerd. In Figuur 20 CucumberPeople Scenariostep complete xiv wordt geïllustreerd hoe de auto complete van CucumberPeople werkt.

In

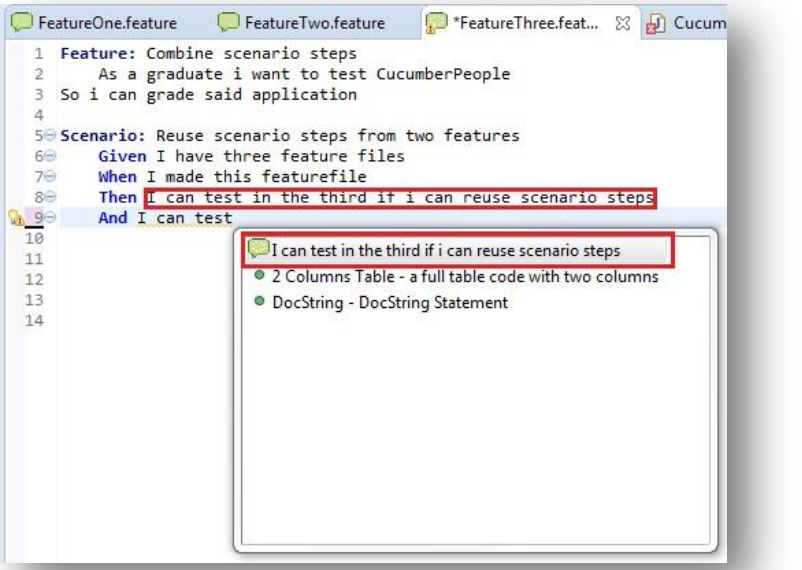
Figuur 21 CucumberPeople stepdefinition complete

De volgende belangrijke functies in CucumberPeople zijn bevestigd:

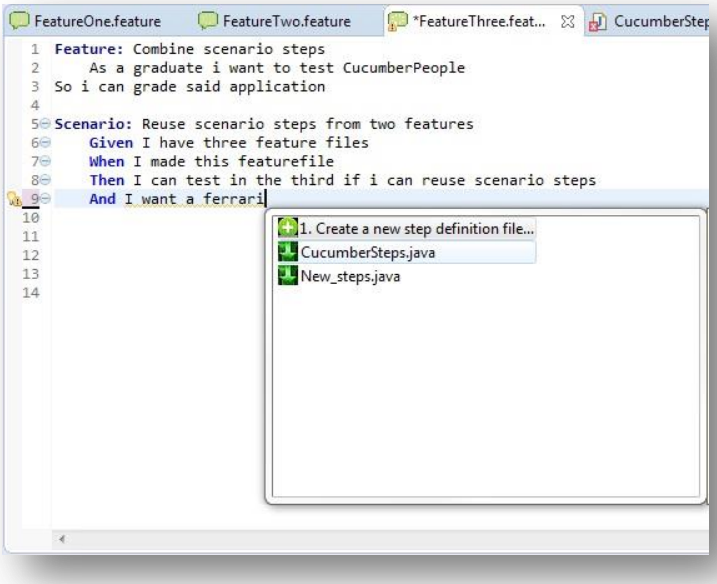
- Scenario step completion
- Checken van scenario steps met stepdefinities.
- Met ctrl click van scenario step naar stepdefinitie.
- Stepdefinities creëren met een quick fix.
- Syntax highlighting
- Auto format
- Error *message* bij dubbele of ontbrekende step definities.

en hier wordt weergegeven hoe de functie benaderd kan worden die er voor zorgt dat stepdefinities automatisch worden gegenereerd.

Bovendien is geconcludeerd dat de plug-in errors geeft als een feature of scenario geen stepdifinities bevat.




Figuur 20 CucumberPeople Scenariostep complete xiv




Figuur 21 CucumberPeople stepdefinition complete

De volgende belangrijke functies in CucumberPeople zijn bevestigd:

- Scenario step completion
- Checken van scenario steps met stepdefinities.
- Met ctrl click van scenario step naar stepdefinitie.
- Stepdefinities creëren met een quick fix.

- Syntax highlighting
- Auto format
- Error *message* bij dubbele of ontbrekende step definities.

Antwoorden op de onderzoeksvragen:

- In welke mate zou CCL dubbele aanmaak van "testcases" of steps kunnen voorkomen?
- In welke mate zou testcase creatie eenvoudiger met CCL?

CucumberPeople verhoogt de vindbaarheid van scenariosteps door de autocompleet functie. Hierdoor kunnen scenariosteps sneller gevonden worden en worden aanpassen minder fouten gemaakt. De tijdsbesparing die gemeten is was ongeveer 1 minuut per scenario door besparing op zoeken en veranderen van scenario steps. Het grote bijkomende voordeel van de "code completion library" is dat de scenario steps nauwkeuriger worden gedaan. Volgens Dhr. L. Koning scheelt deze nauwkeurigheid per scenario het een keer debuggen van een scenario. Dit komt neer op een tijdsbesparing van 3 min.
*Schatting van tijdsbesparing per sprint*[9]

| Gegevens | | Tijdsbesparing |
|---|---|---|
| Gemiddelde aantal scenario's per sprint per team | 75 Totaal is dit 3 teams maal 25 scenario's. | |
| Ervaren testers | 1 | Tijdsbesparing ervaren tester = 25% van 75 scenario's maal 3 minuten = 56,25 minuten per sprint |
| Minder ervaren testers[10] | 3 | 75% van 75 scenario's maal 3 minuten, maal 1,5 = 253 minuten per sprint. |
| Gemiddelde tijdsbesparing van 1 keer debuggen per scenario | 3 minuten | |
| Totale tijdsbesparing | | Dit komt neer op een totale tijdsbesparing per sprint op 309 minuten ofwel 5,2 uur tijdsbesparing per sprint.<br><br>Bij 25 sprints (ongeveer een jaar) komt dit neer op 25 maal 5,2 is 130 uur is ongeveer 5,4 dagen per jaar. |

[9]Gegevens zijn gebaseerd op schattingen van Dhr. L. Koning en de bevindingen uit het "proof of concept".

[10] Conservatief is de tijdsbesparing van CucumberPeople voor minder ervaren testers geschat op 150% van de ervaren tester.

Tabel 2 CCL tijdsbesparing

### Opinie Dhr. L. Koning en het testteam

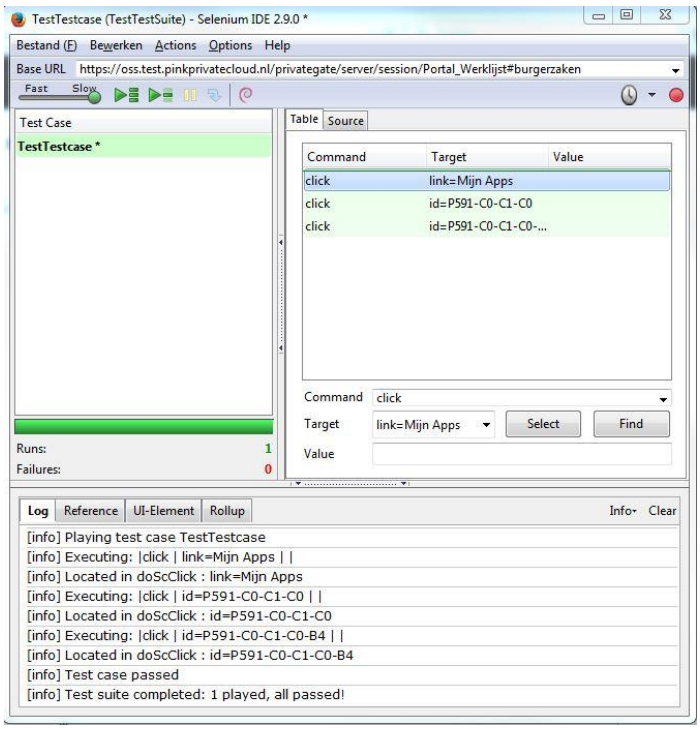
Dhr. L. Koning bevestigt de werking van de auto complete "content" en auto stepgeneratie functie. Ook bevestigt hij de werking van error massages die veroorzaakt zijn door dubbele of ontbrekende step definities.

De syntax highlighting werkte volgens Dhr. L. Koning,maar hij vindt deze vrij beperkt. In de huidige situatie is de Cucumber plug-in uitgebreider. Bovendien werkt de autoformat functie niet altijd naar behoren.

### 4.2.2 Resultaten Variant 2 Selenium IDE

Figuur 22 geeft het resultaat weer van een uitgevoerde testcase. Zoals is geïllustreerd zijn 3 acties opgenomen en succesvol afgespeeld. Helaas toen de testcase werd uitgebreid resulteerde dit in falen bij het afspelen. Dit kwam grotendeels doordat pauzes tussen de acties niet worden opgenomen. Deze pauzes zijn belangrijk omdat de tool expliciet moet wachten op het laden van de pagina. Als de tool te snel is dan herkend deze de juiste elementen niet. Het toevoegen van de juiste wachtlengte op de juiste plekken kostte relatief veel tijd.

Hergebruik van "testcases" was bijna niet te doen doordat elementen die opgenomen worden geen duidelijk naamgeving hebben. Deze zou je kunnen toevoegen, maar een dergelijke documentatiestap maakt dat de testcase niet sneller wordt gebouwd dan op de huidige manier.




Figuur 22 resultaten Selenium IDE xv

### 4.2.3 Resultaten Variant 3 PictMaster

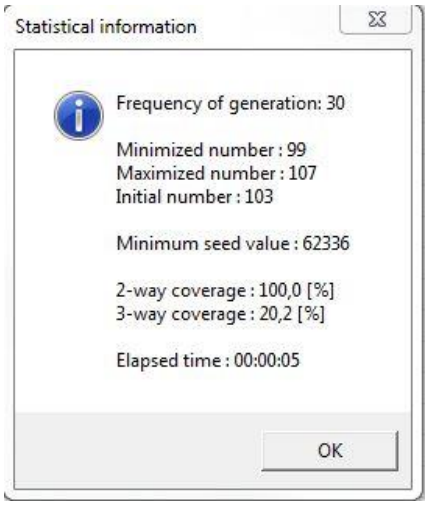
In deze paragraaf zijn de resultaten opgenomen van het onderzoek naar PictMaster. De volgende vragen waren vooraf opgesteld:

- Hoe kunnen combinaties worden opgeslagen en gedeeld worden?
- Kan de desbetreffende tool "pairwise" combinaties maken?
- Kan de desbetreffende tool de dekkingsgraad weergeven?

Het opslaan van de "testcases" kan centraal, maar gelijktijdig eraan werken gaat niet omdat de tool in Excel is geschreven. De tester voert lokaal zijn eigen test uit op zijn kopie van PictMaster. Het delen van "testcases" zal hierdoor buiten de tool om geregeld moeten worden.

Door tool later samen met Dhr. L. Koning te testen werd inzichtelijk dat de aanleiding om deze tool te gebruiken bij het dagelijkse testen niet altijd duidelijk naar voren komt. Niet altijd is het vanzelfsprekend dat het testen van een story resulteert in veel testgevallen. Hierdoor is de toepassing nog niet altijd eenvoudig aan de hand van een story. In veel situaties volgens Dhr. L. Koning moeten deze worden uitgedacht, pas als de resultaten van de test voorspeld zijn komt de tester tot de conclusie dat er te veel testgevallen zijn om deze allemaal mee te nemen. Op dat moment moet de conclusie worden getrokken om "pairwise" toe te passen.

Na het uitvoeren van de vooraf gedefinieerde testcase volgde de volgende resultaten (Figuur 23). Door 30 generaties (toepassen van het "pairwise" algoritme door PictMaster) uit te voeren is het resultaat 99 minimum "testcases", maximum van 107 en 103 initieel (standaard). Bij het genereren van "testcases" door PictMaster ontstaat een maximale afwijking van 10 procent. De kans op minder "testcases" kan vergroot worden door het "minimum seed value" te verhogen [xvi]. De dekkingsgraad is vermeld bij 2-paar en 3-paar combinaties. In figuur 23 zijn onder andere de dekkingsgraden te zien bij combinaties van 2 en 3.




Figuur 23 PM Dekkingsoverzicht xvi

### 4.2.4 Resultaten Variant 4 Dumb Monkey

De Dumb Monkey applicatie is ontwikkeld voor de ambtenarenzijde en de burgerzijde van iBurgerzaken. De ambtenarenzijde is alleen benaderbaar door de ambtenaar. De burgerzijde is benaderbaar door de burger. Hier kunnen door de burger de intakes worden gedaan op de processen: Geboorte aangeven, verhuizen naar buitenland, geboorte aangifte, huwelijk en overlijden. De controle en levering van de processen gebeurt op de ambtenaren zijde.

De Dumb Monkey applicatie is op dit moment nog niet af. De Dumb Monkey kan momenteel:

- Een staat iBurgerzaken opdelen in elementen (textvelden, datavelden dropdownvelden) en deze opslaan.
- Random (zo ver mogelijk) alle input mogelijkheden benaderen bijvoorbeeld: dropdownvelden aanklikken, data invullen, textvelden invullen, knoppen klikken.
- Herkennen wanneer een nieuwe staat zich voordoet(na klikken op een knop, invullen van een dropdown waarde etc.).

Uit het interviews met B Visser en L. van der Aalst blijkt dat “model based testing” een belangrijke ontwikkeling is. B. Visser geeft aan dat de dekkingsvorm grenswaarde een goede toevoeging zou zijn voor de Dumb Monkey.

Dhr. L. Koning geeft aan dat de Dumb Monkey een goede toevoeging kan vormen op de bestaande testaanpak, maar spreekt daarnaast wel de twijfel uit over de mogelijke testduur, deze zou volgens hem te lang kunnen worden. Dit komt volgens hem door het gebrek aan linken en knoppen aan de ambtenarenzijde. Daardoor kunnen niet genoeg combinaties gemaakt worden.

Hierdoor ziet Dhr. L. Koning vooral mogelijkheden in toepassing aan de burgerkant van iBurgerzaken naast de ambtenarenkant. Vooral aan de burgerkant zijn unieke combinaties van paden en invulmogelijkheden zeer relevant omdat deze door veel soorten gebruikers benaderd wordt en de applicatie niet in alle huidige tests mee wordt genomen. Ook is aan de burgerkant de kans van een incident groter dan aan de ambtenarenkant. Dit komt omdat de burgerzijde door meer gebruikers wordt benaderd dan de ambtenaren zijde.

#### Zoekalgoritmes

Het “breadth” en “depth first” algoritme kan op dit moment niet toegepast worden. Dit komt doordat de Dumb Monkey niet in ieder geval terug kan in de staat waar hij vandaan kwam. Dit komt door afhankelijkheden in iBurgerzaken en door de Dumb Monkey niet meer dan 1 staat vooruit kan kijken. Daarom volgt hij geen uitgestippeld pad.

#### Toekomstige functies

Om de Dumb Monkey applicatie voldoende te kunnen testen middels een “proof of concept” zijn de volgende functies gewenst:

- Herkennen van een foutmelding.s

- Rapporteren van betrokken staat en parameters bij een foutmelding in de vorm van een tekstbestand.
- Kunnen selecteren van een startpositie. (welk proces(sen) je wil testen)
- De dekkingsvorm grenswaarde toevoegen op de inputwaarden.

### 4.2.5 Resultaten Variant 5 Brilliant Monkey

Momenteel zijn er (nog) geen resultaten.

#### Verwachte resultaten:

De tools van Sogeti zullen naar waarschijnlijkheid de dekkingsvormen bezitten die de Dumb Monkey intelligenter kunnen maken. Om deze dekkingsvormen in te kunnen zetten zullen de Dumb Monkey applicatie en de tools van Sogeti met elkaar moeten communiceren.

Hiervoor moet worden onderzocht welke input de tools van Sogeti vereisen en of de Dumb Monkey deze kan produceren. Ook is nog niet duidelijk waar in het Dumb Monkey proces de tools van Sogeti gebruikt kunnen worden en hoe de input-proces-output moet verlopen.

## 4.3 Discussie

Hier wordt de terugkoppeling gedaan van de onderzoeksvragen die per variant zijn opgesteld. Bovendien wordt aan de hand van de criteria, ratings en tester ratings uiteengezet welke tool voor welke variant het meest geschikt is.

In de paragraaf combinaties wordt bediscussieerd welke tools naast elkaar gezet kunnen worden om een zo groot mogelijk voordeel te creëren voor testdekking en tijdswinst.

### 4.3.1 Variant 1 CucumberPeople

Het merendeel van de beloofde functies van CucumberPeople werkt naar behoren. Dhr. L. Koning en het testteam zijn tevreden over de belangrijkste functie: code completion. Bij gebruik van CucumberPeople worden “testcases” nauwkeuriger gemaakt het risico kleiner wordt op foutieve “testcases” wat resulteert in tijdsbesparing van 5,2 uur per sprint en 5,4 dagen per jaar. De besparing is naar schatting groter bij minder ervaren testers.

| Variant 1 CucumberPeople | |
|---|---|
| Tijdsbesparing | ★ |
| Dekking | ☆ |
| Feasibility | ★ |
| Gebruiksvriendelijkheid | ★ |

Figuur 24 beoordeling CucumberPeople

Kijkend naar de criteria:

| Criteria | Waardering |
|---|---|
| Tijdsbesparing | Goed, 5,4 dagen per jaar worden ermee bespaard. |

| | |
|---|---|
| Dekking | Deze variant voegt niet direct toe aan de dekkingsgraad, maar wel aan de verlaging van fouten in scenario's. |
| "feasibility" | Makkelijk te implementeren, binnen 20 min voldaan. Deze tool kan (moet) goed naast de huidige testopstelling gebruikt worden. |
| Gebruiksvriendelijkheid | Zeer gebruiksvriendelijk. Weinig commando's en heeft sneltoets combinaties. |

### 4.3.2 Variant 2 Selenium IDE

Selenium IDE doet weinig automatisch, je moet zelf bepalen na iedere handeling die is opgenomen welke actie ondernomen moet worden, bijvoorbeeld wachten op het laden van een knop. Het testen op de zelfgebouwde site ging vlekkeloos. Door een beveiligings feature kan Selenium IDE niet zelf inloggen in de iBurgerzaken applicatie. Hiervoor werd een "work around" gemaakt. De "testcases" waren eenvoudig en snel gemaakt. De installatie was eenvoudig. Deze applicatie is licht genoeg om naast bijvoorbeeld Eclipse te draaien.

Bij een andere test van deze applicatie werd duidelijk dat er meerdere wachtmomenten in het afspeelproces moeten worden geplaatst omdat anders de test faalt.

Deze variant is vooral geschikt om snel kleine "testcases" uit te voeren. Bijvoorbeeld bij ontwikkeltests. Hergebruik van "testcases" zou in theorie mogelijk zijn, maar daar is deze tool niet echt geschikt voor door zijn beperkingen in het gebruik.

Dhr. L. Koning geeft aan dat hij Selenium IDE gebruikt om Xpath's in de iBurgerzaken te vinden. Dit is volgens hem een efficiënte en snelle manier om deze te vinden.

| Variant 2 Selenium IDE | |
|---|---|
| Tijdsbesparing | ☆ |
| Dekking | ☆ |
| Feasibility | ★ |
| Gebruiksvriendelijkheid | ★½ |

Figuur 25 beoordeling Selenium IDE

Kijkend naar de criteria:

| Criteria | Waardering |
|---|---|
| Tijdsbesparing | Nauwelijks tijdsbesparing bij ontwikkeltests, geen tijdsbesparing bij systeemtests, door gebrek aan (eenvoudig) hergebruik van testcase. |
| Dekking | Vergelijkbare dekking als de huidige ART test, gui based testen namelijk. |
| "feasibility" | Zeer eenvoudig te installeren, 3 minuten werk. Deze tool kan goed naast de huidige testopstelling gebruikt worden. |

| | |
|---|---|
| Gebruiksvriendelijkheid | Makkelijk, binnen 20 minuten kun je een simpele testcase maken. Alleen het kost tijd om deze met succes af te spelen. Het inbouwen van wachtmomenten is noodzakelijk. |

### 4.3.3 Variant 3 PictMaster

Een simpele, maar doeltreffende tool om slimme combinaties te maken. Door een testcase te creëren met combinaties van setjes van 2 of 3 en de dekkingsgraad te kiezen wordt het aantal permutaties gereduceerd en dus ook het aantal "testcases".

Dhr. L. Koning geeft aan dat deze tool erg waardevol is voor het testen van iBurgerzaken. Momenteel loopt PinkRoccade risico's die volgen uit het verzuimen van niet geanticipeerde combinaties van parameters. Door structureel gebruik van "pairwise" worden combinaties van parameters meegenomen waar anders niet aan gedacht zou zijn.

Om "pairwise" en PictMaster succesvol in te zetten is het essentieel om een gebruikshandleiding te publiceren. Naar aanleiding van de presentatie van deze variant in het testteam blijkt dat een handleiding en de toewijzing van een zogenaamde test tool beheerder helpend kan zijn. De test tool beheerder weet alle ins and outs van PictMaster en kan tips geven in wat voor situaties, story's en testgevallen PictMaster gebruikt is, of kan worden. Het documenteren van story's waarbij "pairwise" is toegepast kan bovendien het gebruik bevorderen.

#### Interviews

Uit de interviews met B. Visser en L. van der Aalst blijkt dat "pairwise" testen een sterke dekkingsvorm is. L. van der Aalst geeft aan dat veel testers over het algemeen "pairwise" niet goed inzetten door gebrek aan kennis en kunde. Dit zou een mogelijke valkuil kunnen zijn bij eventuele implementatie.

#### Deskresearch

Uit het deskresearch blijkt "pairwise" een effectieve methode is om fouten te vinden die veroorzaakt worden door een combinatie parameters xxxv ,xxxvi.

#### Beoordeling

| Variant 3 PictMaster | |
|---|---|
| Tijdsbesparing | ☆ |
| Dekking | ★ |
| Feasibility | ★ |
| Gebruiksvriendelijkheid | ☆ |

Figuur 26 beoordeling PictMaster

Kijkend naar de criteria uit het deskresearch en "proof of concept"s blijkt dat:

| | |
|---|---|
| Criteria | Waardering |

| | |
|---|---|
| Tijdsbesparing | Theoretisch gezien bespaard deze tool veel test tijd door het reduceren van “testcases”. Echter in de praktijk worden nu ook niet alle mogelijke “testcases” van combinaties meegenomen. Bovendien kost het genereren van pairs met “pairwise” ook tijd. Deze tijd komt bovenop de tijd die het kost om Cucumber scenario’s te maken. |
| Dekking | De tester heeft meer inzicht in de dekkingsgraad, bovendien worden voldoende tests uitgevoerd om deze te halen. Door het structureel herkennen van “testcases” waar veel combinaties zijn en daar “pairwise” toe te passen wordt het risico op fouten die niet herkend worden lager. |
| “feasibility” | Goed, Excel is al standaard geïnstalleerd. De tool zelf vereist geen installatie. Deze tool kan goed naast de huidige testopstelling gebruikt worden. |
| Gebruiksvriendelijkheid | Als je de bijgeleverde instructies doorleest kun je binnen 30 min een testcase maken. Echter wanneer “pairwise” ingezet moet worden is niet altijd duidelijk. Zonder verdere instructie kan deze methode niet goed gebruikt worden. |

### 4.3.4 Variant 4 Dumb Monkey

De Dumb Monkey applicatie is nog niet volledig ontwikkeld. Mede hierdoor kan (nog) niet bewezen worden of de beoogde voordelen daadwerkelijk realiseerbaar zijn.

| Variant 4 Dumb monkey | |
|---|---|
| Tijdsbesparing | ½ |
| Dekking | ½ |
| Feasibility | ½ |
| Gebruiksvriendelijkheid | ★ |

Figuur 27 beoordeling Dumb Monkey

Kijkend naar de criteria:

| Criteria | Waardering |
|---|---|
| Tijdsbesparing | De Dumb Monkey is een volledig automatische test. De tool is ontworpen voor “exploratory testing” en voor systeemtesten. Een groot deel van de huidige testcase creatie kan vervangen worden in de specifieke situaties (bijvoorbeeld kleine ontwikkeltesten) waarin de Dumb Monkey kan worden gebruikt. Door een aantal voor gedefinieerde, niet random waarden is de applicatie wel onderhoudsgevoelig. Als je dit optelt bij de testtijd, gaat de tijdsbesparing wel omlaag. |
| Dekking | Door automatisch “exploratory testing” kan de applicatie in een tijd van 20 uur een aantal fouten vinden. De applicatie loopt minder risico hierdoor dat deze fouten (te) laat worden gevonden. Door het toevoegen van de |

| | |
|---|---|
| | dekkingsvorm grenswaarde levert de tool naast exploratory combinaties fouten ook grenswaarde fouten op. Echter het is nog niet bewezen dat 20 uur testen voldoende is. Dit maakt de dekking nog onzeker. |
| "feasibility" | De tool is in Java geschreven wat maakt dat deze voor bijna ieder platform werkend kan worden gemaakt. De tool zou ook via Eclipse gestart kunnen worden. Iedere tester en een aantal ontwikkelaars beschikken al over Eclipse. IBurgerzaken maakt dat de tool door verschillende afhankelijkheden niet geheel zelfstandig kan opereren. Sommige parameters dienen vooraf gedefinieerd te worden. |
| Gebruiksvriendelijkheid | De tool is erg overzichtelijk in haar functies. Na het selecteren van een start positie hoeven alleen nog maar de resultaten gelezen te worden. |

### 4.3.5 Variant 5 Brilliant Monkey

Deze resultaten zijn (nog) onbekend. Verder onderzoek is nodig om een verwachte resultaten te formuleren.

| Variant 5 Brilliant monkey | |
|---|---|
| Tijdsbesparing | ? |
| Dekking | ? |
| Feasibility | ? |
| Gebruiksvriendelijkheid | ? |

Figuur 28 Beoordeling Brilliant Monkey

# 5 CONCLUSIES EN AANBEVELINGEN

In dit hoofdstuk zijn de conclusies getrokken uit PATG. Naast de conclusies zijn aanbevelingen gedaan, deze hebben betrekking op de vijf onderzochte varianten.

## 5.1 Conclusies uit het onderzoek

Uit onderzoek is gebleken dat volledige testcase automatisering met behulp van "model based"op termijn mogelijk is. Echter vroeg in het onderzoek bleek dat volledige automatisering van "testcases" op de huidige situatie op dit moment onhaalbaar is. Met deze kennis heeft het onderzoek zicht gericht op het zoeken naar semi automatische testcase generatie. Na deskresearch en het houden van interviews heeft het onderzoek zich gericht op vijf verschillende varianten die elk potentieel een bijdrage aan de testdekking konden doen. Deze vijf varianten zijn beoordeeld vier punten door de stagiair, eveneens zijn de meningen van het testteam, Dhr. L. Koning en J.D. van der Velde hierin opgenomen.

Een samenvatting van alle beoordelingen is weergegeven in figuur 29.

| | Tijdsbesparing | Dekking | Feasibility | Gebruiksvriendelijkheid |
|---|---|---|---|---|
| Variant 1 CucumberPeople | ★ | ☆ | ★ | ★ |
| Variant 2 Selenium IDE | ☆ | ☆ | ★ | ½★ |
| Variant 3 PictMaster | ☆ | ★ | ★ | ☆ |
| Variant 4 Dumb monkey | ½★ | ½★ | ½★ | ★ |
| Variant 5 Brilliant monkey | ? | ? | ? | ? |

Figuur 29 Samenvatting beoordelingen

De conclusies per variant zijn als volgt:

### CucumberPeople

Deze tool bespaart relatief veel tijd op het huidige testproces. Het risico op foutieve scenario steps wordt verkleind. Doordat de applicatie binnen Eclipse draait kan het hele team zonder weinig handelingen te tool in hun framework integreren. De functies van CucumberPeople zijn eenvoudig te benaderen. De tool heeft toegevoegde waarde voor de ervaren en minder ervaren tester.

### Selenium IDE

Uit de resultaten is gebleken dat de huidige manier van testcase generatie beter en sneller te onderhouden is en dat de snellere generatie van "testcases" bij "capture and replay" daar niet tegenop weegt. Bovendien is hergebruik van deze "testcases" in de huidige situatie niet mogelijk. De testvorm functioneert pas na het toevoegen van diverse wachtmomenten. Doordat het veel tijd kost om wachtmomenten in te bouwen heeft de tool bij systeemtest en ontwikkeltest doeleinden geen toegevoegde waarde.

### PictMaster

Een belangrijke dekkingsvorm om fouten die resulteren uit combinaties van parameters te detecteren. Deze tool is volgens Dhr. L. Koning een essentiële toevoeging voor iBurgerzaken. Het gebruik van de tool zelf is relatief eenvoudig, echter het herkennen van situaties waar de tool gebruikt moet worden

blijkt lastig. Bovendien geeft PictMaster inzicht in hoeveel testen gedaan moeten worden om een bepaalde testdekking te kunnen behalen. Dit is volgens Dhr. L. Koning een zeer nuttige bijkomstigheid.

### Dumb Monkey

Naar verwachting is de Dumb Monkey een aanvulling op de huidige testsituatie. De toepassing van de tool is erg breed, van "exploratory testing" tot systeemtests. Het feit dat de tool gebruikt zou kunnen worden door testers en ontwikkelaars maakt de tool ook breed inzetbaar. Gebruik en installatie van de tool zijn naar verwachting makkelijk.

Echter door voortschrijdend inzicht blijkt de tool niet volledig automatisch te werken. De iBurgerzaken applicatie maakt dat sommige parameters vooraf ingevuld dienen te worden. Het gaat hier om bijvoorbeeld Burgerservicenummers, maar ook de definitie van sommige knoppen. Dit maakt de applicatie onderhoudsgevoeliger.

De applicatie is momenteel niet compleet genoeg om door middel van "proof of concept" te beoordelen. Door de ontwikkeling van de in 5.2.4 benoemde toekomstige functies is de applicatie volwassen genoeg om getest te kunnen worden.

### Brilliant Monkey

Er is nog te weinig onderzoek gedaan om mogelijke resultaten te bespreken van deze tool.

### Combinaties

De varianten kunnen naast elkaar gebruikt worden. Maar dit is niet in alle gevallen nuttig. De combinatie Selenium IDE met de Monkey applicaties is niet nuttig. De Monkey zou bij ontwikkeltests in veel gevallen veel tijdbesparender zijn dan Selenium IDE.

## 5.2 Aanbevelingen

Aan de hand van de conclusies zijn de volgende aanbevelingen gedaan per variant[11]:

| Variant | Implementatie |
|---|---|
| CucumberPeople | ✔ |
| Selenium IDE | ✘ |
| PictMaster | ✔ |
| Dumb monkey | ✔ |
| Brilliant monkey | ? |

Figuur 30 Samenvatting aanbevelingen

[11] Kosten zijn niet meegenomen omdat alle varianten(op Cover na) open source of zelf ontwikkeld zijn.

### CucumberPeople

Deze tool levert het meest op als deze zo snel mogelijk wordt ingevoerd. De belangrijkste reden hiervoor zijn de risico verlaging van foutieve "testcases". De tool werkt naar behoren en de voordelen zijn meetbaar. De tool heeft geen grote nadelen. De tijdsbesparing is goed.

### Selenium IDE

Selenium IDE niet in gebruik nemen. Het maken van "testcases" werkt niet goed genoeg. Bovendien Kunnen de Monkey applicaties dezelfde tests uitvoeren zonder dat dit tijd kost voor de tester zelf. Het maken van Selenium IDE tests kost ook tijd en het hergebruiken van deze tests is moeilijk.

### PictMaster

In gebruik nemen. PictMaster bevat de "pairwise" dekkingsvorm is zeer nuttig en komt de testdekking erg ten goede. Dit blijkt uit de interviews met L. van der Aalst, B. Visser en de ervaring van Dhr. L. Koning. De testvorm heeft de potentie om fouten uit de applicatie te halen die momenteel niet gevonden worden.

Dhr. L. Koning en het test team van iBurgerzaken geven aan dat de inzet van "pairwise" een struikelblok kan zijn doordat het moeilijk in te schatten is wanneer de methode het beste ingezet kan worden. Daarom zou het wijs zijn dat iemand verantwoordelijk wordt voor de tool en communiceert naar het testteam hoe en in welke situatie deze tool gebruikt dient te worden. De bijgevoegde bijlage: "pairwise" manual doet al een aardige stap in de communicatie en kennisdeling van deze tool. Het documenteren van story's die betrekking hebben op "pairwise" werkt dit eveneens in de hand.

### Dumb Monkey

De Dumb Monkey applicatie kan in potentie nuttig zijn voor ontwikkeltests, maar ook voor exploratory tests. De variant is (nog) niet bewezen middels een "proof of concept", maar met wat extra tijd kan de tool voldoende functionaliteit bieden om als input te dienen voor een "proof of concept".

Bovendien kan een proefperiode met de tool uitwijzen of de tool een positieve invloed heeft op testdekking en/of testtijd. Dhr. L. Koning zou de tool voor een periode van een maand in gebruik kunnen nemen. Zijn bevinden zijn input voor een evaluatie waarin wordt besloten om verder te ontwikkelen, te stoppen of de tool te verspreiden onder de testers van andere ontwikkelteams.

Als blijkt dat de tool een succes is en er is behoefte aan extra functionaliteit dan kunnen stappen ondernomen worden om de tool verder om te bouwen naar Brilliant Monkey.

### Brilliant Monkey

De stagiair heeft de mogelijkheid om Cover van Sogeti te beoordelen. Deze tool-kit heeft wellicht toegevoegde waarde op de Dumb Monkey (waardoor deze wordt geüpgraded) of op het huidige testproces. Het testen van deze tool-kit is daarom aan te bevelen.

# BIJLAGE A: DEKKINGSVORMEN

| | |
|---|---|
| Afvinklijst | Deze lijst wordt gebruikt om delen van de lijst te testen met een deel per testgeval minimaal. Hierbij is geen variatie in zwaarte mogelijk [xx] |
| Beslissingstabel | Een beslissingstabel is een dekkingsvorm die handig is als je situaties wilt testen die onder meerdere condities voor komen. De tabel bestaat normaal uit vier kwadranten: Condition stub, Action stub, Condition entry en Action entries. Contion stub bestaat uit de criteria of parameters die de conditie weergeven. De action stub geeft de acties weer die getest worden. De condition entry geeft de regels weer van de condities. De action entries geeft het resultaat weer van de condition entries[xxi]. |
| Cause–effect graph | Deze dekkingsvorm mapt de input van de applicatie als cause en geeft de output weer als effect. Meestal wordt links de cause gemapt en rechts het effect [xxii]. |
| CRUD/BREAD | Een acroniem voor Create, read, update en delete [xxiii].<br>Een andere variant is BREAD. Dat is een acroniem voor Browse, Read, Edit, Add en Delete[xxiv]. |
| Equivalentieklassen | Deze dekkingsvorm groepeert dezelfde test condities die het systeem als gelijken behandelt [xxv]. |
| Error guessing | Is een dekkingsvorm waar de tester intuïtief zonder vaste structuur test. Deze test wordt niet gedocumenteerd en er zijn geen gestructureerde vormen van [xxvi]. |
| Event based model | Is een model van een van de events van een GUI [xxvii]. |
| Goedpaden foutpaden | Is een dekkingsvorm waarbij goedpaden (flows die lopen zoals weergegeven in de testbasis) en foutpaden (flows die niet lopen zoals weergegeven in de testbasis). |
| Grenswaarde analyse | Lijkt op de equivalentieklassen maar richt zich op het overschrijden van parameter waarde [xxviii]. |
| Load profiles | Belasten van het systeem door de simulatie gebruik in verschillende kwantiteiten[xxviii]. |
| Operational profiles | Testen door middel van realistische simulatie van het systeemgebruik |
| Paden | Testen door het dekken van combinaties van paden met een schema van beslissingen en paden [xxviii]. |
| State transition analysis | Een component van een applicatie kent meerdere statussen. Een status kan veranderen ondanks de input gelijk blijft. Het testen van statussen wordt aan de hand gedaan van een state transition model. Deze geeft aan in welke staat de applicatie verkeerd, de transitie tussen 2 staten, het event als resultaat van de transitie, de actie als resultaat van de transitie [xxix] |

# BIJLAGE B: INTERVIEWS

## *Samenvatting interview L. van der Aalst inclusief vragen*[12]

Bij dit interview is een transscriptie toegevoegd met een gesprek tussen J. van der Velde en J. Neijts. Dit om de inhoud van het interview verder te verduidelijken. Dit was nodig omdat het interview niet in detail uitgewerkt kon worden door technisch falen van de opname apparatuur.

### Interview: 9-4-2015.

### Samenvatting gesprek:

Tijdens dit interview heeft Leo aangegeven niet in te kunnen gaan op details van de tooling.

We hebben het globaal gehad over de varianten, de business case, dekkingsvormen.

Dekkingsvorm "pairwise", een krachtige tool maar volgens Leo wordt deze tool niet of nauwelijks ingezet door de tester. De rede is vaak dat zij de business logica niet begrijpen. Over het algemeen wordt veel getest wat de tester kent dit zijn vaak testen met een laag risico, juist de testen die een hoog risico dragen worden niet getest. Vanuit het testmanagement wordt te weinig gestuurd hierop.

"model based"testen is volgens Leo de toekomst. Hier gaat het steeds meer naar toe. Binnen zijn organisatie wordt dit toegepast.

Over het hergebruik van "testcases" zegt Leo dat veel testen met "model based"testtechniek "on the fly" worden gegenereerd , men hergebruikt de testcase niet, maar vervangt deze.

Ik vroeg leo hoe hij in een business case kwalificeert. Onlangs heeft hij dit in een project gedaan door middel van een "proof of concept" waar hij handmatige testcase creatie naast automatische testcase creatie zette. Hierbij kon 90 procent sneller worden getest tijdens automatisch testen.

Op de vraag of hij nog iets miste, of incorrect vond antwoordde hij :" ik zie dat je het over "behavior driven" testen hebt en "business driven testing", maar kijk ook eens naar "acceptance test-driven development". Bovendien vroeg hij of ik iets wilde doen met unittests want daar is veel automatisering voor.

Leo gaf aan dat hij binnen zijn organisatie toolspecialisten heeft, die hij vraagt of ze mij kunnen helpen met de techniek van de varianten. Hij heeft in een mail aangegeven dat hij Richard Ammerlaan, Marco Jansen van Doorn en Ben Visser hiervoor heeft benaderd. Zij benaderen mij als ze daar tijd voor hebben.

[12] Helaas door falen van opnameapparatuur is de samenvatting niet in detail volledig.

Transscriptie:

N.B. De tekst zijn letterlijk uit Slack gekopieerd om de interviews beter uit te leggen. Door de informele manier van communiceren is de grammatica en zinsbouw daardoor niet altijd correct.

Commentaar van J. van der Velde met antwoord van J. Neijts

Dat de testers het niet begrijpen? Of de tooling? Ik denk dat je het laatste bedoelt. Waarom is het erg dat de tooling de business logica niet begrijpt?

Hij bedoelde de testers.

Dit is weer een compleet losstaand punt, toch?

Niet helemaal, vanuit het testmanagement zou nagedacht moeten worden welke technieken ingezet moeten worden om welke risico's af te kunnen dekken. Leo zegt dat in het algemeen alleen de technieken worden gebruikt om de lage risico's af te dekken. Hierop zou vanuit het testmanagement meer worden gestuurd.

Hoe moeten ze daarop sturen? Ik zie het nog niet helemaal voor me…

Nu wordt volgens Leo te veel door de tester besloten welke technieken gebruikt worden zonder kennis te hebben van de risico dekking dat heeft op het project.

Woooo…. Dat klinkt bijzonder. Ook bij scrum? Ik kan mij bij waterval hier meer bij voorstellen, als je echt een aparte test-periode en ontwikkel-periode hebt. Bij scrum lijkt me hergebruik noodzakelijk

Hij bedoelde wel bij Scrum, maar hij kon er niet verder op ingaan hoe.

Hmm ik zie nog niet concreet voor me hoe je de snelheid zou kunnen testen, maar dit is ook een samenvatting natuurlijk. Kan je het concreter maken?

Hij heeft hetzelfde project gedupliceerd. 1 handmatig, 1 automatisch.

Ik ben benieuwd!

Super!

## Vragen

Onderstaande vragen zijn gesteld aan dhr. L.van der Aalst en dhr. B. Visser.

VARIANT 1: "CODE COMPLETION LIBRARY"

- In het algemeen hoe zou u hergebruik en onderhoud van de testsuites regelen?
- Kunt u tools in deze richting aanbevelen?

VARIANT 2: "CAPTURE AND REPLAY":

- Wat is uw ervaring met "capture and replay" tools?
- In welk soort project heeft u hier ervaring mee?
- Kunt u tools in deze richting aanbevelen?

VARIANT 3: ORTHOGONALE("PAIRWISE") ARRAYS:

- In uw ervaring, waar in het testproces komt deze variant het meest tot zijn recht?
- Bij wat voor projecten zou u deze dekkingsvorm aanbevelen?
- Kunt u tools in deze richting aanbevelen?

VARIANT 4:DUMB MONKEY

- Kunt u tools in deze richting aanbevelen?
- Bij wat voor projecten zou u deze dekkingsvorm aanbevelen?

VARIANT 5: BRILLIANT MONKEY

- Kunt u tools in deze richting aanbevelen?
- Hoe zou u dit binnen een scrumteam toewijzen
- Kunt u tools in deze richting aanbevelen?

ALGEMEEN

- binnen welke testvormen zou u de varianten toewijzen in een scrum omgeving?
- Hoe zou ik de criteria van deze varianten bepalen?
- Heeft u een voorkeur voor een variant?
- In uw mening welke combinatie van varianten is het sterkst, (meeste tijd opleveren, dekkingsgraad verbeteren)?

- Mist u nog een variant?
- Kent u andere experts die helpend kunnen zijn bij het onderzoek?

*Samenvatting interview B. visser*

24-4-2015

De markt voor Sogeti's Cover:

Geen automatische testgevallen.

Waren wel "model based"testtools voor Cover maar die sloegen niet aan. Dit kwam omdat testers niet onderlegd waren om met UML modellen te werken en de behoefte daar niet lag.

De "model based"testtools konden volgens Ben niet combinaties maken in de paden van "het model".

Cover toolsuite

Is een tool dat testgevallen kan genereren aan de hand van een model.

Als input gebruikt deze alle standaardformaten: xml of xpdl, dit is de output van visio of andere modelleer tools.

Aangeven welke test maat de tool hanteert. Van 1 tot 3 kan de intensiteit en daarmee het aantal testgevallen worden bepaald.

De tool kan ook "pairwise" toepassen op testgevallen.

Aan de hand van de paden die aan de hand van het model zijn bepaald kunnen combinaties worden gemaakt. Combinaties van paden geven een hogere testgraad dan het volgen van een enkel pad.

Deze tool is het meest geschikt voor testen op systeemgedeelte.

Tosca Tricentis

Tosca tricentis stappen (testgeval generatie).

Aan de hand van een voor gedefinieerd sjabloon scrapet deze tool de Attributen en velden af en benoemd deze.

Deze gebruikt CSV als input.

Tools van IBM en HP zijn wat duurder. Volgens Ben.

Eigen variant (4 en 5)

Variant 4 en 5 is een goede richting omdat bij varianten 4 en 5 techniek wordt gescheiden van het testen (black box), dit is ook de trend.

Goede opzet variant 5 maar Categorieën aanbrengen in de tool niet alles meenemen. Uit ervaring wil je niet alles meenemen in de test maar een scheiding kunnen maken.

Het lijkt Ben erg goed om bij deze tool grenswaarde mee te geven als dekkingsvorm omdat je erg met de input bezig bent. Op dat moment is het belangrijk om verschillende inputwaarden toe te kennen.

Half mij kan ik een afspraak maken als ik tijd heb en het nuttig vind om Cover testen.

# BIJLAGE C: NIET MEEGENOMEN TOOLS

De volgende tools zijn kort onderzocht maar niet meegenomen in de discussie. Per variant wordt beschreven waarom deze niet mee zijn genomen.

| Variant | Rede waarom niet meegenomen. |
|---|---|
| Variant 1 JBehave[xxx]. | Te veel overlap met huidige manier van werken. Te veel zou dubbel worden gedaan, daarom is het niet nuttig om deze tool ook te testen. |
| Variant 1 CucumberPro[xxxi] | Is een betaalde tool die meer doet dan CCL o.a. een "cloud" "repository" testmanagement functie. Momenteel wordt de testmanagement module Jira gebruikt zonder problemen. Vervanging is niet nuttig. |
| Variant 1 Intellij IDEA [xxxii] | Is een complete IDE die Eclipse zou moeten vervangen. Deze verandering is ingrijpend, ten opzichte van de toegevoegde waarde. |
| Variant 2 Ranorex[xxxiii] | Is een commerciële tool. Selenium IDE is free en werkt goed. |
| Variant 3- Hexawise[xxxiv] | Commerciële uitvoering van PictMaster. Maakt "reuse" wel makkelijk door parameters op te slaan, maar de nadruk bij "pairwise" ligt niet bij hergebruik. |

# BIJLAGE D: "PAIRWISE" MANUAL [13]

This short manual is created for giving a basic understanding of "pairwise", "When" to use it and why to use it. "pairwise" is a test technique for making pairs of parameters. You use "pairwise" if the count of the to be tested permutations is too high to test. With "pairwise" you often reduce the count of test cases dramatically.

Generally simple bugs will be found by triggering just one parameter. The next category of bugs is to be found by triggering pairs of parameters. So "pairwise" also contributes to a better test coverage. By using PictMaster for "pairwise" you have been "Given" basically a checklist for how much tests you have to run in order to achieve the maximum test coverage, so that is nice too.

In order to get a better understanding in what situation "pairwise" to use the following example is created with help from Dhr. L. Koning:

*Example 1:*

In the process: "Erkenning na geboorte" you have the following parameters(seen in figure 1).

PictMaster

| Item number | 00001 | Item name | testcase1 |
|---|---|---|---|
| Sub-item number | | Sub-item name | |

| Parameters | value hierarchy |
|---|---|
| previous children | yes, no |
| namenreeks moeder en kind | yes, no |
| naamskeuze previous children | yes, no, nvt |
| behandelen als eerste kind | yes , no |
| naamskeuze | yes, no |
| Erkenner | m,v |
| namenreeks erkenner | ja, nee |

| Constraints table | | |
|---|---|---|
| Parameters | Constraint 1 | Constraint 2 |
| previous children | no | |
| namenreeks moeder en kind | | |
| naamskeuze previous children | nvt | |
| behandelen als eerste kind | | |
| naamskeuze | | |
| Erkenner | | |
| namenreeks erkenner | | |

Figure 1 PictMaster parameters [xvi]

[13] Concept version

As you can see the parameters have several values. In order to get an understanding how much test cases you have to make and to understand combinations between them it's useful to use PictMaster. Also there is added a constraint. This is because of the situation where there a no previous children so you cannot use the values yes and no of the parameter "Naamskeuze previous children". After you add the parameters, values and constraints you can generate the test cases. This is the result (figure 2):

| | A | B | C | D | E | F | G | H |
|---|---|---|---|---|---|---|---|---|
| 1 | previous children | namenreeks moeder en kind | naamskeuze previous children | behandelen als eerste kind | naamskeuze | Erkenner | namenreeks erkenner | |
| 2 | yes | yes | no | no | no | m | ja | |
| 3 | yes | yes | yes | yes | yes | v | nee | |
| 4 | no | no | nvt | no | yes | m | nee | |
| 5 | no | no | nvt | yes | no | v | ja | |
| 6 | yes | no | no | no | yes | v | nee | |
| 7 | yes | no | yes | yes | yes | m | ja | |
| 8 | yes | yes | yes | no | no | m | nee | |
| 9 | yes | yes | nvt | no | no | v | nee | |
| 10 | yes | yes | no | yes | yes | v | ja | |
| 11 | no | yes | nvt | no | yes | m | ja | |
| 12 | | | | | | | | |

Figure 2 Result PictMaster[xvi] example 2

## Why use it?

In order to avoid said risk you do have to test some combinations. "pairwise" is a powerful method to test iBurgerzaken with because of the many forms the applications within iBurgerzaken contain. "pairwise" can test pairs of input values, but also triples and quadruples of values. By using "pairwise" you reduce the combinations of parameters to usually 1- 20%[xxxv] of the original set of combinations. According to Dolores R. Wallace and D. Richard KUHN 98%[xxxvi] of recalled medical equipment could have been prevented using "pairwise" testing. There are no best practices for "When" to use "pairwise", so I recommend you use "pairwise" regularly if important business rules affect forms in iBurgerzaken.

# BEGRIPPENLIJST

| Begrip | Afkorting | Uitleg |
|---|---|---|
| Agile[xxxvii] | | Herbergt verschillende software ontwikkelmethodes de volgende kenmerken zijn belangrijk deze ontwikkelmethode : opleveren van kleine werkende sub producten (iteraties).De communicatie is persoonlijk, zo min mogelijk documenten. |
| Apache subversion [xxxviii] | SVN | Versie controle systeem. |
| Automated regression testing | ART | Het automatisch uitvoeren van "testcases" van de regressietest. |
| Black box testing | | Een software test techniek die de input en output test van een software applicatie maar daarbij de werking van binnenuit niet meeneemt. (bron: www.redstonesoftware.com) |
| Compiling | | Omzetten van een programmeertaal. naar een machinetaal Deze is vaak op een lager niveau zo dat machines deze kunnen lezen. |
| Class | | In object georiënteerd programmeren is de class een schema die de methodes bevat van het object. |
| Cucumber [xxxix] | | Applicatie dat gebruikt wordt voor de uitvoer van "testcases". |
| Dekkingsgraad | | Zie paragraaf 4.3 testdekking |
| Dekkingsvorm | | Zie paragraaf 4.3 testdekking |
| Deploying | | Opzetten van software. |
| Domain specific language. | DSL | Een zelf ontwikkelde metataal voor specifiek voor een domein. |
| Executable | | Een bestand wat uitgevoerd kan worden door de computer. |
| Feature (file) | | Een feature is een functie die door middel van Gherkin wordt omschreven. Een feature bestaat uit 1 of meerdere scenario's binnen Gherkin. |
| Gherkin[xxxix] | | DSL van Cucumber. Deze DSL is leesbaar voor zowel mens als machine. |
| Grenswaarde | | Een dekkingsvorm waarbij de testdata op of over de grens is van de parameter. Bijvoorbeeld als de business rule voorschrijft dat een data niet kleiner mag zijn dan 01-01- 2015 dan kun je de waarde groter dan, kleiner dan, gelijk of een belachelijke waarde invullen. |
| JAR | | Java Archive, een datacompressie standaard. |
| Jenkins[xl] | | Tool die bouwen en testen van software makkelijker maakt door het automatiseren van kleine taken. |
| Kwaliteitsattributen[xli] | | Beheerbaarheid, beveiliging, bruikbaarheid, connectiviteit, continuïteit, controleerbaarheid, flexibiliteit, functionaliteit, gebruiksvriendelijkheid, infrastructuur, performance, portabiliteit, testbaarheid, zuinigheid. |
| Library | | Gedragsstappen die zijn opgeslagen in een bestand die aan te |

| | | |
|---|---|---|
| | | roepen zijn door de applicatie. |
| Maven[xlii] | | Tool die projecten opbouwt met behulp van POM. |
| POM | | Hierin wordt de Maven projectinformatie opgenomen. |
| "proof of concept" | POC | Bewijs van de werking door een demonstratie. |
| Regressie testen | | Het testen van een bestaande applicatie waarbij gecontroleerd wordt of de ongewijzigde "content" onveranderd is gebleven na maken van wijzigingen. |
| Repository | | Opslag van (Meta) data. |
| Risico | | Kans x bedreiging |
| Ruby[xliii] | | Programmeertaal |
| Scenario | | Een scenario geeft de interactie weer van de gebruiker en het systeem. Maar een scenario is ook een stap binnen Gherkin wat een gedrag van het systeem weergeeft. |
| Scrum | | Een software ontwikkelmethode die valt onder agile. Belangrijke kenmerken van scrum zijn: sprints (een periode waarin iteraties worden gedaan), dagelijkse meetings over voorgang. |
| Selenium webdriver | | Een applicatie die stuurt en informatie ophaalt van een webbrowser. |
| Step definition | | Binnen Cucumber wordt een stap gekoppeld aan JAVA of Ruby code, deze code is de step definition. |
| Systeem test | | Bij deze test wordt de werking getest van de nieuwe componenten(clusters) in de totale omgeving. Hierbij worden alleen de nieuwe functionaliteiten meegenomen. |
| Test suite | | Een verzameling "testcases" |
| Testcase/testgeval/scenario | | Een test waarmee de werking van een software applicatie of deel daarvan, wordt getoetst. |
| Testdekking | | Het totale verwachte gedrag wat getest wordt met specifieke dekkingsvorm(en). Dit is geen absoluut getal, maar een benadering. Dit komt doordat het totale verwachte gedrag niet meetbaar is. |
| Unit test | | Is een test van een klein deel van de code. En test de werking losstaand van de rest van de applicatie. |
| User story | | Beschrijft in een paar zinnen het te implementeren gedrag (bij "behavior driven development"). |

# BIBLIOGRAFIE


[i] (Koomen, Aalst van der, Broekman, & Vroon, Tmap Next voor resultaat gericht testen, 2006, pp. 48-51)
[ii] (Williams, A (partial) introduction to software engineering practices, 2010, p. 41)
[iii] (Redstonesoftware, Black-box vs. White-box Testing:Choosing the Right Approach to Deliver Quality Applications, 2008)
[iv] (Williams, A (Partial) Introduction to Software Engineering Practices, 2010, pp. 43,44)
[v] (Koomen, Aalst van der, Broekman, & Vroon, Tmap Next voor resultaat gericht testen, 2006, p. 51)
[vi] (Neijts, Adviesdocument Project automatische testcase generatie, 2015)
[vii] (Bertolino, Antonia; Li, Jenny; Zhu, Hong, An Orchestrated Survey on Automated Software Test Case Generation, 2013, pp. 6,7)
[viii] (Test sets generation from algebraic specifications using logic programming, 1986)
[ix] (Méganique Statistique et Irréversibilité, 1913)
[x] (Arnold, Thomas, Visual test bible, 1998)
[xi] (Koomen, Aalst van der, Broekman, & Vroon, Tmap Next voor resultaat gericht testen, 2006, pp. 595-600)
[xii] (Koomen, Aalst van der, Broekman, & Vroon, Tmap Next voor resultaat gericht testen, 2006, p. 596)
[xiii] (https://www.jetbrains.com/ruby/, 2015)
[xiv] (https://eclipse.org/, 2015), (https://cucumber.io/, 2015)
[xv] (http://www.seleniumhq.org/projects/ide/, 2015)
[xvi] (PictMaster, 2012)
[xvii] (http://jsoup.org/, 2015)
[xviii] (http://www.seleniumhq.org/projects/webdriver/, 2015)
[xix] (Sogeti, Cover, 2015)
[xx] (Koomen, Aalst van der, Broekman, & Vroon, Tmap Next voor resultaat gericht testen, 2006, pp. 608-609)
[xxi] (Softwaretestinggenius, 2015)
[xxii] (Wiley, John; Myers; Glenford, J,The Art of Software Testing, 1979)
[xxiii] (Martin, James, Managing the Data-base Environment, 1983)
[xxiv] (Nooku, BREAD.html, 2015)
[xxv] (Istqbexamcertification, what-is-equivalence-partitioning-in-software-testing/, 2015)
[xxvi] (Istqbexamcertification, what-is-error-guessing-in-software-testing/, 2015)
[xxvii] (Tutorialspoint, use_interface_testing.htm)
[xxviii] (Koomen, Aalst van der, Broekman, & Vroon, Tmap Next voor resultaat gericht testen, 2006, p. 609)
[xxix] (Istqbexamcertification, what-is-state-transition-testing-in-software-testing)
[xxx] (http://jbehave.org/, 2015)
[xxxi] (https://cucumber.io/pro, 2015)
[xxxii] (https://www.jetbrains.com/idea/, 2015)
[xxxiii] (http://www.ranorex.com/, 2015)
[xxxiv] (https://hexawise.com/, 2015)
[xxxv] (Softwarecentral, http://www.slideshare.net/Softwarecentral/"pairwise"-testing, 2010)
[xxxvi] (Walles R, Doleres; Kuhn, Richard, FAILURE MODES IN MEDICAL DEVICE SOFTWARE, 2001)

[xxxvii] (Agilealliance, http://www.agilealliance.org/, 2015)
[xxxviii] (Apache Subversion, 2000)
[xxxix] (https://cucumber.io/, 2015)
[xl] (https://jenkins-ci.org/, 2015)
[xli] (Koomen, Aalst van der, Broekman, & Vroon, Tmap Next voor resultaat gericht testen, 2006, pp. 503-509)
[xlii] (https://maven.apache.org/, 2015)
[xliii] (https://www.ruby-lang.org/en/, 2015)