# Testing at OnGuard

Invoeren van gestructureerde testmethodes in een bestaand software ontwikkelproces

Ing. R.J.C. Backus

Eenjarige Master Software Engineering

| | |
|---|---|
| Afstudeerdocent: | Prof. Dr. Jan van Eijck |
| Stagebegeleider: | Ir. Bram van den Abeele |
| Opdrachtgever: | OnGuard Nederland B.V. |

Publicatiestatus: Openbaar (V1.1)

Universiteit van Amsterdam

# Adressenlijst

## *Afstudeerder / Stagiair*

Ing. R.J.C. Backus
Gedempte Voldersgracht 6a
2011 WD Haarlem
Tel: +31(0)6-54700642
E-mail: i-meel@home.nl

## *Stageverlenend bedrijf*

OnGuard Nederland B.V.
Kasteel Nederhorst
Slotlaan 3, Postbus 125
1394 ZJ Nederhorst den Berg
Tel: +31(0)294-256666
Fax: +31(0)294-239420

## *Stagebegeleider*

Ir. B. van den Abeele
Tel: +31(0)294-256666
E-mail: bva@OnGuard.nl

## *Opleidingsbegeleider*

Prof. Dr. J. van Eijck
CWI - Centrum voor Wiskunde en Informatica
Kruislaan 413, Postbus 94079
1090 GB Amsterdam
Tel: +31(0)20-5924052
E-mail: jve@cwi.nl

# Samenvatting

In deze scriptie komt een onderzoek aan bod, uitgevoerd bij OnGuard Nederland B.V., verder te noemen OnGuard, te Nederhorst den Berg. Dit onderzoek is gehouden in het kader van een proeve van bekwaamheid voor het afronden van de Master opleiding Software Engineering aan de Universiteit van Amsterdam.

De initiële onderzoeksvraag horende bij dit onderzoek is als volgt:
*"Hoe kan met behulp van bruikbare testtechnieken de testinspanning gemeten in testpersoon-uren verlaagd en de kwaliteit van het product op een hoog pijl gehouden worden?"*

Voordat er veranderingen binnen het testproces aangebracht konden worden moest er eerst in kaart worden gebracht hoe het testproces van OnGuard er precies uit ziet. Zo is er gekeken naar welke activiteiten en afdelingen bij dit testproces betrokken waren. Hier is een workflowdiagram van opgesteld. Dit diagram is gemaakt op basis van aanwezige procedures en is verder gecompleteerd en gevalideerd door interviews met betrokken medewerkers.

Naast het testproces op zich is ook de visie van betrokken medewerkers met betrekking tot testen belangrijk. Om deze visie in kaart te brengen is een aantal interviews met programmeurs en de senior tester van OnGuard gehouden. Hieruit is gebleken dat programmeurs testen niet als hun taak zien. Ze vinden het "tijdrovend" en zelfs "verschrikkelijk". Verder hebben ze geen inzicht in de taken en verantwoordelijkheden van de tester.

Uit analyse van de workflow van het testproces in combinatie met de gehouden interviews is gebleken dat er een aantal knelpunten binnen het testproces op te noemen was. Dit aantal knelpunten, in combinatie met de manier van testen, bleek onvoldoende om op effectieve wijze individuele testactiviteiten te optimaliseren door bijvoorbeeld testautomatisering. Het testproces van OnGuard ontbrak een gestructureerde manier van werken en een eenduidige uitvoering van testen. Hierdoor heeft er een wending in het onderzoek plaats gevonden.

De onderzoeksvraag is hierop aangepast naar:
*"Kan met behulp van bruikbare testtechnieken het testproces inzichtelijk en beheersbaar gemaakt worden en daarmee efficiënter uitvoerbaar zijn?"*

Met behulp van een referentiemodel is voor het testproces de volwassenheid bepaald. Als referentiemodel is het Test Process Improvement (TPI) model gekozen. Op basis van een uitgevoerd assessment is de volwassenheid, het TPI level, van het huidige testproces vastgesteld.

Nadat bekend was geworden wat het TPI level van het testproces van OnGuard was, is er begonnen met het opstellen van een planning voor het inzichtelijk en beheersbaar maken van het testproces. Er is gekozen voor een planning voor de korte en een planning voor de lange termijn. Dit omdat de afstudeerder op de korte termijn een actieve rol in begeleiding van de medewerkers en implementatie van de veranderingen aan het testproces zou kunnen spelen. Voor de lange termijn is er een plan gemaakt met daarbij aandachtspunten voor de uitvoering van dit plan.

Hierdoor kan OnGuard op eigen kracht verder gaan met het verbeteren van het testproces na afronding van het onderzoek. De TPI methode geeft duidelijk aan wat er nodig is om een volgend level te bereiken en wordt verder in het onderzoek gebruikt om het testproces te structureren en te verbeteren.

Het aanbrengen van structuur is voor de korte termijn het belangrijkste geweest. Structuur in het testproces maakt het proces inzichtelijk en beheersbaar. Verder is er gekozen om naast het aanbrengen van structuur ook te proberen het testproces efficiënter te maken. Deze efficiëntie is gemeten in zogenaamde testdruk. De testdruk is uitgedrukt in het aantal dagen wat de tester over heeft, als het een negatieve waarde betreft, of tekort heeft, als het een positieve waarde betreft. Hoe hoger de testdruk hoe minder tijd de tester over heeft en hoe meer de te testen producten zich ophopen op de testafdeling. Voor de lange termijn is gekozen voor het volledig onder controle krijgen van het testproces.

Of de aangebrachte veranderingen het gewenste effect hebben bereikt is nog niet bekend. De voorgestelde veranderingen zijn nog niet in gebruik binnen OnGuard dus er zijn nog geen meetgegevens beschikbaar. Validatie van de resultaten zal enkele maanden na in gebruik name van de veranderingen plaats kunnen vinden. Voor deze validatie is een duidelijk handvat gegeven in de vorm van de eerder kort beschreven testdruk. Hiermee kan per testmoment de testdruk berekend worden.

Wat wel gevalideerd kon worden was de verandering in de visie met betrekking tot testen bij de medewerkers van OnGuard. Vooral de programmeurs hebben meer inzicht gekregen in en respect gekregen voor het vakgebied van testen. Men ziet testen niet langer als “verschrikkelijk” maar als “noodzakelijk”. Vrijwel alle medewerkers van OnGuard hebben duidelijk vooruitgang op het gebied van testen geboekt, ook en vooral de senior tester.

# Voorwoord

De afstudeerperiode bij OnGuard is een ontzettend prettige geweest. Ondanks dat ik er elke dag een flinke reis voor moest maken om daar aan het werk te gaan ben ik altijd fris, opgewekt en met ontzettend veel plezier vanuit Haarlem naar het pittoreske Nederhorst den Berg gekomen. Als geboren Maastrichtenaar miste ik de heuvels en bossen van Zuid-Limburg. De locatie van OnGuard, een groot kasteel met daarbij de nodige bomen en planten hebben mij aldaar thuis doen voelen.

Binnen OnGuard heerst, ondanks de status die een kasteel met zich meebrengt, een prettige en informele sfeer. Binnen OnGuard heeft iedereen respect voor elkaar en is er een gevoel van eenheid. Hoewel er naar buiten toe een duidelijke hiërarchie binnen de organisatie gevoerd werd was in de omgang met de mensen niet te merken waar in zij zich in de organisatiestructuur bevonden.

Bij de uitvoering van mijn onderzoek heb ik alle vrijheid gekregen. Hoewel ik een soort van expert rol vervuld heb op het gebied van testen en het testproces heb ik nooit het gevoel gehad dat ik er alleen voor stond. De actieve en vooral persoonlijke begeleiding in de persoon van Bram van den Abeele is door mij als ontzettend prettig ervaren. Bij dezen wil ik Bram dan ook hartelijk bedanken voor de goede begeleiding en steun tijdens mijn afstuderen. Gedurende de maanden dat wij intensief samengewerkt hebben is hij naast een begeleider ook een goede vriend geworden.

Daarnaast wil ik mijn dank uitspreken naar alle medewerkers van OnGuard, het management team in het bijzonder. Jullie hebben mij echt op mijn gemak doen voelen en samen hebben we ook menig keer hartelijk om de meest uiteenlopende dingen kunnen lachen. Ik wil jullie bedanken voor de steun en het vertrouwen dat jullie mij gegeven hebben.

Ik ben er trots op deze opleiding te kunnen afronden en zie dit ook als een duidelijke mijlpaal in mijn leven. Met een glimlach op het gezicht zal ik terugdenken aan zowel mijn tijd op het kasteel van OnGuard als aan mijn tijd op de universiteit en omliggende kroegen.

Het afgelopen jaar is een jaar geweest van vallen en opstaan. Ik heb ontzettend veel geleerd, niet alleen op het gebied van pure kennis maar vooral op het gebied van gestructureerd werken en denken. Er is ontzettend hard gewerkt maar vooral ook veel gelachen. Er zijn momenten geweest dat ik me afgevraagd heb of ik het wel zou redden. Maar de actieve begeleiding van de docenten op de Master Software Engineering aan de Universiteit van Amsterdam heeft mij in mezelf doen geloven en mij het vertouwen en de kracht gegeven om deze zware maar ontzettend leerzame opleiding te kunnen afronden. De hartelijkheid, openheid en vooral humor van deze docenten heeft naar mijn mening ieders hart gestolen waardoor ik hen niet alleen als docenten zie, maar vooral als vrienden.

# Inhoudsopgave

# Lijst van Figuren en Tabellen

**Figuren**



**Tabellen**

# 1. Inleiding, Onderzoeksvraag en Hypothesen

Dit onderzoek richt zich op kleine tot middelgrote organisaties in het ICT werkvlak die:

- software produceren
- het belang van kwaliteit inzien
- functionele en/of technische eisen niet, weinig of niet volledig en niet compleet documenteren
- een verhouding tussen testers en developers hebben die kleiner is dan 1:3
- zich het niet kunnen veroorloven hun ontwikkelproces tijdelijk stil te leggen
- een aparte testafdeling hebben of de specifieke rol van tester voeren
- het testproces alleen binnen de testafdeling laat reiken
- inzien dat hun testproces te veel tijd en geld kost
- weinig inzicht hebben in de activiteiten en planning van het testproces

En misschien nog wel het belangrijkste: De organisatie moet het testproces aan willen pakken! Ook organisaties die beginnen met het opzetten van een testproces kunnen hun voordeel doen met dit onderzoek. Drempels, valkuilen en aandachtspunten voor een gestructureerd testproces komen aan de orde.

Testen is bij veel bedrijven tegenwoordig een hot item. Klanten hebben geleerd van de economische malaise en geven niet zo snel meer bakken met geld uit aan ICT-oplossingen die hen de wereld beloven. Kwaliteit is het kernwoord, kwaliteit is waar men naar kijkt. Het testproces is hét gereedschap om die o zo belangrijke kwaliteit te waarborgen en te garanderen.

Testen is een vakgebied wat de afgelopen jaren flink heeft gewonnen aan volwassenheid. Waar het voorheen alleen gebruikt werd om aan te tonen dat een programma doet wat het moet doen, wordt het heden ten dage voor veel meer doeleinden gebruikt. Testen is de basis voor kwaliteit geworden. Niet alleen van de software, maar ook van het softwareontwikkelproces op zich en ondersteunende processen als helpdesk en release management.

Om als organisatie kwaliteit te kunnen garanderen is het belangrijk om een uitgedacht testproces te hebben. Het ongestructureerd uitvoeren van testen kan sporadisch producten opleveren van goede kwaliteit. Om te allen tijde kwaliteit te kunnen garanderen zal het testproces toch op een gestructureerde manier uitgevoerd moeten worden. Hierbij wordt niet alleen gekeken naar de kwaliteit van de te testen software, maar ook naar de kwaliteit van het testproces op zich. Om wijzigingen aan te brengen aan het testproces moet het testproces immers eerst op een eenduidige manier uitgevoerd worden.

Sinds het barsten van de technologische zeepbel en de economische malaise sinds het begin van de 21e eeuw houden ICT-behoevende bedrijven de portemonnee stijf dicht. Bedrijven willen geen risico meer lopen. Een applicatie wordt dan ook niet meer beoordeeld op de schat aan functionaliteit die deze kan bieden. Kwaliteit is het kernwoord.
Kwaliteit wordt immers niet bepaald door het eigen beeld van de applicatie maar door de perceptie van de klant.

De concrete onderzoeksvraag voor dit onderzoek is initieel als volgt opgesteld:
“Hoe kan met behulp van bruikbare testtechnieken de testinspanning gemeten in testpersoon-uren verlaagd en de kwaliteit van het product op een hoog pijl gehouden worden?”
Door nieuw verworven inzichten gedurende de uitvoering van het onderzoek heeft er een wending in de uitvoering plaatsgevonden. Deze wending heeft tot gevolg gehad dat de onderzoeksvraag aangepast moest worden. De aangepaste onderzoeksvraag luidt:
“Kan met behulp van bruikbare testtechnieken het testproces inzichtelijk en beheersbaar gemaakt worden en daarmee efficiënter uitvoerbaar zijn?”

Bij de initiële onderzoeksvraag zijn de volgende hypothesen opgesteld:

- Het aanbrengen van structuur is de basis voor een stabiel, effectief en efficiënt testproces.
- Het verbeteren van het testproces heeft niet alleen betrekking op de testafdeling. De testafdeling is in grote mate afhankelijk van omringende afdelingen.
- Automatisering van testen verlaagt de testinspanning
- Geautomatiseerd testen is een alleen effectief binnen een gestructureerd testproces.
- De verschillende onderdelen van het testproces kunnen alleen verbeterd worden wanneer het testproces op een structurele en eenduidige manier uitgevoerd wordt.

# 2. Context en Achtergrond

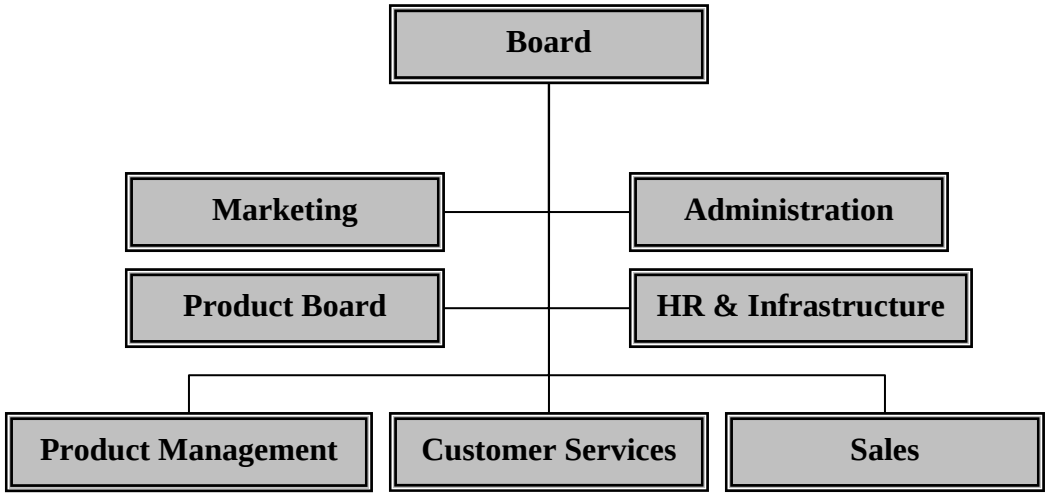
Bepalend voor het uitvoeren van een onderzoek is nog altijd de context waarin dit onderzoek plaats vindt. Belangrijk voor dit onderzoek is dat de organisatie waar het onderzoek uitgevoerd wordt grotendeels voldoet aan de in het vorige hoofdstuk gestelde kenmerken.
In de volgende paragraaf betreffende de context van het onderzoek zal de organisatie worden voorgesteld waar het onderzoek uitgevoerd is. Ook wordt deze organisatie in deze paragraaf geanalyseerd op geschiktheid voor het onderzoek. In paragraaf 2.1 wordt de achtergrond beschreven. Hier worden enkele technieken, methoden en principes uitgelegd die nodig zijn voor het verdere begrip en inzicht in het onderzoek.

## *2.1 Context*

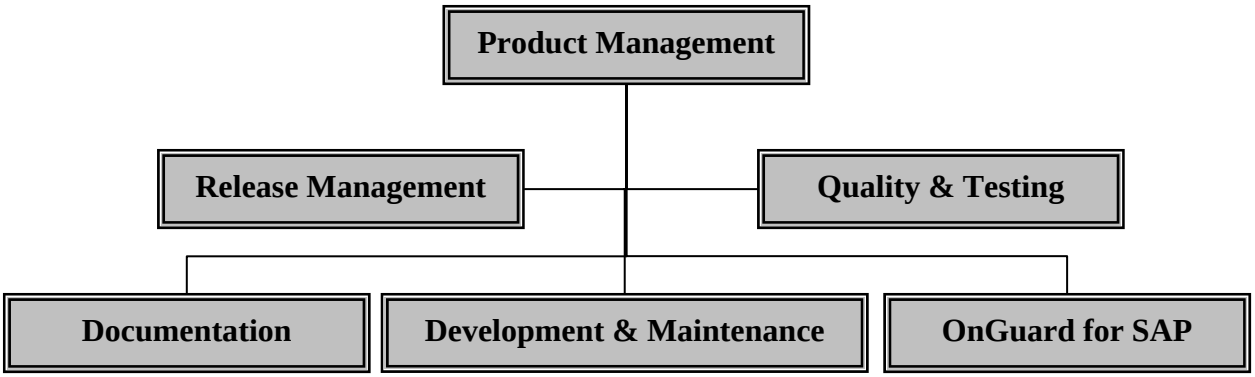
In deze paragraaf komen de organisatie en haar producten aan de orde. Verder zal deze paragraaf inzicht bieden op de vraagstelling of de organisatie OnGuard Nederland B.V., verder te noemen OnGuard, exemplarisch is voor dit onderzoek.

### 2.1.1 Organisatie

Het onderzoek is uitgevoerd bij de organisatie OnGuard te Nederhorst den Berg. Als onafhankelijk Nederlands softwarebedrijf biedt OnGuard het bedrijfsleven totaaloplossingen voor credit- en collections management; van ontwikkeling en implementatie tot aan service en training. Het bedrijf heeft een veertigtal medewerkers, variërend van afdelingen marketing en sales tot aan software ontwikkeling. De structuur van de organisatie is weergegeven in figuren 1 en 2. De afstudeerder is gedurende het onderzoek werkzaam op de afdeling Quality & Testing.




**Figuur 1: Organigram van OnGuard Nederland B.V.**




**Figuur 2: Organigram van afdeling Product Management, OnGuard Nederland B.V.**

OnGuard is op het moment van schrijven in Nederland onbetwist marktleider in haar werkveld en streeft er naar deze positie vast te houden. Het is de missie van OnGuard om de meest belangrijke leverancier op het gebied van credit- en collectionsmanagment te worden, zowel op basis van de totale waarde van uitgegeven licenties als op grootte van het klantenbestand. OnGuard wil dit doel bereiken door middel van een combinatie van actieve verkoop kracht en nauwe samenwerkingsverbanden met professionele partners. Op deze manier wil men de klant een combinatie van standaard software en extra services met toegevoegde waarde kunnen bieden. Dit komt tot uitdrukking in een te implementeren oplossing van hoge kwaliteit en draagt zorg voor het efficiënt en financieel voordelig voeren van een credit- en collectionsbeleid. De productontwikkelingsstrategie wordt gedreven door end user requirements en levert een end user oplossing voor credit- en collectionsmanagement waarbij gebruik gemaakt wordt van bewezen technologie als middel om deze oplossing te bewerkstelligen.
Meer over OnGuard is terug te vinden op http://www.OnGuard.nl.

### 2.1.2 Analyse van de organisatie voor geschiktheid binnen het onderzoek

Om de geschiktheid van OnGuard voor dit onderzoek te bepalen is een analyse uitgevoerd. Met behulp van een aantal documenten en informatie verkregen uit interviews met medewerkers van verschillende afdelingen is in kaart gebracht in hoeverre deze organisatie zich leent voor dit onderzoek.
De bij deze analyse betrokken documenten zijn:

- Algemene bedrijfsinformatie
- Een mission statement dat de intenties van de organisatie weergeeft voor de termijn van minstens twee jaar
- Procedures van verschillende afdelingen met daarin afspraken over hoe er gewerkt wordt en wat de relaties zijn met andere afdelingen
- Workflow van het testproces

Verder is er door middel van interviews achterhaald in hoeverre de bovengenoemde documenten correct en volledig zijn. Indien één of meerdere documenten niet aanwezig zijn is met behulp van interviews en de wél aanwezige documenten getracht de ontbrekende documenten aan te vullen. Ook worden deze interviews gebruikt om extra en meer gedetailleerde informatie naar boven te brengen en om eventuele onduidelijkheden helder te krijgen. De resultaten van deze analyse zullen in tabelvorm worden gepresenteerd.

### 2.1.3 Geschiktheid van OnGuard Nederland B.V.

De resultaten van de in paragraaf 2.1.2 genoemde analyse van OnGuard voor geschiktheid binnen het onderzoek zijn in tabel 1 opgenomen. In deze tabel wordt per eis vermeld of OnGuard er aan voldoet en uit welke informatie dit resultaat is afgeleid.

**Tabel 1: Geschiktheidanalyse van OnGuard Nederland B.V.**

| Kenmerk / Eis | Ok? | Bron | Opmerkingen |
|---|---|---|---|
| klein tot middelgrote organisatie | **JA** | - Algemene bedrijfsinformatie | 50 medewerkers |
| software produceren | **JA** | - Algemene bedrijfsinformatie | |
| het belang van kwaliteit inzien | **JA** | - Mission statement<br>- Interviews | |
| Functionele en/of technische eisen niet, weinig of niet volledig en niet compleet documenteren | **JA** | - Interviews<br>- Procedures | Uit interviews met de manager Program Management en programmeurs blijkt dat functionele eisen, indien ze worden vastgelegd, alleen globaal worden opgesteld. Technische specificaties worden niet of nauwelijks gemaakt. De procedures van deze afdeling ondersteunen deze resultaten. Hoewel de functionaliteit die OnGuard bouwt een directe vertaling van de functionele wensen van de eindgebruikers is, is de manier van het vastleggen van deze eisen voor verbetering vatbaar. |
| Een verhouding tussen testers en developers hebben die kleiner is dan 1:3 | **JA** | - Interviews<br>- Algemene bedrijfsinformatie | 1:5<br>Er is één tester ten opzichte van vijf programmeurs ten tijde van de analyse |
| zich het niet kunnen veroorloven hun ontwikkelproces tijdelijk stil te leggen | **JA** | - Algemene bedrijfsinformatie<br>- Procedures | Vastgestelde data voor uitlevering van software. Harde deadlines. Vastgesteld aantal uitleveringen per jaar. |
| Aparte testafdeling of het voeren van de specifieke rol als tester | **JA** | - Algemene bedrijfsinformatie<br>- Interviews | In figuur 2 is te zien dat de afdeling “Quality & Testing” een aparte afdeling is. |
| Het testproces alleen binnen de testafdeling laat reiken | **JA** | - Interviews<br>- Procedures<br>- Workflow | |

| | | | |
|---|---|---|---|
| inzien dat hun testproces te veel tijd en geld kost | **JA** | - Interviews | Uit interviews met de manager van de afdeling Product Management en de manager Testing blijkt dat er een gevoel heerst dat het testproces teveel middelen in beslag neemt. |
| weinig inzicht hebben in de activiteiten en planning van het testproces | **JA** | - Interviews | Uit interviews met de manager van de afdeling Product Management en programmeurs van deze afdeling blijkt dat deze mensen onvoldoende inzicht hebben in de activiteiten die uitgevoerd worden en zouden moeten worden binnen het testproces. |
| De organisatie moet het testproces aan willen pakken | **JA** | - Interviews | |

### 2.1.4 Software van OnGuard Nederland B.V.

De software die OnGuard maakt voor het voeren van credit- en collections management bestaat uit een aantal subsystemen ofwel modules. Deze modules zijn:

**Tabel 2: Modules van het softwarepakket van OnGuard**

| |
|---|
| **Import** |
| Deze module importeert gegevens met betrekking tot debiteuren en facturen uit de financiële administratie die bij de klant gebruikt wordt en plaatst deze in de interne database die gebruikt wordt door de OnGuard software. |
| **Applicatie Beheer** |
| Zoals de naam van deze module aangeeft is deze module verantwoordelijk voor het beheer van de OnGuard applicatie. In deze module wordt onder andere een administratie gemaakt, databaseverbinding vastgelegd en de importlocaties gespecificeerd. Ook instellingen met betrekking tot valuta, aanmaantaal en profielen worden in deze module onderhouden. |
| **OnGuard Client** |
| De OnGuard Client is de software die de eindgebruiker voor ogen krijgt. Hier wordt het echte credit- en collections management bedreven. |
| **Batch** |
| De Batch module stelt de gebruiker in staat om brieven die verstuurd moeten worden in een batch te zetten. Hiermee kan ervoor gezorgd worden dat deze brieven in een bulkverwerking afgedrukt worden of per e-mail verzonden worden. |

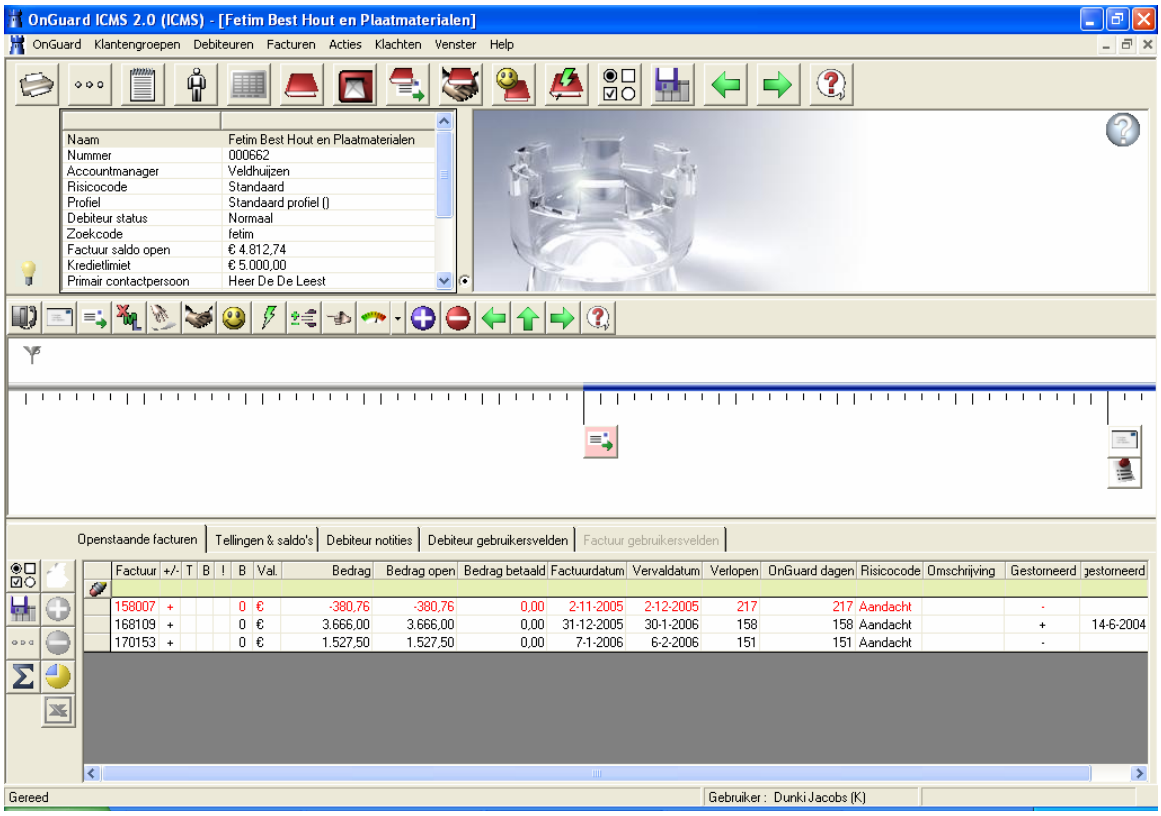
Met uitzondering van de import module zijn alle modules geschreven in Visual Basic 6 met gebruik van de Visual Studio omgeving. De import module is geschreven in VB.NET.
De programmatuur van OnGuard kan voor interne verwerking en opslag overweg met zowel SQL Server als met Oracle databases. Met deze databases wordt zowel historie als debiteuren- en factureninformatie opgeslagen.

Uit interviews met programmeurs binnen OnGuard is gebleken dat de software vrij gevoelig is voor bugs. Het is een complex systeem met veel interne samenhang wat er toe leidt dat het niet

altijd duidelijk is of het oplossen van een probleem geen ander probleem op een andere plaats in de software veroorzaakt.

Wat de software van OnGuard verder complex maakt is de genericiteit die nodig is om de klanten te bedienen. Geen enkele klant is hetzelfde maar men wil toch met één pakket al deze klanten tevreden stellen. Soms is het zelfs nodig om een speciale versie te maken voor een specifieke klant. Belangrijk is dat klanten in grote mate afhankelijk zijn van de software van OnGuard. Het niet functioneren van de software kan voor grote klanten een fikse schadepost zijn. Het is daarom van belang dat de software van hoge kwaliteit is en er kordaat wordt gereageerd wanneer een klant door een fout in de software deze software niet kan gebruiken.

Een voorbeeld van het zogenaamde “Behandelscherm” uit de OnGuard Client module wordt getoond in figuur 3. Vanuit dit scherm kunnen er acties worden ondernomen per factuur op debiteur niveau. Er wordt een tijdlijn getoond met acties die na vervaldatum van de factuur worden uitgevoerd. In het midden van deze tijdlijn is in dit geval een zogenaamde brief-actie, het sturen van een brief aan de in gebreken zijnde debiteur, te zien. Alle openstaande facturen behorende bij de geselecteerde debiteur worden onderin het scherm getoond.




**Figuur 3: Behandelscherm van de OnGuard Client module**

## 2.2 Achtergrond

### 2.2.1 Waarom testen?

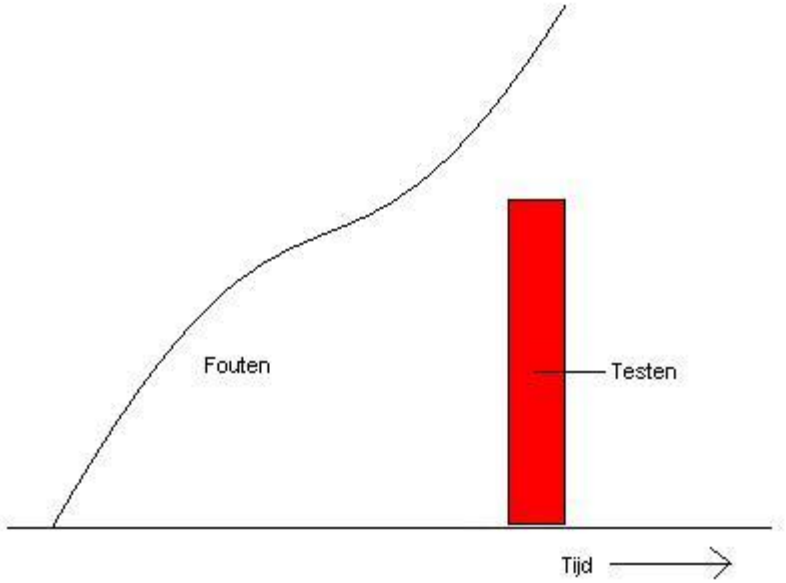
Fouten zorgen ervoor dat het vertrouwen benodigd voor effectieve softwareontwikkeling geschaad wordt [Bec99]. Beck stelt verder dat klanten vertrouwen moeten kunnen hebben in de software. Maar ook dat managers vertrouwen moeten kunnen hebben in voortgangsrapportages en programmeurs vertrouwen moeten hebben in elkaar [Bec99].
Het is onmogelijk om alle fouten op te lossen of te voorkomen. Het probleem is dat zowel het voorkomen als het oplossen van fouten veel geld kost [Bec99]. In veel gevallen is het oplossen van fouten duurder dan het voorkomen ervan [Bec99]. Dit komt omdat het oplossen van fouten niet alleen geld kost voor het maken van de oplossing, er zijn namelijk ook indirecte kosten. Op het moment dat een fout ontdekt wordt kost het indirect geld doordat het vertrouwen binnen relaties geschaad kan zijn, business verloren is en ontwikkeltijd gespendeerd is aan het oplossen van de fout in plaats van het doorontwikkelen van de software [Bec99].
Testen geeft inzicht in en advies over de risico's ten aanzien van de kwaliteit van het geteste systeem [Pol05]. Fouten worden in een vroeg stadium gevonden [Pol05]. Fouten worden voorkomen [Pol05].
**Hoe staat het bij OnGuard?**
Bij OnGuard wordt er getest om fouten in de software te vinden. Een andere reden is er bij aanvang van het onderzoek niet gegeven. Men ziet het belang in van het opleveren van software van hoge kwaliteit maar heeft weinig middelen om die kwaliteit te garanderen en te controleren.

### 2.2.2 Pair Programming

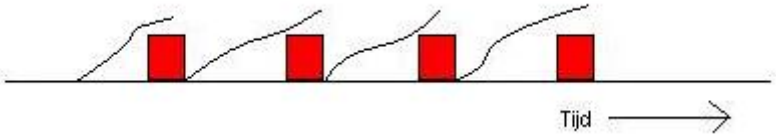
Als het gehele ontwikkelproces in paren van twee personen doorlopen wordt spreken we van Pair Programming. Hierbij wordt dus de analyse, het ontwerp, de implementatie en het testen in paarvorm gedaan [Wil99][Wil00].
**Hoe staat het bij OnGuard?**
Binnen OnGuard wordt volgens het document Procedure Software Ontwikkeling niet op deze manier gewerkt [OnG05a]. Er is verschil tussen de Pair Programming methode en de bij OnGuard gehanteerde methode.
Waar bij de Pair Programming methode de beide ontwikkelaars parallel het gehele ontwikkeltraject samen doorlopen, wordt bij OnGuard een soort sequentieel proces gebruikt. In plaats van de taken samen uit te voeren worden de taken verdeeld onder twee programmeurs. De eerste programmeur (Prog1) neemt de melding van een bug / nieuwe feature door. Hierna zoekt hij het betreffende stuk code op en stelt een oplossing voor. De tweede programmeur (Prog2) neemt de door Prog1 voorgestelde oplossing door en beoordeeld deze oplossing op haalbaarheid, doeltreffendheid en correctheid. Hierna begint Prog2 aan het programmeren van de oplossing. Wanneer Prog2 de oplossing geïmplementeerd heeft en er van overtuigd is, door testen, dat alles naar wens werkt, wordt de code overgedragen aan Prog1. Prog1 test de code van Prog2 opnieuw. Als Prog1 het eens is met de gemaakte oplossing wordt aangegeven dat de code gereed is voor verder testen en kunnen beide programmeurs andere werkzaamheden uit gaan oefenen.
Waar het bij het gebruik van Pair Programming vaak voor komt dat een vast koppel van ontwikkelaars samenwerkt [Wil99], staan de paren binnen de methode van OnGuard niet vast [OnG05a].

### 2.2.3 Het moment van testen

Waarom zouden we software aan testen onderwerpen voordat er überhaupt een product opgeleverd wordt? Boehm stelt dat hoe eerder een fout in het systeem ontdekt wordt, des te minder impact deze heeft fout heeft wanneer deze opgelost moet worden [Boe76]. Deze stelling geldt niet alleen voor de arbeid maar ook voor het geld dat er in moet worden gestoken en wordt ondersteund door zowel Beck als McConnell [Bec99][McC04]. De fout kan op verschillende vlakken in het ontwikkelproces gemaakt worden, van een fout in de software requirements en de software architectuur tot aan een programmeerfout. Wat er feitelijk gezegd word is dat hoe langer een fout onopgemerkt blijft, des te duurder en lastiger het wordt om de fout op te lossen. Het is daarom van belang om fouten zo snel mogelijk aan het licht te brengen. Testing speelt hierin een hoofdrol. Hoe eerder er getest wordt, des te eerder fouten aan de oppervlakte komen.
Verder is het ook belangrijk om vooral regelmatig te testen [Bec99]. Het regelmatig testen zorgt voor een afname van het aantal fouten en lagere testkosten [Bec99]. Dit wordt geïllustreerd in onderstaande afbeeldingen:




**Figuur 4: Laat en duur testen laat veel fouten onberoerd [Bec99]**




**Figuur 5: Regelmatig testen vermindert kosten en fouten [Bec99]**

Niet alleen het management is gebaat bij regelmatig testen. Ook ontwikkelaars zelf hebben voordeel aan minder fouten [Bec99]. Dit komt omdat er een verband bestaat tussen het aantal fouten in de software en het moreel van de groep van ontwikkelaars. Een gevonden fout leidt namelijk tot extra werk en in het ergste geval stress [Bec99].
Hoe minder fouten, des te meer vertrouwen de ontwikkelaars in het systeem en in elkaar krijgen.

**Hoe staat het bij OnGuard?**
Het testproces van OnGuard kent een aantal testmomenten. Helaas vallen deze momenten allemaal laat in het ontwikkelproces. De last van het testen wordt niet over de tijd verdeeld en komt nagenoeg overeen met de in figuur 1 getoonde grafiek. In paragraaf 4.2.3 staan de resultaten met betrekking tot hoe vaak en wanneer er getest wordt.

### 2.2.4 Unit Testing

Een unit test is een procedure die gebruikt wordt om een specifieke module van broncode te testen [WiPe-1w]. Het idee achter de unit test is zorgen dat iedere functie, iedere class en iedere module zijn eigen unit tests krijgt. Op deze manier kan er, door uitvoering van alle unit tests, bij een verandering aan de broncode van een bepaalde module meteen gezien worden of de wijziging in die module gevolgen heeft voor andere modules [WiPe-1w]. Alle unit tests bij elkaar vormen eigenlijk een soort regressietest suite.
Het voordeel van unit tests is dat ze ieder betrekking hebben op een klein stukje van de broncode. Op het moment dat er een bug gevonden wordt is het, na uitvoeren van de unit tests, meteen duidelijk dat de fout zich in de buurt van de code van de falende unit test bevindt [WiPe-1w][Zel06]. Verder zorgt de aanwezigheid van unit tests voor meer vertrouwen in het software systeem. Een programmeur kan gemakkelijk iets wijzigen en meteen kijken aan de hand van de reeds aanwezige unit test of de betreffende broncode nog doet wat deze zou moeten doen [WiPe-1w][Zel06].
Unit testing heeft ook een beperking. Zo zal het gebruik van unit tests niet alle fouten in het software systeem achterhalen. Per definitie richt een unit test zich alleen op de functionaliteit van de betreffende module waar deze voor geschreven is. Om deze reden zullen unit test vanuit zichzelf geen integratie fouten of performance problemen aan het licht brengen [WiPe-1w]. Unit testing zal dus altijd gebruikt moeten worden in combinatie met andere test technieken zoals integratietesten [WiPe-1w].
Unit testing is een vast en belangrijk onderdeel van de software ontwikkelmethodiek Extreme Programming [Bec99].
Het invoeren van unit testing in een bestaand software systeem kan twee problemen opleveren [Fre03]. Ten eerste heeft een ontwikkelteam niet de tijd en het budget om het hele systeem van unit tests te gaan voorzien [Fre03]. Ten tweede kan het lastig worden om bij alle code unit test toe te voegen omdat er bij het schrijven van de code geen rekening gehouden is met unit tests [Fre03]. Hierdoor kan sommige code zich niet lenen voor het toevoegen van unit tests en zal eerst een deel van de code herschreven moeten worden om hier wel aan te voldoen [Fre03].
**Hoe staat het bij OnGuard?**
Bij aanvang van het onderzoek wordt er vrijwel niets aan developer testing gedaan. Dynamisch testen door developers in de vorm van Unit Testing komt niet voor. Uit interviews met programmeurs binnen OnGuard is gebleken dat programmeurs hun eigen werk puur functioneel testen zonder vastgestelde scenario's. Meer resultaten van dit interview zijn te vinden in paragraaf 4.2.1. Volgens procedures wordt er wel statisch getest in de vorm van informele code reviews waarbij programmeurs elkaars werk beoordelen. Helaas wordt deze procedure niet of nauwelijks nageleefd. Volgens de programmeurs heeft dit onder andere te maken met de tijdsdruk die op de hen gelegd wordt omdat men werkt met vaste release data.

### 2.2.5 Testing process

Zeller geeft aan dat het proces, van het vinden van een bug tot het oplossen ervan, er als volgt uit ziet [Zel05]:

- Een eventuele bug in het systeem wordt ontdekt
- Een Problem Report wordt aangemaakt
- De bug wordt gedocumenteerd en samen met het problem report opgenomen in een bug-beheersysteem
- De bug wordt beoordeeld en geclassificeerd op de onderdelen:
    - Severity
      *Hoe hevig is de invloed van een bug? Er worden door Zeller de volgende typen onderscheiden* [Zel05]:
        - Blocker
          Zorgt ervoor dat het hele ontwikkelings- of testproces stil ligt. Dit is het hoogste hevigheidslevel.
        - Critical
          *Fouten die crashes, dataverlies en grote memory leaks veroorzaken*
        - Major
          *Verlies van functionaliteit*
        - Normal
          *"Standaard" probleem*
        - Minor
          *Verlies van een klein deel van functionaliteit of andere problemen waar een makkelijke oplossing / workaround aanwezig is*
        - Trivial
          *Een fout die ontzettend makkelijk op te lossen is. Spel- of uitlijningsfouten in de user interface zijn goede voorbeelden hiervan.*
        - Enhancement
          *Verzoek voor uitbreiding. Dit betekent dat het geen echte fout betreft, maar een gewenste functionaliteit. Niet te verwarren met een ontbrekende functionaliteit: als een product niet aan requirements voldoet is dit een fout van de categorie major* [Zel05].
    - Priority
      *Hoe hoger de prioriteit van een fout, des te eerder moet er begonnen worden met het oplossen van de fout. De prioriteit van een fout wordt meestal bepaald door het management* [Zel05].
    - Identifier
      *Ieder probleem krijgt zijn eigen unieke identifier. Op deze manier kan de ontwikkelaar verwijzen naar de fout in het debugging proces, change logs, status rapportages, e-mails en bijlages* [Zel05].
    - Comments
      *Al het commentaar betrekking hebbende op het probleem wordt bij het probleem opgeslagen. Het betreft hier informatie over de omstandigheden van het probleem, eventuele oorzaken, tussenrapportages of hoe het probleem opgelost zou kunnen worden* [Zel05].
    - Notifiation

*Ontwikkelaars en gebruikers kunnen op de hoogte gehouden worden van de status van een fout. Dit kan automatisch door gebruik te maken van een bug reporting / managing systeem. De meeste van deze systemen ondersteunen deze functionaliteit.*

- Het probleem wordt in behandeling genomen en eventueel opgelost

**Hoe staat het bij OnGuard?**

De in kaart gebrachte workflow van de huidige situatie in het testproces van OnGuard geeft aan dat er op de door Zeller voorgeschreven manier gewerkt wordt. Echter op het punt van de informatie die opgeslagen wordt bij de bugreports verschilt de werkwijze van OnGuard van die van Zeller. Zo wordt er bij aanvang van het onderzoek bijvoorbeeld geen ernstcategorie aan de bug toegekend. Verder wordt er bij OnGuard ook voor nieuwe functionaliteit een melding aangemaakt. Uiteraard loopt het proces door nadat het probleem in behandeling wordt genomen. Er zal getracht worden het probleem op te lossen. Wanneer men duidelijkheid heeft over de status van het probleem zal er terugkoppeling plaats vinden naar de melder ervan.

### 2.2.6 Gestructureerd testen

Gestructureerd testen is belangrijk voor een beheersbaar en inzichtelijk testproces [Pol05]. Volgens de TMap methode, een methode die een kant en klare manier van werken omtrent testen en de organisatie daar om heen beschrijft, zorgt gestructureerd testen verder voor een reductie van de tijdsdruk op de testafdeling [Pol05]. Dit kan volgens Pol et al. in [Pol05] bereikt worden door een tijdige, flexibele planning, voortgangsbewaking, een tijdige start van testen, voldoende en geschikt personeel en sluitende afspraken zowel binnen de testafdeling als met de afdelingen daarbuiten.

De vier pijlers onder een gestructureerde testaanpak zijn fasering, technieken, infrastructuur en organisatie [Pol05]. De TMap methodiek is een van de vele methoden om het testproces in te richten. Het toepassen van het zogenaamde V-Model zorgt bijvoorbeeld ook voor een gestructureerde aanpak.

**Hoe staat het bij OnGuard?**

Binnen OnGuard wordt er niet op een gestructureerde manier getest. Er is geen planning met betrekking tot de uit te voeren testen. Doordat er geen planning is, is voortgangsbewaking vrijwel onmogelijk. Binnen OnGuard is er één tester. De vraag is of deze persoon het testen alleen af kan.

### 2.2.7 Testing process referentiemodellen

Referentiemodellen bieden een duidelijk referentiekader om sterke en zwakke punten van het testproces te inventariseren [Pol05]. Een referentiemodel kan gebruikt worden om de mate van volwassenheid te beoordelen van het eigen testproces [Pol05]. Vaak hebben deze modellen verschillende niveaus van volwassenheid, zodat men het testproces kan schalen naar een van deze niveaus. Vaak is er door te kijken naar de eigenschappen van een hoger niveau duidelijk welke veranderingen er plaats moeten vinden alvorens dit hoger niveau van volwassenheid van het testproces bereikt kan worden. Sommige referentiemodellen geven zelfs aanwijzingen met betrekking tot het opstellen van eisen waar het testproces aan moet voldoen.

**Hoe staat het bij OnGuard?**

Bij OnGuard is de volwassenheid van het testproces niet bekend. Er is nog nooit gemeten hoe volwassen hun testproces nu eigenlijk is.

# 3. Plan van Aanpak

## *3.1 Planning*

Het project is in een viertal fasen op te delen.

### 3.1.1 Huidige situatie

In kaart brengen van het huidige softwareontwikkelproces en de plaats van testen daarbinnen. Hierbij wordt zowel een tekstuele beschrijving gegeven van alle activiteiten en de producten die deze activiteiten opleveren als een visuele representatie. De visualisatie zal in de vorm van een workflow diagram gepresenteerd worden. Hierbij is vooral van belang aan te geven waar in het proces in de huidige situatie testing een rol speelt.

### 3.1.2 Analyse huidige situatie

Analyse van het huidige softwareontwikkelproces. Hierbij wordt er gekeken naar op welke plaatsen kwaliteitszorg ingebouwd kan worden, welke testmethoden er gebruikt worden en waar de testknelpunten liggen binnen de huidige manier van softwareontwikkeling.

### 3.1.3 Opstellen gewenste situatie

Opstellen van het verbeterde softwareontwikkelproces. Dit wordt op dezelfde wijze gedaan als bij het in kaart brengen van het huidige softwareontwikkelproces in fase 1.

### 3.1.4 Aanbeveling

Aanbeveling voor invoering van het verbeterde softwareontwikkelproces. In deze aanbeveling zullen aandachtspunten bij de invoering van het verbeterde softwareontwikkelproces, een stapsgewijze invoering en het gebruik van tooling voor het automatiseren van testing aan de orde komen.
De planning ziet er hierdoor als volgt uit:

**Tabel 3: Planning**

| 1 (14) | 2 (15) | 3 (16) | 4 (17) | 5 (18) | 6 (19) | 7 (20) | 8 (21) | 9 (22) | 10 (23) | 11 (24) | 12 (25) | 13 (26) | 14 (27) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | | | | |
| | | | | | | | | | | | | | |
| | | | | | | | | | | | | | |
| | | | | | | | | | | | | | |
| | | | | | | | | | | | | | |

**Tabel 4: Legenda Tabel 3**

| Kleur | Beschrijving |
|---|---|
| | Stageweek- en kalenderweeknummer |
| | Fase1: Huidige situatie |
| | Fase 2: Analyse huidige situatie |
| | Fase 3: Opstellen gewenste situatie |

| | |
|---|---|
| | Fase 4: Aanbeveling |
| | Scriptie |

## 3.2 Benodigde expertise voor dit project

- Toepassing van testmethodieken
- TMap
- Software Construction
- Crediteuren- en debiteurenadministratie domeinkennis
- Kennis van de bij OnGuard gebruikte ontwikkelomgevingen
- Diplomatieke omgang met verschillende tegenover elkaar staande partijen

## 3.3 Risico's

R1: Er kan teveel focus gelegd worden op het maken van veranderingen in het ontwikkelproces
R3: Veranderingen kunnen vaak tot weerstand bij medewerkers leiden
R4: Indien de bewoording bij rapportage niet diplomatiek genoeg opgesteld wordt kan dit leiden tot weerstand binnen OnGuard.
R5: Het niet toepasbaar / implementeerbaar zijn van de aanbeveling.

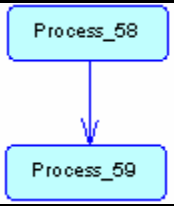
## 3.4 Methoden

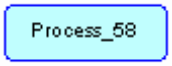
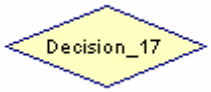
Het is belangrijk dat binnen de uitvoering van dit project methoden worden gebruikt die zichzelf bewezen hebben. Dit is in lijn met het door OnGuard opgestelde mission statement. Het mission statement is als bijlage A toegevoegd.

Voor het achterhalen van de huidige situatie worden de volgende methoden gebruikt:

- interviews:
  Voor het houden van de interviews wordt gebruik gemaakt van gestructureerde interviews. De vragen worden vooraf op papier gezet maar hoeven niet in de opgestelde volgorde gesteld te worden.
- workflow:
  Het workflow diagram is een visuele representatie van activiteiten en beslissingen die binnen een proces achtereenvolgens genomen worden. De verschillende onderdelen van dit diagram worden kort toegelicht in onderstaande tabel.

**Tabel 5: Legenda Workflow Diagram**

| Symbool | Toelichting |
|---|---|
|  | Het pijltje tussen de twee processen of activiteiten geeft de workflow aan. Zodra de activiteit aan het begin van de pijl afgerond is wordt vervolgd met de activiteit waar de pijl naar wijst. De pijl is een pad door de workflow. |

| | |
|---|---|
|  | Dit symbool geeft een activiteit of proces weer |
|  | Dit symbool representeert een vraag ofwel beslispunt. Afhankelijk van hoe de vraag beantwoord wordt kunnen er andere paden gevolgd worden. |
|  | Synchronisatie van gegevens en gegevensstromen worden weergegeven met dit symbool. In de context van het registreren van bugs kan dit dus betekenen dat alle geregistreerde bugs eerst verzameld worden om vervolgens als bulk overgedragen te worden aan een volgende afdeling. |

De overige te gebruiken methoden zijn afhankelijk van de resultaten van de analyse van de huidige situatie en voortschrijdend inzicht.

## 3.5 Validatie

Men spreekt van een efficiënt testproces wanneer afdelingen op elkaar afgestemd zijn, men precies weet wat men van anderen in het proces kan en mag verwachten, wanneer de juiste mensen op de juiste plaats staan waardoor de werklast over het gehele testproces verdeeld wordt.

Er zijn drie factoren te halen uit de bij OnGuard bijgehouden metrieken die betrekking hebben op de hierboven genoemde efficiëntie van het testproces. De efficiëntie van het testproces wordt gemeten in:

**Het gemiddelde verschil in dagen tussen opeenvolgende opleveringen van meldingen aan de testafdeling per uitlevering**
Dit getal geeft aan hoeveel dagen er gemiddeld tussen twee aan de testafdeling opgeleverde meldingen zit waarmee wordt aangegeven hoeveel tijd men op de testafdeling heeft om deze opgeleverde melding te testen. Dit getal geeft dus ook aan in hoeverre de ontwikkelafdeling en de testafdeling op elkaar afgestemd zijn. Een laag gemiddelde houdt in dat de druk op de testafdeling vanaf de ontwikkelafdeling hoog is.

**Per uitlevering, het gemiddelde verschil in dagen tussen het moment dat een melding aan de testafdeling opgeleverd wordt en het moment dat deze melding voor het eerst getest is**
Dit getal geeft weer hoeveel tijd het kost voordat men kan beginnen met het daadwerkelijk testen van een opgeleverde melding. Ook geeft dit getal aan hoe lang het duurt voordat de programmeur feedback krijgt van de testafdeling op zijn geleverde werk. Een hoog gemiddelde houdt in dat de tester veel tijd nodig heeft om de testen voor te bereiden en zorgt ervoor dat, naar mate de tijd verstrijkt, de kennis van de programmeur op het gebied van de opgeleverde code steeds meer weg zakt.

**Het gemiddelde verschil tussen het gemiddelde verschil in dagen tussen het moment van opleveren en het moment van testen en de gemiddelde tijd in dagen tussen opeenvolgende opleveringen van meldingen**
Dit getal geeft een goed beeld van de werklast van de testafdeling. Een positief getal betekent een hogere druk en houdt in dat men op de testafdeling nog bezig is met het testen van de vorige opgeleverde melding wanneer de nieuwe melding alweer opgeleverd wordt. Een negatief getal betekent een lagere druk en houdt in dat de tester op de testafdeling tijd over houdt voor andere zaken. Dit getal wordt verder de “testdruk” genoemd.

**Gemiddelde testdruk over de totale periode vanaf de eerste beschikbare metingen**
Dit getal is een mooie graadmeter om de efficiëntie het testproces over een langere termijn te beoordelen.

Deze voorgestelde getallen zullen een goede indicatie geven van de efficiëntie van het testproces.

# 4. Uitvoering

In dit hoofdstuk wordt de uitvoering van de in de vorige hoofdstukken beschreven opdracht per fase besproken.

## *4.1 Fase1: Achterhalen huidige situatie*

### 4.1.1 Het belang van het achterhalen van de huidige situatie

Om het testproces te kunnen verbeteren moet eerst duidelijk zijn hoe het testproces er op dit moment uit ziet; hoe de huidige situatie er uit ziet. Het is niet mogelijk om de gehele organisatie in één keer te veranderen. De verschillende bedrijfsprocessen binnen OnGuard moeten namelijk door kunnen blijven lopen. Verandering binnen deze processen of in de samenstelling hiervan moet op dusdanige wijze plaats vinden dat de rest van de organisatie er geen hinder van ondervindt.

Om toch veranderingen binnen de organisatie door te kunnen voeren is het van belang te weten welke bedrijfprocessen afhankelijk van elkaar zijn. Deze afhankelijkheid kan op twee verschillende manieren ontstaan. Ten eerste kan een proces van een ander proces afhankelijk zijn op basis van volgorde van uitvoering. Het eerste proces moet uitgevoerd zijn voordat het tweede proces zijn aanvang kan vinden. Ten tweede kan een proces afhankelijk zijn van de deliverables, de opgeleverde producten, van een ander proces. Het in kaart brengen en dus achterhalen van de huidige situatie is van groot belang voor het onderzoek.

De in kaart gebrachte huidige situatie zal als basis dienen voor de door te voeren veranderingen binnen de organisatie. Het is van belang dat deze veranderingen niet klakkeloos ingevoerd worden maar conformeren aan zowel de visie en missie als aan de manier van werken van OnGuard. De workflow zal een belangrijke rol spelen bij het nemen van beslissingen omtrent veranderingen aan het testproces omdat de impact van veranderingen beter te analyseren is.

Onder de huidige situatie wordt het samenspel van processen verstaan dat in werking treedt vanaf het moment dat een fout in de software of een verzoek voor verandering van de software bij OnGuard aangemeld wordt, tot aan het moment dat de melder terugkoppeling krijgt en de fout voor deze melder opgelost is.

Naast het achterhalen van de huidige situatie van het testproces is het ook van belang om de visie van medewerkers van verschillende afdelingen binnen OnGuard met betrekking tot testen in kaart te brengen. Op basis van deze visies is namelijk een globaal beeld te vormen van de toewijding die te verwachten valt van de verschillende medewerkers om het testproces te verbeteren. Om deze visies te achterhalen is een aantal interviews gehouden.

Maar niet alleen de visies met betrekking tot testen zijn belangrijk om in kaart te brengen. Het testproces wordt in sterke mate beïnvloed door het softwareontwikkelproces [Pol05]. De mate van structuur in het softwareontwikkelproces heeft direct invloed op het testproces [Pol05]. Om

een beeld te krijgen van het softwareontwikkelproces binnen OnGuard zijn er interviews gehouden met een aantal programmeurs van de afdeling Product Management.

### 4.1.2 Werkwijze voor het achterhalen van de huidige situatie

Bij het achterhalen van de huidige situatie is gebruik gemaakt van aanwezige procedures van de betrokken afdelingen. Het betreft hier de afdelingen Service Desk, Product Management en Testing & Quality. Op basis van deze procedures is een eerste versie van de workflow van de huidige situatie opgesteld.

Om de visie met betrekking tot testen en softwareontwikkeling in kaart te brengen zijn er interviews gehouden met programmeurs en testers. De resultaten van deze interviews en de conclusies die hieruit getrokken kunnen worden zijn in paragrafen 4.2.1 en 4.2.2 terug te vinden.

Om de workflow van de huidige situatie in groter detail op te zetten is er met een aantal medewerkers van de betrokken afdelingen een aantal interviews gehouden. Deze interviews zijn uitermate geschikt gebleken om de tot dan toe in kaart gebrachte workflow te valideren.

Vanaf het moment dat alle partijen het eens waren met de weergave van hun manier van werken binnen de workflow van de huidige situatie is het document bevroren. Een diagram van de workflow is aan dit document toegevoegd als bijlage B. Verdere analyse van de in kaart gebrachte workflow is te lezen in paragraaf 4.2.3.

## *4.2 Fase2: Analyse huidige situatie*

Na het in kaart brengen van het huidige bug-verwerkingsproces, ofwel meldingsverwerkingsproces, waar het testproces deel van uitmaakt is het testproces geanalyseerd. Hierbij is gekeken naar het moment van testen, welke testen uitgevoerd worden, knelpunten in het ontwikkelproces voor zowel testafdeling als voor ontwikkelafdeling, planning en management.

### 4.2.1 Resultaten van de interviews met de programmeurs

Testen wordt door de programmeurs als “tijdrovend” en “verschrikkelijk” ervaren. Vanuit het perspectief van programmeur wordt testen gezien als debuggen. Debuggen is in dit geval tijdrovend omdat volgens de programmeurs systeemdocumentatie ontbreekt. Dit heeft tot gevolg dat de het niet duidelijk is wat, bij het optreden van een bug, de samenhang is tussen de bug en de rest van de code. Debuggen wordt verder gezien als een manier om fouten op te lossen en te laten zien dat een programma doet wat ervan gevraagd wordt wanneer het programma op de voorgeschreven wijze gebruikt wordt.

Met de uitlating “Testen is niet het werk van de programmeur” wordt duidelijk aangegeven dat het testen van software door de ondervraagden niet gezien wordt als een taak waar zij een steentje aan moeten bijdragen. Testen hoort volgens de ondervraagden thuis op de testafdeling. Het ontbreken van tooling wordt verder aangegeven waarom developers zelf niet testen.

“Controleren of een programma werkt op alle beschikbare platformen” wordt door de

programmeurs aangegeven als taak van de tester op de testafdeling. Taken van de tester komen volgens de programmeurs neer op puur functioneel testen van het gehele systeem.

De programmeurs geven verder aan dat binnen de ontwikkelafdeling alleen de delen van het programma getest worden waar op dat moment aan gewerkt wordt. Er wordt onderling geen rekening gehouden met bugs die eventueel aan elkaar gerelateerd zouden kunnen zijn. Iedereen werkt aan zijn eigen opdracht.

Kennis op het gebied van testen is er bij de programmeurs vrijwel niet. Slechts de helft van de ondervraagden wist één of meer testsoorten op te noemen. Hierbij werden overigens geen officiële termen gebruikt. De meest genoemde testsoort is black-box testing.

Na uitleg over verschillende testsoorten is de programmeurs nogmaals gevraagd wat ze zelf aan testen uitvoeren. Opmerkelijk is dat alle programmeurs black-box en dus puur functioneel testen. Er wordt gekeken of het programma doet wat het zou moeten doen bij normaal gebruik. Het verzinnen van testcases die horen bij verkeerd gebruik van de software wordt niet door de programmeurs gedaan.

Programmeurs besteden naar eigen zeggen tussen de 33 en 40% van de totale ontwikkeltijd aan testen. Hieronder valt echter ook het debuggen van de software wat zij nog altijd als testen zien. Debuggen maakt volgens de programmeurs ongeveer 60% uit van de totale tijd die door programmeurs aan testen besteed wordt.

Het systeem waar men bij OnGuard de bugs en andere meldingen mee beheert wordt door de programmeurs als sterkste punt gezien in de manier waarop OnGuard met testing omgaat. Per melding wordt specifieke informatie op een centrale plek opgeslagen waar iedereen bij kan. Als zwakte punt wordt het ontbreken van functionele en technische documentatie aangemerkt. Ook de matige controle op de naleving van procedures en afspraken wordt gezien als een verbeterpunt.

Het vertrouwen dat de programmeurs hebben in het testproces is niet hoog. Een programmeur stelde: “Ik heb zelfs het gevoel dat er zonder testproces gewerkt wordt”. Het is de mening van de programmeurs dat de prioriteiten bij de testafdeling anders liggen dan deze bij de ontwikkelafdeling liggen. Zo vinden zij dat fouten in de user interface te veel aandacht krijgen omdat zij van mening zijn dat het om het functionele aspect van het oplossen van de bug gaat.

Naar inzicht van de programmeurs wordt er binnen het softwareontwikkelproces van OnGuard geen ontwikkelmethodiek gebruikt. Zoals een van de programmeurs stelde: “We gebruiken de JBF methode... de Jan Boeren Fluitjes methode… geen dus”.

*Welke conclusies zijn uit deze interviews te trekken?*

- Testen wordt door de programmeurs als een last beschouwd.
- Men ziet testen niet als een ondersteunend proces, zoals een van de programmeurs stelde: “Testen voelt aan als een bijzaak”.
- Testen wordt verward met debuggen. Debuggen is echter maar de helft van het werk wat

bij testen hoort. Testen is het aantonen van kwaliteit. Er moet dus ook getest worden of het programma correct functioneert wanneer er van de voorgeschreven wijze van gebruik afgeweken wordt.
- Voor programmeurs reikt het testproces niet verder dan de muren van de testafdeling
- Programmeurs hebben geen beeld van wat een tester doet en zou moeten doen
- Programmeurs hebben weinig inzicht in de impact van bugs op de code. Men kan nooit garanderen dat de rest van het programma nog werkt op het moment dat er een bug is opgelost
- De domeinkennis van de programmeurs met betrekking tot testen en testjargon is vrijwel niet aanwezig.
- Programmeurs testen de eigen software alleen functioneel. Hier worden alleen de situaties getest die horen bij normaal gebruik van de applicatie. Vreemde invoer van de gebruiker wordt bijvoorbeeld niet op getest. Programmeurs maken testen puur en alleen om te laten zien dat hun programma werkt bij normaal gebruik.
- Programmeurs zien veel heil in ondersteunende systemen als het meldingsbeheersysteem systeem
- Programmeurs hebben weinig vertrouwen in het testproces omdat ze niet weten wat er gebeurt en altijd in hun ogen negatieve dingen te horen krijgen van de testers.
- Programmeurs geven te kennen er geen gestandaardiseerde softwareontwikkelmethodiek gebruikt wordt voor het maken van de software.

### 4.2.2 Resultaten van de interviews met de tester

Binnen de testafdeling van OnGuard is het bijna vanzelfsprekend dat het belang van testen duidelijk is. Helaas voelt men zich vaak onbegrepen door omringende betrokken afdelingen. Testen wordt door het enige lid van de testafdeling als noodzakelijk gezien.

Er worden geen specificatietechnieken gebruikt om de testscenario's te achterhalen. Op het moment dat er een bug gemeld wordt, wordt er voor deze bug een scenario opgesteld om deze bug te reproduceren en te testen. Eventuele raakvlakken van deze bug met andere modules worden, voor zover mogelijk, in kaart gebracht. De betreffende modules zullen in de regressietest betrokken worden.

Het testen gebeurt veelal puur op functioneel niveau. Wanneer het een nieuw te implementeren functionaliteit betreft wordt er, net als bij een bug, bepaald wat de raakvlakken zijn met de rest van het systeem.
De raakvlakken worden bepaald aan de hand van inzicht van de programmeur. Er is géén technische systeem documentatie aanwezig.

Wanneer bekend is wat er getest moet worden wordt er een aantal stappen ondernomen om de wijzigingen te testen. Er wordt hiervoor gebruik gemaakt van een zogenaamde demo-database. Deze database is gevuld met een standaard gegevensset van debiteuren en facturen. Allereerst wordt de melding van de bug of nieuwe feature grondig onderzocht totdat de melding duidelijk is. Vervolgens wordt er voor iedere soort database, zowel voor SQL Server als Oracle, gezorgd dat er een scenario uitgevoerd wordt waarmee de melding gereproduceerd kan worden. Dit scenario wordt vooraf door de afdeling Service Desk opgesteld. Na het uitvoeren van de

scenario's om de fout te reproduceren wordt er verder functioneel getest. Mocht er een fout ontdekt worden, wordt het betreffende onderdeel teruggegeven aan de programmeur en zal deze de fout proberen te corrigeren. Deze cyclus herhaalt zich net zolang totdat de bug opgelost is. De testafdeling komt eigenlijk pas in beeld nadat de bug is opgelost.
De bug verhuist hierna naar het zogenaamde "verbandenoverleg", een vergadering waarin alle bugs en nieuwe functionaliteiten nogmaals doorlopen worden, alwaar nogmaals getest wordt of de bug opgelost is.

Als sterkste punt van het huidige testproces wordt genoegd dat er vier testmomenten zijn; de eerste test, de eventuele hertest, het verbandenoverleg en de overall test. De testmomenten en de inhoud daarvan zullen verder worden toegelicht in paragraaf 4.2.3. Helaas gaat dit alleen op wanneer men zich aan de vastgestelde procedures houdt. De tester geeft aan dat er vaak door tijdsdruk of gebrek aan controle van deze procedures wordt afgeweken. Hierdoor soms een of meerdere testmomenten overgeslagen.

Het zwakke punt van het huidige testproces is het gebrek aan structuur. Er wordt gewerkt met gezond verstand. Er is maar één tester binnen de organisatie. Deze tester werkt op basis van ervaring en wordt niet ondersteund door anderen en/of tools. Met betrekking tot testscenario's of andere informatie betreffende testen wordt niets vastgelegd. Op het moment dat deze persoon niet aanwezig is ligt het testproces stil. Verder wordt er alleen functioneel getest. Er worden geen statische testen of andere specifieke testen uitgevoerd. Als laatste heerst er het gevoel dat er te veel testwerk ligt voor slechts één enkele tester. Daarom wordt door de tester een duidelijke wens uitgesproken voor het automatiseren van het testen.

*Welke conclusies zijn uit dit interview te trekken?*

- De tester heeft het gevoel dat testen niet begrepen wordt buiten de testafdeling
- Het werk wat er door de tester gedaan moet worden is niet door slechts een persoon te behappen. Er is behoefte aan een sterke mate van automatisering om de testlast te verlichten.
- Er wordt voor het testproces te weinig documentatie vastgelegd voor het vaststellen van een testbasis
- Er wordt door de testafdeling geen documentatie vastgelegd in de vorm van scenario's of testscripts
- De testkennis is niet overdraagbaar. Bij het wegvallen van de tester is er niemand die de rol over kan nemen.
- Er worden geen methodieken gebruikt voor het vaststellen van scenario's
- De tester geeft aan dat er vier testmomenten zijn. Dit betreft testmomenten binnen de testafdeling. Bij analyse van de workflow zal blijken dat er meer testmomenten binnen het testproces van OnGuard opgenomen zijn. De tester geeft hiermee aan dat testen alleen binnen de muren van de testafdeling bestaat.
- Er heerst onbegrip ten opzichte van programmeurs omdat ze hun eigen code niet of niet goed testen voordat deze aan de testafdeling overgedragen wordt.
- Er zijn meerdere testmomenten waardoor de kans kleiner is dat een fout door de controle heen glipt. Maar deze worden niet altijd benut.

- De tester voert veel testen uit, maar heeft niet de kennis van het testjargon om deze te benoemen.
- Er wordt alleen black-box testing uitgevoerd.

### 4.2.3 Analyse van de workflow van het huidige testproces

In deze paragraaf zal de in kaart gebrachte workflow van het testproces binnen OnGuard een analyse ondergaan. Zo zal er uitspraak worden gedaan op de volgende vragen:

- Welke afdelingen zijn er betrokken bij het huidige testproces?
- Wat is het moment van betrokkenheid van de testafdeling binnen het huidige testproces?
- Wat zijn de testmomenten en hoe vaak wordt er getest?
- In welk testmoment worden welke testsoorten toegepast?
- Zijn er knelpunten in het huidige testproces?
- Wat is de efficiëntie van het huidige testproces?

Bij iedere vraag zullen, indien nodig, een of meerdere plaatjes van de workflow getoond worden ter verduidelijking van het antwoord op de vraag. Een afbeelding van de volledige workflow van het testproces is terug te vinden als bijlage B.

**Betrokken afdelingen**

In de workflow is duidelijk te zien dat de testafdeling met andere afdelingen te maken heeft. Het betreft hier allereerst de afdelingen Service Desk en Product Management. De rol van de eerstgenoemde afdeling speelt zich af aan het begin van het testproces en aan het eind ervan.

De afdeling Service Desk is ervoor verantwoordelijk dat nieuwe bugs en suggesties aan het bug-beheersysteem toegevoegd worden. Indien het een reeds bestaande bug betreft wordt de melding gekoppeld aan de hoofdmelding van de bug. Verder wordt de benodigde informatie voor het reproduceren van de bug toegevoegd aan de melding zoals een kopie van de database en screenshots.

Wanneer alle informatie benodigd voor het oplossen van de bug van de klant ontvangen is wordt de melding overgedragen aan de manager van de afdeling Product Management.
Wanneer de bug helemaal opgelost is geeft de Service Desk weer terugkoppeling aan de klant. Dit gebeurt zoals gezeg aan het eind van het testproces.

De afdeling Product Management is verantwoordelijk het oplossen van de bugs. Zij zijn verantwoordelijk voor het aanpassen van de software. Deze afdeling blijft betrokken zolang de bug nog niet opgelost is; zolang de oplossing nog niet goedgekeurd is door de testafdeling.

De laatste afdeling die bij het testproces betrokken is, is de afdeling Administration, verder te noemen “Administratie”, van OnGuard zelf. Zij doen een gebruikerstest waarbij de nieuwe intern opgeleverde versie uitgebreid gebruikt word in de real-life situatie binnen OnGuard.

*Conclusie(s):*
Het testproces van OnGuard betreft naast de testafdeling een drietal andere afdelingen:

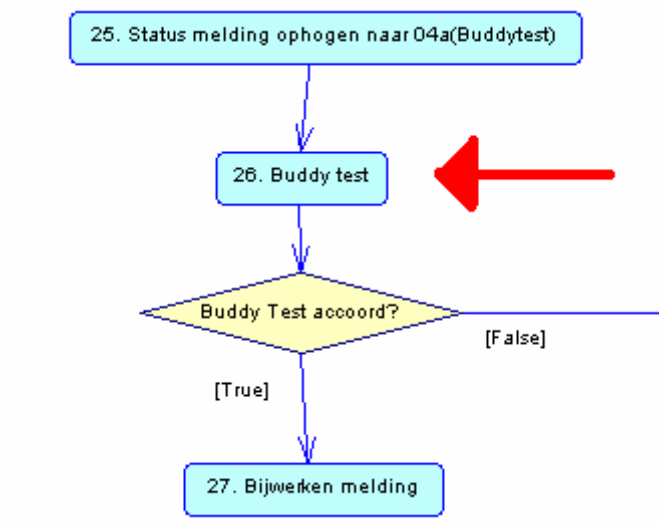
- Service Desk
- Product Management

- Administratie

Hieraan is duidelijk te zien dat het testproces direct betrekking heeft op een groot deel van de organisatie. Het beeld wat de programmeurs hebben dat testen binnen de muren van de Testafdeling plaats vindt is hiermee verworpen.
Maar niet alle afdelingen zijn op dit moment betrokken bij het testproces. De Sales afdeling kan een belangrijke rol spelen bij het boven water krijgen van end-user requirements. Deze requirements hebben direct invloed op het testproces omdat op basis van deze requirements testscenario's geschreven kunnen worden die de correcte werking en aanwezigheid van deze eisen toetsen [Pol05].
Iedere afdeling kan zijn steentje bijdragen aan het testproces. De afdeling Service Desk doet dit onder andere door het registreren van bugs en suggesties in en beheren van het bugbeheersysteem. De afdeling Product Management, de ontwikkelafdeling, is betrokken bij het testproces doordat deze de implementatie door middel van verschillende white-box testsoorten kan toetsen aan de hand van opgestelde eisen. De afdeling Administratie is in dit verhaal een speciaal geval omdat deze fungeren als end-users bij het zogenaamde soak-testen van een nieuwe intern opgeleverde release; Zij werken er een week mee en geven terugkoppeling aan de manager van de afdeling Product Management.

**Moment van betrokkenheid van testen**
Het moment van betrokkenheid van testen heeft eigenlijk niets met de testafdeling te maken. Dit klinkt natuurlijk wat vreemd, maar testen blijft natuurlijk niet alleen binnen de muren van de testafdeling [Pol05]. Tijdens het zogenaamde planningsoverleg wordt de testafdeling voor het eerst in het testproces betrokken bij het testproces. Helaas is de inbreng van de testafdeling aan dit planningsoverleg niet groot en kan men hier niet over betrokkenheid van testen spreken omdat er niet echt iets getest of getoetst wordt. Er wordt namelijk na het planningsoverleg een lijst met aan te pakken meldingen aan de testafdeling overlegd.

De eerste keer dat er binnen het testproces van OnGuard gesproken kan worden over testen is bij de zogenaamde buddytest. Zoals eerder uitgelegd in paragraaf 2.2.2 wordt de oplossing die een programmeur gemaakt heeft door een andere programmeur, zijn buddy, gecontroleerd. Hierbij wordt zowel naar de code als naar de werking ervan gekeken.




**Figuur 6: Buddy test is het eerste moment van betrokkenheid van testen**

*Conclusie(s):*
Er wordt binnen het testproces van OnGuard erg laat getest. Er wordt pas getest wanneer de eerste iteratie van de implementatie van de oplossing voor een bug of nieuwe functionaliteit voltooid is. Nadeel hiervan is dat, mocht er een fout gemaakt zijn in de eisen die opgesteld zijn voor de implementatie, deze fout pas bij de buddytest of bij de eerste test van de testafdeling naar boven komt. Dan bestaat er een kans dat niemand de fout in de eisen opmerkt en kan de fout ontdekt worden door een klant.
Het moment van betrokkenheid van testen moet dus al liggen bij het opstellen van de end-user requirements voor het systeem. Deze requirements moeten getoetst worden op haalbaarheid, compleetheid en correctheid [Lau02]. Ook moet er getoetst worden op het niet ambigu zijn van de opgestelde eisen en mogen ze elkaar niet tegen spreken [Lau02].
Zoals in paragraaf 2.2 aangegeven: Hoe later een fout aan het licht komt, des te duurder is deze fout om te repareren. Dit wordt geïllustreerd in tabel 6.

**Tabel 6: Gemiddelde relatieve kosten voor het oplossen van bugs gebaseerd op de tijd tussen invoering en ontdekking ervan [McC04]**

| | **Fase ontdekking** | | | | |
|---|---|---|---|---|---|
| **Fase invoering** | Requirements | Architecture | Construction | System Test | Post-Release |
| **Requirements** | 1x | 3x | 5x – 10x | 10x | 10x – 100x |
| **Architecture** | - | 1x | 10x | 15x | 25x – 100x |
| **Construction** | - | - | 1x | 10x | 10x – 25x |

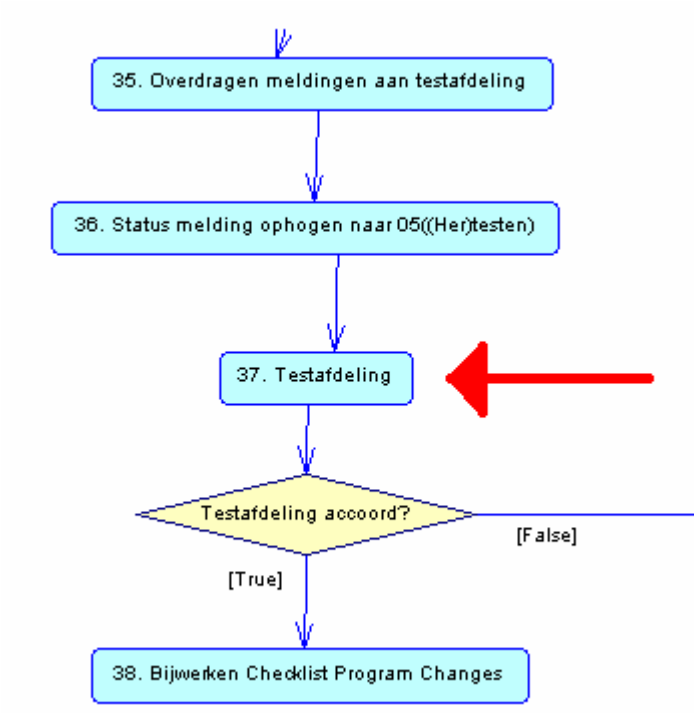
Het is duidelijk te zien dat als fouten ontdekt worden in een andere, latere fase als de fase waarin ze gemaakt zijn de kosten voor het repareren van de fouten ettelijke malen groter zijn dan wanneer ze in dezelfde fase ontdekt worden waar ze erin geslopen zijn. Als voorbeeld: Een fout gemaakt in de requirements fase die ontdekt wordt gedurende de systeem test kost gemiddeld tien

keer zo veel om te repareren dan wanneer deze fout tijdens de requirements fase ontdekt zou worden.
Het laat vinden van een fout verhoogt de testdruk op dit late testmoment. Was deze fout eerder ontdekt dan had de programmeur deze zelf meteen aan kunnen pakken op het moment dat hij nog met de implementatie ervan bezig was. De gemaakte fout zit sowieso in het systeem, maar door deze vroeg te ontdekken behoudt men zelf de mogelijkheid te bepalen wanneer deze fout opgelost word. Men houdt het heft in eigen hand. Hiermee kan gezorgd worden voor een goede verdeling van de testdruk over alle testmomenten.

**Testmomenten**
Zoals in paragraaf 2.2.3 beschreven is het belangrijk om vaak en gespreid over het ontwikkeltraject te testen. Dit levert een besparing op in de testkosten en geeft op tijd een goed beeld van de kwaliteit van het product.

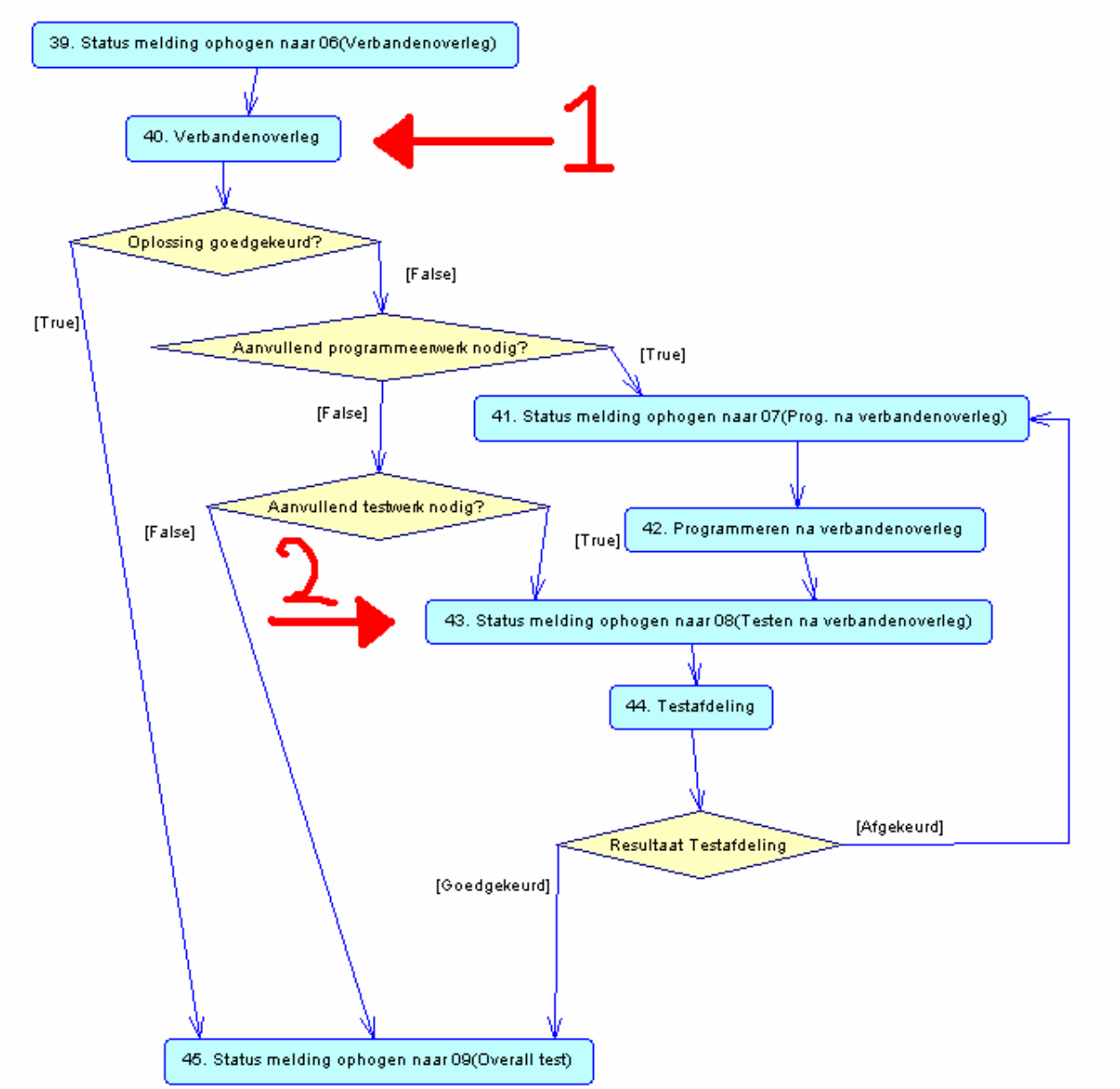
Uit de workflow van de huidige situatie is op te maken dat er op zes momenten getest wordt. Het betreft hier:

- Buddy Test
  Een afbeelding hiervan is hierboven terug te vinden als figuur 6.
- 05 Test
  Dit is het eerste testmoment waarbij de testafdeling zelf direct betrokken is. Bij dit moment van testen wordt getest aan de hand van het scenario, gegeven voor het reproduceren van de bug, of de bug opgelost is. Verder bedenkt de tester zelf nog een aantal scenario's, dat de correcte werking van de functionaliteit en omliggende functionaliteiten bevestigt. Indien meteen goed bevonden wordt de melding verder in het proces getest bij het "Verbandenoverleg". In het andere geval wordt de bug geretourneerd aan de programmeur verantwoordelijk voor het oplossen ervan en zal het volgende testmoment weer de 05 test zijn.




**Figuur 7: 05 Test binnen het testproces van OnGuard**

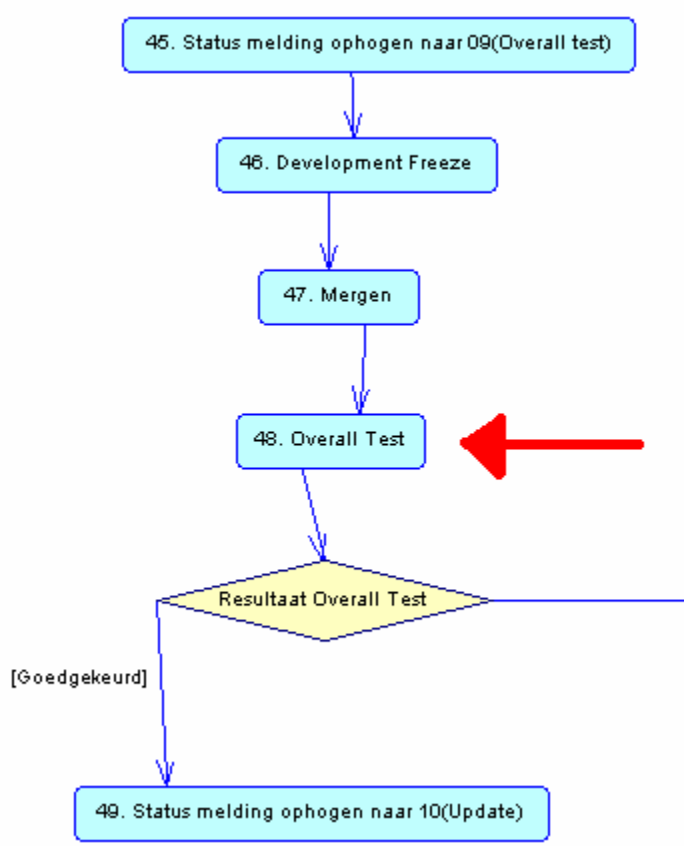
- Verbandenoverleg
  In het verbandenoverleg wordt bekeken of alle meldingen correct zijn opgelost en geen verdere neveneffecten hebben op andere functionaliteiten. Mocht het voorkomen dat er nog fouten in de software zitten wordt de betreffende melding teruggezet naar de ontwikkelafdeling en zal van daaruit weer door de verschillende testmomenten heen moeten. Indien het goed wordt bevonden kan de melding verder naar de Overall Test. Het verbandenoverleg wordt in figuur 8 aangegeven met pijl nummer 1.




**Figuur 8: Verbandenoverleg binnen het testproces van OnGuard**

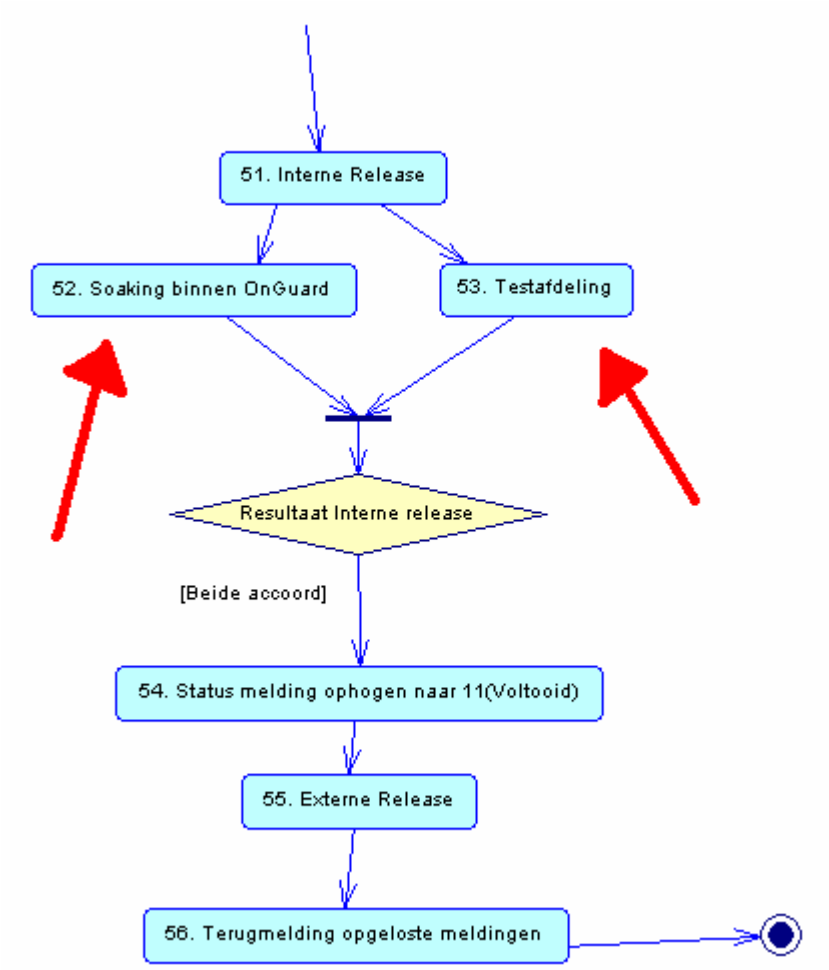
- 08 Test
  Zoals zojuist beschreven komt dit testmoment alleen voor wanneer gedurende het Verbandenoverleg fouten ontdekt worden en de melding weer teruggegeven wordt aan de ontwikkelafdeling. Indien tijdens deze test blijkt dat de fout opgelost is wordt de status van de melding opgehoogd naar “Overall Test”. Zolang de fout niet opgelost is blijft de melding circuleren tussen de ontwikkelafdeling en de testafdeling bij de 08 Test. Dit is aangegeven in figuur 8 bij pijl nummer 2.
- Overall Test

Tijdens de Overall Test wordt maar liefst 80% van de standaard functionaliteiten getest op correctheid. Indien de belangrijkste functionaliteiten naar behoren werken wordt de applicatie voorgelegd aan de afdeling Administratie om aldaar een aantal dagen gebruikt te worden, de soak-periode. De Overall Test wordt getoond in figuur 9.




**Figuur 9: Overall Test binnen het testproces van OnGuard**

- Soak periode
  Zoals beschreven wordt de applicatie intern aan de afdeling Administratie opgeleverd. Indien zich gedurende deze periode fouten mochten voordoen zullen deze gemeld worden aan de ontwikkelafdeling. Gedurende deze Soak periode wordt er door de testafdeling nog extra getest op kwetsbare onderdelen van het systeem. Indien de interne release door deze testen heen komt wordt de software extern uitgeleverd. De laatste twee testmomenten worden getoond in figuur 10.




**Figuur 10: Soak periode en aanvullende tests binnen het testproces van OnGuard**

*Conclusie(s):*
Als er gekeken wordt naar het aantal testmomenten binnen het testproces van OnGuard valt op dat dit er maarliefst zeven zijn. Dit heeft als voordeel dat de kans dat een fout door het testproces heen glipt en door de klant ontdekt wordt ontzettend klein is [Pol05]. Dit gaat echter alleen op mits het testproces ook gevolgd wordt. Vaak komt het voor dat door tijdsdruk procedures niet nageleefd worden waardoor testmomenten overgeslagen worden. Telkens wanneer er een testmoment overgeslagen wordt moet men zich realiseren dat de kans dat er een fout doorheen glipt steeds groter wordt.
De aanwezigheid van meerdere testmomenten zorgt voor een spreiding van de testdruk over het ontwikkelproces. Dit omdat men bij ieder testmoment een andere strategie kan gebruiken. Hierdoor weet men bijvoorbeeld dat wanneer er een functionele test gedaan wordt bij het eerste testmoment en een user interface test bij het tweede testmoment dat men niet beide testen tegelijk hoeft voor te bereiden. De user interface testen worden pas voorbereid op het moment dat men zeker weet dat de functionele test geslaagd is. Mocht men, bij het tegelijk testen, namelijk een fout vinden in de functionaliteit heeft men de user interface test voor niets voorbereid.

Uiteindelijk heeft ieder testmoment evenveel waarde in het testproces en hebben ze allen het doel om een fout te ontdekken voordat de software aan de klant opgeleverd wordt. Zolang de klant niet gedupeerd wordt door een fout in de software heeft de organisatie zelf in de hand wanneer en hoe deze fout opgelost gaat worden. Wanneer de klant met een probleem zit moet de betreffende bug zo snel mogelijk aangepakt worden.

**Testsoorten**
Per testmoment wordt besproken welke testsoorten er uitgevoerd worden. Resultaten zijn verkregen door het bestuderen van de verschillende aanwezige procedures en gevalideerd en eventueel gecorrigeerd aan de hand van interviews met de betrokken personen.

Buddytest:

Volgens de procedures wordt er tijdens de buddy test een zogenaamde code review gedaan door een andere programmeur. De code review wordt gedaan op basis van kennis van de controlerende programmeur. Er zijn geen toetsen opgesteld waaraan de programmeur kan confirmeren; Er worden geen testspecificatietechnieken gebruikt. Code reviews vallen onder white-box testing. Terugkoppeling vindt mondeling plaats.

05 Test:

Tijdens de 05 Test wordt er door de tester op de testafdeling getoetst of het te testen onderdeel doet wat het zou moeten doen. Dit wordt soms gedaan aan de hand van functionele specificaties van het onderdeel. Helaas zijn deze functionele specificaties vaak onvolledig en globaal opgesteld. Er wordt geen testspecificatietechniek gebruikt voor het achterhalen van testscenario's of het maken van testcases. De uitgevoerde testen worden niet gedocumenteerd.
Afgezien van een functionele test wordt ook de user interface gecontroleerd. Hierbij wordt gekeken of alles netjes uitgelijnd is en of het scherm dezelfde look-and-feel heeft als de andere schermen van de applicatie. Hierbij wordt ook een syntactische test gedaan.
De 05 Test omvat enkel black-box testing technieken.
Terugkoppeling wordt gegeven door middel van het logboek bij de melding en het sturen van een e-mail aan de betreffende programmeur. Hierbij wordt geen standaard testrapport gebruikt.

Verbandenoverleg:

Het verbandenoverleg staat in het teken van voornamelijk het toetsen van de oplossing. Alle aangepaste delen van de software worden hierbij onder de loep genomen. Indien nodig wordt de code van sommige oplossingen bekeken in een code review.
Terugkoppeling aan de programmeurs vindt mondeling plaats.

08 Test:

De 08 Test omvat dezelfde tests als de 05 Test. Hierbij wordt echter vaak alleen getest of de tijdens het verbandenoverleg ontdekte fouten opgelost zijn. Van de uitgevoerde testen wordt geen documentatie opgemaakt.
Terugkoppeling naar de programmeur vindt plaats via e-mail. Ook hier wordt geen standaard testrapport gebruikt.

Overall test:

Tijdens de overall test wordt 80% van de totale functionaliteit van het systeem getest op correcte werking. Omdat geen specificatietechniek gebruikt wordt en tijdens de voorgaande fasen de gebruikte testscenario's en testcases niet zijn vastgelegd verschillen de in de overall test gebruikte scenario's en cases van de voorgaande.
Er wordt door middel van gezond verstand en kennis van de applicatie een regressie test uitgevoerd. Er is geen documentatie aanwezig die kan helpen bij het achterhalen van verbanden tussen functionaliteiten.
Terugkoppeling naar de ontwikkelafdeling vindt plaats door middel van het sturen van een e-mail.

Soak periode:

Op de afdeling Administratie van OnGuard voeren de aanwezige medewerkers een real-life test op de software uit. De intern opgeleverde software wordt een week gebruikt door deze medewerkers.
Tegelijkertijd wordt er op de testafdeling een aantal aanvullende testen uitgevoerd om de functionaliteit te testen.

*Conclusie(s):*
Doordat het niet duidelijk is welke functionaliteiten binnen de software van OnGuard aan elkaar gerelateerd zijn is het ontdekken van regressie fouten ontzettend lastig. Om deze fouten te ontdekken moet vrijwel de gehele applicatie getest worden. Dit levert veel dubbel testwerk en vaak ook onnodig testwerk op.
Er wordt weinig tot niet getest op eisen van eindgebruikers. Dit omdat deze eisen meestal niet volledig of erg globaal vastgelegd zijn. Zo kan het voorkomen dat een functionaliteit door het testproces heen komt maar niet overeen komt met de wensen van de klant [Lau02][Pol05].
Er worden geen testspecificatietechnieken gebruikt voor het achterhalen van testscenario's. Hierdoor is men er nooit zeker van dat elk deel van de applicatie getest wordt[Pol05]. Het niet documenteren van de uitgevoerde scenario's leidt tot tijdsverlies en een verhoging van de testdruk bij het testen omdat de scenario's telkens weer opnieuw bedacht moeten worden. De ene keer kan er dus beter getest worden dan de andere keer. En telkens weer moet er tijd gestoken worden in het verzinnen van scenario's terwijl, als de testscenario's vastgelegd worden, men alleen de nieuwe scenario's hoeft te bedenken en vast te leggen.
Het intern op de afdeling Administratie real-life testen van de software is een positieve ontwikkeling. Door de software intern te gebruiken alvorens deze naar buiten te brengen komen fouten aan het licht die niet door de uitgevoerde scenario's naar boven komen [McC04].
De terugkoppeling van tester naar ontwikkelaar loopt stroef. Dit komt mede doordat er geen standaard testrapport gebruikt wordt [Pol05]. Ontwikkelaars krijgen vaak een negatieve terugkoppeling terwijl er vast ook genoeg goede dingen te melden zijn over de gemaakte oplossing. De invoering van een standaard testrapport geeft zowel de tester als de ontwikkelaar een goed beeld van de kwaliteit van de gekozen oplossing en zal eerder door de ontwikkelaar gezien worden als positieve kritiek omdat de correct verlopen testen ook op dit rapport vermeld staan. Zo kan er dan gemeld worden dat het onderdeel functioneel gezien werkt maar er nog een klein foutje in de user interface zit.

### 4.2.4 Knelpunten in het huidige testproces

In deze paragraaf wordt een overzicht gegeven van de knelpunten van de in kaart gebrachte workflow en dus het testproces. Deze knelpunten worden aan het licht gebracht door een combinatie van de workflow analyse en resultaten uit de interviews met programmeurs en testers. Conclusies getrokken uit de workflow analyse worden kort herhaald.

**Laat eerste testmoment**

Zoals beschreven in de workflow analyse bij het onderdeel “Moment van betrokkenheid van testen” wordt er vrij laat in het ontwikkelproces voor het eerst getest. Dit gebeurt pas na de eerste implementatie iteratie.
Fouten in de eisen van de te repareren of nieuw te ontwikkelen functionaliteit komen pas laat aan het licht. In het ergste geval ontdekt de klant deze fout omdat deze een functionaliteit heeft die niet werkt naar zijn wensen.
Doordat fouten pas laat aan het licht komen is de tijd die men er over doet om een functionaliteit te implementeren langer dan wanneer men vooraf weet dat de eisen aan controle zijn onderworpen. Natuurlijk is nooit te garanderen dat er geen fouten gemaakt worden, maar door tijdige controle kunnen deze fouten wel in de kiem gesmoord of zelfs voorkomen worden. Verder heeft men de mogelijkheid om de testdruk te spreiden.

**Dubbel en onnodig testen van functionaliteit**

Uit de analyse van de workflow en interviews met de tester blijkt dat men geen goed beeld heeft van de impact van een wijziging in het systeem. Er is geen systeemdocumentatie dus enkel gezond verstand wordt gebruikt om te bepalen welke onderdelen van het systeem opnieuw getest moeten worden nadat een wijziging aangebracht is [McC04]. Doordat men niet weet hoe functionaliteiten aan elkaar gerelateerd zijn wordt veel van het systeem onnodig getest. Dit gebeurt onder het mom van “beter een keer extra testen dan er straks achter komen dat er toch iets fout zat”.
Het moge duidelijk zijn dat het testen van functionaliteit omdat men denkt dat het gerelateerd is aan een wijziging in de software grote kans tot verspilling van tijd oplevert.
Het gebruiken van gezond verstand om te achterhalen welke onderdelen aan een wijziging gerelateerd zijn werkt prima wanneer men het systeem op zijn duimpje kent. Nadeel hiervan is echter dat wanneer de persoon met deze uitgebreide systeemkennis niet aanwezig is dit niet gedaan kan worden en het testen ofwel stil komt te liggen, danwel niet correct uitgevoerd wordt.
Ook het dubbel testen van functionaliteit kost veel tijd en verhoogt de testdruk. Natuurlijk wil men zeker weten dat alles correct werkt voor het de deur uit gaat, maar als tijdens de eerste keer testen blijkt dat het werkt en er is verder niets veranderd, waarom zou je het dan nog eens testen? Het is echter begrijpelijk dat men dit doet omdat ook dit terug te leiden is naar het gebrek aan inzicht in de impact van wijzigingen.

**Afwezigheid en ongeschiktheid van documentatie**

Uit interviews met programmeurs en testers is gebleken dat er binnen OnGuard weinig systeemdocumentatie gemaakt wordt. Ook is hieruit gebleken dat functionele specificaties vaak te globaal en onvolledig zijn.
In het mission statement van OnGuard wordt vermeld dat men het systeem baseert op end-user requirements. Deze requirements worden echter in zeer kleine getalen vastgelegd en zijn verre

van compleet, volledig en correct. De eisen aan de programmatuur zijn dus niet altijd in dezelfde mate bekend bij alle betrokken partijen. De persoon die de functionele specificatie geschreven heeft, heeft de wijsheid in pacht. Maar deze wijsheid wordt door te weinig kennis met betrekking tot het opstellen van system requirements niet altijd correct overgedragen aan anderen. Nu is de afdeling niet al te groot en de drempel om naar deze manager toe te stappen is laag dus men kan gerust even binnen stappen om wat duidelijkheid te krijgen. Maar wanneer deze manager niet aanwezig is door bijvoorbeeld ziekte, zit men met een probleem.
Door het ontbreken van documentatie ontbreekt het bij het testen aan een nuttige testbasis [Pol05]. Voor het testen gebruikt men de wel aanwezige documentatie, maar deze documenten verschillen telkens in volledigheid.
Omdat er verder ook niets betreft testscenario's wordt vastgelegd behalve een scenario om een bestaande bug te reproduceren blijft ook de testset beperkt tot de ingevingen die de tester op de betreffende dag heeft. De testscenario's waarmee bugs gereproduceerd kunnen worden bevinden zich bij de melding in het meldingbeheersysteem. Men kan dus niet alle testscenario's achter elkaar uitvoeren zonder lange tijd met dit systeem bezig te zijn om de gegevens per melding op te vragen. Ook hier geldt weer hoe meer tijd men moet besteden aan het testen van een enkele melding, hoe hoger de testdruk op dit betreffende testmoment wordt.

**Gebrek aan planning**
Uit interviews en procedures blijkt dat er weinig gepland wordt binnen zowel ontwikkelafdeling als testafdeling. Er worden enkel afspraken gemaakt met betrekking tot de data waarop een release uitgebracht moet worden en wanneer men stopt met het implementeren van nieuwe functionaliteit hiervoor.
Doordat er bij de ontwikkelafdeling niet gepland wordt is er daar weinig inzicht met betrekking tot de afronding van de verschillende onderdelen zoals bug-fixes of nieuwe functionaliteit. Dit heeft tot gevolg dat men op de testafdeling veel te laat weet welke onderdelen er getest moeten worden. Dit in combinatie met de hierboven genoemde afwezigheid van testbasis, de globale functionele specificaties en het niet vastleggen van uitgevoerde testscenario's heeft tot gevolg dat de voorbereiding voordat men kan beginnen met testen erg veel tijd kost en dus een verhoging van de testdruk veroorzaakt.
De testafdeling moet geruime tijd van tevoren weten wanneer testwerk plaats kan gaat vinden. Dit hoeft uiteraard niet precies op de dag nauwkeurig, maar men moet in de gelegenheid zijn zelf te plannen en de testen vooraf te specificeren [Pol05].
Gevolg voor de ontwikkelafdeling van werken zonder planning is dat men lang moet wachten op feedback van de testafdeling. Omdat men niet de tijd heeft om hierop te wachten werkt men vast aan het oplossen van de volgende bug. Hierdoor zijn de programmeurs volledig uit de materie op het moment dat ze een eerder "opgeloste" bug weer moeten oppakken omdat er bij het testen fouten naar boven zijn gekomen [McC04][Zel06].
Daarnaast wil het nog wel eens vaker voorkomen dat de opgestelde procedures niet nageleefd worden. De belangrijkste reden die hiervoor gegeven wordt betreft tijdsgebrek. Dit vormt meteen een risico voor het slagen van het verbeterde testproces. Men kan zo veel regels en werkwijzen opstellen als men wil, maar als ze niet nageleefd worden is het allemaal niets meer dan gebakken lucht.

**Het doel van testen**
Het testproces van OnGuard heeft als doel om fouten te ontdekken of bevestiging te leveren dat de fouten opgelost zijn. Het testproces zou eigenlijk de bedrijfsdoelen en de missie van het bedrijf moeten ondersteunen [McC04][Pol05][Zel06]. Speciale testscenario's zouden hierop gebaseerd moeten worden om de garantie te kunnen geven dat er alles aan gedaan wordt om deze doelen te bereiken. Het testproces kan hierin een controlerende en informerende rol in spelen. Er kan gecontroleerd worden in hoeverre er aan bijvoorbeeld performance van de applicatie gewerkt wordt en het management informeren van de voortgang hierbij.

**Testers vs. Programmeurs**
Vanaf het moment dat er mensen zich specifiek op testen zijn gaan concentreren is er een strijd gaande tussen testers en programmeurs. Deze strijd komt voort uit het "kunstenaarschap" dat de programmeur zich toekent. De programmeur heeft met een stuk code een kunstwerk gemaakt en niemand mag daar commentaar op leveren. Zeker niet wanneer iemand zegt dat er niets klopt.
De testers snappen niet dat programmeurs bepaalde fouten in hun programma's laten zitten en dat ze die zelf niet hadden kunnen ontdekken. Zoals uit de interviews met de programmeurs en testers blijkt liggen prioriteiten van fouten bij programmeurs anders dan bij testers. Een tester vindt een fout in de user interface erger dan een programmeur deze fout vindt. De programmeur redeneert namelijk vanuit de code en de tester vanuit de eindgebruiker [Koh04].
De afkeer van de programmeurs met betrekking tot testen heeft ook te maken met het gebrek aan inzicht in het werkgebied van de tester. Ze weten niet wat deze mensen op de testafdeling precies doen maar krijgen altijd commentaar op hun opgeleverde werk.
Binnen OnGuard zijn de programmeurs van mening dat de feedback van de testafdeling op de door hun opgeleverde code altijd met een negatieve inslag is. Ze krijgen altijd te horen dat er iets niet klopt. Handiger van de testafdeling zou zijn om, zoals eerder beschreven, een standaard testrapport met daarin feedback op te leveren. Hierin kunnen programmeurs zien welke testen er uitgevoerd zijn en, naast dat ene dingetje wat er fout is, de lijst met dingen zien die goed zijn!

*Conclusie:*
Het testproces van OnGuard ontbreekt een fundament. Er zijn weinig afspraken tussen afdelingen onderling op testgebied en er wordt vaak niet volgens een standaard gehandeld. Ook op de testafdeling wordt er niet volgens een vaste procedure gewerkt die voor iedereen bekend en door iedereen uit te voeren is.
De fundering waarop het testproces zou moeten staan, de afspraken, documentatie en procedures is in te geringe mate aanwezig. Dit levert problemen op bij het automatiseren van testen omdat men bij het automatiseren precies moet weten welk deel geautomatiseerd moet worden en er duidelijkheid moet zijn over hoe deze test op een correcte en doeltreffende wijze uitgevoerd zou moeten worden.
Het is daarom belangrijk een duidelijke structuur in het testproces te krijgen waarop men kan bouwen. Pas wanneer deze structuur aanwezig is, is het mogelijk om het testproces op een dusdanige manier te optimaliseren dat men de testinspanning en dus de testdruk kan verlagen. Natuurlijk is het mogelijk om eerder te optimaliseren, maar dat is een korte termijn visie. Uiteindelijk gaat het er om dat men het testproces inzichtelijk en beheersbaar kan maken op een dusdanige manier dat verdere verbeteringen op effectieve en efficiënte wijze doorgevoerd kunnen worden. Het optimaliseren van testonderdelen zonder de aanwezigheid van structuur is als

dweilen met de kraan open omdat alleen geautomatiseerd kan worden wat steeds op dezelfde wijze uitgevoerd wordt.

### 4.2.5 Efficiëntie van het huidige testproces

Aan de hand van de in paragraaf 3.5 genoemde metrieken wordt de efficiëntie van het huidige testproces gemeten. De volgende metrieken zijn uitgevoerd:

- Testdruk voor de uitlevering Gemiddelde testdruk over de periode van 27-05-2005 tot en met 14-06-2006
- Per uitlevering:
    - Gemiddeld aantal dagen tussen opleveringen aan de testafdeling
    - Gemiddeld aantal dagen tussen oplevering aan de testafdeling en de eerste uitgevoerde 05- of 08 Test
    - Gemiddelde testdruk voor deze uitlevering

**Tabel 7: Meetresultaten van de efficiëntiemeting voor het huidige testproces**

| Totale periode | |
|---|---|
| **Over periode van 27-05-05 t/m 14-06-06** | (dagen) |
| Gemiddeld aantal dagen tussen oplevering en 1e 05/08 test: | 6,3 |
| Gemiddeld Aantal dagen tussen opleveringen aan testafdeling: | 1,1 |
| Testdruk: | 5,2 |

| Per uitlevering waarbij overlap van andere uitleveringen meegenomen is | |
|---|---|
| **5.4.0** 27-05-05 t/m 09-08-05 | (dagen) |
| Gemiddeld aantal dagen tussen oplevering en 1e 05/08 test: | 7,5 |
| Gemiddeld Aantal dagen tussen opleveringen aan testafdeling: | 0,8 |
| Testdruk: | 6,7 |

| **5.4.n** 01-08-05 t/m 04-10-05 | |
|---|---|
| Gemiddeld aantal dagen tussen oplevering en 1e 05/08 test: | 13,4 |
| Gemiddeld Aantal dagen tussen opleveringen aan testafdeling: | 1,2 |
| Testdruk: | 12,2 |

| **5.4.4** 17-10-05 t/m 19-10-05 | |
|---|---|
| Gemiddeld aantal dagen tussen oplevering en 1e 05/08 test: | 1,0 |
| Gemiddeld Aantal dagen tussen opleveringen aan testafdeling: | 3,0 |
| Testdruk: | -2,0 |

| **5.4.6** 04-11-05 t/m 09-11-05 | |
|---|---|
| Gemiddeld aantal dagen tussen oplevering en 1e 05/08 test: | 3,8 |
| Gemiddeld Aantal dagen tussen opleveringen aan testafdeling: | 0,8 |
| Testdruk: | 3,0 |

| **5.5.0** 29-09-05 t/m 27-02-06 | |
|---|---|
| Gemiddeld aantal dagen tussen oplevering en 1e 05/08 test: | 6,0 |
| Gemiddeld Aantal dagen tussen opleveringen aan testafdeling: | 1,3 |
| Testdruk: | 4,7 |

| **5.5.n** | 30-01-06 t/m 08-03-06 |
|---|---|
| Gemiddeld aantal dagen tussen oplevering en 1e 05/08 test: | 4,1 |
| Gemiddeld Aantal dagen tussen opleveringen aan testafdeling: | 1,3 |
| Testdruk: | 2,8 |

| **5.5.m** | 10-03-06 t/m 31-03-06 |
|---|---|
| Gemiddeld aantal dagen tussen oplevering en 1e 05/08 test: | 3,4 |
| Gemiddeld Aantal dagen tussen opleveringen aan testafdeling: | 0,7 |
| Testdruk: | 2,7 |

| **5.6.0** | 17-03-06 t/m 06-06-06 |
|---|---|
| Gemiddeld aantal dagen tussen oplevering en 1e 05/08 test: | 7,3 |
| Gemiddeld Aantal dagen tussen opleveringen aan testafdeling: | 1,0 |
| Testdruk: | 6,3 |

| **5.6.s** | 20-05-06 t/m 14-06-06 |
|---|---|
| Gemiddeld aantal dagen tussen oplevering en 1e 05/08 test: | 12,8 |
| Gemiddeld Aantal dagen tussen opleveringen aan testafdeling: | 1,1 |
| Testdruk: | 11,7 |

Wat bij deze meetresultaten in acht moet worden genomen is het feit dat niet alle testmomenten er in zijn opgenomen. Alleen de uitvoeringsdata van de eerste keer dat een 05 of 08 test uitgevoerd is zijn beschikbaar voor verwerking in de efficiëntiemetingen. Hierbij is er, omdat er binnen de meetgegevens geen onderscheid gemaakt kan worden tussen verschillende ernstcategorieën van meldingen, aangenomen dat het testen van een melding één dag in beslag neemt. Er zijn immers meldingen die betrekking hebben op kleine aanpassingen in bijvoorbeeld de user interface. Deze meldingen zijn ontzettend snel te testen. Maar er zijn ook meldingen die langere tijd nodig hebben om volledig te testen. Denk hierbij aan nieuwe functionaliteit waarbij alle mogelijke manieren van gebruik getest moeten worden.
Er is een duidelijke overlap tussen de verschillende versies ofwel uitleveringen. Dit is te zien aan de periode waar men op de testafdeling aan een bepaalde uitlevering werkt. Deze periode begint op het moment dat de eerste melding van de betreffende versie bij de testafdeling binnenkomt en eindigt op het moment dat de laatste melding voor het eerst de 05- of 08 Test doorlopen heeft. Deze overlap zou dus groter zijn wanneer er ook meetgegevens beschikbaar zouden zijn van begin- en einddata van andere testmomenten.

**Analyse van de meetresultaten**
Zoals te zien in tabel 7 is voor de totale periode dat er gegevens met betrekking tot testen op de testafdeling worden bijgehouden een gemiddeld aantal dagen tussen opleveringen aan de testafdeling van 1,1 dagen. Dit houdt in dat er gemiddeld gezien iedere dag een melding wordt overgedragen aan de testafdeling om voor het eerst getest te worden.
Om de meldingen niet op te laten hopen op de testafdeling moet de melding eigenlijk op dezelfde dag nog voor het eerst getest worden zodat men de volgende dag kan beginnen met een nieuw opgeleverde melding. Het resultaat met betrekking tot het gemiddeld aantal dagen tussen oplevering van een melding aan de testafdeling en de eerste 05- of 08 Test die op deze meldingen

uitgevoerd wordt is echter 6,3 dagen. Er kan een aantal verklaringen gegeven worden voor de hoogte van dit getal. Overlap met latere testmomenten van een eerdere uitlevering kan er voor zorgen dat een melding later getest wordt. Dit kan te wijten zijn aan het feit dat eerdere uitleveringen veelal een hogere prioriteit hebben omdat deze zo snel mogelijk afgerond moeten worden. Ook het specificeren van de uit te voeren testen, het opzetten van de testomgeving en het afronden van andere uit te voeren testen zoals het verbandenoverleg, de overall test en de soak periode spelen een rol bij het hoog uitvallen van dit getal. Als laatste reden voor het hoog uitvallen van dit getal is dat uit de meetresultaten blijkt dat veel meldingen tegelijk aan de testafdeling opgeleverd worden. Aangezien er maar één tester aanwezig is kan er maar één melding tegelijk aangepakt worden.
Deze twee zojuist genoemde getallen leiden voor de totale periode voor een gemiddelde testdruk van 5,2 dagen. Dit houdt in dat de tester gemiddeld iets meer dan 5 dagen tekort komt om een melding te testen voordat een nieuwe melding opgeleverd wordt aan de testafdeling. De meldingen hopen zich dus op binnen de testafdeling. Hier bevindt zich een duidelijke bottleneck in het testproces.
De testdruk kan op twee manieren verlaagd worden. Ten eerste kunnen er meer testers ingezet worden om de meldingen te verwerken. Hierbij moet goed gelet worden op de taakverdeling binnen de testafdeling en moet er voldoende testkennis bij deze mensen aanwezig zijn. Ten tweede kan het testproces verbeterd worden zodat de meldingen niet allemaal tegelijk opgeleverd worden en vooraf duidelijk is wanneer meldingen ongeveer aan de testafdeling opgeleverd worden. Door vooraf de planningen van ontwikkelafdeling en testafdeling op elkaar af te stemmen heeft men meer inzicht in elkaars werk. Zo kunnen door een heldere en gedetailleerde planning de uit te voeren testscenario's voor het testen van een melding opgesteld worden voordat de melding over wordt gedragen aan de testafdeling omdat men vooraf weet welke melding men op welke dag kan verwachten. Uiteraard is dit alleen mogelijk wanneer er een "planningsperiode" ingesteld wordt indien men er voor kiest om met slechts één tester door te gaan. Bij meerdere testers kunnen verantwoordelijkheden verdeeld worden. Een tester zou zich kunnen richten op het tijdig klaarmaken van testscenario's voor meldingen die opgeleverd gaan worden terwijl de andere tester zich alleen bezig houdt met het uitvoeren van de testen op zich.

Wanneer we kijken naar de resultaten van afzonderlijke uitleveringen is te zien dat er op één uitlevering na overal een negatieve testdruk te zien is. Bij deze uitlevering, versie 5.4.4, is duidelijk te zien dat er gemiddeld 3 dagen tussen de verschillende opleveringen aan de testafdeling zit en de tester er gemiddeld 1 dag over doet om deze meldingen te testen. Hierdoor heeft de tester gemiddeld 2 dagen per melding over om aan andere zaken te werken.

Wat is nu de ideale testdruk? Een positieve testdruk is sowieso niet wenselijk omdat dit een opstopping in het testproces veroorzaakt. Een negatieve testdruk die veel te laag is, is ook niet wenselijk. Dit houdt in dat de tester, mits deze geen andere taken binnen de organisatie heeft, niets te doen heeft. Een gemiddelde testdruk tussen -0,5 en -1,0 zal naar alle waarschijnlijkheid het beste werken. Dit geeft aan dat de tester tussen de eerste test van een melding en de oplevering van de volgende melding tussen 0,5 en 1 dag de tijd heeft om voorbereidingen te treffen voor de volgende uit te voeren test.

### 4.2.6 Wending in het onderzoek

Uit interviews en de workflowanalyse bleek dat er geen testfundament aanwezig was. Dit heeft voor een verandering van de onderzoeksvraag gezorgd. In eerste instantie zou het automatiseren van testen, als efficiëntieverbetering, een belangrijke rol in het onderzoek spelen, maar voordat men kan beginnen met het automatiseren van testen moet er eerst op een structurele en eenduidige manier gewerkt worden. Testautomatisering gaat ervan uit dat alle testen op dezelfde manier gebeuren en dat er volgens bepaalde methodieken getest wordt [Pol05]. Bij verschillende test process maturity modellen is te zien dat automatisering van testen pas in een stadium zijn intrede doet wanneer er een duidelijk testfundament gelegd is; Wanneer er getest wordt op basis van een testplan met gefundeerde teststrategie en gedocumenteerde testspecificatietechnieken en deze testen op eenduidige wijze uitgevoerd worden. In samenspraak met OnGuard is de prioriteit uiteindelijk gelegd bij het structureren van het testproces; het leggen van het testfundament.

### 4.2.7 Test process maturity model selectie

Het testproces binnen OnGuard kan niet zomaar lukraak aangepakt worden. Alvorens verbeteringen aan te brengen is het nodig om een duidelijk beeld te krijgen van de volwassenheid van het huidige testproces. Deze volwassenheid komt het best naar voren door het testproces van OnGuard te vergelijken met een Test Proces Improvement reference model.
Een referentiemodel leent zich uitermate goed voor het bepalen van de volwassenheid van het testproces. Verder kan het referentiemodel gebruikt worden ter validatie van door te voeren verbeteringen. Ook valt het gebruik van een referentiemodel samen met het mission statement van OnGuard omdat op deze manier bewezen technieken gebruikt worden om het testproces van OnGuard met de buitenwereld te vergelijken.

Er zijn verschillende soorten referentiemodellen en verschillende methoden om de volwassenheid van het testproces binnen een organisatie vast te stellen. Om de methode te selecteren die zich het best leent voor gebruik binnen dit project en binnen OnGuard is gekeken naar een aantal verschillende modellen. Belangrijk bij de keuze voor een referentiemodel is dat het voldoet aan de volgende eisen:

- Meerdere niveaus
- Verschillende aandachtspunten
- Pragmatische instelling van het referentiemodel
- de verbeterpunten komende uit de analyse en het gebruik van het referentiemodel moeten binnen korte termijn realiseerbaar zijn
- Beschikbaarheid van een assessment om het huidige proces te kunnen schalen binnen het referentiemodel

Een vergelijkbaar onderzoek naar een te gebruiken referentiemodel is reeds gedaan door Swinkels [Swi00]. In dit onderzoek wordt een aantal Test Process Improvement referentiemodellen met elkaar vergeleken en komen duidelijke karakteristieken van de verschillende beschikbare referentiemodellen aan het licht.
In het onderzoek van Swinkels worden de volgende referentiemodellen met elkaar vergeleken:

- Test Maturity Model (TMM)
- Maturity Model for Automated Software Testing (MMAST)
- Testing Assessment Programme (TAP)

- Testing Capability Maturity Model (TCMM)
- Test Improvement Model (TIM)
- Test Organization Maturity Model (TOM)
- Test Process Improvement Model (TPI)
- Testability Support Model (TSM)

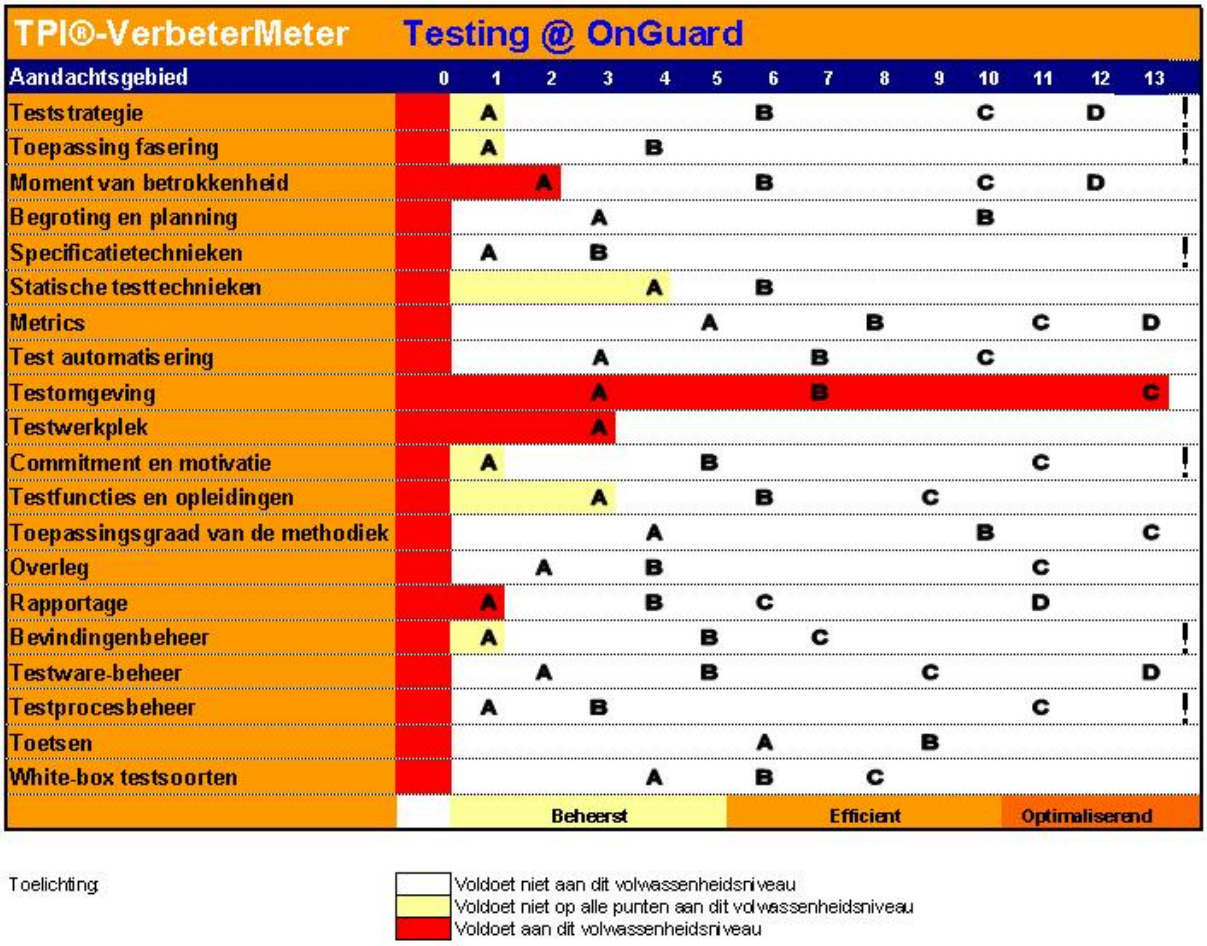
De resultaten van dit onderzoek van Swinkels zullen gebruikt worden voor het selecteren van een geschikt referentiemodel voor het testproces van OnGuard.

De keuze is uiteindelijk gevallen op het Test Process Improvement (TPI) model van Sogeti. TPI is direct gerelateerd aan TMap en biedt de mogelijkheid om TMap gefaseerd en in kleine stapjes in een organisatie in te voeren. Hierdoor is het testproces vrijwel direct aan te pakken zonder dat er eerst veel voorbereiding nodig is binnen de organisatie. TMap past verder perfect binnen het mission statement van OnGuard omdat het een standaard en bewezen techniek is. TPI voldoet aan alle hierboven gestelde eisen.
TPI beschikt over meerdere niveaus en aandachtspunten die de volwassenheid van het testproces bepalen. Het belangrijkste is dat TPI vergeleken met de andere referentiemodellen erg pragmatisch ingesteld is en geen specialist op testgebied vereist. Dit in tegenstelling tot het bekende TMM wat erg conceptueel is en continu begeleid dient te worden door iemand die expert op het gebied van testprocessen is. Dit is belangrijk omdat men binnen OnGuard na afronding van dit onderzoek zelf verder moet kunnen gaan met het verbeteren van het testproces. Verder heeft TPI de beste dekking met betrekking tot de verschillende testactiviteiten waardoor niet alleen het testen binnen de testafdeling aangepakt kan worden, maar het gehele testproces.
TPI biedt verder een duidelijke checklist die als assessment gebruikt kan worden om de volwassenheid te bepalen.
Als laatste is er over TPI veel informatie te verkrijgen in de vorm van boeken en wetenschappelijke artikelen.

### 4.2.8 Test Process Improvement maturity meting

In de vorige paragraaf is besproken waarom er gekozen is voor TPI als referentie model. TPI biedt een assessment in de vorm van een Excel document, de zogenaamde “TPI-verbetermeter”. Het voordeel van de versie in Excel is dat deze automatisch een overzicht weergeeft en makkelijker werkt. Dit document is een verzameling vragen waarop alleen ja of nee geantwoord kan worden. Deze vragen zijn onderverdeeld in twintig key-area’s. Binnen deze key-area’s zijn vragen verdeeld per level. De levels lopen van A tot en met D. Op het moment dat op alle vragen binnen een level van een key-area “ja” geantwoord is én alle eventueel betrokken key-area’s het vereiste level hebben behaald, gaat het key-area een level omhoog.

Het hierboven besproken assessment is binnen OnGuard uitgevoerd. De afstudeerder heeft samen met de manager van de testafdeling de vragen per key-area beantwoord. Eventuele onzekerheden in de antwoorden zijn nagevraagd bij de manager van de ontwikkelafdeling. Uitkomst van dit assessment is dat het TPI-niveau binnen OnGuard op niveau 0 staat. Dit wordt hieronder geïllustreerd door figuur 11.




**Figuur 11: Resultaten van het eerste TPI-assessment van het huidige testproces binnen OnGuard**

Duidelijk te zien is dat op het gebied van Testomgeving het testproces van OnGuard geheel naar wens verloopt. Op dit gebied scoort OnGuard dan ook level C en bevindt zich dus op niveau 13. Verder is er level A gescoord op de gebieden: Moment van betrokkenheid, Testwerkplek en Rapportage. Op de overige gebieden staat het testproces nog in de kinderschoenen of is slechts een deel van de vereisten voor level A behaald.
Een opmerking moet er nog gemaakt worden op het gebied van Bevindingenbeheer. OnGuard mist in dat traject enkele details, maar level B zou op zijn plaats zijn. Door het ontbreken van een detail geeft de TPI-meter niet eens level A aan.

## *4.3 Fase3: Verbeteren situatie*

### 4.3.1 Selectie van TPI verbeterpunten

Nadat het resultaat van het initiële assessment binnen OnGuard besproken was werd het tijd om een aantal verbeterpunten op te stellen. De keuze voor de te verbeteren key-area's van TPI moet

gemaakt worden door de manager van de afdeling Product Management en de senior tester. Om deze keuze eenvoudiger te maken is er door de afstudeerder een adviesrapport opgesteld voor de te maken keuzes. In dit rapport wordt een voorstel gedaan voor de te selecteren verbeterpunten. Ook wordt er een overzicht gegeven van de zogenaamde “easy wins”. Easy wins zijn de punten waarop met relatief weinig inspanning veel resultaat geboekt kan worden. Enkele sleutelgebieden zijn bijvoorbeeld met minimale inspanning naar level A of zelfs hoger te brengen. Afgezien van de kritieke punten waar aan gewerkt moet worden zijn dit natuurlijk makkelijk te behalen winsten. De betreffende sleutelgebieden en een omschrijving van de bijbehorende inspanning om deze tot een hoger level te brengen zijn weergegeven in de onderstaande tabel 8.

**Tabel 8: Easy wins voor verbetering van het huidige testproces van OnGuard**

| Sleutelgebied | van level | naar level | Toelichting |
|---|---|---|---|
| **Teststrategie** | - | A | Vereist het gebruik van één of meerdere testspecificatie technieken. Deze technieken vormen een basis voor een gestructureerd testproces. |
| **Statische testtechnieken** | - | A | Het opstellen van een checklist voor het voeren van de intake op de testbaarheid van de testbasis. |
| **Testfuncties en opleidingen** | - | A | Het vastleggen van de taken van de verschillende functies binnen de testafdeling, het uitbreiden van het testteam met eventueel een programmeur die test en het lezen van een boek met betrekking tot testtechnieken en –management. |
| **Toepassingsgraad methodiek** | - | C | Het opstellen van een generieke testmethodiek en deze gebruiken en onderhouden. (Is te achterhalen aan de hand van de workflow nieuwe situatie) |
| **Overleg** | - | C | Het uitbreiden van het testteam met eventueel een programmeur die test of een fulltime tester. De structuur binnen de afdelingen van OnGuard zorgt er verder dan voor dat overleg geregeld plaats vindt. |
| **Rapportage** | A | B | Door testrapporten op te leveren van iedere uitgevoerde test. |
| **Bevindingenbeheer** | - | A | Het toevoegen van een ernstcategorie aan een melding (misschien zelfs door het aangeven van de ernstcategorie in de beschrijving van de melding) |
| **Testware-beheer** | - | A | Het documenteren van het testproces, wat er gedaan wordt en aanwezigheid van een testbasis. |
| **Testprocesbeheer** | - | A | Schrijven van een testplan per versie(?) |

Het advies dat door de afstudeerder aan OnGuard gegeven is betreft verbeterpunten die betrekking hebben op de structuur van het testproces, het maken van afspraken met de omliggende afdelingen en het creëren van motivatie binnen OnGuard voor het verbeteren van het testproces. Deze keuze is gemaakt omdat uit de knelpunten van het testproces van OnGuard is gebleken dat het vooral aan testfundament ontbreekt. Dit advies wordt gesteund door aanbevelingen van het TPI assessment. De te verbeteren key areas met een korte uitleg worden weergegeven in tabel 9.

**Tabel 9: Advies van afstudeerder voor de aan te pakken TPI key-areas**

| |
|---|
| **Toepassing fasering** |
| Om het testproces binnen de testafdeling overzichtelijk en beter beheersbaar te maken is het nodig om de taken van de testafdeling in een aantal fasen op te delen [Pol05]. Op deze manier is het zowel voor de testafdeling zelf makkelijker om in te schatten hoeveel tijd er nodig is om bepaalde onderdelen te testen. Fasering van taken is verder een krachtig gereedschap in de verantwoording aan bijvoorbeeld management en andere afdelingen. Door het opsplitsen van de taken binnen de testafdeling is het bijhouden en rapporteren van voortgang en tussentijdse resultaten eenvoudiger en inzichtelijker. Het faseren van de taken binnen de testafdeling geeft omliggende afdelingen een beter inzicht en begrip in de verschillende handelingen die de testafdeling uit moet voeren om kwaliteit te waarborgen en verbeteren. Door het faseren is het mogelijk om individuele fasen te beoordelen en te verbeteren. |
| **Teststrategie** |
| Een teststrategie stelt vast welke testen bepaalde requirements en risico's dekken [Pol05]. Een goede teststrategie dekt alle requirements en risico's met zo min mogelijk testen [Pol05]. Hierdoor wordt de kwaliteit van de software hoger en zullen er minder bugs op functioneel gebied bij de klant terecht komen. Een algemene teststrategie geeft de tester een stuk gereedschap in handen waar hij op kan bouwen. Zo hoeft hij alleen na te denken over welke testen hij uit gaat voeren en niet op welke manier deze testen uitgevoerd dienen te worden. Daarnaast hebben andere afdelingen een beter inzicht in het werk van de tester. Ze weten precies welke testen uitgevoerd zullen gaan worden. |
| **Specificatietechnieken** |
| Om met zo min mogelijk testen een zo groot mogelijk deel van de applicatie te dekken is het belangrijk dat de testen op een zo efficiënt mogelijke manier opgesteld worden. Testen moeten nuttig, volledig en correct zijn. Bij het opstellen van testen is het wenselijk dat dit op een uniforme wijze gebeurd. Hierdoor zijn nieuwe testen eerder te begrijpen door buitenstaanders. |
| **Commitment en motivatie** |
| Mensen moeten het belang van testen in het ontwikkelproces in gaan zien. Op het moment dat iedereen weet waarom er getest moet worden en wat er met testen te winnen valt is het "makkelijker" veranderingen in de organisatie aan te brengen. Uitblijven van commitment en motivatie leidt tot grote weerstand en is inherent aan het falen van het project. Dit geldt voornamelijk voor het management, zij spelen in dit proces een belangrijke rol omdat zij een voorbeeldfunctie hebben. |
| **Bevindingenbeheer** |
| De wijze waarop een bug of suggestie beheerd wordt binnen de organisatie beïnvloed in sterke mate de structuur van het testproces. Het is hierbij belangrijk dat alle informatie die nodig is |

| |
|---|
| voor het oplossen en het testen van de bug in het systeem opgenomen zijn. Verder dient informatie opgenomen te zijn waarmee het mogelijk is om prioriteiten te stellen binnen de aan te pakken bugs of meldingen. |
| **Testprocesbeheer** |
| Het beheren en beheerbaar maken van het testproces is van groot belang als basis voor verbetering van het testproces. Pas wanneer de verschillende taken binnen het testproces beheerd kunnen worden is het mogelijk om individuele taken aan te pakken. Onder testprocesbeheer valt in den beginne het opstellen van een testplan. In dit testplan worden alle uit te voeren activiteiten benoemd. Verder wordt er per activiteit aangegeven wat de vereisten zijn, hoe lang het duurt om de activiteit uit te voeren, hoe de activiteit uitgevoerd wordt en wat de activiteit oplevert. |

De uiteindelijke keuze voor de te verbeteren key-area's is gemaakt in een tweetal vergaderingen waarbij de manager van de afdeling Product Management, de senior tester en de afstudeerder aanwezig waren. De rol van de afstudeerder bij deze vergaderingen is beperkt tot het geven van advies en uitleg bij de selectie van de verbeterpunten. Het is aan beide andere aanwezigen om te bepalen of het door de afstudeerder gegeven advies opgevolgd wordt. Bij het opstellen van deze verbeterpunten wordt er onderscheid gemaakt tussen verbeterpunten voor de korte termijn en verbeterpunten voor de lange termijn.

Er is gekozen voor dit onderscheid omdat er op deze wijze op een relatief veilige manier kennis gemaakt kan worden met de TPI methode. Mocht tijdens de korte termijn blijken dat de realisatie van verbeterpunten niet naar wens mocht verlopen, kan tijdig gekozen worden voor een andere methode om het testproces aan te pakken. Verder zou er zoals beschreven in hoofdstuk 3 alleen een advies of aanbeveling worden gegeven in dit project. Er is echter in overeenstemming met OnGuard besloten dat de afstudeerder advies en uitvoering voor de korte termijn op zich zou nemen. Met betrekking tot de lange termijn wordt wel een aanbeveling gegeven. Dit heeft tot voordeel dat men actief begeleid wordt in het verbeteren van het testproces zodat men uiteindelijk, wanneer het onderzoek afgerond is, zelf de kennis heeft om het testproces verder te optimaliseren.

### Korte termijn

De korte termijn betreft een periode van ongeveer twee maanden. Deze twee maanden zijn ingegaan op het moment dat besloten is welke verbeterpunten aangepakt werden.
Gedurende deze twee maanden heeft de afstudeerder een actieve rol in het verbeterproces. Zowel het geven van advies en uitleg als het opstellen van procedures en ondersteunende documenten vallen onder het takenpakket van de afstudeerder.
Tijdens de eerste zogenaamde "TPI vergadering" is besloten om het behalen van TPI level 1 als doel te stellen. Hiervoor zijn hier, zoals eerder genoemd, twee maanden voor uitgetrokken.
Er is gekozen voor het behalen van dit doel om een aantal redenen. Ten eerste is het doel concreet en realiseerbaar. Ten tweede betreft het behalen van TPI level 1 een relatief grote aanpassing aan het testproces en omliggende processen. Hierdoor is het een interessante, veilige en doeltreffende manier om met de TPI methode kennis te maken. De verbeterpunten voor het behalen van TPI level 1, alsmede easy wins en een planning zijn als bijlage C opgenomen in deze scriptie.

**Lange termijn**

De lange termijn is een periode in de toekomst van nog niet vastgestelde duur. Dit is mogelijk omdat er een duidelijk doel gesteld is wat men wil bereiken.
Het doel voor de lange termijn is het behalen van TPI level 5. Er is gekozen voor dit doel omdat als men level 5 onder de knie heeft het gehele testproces onder controle is. Verder is de afspraak gemaakt dat tijd van ondergeschikt belang is ten opzichte van het te behalen eindresultaat. Omdat er veel veranderingen plaats vinden in de manier van werken van de testafdeling en de interactie tussen testafdeling en omliggende afdelingen is er door de afstudeerder het advies gegeven om na het behalen van TPI level 2 een pas op de plaats te maken. Hierbij zal gedurende een uitleveringstraject van een aantal maanden gebruik worden gemaakt van de nieuw verworven kennis en procedures. Tijdens dit traject kunnen eventuele oneffenheden en inefficiënties uit het testproces en de procedures worden gehaald. Op het moment dat de organisatie naar volle tevredenheid op TPI level 2 functioneert, zal vervolgd worden met de weg naar TPI level 5.
Daar de lange termijn buiten de stageperiode van de afstudeerder valt is de rol van de afstudeerder beperkt tot het uitbrengen van advies. Dit advies heeft betrekking op de wijze waarop de verschillende TPI levels behaald kunnen worden en welke zaken daarbij in het oog moeten worden gehouden voor een soepele invoering.
De verbeterpunten voor het behalen van TPI level 5 is als bijlage D opgenomen in deze scriptie.

### 4.3.2 Realisatie van TPI verbeterpunten

Na een aantal verbeterpunten geselecteerd te hebben is de tijd aangebroken om de punten daadwerkelijk te verbeteren. TPI geeft door de vragen in zijn assessment enkele handvatten met betrekking tot zaken die per key-area nodig zijn om een bepaald level te behalen. Geen enkele organisatie is hetzelfde dus wat er gedaan moet worden om aan de voorwaarden te voldoen is geheel aan de uitvoerder. De invulling van de verbeterpunten verschilt dus per organisatie.

Er is gekeken naar de specifieke situatie van OnGuard om te bepalen hoe de verbeterpunten gerealiseerd konden worden. Hierbij is gebruik gemaakt van de in Fase 1 in kaart gebrachte workflow. Zoals eerder aangegeven is het niet mogelijk om de hele organisatie om te gooien. OnGuard kan het zich niet permitteren om tijdelijk geen software op te leveren. Wat van uitzonderlijk belang is in deze situatie is dat er verbeteringen in de workflow waar het testproces deel van uitmaakt aangebracht moeten worden zonder dat hierbij het algehele bedrijfsproces stil komt te liggen. Dit is een extra reden om de realisatie van de verbeterpunten aan te passen aan de werkwijze van OnGuard en niet andersom.

Ook hier is er onderscheid gemaakt tussen realisatie van de verbeterpunten op de korte en op de lange termijn. Zoals in paragraaf 4.3.1 beschreven zal de afstudeerder zowel een adviserende als uitvoerende rol hebben in de realisatie van de TPI verbeterpunten op de korte termijn. Met betrekking tot de realisatie van verbeterpunten op de lange termijn wordt een advies gegeven in de vorm van een aanbeveling waarin aandachtspunten en volgorde nadrukkelijk aan de orde komen.

**Lange termijn**
Omdat er verder qua verbeterpunten voor de lange termijn op het moment van schrijven nog niets geïmplementeerd is zal in deze paragraaf verder niet ingegaan worden op de verbeterpunten voor de lange termijn.
Zoals eerder aangegeven in paragraaf 4.3.1 zal de afstudeerder geen actieve rol spelen bij de implementatie van TPI verbeterpunten voor de lange termijn. Er wordt echter wel een aanbeveling gedaan.
De aan te pakken verbeterpunten voor de lange termijn, om uiteindelijk als organisatie op TPI level 5 uit te komen, zijn uitgewerkt per TPI level. Dit document is terug te vinden als bijlage D bij deze scriptie. Een aanbeveling in de vorm van aandachtspunten voor het behalen van TPI level 5 is in hoofdstuk 5 terug te lezen onder paragraaf 5.1.7.

**Korte termijn**
De verbeterpunten die voor de korte termijn aangepakt zullen worden zijn opgenomen in tabel 10. In deze tabel is verder te zien wat er gedaan moet worden om de betreffende key area's op TPI niveau A te krijgen en wordt een schatting gegeven van het hiervoor benodigd aantal uren. Voor het realiseren van de verbeterpunten voor de korte termijn is een planning opgesteld.

**Tabel 10: Aan te pakken TPI verbeterpunten voor de korte termijn**

| **Teststrategie Doel: Level A** | | | | |
|---|---|---|---|---|
| *TaakNr* | *Taak* | *Hoe* | *Afdeling* | *Gepland aantal uren* |
| 1.1 | Inzicht verschaffen in testsoorten | Lezen TMap hoofdstuk 8 en 9 | Testing | 4 |
| 1.2 | Testspecificatietechnieken | Lezen TMap hoofdstuk 15 | Testing | (4) |
| 1.3 | Wat is een teststrategie | Verzamelen informatie TMap en opzoeken definitie teststrategie | Testing | 4 |
| 1.4 | Opstellen formele strategie | - | Testing | 2 |
| 1.5 | Opstellen template voor risicoanalyse | - | Testing | 2 |
| **Toepassing fasering Doel: Level A** | | | | |
| *TaakNr* | *Taak* | *Hoe* | *Afdeling* | *Gepland aantal uren* |
| 2.1 | Maken checklist per fase | - | Testing | 2 |
| 2.2 | Optimaliseren planning ontwikkelafdeling | Afspraken maken over het gebruik van statusovergangen en planning geregeld overdragen aan testafdeling | Ontwikkeling | 4 |

| **Specificatietechnieken** | **Doel: Level A** | | | |
|---|---|---|---|---|
| *TaakNr* | *Taak* | *Hoe* | *Afdeling* | *Gepland aantal uren* |
| 3.1.1 | Inlezen in specificatietechnieken | Lezen TMap hoofdstuk 15 | Testing | 4 |
| 3.1.2 | Inlezen in specificatietechnieken | Lezen TMap hoofdstuk 15 | Ontwikkeling | 4 |
| 3.2 | Vaststellen specificatietechnieken Testing | - | Testing | 4 |
| 3.3 | Vaststellen specificatietechnieken Ontwikkeling | - | Ontwikkeling | 4 |

| **Commitment en Motivatie** | **Doel: Level A** | | | |
|---|---|---|---|---|
| *TaakNr* | *Taak* | *Hoe* | *Afdeling* | *Gepland aantal uren* |
| 4.1 | Creëren Commitment en Motivatie bij het Management Team | Presentatie voor het Management Team over het belang van een gestructureerd testproces en de winsten die hiermee te behalen zijn | Testing | 8 |
| 4.2 | Realiseren planning testafdeling | - | Testing/Ontwikkeling | 4 |
| 4.3 | Inzicht specificatietechnieken | Lezen TMap hoofdstuk 15 | Testing | (4) |
| 4.4 | Inzicht testsoorten | Lezen TMap hoofdstuk X | Testing | (4) |
| 4.5 | Wederzijds respect Testing/Development | Pair Testing, presentatie testafdeling | Testing/Ontwikkeling | 8 |

| **Bevindingenbeheer** | **Doel: Level A** | | | |
|---|---|---|---|---|
| *TaakNr* | *Taak* | *Hoe* | *Afdeling* | *Gepland aantal uren* |
| 5.1 | Toevoegen ernstcategorie | Zonder DB aanpassing: Vermelden ernstcategorie in meldingsinformatieveld. | Testing / Servicedesk / Ontwikkeling | 2 |
| | | Met DB aanpassing: Hernoemen veld | | Nader te bepalen. |

| | | | | |
|---|---|---|---|---|
| | | “prioriteit” naar “ernstcategorie” en toevoegen 3 radio buttons in het HelpDeskV5 systeem onder prioriteit. Deze radiogroup krijgt vanzelfsprekend de caption “Ernstcategorie”. Prioriteit wordt niet meer bepaald door Servicedesk maar door Release Manager en/of Manager PMDG | | |

| **Testprocesbeheer Doel: Level A** | | | | |
|---|---|---|---|---|
| *TaakNr* | *Taak* | *Hoe* | *Afdeling* | *Gepland aantal uren* |
| 6.1 | Template voor het testplan | Op basis van een voorbeeld testplan uit TMap bijlage A kan een template worden opgesteld | Testing | 4 |
| 6.2 | Opstellen testplan | Testplan moet ingevuld worden voor het testproces en de software van OnGuard | Testing | 4 |

Nadat er vastgesteld is wat de te verbeteren punten zijn en wat er bij deze verbeteringen komt kijken is er begonnen met de implementatie. Per key area wordt kort besproken wat er gedaan is om de wijzigingen te implementeren. Belangrijk bij deze implementatie is dat deze zo veel mogelijk aan moet sluiten op de manier van werken binnen OnGuard.

**Teststrategie**

Om het key area Teststrategie op niveau A te krijgen is het volgens de TPI Verbetermeter nodig om één of meerdere formele of informele testspecificatietechnieken te gebruiken. Omdat er binnen de testafdeling van OnGuard nog niet met deze specificatietechnieken gewerkt werd moest er eerst worden uitgelegd wat een testspecificatietechniek inhoudt. Om dit te bereiken is voorgesteld om literatuur te gebruiken die hierbij meer inzicht kan bieden. De aanwezige tester heeft zich er dan ook op toegelegd het betreffende hoofdstuk uit [Pol05] te lezen.
De te gebruiken testspecificatietechnieken moeten nog nader bepaald worden.
Om meer lijn te geven aan het testproces en de teststrategieën zijn de kwaliteitsattributen van de software van OnGuard bepaald. Een kwaliteitsattribuut is volgens het Software Engineering

Institute (SEI): “Een eigenschap van een object waarop de kwaliteit van dit object beoordeeld wordt door een of meerdere belanghebbenden”.
De kwaliteitsattributen zijn dus eigenschappen van een object die, indien er voldoende rekening mee gehouden wordt, het succes van dat object kunnen garanderen. Per kwaliteitsattribuut moeten eisen worden opgesteld waar aan voldaan moet worden.
Een voorbeeld hiervan is performance. Wanneer gekozen wordt voor performance als speerpunt voor de applicatie zal binnen het testplan een teststrategie opgesteld moeten worden waarin beschreven wordt op welke wijze getoetst gaat worden of het systeem aan de performance-eisen voldoet. Een eis zou kunnen zijn dat een scherm met debiteureninformatie binnen 0,5 seconden getoond moet zijn.
In een vergadering met de algemeen directeur, de manager van de ontwikkelafdeling, de senior tester en de afstudeerder is bepaald wat de kwaliteitsattributen zijn voor de software van OnGuard. De keuze is gevallen op de volgende kwaliteitsattributen:

- Usability
- Funcitonality
- Performance
- Maintainability

Binnen het op te stellen testplan zal dus rekening moeten worden gehouden dat er beschreven staat hoe getoetst kan worden of de software van OnGuard voldoet aan de gestelde eisen op de gebieden van de genoemde kwaliteitsattributen. Ook de ontwikkelafdeling moet in zijn manier van werken rekening houden met deze kwaliteitsattributen.

**Toepassing Fasering**
Zoals beschreven in paragraaf 4.3.1 heeft het faseren van het testproces binnen de testafdeling een positief effect op de beheersbaarheid van de taken waaruit het proces bestaat. Zo bestaat het testproces op de testafdeling in eerste instantie uit drie fasen: Planning, Specificatie en Uitvoering[Pol05]. Deze fasen bestaan ieder uit een aantal activiteiten. TPI schrijft voor om een checklist te gebruiken waarmee iedere activiteit achtereenvolgens afgevinkt kan worden. Dit advies is dan ook overgenomen.
Er is echter gekozen voor een elektronische checklist. De ontwikkelde checklist heeft de vorm van een Excel document. Deze keuze is gemaakt omdat door alle overhead en in te vullen papieren er te weinig tijd over blijft voor het testen op zich.
Met behulp van de Excel checklist, die voor iedere verschillende uitlevering van de software van OnGuard aangemaakt wordt, is het mogelijk om meldingsnummers te registreren. Op basis van een template werkblad voor de checklist op zich wordt een nieuwe checklist voor een melding gemaakt. Door het toevoegen van de template is het makkelijk om aanpassingen te maken aan het Excel document.
Om de checklist goed te laten werken is een deel van de achterliggende intelligentie geprogrammeerd in VBA, een op Visual Basic gebaseerde programmeertaal die binnen Microsoft Office pakketten gebruikt kan worden. Bij het implementeren is rekening gehouden met het feit dat er makkelijk aanpassingen gedaan moeten kunnen worden aan het document zonder dat specifieke programmeerkennis nodig is.
Een voorbeeld van de elektronische checklist is als bijlage E toegevoegd aan dit document.

**Specificatietechnieken**
Testspecificatietechnieken beperken zich niet alleen tot de testafdeling. Ook de ontwikkelaars moeten op den duur gebruik gaan maken van testspecificatietechnieken om de code te testen bij het white-box testen. Het is hier belangrijk dat men allereerst op de hoogte is van het nut van deze technieken. Daarom heeft, naast de tester, ook de manager van de afdeling Product Management zich moeten verdiepen in testspecificatie. Hij moet uiteindelijk zijn mensen aansturen en het goede voorbeeld geven.
De op de ontwikkelafdeling te gebruiken specificatietechniek moet nog worden vastgesteld. Waar men het wel over eens is, is het feit dat de te gebruiken techniek moet zorgen voor een zo volledig mogelijke dekking van de code. Bij de testafdeling is het argument een zo groot mogelijke dekking van de opgestelde eisen van de eindgebruiker.

**Commitment en Motivatie**
Om het key area Commitment en Motivatie op level A te krijgen moet een aantal zaken binnen het testproces aangepakt worden. Allereerst moet het testen door het management aangestuurd worden. Dit is alleen mogelijk op het moment dat er een duidelijke planning aanwezig is waarmee kosten van testen in tijd en geld verantwoord kan worden. Zo moet niet alleen op de ontwikkelafdeling gekeken worden of het werk haalbaar is. Bij uitloop van testtijd of testbudget moet binnen testen naar een oplossing worden gezocht [Pol05]. Dit kan door overwerken of de inzet van extra mensen.
Verder is kennis en ervaring op testgebied van de testers een vereiste voor het behalen van level A. Bij de tester van OnGuard is voldoende ervaring aanwezig. Zoals gebleken uit de gehouden interviews is er echter weinig kennis van testen en testjargon. Om dit te verbeteren heeft de tester een aantal door de afstudeerder voorgeschreven hoofdstukken van [Pol05] doorgenomen. Onduidelijkheden met betrekking tot deze hoofdstukken zijn door de afstudeerder uitgelegd.
Als laatste is een goede verstandhouding tussen testers en andere disciplines in het project en de organisatie vereist. Er is besloten het management te informeren over het belang van een gestructureerd testproces. Omdat de verschillende leden van het management alleen een voorbeeldfunctie hebben kunnen zij de verworven kennis overdragen aan hun medewerkers. Om commitment en motivatie bij de managers te creëren is door de afstudeerder een presentatie gehouden die voornamelijk gericht was op de te behalen winsten door het hebben van een goed gestructureerd, beheersbaar en inzichtelijk testproces. Dit is verder versterkt door het aantonen van gevolgen van het ontbreken van structuur in het testproces.

**Bevindingenbeheer**
Zoals eerder aangegeven in paragraaf 4.2.8 is men op het gebied van Bevindingenbeheer een goede weg ingeslagen bij OnGuard. Het systeem wat gebruikt wordt om meldingen te registreren voldoet aan vrijwel alle eisen die TPI op dit gebied stelt.
Om op level A te komen moet er slechts één klein attribuutje toegevoegd worden aan de informatie die al opgeslagen wordt. Het betreft hier de zogenaamde ernstcategorie. De ernstcategorie is belangrijk omdat deze het selecteren en prioritiseren van op te lossen bugs en te implementeren functionaliteiten makkelijker maakt. Dit doordat men bijvoorbeeld vooraf kan bepalen dat alle meldingen met de hoogste ernstcategorie in de komende release opgelost zullen gaan worden. Ook kan de ernstcategorie dienen als extra managementstool. Er kunnen op deze manier afspraken gemaakt worden over hoe lang meldingen met een bepaalde ernstcategorie

open mogen staan voordat deze aangepakt worden. Dit maakt het bepalen van welke meldingen in een komende release opgenomen worden een stuk makkelijker.
De ernstcategorie is makkelijk toe te voegen aan het proces. Omdat er geen databaseveld beschikbaar is kan deze al in de omschrijving van een melding opgenomen worden. Op de lange termijn moet er uiteraard een wijziging komen in het meldingenbeheersysteem waardoor de ernstcategorie een eigen plaats krijgt in het informatiescherm.

**Testprocesbeheer**
Level A op het gebied van Testprocesbeheer wordt bereikt door het opstellen van een testplan waarin alle uit te voeren activiteiten benoemd staan. Per activiteit wordt hier beschreven in welke periode deze loopt, welke resources benodigd zijn en wat de op te leveren producten zijn.
Door het opstellen van een testplan wordt het voor iedereen die het testplan leest duidelijk wat de taken van de verschillende afdelingen binnen de organisatie zijn met betrekking tot testen. Anderen krijgen hiermee dus ook meer inzicht in de taken van de testafdeling. Dit laatste is ook een belangrijk onderdeel voor het onderdeel Commitment en Motivatie.

### 4.3.3 Bevindingen tijdens realisatie

Tijdens de realisatie van de verbeterpunten is een aantal zaken opgevallen. Deze zaken worden in deze paragraaf kort toegelicht.

Het invoeren van een nieuwe manier van werken is gebleken aan dezelfde zaken onderhevig te zijn als het invoeren van een voor een organisatie nieuwe technologie. Men stuit op weerstand van medewerkers. Ook tijdens het invoeren van verbeteringen aan het testproces is dit eens te meer ondervonden.

Er is een aantal redenen waarom medewerkers weerstand bieden tegen invoering van een nieuwe technologie [Irw00]. Zo hebben mensen en ook organisaties een natuurlijke angst voor verandering. Personen in een organisatie die vrezen dat hun baan overbodig wordt bij het invoeren van een nieuwe technologie zullen deze verandering vaak bewust tegenwerken [Irw00]. Door gebrek aan interesse of onvoldoende inzicht in de nieuwe technologie kunnen medewerkers onbewust de invoering belemmeren. Zelfs managers die een nieuwe technologie graag tegemoet zien kunnen de invoering tegenwerken doordat het budget dat zij toewijzen voor de invoering niet toereikend is voor het volledig en correct uitvoeren van de invoering [Irw00]. Het is gebleken dat het lastig is om mensen te vinden die zich verantwoordelijk voelen voor een nieuwe taak. Wat bij OnGuard gebeurd is tijdens de invoering is dat medewerkers niet van mening zijn dat testen ook tot hun takenpakket hoort. Ze vinden dat ze al genoeg taken hebben en zien er niet graag een bij komen omdat ze bang zijn dat de werkdruk hierdoor hoger wordt.

Tijdens realisatie van de verbeterpunten kwam duidelijk naar boven dat het lastig is om binnen een middelgrote organisatie alle neuzen dezelfde richting op te krijgen. Verder loopt men binnen de organisatie aan tegen problemen met betrekking tot het samenbrengen van de juiste personen. Iedereen heeft een volle agenda en het is lastig om een moment te vinden wat iedereen schikt. Gevolg hiervan is dat belangrijke vergaderingen, die zo snel mogelijk gehouden moeten worden, weken uitgesteld moeten worden omdat de benodigde mensen dan pas een gaatje in de agenda gevonden hebben.

# 5. Resultaten

Binnen dit hoofdstuk worden de resultaten van het onderzoek besproken. Er wordt hierbij onderscheid gemaakt tussen resultaten die voor OnGuard gelden en resultaten die van belang zijn voor andere organisaties die hun testproces in kaart willen brengen en willen verbeteren.

## *5.1 OnGuard Nederland B.V.*

In deze paragraaf worden de resultaten voor OnGuard besproken. Zo wordt er gekeken naar de verschillen binnen de organisatie die op te merken zijn tussen het testproces bij aanvang van het onderzoek en het testproces bij afsluiting van het onderzoek. Verder komen ook de validatie van de resultaten en de risico's aan bod. Daarna wordt er antwoord gegeven op de onderzoeksvraag en worden de gestelde hypothesen bevestigd of ontkracht. Als laatste wordt er in deze paragraaf ingegaan op hoe OnGuard op eigen kracht verder kan gaan met het verbeteren van het testproces.

### 5.1.1 Visie met betrekking tot testen

Aan het begin van het onderzoek is er bij een aantal programmeurs en een tester door de afstudeerder een interview afgenomen. Met behulp van deze interviews is niet alleen de manier van werken binnen OnGuard in kaart gebracht. Ook de visie van deze mensen met betrekking tot testen is vastgelegd. De resultaten van deze interviews zijn te lezen in de paragrafen 4.2.1 en 4.2.2.

Omdat de testvisie in sterke mate samen hangt met de weerstand die deze mensen zouden kunnen bieden tegen verandering van het testproces is het belangrijk te achterhalen of de afstudeerder in staat is geweest om gedurende het onderzoek deze visie in positieve richting bij te stellen. Om dit te verifiëren is er met dezelfde mensen aan het einde van het onderzoek nogmaals hetzelfde interview afgenomen. Hierbij zijn na het interview de verschillen in antwoorden met de geïnterviewden besproken.

**Programmeurs**

Er zijn opmerkelijke verschillen te ontdekken in de visie van de ondervraagden met betrekking tot testen. Waar programmeurs testen eerder nog als "verschrikkelijk" en "tijdrovend" betitelden, werd testen nu gezien als "noodzakelijk". Ook met betrekking tot het testen van programmeurs zelf is een verandering opgetreden. Programmeurs waren aan het begin van het onderzoek van mening dat testen een apart vakgebied was en dat zij, als programmeurs zijnde, er verder niets mee van doen hadden. Door een aantal programmeurs is aangegeven dat de weerstand tegen testen mede komt door het woord op zich. Het woord "testen" kleeft te veel aan de testafdeling waardoor het lijkt alsof het alleen een taak van de testafdeling is. Het woord "kwaliteitscontrole" valt bij de programmeurs in betere aarde. Tijdens het laatste interview zei een programmeur: "Programmeren en testen gaan hand in hand". Dit is een totale ommekeer.

Gedurende het onderzoek is er met programmeurs meerdere malen gesproken over het zogenaamde developer testen. Waar aan het begin ontzettend veel weerstand hier tegen geboden werd onder het mom "Het is niet mijn taak" wordt er in het laatste interview aangegeven dat

developer testing als nuttig gezien wordt. Men wil het zeker eens proberen. De programmeurs hebben door het onderzoek naar eigen zeggen ingezien dat de testafdeling geen invloed heeft op de code. Er wordt namelijk alleen black-box getest. De programmeurs beseffen dit nu ook en geven aan dat de verantwoordelijkheid van de kwaliteit en correctheid van de code bij hen zelf ligt.

Programmeurs geven verder aan dat ze meer zelf zijn gaan testen. Waar aan het begin van het onderzoek aangegeven werd dat gemiddeld 40% van de totale ontwikkeltijd aan testen en debuggen besteed werd. Aan het eind van het onderzoek geven ze aan 60% van deze tijd aan testen en debuggen te besteden. Het aandeel van debuggen hierin is afgenomen doordat sommige programmeurs begonnen zijn met vooraf testen te schrijven.

**Managers**
Vooral op het gebied van kennis van testen is veel vooruitgang geboekt bij de managers. Het Management Team heeft altijd het belang van kwaliteit en de rol die de testafdeling hierin speelt ingezien. Er wordt aangegeven dat men dacht dat het testproces alleen binnen de muren van de testafdeling speelde. Bij het afronden van het onderzoek wordt aangegeven dat men het inzicht gekregen heeft dat het testproces de gehele organisatie omvat. Iedereen kan zijn steentje bijdragen aan de kwaliteit van de software.

Vooral bij de manager van de afdeling Product Management is er qua inzicht in testtechnieken op vooruit gegaan. Hij heeft een veel beter beeld van wat de verantwoordelijkheden en taken zijn van de testafdeling en van zijn programmeurs. Het testproces is voor hem een stuk inzichtelijker en beter beheersbaar geworden. Hierdoor kan hij nu ook de testafdeling beter managen.

**Testers**
Het is duidelijk dat de aanwezige tester bij OnGuard het belang van testen altijd ingezien heeft. Voor hem heeft het onderzoek vooral inzicht gebracht in wat de taken van de testafdeling zijn binnen het testproces. Hij merkt verder dat sommige programmeurs op een andere manier met de testafdeling omgaan dan voorheen. Hierbij moet wel aangemerkt worden dat niet alle programmeurs zo ver zijn. Ook merkt de tester op dat het begrip van de managers voor het werk van de testafdeling toegenomen is. Dit komt onder meer doordat de managers meer inzicht hebben in het testproces en veel kennis met betrekking tot testen hebben opgedaan.

Het is duidelijk dat er op het gebied van de visie met betrekking tot testen een aantal opmerkelijke veranderingen plaats heeft gevonden. Het onderzoek heeft hier een positieve uitwerking op gehad; er is veel vooruitgang geboekt.

Zowel de tester als de manager van de afdeling Product Management zien in dat er nog veel werk te verzetten is voordat het testproces op een gestructureerde, inzichtelijke en vooral efficiënte manier uitgevoerd word. Men schrikt hier echt niet van en wil tezamen de koe bij de horens vatten.

### 5.1.2 Workflow van het testproces

Met betrekking tot de workflow van het huidige testproces is een aantal zaken op te merken. Uit de workflow blijkt dat er binnen OnGuard veel goede dingen gebeuren op het gebied van testen, maar er is ook een aantal knelpunten op te noemen. Deze knelpunten hebben vooral betrekking op een tekort aan structuur. Deze structuur maakt het testproces meer inzichtelijk en beheersbaar. Hierdoor zijn verbeteringen in het testproces aan te brengen die de testdruk kunnen verlagen.

**Goede punten**
Het testproces van OnGuard kent een aantal goede punten die vermeld moeten worden. Allereerst kent het testproces een groot aantal testmomenten. Dit aantal testmomenten zorgt ervoor dat men vrijwel gedurende het gehele ontwikkeltraject een beeld kan krijgen van de kwaliteit van de software. De kans dat een fout door het testproces heen sluipt en ontdekt wordt door een klant wordt hierdoor ontzettend verkleind.

Het systeem dat gebruikt wordt om de meldingen van klanten en medewerkers met betrekking tot bugs en wensen met betrekking tot nieuwe of een verandering in functionaliteit te registreren is een aanwinst voor het testproces. Door alle informatie met betrekking tot deze meldingen op een centrale plaats te zetten is deze voor iedereen toegankelijk en bij goed gebruik beschikt iedereen over alle relevante informatie met betrekking tot een op te lossen bug of te implementeren nieuwe functionaliteit.

Ook de testomgeving die gebruikt wordt op de testafdeling voorziet in vrijwel alle wensen van de tester. Zonder al te veel moeite kan de omgeving aangepast worden aan het te testen systeemonderdeel. Dit zorgt voor een kortere testtijd en een lagere testdruk. Hier heeft de ook de programmeur voordeel van, omdat deze sneller een testrapport van de testafdeling kan verwachten met betrekking tot de door hem aangeleverde melding.

Als laatste zijn er voor iedere afdeling procedures opgesteld. Hierin wordt vermeld wat de taken van de verschillende afdelingen en rollen binnen die afdelingen zijn. Dit zorgt ervoor dat, wanneer deze procedures voor de gehele organisatie beschikbaar zijn, iedereen niet alleen inzicht kan krijgen in zijn eigen taken en verantwoordelijkheden maar ook in die van anderen zodat men ook weet wat men van anderen mag verwachten. Hierbij moet echter wel een kanttekening gemaakt worden. Tijdens het onderzoek is gebleken dat een aantal procedures niet up-to-date was. Het bijwerken en bijhouden van de correctheid van deze procedures is natuurlijk een vereiste en kan veel onduidelijkheid en verwarring binnen de organisatie voorkomen.

**Knelpunten**
Naast de goede punten van de workflow van het testproces van OnGuard is ook een aantal verbeterpunten te noemen. Deze verbeterpunten worden hier kort genoemd. Meer over deze verbeterpunten is te lezen in de paragrafen 4.2.3 en 4.2.4.

Als eerste knelpunt kan het moment van betrokkenheid van testen in het testproces genoemd worden. Het eerste testmoment ligt vrij laat in het ontwikkeltraject. Er wordt pas voor het eerst getest op het moment dat een eerste iteratie van de implementatie van de oplossing van een bug of nieuwe functionaliteit voltooid is. Hierdoor kunnen fouten in de eisen of specificaties van het

betreffende onderdeel over het hoofd worden gezien. Eerder testen geeft nog meer inzicht in de kwaliteit van het product en kan de testdruk op het einde van het ontwikkeltraject aanzienlijk verlagen en meer verspreiden over het gehele traject.

Als tweede knelpunt kan het gebrek aan kennis van impact van veranderingen op het systeem genoemd worden. Men heeft nooit een compleet beeld van wat de invloed van een verandering in het systeem is op de rest van het systeem. Er is wel een aantal medewerkers dat door ervaring met de software het beeld goed in zijn hoofd heeft zitten, maar ook zij kunnen ergens even niet aan denken. Daarnaast is het belangrijk dat ook nieuwe mensen en mensen die minder inzicht in de samenhang hebben een beeld kunnen vormen van de impact van veranderingen. De tester heeft bijvoorbeeld weinig tot geen zicht over de interne werking van de code en moet vaak stukken van de software testen omdat hij een vermoeden heeft dat deze te maken hebben met een gemaakte verandering. Inzicht in de impact van veranderingen heeft een positief effect op de testdruk omdat men zeker weet welke onderdelen opnieuw getest moeten worden door een verandering. De kans dat er onnodig onderdelen getest worden wordt hiermee een stuk kleiner.

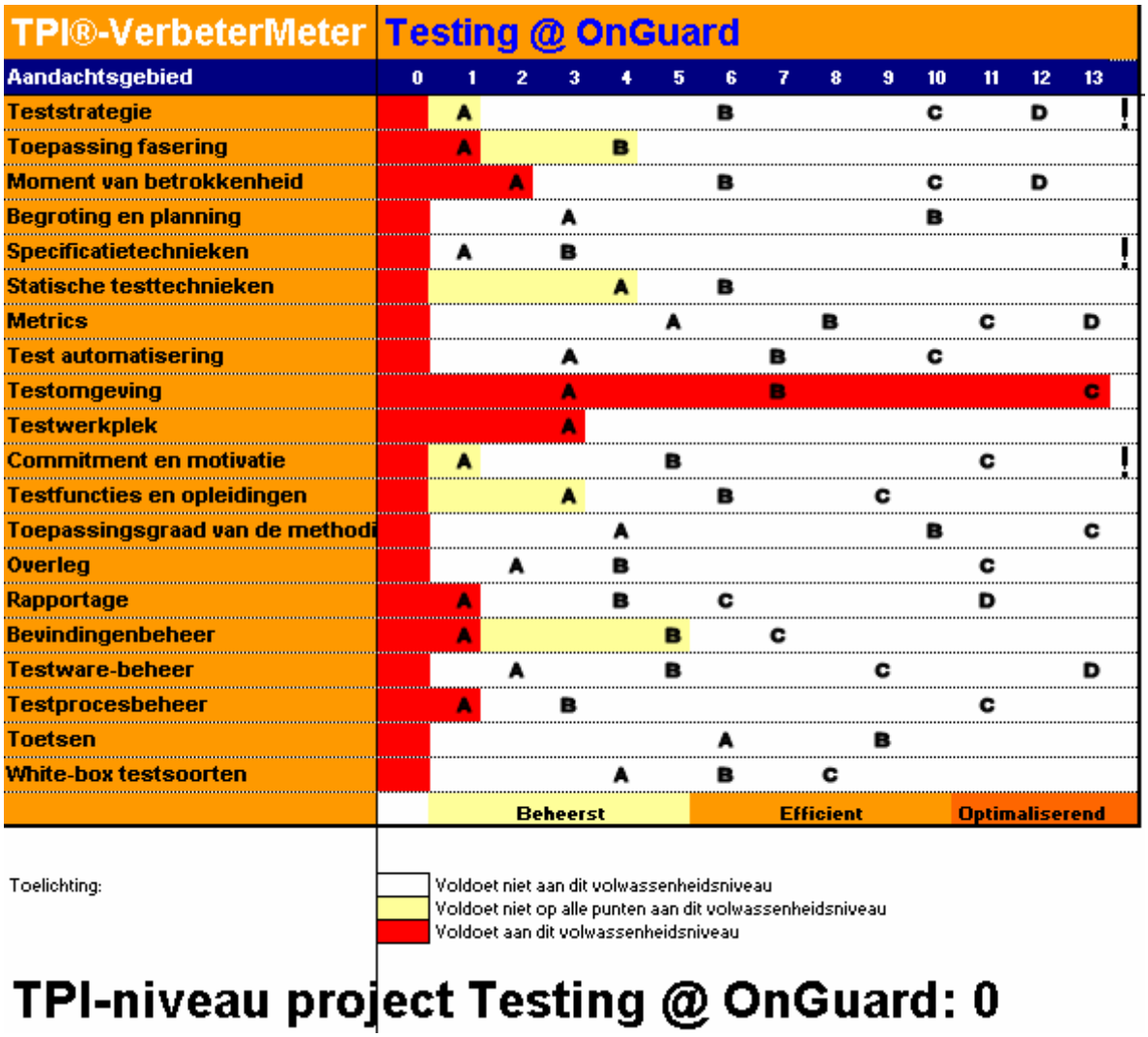
Het ontbreken of onvolledig zijn van documentatie is het volgende knelpunt. Deze documentatie moet uiteindelijk de testbasis vormen waarmee gevalideerd kan worden of de software doet wat afgesproken is. Doordat bijvoorbeeld sommige eisen alleen in het hoofd van een opdrachtgever staan en niet op papier, komt pas op een laat moment naar boven dat aan deze extra eisen ook voldaan moet worden. De programmeurs en testers kunnen immers geen gedachten lezen en zullen altijd zorgen dat de software voldoet aan de opgestelde eisen die zij voor zich hebben liggen voordat ze het stukje software goedkeuren en verder laten gaan in het ontwikkelproces. Testscenario’s worden gebaseerd op de voor de tester bekende eisen. De aanwezigheid, volledigheid en correctheid van documentatie draagt bij aan een lagere testdruk en een korter ontwikkeltraject. Omdat meldingen pas overgedragen worden aan een andere afdeling wanneer men zeker weet dat het aan alle eisen voldoet zal er minder vaak blijken dat er iets niet klopt door een fout in de eisen. Hierdoor wordt er minder vaak overgedragen aan de testafdeling waardoor deze minder tijd hoeft te besteden aan het testen van deze melding. Ook zal de melding minder vaak circuleren tussen testafdeling en ontwikkelafdeling wat ten gunste komt van de ontwikkeltijd, die daardoor lager wordt. Voorbereiden van testen, testen en opnieuw inleven in de op te lossen materie kosten immers allemaal tijd.

Als laatste is een gebrek aan gedetailleerde planning en afstemming van deze planningen tussen de ontwikkelafdeling en testafdeling aan te merken als knelpunt in de workflow van het testproces. Doordat de afdelingen niet van elkaar weten wat ze aan het doen zijn, wanneer het af is en wat er daarna gedaan gaat worden is het heel lastig om goed op elkaar in te spelen. Gevolg hiervan is dat men op de testafdeling niet weet welke melding de eerstvolgende is die hun kant op komt. Hierdoor kan men pas met de voorbereiding voor het testen van de melding beginnen wanneer deze opgeleverd wordt. Dit heeft tot gevolg dat de testdruk hoger wordt en de programmeur langer op feedback moet wachten.

### 5.1.3 Verbeteringen in het testproces

Tijdens het onderzoek is een aantal verbeteringen in het testproces aangebracht. Dit is gedaan aan de hand van een testproces referentiemodel. Als referentiemodel is het TPI model gebruikt. Meer over het TPI model, de keuze en de eerste meting is terug te vinden in paragraaf 4.3.

Aan het einde van het onderzoek is nogmaals een TPI maturity meting gedaan. Resultaten hiervan zijn te zien in figuur12.




**Figuur 12: Resultaat van de TPI Maturity meting aan het eind van het onderzoek**

Wat te zien is, is dat het TPI niveau van het testproces van OnGuard nog steeds op 0 staat. Zodra het testplan echter afgerond is zal men niveau 1 bereikt hebben. Dit is een grote stap voor OnGuard en een grote verbetering van vooral de structuur van het testproces.

Er is gekozen om een master testplan te maken. Er wordt voor iedere uitlevering een apart testplan opgesteld wat gebaseerd is op dit master testplan. Reden hiervoor is dat hiermee per

uitlevering de uit te voeren teststrategie gewijzigd kan worden. De speerpunten van de uitleveringen hoeven immers niet altijd dezelfde te zijn. De ene keer kan er gekozen worden om extra aandacht te schenken aan performance van de applicatie, terwijl een andere keer gekozen wordt voor het uitvoeren van extra testen om de mate gebruikersvriendelijkheid aan te tonen.

Het is te zien dat veranderingen in het testproces vaak gepaard gaan met veranderingen in het ontwikkelproces. Op het gebied van afspraken tussen testafdeling en ontwikkelafdeling is vooruitgang geboekt doordat men besloten heeft een meer gedetailleerde planning binnen de ontwikkelafdeling te gebruiken. Hierdoor weet men op de testafdeling bij benadering welke meldingen binnen afzienbare tijd in de richting van de testafdeling worden opgeleverd. Men kan de testen voor deze meldingen dan op tijd voorbereiden waardoor de testdruk lager wordt en de programmeur eerder feedback krijgt op zijn opgeleverde werk.

Ook is voor de software van OnGuard vastgesteld wat de belangrijkste kwaliteitsattributen zijn. Op basis van deze kwaliteitsattributen kunnen speciale testscenario's opgesteld worden die extra aandacht geven aan de zorg en mate waarin de software in deze kwaliteitsattributen voorziet. Dit is een extra controle op de kwaliteit van de software. Ook gaat er extra aandacht besteed worden aan de vastlegging van de eisen aan de sofware. Eisen worden vooraf verder uitgewerkt waardoor zoals eerder beschreven de testdruk verlaagd en de ontwikkeltijd verkort wordt.

Verder is in een vergadering afgesproken dat de ernstcategorie van meldingen als extra attribuut aan het meldingen beheersysteem wordt toegevoegd. Dit maakt het selecteren en prioritiseren van meldingen voor een uitlevering een stuk makkelijker. Verder biedt deze toevoeging mogelijkheid tot extra metrieken en speciale teststrategieën. Men kan ervoor kiezen om voor zeer ernstige bugs een andere teststrategie te gebruiken dan voor triviale bugs als foutjes in de user interface. Zo krijgt een melding de aandacht die deze verdient en nodig heeft.

Binnen de korte tijd van het onderzoek zijn toch veel dingen verbeterd aan het testproces. Ook de mensen binnen OnGuard hebben veel veerkracht getoond wat te zien is aan de verandering met betrekking tot de visie op testen. Men moet zich realiseren dat deze verbeteringen geen dingen zijn wat men zomaar even doet. Het verbeteren van het testproces is een langdurig traject van geduld, aanpassingen, evaluaties en bijschaven. Er is veel gedaan, maar er moet nog veel meer gebeuren. Men is bij OnGuard nog maar net begonnen aan de reis, maar is duidelijk een goede weg ingeslagen.

### 5.1.4 Validatie van de resultaten

Zoals in de vorige paragraaf beschreven is er pas een aantal verbeteringen toegepast aan het testproces. Dit "nieuwe" testproces is nog niet in gebruik genomen waardoor er nog geen meetresultaten beschikbaar zijn waaruit te zien is of de aanpassingen het gewenste resultaat behaald hebben. Het nieuwe testproces wordt pas in gebruik genomen bij afronding van het testplan en zal gedurende een heel uitleveringstraject gebruikt worden.

Er is echter wel een handvat aan OnGuard gegeven voor het valideren van de resultaten. De validatiemetrieken gebruikt in dit onderzoek zijn makkelijk uit de meetgegevens van OnGuard te halen. Zo kan men zelf op een later moment de testdruk monitoren.

De elektronische Excel checklist, gemaakt om de tester te helpen bij het sequentieel volgen van de testfasen en testtaken biedt extra ondersteuning in het vergaren van meetgegevens omdat van alle testmomenten binnen de testafdeling de begin- en einddata bijgehouden kan worden. Hierdoor is de testdruk te specificeren per testmoment zodat te zien is op welk testmoment de testdruk het hoogst is. Zo kan men afzonderlijke testmomenten aanpakken om de testdruk te verlagen.

### 5.1.5 Risico's

In paragraaf 3.3 is een aantal risico's beschreven die een bedreiging kunnen vormen voor het succesvol uitvoeren van het onderzoek. In deze paragraaf wordt er gekeken welke risico's zich voor hebben gedaan, wat de gevolgen waren en wat er gedaan is om deze gevolgen tegen te gaan.

**R1: Er kan teveel focus gelegd worden op het maken van veranderingen in het ontwikkelproces**

Er is gedurende het gehele onderzoek rekening gehouden met dit risico. Het testproces, niet het ontwikkelproces, is het te bestuderen proces. Het is belangrijk dat wijzigingen aan het testproces zo veel mogelijk zonder wijzigingen aan het ontwikkelproces doorgevoerd worden. Dit omdat het ontwikkelproces van de software niet stil mocht komen te liggen of ernstig vertraagd mocht worden. Echter, het testproces en ontwikkelproces zijn nauw met elkaar verbonden. Er kan gezegd worden dat het ontwikkelproces onderdeel uitmaakt van het testproces. Sommige veranderingen in het testproces moeten vooraf worden gegaan door een verandering in het ontwikkelproces [Pol05]. Een voorbeeld hiervan is de voor het testen benodigde testbasis in de vorm van documentatie. Om dit te realiseren moet er binnen het ontwikkelproces gekeken worden waar deze documentatie gemaakt kan worden en wiens verantwoordelijkheid dit is. Het vergt dus een verandering in het ontwikkelproces.

**R3: Veranderingen kunnen vaak tot weerstand bij medewerkers leiden**

Dit risico is duidelijk naar voren gekomen gedurende het onderzoek. In paragraaf 4.3.3 is een aantal redenen gegeven waarom mensen weerstand bieden tegen verandering. Mensen en organisaties hebben een natuurlijke angst voor verandering en gedijen het best onder een stabiele omgeving. Veranderingen worden altijd als gevaar gezien.
De angst voor verandering van het testproces kan gedeeltelijk worden weggenomen door mensen goed te informeren. Niet alleen door de afstudeerder maar ook door het eigen management.

**R4: Indien de bewoording bij rapportage niet diplomatiek genoeg opgesteld wordt kan dit leiden tot weerstand binnen OnGuard**

Wat men zich duidelijk voor ogen moet houden bij het verbeteren van het testproces is dat deze verbeteringen niet plaats vinden omdat iemand zijn werk niet of niet goed doet. Maar men heeft te maken met mensen; iedere mens is anders. Waar de een de verbeteringen ziet als positieve kritiek ziet de ander het positieve niet van de kritiek en kan zich op de tenen getrapt voelen of de neiging krijgen zichzelf te verdedigen. Sommige mensen hebben immers de neiging om alles persoonlijk op te vatten.
Het is daarom belangrijk dat met ontzettend veel aandacht voor gevoelens van anderen gewerkt wordt. Wanneer iemand zich tekort gedaan of op zijn tenen getrapt voelt kan deze persoon de

veranderingen gaan blokkeren. Dit kan vertraging of zelfs het totaal mislukken van de verbeteringen tot gevolg hebben.
Om dit risico te voorkomen is extra aandacht geschonken aan de manier waarop er met de mensen binnen OnGuard gecommuniceerd is. Dit is uiteraard makkelijker bij het sturen van e-mails of het schrijven van documenten dan bij gesprekken.
Het is een aantal keren voorgekomen dat er iets niet diplomatiek genoeg gebracht is. Vaak kon wat er gezegd was meteen geparafraseerd worden zodat de gesprekspartner zich niet beledigd voelde. Moeilijkheid hierbij is dat iedereen voor zich in de gaten moet houden hoe dingen gebracht worden.

**R5: Het niet toepasbaar / implementeerbaar zijn van de aanbeveling**
Dit is een risico wat funest kan zijn voor het onderzoek. Al zijn de ideeën nog zo goed, de kans bestaat altijd dat er factoren spelen waardoor deze ideeën niet toegepast kunnen worden. Dit risico is ondervangen doordat de afstudeerder zelf een actieve bijdrage heeft geleverd aan de implementatie van de verbeteringen op de korte termijn. Hierdoor kan in een vroeg stadium gekeken worden of de gebruikte methode toepasbaar is.
Uiteraard kan in een later stadium blijken dat men extra middelen nodig heeft in de vorm van bijvoorbeeld mensen om het testproces verder te verbeteren. De aanstelling van een extra tester is hiervan een goed voorbeeld. In dit geval zal OnGuard zelf moeten bepalen in hoeverre zij de voorgestelde aanpassingen door gaat voeren.

### 5.1.6 Onderzoeksvraag en Hypothesen

In deze paragraaf zal antwoord gegeven worden op de in de inleiding gestelde onderzoeksvraag en zullen de gestelde hypothesen bevestigd of ontkracht worden.

**Onderzoeksvraag**
De onderzoeksvraag, na aanpassing door een wending in het onderzoek, luidt:
“Kan met behulp van bruikbare testtechnieken het testproces inzichtelijk en beheersbaar gemaakt worden en daarmee efficiënter uitvoerbaar zijn?”

Het antwoord op deze vraag is een luidkeelse “ja”. Vooral door het gebruik van de TPI methode kunnen er aanpassingen gedaan worden die een directe invloed hebben op het inzichtelijk en beheersbaar zijn van het testproces. Verder is tijdens de analyses van interviews en de workflow duidelijk gemaakt dat de voorgestelde veranderingen allemaal een positief effect hebben op de testdruk. De testdruk is de validatiemetriek die in dit onderzoek gebruikt wordt om de efficiëntie van het testproces aan te tonen.

Helaas is het antwoord op deze vraag nog niet te valideren. Theoretisch gezien zou door het aanbrengen van structuur het testproces inzichtelijk en beheersbaar moeten worden waardoor de testdruk verlaagd wordt. Er zal echter pas een aantal maanden na afronding van het onderzoek duidelijk worden of de aangebrachte veranderingen het gewenste resultaat gebracht hebben. Waar hierbij rekening gehouden moet worden is dat men te maken krijgt met een overgangsperiode. In deze periode zal extra aandacht geschonken moeten worden aan het op peil brengen van de testbasis; de documenten en producten waar de testscenario’s op gebaseerd worden. Ook het vastleggen van testscenario’s zal tijd vergen. Wanneer dit achter de rug is zal men de vruchten

kunnen plukken van een testproces dat loopt als een geoliede machine en duidelijk moeten merken dat verschillende gewenste verbeteringen aan het testproces makkelijker in te voeren zijn. Zodra men het testproces immers beheerst kan men beginnen met het verbeteren van de individuele onderdelen van dit proces.

**Hypothesen**

De verschillende in de inleiding opgestelde hypothesen voor het onderzoek zullen nu behandeld worden.

**Tabel 11: Bevestiging of ontkrachting van de gestelde hypothesen**

| |
|---|
| **Het aanbrengen van structuur is de basis voor een stabiel, effectief en efficiënt testproces.** |
| Deze hypothese kan gedeeltelijk bevestigd worden. Structuur maakt het testproces stabiel. Verder wanneer men een duidelijke structuur in het testproces heeft kan er onderscheid gemaakt worden tussen individuele onderdelen die op hun beurt weer aangepakt kunnen worden om op efficiënte wijze uitgevoerd te worden. Echter of structuur het testproces effectief maakt is niet te bevestigen. Dit heeft namelijk meer te maken met hoe middelen binnen de individuele activiteiten geplaatst worden en de kwaliteit van deze middelen zoals de bekwaamheid van testers. Structuur maakt het wel makkelijker om te bepalen welke middelen op welke activiteit ingezet moeten worden. |
| **Het verbeteren van het testproces heeft niet alleen betrekking op de testafdeling. De testafdeling is in grote mate afhankelijk van omringende afdelingen.** |
| Deze hypothese kan bevestigd worden. Bij verschillende veranderingen is gebleken dat deze niet plaats konden vinden zonder medewerking of verandering bij een andere afdeling. Het testproces is immers een proces wat de gehele organisatie bezig houdt [Pol05]. Wanneer de testafdeling wil valideren of een onderdeel van het systeem voldoet aan gestelde eisen zullen deze eisen eerst opgesteld moeten worden. Het opstellen van deze eisen ligt niet binnen de verantwoordelijkheid van de testafdeling waardoor de testafdeling afhankelijk wordt van de afdeling die daar wel voor verantwoordelijk is. |
| **Automatisering van testen verlaagt de testinspanning** |
| Deze hypothese kan door dit onderzoek niet bevestigd noch ontkracht worden. Literatuur kan deze hypothese wel bevestigen. McConnel stelt in [McC04] dat door automatisering van testen de testen sneller en steeds op dezelfde manier uitgevoerd kunnen worden. Wanneer testen uitgevoerd kunnen worden door een druk op een knop is dit een aanzienlijk lagere testinspanning dan de inspanning die vereist is wanneer men handmatig test. Boven dien kan automatisch testen de testdruk aanzienlijk verlagen [McC04]. |
| **Geautomatiseerd testen is een alleen effectief binnen een gestructureerd testproces.** |
| Deze hypothese kan gedeeltelijk bevestigd worden. Waar in eerste instantie automatisering van testen een belangrijke rol zou spelen is gebleken dat die niet mogelijk was door het ontbreken van een duidelijke structuur in het testproces. Wanneer een test geautomatiseerd wordt betekend dit dat deze telkens op dezelfde manier uitgevoerd gaat worden [McC04]. Hierbij moet men er zeker van zijn dat deze manier ook de correcte manier is. Wanneer men steeds op een andere manier test is het verstandig om eerst op een structurele manier handmatig te testen om, wanneer blijkt dat de test effectief is, deze te automatiseren. |

**De verschillende onderdelen van het testproces kunnen alleen verbeterd worden wanneer het testproces op een structurele en eenduidige manier uitgevoerd wordt.**

Deze hypothese kan bevestigd worden. Wanneer men het testproces niet op een eenduidige manier uitvoert kan men geen goed onderscheid maken tussen verschillende activiteiten. Er moet eerst per activiteit duidelijk zijn wat nodig is om de activiteit uit te voeren en wat de activiteit oplevert. Pas dan kan men deze activiteit individueel verbeteren zonder dat dit effect heeft op de werking van omliggende activiteiten. Wanneer men geen inzicht heeft in de structuur van het testproces is het zeer lastig om afzonderlijke taken te isoleren.

### 5.1.7 Hoe moet OnGuard verder?

Om OnGuard een duw in de goede richting te geven zal er in deze paragraaf invulling gegeven worden aan een advies voor verdere verbetering van het testproces. Hierbij wordt een aantal voorstellen gedaan en worden enkele belangrijke aandachtspunten genoemd die bepalend zijn voor het succesvol doorzetten van de verbetering van het testproces.

**TPI verbeterpunten voor de lange termijn**

Allereerst is er door de afstudeerder een document gemaakt met betrekking tot een werkwijze om de TPI verbeterpunten op lange termijn aan te pakken. Dit document is terug te vinden als bijlage D. In dit document staat per TPI level beschreven welke key area's aangepakt moeten worden alsmede een korte beschrijving van hoe dat te bereiken is. Wat in acht moet worden gehouden is dat er meerdere wegen zijn die naar Rome leiden. De voorgestelde manier hoeft niet de beste manier te zijn. Vaak zal de situatie vanzelf uitwijzen welke methode het best past binnen de bedrijfsvoering van dat moment.
De TPI verbetermeter is een hele goede en praktische leidraad voor het aanpakken van de verbeterpunten. De vertaling naar hoe de oplossing het best binnen OnGuard past moet komen uit het inzicht in de organisatie en processen van de voor het verbeteren van het testproces verantwoordelijke personen.

**Aandachtspunten**

Het belangrijkste punt om mee te geven aan OnGuard is het warm houden van het belang van testen en het verbeteren van het testproces; de zogenaamde Commitment en Motivatie. Zolang het project de steun geniet van het management team en de medewerkers zullen er veranderingen in de organisatie aan gebracht kunnen worden zonder al te veel tegenwerking. Mocht de Commitment en Motivatie wegvallen is het een bijna onmogelijke taak om veranderingen door te voeren omdat er te veel weerstand ontstaat. Een voorstel om de Commitment en Motivatie hoog te houden is om mensen eens een dagdeel mee te laten draaien op een andere afdeling om inzicht te krijgen in en respect te krijgen voor andermans werk. Liefst op een afdeling die afhankelijk is van de input van de afdeling waar de betreffende medewerker vandaan komt. Hierdoor ziet deze medewerker precies wat verderop in het proces de gevolgen zijn van zijn acties.

Ontzettend belangrijk is het om niet constant veranderingen door te voeren maar ook de rust en tijd nemen om te wennen aan de doorgevoerde veranderingen. Op deze manier is er ook de tijd om de veranderingen te evalueren en eventueel bij te sturen. Zie het als het toetsen van de verandering. Hoe eerder men ontdekt dat men met een verandering niet het gewenste resultaat bereikt des te eerder kan men er iets aan doen. Mocht men er later in het proces achter komen dat

ergens aan het begin een fout gemaakt is kost het, net als bij het ontwikkelproces, veel meer moeite om de fout te corrigeren. Het voorstel is dan ook om per TPI level veranderingen door te voeren en dan per key area evalueren of het gewenste resultaat bereikt is door er een tijdje mee te werken. Pas wanneer men er zeker van is dat men een TPI level helemaal onder de knie heeft kan men verder met het aanbrengen van nieuwe veranderingen.

Tijdens het in kaart brengen van de workflow van het testproces bleek dat veel procedures van de betrokken afdelingen niet up-to-date waren. Procedures helpen mee om andere medewerkers en afdelingen inzicht te geven in de uit te voeren taken en verantwoordelijkheden van een rol of afdeling. Het up-to-date maken van deze procedures heeft verder als voordeel dat het voor nieuwe medewerkers makkelijker is bekend te raken met de vereiste manier van werken.

Wat men zich goed moet realiseren is, zoals eerder beschreven in paragraaf 5.1.6, is er sprake van een overgangsfase. Er moet een aantal achterstanden op het gebied van het vastleggen van eisen, testscenario's en andere functionele en technische documentatie worden ingehaald. Ook het wennen aan een nieuwe, onbekende manier van werken kost tijd. Wees kritisch op de aangebrachte veranderingen, maar geef de veranderingen een kans om zichzelf te bewijzen. Uiteindelijk betalen de extra inspanningen die verricht moeten worden zichzelf dubbel en dwars terug in een verlaging van de testdruk en een beter beheersbaar en inzichtelijk testproces.

**Het Test Process Improvement Team**
Als laatste wordt er door de afstudeerder voorgesteld om een Test Process Improvement Team op te richten. Dit team bestaat uit een drietal mensen en heeft een aantal taken. Het TPI-Team heeft de taak om:

- de voortgang van het project voor verbetering van het testproces te monitoren
- commitment en motivatie binnen de organisatie warm te houden
- invulling te geven aan door de afstudeerder en de TPI verbetermeter voorgestelde veranderingen aan het testproces
- de effectiviteit en efficiency van het testproces te monitoren

Een voorstel voor de samenstelling van het TPI-Team is weergegeven in tabel 12. In deze tabel is te zien wie er deel uitmaken van het team en wat hun individuele verantwoordelijkheid is.

**Tabel 12: Samenstelling en verantwoordelijkheden van het Test Process Improvement Team**

| TPI-Team lid | Verantwoordelijkheidsgebied |
|---|---|
| Algemeen directeur | De algemeen directeur is verantwoordelijk voor het aangeven van de bedrijfsdoelen die door het testproces ondersteund moeten worden. Hij heeft de meeste kennis van de bedrijfsvoering en de koers die de organisatie wil varen. |
| Manager van de afdeling Product Management | De manager van de ontwikkelafdeling heeft als taak het aansturen van de ontwikkelaars. In een later stadium van testprocesverbetering zullen ook de programmeurs hun steentje bijdragen aan het testproces door het uitvoeren van zogenaamde developer testing. Hij is verantwoordelijk voor de kwaliteit van de unit testen en andere vormen van white-box testing uitgevoerd door de ontwikkelafdeling. |

| | |
|---|---|
| Manager van de afdeling Service Desk | De manager van de afdeling Service Desk heeft het meeste inzicht in de reactie van de klant. Hij kan aangeven hoeveel fouten er door de klant ontdekt worden en wat de ernst van deze fouten is. Hij heeft verder inzicht door de reactie van klanten in de mate waarin het opgeleverde systeem voldoet aan de door de klanten gestelde eisen. Ook heeft hij de taak om metrieken met betrekking tot bugs bij te houden waarbij rekening gehouden dient te worden met de ernst en de aard van de bugs. Ook het rapporteren van de looptijd van bugs van verschillende ernst valt onder zijn verantwoordelijkheid. |
| Senior Tester | De senior tester is verantwoordelijk voor het efficiënt en effectief uitvoeren van de verschillende testmomenten binnen de testafdeling. Hij is ook degene die kan beoordelen of de testbasis van voldoende kwaliteit is en heeft als taak om metrieken met betrekking tot het de binnen de testafdeling uitgevoerde testmomenten bij te houden; zoals onder andere de testdruk. Verder dient de senior tester de testprioriteiten gesteld door de algemeen directeur te implementeren in de vorm van testscenario's waarmee de mate aangeven kan worden in hoeverre deze doelen gerealiseerd zijn door de software. |

## *5.2 Andere organisaties*

Uit de resultaten van de uitvoering van het onderzoek bij OnGuard is een aantal aandachtspunten te halen die van belang zijn voor andere, soortgelijke organisaties die hun testproces inzichtelijker en beter beheersbaar willen maken. In deze paragraaf worden deze aandachtspunten besproken.

**Verbeteren met een doel**
Zoals in hoofdstuk 6 te lezen is, is het verbeteren van het testproces zonder een doel waar men naar toe wil werken zeer lastig. Het is daarom belangrijk om een doel te stellen voor een bepaalde termijn. Op deze manier kan men gericht werken aan verbeteringen aan het testproces.

**Commitment en Motivatie**
Zoals eerder aangegeven is de volledige steun van het management team nodig om de verbeteringen aan te brengen. De verschillende managers hebben een voorbeeldfunctie en kunnen in sterke mate invloed hebben op de medewerkers van hun afdeling.

Wat heel belangrijk is gebleken tijdens het uitvoeren van het onderzoek is dat iedereen zijn eigen manier heeft om gemotiveerd te worden. Zo zullen managers overtuigd kunnen worden door het verbeteren van het testproces te relateren aan een besparing van de kosten of het stijgen van de winsten. Programmeurs zullen hun steun geven aan verbeteringen wanneer ze voelen dat er waardering is voor het werk wat ze leveren. Voor hen moet alles in het teken staan van wat het hen op kan leveren; hoe een verbetering van het testproces hun werk leuker en makkelijker kan maken. Ditzelfde geldt voor testers. Het verschil tussen programmeurs en testers is in dit geval vaak dat alleen testers in eerste instantie het belang van een gestructureerd testproces inzien.

**Het verbeteren van het testproces is werken met mensen**
Men moet niet vergeten dat de technologie niet de bottleneck is. Het werken met mensen is de grote moeilijkheid in dit verhaal. Zoals reeds eerder aangegeven hebben mensen een natuurlijke angst voor verandering. Deze angst kan weggenomen worden door mensen op een voor hen geschikte manier voor te lichten. Zolang mensen niet begrijpen wat er gaat gebeuren, waarom het moet gebeuren en wat dat voor hen betekend zullen ze nooit volledig meewerken.

Verder kan het soms zo zijn dat mensen bewust tegenwerken omdat ze vinden dat het goed gaat zoals het op dat moment gaat. In hoeverre de grootte van de organisatie hierin een rol speelt is de vraag. In een grote organisatie zullen de medewerkers over het algemeen minder eigen inbreng hebben dan de medewerkers in een kleinere organisatie. Maar mensen in een grote organisatie kunnen dan ook sneller het gevoel krijgen dat iets moet en daarom tegenwerken. Waar bij kleinere organisaties de extra inbreng van de medewerkers een motivatie zou kunnen zijn voor het actief meedenken in het verbeteringsproces.

Er kan gezegd worden dat over het algemeen organisaties er dus goed aan doen om hun medewerkers het gevoel te geven dat ze mee mogen denken over veranderingen binnen de organisatie. Zolang deze medewerkers voldoende geïnformeerd worden over het nut van de veranderingen maakt de grootte van de organisatie in dit geval qua weerstand van medewerkers weinig uit.

**Werken op het eigen tempo**
Het verbeteren van het testproces binnen de organisatie is niet gebonden aan tijd. Het is belangrijk dat iedere organisatie in zijn eigen tempo verbeteringen aanbrengt. Het is geen race om in zo kort mogelijke tijd zo veel mogelijk veranderingen aan te brengen. Veranderingen moeten met beleid en doel worden gedaan. Nadat een verandering heeft plaatsgevonden moet de verandering geëvalueerd worden. Heeft de verandering opgeleverd wat er van verwacht werd of moet er een aanpassing gedaan worden?

Neem vooral ook de tijd om de huidige situatie in kaart te brengen. Het is belangrijk om niet zomaar wat veranderingen aan te brengen. Men moet bewust ervoor kiezen om op een bepaalde plaats een verandering door te voeren. Wanneer men de huidige situatie van het testproces niet correct of volledig in beeld heeft kan een verandering verkeerd uitpakken en kunnen mensen langs elkaar heen gaan werken waardoor het tegenovergestelde effect bereikt wordt van wat men eigenlijk wil bereiken; een inzichtelijk en beheersbaar testproces

**Overgangsfase**
Net als in de vorige paragraaf aangegeven voor OnGuard is de kans groot dat bij andere organisaties een overgangsfase plaats gaat vinden. Men zal een tijdje moeten wennen aan de nieuwe manier van werken en eventuele achterstanden op het gebied van bijvoorbeeld documentatie in moeten halen. Op het moment dat deze fase achter de rug is wordt duidelijk wat de verbeteringen teweeg gebracht hebben. Een gestructureerd en efficiënt testproces werpt zijn vruchten uiteindelijk dubbel en dwars af.

**Geen enkele organisatie is precies hetzelfde**
Een oplossing die bij de ene organisatie goed werkt hoeft niet te werken in het testproces van een andere organisatie. Maak vooral ook gebruik van de vrijheid om het testproces op een eigen manier in de delen. De TPI methode is een leidraad, geen regelgeving. Zolang men rekening houdt met een aantal bewezen testprincipes, als beschreven in paragraaf 2.2, en afspraken duidelijk vastlegt heeft kan binnen iedere organisatie het testproces verbeterd worden.

# 6. Evaluatie

Wanneer een onderzoek afgerond is, is het altijd interessant om terug te kijken op wat er gedaan is en de manier waarop dit is gebeurd. Er zijn vaak dingen waarvan achteraf gezegd wordt: “Als ik het nog eens zou mogen doen, dan zou ik het een en ander toch anders aanpakken”. Zo ook in dit onderzoek.

Er is veel aandacht besteed aan het creëren van commitment en motivatie bij het management team. Er is vanuit gegaan dat de managers de opgedane kennis over zouden kunnen dragen aan de mensen van hun eigen afdelingen. Achteraf blijkt dat het management niet geheel in staat is geweest om de kennis over te brengen op hun medewerkers. Dit heeft er toe geleid dat er relatief veel weerstand is tegen de veranderingen voor verbetering van het testproces. Het persoonlijk, door de afstudeerder, overbrengen van de kennis met betrekking tot de voordelen van een gestructureerd testproces door middel van bijvoorbeeld een presentatie zou veel spanning weg hebben genomen. Dit is een van de redenen waarom is afgesproken een eindpresentatie te houden bij OnGuard. In deze presentatie zullen behalve de resultaten van het onderzoek ook de voordelen voor de verschillende afdelingen aangehaald worden.

Een andere factor die een rol heeft gespeeld in het begin van het onderzoek is het feit dat er niet snel genoeg ingesprongen is op de bedrijfsdoelen. Dit is gelukkig wel opgepakt maar heeft een kleine vertraging opgeleverd. Het eerder inspringen op bedrijfsdoelen zou het verbeteren van het testproces meer lijn gegeven hebben.

Zoals ook vermeld in paragraaf 5.1.4 zou er vaker iets diplomatieker tewerk gegaan kunnen worden. Het is gebleken ontzettend lastig te zijn om te allen tijde op te letten niemand op de tenen te trappen. Een aantal keren is gebleken dat ondanks dat er nagedacht is hoe bepaalde dingen zo diplomatiek mogelijk te brengen waren, mensen het toch nog persoonlijk op konden vatten. Dit heeft wellicht tot onnodige tegenwerking geleid.

Wat erg jammer is, is het feit dat de tijd te kort is om de invoering van het testproces helemaal te verwezenlijken. Validatie van de resultaten van de verbeteringen  kan pas over enkele maanden plaats vinden.

Uiteraard ging er ook heel veel ontzettend goed. De medewerkers van OnGuard hebben ontzettend veel steun en medewerking getoond aan het onderzoek. Ondanks dat er iemand het hele testproces binnenstebuiten gehaald heeft bleven deze mensen toch vriendelijk en oprecht geïnteresseerd.

Het gebruik van een testproces maturity referentiemodel heeft een nuttige en praktische impuls gegeven aan de voorgang van het onderzoek. Door het gebruik van Test Process Improvement referentiemodel is er bij de afstudeerder geen twijfel dat men binnen OnGuard de ingeslagen weg op eigen kracht kan blijven vervolgen. Het testproces van OnGuard zou op den duur een voorbeeld en streven kunnen zijn waar andere bedrijven naar toe willen werken.

**Toekomstig werk**
Na afronding van het onderzoek is er nog een aantal zaken wat interessant is om uit te zoeken en een nuttige bijdragen kan leveren aan het verder efficiënter maken van het testproces.
Uiteraard moet men bij OnGuard de weg blijven volgen die door uitvoering van dit onderzoek is ingeslagen. Maar behalve er is een aantal dingen dat een positieve invloed kan hebben op de verbetering van het testproces.

Zo zou er onderzocht kunnen worden welke processen het best geautomatiseerd kunnen worden om de testdruk te verlagen en op welke manier kan dit gebeuren. Dit zou een goede voorbereiding zijn voor het behalen van hogere TPI levels.
Verder is het zeer nuttig om een praktische manier te vinden waarop meer inzicht te krijgen is in de impact van veranderingen op het systeem van OnGuard. Vooral met het oog op developer testing in de vorm van unit testing. In hoeverre draagt unit testing bij aan het meten van de impact van een verandering aan het systeem?
Ook kan er onderzocht worden welke middelen en tools gebruikt kunnen worden om het testproces van OnGuard nog efficiënter te maken.

# Bibliografie


[Bas03] Bass Len, Clements Paul, Kazman Rick, "Software Architecture in Practice: Second Edition", Addison Wesley, 2003, ISBN 0321154959

[Bec01] Beck, Kent, "Aim, Fire", Three Rivers Institute, Merlin, Oregon, Verenigde Staten, September 2001

[Bec99] Beck Kent, "Extreme Programming Explained", Addison Wesley, 1999, ISBN 0201616416

[Boe76] Boehm Barry W., "Software Engineering", IEEE Trans. Computer, C-25,12,1226-1241, 1976

[Fre03] Freeman, Steve et al., "Retrofitting unit tests", M3P, Londen, Engeland, 2002

[Irw00] Irwin Lynne H., "Technology Transfer", Cornel University Local Roads Program, 2000

[Koh04] Kohl, Jonathan, "Pair Testing: How I brought developers into the test lab", StickyMinds.com: Better Software, Verenigde Staten, 2004

[Lau02] Lauesen Soren, "Software Requirements: Styles and Techniques", Addison-Wesley, 2002, ISBN 0201745704

[McC04] McConnell Steve, "Code Complete, Second edition", MicroSoft Press, Verenigde Staten, 2004

[Pol05] Pol Martin, Teunissen Ruud, Veenendaal van Erik, "Testen volgens TMap®: 2e druk", Uitgeverij Tutein Nolthenius, 2005, ISBN 9072194586

[Swi00] Swinkels Ron, "A comparison of TMM and other Test Process Improvement Models", Frits Philips Institute, Nederland, 2000

[Voa98] Voas Jeffrey M., Miller Keith W., "Improving the software development process using testability research", Reliable Software Technologies Corp., Arlington, Virginia, Verenigde Staten, Department of Computer Science, Williamsburg, Virginia, Verenigde Staten, 1998

[Wil99] Williams Laurie A., Kessler Robert R., "All I really need to know about Pair Programming I learned in kindergarten", University of Utah, Utah, Verenigde Staten, 1999

[Wil00] Williams Laurie, Kessler Robert R., Cunningham Ward, Jeffries Ron, "Strengthening the case for Pair Programming", IEEE software, 2000

[WiPe-1w] Unit Test wikipedia page, adress: http://en.wikipedia.org/wiki/Unit_test, laaste bezoek op 10-07-06

[Ync01] Ynchausti, Randy A., "Integrating Unit Testing Into A Software Development Team's Process", Monster Consulting Inc., Bountyful, Utah, Verenigde Staten, 2001

[Zel06] Zeller, Andreas, "Why programs fail: a guide to systematic debugging", Morgan Kaufmann, 2006, ISBN 1558608664

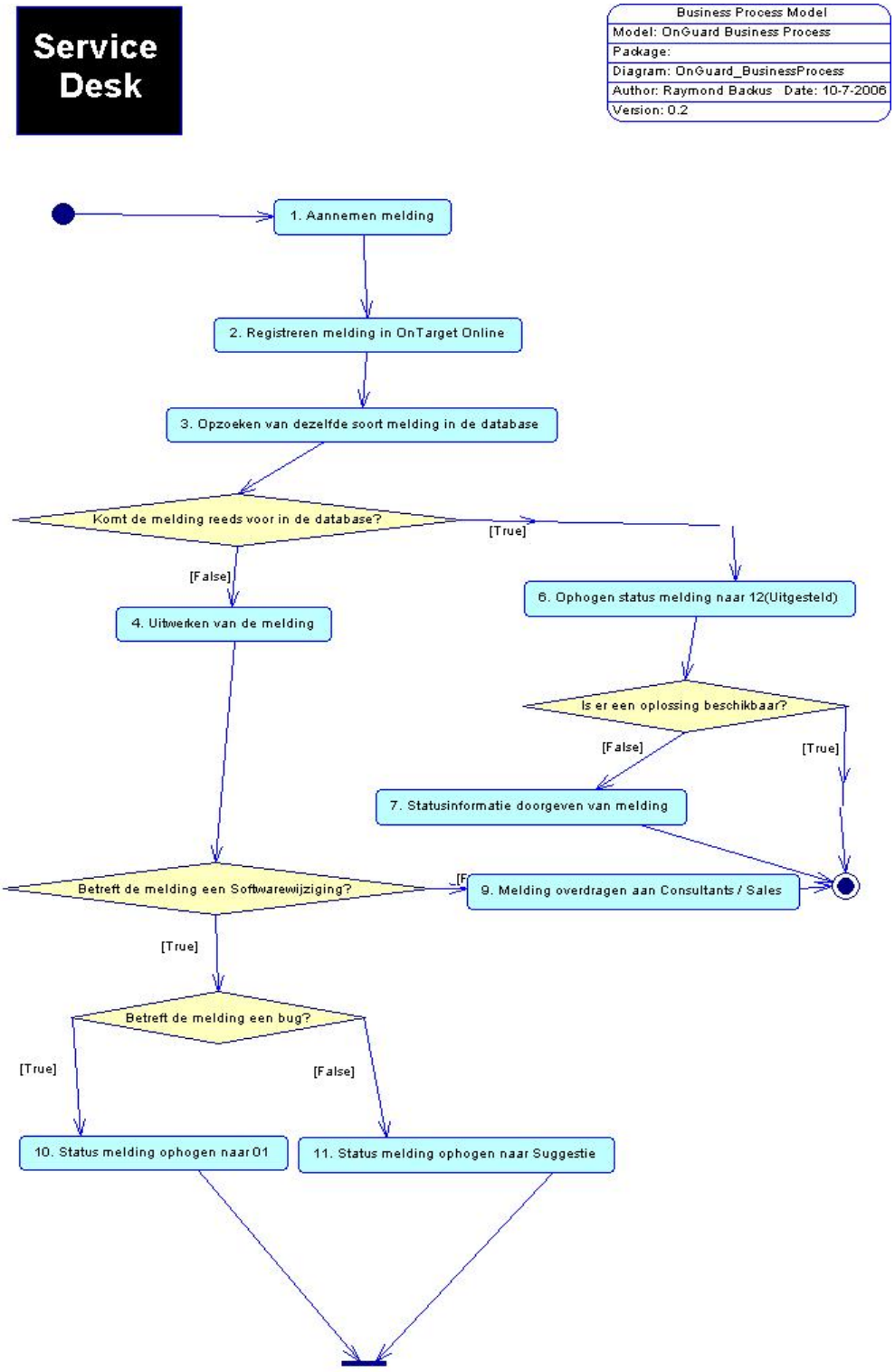
# A. Eerste Bijlage: Mission Statement

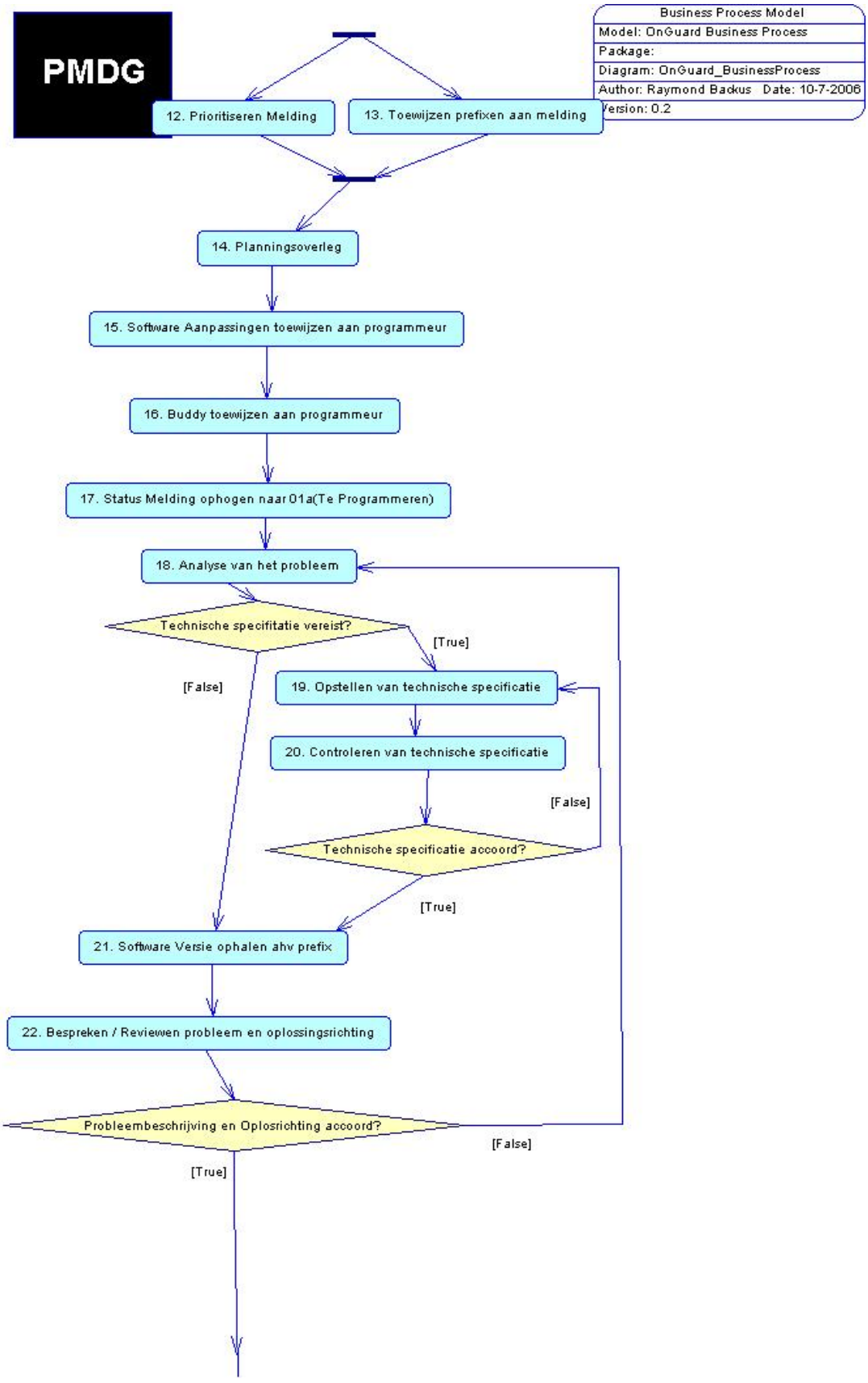
**Mission Statement 2004 – 2009**
OnGuard's mission is to become the leading supplier – both in terms of installed licence value and customer base – of Credit and Collections solutions in the European market. We will achieve this through a combination of a direct sales force and close alliances with professional partners, to supply a combination of standard software and added value services that offer our customers a high quality solution to implement and maintain an efficient and financially beneficial credit and collections policy. Our product development strategy is driven by end user requirements and provides and end-user solution for credit and collection management using proven technology as a means to provide that solution.

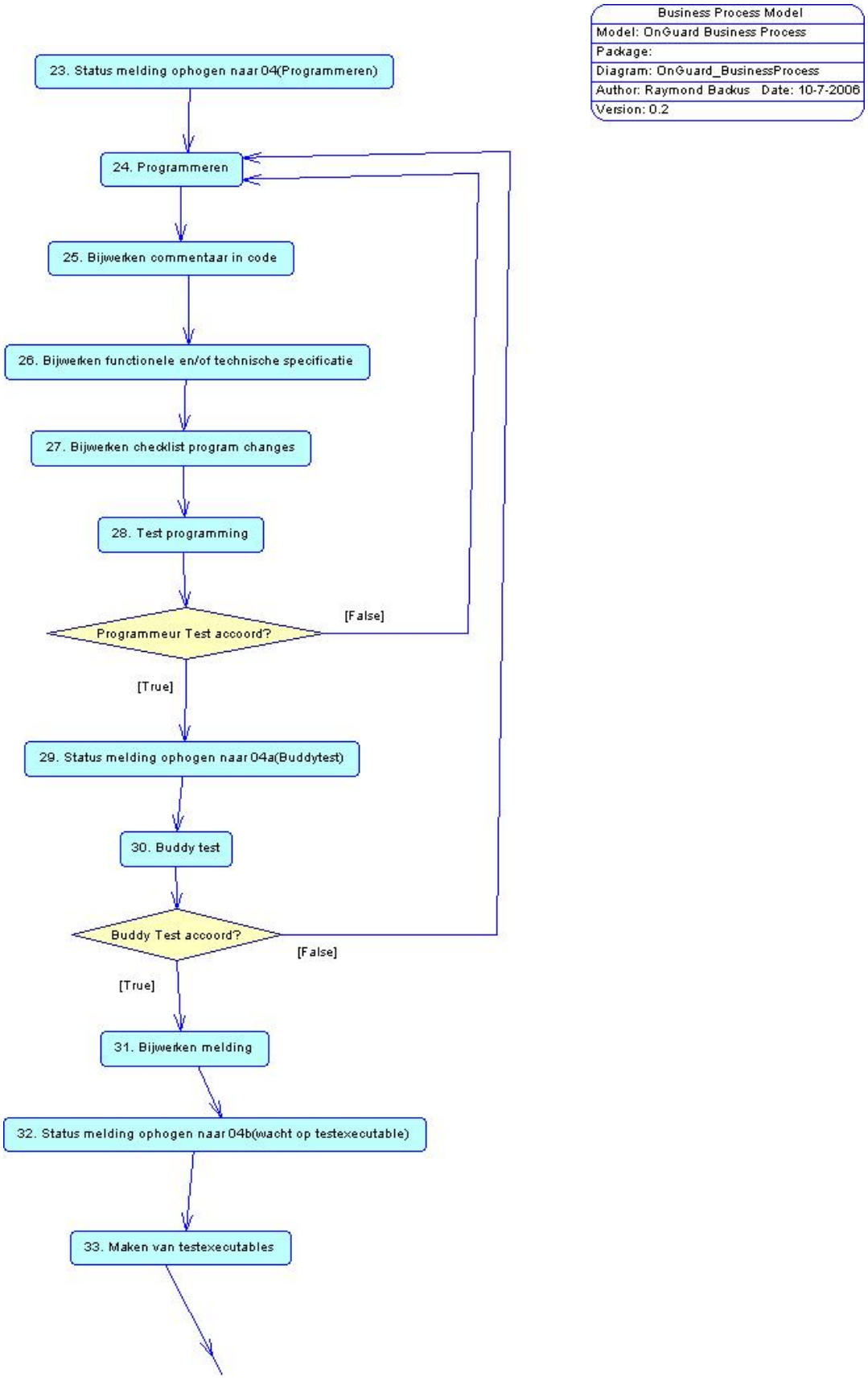
OnGuard provides a work environment which will stimulate the highest ethical and professional standards; in which motivated and qualified professionals can work with pleasure. An environment that provides its employees with financial security to enable them to enjoy their lives to the full; whilst acting for our customers as a professional and complete supply partner.

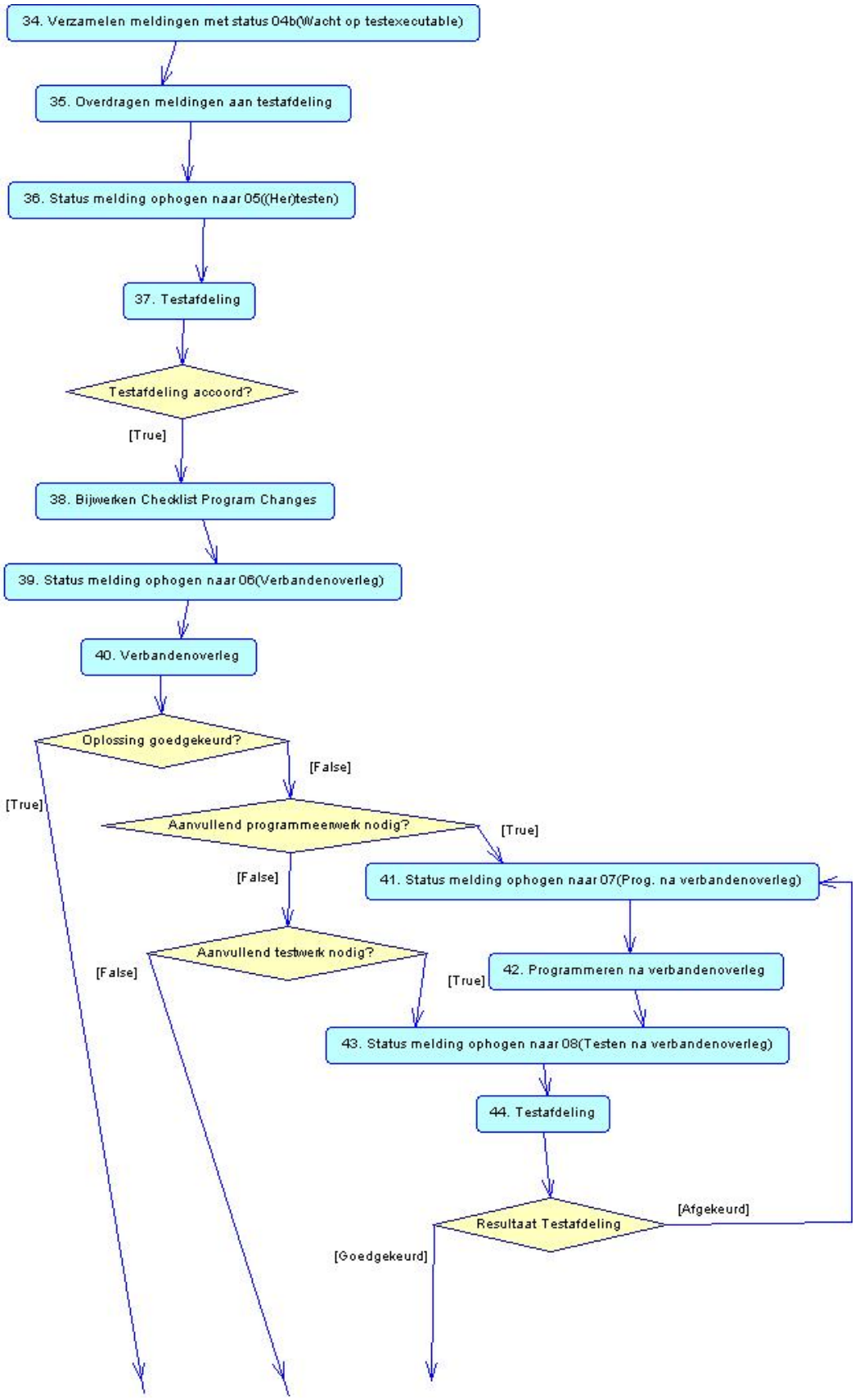
OnGuard Nederland B.V.
Kasteel Nederhorst
Slotlaan 3 te Nederhorst den Berg
Telefoon: (0249) 25 66 66

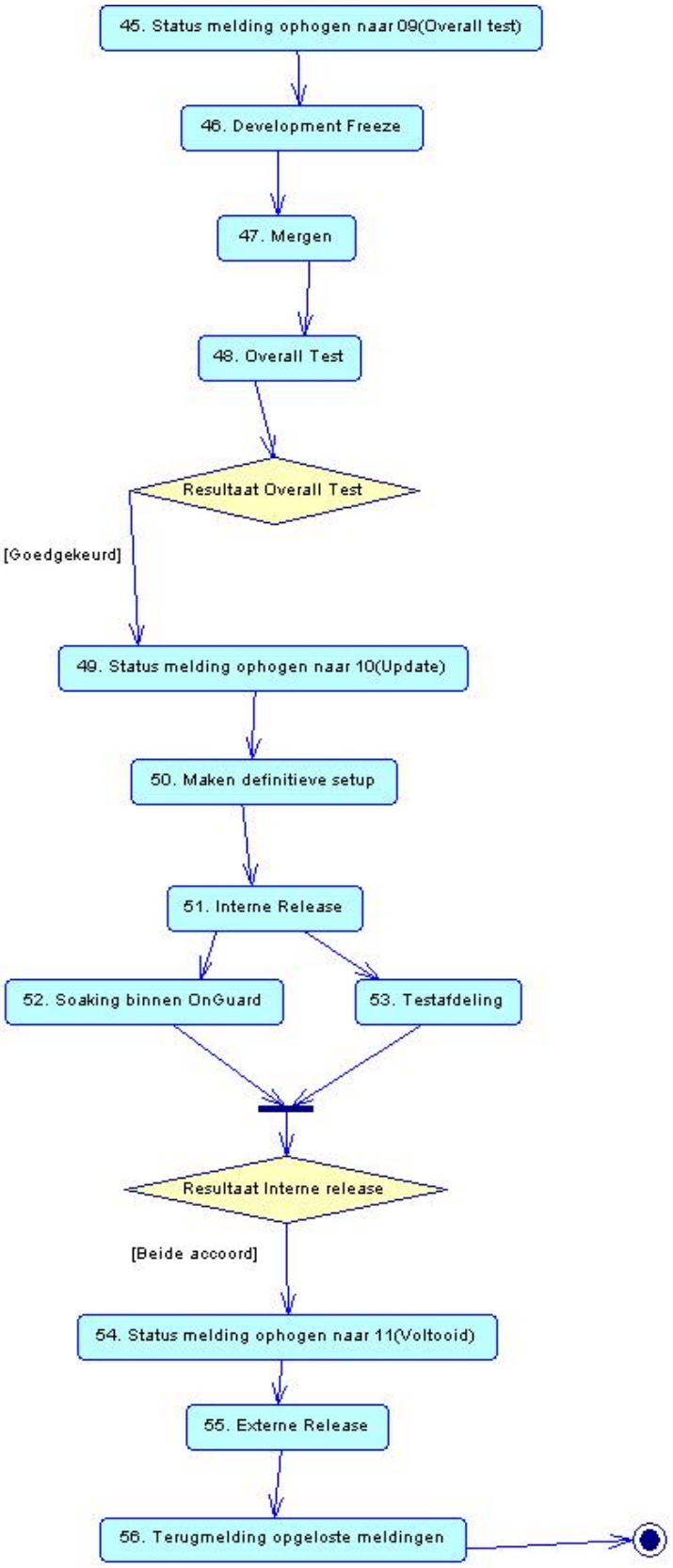
# B. Tweede bijlage: Workflow van het testproces




PMDG
Business Process Model
Model: OnGuard Business Process
Package:
Diagram: OnGuard_BusinessProcess
Author: Raymond Backus Date: 10-7-2006
Version: 0.2
12. Prioritiseren Melding
13. Toewijzen prefixen aan melding
14. Planningsoverleg
15. Software Aanpassingen toewijzen aan programmeur
16. Buddy toewijzen aan programmeur
17. Status Melding ophogen naar 01a(Te Programmeren)
18. Analyse van het probleem
Technische specifitatie vereist?
[True]
[False]
19. Opstellen van technische specificatie
20. Controleren van technische specificatie
[False]
Technische specificatie accoord?
[True]
21. Software Versie ophalen ahv prefix
22. Bespreken / Reviewen probleem en oplossingsrichting
Probleembeschrijving en Oplosrichting accoord?
[False]
[True]


Business Process Model
Model: OnGuard Business Process
Package:
Diagram: OnGuard_BusinessProcess
Author: Raymond Backus Date: 10-7-2006
Version: 0.2
23. Status melding ophogen naar 04(Programmeren)
24. Programmeren
25. Bijwerken commentaar in code
26. Bijwerken functionele en/of technische specificatie
27. Bijwerken checklist program changes
28. Test programming
Programmeur Test accoord?
[False]
[True]
29. Status melding ophogen naar 04a(Buddytest)
30. Buddy test
Buddy Test accoord?
[False]
[True]
31. Bijwerken melding
32. Status melding ophogen naar 04b(wacht op testexecutable)
33. Maken van testexecutables


34. Verzamelen meldingen met status 04b(Wacht op testexecutable)
35. Overdragen meldingen aan testafdeling
36. Status melding ophogen naar 05((Her)testen)
37. Testafdeling
Testafdeling accoord?
[True]
38. Bijwerken Checklist Program Changes
39. Status melding ophogen naar 06(Verbandenoverleg)
40. Verbandenoverleg
Oplossing goedgekeurd?
[False]
[True]
Aanvullend programmeerwerk nodig?
[True]
[False]
41. Status melding ophogen naar 07(Prog. na verbandenoverleg)
Aanvullend testwerk nodig?
[False]
[True]
42. Programmeren na verbandenoverleg
43. Status melding ophogen naar 08(Testen na verbandenoverleg)
44. Testafdeling
Resultaat Testafdeling
[Afgekeurd]
[Goedgekeurd]


45. Status melding ophogen naar 09(Overall test)
46. Development Freeze
47. Mergen
48. Overall Test
Resultaat Overall Test
[Goedgekeurd]
49. Status melding ophogen naar 10(Update)
50. Maken definitieve setup
51. Interne Release
52. Soaking binnen OnGuard
53. Testafdeling
Resultaat Interne release
[Beide accoord]
54. Status melding ophogen naar 11(Voltooid)
55. Externe Release
56. Terugmelding opgeloste meldingen

# C. Derde bijlage: TPI planning voor de korte termijn

Aan te pakken TPI punten, Korte termijn:

## TPI verbeterpunten, korte termijn

Uit een meeting met als doel het selecteren van TPI verbeterpunten voor de korte termijn is een aantal punten gekomen. Bij deze meeting waren de volgende personen aanwezig:

- Stagiair
- Manager PMDG
- Senior Tester

Om de meeting soepel te laten verlopen en de keuze voor de TPI verbeterpunten op een gefundeerde wijze te maken is voorafgaand aan deze meeting een tweetal documenten met betrekking tot TPI aan de genodigden verstrekt. Het betreft hier de volgende documenten:

- Test Process Improvement, leidraad voor stapsgewijs beter testen, document met algemene informatie betreffende TPI
- TPI verbetermeter, Excel-document ingevuld naar de situatie van OnGuard.

Belangrijk bij deze meeting is dat OnGuard zelf bepaald welke punten ze aan willen pakken. Om het niet op natte-vinger-werk aan te laten komen is besloten om twee doelen te stellen. Eén doel voor de korte termijn en één doel voor de lange termijn. Deze doelen hebben betrekking op het te behalen TPI niveau.

Het doel voor de lange termijn wordt vastgesteld op het moment dat de planning betreffende de realisatie van het doel voor de korte termijn gereed is. Het doel voor de korte termijn is als volgt:

- Behalen van TPI niveau 1

Om aan dit doel te voldoen moeten de volgende TPI key area's verbeterd worden:

- Teststrategie
- Toepassing fasering
- Specificatietechnieken
- Commitment en Motivatie
- Bevindingenbeheer
- Testprocesbeheer

Per te verbeteren TPI key area wordt aangegeven waarom verbetering nuttig is en hoe deze verbetering binnen OnGuard te bereiken is.

Voor al deze TPI key area's geldt dat ze door de TPI verbetermeter aangegeven worden als kritieke gebieden. Na nader onderzoek met betrekking tot deze key area's is gebleken dat ze allen te maken hebben met het aanbrengen van structuur in de testorganisatie. Deze gebieden komen overeen met de voorgestelde verbeterpunten.

De planning van de uit te voeren handelingen bij de punten wordt gedaan in overleg met de bij de betreffende punten betrokken personen.

## *Teststrategie*

Doel: Level A

**Hoe**?

- Inzicht in verschillende soorten testen
- Inzicht in testspecificatietechnieken → Key Area specificatietechnieken
- Wat is een teststrategie en waar bestaat deze uit?
- Opstellen van een formele strategie
- Documenteren van overige punten behorende bij level A

**Planning**

| Taak | Hoe | Afdeling | Gepland aantal uren |
|---|---|---|---|
| Inzicht verschaffen in testsoorten | Lezen TMap hoofdstuk 8 en 9 | Testing | |
| Testspecificatietechnieken | Lezen TMap hoofdstuk 15 | Testing | |
| *Wat is een teststrategie | Verzamelen informatie TMap en opzoeken definitie teststrategie | Testing | |
| Opstellen formele strategie | - | Testing | |
| Opstellen template voor risicoanalyse | - | Testing | |

## *Toepassing fasering*

Doel: Level A

**Hoe?**

- Maken checklist per fase op de testafdeling
- Communicatie tussen testafdeling en ontwikkelafdeling verbeteren zodat de testafdeling in staat is een planning te maken. De ontwikkelafdeling zal hiervoor extra aandacht aan planning op de eigen afdeling moeten schenken.
- Correct gebruik van statusovergangen in OnTarget Online. Meer controle op statusovergangen.

**Planning**

| Taak | Hoe | Afdeling | Gepland aantal uren |
|---|---|---|---|
| Maken checklist per fase | - | Testing | |
| Optimaliseren planning ontwikkelafdeling | Afspraken maken over het gebruik van statusovergangen en planning geregeld overdragen aan testafdeling | Ontwikkeling | |

## *Specificatietechnieken*

Doel: Level A

### Hoe?

- Inlezen in specificatietechnieken
- Vaststellen van te gebruiken specificatietechnieken testafdeling
- Vaststellen van te gebruiken specificatietechnieken ontwikkelafdeling

### Planning

| Taak | Hoe | Afdeling | Gepland aantal uren |
|---|---|---|---|
| Inlezen in specificatietechnieken | Lezen TMap hoofdstuk 15 | Testing | |
| Vaststellen specificatietechnieken Testing | - | Testing | |
| Vaststellen specificatietechnieken Ontwikkeling | - | Ontwikkeling | |

## *Commitment en Motivatie*

Doel: Level A

### Hoe?

- Realiseren mogelijkheid tot planning voor de testafdeling. Op het moment dat de testafdeling zijn taken kan plannen is het testproces op de testafdeling inzichtelijker en beter te managen door de manager PMDG
- Inzicht verkrijgen in specificatietechnieken → Key area specificatietechnieken
- Inzicht verkrijgen in verschillende soorten van testen, de manier waarop en wanneer ze gebruikt worden en wat ze opleveren
- Het testproces inzichtelijk maken voor programmeurs, programmeurs kennis laten maken met testing en de testers uit de ogen laten kijken van een programmeur

### Planning

| Taak | Hoe | Afdeling | Gepland aantal uren |
|---|---|---|---|
| Realiseren planning testafdeling | - | Testing/Ontwikkeling | |
| Inzicht specificatietechnieken | Lezen TMap hoofdstuk 15 | Testing | |
| Inzicht testsoorten | Lezen TMap hoofdstuk X | Testing | |
| Wederzijds respect Testing/Development | Pair Testing, presentatie testafdeling | Testing/Ontwikkeling | |

## *Bevindingenbeheer*

Doel: Level A

**Hoe?**

- Toevoegen ernstcategorie aan databasevelden van inkomende meldingen

**Planning**

| Taak | Hoe | Afdeling | Gepland aantal uren |
|---|---|---|---|
| Toevoegen ernstcategorie | Zonder DB aanpassing:<br>Vermelden ernstcategorie in meldingsinformatieveld.<br>Met DB aanpassing:<br>Hernoemen veld “prioriteit” naar “ernstcategorie” en toevoegen 3 radio buttons in het HelpDeskV5 systeem onder prioriteit. Deze radiogroup krijgt vanzelfsprekend de caption “Ernstcategorie”. Prioriteit wordt niet meer bepaald door Servicedesk maar door Release Manager en/of Manager PMDG | Testing | |

## *Testprocesbeheer*

Doel: Level A

**Hoe?**

- Maken van een template voor het testplan
- Opstellen van het testplan

**Planning**

| Taak | Hoe | Afdeling | Gepland aantal uren |
|---|---|---|---|
| Template voor het testplan | Op basis van een voorbeeld testplan uit TMap bijlage A kan een template worden opgesteld | Testing | |
| Opstellen testplan | Testplan moet ingevuld worden voor het testproces en de software van OnGuard | Testing | |

## *Algehele planning, korte termijn*

Voorlopige volgorde:

Bevindingenbeheer, Commitment en Motivatie, Testspecificatietechnieken, Testprocesbeheer, Teststrategie en als laatste Toepassing fasering.

# D. Vierde bijlage: TPI planning voor de lange termijn

Voorstel aan te pakken TPI punten, Lange termijn:

## TPI verbeterpunten, lange termijn

Het doel wat voorgesteld wordt voor verbetering van het testproces op de lange termijn is beheersing van het testproces. Beheersing van het testproces wordt bereikt wanneer het totale TPI level van 5 bereikt is.

Voor het bereiken van TPI level 5 wordt er van uit gegaan dat TPI level 1 reeds behaald is. Vervolgens is level 5 te behalen in vier stappen. Elke stap beslaat de upgrade van het TPI level met 1 niveau. De stappen zijn dus:

- Bereiken TPI level 2
- Bereiken TPI level 3
- Bereiken TPI level 4
- Bereiken TPI level 5

Per niveau zal beschreven worden welke key area's betrokken zijn bij de verbetering en wat er gedaan moet worden om dit gebied te verbeteren.

### *TPI level 2*

Voor het bereiken van TPI level 2 moeten de volgende key area's aangepakt worden:

- Testware
- Overleg

### Testware

Doel: Level A

**Hoe?**

**Planning**

| TaakNr | Taak | Hoe | Afdeling | Gepland aantal uren |
|---|---|---|---|---|
| 1.1.1 | Opstellen procedure voor aanleveren, registreren, archiveren en raadplegen | Afspraken maken over hoe met testware omgegaan wordt en deze vastleggen in een document | Testing | 8 |
| 1.1.2 | Opstellen procedure voor overdracht van testware | Afspraken maken over de overdracht van testware | Testing | 4 |

### Overleg
Doel: Level A

**Hoe?**

**Planning**

| TaakNr | Taak | Hoe | Afdeling | Gepland aantal uren |
|---|---|---|---|---|
| 1.2.1 | Vastleggen periodiek overleg | Afspraken maken voor periodiek overleg binnen testafdeling | Testing | 1 |
| 1.2.2 | Fixeren testplan | Fixeren van het testplan. Wijzigingen moeten worden gecommuniceerd | Testing | 1 |

## *TPI Level 3*
Om TPI level 3 te bereiken moeten de volgende key area's aangepakt worden:
- Begroting en Planning
- Specificatietechnieken
- Test Automatisering
- Testfuncties en Opleidingen
- Testprocesbeheer

### Begroting en Planning
Doel: Level A

**Hoe?**

**Planning**

| TaakNr | Taak | Hoe | Afdeling | Gepland aantal uren |
|---|---|---|---|---|
| 2.1.1 | Bewaking planning en begroting | Periodieke meeting plannen met manager PM | PM | 1 |
| 2.1.2 | Onderbouwen begroting en planning | Testplan | Testing | 1 |

### Specificatietechnieken
Doel: Level B

**Hoe?**

**Planning**

| TaakNr | Taak | Hoe | Afdeling | Gepland aantal uren |
|---|---|---|---|---|
| 2.2.1 | Opstellen formele specificatietechnieken | Testplan | Testing | 4 |
| 2.2.2 | Dekkingsgraad black-box testing vaststellen | Onderzoek dekkingsgraad black-box testing tov testbasis | Testing | 16 |
| 2.2.3 | Uniforme werkwijze | Testplan, fasering | Testing | (0) |

## Test automatisering

Doel: Level A

**Hoe?**

**Planning**

| TaakNr | Taak | Hoe | Afdeling | Gepland aantal uren |
|---|---|---|---|---|
| 2.3.1 | Afspraken waar automatisering toe te passen | Meeting | PM | 2 |
| 2.3.2 | Onderzoek automatisering | Onderzoek naar welke onderdelen te automatiseren zijn | Testing | 24 |
| 2.3.3 | Presentatie testautomatisering voor MT | Presentatie | Testing | 8 |

## Testfuncties en Opleidingen

Doel: Level A

**Hoe?**

**Planning**

| TaakNr | Taak | Hoe | Afdeling | Gepland aantal uren |
|---|---|---|---|---|
| 2.4.1 | Onderscheiden testmanager / tester functies | | PM / Testing | 1 |
| 2.4.2 | Vastleggen taken en verantwoordelijkheden testfuncties | | PM / Testing | 8 |
| 2.4.3 | Testkennis op peil brengen bij medewerkers Testing | | Testing | 8 |

## Testprocesbeheer
Doel: Level B

**Hoe?**

**Planning**

| TaakNr | Taak | Hoe | Afdeling | Gepland aantal uren |
|---|---|---|---|---|
| 2.5.1 | Opstellen procedure planningsbewaking | | Testing | 4 |

## *TPI Level 4*
Voor het bereiken van TPI level 4 moeten de volgende key area's aangepakt worden:
- Toepassing Fasering
- Statische Testtechnieken
- Toepassingsgraad van de Methodiek
- Overleg
- Rapportage
- White-box Testsoorten

## Toepassing fasering
Doel: Level B

**Hoe?**

**Planning**

| TaakNr | Taak | Hoe | Afdeling | Gepland aantal uren |
|---|---|---|---|---|
| 3.1.1 | Toevoegen fases: Voorbereiding en Afronding | Aanpassen checklist fasering | Testing | |
| 3.1.2 | Aanpassen procedure | Aanpassen procedure testing en testplan | Testing | |

## Statische Testtechnieken
Doel: Level A

**Hoe?**

**Planning**

| TaakNr | Taak | Hoe | Afdeling | Gepland aantal uren |
|---|---|---|---|---|
| 3.2.1 | Checklist voor de intake van testobject | | Testing | 4 |

## Toepassingsgraad van de Methodiek

Doel: Level A

**Hoe?**

**Planning**

| TaakNr | Taak | Hoe | Afdeling | Gepland aantal uren |
|---|---|---|---|---|
| 3.3.1 | Volgen van het testplan | | Testing | |
| 3.3.2 | Afspraken mbt volgen testplan en controle daarvan | | Testing / PM | |

## Overleg

Doel: Level B

**Hoe?**

**Planning**

| TaakNr | Taak | Hoe | Afdeling | Gepland aantal uren |
|---|---|---|---|---|
| 3.4.1 | Zie verbetermeter | | | |
| 3.4.2 | | | | |

## Rapportage

Doel: Level B

**Hoe?**

**Planning**

| TaakNr | Taak | Hoe | Afdeling | Gepland aantal uren |
|---|---|---|---|---|
| 3.5.1 | Vaststellen rapportage | | | |
| 3.5.2 | | | | |

### White-box Testsoorten
Doel: Level A

**Hoe?**

**Planning**

| TaakNr | Taak | Hoe | Afdeling | Gepland aantal uren |
|---|---|---|---|---|
| 3.6.1 | Opzetten testplan met whitebox testen | | | |

## *TPI Level 5*
De volgende key area's moeten onder handen genomen worden om TPI level 5 te bereiken:
- Metrics
- Commitment en Motivatie
- Bevindingenbeheer
- Testware Beheer

### Metrics
Doel: Level A

**Hoe?**

**Planning**

| TaakNr | Taak | Hoe | Afdeling | Gepland aantal uren |
|---|---|---|---|---|
| 4.1.1 | Bijhouden Input-metrieken | Opstellen welke metrieken worden bijgehouden en dit naleven | Testing | |
| 4.1.2 | Bijhouden output-metrieken | Opstellen welke metrieken worden bijgehouden en dit naleven | Testing | |
| 4.1.3 | Gebruiken metrieken bij rapportage | - | Testing | |

## Commitment en Motivatie

Doel: Level B

**Hoe?**

**Planning**

| TaakNr | Taak | Hoe | Afdeling | Gepland aantal uren |
|---|---|---|---|---|
| 4.2.1 | Management wil inzicht in diepgang en kwaliteit | Meetings / Rapportage | Testing | |
| 4.2.2 | Managementaansturing van testproces op tijd, geld en kwaliteit | Planning / begroting van testafdeling | PM | |
| 4.2.3 | Testing heeft inspraak op volgorde oplevering Development | | PM / Testing | |
| 4.2.4 | Adviezen van testing worden bij project-overleg besproken | | PM / Testing | |

## Bevindingenbeheer

Doel: Level B

**Hoe?**

**Planning**

| TaakNr | Taak | Hoe | Afdeling | Gepland aantal uren |
|---|---|---|---|---|
| 4.3.1 | Toevoegen testgeval aan melding | | PM / Testing | |
| 4.3.2 | Toevoegen test aan melding | | PM / Testing | |
| 4.3.3 | Vermelden testbasis + versie in melding | | PM / Testing | |
| 4.3.4 | Alle statusovergangen van melding moeten datum / tijd bevatten | | PM / Testing | |
| 4.3.5 | Aanstellen persoon die verantwoordelijk | Meldingen moeten juist, consequent en consistent ingevuld worden en | PM / Testing | |

| | | | | |
|---|---|---|---|---|
| | is voor meldingen | gecontroleerd worden door een daarvoor verantwoordelijk gemaakte persoon | | |

## Testware Beheer

Doel: Level B

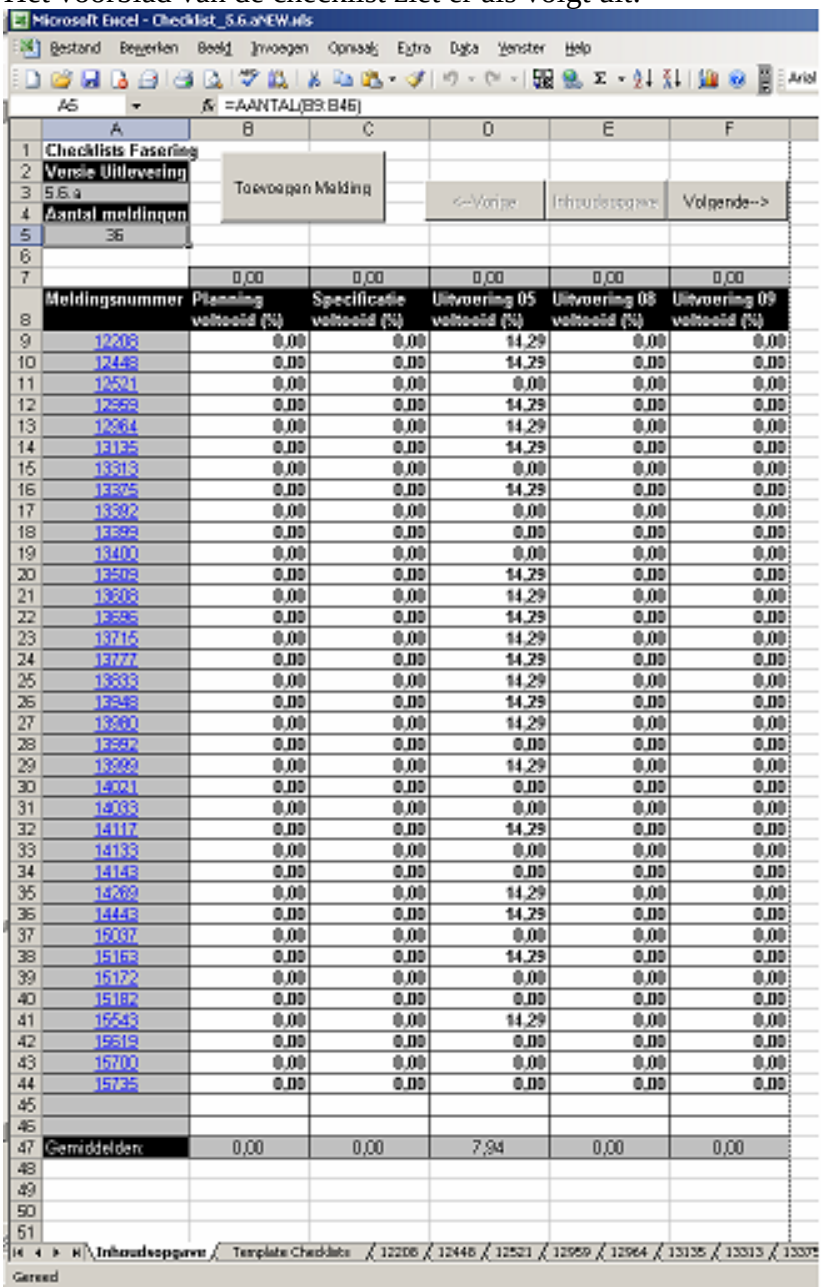
**Hoe?**

**Planning**

| TaakNr | Taak | Hoe | Afdeling | Gepland aantal uren |
|---|---|---|---|---|
| 4.4.1 | Testware wordt extern beheerd (per projects) | | Testing | |
| 4.4.2 | Testafdeling wordt op de hoogte gesteld van wijzigingen in de testware | | | |

# E. Vijfde Bijlage: Checklist Fasering Testen

Om de testafdeling te helpen met het gestructureerd en eenduidig volgen van het testproces binnen de afdeling is een checklist opgesteld. Deze checklist stelt de tester in staat om per melding alle vereisten stappen af te vinken.
Het voorblad van de checklist ziet er als volgt uit:



Microsoft Excel - Checklist_5.6.a(NEW).xls

Bestand Bewerken Beeld Invoegen Opmaak Extra Data Venster Help

A5 =AANTAL(B9:B46)

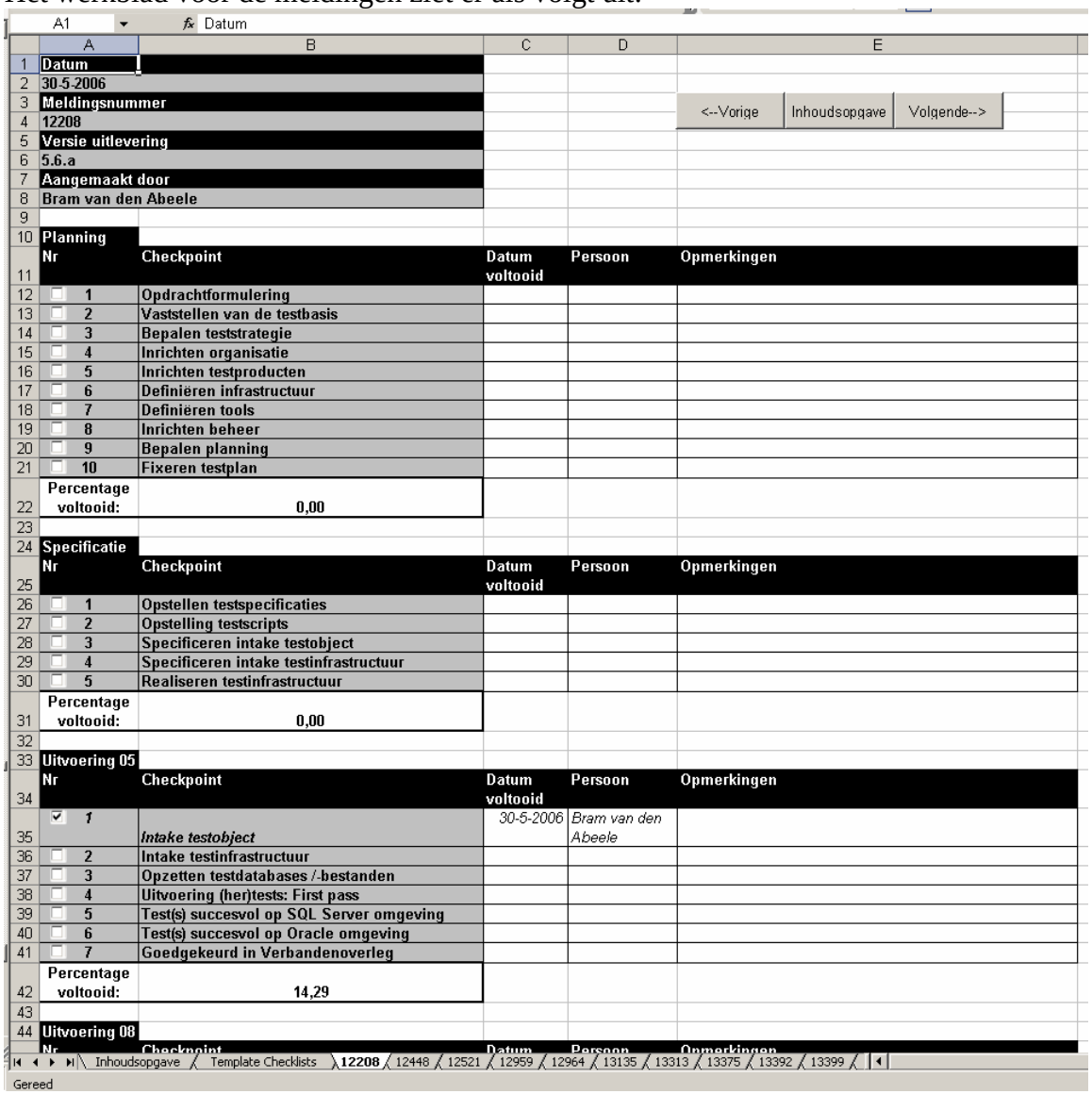
| | A | B | C | D | E | F |
|---|---|---|---|---|---|---|
| 1 | Checklists Fasering | | | | | |
| 2 | Versie Uitlevering | Toevoegen Melding | | <--Vorige | Inhoudsopgave | Volgende--> |
| 3 | 5.6.a | | | | | |
| 4 | Aantal meldingen | | | | | |
| 5 | 36 | | | | | |
| 6 | | | | | | |
| 7 | | 0,00 | 0,00 | 0,00 | 0,00 | 0,00 |
| 8 | Meldingsnummer | Planning voltooid (%) | Specificatie voltooid (%) | Uitvoering 05 voltooid (%) | Uitvoering 08 voltooid (%) | Uitvoering 09 voltooid (%) |
| 9 | 12208 | 0,00 | 0,00 | 14,29 | 0,00 | 0,00 |
| 10 | 12448 | 0,00 | 0,00 | 14,29 | 0,00 | 0,00 |
| 11 | 12521 | 0,00 | 0,00 | 0,00 | 0,00 | 0,00 |
| 12 | 12959 | 0,00 | 0,00 | 14,29 | 0,00 | 0,00 |
| 13 | 12964 | 0,00 | 0,00 | 14,29 | 0,00 | 0,00 |
| 14 | 13135 | 0,00 | 0,00 | 14,29 | 0,00 | 0,00 |
| 15 | 13313 | 0,00 | 0,00 | 0,00 | 0,00 | 0,00 |
| 16 | 13375 | 0,00 | 0,00 | 14,29 | 0,00 | 0,00 |
| 17 | 13392 | 0,00 | 0,00 | 0,00 | 0,00 | 0,00 |
| 18 | 13399 | 0,00 | 0,00 | 0,00 | 0,00 | 0,00 |
| 19 | 13400 | 0,00 | 0,00 | 0,00 | 0,00 | 0,00 |
| 20 | 13509 | 0,00 | 0,00 | 14,29 | 0,00 | 0,00 |
| 21 | 13606 | 0,00 | 0,00 | 14,29 | 0,00 | 0,00 |
| 22 | 13686 | 0,00 | 0,00 | 14,29 | 0,00 | 0,00 |
| 23 | 13715 | 0,00 | 0,00 | 14,29 | 0,00 | 0,00 |
| 24 | 13777 | 0,00 | 0,00 | 14,29 | 0,00 | 0,00 |
| 25 | 13833 | 0,00 | 0,00 | 14,29 | 0,00 | 0,00 |
| 26 | 13948 | 0,00 | 0,00 | 14,29 | 0,00 | 0,00 |
| 27 | 13960 | 0,00 | 0,00 | 14,29 | 0,00 | 0,00 |
| 28 | 13992 | 0,00 | 0,00 | 0,00 | 0,00 | 0,00 |
| 29 | 13999 | 0,00 | 0,00 | 14,29 | 0,00 | 0,00 |
| 30 | 14021 | 0,00 | 0,00 | 0,00 | 0,00 | 0,00 |
| 31 | 14063 | 0,00 | 0,00 | 0,00 | 0,00 | 0,00 |
| 32 | 14117 | 0,00 | 0,00 | 14,29 | 0,00 | 0,00 |
| 33 | 14133 | 0,00 | 0,00 | 0,00 | 0,00 | 0,00 |
| 34 | 14143 | 0,00 | 0,00 | 0,00 | 0,00 | 0,00 |
| 35 | 14269 | 0,00 | 0,00 | 14,29 | 0,00 | 0,00 |
| 36 | 14443 | 0,00 | 0,00 | 14,29 | 0,00 | 0,00 |
| 37 | 15097 | 0,00 | 0,00 | 0,00 | 0,00 | 0,00 |
| 38 | 15163 | 0,00 | 0,00 | 14,29 | 0,00 | 0,00 |
| 39 | 15172 | 0,00 | 0,00 | 0,00 | 0,00 | 0,00 |
| 40 | 15182 | 0,00 | 0,00 | 0,00 | 0,00 | 0,00 |
| 41 | 15543 | 0,00 | 0,00 | 14,29 | 0,00 | 0,00 |
| 42 | 15619 | 0,00 | 0,00 | 0,00 | 0,00 | 0,00 |
| 43 | 15700 | 0,00 | 0,00 | 0,00 | 0,00 | 0,00 |
| 44 | 15735 | 0,00 | 0,00 | 0,00 | 0,00 | 0,00 |
| 45 | | | | | | |
| 46 | | | | | | |
| 47 | Gemiddelden: | 0,00 | 0,00 | 7,94 | 0,00 | 0,00 |
| 48 | | | | | | |
| 49 | | | | | | |
| 50 | | | | | | |
| 51 | | | | | | |

Inhoudsopgave Template Checklists 12208 12448 12521 12959 12964 13135 13313 13375

Gereed

Op dit voorblad kunnen meldingen worden toegevoegd aan de checklist door op de knop “Toevoegen Melding” te klikken. Er wordt voor deze melding een nieuw tabblad aangemaakt op basis van de template checklist die als tweede tabblad staat weergegeven. Verder is het mogelijk om met de knoppen bovenin het blad tussen de verschillende werkbladen van de checklist te

navigeren. Ieder toegevoegd meldingsnummer is op te klikken. Wanneer men op het nummer klikt wordt automatisch de checklist horende bij dat meldingsnummer getoond. Statistieken met betrekking tot de voorgang van de verschillende fasen in het testproces binnen de testafdeling worden zowel per meldingsnummer als voor de gehele uitlevering getoond.
Voor historische en veiligheidsredenen is ervoor gekozen om meldingen niet met een knop te laten verwijderen.
Het werkblad voor de meldingen ziet er als volgt uit:



| | A | B | C | D | E |
|---|---|---|---|---|---|
| 1 | **Datum** | | | | |
| 2 | 30-5-2006 | | | | |
| 3 | **Meldingsnummer** | | | | <--Vorige Inhoudsopgave Volgende--> |
| 4 | 12208 | | | | |
| 5 | **Versie uitlevering** | | | | |
| 6 | 5.6.a | | | | |
| 7 | **Aangemaakt door** | | | | |
| 8 | Bram van den Abeele | | | | |
| 9 | | | | | |
| 10 | **Planning** | | | | |
| 11 | **Nr** | **Checkpoint** | **Datum voltooid** | **Persoon** | **Opmerkingen** |
| 12 | 1 | Opdrachtformulering | | | |
| 13 | 2 | Vaststellen van de testbasis | | | |
| 14 | 3 | Bepalen teststrategie | | | |
| 15 | 4 | Inrichten organisatie | | | |
| 16 | 5 | Inrichten testproducten | | | |
| 17 | 6 | Definiëren infrastructuur | | | |
| 18 | 7 | Definiëren tools | | | |
| 19 | 8 | Inrichten beheer | | | |
| 20 | 9 | Bepalen planning | | | |
| 21 | 10 | Fixeren testplan | | | |
| 22 | **Percentage voltooid:** | 0,00 | | | |
| 23 | | | | | |
| 24 | **Specificatie** | | | | |
| 25 | **Nr** | **Checkpoint** | **Datum voltooid** | **Persoon** | **Opmerkingen** |
| 26 | 1 | Opstellen testspecificaties | | | |
| 27 | 2 | Opstelling testscripts | | | |
| 28 | 3 | Specificeren intake testobject | | | |
| 29 | 4 | Specificeren intake testinfrastructuur | | | |
| 30 | 5 | Realiseren testinfrastructuur | | | |
| 31 | **Percentage voltooid:** | 0,00 | | | |
| 32 | | | | | |
| 33 | **Uitvoering 05** | | | | |
| 34 | **Nr** | **Checkpoint** | **Datum voltooid** | **Persoon** | **Opmerkingen** |
| 35 | 1 | ***Intake testobject*** | *30-5-2006* | *Bram van den Abeele* | |
| 36 | 2 | Intake testinfrastructuur | | | |
| 37 | 3 | Opzetten testdatabases /-bestanden | | | |
| 38 | 4 | Uitvoering (her)tests: First pass | | | |
| 39 | 5 | Test(s) succesvol op SQL Server omgeving | | | |
| 40 | 6 | Test(s) succesvol op Oracle omgeving | | | |
| 41 | 7 | Goedgekeurd in Verbandenoverleg | | | |
| 42 | **Percentage voltooid:** | 14,29 | | | |
| 43 | | | | | |
| 44 | **Uitvoering 08** | | | | |
| | **Nr** | **Checkpoint** | **Datum** | **Persoon** | **Opmerkingen** |

Inhoudsopgave | Template Checklists | 12208 | 12448 | 12521 | 12959 | 12964 | 13135 | 13313 | 13375 | 13392 | 13399

Gereed

Met behulp van de checkboxen kan aangevinkt worden welke activiteiten voltooid zijn. Er wordt op dat moment automatisch een datum en de naam van de ingelogde persoon ingevuld. Dit omdat het invullen van de checklist anders ontzettend veel werk zou worden als voor iedere melding alles handmatig ingevuld zou moeten worden. Er kan, indien gewenst, ook een opmerking ingevuld worden. Per fase wordt verder bijgehouden in hoeverre deze voltooid is.
De navigatieknoppen voor het bladeren door het checklist document zijn ook hier bovenin het scherm te vinden.