|  |
| --- |
|  |



Afstudeerverslag

Titel: TJIP Processmanager  
Student: Dhr. J. Niesen  
Studentnummer: 08033625  
School: De Haagse Hogeschool  
Opleiding: Informatica  
  
Afstudeerperiode: 06-02-2012 / 01-06-2012  
Examinatoren: Dhr. V. E. Broeren  
 Dhr. R. Ruijsenaars

Bedrijf: TJIP B.V.  
Bedrijfsbegeleider: Dhr. E. van Doorn

# Voorwoord

Dit verslag is het resultaat van 17 weken afstuderen. Het verslag is bedoeld om een beeld te geven van mijn werkzaamheden bij TJIP. Er wordt vooral in gegaan op keuzes die zijn gemaakt, en waarom ik deze heb gemaakt. Het is de bedoeling dat middels dit verslag en eventuele bijlage een beoordeling voor deze afstudeeropdracht kan worden geformuleerd.

Ik wil van dit moment gebruik maken om enkele mensen te bedanken. In de eerste plaats wil ik Eric van Doorn bedanken. Eric heeft mij een hoop geleerd, en veel tijd geïnvesteerd in deze opdracht, bedankt daarvoor!

In de tweede plaats wil ik Vincent Broeren en Remco Ruijsenaars bedanken. Zij waren mijn examinatoren en afstudeerbegeleiders bij de Haagse Hogeschool. Ik wil ze graag bedanken voor de adviezen betreffende het afstuderen. Ook wil ik ze alvast bedanken voor het beoordelen van deze afstudeer opdracht.

In de derde plaatst, wil ik graag een medestudent en goede vriend, Jesse van Assen, bedanken. Jesse heeft gedurende mijn afstudeerstage diverse malen feedback gegeven. Ook wil ik hem bedanken voor het lezen van dit verslag. Bedankt hiervoor!

Inhoudsopgave

[Voorwoord 1](#_Toc326225485)

[1 Inleiding 4](#_Toc326225486)

[2 TJIP BV 5](#_Toc326225487)

[2.1 Organisatie 5](#_Toc326225488)

[2.2 Werkzaamheden 7](#_Toc326225489)

[2.3 Klanten 7](#_Toc326225490)

[3 Opdracht 8](#_Toc326225491)

[3.1 Achtergrondinformatie 8](#_Toc326225492)

[3.2 Probleem 10](#_Toc326225493)

[3.3 Doelstelling 10](#_Toc326225494)

[3.4 Requirements Managementconsole 11](#_Toc326225495)

[4 Tot stand komen PVA 12](#_Toc326225496)

[4.1 Fasering 12](#_Toc326225497)

[4.2 Aanpak softwareontwikkeling 12](#_Toc326225498)

[4.3 Ontwikkelomgeving 14](#_Toc326225499)

[4.4 Te gebruiken technieken 14](#_Toc326225500)

[5 Activiteiten 15](#_Toc326225501)

[5.1 Testactiviteiten 15](#_Toc326225502)

[5.2 Sprint 1 19](#_Toc326225503)

[5.3 Sprint 2 33](#_Toc326225504)

[5.4 Sprint 3 37](#_Toc326225505)

[5.5 Sprint 4 43](#_Toc326225506)

[6 Evaluatie 44](#_Toc326225507)

[6.1 Productevaluatie 44](#_Toc326225508)

[6.2 Procesevaluatie 46](#_Toc326225509)

[6.3 Evaluatie van de verrichtte competenties 48](#_Toc326225510)

[7 Bijlage A – Plan van Aanpak 49](#_Toc326225511)

[7.1 Versiebeheer 49](#_Toc326225512)

[7.2 Opdracht 50](#_Toc326225513)

[7.3 Aanpak 52](#_Toc326225514)

[7.4 Ontwikkelstraat 54](#_Toc326225515)

[7.5 Kwaliteit 56](#_Toc326225516)

[8 Bijlage B – Testplan 57](#_Toc326225517)

[8.1 Versiebeheer 57](#_Toc326225518)

[8.2 Inleiding 57](#_Toc326225519)

[8.3 Documenten 57](#_Toc326225520)

[8.4 Testobjecten 57](#_Toc326225521)

[8.5 Testonderwerpen 57](#_Toc326225522)

[8.6 Testbenadering 58](#_Toc326225523)

[8.7 Done criteria 59](#_Toc326225524)

[8.8 Testproducten 59](#_Toc326225525)

[8.9 Eisen omgeving 60](#_Toc326225526)

[8.10 Testtools 61](#_Toc326225527)

[9 Bijlage C – Functioneel Ontwerp 62](#_Toc326225528)

[9.1 Inleiding 62](#_Toc326225529)

[9.2 Opzet 63](#_Toc326225530)

[9.3 Management-Console Algemeen 64](#_Toc326225531)

[9.4 Management-Console Dashboard 65](#_Toc326225532)

[9.5 Management-Console Configuratie 68](#_Toc326225533)

[9.6 Management-Console Operation 71](#_Toc326225534)

[9.7 Management-Console Tools 85](#_Toc326225535)

[10 Bijlage D – Testscripts 91](#_Toc326225536)

[10.1 Testscripts 91](#_Toc326225537)

[11 Bijlage E - Performancetest Prototype 101](#_Toc326225538)

[11.1 Bevindingen 102](#_Toc326225539)

[11.2 Verbeteringen 102](#_Toc326225540)

[12 Bijlage F – WCF Performancetest 103](#_Toc326225541)

[12.1 Bevindingen 103](#_Toc326225542)

[12.2 Verbeteringen 103](#_Toc326225543)

[13 Bijlage G – Testrapport 104](#_Toc326225544)

[13.1 Testonderwerpen 104](#_Toc326225545)

[13.2 Uitgevoerde testen 104](#_Toc326225546)

[13.3 Bevindingen 105](#_Toc326225547)

[13.4 Advies 105](#_Toc326225548)

# Inleiding

In het laatste jaar van de opleiding Informatica aan de Haagse Hogeschool moet een student afstuderen. Tijdens de afstudeerperiode toont de student aan dat hij/zij in staat is om zelfstandig te functioneren in het bedrijfsleven.  
Dit document doet verslag van mijn afstudeerperiode bij TJIP BV. Het doel van dit document is inzichtelijk te maken welke activiteiten ik heb uitgevoerd en waarom ik deze zo heb uitgevoerd. De informatie in dit verslag moet gebruikt kunnen worden om mijn functioneren gedurende de afstudeerperiode te kunnen beoordelen.

Dit document is opgebouwd uit drie onderdelen:

1. In het eerste onderdeel wordt achtergrondinformatie gegeven. Er valt hier te lezen wat TJIP voor een bedrijf is, welke functie ik hier heb vervuld en wat mijn opdracht was bij dit bedrijf. Ook valt hier te lezen hoe het plan van aanpak tot stand is gekomen.
2. In het tweede deel geeft een beschrijving van werkzaamheden bij TJIP. In wordt gegaan op de activiteiten die ik heb uitgevoerd, en waarom ik deze zo heb uitgevoerd.
3. Het laatste onderdeel is een evaluatie. Er vinden hier twee evaluaties plaats; een procesevaluatie en een productevaluatie.

# TJIP BV

Gedurende mijn afstudeerperiode ben ik in dienst geweest van TJIP BV. TJIP is een bedrijf wat zich bezighoud met het ontwikkelen van software voor onder andere de financiële wereld en de zorgsector. Het bedrijf is opgericht in oktober 1999, en telde bij de oprichting twee medewerkers. Het jaar daarop sloot TJIP af met negen medewerkers. TJIP heeft op dit moment zo’n 90 medewerkers in dienst, en is gevestigd in Delft en Helmond.

Door de marktkeuze van TJIP moeten applicaties betrouwbaar zijn en moeten er vrijwel altijd koppelingen worden gemaakt met externe systemen. Dit maakt de software in veel gevallen erg complex. Om het ontwikkeltraject te versnellen probeert TJIP zoveel mogelijk gebruik te maken van standaard (zelf ontwikkelde) componenten.

Tijdens het afstuderen bij TJIP ben ik te vinden op de ontwikkelafdeling. Ik zal werken aan een standaard component, wat de naam “Processmanager” draagt. Dit component zal na voltooiing gebruikt worden in een aantal andere ontwikkelprojecten die nu lopen.

## Organisatie

TJIP werkt hoofdzakelijk op projectbasis. Voor iedere applicatie die moet worden ontwikkeld, wordt een project opgezet. Binnen projecten zijn mensen met verschillende competenties aanwezig. Doorgaans zijn de volgende competenties in projecten bij TJIP te vinden:

* Projectmanager
* Software architect
* Software ontwikkelaar
* Test coördinator
* Software tester

Afhankelijk van de grote van het project, zullen sommige competenties meer of minder aanwezig zijn.



Organogram van TJIP, in deze organogram is de focus gelegd op de afdeling softwareontwikkeling.

## Werkzaamheden

Tjip richt zich hoofdzakelijk op de volgende branches: banken, de zorg en verzekeringen. Dit betekend dat TJIP een technisch profiel heeft. In de applicaties die ze bouwen wordt veel gerekend, en zijn er vaak veel externe koppelingen aanwezig. De applicaties die TJIP maakt zijn doorgaans maatwerkapplicaties

Voorheen was er weinig tot geen aandacht voor het visuele deel van een applicatie. Tegenwoordig willen klanten dat hun applicatie er ook mooi en fancy uit ziet. Om deze reden zijn er de afgelopen tijd ook visual designers aangenomen bij TJIP. Zij ontwikkelen de grafische interface voor applicaties.

### Visie

TJIP heeft de volgende visie gedefinieerd:

***Business Runs On Code****Kwalitatief hoogwaardige software is essentieel voor bedrijven om succesvol en onderscheidend te zijn in de uitvoering van hun business.*

*Goede software is meer dan alleen code. Kwaliteit van software uit zich ook in flexibiliteit om te anticiperen op marktontwikkelingen met een leverancier die tevens business partner is. Een business partner die begrijpt hoe software bijdraagt aan het succes van zijn klant en hem proactief ondersteunt om de doelstellingen van zijn klant te bereiken.*

*Vanuit deze visie biedt TJIP al meer dan 10 jaar robuuste en vernieuwende oplossingen voor haar klanten.*

### Missie

TJIP heeft de volgende missie gedefinieerd:

*Met een passie voor hoogwaardige technologie realiseert TJIP vernieuwing en efficiëntie voor haar klanten. Hierbij vormen kennis, integriteit, partnering en een visie op toekomstige ontwikkelingen het fundament voor onze oplossingen.*

*TJIP wil een goede en betrouwbare leverancier voor haar klanten en een goede werkgever voor haar medewerkers zijn.*

## Klanten

Onder de klanten van TJIP bevinden zich een aantal grote namen, waarvan de grootste ABN Amro en Achmea zijn.

TJIP probeert klanten te betrekken bij de software ontwikkeling. Dit wordt onder andere gedaan door klanten enkele dagen in de week te laten werken in het kantoor van TJIP. Op deze manier blijven de communicatielijnen met de klanten kort.

# Opdracht

## Achtergrondinformatie

### Wat is Processmanager

Vrijwel alle projecten binnen TJIP betreffen web-applicaties, waarbij er processing plaats vind via een korte transactie op basis van een request-response mechanisme. Een webserver is daarvoor erg geschikt. Echter wil het regelmatig voorkomen dat er in projecten bepaalde zaken spelen waar een webserver niet voor geschikt is:

* Langlopende processen
* Resource intensieve processen
* Processen die op een bepaald tijdstip moeten draaien
* Berichtverwerking
* Externe communicatie

Om dit soort processen beter te ondersteunen is TJIP begonnen met het bouwen van een prototype dat “Processmanager” heet. Processmanager biedt functionaliteit om bovenstaand soort processen te ondersteunen.

### Hoe werkt Processmanager

Processmanager werkt op basis van jobs. Een job is het onderdeel binnen Processmanager wat het bericht daadwerkelijk verwerkt. Het verwerken van een bericht zou bijvoorbeeld kunnen zijn: het opslaan van gegevens (afkomstig uit het bericht) in een database.  
Een job maakt gebruik van connectoren. Connectoren zijn onderdelen binnen Processmanager die een bericht downloaden van de bron (POP3 e-mailbox, map op de hardeschrijf, IMAP mailbox etc.), en beschikbaar stellen aan de Job. Ook zijn er uitgaande connectoren. Het is ook mogelijk om berichten te versturen vanuit een job. Hiervoor worden uitgaande connectoren gebruikt. Enkele voorbeelden van uitgaande connectoren zijn: het versturen van een fax, het verzenden van een e-mail, het plaatsen van een bestand in een map. Het is mogelijk om jobs te hebben die enkel uitgaande connectoren bevatten.  
Jobs kunnen worden getriggerd. Een trigger zorgt ervoor dat een job uitgevoerd kan worden op een bepaald tijdstip. Dit kan een vast tijdstip zijn (1 juni 2012 om 13:00 uur) maar ook een herhalend tijdstip (iedere maandag om 2 uur).



De samenhang van een job, connectoren en triggers. Als voorbeeld is hier een job met drie inkomende connectoren, twee triggers en één uitgaande connector getekend.

De jobs bevinden zich in een losse class-library die wordt ingelezen door Processmanager. Hierdoor kan per project dus een eigen set aan jobs worden ontwikkeld. De configuratie is op dit moment uitgevoerd in XML. Tijdens het starten van Processmanager worden de XML-bestanden met de configuratie uitgelezen.

Intern in Processmanager wordt de berichtverwerking afgehandeld volgens een store-forward principe. Dit houd in dat bij de ontvangst van een bericht, of het versturen van een bericht, het bericht wordt opgeslagen in een database. Op een later tijdstip zal het bericht worden doorgegeven aan een job. Het voordeel hieraan is dat wanneer een bericht niet verstuurd kan worden, dit op een later tijdstip opnieuw gedaan kan worden.

Processmanager draait als een Windows service.

### Hoe wordt Processmanager ingezet

Processmanager wordt ingezet als onderdeel binnen een project. Om e-mails e.d. te versturen vanuit de applicatie kan er een clientlibrary worden gereferenced. Deze maakt via Windows Communcication Foundation (WCF) verbinding met de Processmanager service. Middels de client-library is het mogelijk om berichten door process-manager af te laten leveren.

Windows Communication Foundation (WCF)  
WCF is een onderdeel van het Microsoft .net framework wat het mogelijk maakt om op eenvoudige wijze client-server applicaties te maken. Het is een infrastructuur voor communicatie op basis van onder andere webservices. Een webservice kan worden omschreven als een interface van een applicatieonderdeel dat toegankelijk is via standaard webprotocollen, en waarbij wordt gecommuniceerd via XML.

Verder worden eventuele inkomende berichten verwerkt middels de eerder besproken job/connector constructie.



Schematische weergave hoe Processmanager werkt.

## Probleem

Er is op dit moment enkel een prototype beschikbaar van TJIP Processmanager. Dit prototype zal moeten worden getest, en eventueel moeten worden aangepast.   
Het prototype is op dit moment ook nog niet te beheren. Om het beheer uit te kunnen voeren zal er een managementconsole moeten worden ontwikkeld.

## Doelstelling

Om bovenstaand probleem te verhelpen zal in eerste instantie het prototype worden getest. Het prototype vormt de basis van Processmanager, en moet dus goed werken. De testcriteria zullen worden opgesteld in een testplan. In de tweede plaats zal er een managementconsole worden ontwikkeld waarmee Processmanager is te beheren.

## Requirements Managementconsole

### Functional Requirements

|  |  |  |
| --- | --- | --- |
| 1 | Dashboard | Er moet een dashboard ontwikkeld worden waarmee direct de volgende zaken inzichtelijk zijn:   * Errors & Warnings * Currently executing jobs * Probleemberichten |
| 2 | Configuratie (read only) | De volgende onderdelen van de configuratie moeten inzichtelijk worden:   * Database * SMTP * SMS * Fax * Jobs * Calendars * Incoming connectoren * Outgoing connectoren * Servicehost * Assemblies |
| 3 | Bekijken berichteigenschappen | Bekijken van email, fax, sms en algemeen bericht, zowel binnenkomend als uitgaand. |
| 4 | Bekijken berichtinhoud (downloaden bericht) | Het downloaden van het bericht, en het bericht vervolgens openen met een daarvoor geschikte applicatie. |
| 5 | Bekijken tracelog | Tijdens jobs kan er worden gelogd. Deze logs moeten inzichtelijk worden in de managementconsole. |
| 6 | Verzenden berichten | Mogelijkheid tot het verzenden van berichten   * Verzenden van e-mail * Verzenden van fax * Verzenden van SMS |
| 7 | Starten trigger | Mogelijkheid tot het handmatig starten van een trigger |
| 8 | Starten job | Mogelijkheid tot het handmatig starten van een job |
| 9 | Visualisatie van jobs | Visualisatie van de job en de daaraan hangende connectoren en triggers beschikbaar maken in de applicatie |
| 10 | Opvragen status berichtverwerking | Er moet een mogelijkheid komen om binnen een project de status van de verwerking van een bericht op te vragen. |

### Non functional requirements

|  |  |  |
| --- | --- | --- |
| 1 | Uitbreidbaarheid | De managementconsole moet uitbreidbaar zijn. Zodat er in de toekomst specifieke functionaliteit kan worden toegevoegd aan de standaard console. |

# Tot stand komen PVA

De eerste dagen op mijn afstudeeradres heb ik gebruikt voor het opstellen van een plan van aanpak. Dit plan was bedoeld om “de te volgen weg” vast te leggen en de opdracht af te bakenen.

## Fasering

Het leek mij verstandig om dit project in te delen in twee fases. Dit omdat ik eerst de kwaliteit van het prototype wilde testen, en waar nodig verbeteren. Dit vond ik belangrijk omdat het prototype de basis zal zijn waar verder op doorgebouwd zal worden. Hierdoor ontstaan de volgende fases met activiteiten:

|  |
| --- |
| **Fase 1 (Bepalen kwaliteit prototype) Activiteiten** |
| 1. Vaststellen kwaliteit van het prototype middels ketentest en WCF-test. |
| 1. Verbeteren van fouten in het prototype. |
| **Fase 2 (Uitbouwen prototype) Activiteiten** |
| 1. Verzamelen van de requirements. |
| 1. Bouwen van een functioneel ontwerp voor de nieuwe functionaliteit. |
| 1. Technisch ontwerp maken voor nieuw te bouwen functionaliteit. |
| 1. Bouwen van de software op basis van het functioneel ontwerp en technisch ontwerp. |
| 1. Testen van de gebouwde software aan hand van het functioneel ontwerp. |

## Aanpak softwareontwikkeling

### Fase 1

Voor de eerste fase is gekozen om de watervalmethode te hanteren. Dit omdat er enkel wordt getest en bugs opgelost en er geen nieuwe functionaliteit wordt toegevoegd.

### Fase 2

Ik heb de keuze gemaakt om dit project iteratief aan te pakken. De keuze voor een iteratieve aanpak heeft de volgende redenen:

* Het minimale wordt gebouwd om feedback te krijgen van de opdrachtgever.
* Het is onmogelijk om in één keer alle requirements in te zamelen.

Fase twee zal worden uitgevoerd middels scrum. Iedere iteratie bevat de volgende onderdelen:

* Opstellen functioneel ontwerp
* Opstellen technisch ontwerp (technische documentatie als UML diagrammen)
* Ontwikkelen
* Testen

Een functioneel ontwerp is nodig om “dure fouten” te voorkomen en helder te hebben wat er precies gebouwd moet worden. Onder “dure fouten” versta ik fouten welke pas op een laat moment aan bod komen. Deze fouten zorgen er meestal voor dat een stuk functioneel ontwerp moet worden herschreven, code moet worden herzien en er opnieuw getest moet worden. Dit kost veel tijd, vandaar de term “dure fouten.”  
Na het verkrijgen van requirements worden deze verwerkt in het functioneel ontwerp. Het functioneel ontwerp wordt vervolgens geverifieerd door de opdrachtgever. Door deze verificatie is het functioneel ontwerp ook gelijk de testbasis, bij het testen kan worden gecontroleerd of het juiste gebouwd is. Na het opstellen van het functioneel ontwerp zal er een technisch ontwerp worden gemaakt. Het technisch ontwerp is bedoeld om refactoring door denkfouten tijdens het ontwikkelen te voorkomen. Technische ontwerpen (waar nuttig) zullen worden gemaakt aan hand van UML, dit omdat UML een wereldwijde standaard is voor softwaremodeling.



Processen welke uitgevoerd worden gedurende een iteratie

### Backlog

Normaliter wordt een backlog gevuld met “user stories.” Mijn persoonlijke ervaring is dat een use-case een beter beeld geeft van “hoe de applicatie zal werken.” Dit komt omdat use-cases meer gedetailleerd zijn dan user-stories. Door deze detaillering gaat de functionaliteit een beter beeld bij de opdrachtgever/klant, en kunnen zij hier goed feedback op geven. Omdat een use-case meer detaillering biedt dan een user-story, en functionaliteit hierdoor een beter beeld krijgt bij de opdrachtgever, heb ik besloten om use-cases te gebruiken in plaats van user-stories. De project-aanpak zal verder wel op de reguliere scrum-aanpak zijn.

De backlog zal worden gepland op basis van prioriteit. Wanneer de prioriteit van een use-case hoog is, zal deze eerder worden verwerkt dan een use-case met een lage prioriteit. De prioriteit van use-cases is bepaald in overleg met de opdrachtgever. De opdrachtgever is ook de product owner.

## Ontwikkelomgeving

Ik heb besloten om gebruik te maken van een OTAP ontwikkeling. Niet alle onderdelen van OTAP kunnen gebruikt worden in dit project, maar het gebruik van OTAP heeft wel voordelen. Zo wordt er niet getest op een ontwikkelomgeving. Dit heeft als voordeel dat een testomgeving kan worden ingericht zoals de productieomgeving, en de applicatie dus draait zoals die ook bij klanten zou draaien.

Voor een aantal tests is Visual Studio vereist. Processmanager geeft gebruikersvriendelijke meldingen wanneer er een exceptie wordt gegooid. Middels Visual Studio is de inhoud van die excepties te zien. Enkele testen waarbij Visual Studio is vereist duren erg lang. Hiervoor heb ik besloten om de ontwikkel-test omgeving (OT –omgeving) toe te voegen aan de OTAP-ontwikkelstraat. Deze omgeving zal gebruikt worden om langdurige tests uit te voeren waarvoor eigenlijk een ontwikkelomgeving vereist is.   
Door deze omgeving toe te voegen blijft de ontwikkelomgeving beschikbaar om te ontwikkelen. Een mooi voorbeeld om het nut te demonstreren van de OT-omgeving is het uitvoeren van een performancetest. Wanneer er excepties optreden wil ik deze graag inzien, en de test duurt enkele dagen. Terwijl deze test op de OT-omgeving draait, kan ik op de ontwikkelomgeving gewoon door ontwikkelen.



*Overzicht omgevingen: OTAP met een uitbreiding van de Ontwikkel-Test omgeving*

## Te gebruiken technieken

De te gebruiken technieken is een lijst geworden welke gedurende het project aangepast kon worden. De enige technieken welke vaststonden; waren de technieken welke verwerkt zaten in het prototype en WPF.

Windows Presentation Foundation (WPF)  
WPF is een onderdeel in het Microsoft .Net framework waarmee het mogelijk is om grafische interfaces te bouwen. Deze interfaces worden gebouwd in XAML-files. XAML-Files zijn xml-bestanden waarin de grafische interface is beschreven. WPF biedt ondersteuning voor het MVVM-pattern (model-view-viewmodel).

# Activiteiten

## Testactiviteiten

De activiteiten beschreven in dit deel van het verslag gaan over de eerste fase van het project. In deze fase wordt er getest om de kwaliteit van het prototype vast te stellen.

### **Testplan**

Tegelijk met het opstellen van het plan van aanpak ben ik begonnen aan het opstellen van een testplan. Omdat de eerste fase in het teken staat van het bepalen van de kwaliteit van het prototype, is het belangrijk dat hier een testplan voor beschikbaar is. In het testplan staat geschreven aan welke eisen software moet voldoen, en hoe er getest zal worden of de software hier ook aan voldoet.

Het testplan dat ik heb opgesteld is bedoelt als voorstel. De opdrachtgever zou op dit voorstel feedback kunnen leveren, zodat er op den duur een definitieve versie ontstaat.

Omdat dit project binnen een bepaalde tijd af moet zijn, heb ik besloten om belang toe te kennen aan kwaliteitsattributen. Hiermee kunnen kwaliteitsattributen worden geprioriteerd.

|  |  |  |
| --- | --- | --- |
| **Kwaliteitsattribuut** | **Relatief Belang** | **Toelichting** |
| Betrouwbaarheid | +++ | Bedrijfszekerheid, Foutbestendigheid |
| Bruikbaarheid |  |  |
| Efficiëntie | +++ | Middelenbeslag, Tijdbeslag |
| Effectiviteit | + | Juistheid |
| Onderhoudbaarheid | + | Analyseerbaarheid, Wijzigbaarheid, Stabiliteit |
| Portabiliteit | + | Installeerbaarheid |

In het voorstel is bovenstaande tabel gebruikt. Zoals te zien heb ik enkele onderdelen meer “belang” gegeven dan andere.

* Betrouwbaarheid staat centraal in de applicatie. Bedrijfsprocessen kunnen verstoord worden wanneer TJIP Processmanager niet betrouwbaar is. Hierdoor heb ik dit kwaliteitsattribuut een hoger belang gegeven.
* Bruikbaarheid is iets wat enkel door gebruikers kan worden getest. Hierdoor heeft dit geen belang.
* TJIP Processmanager kan in het bedrijfsleven zwaar belast worden, en wordt daarom geacht efficiënt om te gaan met middelen en tijd. Omdat een langzaam functionerende versie van TJIP Processmanager de snelheid waarmee bedrijfsprocessen kunnen worden doorlopen kan aantasten, heb ik besloten hier meer belang aan toe te kennen.
* Het is belangrijk dat TJIP Processmanager functioneert zoals omschreven in het functioneel ontwerp. Omdat er echter ook externe factoren zijn welke invloed hebben op de juistheid van functioneren van TJIP Processmanager, heeft dit niet een hoog belang.
* De onderhoudbaarheid van TJIP Processmanager is belangrijk. Echter levert de onderhoudbaarheid van TJIP Processmanager geen gevaar voor de bedrijfsprocessen, hierdoor heeft dit geen hoog belang.
* Het is belangrijk dat TJIP Processmanager geïnstalleerd kan worden. De installeerbaarheid van TJIP Processmanager vormt echter geen gevaar voor de bedrijfsprocessen. Hierdoor heeft dit geen hoog belang.

Na het opsturen van het voorstel, is deze direct goedgekeurd.

### Loadtest

Na het opstellen van het testplan ben ik begonnen met het testen van het prototype. Ik heb er voor gekozen om het prototype te testen middels een loadtest. Omdat vrijwel alle beschikbare connectoren aan elkaar worden geknoopt, heeft deze test een erg hoge dekkingsgraad. Doordat alle connectoren aan elkaar waren geknoopt werd vrijwel alle functionaliteit binnen het prototype gebruikt.



Uitgevoerde loadtest

Als loadtest heb ik bovenstaande test opgesteld. Deze test heeft de volgende voordelen:

1. Er kan een uitspraak worden gedaan over de kwaliteit van TJIP Processmanager.
2. Eventuele bugs boven water halen.
3. Bekend raken met de code van TJIP Processmanager.
4. De test later hergebruiken wanneer TJIP Processmanager is aangepast.

Zoals eerder in dit document beschreven is er een extra testomgeving beschikbaar waarbij applicaties kunnen draaien vanuit Visual Studio. Bij deze test was het handig om excepties te kunnen zien. Ook had dit het bijkomend voordeel dat ik direct kon zien waar de exceptie gegooid werd. Omdat ik niet bekend was met de code, scheelde mij dit een hoop zoekwerk.

Bij het uitvoeren van de loadtest kwamen de volgende bevindingen aan het licht:

* GoSecure incoming connector verwerkt geen bestanden
* SQL gooit Server deadlock excepties op diverse plaatsen in de applicatie bij een grote hoeveelheid berichtenverkeer
* Conversieprobleem bij Mime-type berichten (veel whitespace werd toegevoegd)
* Probleem bij het plaatsen van grote hoeveelheden bestanden in de “test-in”-map. Wanneer er grote hoeveelheden bestanden worden geplaatst, ontstaan er geheugen errors.

Aan hand van bovenstaande bevindingen heb ik een aantal wijzigingen uitgevoerd. Deze wijzigingen loste enkel bugs op en gaven geen nieuwe functionaliteit.

* GoSecure connector aangepast. Hier werd geprobeerd het attachment-bericht te verwerken, in plaatst van het XML-bestand. Ik heb ervoor gezorgd dat nu het XML-bestand en het attachment-bestand verwerkt worden.
* Het lock-statement toegevoegd op diverse plaatsen in de code, en waar nodig het transaction isolation level aangepast, zodat er geen deadlocks meer optreden.
* Het verwerken van e-mailberichten aangepast zodat er geen whitespace meer wordt toegevoegd aan het bericht.
* Timer toegevoegd welke mappen uitleest met een bepaald interval. Wanneer er een bestand wordt geplaatst in een map, en dit zelfde bestand direct door een andere applicatie uitgelezen moet worden, ontstaat er een probleem. Het bestand wordt nog weggeschreven en is daardoor nog in gebruik. Hiervoor is een timer toegevoegd welke met een interval een map uitleest. Ook wordt er gebruik gemaakt van een retry-mechanisme.
* ConcurrentQueue toegevoegd voor het uitlezen van bestanden in mappen. Er zijn situaties waar er bestanden kunnen worden toegevoegd aan de wachtrij, en aan de andere kant tegelijkertijd bestanden uit de wachtrij kunnen worden gehaald. De huidige wachtrij had hier geen ondersteuning voor.

### WCF Performance-test

Vanuit TJIP was er de wens om de performance van de WCF-service welke in TJIP Processmanager zit te testen. Omdat de WCF service nog niet getest was, heb ik besloten om hier aandacht aan te besteden. Het was belangrijk dat met een realistisch aantal clients de WCF service kon worden benaderd. Daar het aantal clients per project kan verschillen, heb ik besloten om dit op te nemen als variabele.

Omdat mij eerder kenbaar is gemaakt dat de ManagementConsole welke later gebouwd gaat worden, gemaakt zal moeten worden in WPF, heb ik besloten om een testapplicatie te bouwen die daar gebruik van maakt.

De client die ik gebouwd heb, maakt gebruik van de Processmanager Clientlibrary die onderdeel is van het prototype. Om de dekkingsgraad van de code hoog te houden, heb ik besloten om deze client te gebruiken.   
Deze keuze introduceerde direct een aantal technische problemen. Het is mogelijk om één connectie per instantie van de clientlibrary te maken. Dit komt omdat de connectie geinstantieerd wordt als singleton. Dit zou effectief betekenen dat er normaliter maar één connectie kan zijn per test-applicatie. Na wat onderzoek en gesprekken met collega’s op de werkvloer blijkt dit niet helemaal waar te zijn. Er kan van een singleton één instantie bestaan per applicatie-domein, en een applicatie kan gebruik maken van meerdere applicatie-domeinen. Ik heb mij dus verdiept in applicatie-domeinen en heb uiteindelijk een kleine applicatie gebouwd, die onderstaande lagenstructuur bevat.

  
Lagenmodel zoals dit in TJIP Processmanager Client Simulation is verwerkt.

Tijdens het uitvoeren van de request, werd de eerder beschreven loadtest uitgevoerd. Dit om een situatie te creëren waarbij TJIP Processmanager regulier werk verricht, en er tegelijk requests worden gestuurd naar de WCF Service. De test werd uitgevoerd met 270 clients, dit is een scenario van een project waar TJIP Processmanager in gebruikt zal gaan worden.

Bij het uitvoeren van deze test viel op dat na een bepaalde tijd clients geen request meer uit konden voeren. Dit had er mee te maken dat de Processmanager te druk bezig was met het uitvoeren van database transacties. Op basis van deze bevinding zijn de volgende aanpassingen uitgevoerd om transacties sneller uit te voeren:

* LINQ query’s zijn vervangen voor pre-compiled LINQ-queries. Deze queries zorgen ervoor dat het uitvoeren van een query minder resources vraagt.
* Een aantal indexen en statistics toegevoegd aan de database, om queries sneller uit te laten voeren.

Met deze verbetering is de eerste fase van het project gesloten.

## Sprint 1

Voordat ik ben begonnen aan deze sprint heb ik aan een aantal gesprekken gehad met de opdrachtgever om requirements in te zamelen. In de opdrachtomschrijving van mijn afstudeeropdracht staat dat er een webbased-managementconsole gebouwd gaat worden. Vanuit het architectenteam van TJIP is besloten dat er een desktop-client moet komen in plaats van een webbased client. Dit omdat dit meer aansluit bij de wensen van de klanten.   
Omdat deze wijziging beter aansluit bij de eindgebruikers van het product en het geen ingrijpende veranderingen aan de opdrachtomschrijving te weeg heeft gebracht, ben ik met de wijziging akkoord gegaan.

Het doel van de eerste sprint was het opzetten van een framework en het vormgeven van de hoofdschermen. Ook moesten de eerste lees-operaties uitgevoerd kunnen worden. Ik heb hiertoe een stuk functioneel ontwerp geschreven aan hand van requirements welke ik in eerdere gesprekken heb ontvangen.  
Omdat deze sprint enkel in het teken staat van het framework, het vormgeven van de hoofdschermen, en lees-operaties heb ik besloten om enkele schermen en use-cases te ontwerpen en deze op te nemen in het functioneel ontwerp. De achterliggende gedachte hierbij is om een opzet te geven van wat er mogelijk is, en een discussie uit te lokken met betrekking tot de navigatie in de applicatie.  
Ieder scherm dat ik heb opgenomen in het functioneel ontwerp bestond uit twee afbeeldingen. Een afbeelding waarop zichtbaar is welke regio’s er binnen het scherm zijn, en een voorbeeld van hoe het scherm er uit zou zien wanneer het is ingevuld.

Voorbeeldscherm uit functioneel ontwerp, de instellingenpagina met de outlookbar-navigatie.

Aan hand van de discussie is gebleken dat de navigatie in orde was. Ook werd er uit de discussie duidelijk dat er nagedacht moest worden over de plaats waarop onderdelen weergegeven moesten worden. Hiermee bedoel ik op: welke items moeten er op het dashboard worden weergegeven, onder welke categorieën vallen bepaalde configuratie-items etc. Afgesproken is dat hier door de opdrachtgever over wordt nagedacht terwijl ik bezig ben met het ontwikkelen van het framework.

### WPF & WCF

Van uit het architectenteam binnen TJIP was besloten om deze managementconsole te maken in WPF. Deze WPF applicatie moet verbinding maken met de WCF-service van Processmanager. De WCF service bevat nog niet alle gewenste functionaliteit, en moest voor een aantal dingen worden uitgebreid.   
Er moest ook een mogelijkheid zijn om TJIP Processmanager aan te sturen zonder het gebruik van de management-console, bijvoorbeeld wanneer er de keuze wordt gemaakt om TJIP Processmanager te integreren in andere software projecten. Nu zal het in de toekomst ongetwijfeld voorkomen dat er een wijziging zal plaatsvinden aan de WCF-service in TJIP Processmanager. Afhankelijk van de wijziging zou dit tot gevolg kunnen hebben dat de implementatie van de WCF-service aangepast moet worden. Om dit probleem te verhelpen is besloten om gebruik te maken van een losse class-library die de implementatie van de WCF-service bevat. Wanneer er nu iets gewijzigd moet worden aan de WCF-service, zal dit enkel tot gevolg hebben dat de class-library aangepast moet worden. Deze class-library (ook wel client-library genoemd) was al voor een klein deel gebouwd in het prototype. De WCF service zal in meerdere applicaties gebruikt worden, vandaar dat er de keuze is gemaakt om een client library te gebruiken.





Voorbeeld van een call welke wordt uitgevoerd via de WCF service.

### Prism

Omdat de applicatie opgedeeld is in regio’s, en er wensen zijn om in de toekomst modules toe te voegen aan de applicatie, heb ik geadviseerd om gebruik te maken van Prism. Prism is een framework waarmee het mogelijk is om applicaties op te delen in modules. Ik heb dit advies uitgebracht om de volgende redenen:

* Het toekomstig gebruik van modules
* Prism heeft een aantal standaardcomponenten in zich zitten, welke het werken met WPF een stuk eenvoudiger maakt (denk hierbij aan een notification-object, een object wat automatisch terugkoppeling kan geven aan een view, wanneer het object gewijzigd is).
* Prism biedt navigatie middels regions
* Er wordt in Prism gebruik gemaakt van een Event-Aggregator. Wanneer er meerdere views zichtbaar zijn op een scherm, kunnen deze met elkaar communiceren.

Op school heb ik al eerder een applicatie gebouwd met Prism. Het initialiseren en werkend krijgen van Prism was dus niet heel erg lastig. Ik heb gekozen om Prism op te zetten met MEF (Managed Extensibility Framework) omdat MEF aan directory-discovery kan doen. Dit houdt in dat tijdens het opstarten van de applicatie een map wordt uitgelezen, en modules welke in die map staan worden geactiveerd. Hiermee voorkom ik dus dat de applicatie opnieuw gecompileerd moet worden wanneer er een nieuwe module verschijnt.

Prism  
Prism biedt een leidraad die helpt bij het ontwerpen en bouwen van flexibele, eenvoudig te onderhouden applicaties die gebruik maken van WPF of Silverlight. Met behulp van design-patterns en “architectural principles” helpt Prism bij het bouwen van “loosly coupled” applicatie, waarvan de onderdelen onafhankelijk kunnen worden ontwikkeld, en naadloos integreren in de overkoepelende applicatie. Dergelijke applicaties worden ook wel “composite applications” genoemd.

Middels prism is het mogelijk om schermen in applicaties op te delen in regio’s. In deze regio’s kan vervolgens content worden geplaatst. Prism biedt functionaliteit om content te koppelen aan een dergelijke regio. Ook biedt Prism uitgebreide functionaliteit om door applicaties te navigeren (andere content tonen in bepaalde scherm regio’s).

Prism voegt een plugin-in structuur toe aan de applicatie. Middels deze structuur is het mogelijk om modules (schermen en functionaliteit) toe te voegen aan de applicatie.

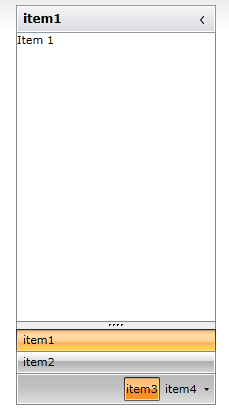
Ook heeft Prism een aantal standaard functionaliteiten in zich zitten die het gebruik van WPF eenvoudiger maken. Hierbij valt te denken aan een implementatie van DelegateCommand (wordt gebruikt om een knop aan een method in het viewmodel te koppelen), maar ook een baseclass voor viewmodels. Deze baseclass bevat functionaliteit om aan de view te melden dat er gegevens in het viewmodel gewijzigd zijn.

Managed Extensebility Framework (MEF)  
MEF is een onderdeel in het Microsoft .Net framework wat het mogelijk maakt om een plugin-architectuur te implementeren. Het implenteert deze plugin-architectuur door gebruik te maken van inversion of control (IoC) en dependency injection (DI). Door IoC instantiëren objecten niet meer hun eigen objecten waar ze van afhankelijk zijn, maar verkrijgen deze vanuit een externe bron, bijvoorbeeld een XML-file. Middels DI worden de afhankelijke objecten geïnjecteerd in het object. Dit gebeurt doorgaans door een los framework, die de afhankelijke objecten meegeeft via de constructor, of meegeeft middels properties. MEF is een dergelijk framework wat “dependency-objects” injecteert.

### Telerik

Er was een wens om de navigatie voor het onderdeel “settings” gebruik te laten maken van een “outlook-bar.” Dit is een menu wat ook in outlook zit, en ook wel bekend staat als accordeon. Het ontwikkelen van een dergelijk menu zou enorm veel tijd gaan kosten, en is eigenlijk een ontwikkelproject op zichzelf. Om die reden heb ik aangedrongen om gebruik te gaan maken van Telerik WPF controls. Die set aan controls bevat al een Outlook-bar, en biedt tevens een groot aantal andere potentieel interessante controls.

In de outlook-bar moest een tree-view komen met de beschikbare instellingen. De standaard tree-view van WPF staat het niet toe om aan een tree-item een command te binden. Een command is de koppeling tussen de knop in het scherm en de actie die uitgevoerd wordt in het view-model. Wanneer ik de WPF-implementatie van de tree-view zou gebruiken, moest ik het command binden aan de gehele tree-view. Wanneer er op iets geklikt wordt, zou ik de hele tree-view moeten uitlezen en kijken welk item er geselecteerd is. In de implementatie van de tree-view van Telerik is het wel mogelijk om een command te binden aan een tree-item. Tevens sloot deze tree-view visueel ook aan bij de outlook-bar. Ik heb er dus voor gekozen om de Telerik tree-view te implementeren.

  
Voorbeeld van de outlook-bar in Telerik.

Een andere belangrijk punt voor de keuze om Telerik te gebruiken had te maken met de datagrid. Men wilde kunnen sorteren en groeperen. Telerik biedt standaard een zeer uitgebreide datagrid, waarin het mogelijk is om te sorteren, te groeperen en te filteren. Deze datagrid was een belangrijk argument, omdat het veel tijd zou schelen tijdens het bouwen van de applicatie.

### Projectindeling

Ik vind het belangrijk dat het project goed ingedeeld is, en heb hier dus ook specifiek aandacht aan besteed. Wanneer een project groter wordt, zou een slechte indeling kunnen leiden tot een onoverzichtelijk geheel.

* Data
  + Icons
* Helpers
* Style
* Model
* View
  + Dashboard
    - Items
  + Operational
    - Items
  + Settings
    - Items
    - Navigation
* Viewmodel
  + Dashboard
  + Operational
    - Items
  + Settings
    - Items
    - Navigation

Door het gebruik van WPF is het goed mogelijk om onderscheid te maken tussen views en andere code. In een view en de code-behind zit namelijk geen enkel stuk code wat functionaliteit implementeert. De enige code welke ik toesta in de code-behind van een view is het koppelen van een viewmodel. Dit is puur omdat het noodzakelijk is, en dit niet op een andere plaats kan.   
Ik wilde geen implementatie in de views, zodat er een duidelijk onderscheid is tussen functionaliteit en views. Dit heeft als voordeel dat als een view ooit gewijzigd moet worden, dit gedaan kan worden door een visual-designer, en hier vaak geen programmeur bij aanwezig hoeft te zijn.

### Opzetten project met Prism

Prism applicaties kunnen gebruik maken van modules. Deze modules kunnen los worden ontwikkeld, en later integreren in de applicatie, zonder dat deze opnieuw gecompileerd hoeven te worden.  
In de eerste sprint heb ik geen aandacht aan dit principe besteed. Wat ik in deze sprint heb ontwikkeld behoord tot de basisfunctionaliteit. Het is naar mijn mening niet verstandig om de basisfunctionaliteit in een losse module te plaatsen. Dit omdat later te ontwikkelen modules hierdoor zowel afhankelijk zijn van de overkoepelende applicatie én de module welke de basisfunctionaliteit implementeert. Door de basisfunctionaliteit te implementeren in de overkoepelende applicatie, is dit een afhankelijkheid minder, en is het zeker dat de basisfunctionaliteit altijd beschikbaar is.

Prism kan gebruik maken van een logger. Door een logger te implementeren in Prism, kan Prism gebruik maken van de logger, en gelijk de rest van de applicatie. Omdat het management console een applicatie is die communiceert met een extern systeem is het belangrijk om een log beschikbaar te hebben. Wanneer er iets fout gaat tijdens de communicatie, is er in de log te lezen wat er fout is gegaan. Binnen TJIP wordt vrijwel overal gebruik gemaakt van “log4net.” Omdat men bekend is met deze logger, en hier uit de praktijk is gebleken dat de logger goed integreert, heb ik besloten om deze te gebruiken.

Ik heb uiteraard nog wel naar een andere logger gekeken voordat ik deze beslissing definitief nam. De voordelen van log4net zijn dat deze eenvoudig integreert, en hij sneller is dan zijn concurrent Enterprise Library Logging. Enterprise Library is een bibliotheek aan componenten die helpen bij het bouwen van een “Enterprise application.” Omdat ik de overige componenten niet nodig heb, en log4net qua performance beter uit de verf komt, heb ik mijn keuze om log4net te gebruiken definitief gemaakt.

Zoals eerder was te lezen, werkt Prism met “regio’s” op het scherm. Het is mogelijk om aan een usercontrol (bijvoorbeeld een tabcontrol) een region-name te koppelen. Het is vervolgens weer mogelijk om aan die region een “view” te koppelen. Ik heb deze koppeling laten plaatsvinden in de bootstrapper. De bootstrapper is het onderdeel dat een Prism-applicatie start. Voordat er maar iets op het scherm zichtbaar is, wordt de bootstrapper aangeroepen.   
Ik laat de koppeling met opzet al zo vroeg tijdens het starten van de applicatie plaatsvinden. Dit doe ik om te voorkomen dat de gebruiker moet wachten tot schermen (views) geïnstantieerd zijn. Omdat dit een kleine applicatie betreft, is het geen probleem. Wanneer applicaties groot zijn, kunnen al deze geïnstantieerde views leiden tot een onnodig hoog geheugengebruik.

Bootstrapper

De bootstrapper is een onderdeel van Prism. Normaliter wordt bij een WPF applicatie het hoofdscherm geïnitialiseerd, maar wanneer er gebruik wordt gemaakt van Prism wordt de bootstrapper geïnitialiseerd. De bootstrapper is een class die alle opstarttaken voor een applicatie verricht.

Onder opstartaken wordt verstaan:

* Het koppelen van content aan regio’s
* Het initialiseren van het dependancy-injection framework (MEF)
* Het inladen van eventuele modules
* Het initialiseren van de logging

Wanneer de opstarttaken uitgevoerd zijn, zal de bootstrapper het hoofdscherm starten.

In vrijwel alle gevallen waar de bootstrapper externe resources nodig heeft, wordt gebruik gemaakt van dependency-injection. Omdat in de views gebruik wordt gemaakt van componenten uit het Prism framework, worden views en viewmodels geïnstantieerd via dependancy-injection.

Type Injecties

* Constructor Injection – Entiteiten worden meegegeven via de constructor. Het framework wat verantwoordelijk is voor de dependency-injection zoekt uit een container met instanties het juiste object, en geeft deze tijdens het instantieren van het object mee via de constructor.
* Method Injection – Vergelijkbaar met constructor injection. Het framework dat verantwoordelijk is voor de dependency injection geeft een dependency-object mee als attribuut van een method welke aangeroepen wordt.
* Property Injection – De setter van een property wordt gebruikt om een dependency-object te injecteren.
* Service Locator – *In sommige gevallen is het niet handig om dependency-objecten te injecteren via methods, properties of de constructor. Een voorbeeld hiervoor is als er veel dependency-objecten nodig zijn. Je zou hierdoor methods krijgen met een erg grote aanroep. Het nadeel van deze method is dat het testen meer werk vereist. Er moet een mockup gemaakt worden van de container met dependency-objecten.*

// Dashboard

regionManager.RegisterViewWithRegion(RegionNames.DashboarItemsRegion, () => ServiceLocator.Current.GetInstance<ErrorsWarningsItemView>());

regionManager.RegisterViewWithRegion(RegionNames.DashboarItemsRegion, () => ServiceLocator.Current.GetInstance<ExecutingJobsItemView>());

regionManager.RegisterViewWithRegion(RegionNames.DashboarItemsRegion, () => ServiceLocator.Current.GetInstance<ProblemEntitiesView>());

Voorbeeld van het gebruik van de Serivce Locator om views te instantiëren en te koppelen aan een regio.

Omdat de views afhankelijk zijn van viewmodels, maak ik in de views gebruik van constructor injection. Op deze manier zorgt het framework dat verantwoordelijk is voor de dependency injection dat het juiste viewmodel beschikbaar komt in de view. Omdat een view enkel voorschrijft dat een viewmodel een object moet zijn, was het niet mogelijk om property injection te gebruiken, ik heb daarom gebruik gemaakt van constructor injection.

[Export]

public partial class ErrorsWarningsItemView : UserControl

{

[ImportingConstructor]

public ErrorsWarningsItemView(ErrorsWarningsItemViewModel vm)

{

InitializeComponent();

DataContext = vm;

}

}

Voorbeeld van het gebruik van een constructor injectie bij een view.

**Patterns**  
Ik heb het framework ontworpen met de gedachtegang dat alle functionaliteit middels een zelfde “patroon” kan worden toegevoegd. Naar mijn mening is een framework er onder andere om veel taken automatisch te verrichten. Er hoeft dan dus ook het minimale te gebeuren om nieuwe functionaliteit (een scherm) toe te voegen. Het patroon dat ik gebruik sluit aan bij het model-view-viewmodel (MVVM) pattern.

Model-View-ViewModel pattern (MVVM)

MVVM is een pattern dat veel wordt gebruikt in combinatie met WPF en Silverlight applicaties. Het pattern realiseert een ontkoppeling van de view, het gedrag in een view, de data, en het gedrag dat samen gaat met de data.  
  
In het model bevind zich de data en het daarbij horende gedrag. In het viewmodel wordt het gedrag van de data vertaald naar het gedrag dat gewenst is in een bepaald scherm. In de view wordt de visuele weergave van het viewmodel aangeboden.



*Weergave van het MVVM pattern.*

De volgende stappen moeten worden uitgevoerd om een nieuw scherm toe te voegen:

1. Voeg een view toe
2. Voeg een viewmodel toe, en zorg dat deze overerft van “ViewModelBase”
3. (eventueel) voeg een model toe
4. Koppel het viewmodel aan de view
5. Programmeer de view en interactie met het viewmodel
6. Registreer de view in de “ViewRegistry”

Doordat de view geregistreerd staat in de ViewRegistry, zal deze automatisch tijdens het opstarten van de applicatie worden geïnstantieerd. Het is een bewuste keuze geweest om alle views te instantiëren tijdens het opstarten. Wanneer een gebruiker op een knop in de applicatie drukt, moet direct het volgende scherm verschijnen, het is niet prettig als je hier enige tijd moet wachten tot een scherm is geïnstantieerd.



Hierboven een schematische weergave (domeinmodel) zoals het pattern wordt geïmplementeerd.

In bovenstaande afbeelding is ook een “ViewModelBase” te zien. Deze base-class biedt functionaliteit om views te laten updaten wanneer het viewmodel is aangepast. Omdat dit voor alle viewmodels het zelfde is, heb ik gekozen voor een constructie met overerving.

Vrijwel alle functionaliteit in de applicatie is middels bovenstaand pattern geïmplementeerd. Ik vond het belangrijk dat alles op een gelijke manier geïmplementeerd zou worden. Dit vergroot namelijk de onderhoudbaarheid van de applicatie. Als een ontwikkelaar dit “patroon” begrijpt, kan hij eenvoudig nieuwe functionaliteit toevoegen en eenvoudig functionaliteit bewerken.

### ConnectionHelper

Omdat de applicatie vrijwel alle data ophaalt bij een WCF-service, moet er een onderdeel zijn wat de status controleert. Het uitvoeren van acties op een webservice die niet reageert leidt tot excepties. Het is helaas niet mogelijk om events aan een WCF-service te hangen die af gaan op het moment dat de connectie verbroken is. Deze functionaliteit was wel gewenst omdat de dashboard-items moeten stoppen met data ophalen op het moment dat de WCF-service niet bereikbaar is.

Ook zullen andere onderdelen van de applicatie moeten reageren op een wijziging in de status van de connectie. Hierbij valt te denken aan lijsten die moeten worden ververst, stukken applicatie die moeten worden vergrendeld, etc.  
Omdat de ConnectionHelper door de gehele applicatie beschikbaar moet zijn, heb ik deze ontworpen als Singleton. Op deze manier kunnen alle viewmodels bij de ConnectionHelper, en wordt er voorkomen dat er meerdere instanties actief zijn.

De ConnectionHelper controleert met een bepaald interval of de WCF-service nog actief is. Wanneer status van de service gewijzigd is, wordt het “ConnectionStatusChanged” event afgevuurd. Functionaliteit waarbij de status van belang is, kan zich aanmelden op dit event. Wanneer de status van de connectie wijzigt, kan functionaliteit hier rekening mee houden.

### Dashboard items

Toen een groot deel van het framework opgezet was, was het taak om te werken aan de eerste “dashboard items.” Uit gesprekken welke ik met de opdrachtgever heb gevoerd, is gebleken dat de applicatie moet opstarten, met als eerste scherm een dashboard. Op dit dashboard moet duidelijk zijn wat de status is van het berichten verkeer (zijn er errors, warnings, berichten die niet verwerkt kunnen worden etc.). Er is besloten om op het dashboard meerdere kleine schermen te tonen, die elk informatie tonen met betrekking tot de status van het berichten verkeer.



Het dashboard scherm zoals deze in de eerste iteratie is opgenomen in het functioneel ontwerp.

Interessant aan de dashboard items is ook dat deze “live” informatie moeten kunnen tonen. Informatie wordt hier op de achtergrond opgehaald. Dit moet omdat op de voorgrond de “UI-thread” draait. Deze thread is verantwoordelijk voor het updaten van de user interface. Interessante problemen die ik hier tegenkwam hadden te maken met het feit dat ik informatie ophaal via de achtergrond thread. Het is namelijk niet mogelijk om vanuit een achtergrond-thread iets op de userinterface aan te passen. Dit heb ik moeten oplossen met een door de user interface te updaten middels een “Invoke.” Met een invoke wordt er beroep gedaan op een de ui-thread om vervolgens de update uit te voeren.

protected override void DoWork(object sender, DoWorkEventArgs doWorkEventArgs)

{

while (ConnectionHelper.GetInstange().IsConnected)

{

var logs = Client.Log.GetWarningsAndErrors(50, 0);

var useInsert = LogList.Count != 0;

foreach (var processingLogData in logs)

{

var data = processingLogData;

if (LogList.Any(logData => logData.Id == data.Id))

continue;

if(useInsert)

{

Application.Current.Dispatcher.Invoke((Action)(() => LogList.Insert(0, data)));

}

else

{

Application.Current.Dispatcher.Invoke((Action)(() => LogList.Add(data)));

}

}

Thread.Sleep(5000);

}

}

Een voorbeeld van een methode die wordt gebruikt door een background-thread.

Omdat voor alle dashboard-items geldt dat deze “live” data moeten weergeven, is hier het eerder beschreven “pattern” iets aangepast. Het viewmodel erft hier over van “DashboardItemViewModelBase.” Buiten de standaard functionaliteit om views te updaten, biedt dit viewmodel ook functionaliteit voor het ophalen van data middels achtergrondtaken.



Het domeinmodel voor dashboard-items.

### Feedback op applicatie

Gedurende de eerste sprint is er zo goed als geen feedback gegeven op het functioneel ontwerp. Dit heeft als nadeel dat wanneer er fouten gevonden worden, welke hun oorsprong vinden in het functioneel ontwerp, dat deze veel tijd kunnen kosten om op te lossen. Om de eventuele schade beperkt te houden, heb ik geregeld feedback gevraagd op de applicatie.  
Op basis van deze feedback gesprekken heb ik verbeteringen uitgevoerd. Deze verbeteringen waren veelal van kleine aard, en hadden daardoor weinig schade. Enkele voorbeelden van verbeteringen zijn:

* Knoppen op een andere plaats zetten
* Items in lijsten anders sorteren
* Velden read-only weergeven

Deze gesprekken waren ook erg constructief. Omdat er beeld was bij de applicatie, kwamen er veel nieuwe requirements. Ik had al verwacht dat er gedurende de ontwikkeling van de applicatie nieuwe requirements bij zouden komen. Om deze reden wilde ik dan ook gebruik maken van scrum, zodat de requirements voor een latere sprint kunnen worden ingepland.

### Code Analysis & Code Metrics

Aan het einde van de eerste sprint is besloten om Code Analysis en Code Metrics uit te voeren. Er is voor dit moment gekozen omdat er veel code geschreven was, en het project nog in een relatief vroeg stadium was. Hierdoor kan ik leren van de fouten die Code Analysis geeft, en kan ik voorkomen dat ik ze in een later stadium weer maak. Code Analysis bevat een set aan regels waar de code aan voldaan moet worden. Wanneer code analysis uitgevoerd wordt, verschijnen er melden over stukken code die niet aan de regels voldoen. Omdat ik werkte aan een component wat in diverse projecten gebruikt zal gaan worden, stond Code Analysis streng ingesteld.

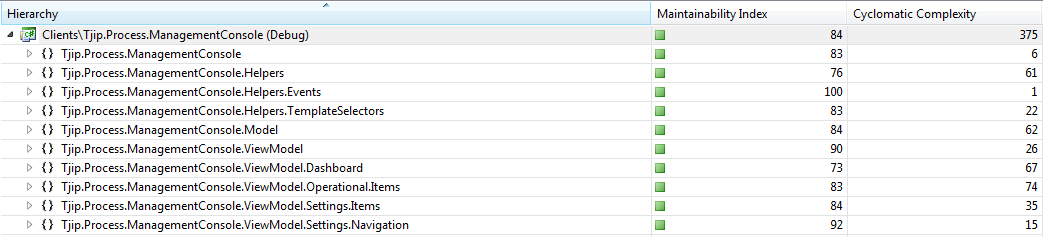
Tijdens het uitvoeren van Code Analysis kon ik concluderen dat ik alles redelijk in elkaar had gezet. Er waren negen meldingen, waarvan het grootste deel ging over access-levels van methods en properties. Ik had op veel plaatsen gebruik gemaakt van een public access level, waar deze eigenlijk protected of private had moeten zijn.   
Een andere melding ging over de implementatie van een interface die werd gebruikt in .NET Van deze interface is later een “generic” versie verschenen in .NET 2.0. Volgens de regel moest deze “generic” interface ook worden geïmplementeerd, zodat mijn implementatie ook compatible is met “generics.”

Dit is de eerste keer dat ik werkte met Code Analysis, en ik vond de meldingen die werden gegeven erg interessant. Het zorgt er voor dat je nogmaals naar de code kijkt, en het dwingt af om net te programmeren. Bij de meldingen die worden gegeven wordt ook een korte omschrijving gegeven waarom de regel bestaat, en welk risico je loopt wanneer de code niet aangepast wordt. Ik vond dit erg leerzaam, en het verbeterd ook de kwaliteit van de code.

Veel code-analysismeldingen konden worden voorkomen door het gebruik van Resharper. Resharper is een gereedschap dat binnen TJIP wordt gebruikt om code leesbaar te houden. De code-conventie wordt ook afgedwongen door dit programma.

Naast Code Analysis werd ook Code Metrics uitgevoerd. Code Metrics levert net als Code Analysis meldingen over stukken code. Code Metrics let echter niet op een set met regels, maar controleert de code op complexiteit. Wanneer er teveel logica in een method zit, zal Code Metrics een laag cijfer aan deze method geven. Het doel hierachter is om functionaliteit (waar mogelijk) op te splitsen in kleine logische stukken.

De resultaten van CodeMetrics waren positief. Er was slechts één plek in de code waar software te complex werd. Dit betrof het stuk code waar views aan regions worden gekoppeld. Iedere view moet hier handmatig gekoppeld worden aan een region. Door de hoeveelheid views werd deze method te groot. Ik heb hiertoe de views uitgesplitst per categorie in de applicatie, en deze een losse method gegeven. Dit werd goed bevonden door Code Metrics.



Een overzicht van de resultaten afkomstig uit Code Metrics na verbetering van de code.

### Paged WCF Collection

Een uitdaging was het bouwen van een “Paged collection” waarbij de data verkregen moest worden via de WCF-service. Met de user-controls van Telerik is het mogelijk om paginatie in te gebruiken. De gebruiker krijgt dan de mogelijkheid om door een collectie heen te bladeren. Paginatie moest onder andere gebruikt worden voor de lijst met berichten, en in de executielog.

Er waren twee redenen waarom ik heb gekozen om paginatie te gebruiken:

* De gebruiker krijgt niet een lijst gepresenteerd welke oneindig door kan groeien
* Het is mogelijk om een klein set aan berichten op te halen bij de WCF-service in plaats van de volledige lijst. De volledige lijst ophalen zou teveel tijd gaan kosten, en daar deze lijst alsmaar groter wordt, zal het laden steeds langer duren. Door slechts enkele records uit de lijst te laden, blijft het laden van de lijst altijd even snel.

De uitdaging zat hem in het feit dat ik standaardfunctionaliteit van Telerik moest laten werken met een WCF-service. Na het lezen van diverse websites ben ik er achter gekomen dat ik drie interfaces moest implementeren, te weten:

* IEnumerable<T>,
* IPagedCollectionView,
* INotifyPropertyChanged,
* INotifyCollectionChanged

Middels de combinatie van deze interfaces is het mogelijk om een collectie te maken waarin paginatie wordt toegepast. Omdat ik hier zelf een implementatie kon maken van de collectie, was het mogelijk om deze te laten communiceren met de WCF service. Op het moment dat een user-control de enumerator opvraagt, zal deze collectie een request uitvoeren naar de webservice. Deze webservice geeft een collectie terug; van deze collectie wordt de enumerator vervolgens weer doorgegeven.

### Maken testscripts

Conform het testplan moest er getest worden om te kunnen garanderen dat de applicatie een bepaalde kwaliteit levert. Veel van de functionaliteit maakt gebruik van de WCF service. Wanneer je de applicatie goed zou willen testen, zou je een mockup moeten maken van de WCF service. Een mockup is een versie welke gebruikt kan worden voor testdoeleinden. Het maken van een mockup vergt echter veel tijd.   
In overleg met de opdrachtgever heb ik besloten om geen mockup te maken. Het resultaat hiervan is dat een groot aantal unit-tests niet gebouwd kunnen worden. De code is echter wel gedekt. Ik heb getest door systeemtesten te ontwikkelen, en deze te automatiseren. Bij een unit-test ligt de focus op een specifiek stuk code, bij een systeemtest op de volledige keten. Het nadeel hieraan is dat wanneer er een bevinding wordt gedaan, deze lastiger is om te adresseren in de code.

Het maken van testscripts vond ik een ingewikkelde zaak. Mijn doelstelling was om per besproken stuk functionaliteit in het functioneel ontwerp een test te ontwikkelen. Voor de functionele eisen waarbij uitzonderingen voorkomen, zijn dit uiteraard meerdere tests geworden. Tijdens het bouwen van de test moet je even uit de “ontwikkelaars schoenen” stappen. Als ontwikkelaar test je tijdens het debuggen vaak met extreme waarden (extreem onjuiste waarde, en waarde welke vaak ruim binnen het bereik liggen) en waarden waarvan je zeker bent dat deze een bepaald resultaat opleveren. Tijdens het testen vind ik het belangrijk om op de grenswaarde te testen, hiermee is het namelijk direct ook duidelijk of de grens bij de juiste waarde is geïmplementeerd.

Omdat de applicatie veelal bedoelt is om de status van TJIP Processmanager af te lezen, heb ik in veel situaties ook gebruik gemaakt van testen waarbij twee clients verbonden zijn met de Processmanager. Hiermee kan de gebruiker in de ene client een actie uitvoeren, waarvan het resultaat in de andere client zichtbaar moet komen.

|  |  |  |
| --- | --- | --- |
| **Stap** | **Actie** | **Verwacht resultaat** |
| 1 | Start Processmanager Start een Processmanager Client (A)  Start een Processmanager Client (B) | Er draaien twee clients en een instantie van TJIP Processmanager |
| 2 | Ga in client A naar “Operations 🡪 Run Jobs”. | Een lijst met uit te voeren jobs wordt getoond. |
| 3 | Selecteer Job: “TimeSpandingJob3” | “TimeSpandingJob3” wordt gemarkeerd als geselecteerd |
| 4 | Klik op de knop “Run Job” | In client B moet op het dashboard verschijnen bij het onderdeel “Executing Jobs” dat “TimeSpandingJob3” aan het draaien is |

Eenvoudig voorbeeld waar twee clients in een test actief zijn op een enkele instantie van TJIP Processmanager.

De uit de test gekomen bevindingen zijn geregistreerd, en als bijlage toegevoegd aan de testscripts. Dit laatste was een wens vanuit TJIP. Op deze manier is het voor hun inzichtelijk welke bevindingen er gekomen zijn uit een bepaalde test.

### Scrum

Gedurende deze spint zijn er requirements bij gekomen of gewijzigd. Ook zijn er een aantal dingen anders verlopen dan gewenst. In overleg met de opdrachtgever is besloten welke prioriteit deze requirements krijgen.

Het is de bedoeling dat Scrum wordt uitgevoerd in teamverband, doordat dit een project is waar enkel een persoon op actief is, zijn er geen daily standups geweest.

Hierboven een weergave van de burndown die gemaakt is voor sprint één. De grafiek is in het begin erg volgens het ideale beeld. Hier verliep alles nagenoeg volgens planning. Vanaf halverwege week twee tot het einde, liep alles iets minder volgens de planning. Dit had ermee te maken dat het bouwen van bepaalde onderdelen in het framework meer tijd in beslag namen dan verwacht. In dit geval ging het om de ConnectionHelper, hier was een probleem dat requests nog werden uitgevoerd op het moment dat de Processmanager niet beschikbaar was. Op het einde is deze tijd weer ingehaald doordat de detailschermen van berichten te ruim gepland waren.

## Sprint 2

De tweede sprint stond in het teken van het toevoegen van nieuwe functionaliteit aan de applicatie. Met name het operationele gedeelte, het gedeelte waar het berichtenverkeer gemonitord kan worden, moest uitgebreid worden.

### Feedback gesprek/brainstorm sessie met klant

Tot nu toe waren alle requirements opgesteld door een medewerker van TJIP. In sprint twee was een sessie georganiseerd waarbij een korte demo werd gegeven aan een klant, en er hierna gekeken zou worden of er nog aanvullende requirements waren. Dit was een zeer nuttige sessie. De klant gaf hierin aan dat voornamelijk de configuratie van Processmanager lastig is. Dit werkt via XML bestanden, en is daardoor lastiger om aan te passen dan een configuratie via een GUI.

Op basis van wensen van de klant is in overleg met de opdrachtgever besloten om de planning te herzien, en meer focus te leggen op de configuratie. Een eerste stap hier naartoe was om de volledige configuratie zichtbaar te hebben, in plaats van bepaalde onderdelen. Deze stap is ingepland voor sprint drie.

De klant gaf ook nog andere zaken aan welke interessant waren om in Processmanager te krijgen:

* Mogelijkheid tot het aansluiten van een archief-systeem
* Statistieken

### Downloaden bericht

Een requirement was dat het systeem in staat was om ook de inhoud van berichten te tonen. Dit was in eerste instantie makkelijk. In het detailscherm van een bericht zou een onderdeel “Message” komen met daarin het volledige bericht. Dit werkte allemaal goed; totdat ik ging testen met grotere berichten. Het duurde bij grotere berichten langer om hiervan de details op te roepen.  
Gezien het feit dat je lang niet altijd het bericht hoeft in te zien, heb ik hier in overleg met de opdrachtgever een passende oplossing voor bedacht, namelijk een knop “open message.” Wanneer de gebruiker op deze knop klikt, wordt het bericht op de achtergrond gedownload naar de tijdelijke map van de aangemelde gebruiker, en wordt het bericht vervolgens geopend met een passende applicatie. Voor vrijwel alle berichten geld dat deze worden opgeslagen als tekstbestand of xml-bestand. Ingaande en uitgaande e-mails worden opgeslagen als “.eml” bestand. Dit bestand kan vervolgens met bijvoorbeeld Microsoft Outlook worden geopend.

### Template Selector

De detailschermen voor berichten waren een interessant issue. Veel eigenschappen van de berichten zijn gemeenschappelijk, en een ander deel van de eigenschappen zijn weer verschillend. Op view-model niveau heb ik dit opgelost middels overerving. Ik heb alle gelijkwaardige attributen in een base-class gezet, en de verschillende attributen per bericht in classes welke overerven van de base-class. Dit was op zichzelf geen uitdaging. De uitdaging zat in het bouwen van de schermen.   
Het leek mij niet handig om voor ieder type bericht een nieuw scherm te ontwerpen. Wanneer er dan één generiek attribuut wijzigt, betekend dit dat ik voor alle type berichten het scherm moet aanpassen. Binnen TJIP was mij meerdere malen verteld dat WPF een nieuwe technologie is, en er voor veel standaardproblemen oplossingen zijn. Ik ben gaan zoeken of WPF een oplossing heeft voor dit probleem. Binnen WPF kun je dergelijke problemen oplossen met een template-selector.   
Het idee achter de template-selector is dat er een generiek scherm is, met daarin een bepaald component (ContentControl). In d-it component is het mogelijk om een template (deel van een scherm) in te laden. Afhankelijk van een zelf te bepalen waarde is het mogelijk om de template-selector een template te laten selecteren.

<ContentControl ContentTemplateSelector="{StaticResource ResourceKey=MessagesDetailsTemplateSelector}" Content="{Binding EntityData}"/>

<GroupBox Header="Common Properties">

<Grid>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="150" />

<ColumnDefinition />

</Grid.ColumnDefinitions>

…

XAML code waarin het component “ContentControl” is opgenomen waar datatemplates in geladen kunnen worden.

public override DataTemplate SelectTemplate(object item, System.Windows.DependencyObject container)

{

var message = (EntityData) item;

DataTemplate result = null;

if (message != null)

switch(message.EntityTypeName)

{

case "IncomingEmail":

result = IncommingEmailTemplate;

break;

case "OutgoingEmail":

result = OutgoingEmailTemplate;

break;

case "OutgoingSms":

result = OutgoingSmsTemplate;

break;

case "OutgoingFax":

result = OutgoingFaxTemplate;

break;

case "IncomingMessage":

result = IncomingMessageTemplate;

break;

case "OutgoingMessage":

result = OutgoingMessageTemplate;

break;

}

return result;

}

### Project voor datageneratie

Vanuit TJIP bestond er de wens om een set aan jobs te hebben, die Processmanager vullen met testdata. Dergelijke jobs zouden gebruikt kunnen worden wanneer er een demonstratie moet worden gegeven aan klanten. Ik heb dergelijke testjobs in sprint twee ontwikkeld om de volgende redenen:

* Er is in sprint drie een demo voor het architectenteam van TJIP. Het is verstandig om realistische data te hebben voor deze demo, zodat er een realistisch beeld kan ontstaan bij de architecten. De architecten zijn immers steakholders, en kunnen requirements aandragen.
* Testdata is handig om te controleren of de gebruikersinterface goed werkt. In veel gevallen moeten schermen inkrimpen of uitrekken aan hand van de hoeveelheid data. Middels de data die voortkomt uit deze test is het mogelijk om dit te testen.

Dit is niet de eerste keer dat ik iets uitvoer omtrent datageneratie. Het is wel de eerste keer dat ik de applicatie aanstuur om testdata te genereren. Normaal zou ik kiezen voor een “data generation plan” in een database project. Er zijn in dit specifieke geval een aantal reden waarom ik dat nu anders doe:

* Het is niet mogelijk om berichten te genereren met bijlagen die ook door externe applicaties geopend kunnen worden.
* Er is geen database project in de solution aanwezig.

Een bijkomend voordeel van het schrijven van dergelijke testjobs is dat Processmanager nogmaals getest wordt.

### Dashboard schermen

De dashboardschermen bieden in een oogopslag de actuele stand van zaken: welke jobs er draaien, welke errors er waren en welke probleemberichten er zijn. Deze data moet actueel zijn. Het is erg vervelend als medewerker “A” bezig een probleem te verhelpen, en er op een laat moment achter komt dat medewerker “B” net klaar is met het oplossen van een probleem.   
Om dit te voorkomen worden de dashboardschermen voortdurend bijgewerkt. Tijdens sprint 1 is er een implementatie gedaan waarbij de schermen volledig leeg worden gemaakt, en direct daarna opnieuw gevuld met actuele data. Dit leverde echter een probleem op. Doordat de tabel zo snel wordt leeg gemaakt, en gevult, leek het alsof het dashboard item flitste. Dit was niet prettig om lang naar te moeten kijken.

De oplossing die ik hierop bedacht heb was om iets te doen met het verschil tussen de tabel welke opgehaald is via de WCF service en de tabel op het scherm. Afhankelijk van het type verschil moest er een item worden verwijderd uit de tabel van het scherm of een item worden toegevoegd aan de tabel op het scherm. Dit leverde het volgende stuk code op:

List<EntityData> entities = Client.Operation.GetProblemEntities(50, 0);

// Remove old items from the list

foreach (var entityData in Entities.Except(entities).ToList())

{

var data = entityData;

Application.Current.Dispatcher.Invoke((Action)(() => Entities.Remove(data)));

}

// Add new items to the list

foreach (var entityData in entities)

{

if (!Entities.Contains(entityData))

{

var data = entityData;

Application.Current.Dispatcher.Invoke((Action)(() => Entities.Insert(0, data)));

}

}

Het verversen van een dashboard-scherm. Middels het Except() statement wordt het verschil berekend tussen de twee collecties.

### Testen

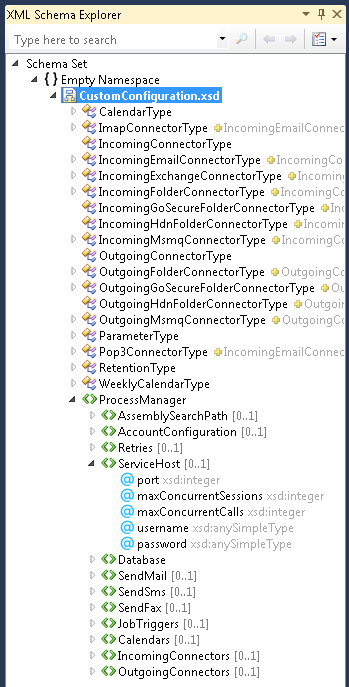
Nu er een project beschikbaar was voor het genereren van testdata, kon ook de GUI goed getest worden. Daar er voor mij geen methode bekend was om een GUI te testen, heb ik besloten om de schermen stuk voor stuk te gaan bekijken, en controleren of deze aansluiten bij het functioneel ontwerp.  
Bevindingen die uit deze test kwamen hadden met name te maken met kolommen die niet breed genoeg waren en teksten die onjuist waren overgenomen uit het functioneel ontwerp.

## Sprint 3

### XML & XSD

Tot dit punt was het tot nu toe altijd mogelijk om de WCF service uit te bouwen middels aanwezige gegevens in het prototype en/of gegevens halen uit de database.   
Een belangrijk deel van de Processmanager ManagementConsole is dat daar op den duur de Processmanager mee ingesteld en ingeregeld kan worden, in plaats van werken met XML bestanden. De eerste stap hier naartoe was het uitlezen van de configuratie en deze weergeven in de Managementconsole.   
Een probleem waar ik hier tegen aan liep was: “Welke instellingen zijn er allemaal, en welke type waarde moeten deze bevatten?” Met deze vraag ben ik ook langs de opdrachtgever gegaan. De opdrachtgever wist mij te vertellen dat er van het XML-bestand een XSD-schema is, waar deze gegevens in staan.

Een XSD-schema is een schema waarin vermeld staat welke gegevens er in een Xml-bestand kunnen worden bijgehouden, en hoe de hiërarchie van deze gegevens is. Gezien de volledige configuratie in XML instelbaar is, was dit dus een mooi middel om te achterhalen welke instellingen er waren.



Weergave van een XSD in Microsoft Visual Studio

### Editors voor SMS, Mail & Fax

Om te kunnen testen, en handmatig e-mails/Sms’en en Faxen te versturen was er de wens om editors te ontwikkelen. In deze editors moest het mogelijk zijn om een bericht (inclusief bijlagen) samen te stellen, en vervolgens te versturen.

Voor deze schermen moest een nieuw deel aan het framework worden gebouwd. Ik had in de opzet geen rekening ermee gehouden dat er ook dergelijke “tools” in moesten komen. Ik doel hier vooral mee op het visuele deel van het framework. Een e-mail editor valt niet onder de operationele sectie van Processmanager, en al helemaal niet onder de sectie “Configuratie.” Omdat er wellicht meer tools volgen in de toekomst, heb ik besloten om de sectie “tools” in het leven te roepen. Om de tools toegankelijk te houden, heb ik besloten om er een menu voor aan te maken.

### Verbeteren ConnectionHelper

Er waren een aantal nadelen aan het gebruik van de ConnectionHelper zoals deze is ontwikkeld in de eerste sprint. In de oude vorm haalt de ConnectionHelper om de 500ms het versienummer op. Indien dit lukt, is dit het teken dat de verbinding nog actief is. Wanneer dit mislukt, betekend het dat de applicatie weer in de state “verbinden” moet komen.

De oude manier had als nadeel dat er onnodig veel requests werden uitgevoerd naar de WCF service. Bij grote hoeveelheden verkeer zou dit eventueel tot problemen kunnen leiden.   
Om dit probleem te verhelpen heb ik de ConnectionHelper herzien. In de nieuwe versie zit enkel nog een polling-mechanisme wanneer de state “verbinden” actief is. De nieuwe ConnectionHelper wijzigt van state op het moment dat een call richting de WCF server resulteert in een “EndpointNotFoundException.” Deze exceptie wordt gegooid op het moment dat er geen verbinding gemaakt kan worden met de WCF server.

Op basis van deze exceptie werd in het polling-mechanisme van de oude ConnectionHelper bepaald wat de status van de verbinding is. Na even logisch nadenken kwam ik tot de conclusie dat het hele polling-mechanisme in de oude ConnectionHelper overbodig is. Op het moment dat er een actie wordt uitgevoerd in de ManagementConsole die leidt tot deze exceptie, kan de state van de verbinding ook worden aangepast. Hierdoor wordt de state van de verbinding niet actief bijgehouden, maar gewijzigd op het moment dat er pas interactie hoeft te zijn met de WCF service.

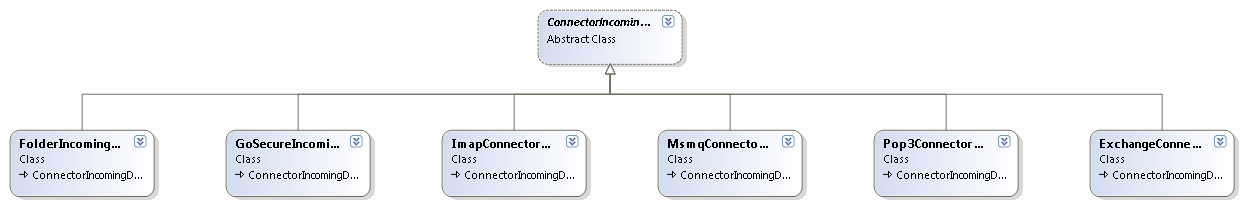


De verbinding van de ManagementConsole weergegeven in een state-diagram. In de opmerking velden wordt aangegeven wanneer de het polling-mechanisme in werking wordt gezet.

### WCF & Serialization

In de WCF-service maak ik gebruik van data-objecten. Dit zijn objecten welke enkel data bevatten en geen functionaliteit in zich hebben. De data-objecten zijn objecten die via de WCF-service doorgegeven kunnen worden aan de ManagementConsole. Omdat ik niet alle velden over wil nemen, en de normale objecten in Processmanager gegenereerd worden door EntityFramework, heb ik besloten om data-objecten te gebruiken. Hiermee heb ik zelf in de hand welke informatie er uit de Processmanager mag komen.

Voor het weergeven van connectoren en hun configuratie had ik besloten om gebruik te maken van een overervingsconstructie. Ik heb dit gedaan omdat er een aantal gegevens zijn in de connectoren welke voor iedere connector aanwezig zijn.



Deze constructie heeft bij mij geleid tot een groot probleem. Na het inbouwen van deze constructie weigerde de WCF service verdere dienst, en gaf geen foutmelding. Hierdoor ben ik een hele dag bezig geweest met het zoeken naar een mogelijke fout. Ik kon hier helaas geen fout vinden. Na een tijd debuggen en zoeken op het internet naar eventuele oplossingen kwam ik er achter dat het in WCF-services in sommige omstandigheden nodig is om “known types” aan te geven bij de base-class. Door aan te geven wat de “known types” zijn, weet WCF welke objecten er kunnen overerven van de base-class.

Vaak is dit niet nodig, en werkt WCF prima door enkel aan te geven dat de base-class een “datacontract” is. (een class die via WCF geserialeerd mag worden, en gebruikt kan worden aan de client-kant.). In mijn geval wilde ik de informatie van de connectoren opslaan in een lijst, en die via WCF versturen. Hierdoor ontstaat er dus een lijst van verschillende type objecten welke allemaal overerven van de base-class.   
In een dergelijk geval is het dus nodig om aan te geven welke objecten er overerven van de base-class. Een nadeel aan deze methode is dat wanneer er in dit geval een nieuwe overerving is op de base-class, de base-class ook bewerkt moet worden. Omdat het enkel één regel code is die erbij geschreven moet worden, heb ik deze oplossing geaccepteerd.

[DataContract]

[KnownType(typeof(ExchangeConnectorIncomingData))]

[KnownType(typeof(GoSecureIncomingConnectorData))]

[KnownType(typeof(ImapConnectorData))]

[KnownType(typeof(FolderIncomingConnectorData))]

[KnownType(typeof(MsmqConnectorIncomingData))]

[KnownType(typeof(Pop3ConnectorData))]

public abstract class ConnectorIncomingDataBase

{

[DataMember]

public string Name { get; set; }

[DataMember]

public string Description { get; set; }

[DataMember]

public string ConsumingJobName { get; set; }

}

Een voorbeeld van het gebruik van “KnownType.”

### Demo aan architecten-team

In deze sprint is er ook een demo aan het architecten-team gegeven. In deze demo heb ik laten zien wat er gebruikt is, en is er een verdere strategie omtrent de ontwikkeling van Processmanager bepaald. Deze strategie is deels van toepassing op deze afstudeeropdracht, maar gaat voor een groot deel over wat er na deze afstudeeropdracht zal gebeuren.

Tijdens de demo ben ik door alle schermen heen gelopen, verteld waar deze schermen voor dienen en hoe de schermen werken. Ik heb hiermee geprobeerd een beeld te geven van wat de applicatie kan, en hoe deze werkt. Doordat ik eerder een set aan testjobs heb gebouwd, was het mogelijk om enkele scenario’s te demonstreren.

Uit de demo kwam naar voren dat er een aantal zaken zijn die ontbraken, en voor klanten wenselijk waren. Deze zaken zijn vervolgens geprioriteerd, en zullen op volgorde van prioriteit worden gerealiseerd:

1. Het visualiseren van connectoren, triggers en jobs.
2. Mogelijkheid tot uitbreiden op berichtniveau (het kunnen archiveren van een bericht nadat deze is verwerkt door Processmanager).
3. Mogelijkheid tot het uitvoeren van rapportages.
4. De configuratie bewerken via de management-console en niet via een XML bestand

Bovenstaande prioritering was een voorstel van een architect in het overleg. Ik heb dit voorstel gesteund om de volgende redenen:

* Uit gesprekken met medewerkers en klanten begrijp ik dat het vaak niet helemaal duidelijk is hoe Processmanager werkt. Een visuele weergave zou hier een hoop duidelijkheid in kunnen verschaffen. Ook valt middels deze visuele weergave te controleren of de instellingen juist zijn ingevoerd.
* Op basis van een feedbackmoment in sprint twee met een klant, werd ook al duidelijk dat er een mogelijkheid ontbreekt om acties op berichten uit te voeren nadat deze verwerkt zijn. Dit zou mogelijk worden op het moment dat er op berichtniveau uitgebreid kan worden. Omdat er op dit moment alternatieve wijze zijn om berichten te archiveren (vanuit de applicatie welke gebruik maakt van TJIP Processmanager), was ik het er mee eens dat dit een lagere prioriteit heeft dan de visualisatie.
* Het is interessant voor systeembeheerders en managementleden om te zien hoeveel berichten er door TJIP Processmanager verwerkt zijn, hoeveel berichten er goed verwerkt zijn, en hoeveel berichten er niet verwerkt zijn. Deze cijfers zijn belangrijk en zeer nuttig. In veel gevallen moeten bedrijven alleen volgens de wet alle berichten richting een klant bewaren. Om deze reden was ik het eens om de rapportage functionaliteit lager te prioriteren dan de uitbreidbaarheid op berichtniveau.
* Het is erg gebruikersvriendelijk om de configuratie in te kunnen regelen via de Management Console. Normaliter zal de configuratie alleen bijna nooit wijzigen. Om deze reden was ik het eens om deze wens als laagste te prioriteren.

Een ander belangrijk punt wat ook uit de demo naar voren kwam had te maken met de uitklapbare tabellen. Deze werden als rommelig ervaren.   
Dit is een prijzige fout welke door het reviewen van het functioneel ontwerp voorkomen had kunnen worden.



Voorbeeld van een uitklapbare tabel voor het weergeven van berichtdetails.

Op basis van het overleg heb ik voorgesteld om een WCF method te ontwikkelen die het mogelijk maakt om de status van een bericht op te vragen. Dit was tot op heden niet mogelijk, en leek mij zeer interessant. Dit zou een eerste kleine stap zijn om acties uit te voeren nadat Processmanager een bericht verwerkt heeft.  
Dit idee werd als enthousiast ontvangen, en mocht geïmplementeerd worden in sprint 4.

### Testen

Buiten de reguliere testen welke ik gemaakt heb voor de gebouwde onderdelen, stond er voor deze sprint ook een speciale test op het programma: “Het verzenden van berichten testen met grote bijlagen.” Ik wilde deze test uitvoeren omdat de opdrachtgever heeft aangegeven zich zorgen te maken om de volgende scenario’s:

* Berichten die niet of onjuist aankomen omdat de bijlage te groot is.
* De mogelijkheid dat Processmanager niet reageert omdat er een bericht wordt gestuurd met een (te) grote bijlage.

Ook was deze test interessant om aan te kunnen geven wat de maximale grote is van een bijlage waar Processmanager mee om kan gaan. De recent ontwikkelde tools boden een mooie uitkomst om een dergelijke test uit te voeren.

Ik ben begonnen met het versturen van een e-mail met een bijlage van 100mb. Na lang laden werd de e-mail verstuurd. Tijdens het versturen van de e-mail was Processmanager nog gewoon toegankelijk. Bij het openen van de e-mail bleek de bijlage te ontbreken. Dit vond ik vreemd omdat het e-mail bestand (.eml) wel groter was dan 100mb.   
Ik vroeg mij hierdoor af of er niet een fysieke beperking zat in de grote van bijlagen in e-mails. Na het een en ander opgezocht te hebben, blijkt er geen fysieke beperking te zijn in de grote van bijlages in een e-mail. Het is alleen niet aan te raden om bijlages te gebruiken die groter zijn dan 10mb. Dit omdat in e-mail servers gebruik wordt gemaakt van een store-forward principe. Dit houdt in dat de e-mail eerst wordt opgeslagen voordat hij verder wordt verstuurd. Veel mailservers hanteren een maximale grote van 10mb per e-mail. Er zijn enkele uitzonderingen waarbij er 25mb mag worden toegevoegd aan een e-mailbericht.   
Met deze informatie ben ik langs de opdrachtgever gedaan, en is er besloten om een maximale grote van 25 te hanteren. Deze grote moet echter aanpasbaar zijn, en is daarom opgenomen als configuratie item van de client-library. Deze zelfde beperking is ook ingesteld voor bijlages aan een fax. Dit omdat de FaxConnector faxen eventueel kan versturen per e-mail.

Het versturen van bijlagen met een grote van 25mb leverde geen problemen op.

## Sprint 4

### WCF method voor het opvragen van status berichtverwerking

Om TJIP Processmanager beter te laten integreren in andere applicaties heb ik in sprint drie een idee bedacht om de status van een bericht op te halen. Dit idee was geïnspireerd door iDEAL. iDEAL zelf geeft geen feedback of een betaling is gelukt. Het resultaat van de betaling kan opgevraagd worden bij de iDEAL service.   
Deze manier om een resultaat te verkrijgen leek mij zeer geschikt om te gebruiken in Processmanager. Dit omdat de totale duur om een bericht te verwerken soms lang kan zijn. De client applicatie kan met een bepaalde interval dan aan processmanager vragen wat de status is. Omdat Processmanager al een webservice heeft, leek het mij ook vrij eenvoudig om deze method in te bouwen.

Interessant aan het bouwen van deze method was dat er een enumeratie wordt terug gegeven als resultaat. Deze enumeratie bevat de volgende statussen:

* Onbekend
* MessageSent
* MessageReceivedAtRecipient
* MessageNotSent

Ik had verwacht dat een enumeratie “out-of-the-box” gebruikt zou kunnen worden. Dit bleek niet het geval. Ook voor enumeraties moet worden aangegeven dat deze door WCF clients gebruikt mogen worden.

[DataContract(Name = "MessageStatus")]

public enum MessageStatus

{

[EnumMember]

Unknown,

[EnumMember]

MessageSent,

[EnumMember]

MessageReceivedAtRecipient,

[EnumMember]

MessageNotSent

}

# Evaluatie

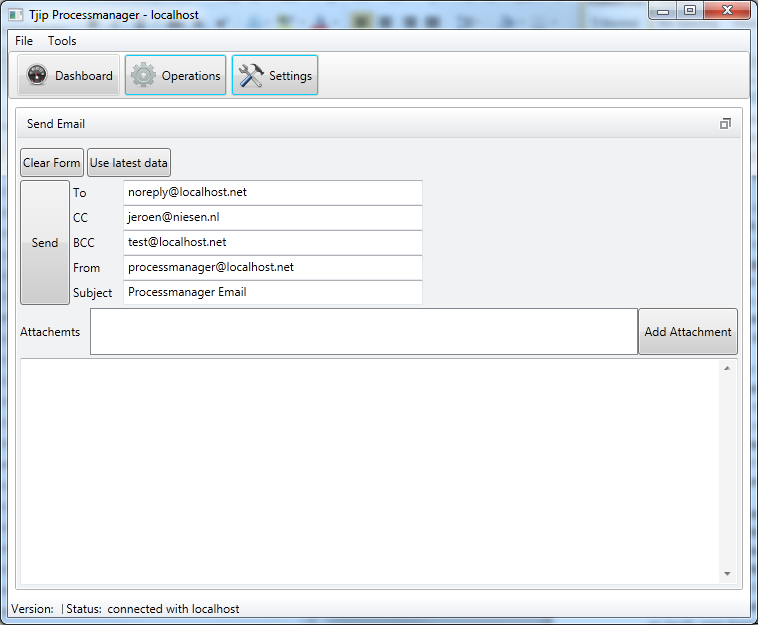
## Productevaluatie

Ik ben zeer tevreden over de applicatie die is neergezet. De applicatie ziet er goed uit, en heeft veel nieuwe functionaliteit toegevoegd aan TJIP Processmanager. In de applicatie zitten een aantal onderdelen waar ik erg tevreden mee ben, en ook een aantal onderdelen waar ik minder tevreden mee ben.

De applicatie is opgezet met Prism en MEF. Dit beschouw ik zelf als een technische hoogstand in de applicatie. De applicatie is hiermee namelijk modulair geworden. Op dit moment zijn er nog geen modules beschikbaar, en is er ook nog geen moeite genomen om DirectoryDiscovery in te schakelen. Met DirectoryDiscovery is het mogelijk om een map uit te lezen, en alle modules die in die map staan te initialiseren. Wanneer dit is ingeschakeld, hoeft er aan de applicatie in principe niks verandert te worden om van een module gebruik te kunnen maken. Door het gebruik van Prism/MEF is de applicatie ook erg loosely coupled geworden, wat de onderhoudbaarheid van de applicatie ten goede komt.

Ik ben ook zeer tevreden over de manier waarop MVVM werkt. Het pattern werkt prettig, en is naar mijn mening ook goed geïmplementeerd in de applicatie. Er is in geen enkel scherm afgeweken van dit pattern. De code-behind is enkel gebruikt om aan te geven welk view-model hoort bij een bepaalde view. Alle functionele code is dus geplaatst in viewmodels. Ik ben hier tevreden over omdat het grafische gedeelte op deze manier dus vrijwel volledig is gescheiden van het functionele gedeelte. Wanneer een grafisch designer enkele schermen zou willen bewerken, hoeft hij dus geen aanpassingen te doen aan de functionele code.

Een punt van verbetering is het uiterlijk op sommige plaatsen in de applicatie. Ik ben zelf geen visueel designer, en dat is op sommige punten ook wel terug te zien in de applicatie. Een goed voorbeeld hiervan is het scherm om e-mails te versturen. Als inspiratie heb ik hier gekeken naar Microsoft Outlook 2010. Ik heb geprobeerd het scherm om e-mails te versturen na te bouwen. De velden staan op de juiste plek, het kleurthema is echter zeer lelijk. Ik heb aan het kleurthema ook geen verdere aandacht besteed. Dit is gekomen omdat er toch een bepaalde tijdsdruk achter het project stond, en de kleuren in schermen daardoor minder prioriteit hadden.



Waar ik verder erg tevreden over ben is het dashboard scherm. In dit scherm is in één keer duidelijk wat voor problemen zich voordoen bij de berichtenverwerking, of dat alles goed gaat. Dit is volledig conform de wensen van de opdrachtgever.  
Buiten dat het scherm functioneel juist werkt, ben ik ook trots op de achterliggende code. De code is op nette wijze geschreven en naar mijn mening zijn de juiste ontwerpkeuzes gemaakt. Door deze ontwerpkeuzes is het ook heel eenvoudig geworden om nieuwe items voor het dashboard te bouwen.   
Ook vind ik het dashboard er grafisch goed uit zien. De items op het dashboard hebben hun eigen “scherm” gekregen waardoor de items duidelijk te onderscheiden zijn.

Er is nog één ander onderdeel in de applicatie waar ik gemengde gevoelens over heb: de ConnectionHelper. Deze klasse werkt op dit moment zoals gewenst. Wel zitten er in de huidige vorm een aantal problemen. Zo is de ConnectionHelper op dit moment niet threadsafe.   
Het is mogelijk om meerdere instanties van de singleton te verkrijgen. Wanneer er meerdere instanties van de Singleton actief zijn, en er slechts op één instantie een wijziging plaatsvind op het moment dat de verbinding met Processmanager wegvalt, kunnen er problemen optreden. Het zou beter zijn als de class niet static was, en als dependancy meegegeven zou worden door MEF. Bij MEF valt aan te geven dat er enkel één instantie mag bestaan van een class.

## Procesevaluatie

Over het algemeen ben ik tevreden over het proces. De ontwikkelketen zoals beschreven in het plan van aanpak is goed doorlopen. Er zijn echter wel nog punten van kritiek

Een punt van het proces waar ik niet geheel tevreden mee ben is de manier waarop er met het functioneel ontwerp is omgegaan. Een van de hoofdredenen voor het schrijven van een functioneel ontwerp was om in een vroeg stadium fouten te voorkomen. Het kan vrij snel gebeuren dat ik als ontwikkelaar een requirement anders interpreteer dan dat deze door de opdrachtgever bedoeld is. Dergelijke zaken had ik willen voorkomen door het FO te laten reviewen door de opdrachtgever. Helaas had de opdrachtgever hier te weinig tijd voor.   
Ook is mijn keuze om het functioneel ontwerp enkel te laten reviewen door de opdrachtgever niet geheel juist geweest. Ik kwam hier achter op het moment dat er een demo was voor het architectenteam. Er was kritiek op de positionering van controls in een scherm. Wanneer ik het functioneel door meerdere mensen had laten reviewen was dit punt van kritiek niet aanwezig geweest. Tevens had dit er ook voor gezorgd dat er meer feedback is op het document (door meer mensen is de kans op feedback groter).  
In mijn geval heb ik me gelukkig goed in kunnen leven in de opdrachtgever. Ik heb regelmatig contact met hem gehad waardoor ik zijn gedachtegang goed kon begrijpen en veel heb kunnen doorvragen om functionaliteit helder te krijgen. De enige dure oplossing die ik heb moeten uitvoeren had te maken met de schermindeling.

Een positief punt is geweest mijn keuze voor scrum. Het project was niet volledig uit gespecificeerd. Een van de voordelen van scrum is dat je gedurende het traject requirements kan inzamelen, en deze inplannen voor een latere sprint. Dit was erg gemakkelijk na de gesprekken met een klant en het geven van een demo. Uit beide meetings zijn diverse requirements naar voren gekomen, welke deels in een latere sprint zijn ingepland.  
Dat dit is gebeurd, sluit volledig aan bij mijn verwachting. Mensen kunnen niet in één keer alle requirements bedenken, en gedachten over requirements kunnen veranderen gedurende het ontwikkeltraject.

Een ander punt van kritiek heeft te maken met het testen en schrijven van het functioneel ontwerp. Ik heb zowel het functioneel ontwerp geschreven als getest. Dit is een risico. Testen is onder andere bedoeld om te controleren of het gene wat in het functioneel ontwerp staat ook daadwerkelijk gebouwd is, en werkt zoals beschreven.   
De kans is aanwezig dat je bij het schrijven van het functioneel ontwerp, het ontwikkelen en het testen dezelfde denkfout maakt. Deze denkfout komt echter pas aan bod op het moment dat de opdrachtgever een acceptatietest uitvoert. Op dat moment is de fout erg duur geworden.  
In deze afstudeeropdracht had ik dit probleem helaas niet anders op kunnen lossen. Vanuit de school is er de wens dat het project individueel uitgevoerd wordt. Aan deze wens heb ik dan ook voldaan.   
Ondanks dat dit project individueel uitgevoerd moest worden heb ik wel geprobeerd om de schade van dergelijke denkfouten zo beperkt mogelijk te houden. Ik heb dit gedaan door veel met de opdrachtgever te overleggen, en gezamenlijk naar de gebouwde software te kijken. Hieruit kwam regelmatig feedback, en waren er kleine wijzigingen. Doordat er veel van dergelijke meetings waren, is de prijs van eventuele denkfouten beperkt gebleven.

Waar ik ook erg over te spreken ben is het programmeren. Buiten een aantal projecten op school was dit het eerste grote project waar ik iets met .NET moest doen. Ik had verwacht dat ik hierdoor wel tegen een aantal problemen zou aanlopen. Toen ik hoorde dat er gebruik gemaakt moest worden van WPF had ik helemaal verwacht dat ik wel tegen problemen aan zou lopen, dit omdat XAML een opmaaktaal is die ik niet ken.   
Uiteindelijk bleek dit heel erg mee te vallen. Uiteraard heb ik zo nu en dan wel is wat moeten opzoeken op het internet. Echt grote problemen zijn er niet aan bod geweest. Ondanks dat er geen grote problemen zijn geweest, heb ik wel veel geleerd over WPF, WCF en Prism. Dit is kennis die ik verwacht in de toekomst zeker nog te gaan gebruiken!

Een laatste punt wat ik een volgend project beter zou aanpakken zou de planning zijn. Omdat ik dit project allen was, heb ik er voor gekozen om de planning redelijk globaal te maken. De planning gaf enkel weer aan welke functionaliteit gewerkt moest worden, en hoeveel tijd daarvoor ingepland was. Er was niet gepland waar tijd binnen het bouwen van deze functionaliteit aan besteed moest worden. Doordat de planning niet gedetailleerd was, was het bij grote items op de backlog niet mogelijk om op ieder willekeurig moment nauwkeurig te bepalen of ik op schema liep. Ook heeft dit geleid tot burncharts die wellicht een onjuist beeld geven.  
Als voorbeeld wil ik graag de burnchart van sprint één gebruiken (eerder opgenomen in dit verslag). De eerste acht dagen stonden daar gepland om het framework op te zetten. De eerste dag ben ik grotendeels bezig geweest met inlezen in de technieken, en heb ik zo goed als niks aan het framework gedaan. Dergelijke vertragingen zijn door deze manier van plannen niet zichtbaar geworden.

## Evaluatie van de verrichtte competenties

### Uitvoeren analyse door definitie van requirements

Ik heb een functioneel ontwerp gebouwd. In dit functioneel ontwerp zijn requirements opgenomen. Ik denk dat ik door het bouwen van dit functioneel ontwerp heb voldaan aan deze competentie. Het functioneel ontwerp geeft een goed inzicht in het gene er gebouwd is. Door schermen op te nemen in het functioneel ontwerp valt er goed te toetsen of de gegevens compleet zijn, het geeft namelijk een goed beeld welke velden voor een bepaald object er opgenomen zijn. Tevens zorgen de schermen er ook voor dat de visuele requirements opgenomen zijn.

### Ontwerpen systeemdeel

Er is een managementconsole uitgedacht. Hiervoor zijn diverse ontwerpbeslissingen genomen. Hieronder vallen de keuzes voor MEF en Prism, maar ook keuzes om bepaalde taken op de achtergrond uit te voeren. Met de keuze voor MEF en Prism komt er ook dependancy injection bij kijken. Dit zijn complexe onderdelen die er voor zorgen dat de applicatie uitbreidbaar is. Hier komt de competentie “Ontwerpen systeemdeel” zeker bij kijken.

### Bouwen applicatie

Ik heb een management-console gebouwd die uitbreidbaar is en gebruik maakt van WCF. Door het ontwikkelen van deze managementconsole denk ik voldaan te hebben aan de competentie “Bouwen applicatie.”

### Initiëren en plannen van het testproces

Om het testen in goede banen te leiden, en vast te stellen wat er getest moet worden, heb ik een testplan opgesteld. Dit testplan geeft richting aan het testproces. Verder heb ik de definitie van “done” bepaald voor een backlog-item. Hieruit komt naar voren dat iets pas af is, wanneer het ook is getest. Ik denk dat ik hiermee heb voldaan aan de competentie “Initiëren en plannen van het testproces.”

### Uitvoeren en rapporteren van het testproces

Tijdens de bouw van de applicatie, maar ook tijdens het vaststellen van de kwaliteit van het prototype zijn er diverse tests ontwikkeld en uitgevoerd. Deze tests hebben bevindingen opgeleverd. De rapportage van deze tests is terug te vinden in een testrapport. Ik denk dat ik hiermee heb voldaan aan de competentie “Uitvoeren en rapporteren van het testproces.”

# Bijlage A – Plan van Aanpak

## Versiebeheer

|  |  |  |  |
| --- | --- | --- | --- |
| **Datum** | **Naam** | **Wijziging** | **Versie** |
| 7-2-2012 | Jeroen Niesen | Nieuw | 0.1 |
| 14-2-2012 | Jeroen Niesen | Versiebeheer toegevoegd, OTAP informatie uitgebreid, Sprint informatie uitgebreid | 0.2 |
| 21-2-2012 | Jeroen Niesen | Fasering toegevoegd, Spelling verbeterd | 0.3 |

## Opdracht

In het verleden is er een framework ontwikkeld voor berichtenverwerking, genaamd TJIP Interface Agent. Dit framework is aan vervanging toe, waarbij een aantal nieuwe wensen en eisen moeten worden meegenomen. In het kader van dat traject is een intern project gestart en een werkend prototype gebouwd. De nieuwe versie moet voorzien in de volgende features:

* Het draaien van jobs op de achtergrond (multi-threading).
* Het starten van jobs op geconfigureerde tijden (scheduling).
* Verwerking van berichten volgens een store and forward principe (message queueuing).
* In het verlengde van berichtenverwerking het uitvoeren van communicatietaken (email, SMS. Fax, etc.) op basis van batches.
* Ondersteuning voor allerlei koppelingen, zoals folders, IMAP, POP3, FTP en message queues.
* Het verzamelen van status- en management informatie over jobs en berichten.
* Een management console voor beheer en rapportage.

### Doelstelling

Er is nu een prototype van de TJIP Processmanager beschikbaar. Dit prototype zal moeten worden doorontwikkeld tot een volwaardig stuk software, welke voorziet in eerder genoemde eisen. De component moet gebruikt kunnen binnen andere projecten.

### Deliverables

1. Plan van aanpak (dit document)
2. Ontwerpdocumentatie
   1. Functioneel ontwerp
   2. Installatiehandleiding
3. Testdocumentatie
   1. Testplan
   2. Testscripts
4. TJIP Processmanager (sources)

### Projectactiviteiten

Binnen dit project zullen de volgende activiteiten worden uitgevoerd:

1. Documentatie
   1. Documenteren van het al bestaande prototype
      1. UML documentatie
      2. Installatiehandleiding
   2. Functioneel ontwerp
   3. Technisch ontwerp
2. Testen
   1. Testen huidige prototype
      1. Loadtest WCF Service
      2. Loadtest berichtenverkeer in cyclus
   2. Unittests
3. Ontwikkelen
   1. Management-tool
      1. Ontwerpen schermen
      2. Ontwikkelen schermen
      3. Uitbouwen WCF Service tbh van management-tool
   2. (na overleg/informeel onderzoek) MEF implementatie voor jobs
   3. (na overleg/informeel onderzoek) Wijzigen configuratie naar database
   4. (na overleg/informeel onderzoek) Uitbouwen beheertool met configuratiemogelijkheden en job-planning
   5. Toevoegen connectoren

## Aanpak

### Fasering

Dit project zal worden ingedeeld in twee fases. De eerste fase zal zich richten op het vaststellen van de kwaliteit van het prototype. In de tweede fase zal het prototype worden voorzien van nieuwe functionaliteit.

|  |
| --- |
| **Fase 1 Activiteiten** |
| 1. Vaststellen kwaliteit van het prototype middels ketentest en WCF-test. |
| 1. Verbeteren van fouten in het prototype. |
| **Fase 2 Activiteiten** |
| 1. Verzamelen van de requirements. |
| 1. Bouwen van een functioneel ontwerp voor de nieuwe functionaliteit. |
| 1. Technisch ontwerp maken voor nieuw te bouwen functionaliteit (indien nuttig). |
| 1. Bouwen van de software op basis van het functioneel ontwerp en eventueel technisch ontwerp. |
| 1. Testen van de gebouwde software aan hand van het functioneel ontwerp. |

Fase 1 zal worden afgesloten met een testrapportage. Deze testrapportage geeft inzicht in de volgende punten:

* Uitgevoerde tests
* Bevindingen
* Opgeloste bevindingen

### Ontwikkelmethode

De gekozen ontwikkelmethode is enkel van toepassing op de tweede fase van dit project. Dit omdat in de eerste fase niet ontwikkeld wordt.

Omdat een deel van de requirements nog niet volledig helder zijn, of nog ter discussie staan, wordt er gebruik gemaakt van een iteratieve ontwikkelmethode. Als ontwikkelmethode zal er gebruikt worden gemaakt van Agile scrum. Redenen om gebruik te maken van scrum zijn:

* Er wordt gewerkt met burn-charts. Via deze diagrammen is goed en duidelijk te zien of het project op tijd klaar zal zijn.
* Door daily standups is het duidelijk waar aan gewerkt wordt, en kunnen (blokkerende) problemen snel worden verholpen.

#### Sprints

Er zal gewerkt worden met sprints. Verderop in dit document is een sprint-planning te vinden. In de backlog zal een waarde worden toegekend aan ieder item. Deze waarde zal staan voor het aantal uren werk van het betreffende item.

Per sprint zal er aan de volgende onderdelen worden gewerkt:

* Functioneel ontwerp (FO)
* Technisch Ontwerp
* Ontwikkelen
* Testen



*Schematisch overzicht voor de werkwijze van een sprint*

#### Opstellen functioneel ontwerp

Er zal een functioneel ontwerp worden opgesteld. Het doel van dit ontwerp is voorkomen dat er dure fouten ontstaan. Een fout is het duurst wanneer er bij het onderdeel “testen” wordt geconstateerd dat bepaalde functionaliteit anders bedoeld is. Het functioneel ontwerp wordt gevalideerd door de opdrachtgever. Hierdoor kan het functioneel ontwerp worden gebruikt als testbasis.

#### Opstellen technisch ontwerp

Het technisch ontwerp is een middel om het ontwikkelen eenvoudiger te maken. Het voorkomt namelijk dat er tijdens het ontwikkelen stukken code gerefactord moeten worden omdat er een denkfout is geweest. Tijdens het opstellen van het technisch ontwerp wordt globaal uitgedacht hoe functionaliteit moet worden geïmplementeerd.

#### Ontwikkelen

Tijdens het ontwikkelen wordt software gemaakt op basis van het functioneel ontwerp en de technische documentatie

#### Testen

Tijdens het testen wordt onder andere gecontroleerd of er gebouwd wordt wat in het functioneel ontwerp staat. Het doel hiervan is om de kwaliteit van de software op een zo hoog mogelijk niveau te houden.

#### Definitie “klaar” voor een backlog-item

Een item op de backlog zal pas de status “klaar” krijgen op het moment dat de volgende activiteiten uitgevoerd zijn:

* Opstellen functioneel ontwerp
* Opstellen technisch ontwerp
* Ontwikkelen
* Testen
  + Oplossen bugs

## Ontwikkelstraat

Er zal worden gewerkt met een uitgebreide vorm van OTAP. Buiten de reguliere ontwikkel, test, acceptatie en productieomgeving zal er ook een ontwikkeltestomgeving zijn. Deze ontwikkeltestomgeving is bedoeld voor het debuggen van operaties welke veel tijdbeslag innemen. De ontwikkelomgeving blijft hierdoor beschikbaar om te ontwikkelen.

1. Ontwikkelomgeving
2. Ontwikkeltestomgeving
3. Testomgeving
4. Acceptatieomgeving
5. Productieomgeving

Dit project zal in gaan op punten één t/m vier. De te ontwikkelen applicatie is een intern component wat gebruikt zal worden in andere projecten van TJIP. Het in productie nemen van de applicatie zal gedaan worden door het projectteam dat het component wil integreren.



*Overzicht omgevingen: OTAP met een uitbreiding van de OntwikkelTest omgeving.*

### Microsoft Team Foundation Server

Het gehele ontwikkeltraject zal worden ondersteund door Microsoft Team Foundation Server (TFS). TFS zal op de volgende punten ondersteuning bieden:

* Beheren sprints en backlog
* Weergeven burncharts
* Bewaken van de kwaliteit van programeercode
* Versiebeheer

### Te gebruiken technieken

**Microsoft .NET C#**Er zal geprogrammeerd worden in C#. In het huidige prototype wordt gebruik gemaakt van de volgende frameworks:

* Quartz.NET (job scheduler)
* WCF Services (technologieën voor het gebruik van netwerksystemen)
* LINQ (datamanipulatietaal)

Bovenstaande lijst kan gedurende het project groeien, dit omdat er zaken zijn waar nog over beslist moet worden.

**UML**Ontwerpdocumentatie zal worden gemaakt middels UML. UML is sinds 1997 een standaard taal om object georiënteerde analyse en ontwerpen voor informatiesystemen te maken.

## Kwaliteit

De kwaliteit van het op te leveren product zal worden gewaarborgd op de volgende manieren:

* Er wordt een functioneel ontwerp opgesteld
  + Het functioneel ontwerp zal worden gevalideerd door de opdrachtgever.
* Er wordt gebruik gemaakt van een OTAP ontwikkelstraat.
* Er zal een degelijk testplan worden opgesteld. Er wordt getest op basis van een functioneel ontwerp.
* Code wordt bij iedere “check-in” gecontroleerd op complexiteit (code analysis en code matrics)

# Bijlage B – Testplan

## Versiebeheer

|  |  |  |  |
| --- | --- | --- | --- |
| **Versie** | **Datum** | **Naam** | **Wijziging** |
| 0.1 | 13-2-2012 | Jeroen Niesen | Nieuw |
| 0.2 | 15-2-2012 | Jeroen Niesen | Diverse wijzigingen aan hand van feedback |

## Inleiding

Het prototype van TJIP Processmanager wordt uitgebouwd tot een volwassen stuk software. De software die wordt ontwikkeld zal worden getest. Dit document geeft kaders en richting aan het testen van TJIP Processmanager.

## Documenten

### Referenties

* Plan van aanpak

### Testbasis

* Functioneel ontwerp
* Technische documentatie

## Testobjecten

Testobjecten zijn onderdelen die onderwerp van test zullen zijn. Globaal komen er (deel)applicaties beschikbaar welke samen TJIP Processmanager vormen. Deze applicaties zullen zijn:

* TJIP Processmanager Service
* TJIP Processmanager Management Console

## Testonderwerpen

* Bestaand prototype
  + Connectors
    - Imap
    - POP3
    - Exchange
    - Folders
    - GoSecure
    - Message Queue
  + WCF Service
* Nieuwe functionaliteit
  + Management console
    - Viewmodels
  + Connectors
  + Configuratiemogelijkheden job-planning

## Testbenadering

De kwaliteit van de software wordt in de eerste plaats bepaald door de geboden functionaliteit, maar ook de hoedanigheid van de implementatie speelt een belangrijke rol.

In onderstaande matrix is het relatieve belang van de geselecteerde kwaliteitsattributen weergegeven. Dit belang heeft betrekking op het te lopen risico. Het relatieve belang geeft dus weer op welke wijze de testinspanning verdeeld moet worden.

|  |  |  |
| --- | --- | --- |
| **Kwaliteitsattribuut** | **Relatief Belang** | **Toelichting** |
| Betrouwbaarheid | +++ | Bedrijfszekerheid, Foutbestendigheid |
| Bruikbaarheid |  |  |
| Efficieentie | +++ | Middelenbeslag, Tijdbeslag |
| Effectiviteit | + | Juistheid |
| Onderhoudbaarheid | + | Analyseerbaarheid, Wijzigbaarheid, Stabiliteit |
| Portabiliteit | + | Installeerbaarheid |

### Betrouwbaarheid

|  |  |  |
| --- | --- | --- |
| **Eigenschap** | **Toelichting** | **Maatregel** |
| Bedrijfszekerheid | De mate waarin het systeem last heeft van uitval ten gevolge van softwarefouten. |  |
| Foutbestendigheid | De mate waarin het systeem een bepaald prestatieniveau kan handhaven. | Loadtest |

### Efficieentie

|  |  |  |
| --- | --- | --- |
| **Eigenschap** | **Toelichting** | **Maatregel** |
| Middelenbeslag | De mate waarin het system een beroep doet op beschikbare operationele middelen (network, disk, geheugen). | Meten footprint |
| Tijdbeslag | De mate waarin het systeem tijd nodig heeft om bepaalde invoer te verwerken. | Loadtest |

### Effectiviteit

|  |  |  |
| --- | --- | --- |
| **Eigenschap** | **Toelichting** | **Maatregel** |
| Bedrijfszekerheid | De mate waarin het systeem de juiste afgesproken resultaten levert. | Unit-test, systeem-test |

### Onderhoudbaarheid

|  |  |  |
| --- | --- | --- |
| **Eigenschap** | **Toelichting** | **Maatregel** |
| Analyseerbaarheid | Het gemak waarmee het systeem tekortkomingen of foutoorzaken kunnen worden opgespoord. | Error-logging |
| Wijzigbaarheid | Het gemak waarmee het systeem gewijzigd kan worden om fouten te herstellen of om het systeem aan te passen aan veranderde omstandigheden. | Code-metrics, Code-analysis |
| Stabiliteit | De mate waarin het systeem bij onverwacht gedrag fouten gaat vertonen | Unit-tests |

### Portabiliteit

|  |  |  |
| --- | --- | --- |
| **Eigenschap** | **Toelichting** | **Maatregel** |
| Installeerbaarheid | Het gemak waarmee een systeem n de bedoelde omgeving geïnstalleerd kan worden | Installatie-test |

## Done criteria

Done criteria leggen vast wanneer een stuk software klaar is. Er zijn done criteria voor het ontwikkelen en voor het testen. In dit document wordt enkel in gegaan op de done criteria voor het testen.

Done criteria testen:

* Testcase is ontworpen
* Testcase is ingechecked
* Testcase is uitgevoerd
* Er zijn geen bekende fouten met een hoge prioriteit
* Alle bevinden met een hoge prioriteit zijn verholpen en een her-test uitgevoerd.

## Testproducten

De documentatie welke resulteert uit dit testplan zal bestaan uit:

* **Testplan**  
  *Dit document en eventuele vorige versies*
* **Bevindingen**  
  *De resultaten van de testen zoals die worden gerapporteerd*
* **Rapportage**  
  *De voortgangs- en kwaliteitsrapportages*

## Eisen omgeving

### Testomgeving

Er worden twee testomgevingen ingericht, een ontwikkeltest omgeving (O) en een systeemtest omgeving (s). De applicatie bestaat uit een presentatielaag, een business logica laag en een database laag. De onderdelen van de presentatielaag staan in een share op een fileserver. Onderdelen van de business-logica laag en database logica-laag staan geinstalleerd op een applicatieserver. De applicatieserver zal ook werken als file-server.



### Ontwikkeltest omgeving

**Geinstalleerde software**

* Windows Server 2003 Standaard Edition, SP2
* Visual Studio 2010 Premium Edition
  + Microsoft .Net Framework 4.0
* SQL Server 2008 Express
* SQL Server 2008 Management Studio

**Inhoud**

* Ingecheckte ontwikkelversie

**Doel**

* Wordt gebruikt voor het draaien van testen met een groot tijdbeslag.

### Systeemtest omgeving

**Geinstalleerde software**

* Windows Server 2003 Standaard Edition, SP2
* Microsoft .Net Framework 4.0
* SQL Server 2008

**Inhoud**

* Deployed versie

**Doel**

* Wordt gebruikt voor het uitvoeren van systeemtesten.

## Testtools

Bevindigenbeheer vind plaats in TFS (Team Foundation Server). Bevindingen krijgen een ernst (serverity) en een prioriteit (priority) toegewezen.   
De ernst van een bevingding geeft de mate van verstoring van productie aan. De ernst wordt vastgesteld aan hand van taxonomie. Over het algemeen veranderd de ernst van een bevinding niet.

### Testcase beheer

Voor het beheer van test-cases zal gebruik worden gemaakt van Microsoft Testmanager. Testmanager ondersteund het testen op de volgende punten:

* Vastleggen van testgevallen (logisch en fysiek)
* Tracebility naar requirements
* Opstellen test (executie) plannen

# Bijlage C – Functioneel Ontwerp

## Inleiding

Dit document is het functioneel ontwerp voor TJIP Processmanager. Doel van dit document is nieuwe functionaliteit vast leggen. Door de functionaliteit van de applicatie vast te leggen in dit document, kan worden voorkomen dat er later “dure reparaties” uitgevoerd moeten worden. Ook vormt dit document de basis bij het systeemtesten.

Dit document zal met enige regelmaat worden aangepast, het functioneel ontwerp wordt per iteratie uitgebreid. Er is gekozen voor deze aanpak omdat het in praktijk lastig blijkt te zijn om in één maal alle functionaliteit van een applicatie te bedenken en te beschrijven.

Tijdens het opstellen van dit document was er al een prototype van TJIP Processmanager beschikbaar. Er is afgesproken dat de functionaliteit van dit prototype juist is, en niet zal worden opgenomen in dit document. Indien nodig, zal functionaliteit van het prototype worden beschreven om vaagheden in dit document te voorkomen.

### Leeswijzer

Het eerste hoofdstuk zal beschrijven welke actoren de applicatie zullen gebruiken. Deze actoren zullen later terugkomen in use-cases. De functionaliteit van deze applicatie zal worden beschreven in use-cases.  
Nadat de actoren bekend zijn, zal er een globaal beeld worden geschetst hoe TJIP Processmanager zal werken. Dit hoofdstuk is bedoeld om een achtergrond op te bouwen, welke nodig is om dit verdere document te begrijpen.   
Verder in dit document zal worden ingegaan op de functionaliteit van TJIP Processmanager. Door gebruik te maken van een visueel design en use-cases moet duidelijk worden hoe de applicatie voor de betreffende functionaliteit zal werken.

## Opzet

TJIP Processmanager is een applicatie welke bedoelt is om berichtenverkeer te laten plaatsvinden. Het is de bedoeling dat TJIP Processmanager zal intergreren in andere projecten welke op dit moment lopen bij TJIP. Dit hoofdstuk omschrijft uit welke onderdelen TJIP Processmanager bestaat.

### Windows Service

De basis van TJIP Processmanager wordt gevormd door een Windows-Service. In deze Windows-service vind het berichtenverkeer plaats. Dit deel van de applicatie maakt gebruik van een Microsoft SQL Server, waar alle gegevens van het berichtenverkeer in worden opgeslagen.   
TJIP Processmanager werkt volgens een store-forward princiepe. Dit houd in dat ieder bericht eerst wordt opgeslagen, voordat het verder wordt gestuurd. Het opslaan van deze berichten gebeurt in de Micosoft SQL Server.

### WCF Service

Bovenop de Windows-service is een WCF-service gebouwd. Deze WCF-service maakt het mogelijk om TJIP Processmanager te intergreren in andere projecten. Via WCF is het mogelijk om rapportages uit TJIP Processmanager te halen, en is het mogelijk om TJIP Processmanager te beheren.

### Management Console

De Managementconsole maakt gebruik van de eerder beschreven WCF-service. Het is met de managementconsole mogelijk TJIP Processmanager te beheren, en rapportages uit te lezen.

## Management-Console Algemeen

### Consistentie in gebruik

Het is belangrijk dat de applicatie consistent is. Hiermee valt te denken aan de termen die worden gebruikt, de taal, maar de positie waarop knoppen en elementen geplaatst kunnen worden.

#### Voertaal

De taal welke gebruikt zal worden in TJIP Processmanager is Engels. Dit omdat TJIP Processmanager mogelijk ook bij internationale klanten gebruikt kan worden. 

#### Header region

De header region zal gebruikt worden om te navigeren door hoofdcategorieën binnen de applicatie. Deze region zal in principe voor alle views dezelfde inhoud hebben.

#### Body region

In de body region zal content worden geplaatst. De content welke weergegeven moet worden is selecteerbaar in de navigatie (header region).

## Management-Console Dashboard

### Visueel

#### Regions

Het dashboard is één view met daarin een tile-view. Aan deze tile-view kunnen losse views worden toegevoegd. Deze losse views zullen de rapportages worden.

#### Schermen

De volledige body-region wordt gebruikt voor het tonen van rapportages. Rapportages kunnen worden getoond in twee colomen.



### Rapportages

#### Errors & Warnings

Deze rapportage toont een eenvoudige lijst met daarin alle warnings en errors. Deze errors en warnings zijn afkomstig uit de processing-log. Belangrijk is dat errors en warnings gegroepeerd worden weergegeven, zodat alle errors en warnings met betrekking tot één beicht, bij elkaar staan.



#### Drill-down op errors/warnings

Hieronder volgt de use-case voor een drill-down op errors/warnings. Met een drill-down wordt bedoelt, dat er meer informatie te zien is met betrekking tot de oorzaak van de error/warning

|  |  |  |
| --- | --- | --- |
| **Stap** | **Actor** | **Omschrijving** |
| 1 | Gebruiker | Gebruiker navigeert naar de betreffende error/warning en dubbelklikt hierop |
| 2 | Systeem | Het systeem navigeert de gebruiker naar een scherm waar de error/warning gedetailleerd kan worden weergegeven. |

#### Currently running jobs

Deze rapportage toont een lijst met jobs welke op dit moment worden uitgevoerd. De lijst toont de titel van de job, en de datum/tijd waarop die gestart is. Er is geen interactie mogelijk met dit overzicht.



#### Problem Messages

Deze rapportage toont een weergave van berichten welke niet verwerkt konden worden. De informatie met betrekking tot de oorzaak van de fout kan worden achterhaald middels een drill-down.

#### Problem messages Drilldown

Hieronder volgt de use-case voor een drill-down op de execution-summary. Met een drill-down wordt bedoeld, dat er meer informatie te zien is met betrekking tot het geselecteerde item.

|  |  |  |
| --- | --- | --- |
| **Stap** | **Actor** | **Omschrijving** |
| 1 | Gebruiker | Gebruiker navigeert naar het betreffende bericht in de lijst, en dubbelklikt hierop |
| 2 | Systeem | Het systeem navigeert de gebruiker naar een scherm (Operation -> Processing Entities -> Detailed view) waar het item gedetailleerd kan worden weergegeven, en er acties met betrekking tot het item kunnen worden uitgevoerd |

## Management-Console Configuratie

De categorie configuratie zal gebruikt worden om de configuratie van TJIP Processmanager uit te lezen en eventueel te wijzigen

### Visueel

#### Regions



#### Navigation Region

Er zal gebruik worden gemaakt van een accordion om door de configuratie van TJIP Processmanager te navigeren. In de accordion is het mogelijk om elementen te selecteren. Voor het geselecteerde element zal vervolgens in de “content region” content worden weergegeven.

#### Content region

In de content-region zullen instellingen en informatie over de configuratie worden weergegeven.

#### Schermen



#### Configuratie navigatie

De navigatie van configuratie-items zal plaatsvinden in een outlookbar/accordion. Op deze manier zullen niet alle configuratie-items in één grote lijst worden getoond, maar worden er kleine lijsten getoond aan hand van categorieën waar deze configuratie-items in thuishoren.

De hoofdcategorieën zijn:

* Common settings
* Scheduler settings
* Job settings

De totale boomstructuur van configuratie-items zal zijn:

* Common settings
  + Database
    - Settings
    - Cleanup
  + SMTP
  + Fax
  + SMS
* Scheduler settings
  + Triggers
  + Scheduled Jobs
  + Calendars
* Jobs
  + Jobs

#### Configuratie-items

Configuratie-items zullen worden weergegeven in losse “tile-windows.” Op deze manier kunnen instellingen binnen configuratie-items worden weergegeven op categorie.

### Functioneel

#### Bekijken configuratie

Hieronder volgt de use-case voor het inzien van de configuratie voor een bepaald onderdeel. Omdat het inzien van de configuratie voor alle onderdelen het zelfde werkt, is er enkel één use-case geschreven welke voor alle items geld.

|  |  |  |
| --- | --- | --- |
| **Stap** | **Actor** | **Omschrijving** |
| 1 | Gebruiker | De actor navigeert naar het onderdeel “Configuratie” |
| 2 | Systeem | Het systeem navigeert de gebruiker naar de configuratie-sectie |
| 3 | Gebruiker | De gebruiker selecteerd de categorie waar het configuratie-item onder valt |
| 4 | Systeem | Het systeem toont een lijst met configuratie-items waarvan de configuratie ingezien kan worden |
| 5 | Gebruiker | De gebruiker selecteerd het item waarvan de configuratie ingezien moet worden. |
| 6 | Systeem | Het systeem toont de configuratie voor het betreffende configuratie item. |

## Management-Console Operation

### Visueel

### Regions

Het onderdeel operations zal bestaan uit één view, met daarin een tabcontrol. Aan deze tabcontrol kunnen nieuwe views worden toegevoegd.

### Schermen



Het scherm voor operationele doeleinden zal bestaan uit tabbladen. Deze keuze is gemaakt om het scherm waarop gewerkt wordt, zo groot mogelijk te houden. De volgende tabbladen zullen bestaan:

* Processing Log
* Messages
* Run Jobs
* Fire Triggers

#### Processing Log

Hieronder een weergave van het scherm “Processing Log” zoals deze geïmplementeerd zal worden in het onderdeel “Operations.” Het scherm groepeert meldingen automatisch op executie. Omdat het aantal meldingen hoog kan oplopen, zal paging worden toegepast.



#### Messages

Hieronder een weergave van het “Messages” scherm zoals dit geïmplementeerd zal worden binnen het onderdeel “Operations.”



Wanneer een bericht geselecteerd wordt, zullen de beknopte details onder het bericht worden weergegeven. Deze details zullen verschillen per berichttype. In de weergave van de beknopte details is een knop “Details” te vinden. Wanneer deze wordt ingedrukt, zullen de volledige details van een bericht worden getoond.

Van alle typen berichten zullen in ieder geval de volgende eigenschappen worden weergegeven:

* Send After
* Last prodiver response

**Details per berichttype**

***Incoming Email***

* From
* To
* CC
* BCC
* Subject

***Outgoing Email***

* From
* To
* CC
* BCC
* Subject

***Outgoing Fax***

* Recpipients
* Attachments

***Outgoing SMS***

* Orginator
* Recipients
* Message

***Outgoing Message***

* Source
* Outgoing Source

***Incoming Message***

* Source
* Consuming Job

#### Incoming e-mail



Bovenstaand scherm is een weergave van een inkomende e-mail.   
  
**Gedrag “Open Message”**Het bestand wordt in de temp-folder van de gebruiker weggeschreven als .eml bestand. Na het opslaan wordt het bestand gestart. De e-mail client zal starten en het e-mailbericht openen.

#### Outgoing e-mail



Bovenstaand scherm is een weergave van een uitgaande e-mail.   
  
**Gedrag “Open Message”**Het bestand wordt in de temp-folder van de gebruiker weggeschreven als .eml bestand. Na het opslaan wordt het bestand gestart. De e-mail client zal starten en het e-mailbericht openen.

#### Outgoing Sms



Bovenstaand scherm is een weergave van een uitgaande SMS.   
  
**Gedrag “Open Message”**Het bestand wordt in de temp-folder van de gebruiker weggeschreven als .txt bestand. Na het opslaan wordt het bestand gestart. Het tekstbestand zal worden geopend met de standaard tekst-editor.

#### Incoming message



Bovenstaand scherm is een weergave van een inkomende message.   
  
**Gedrag “Open Message”**Het bestand wordt in de temp-folder van de gebruiker weggeschreven als .txt bestand. Na het opslaan wordt het bestand gestart. Het tekstbestand zal worden geopend met de standaard tekst-editor.

#### Ougoing message



Bovenstaand scherm is een weergave van een uitgaande message.   
  
**Gedrag “Open Message”**Het bestand wordt in de temp-folder van de gebruiker weggeschreven als .txt bestand. Na het opslaan wordt het bestand gestart. Het tekstbestand zal worden geopend met de standaard tekst-editor.

#### Run Jobs



In bovenstaande view is het mogelijk om een job te selecteren, en daarna op de knop “Run Job” te klikken, welke de Job zal uitvoeren.

#### Fire Triggers



### Functioneel

#### Opnieuw verwerken van een bericht

Hieronder volgt de use-case hoe een bericht opnieuw kan worden verwerkt, nadat het verwerken eerder fout is gegaan (bijvoorbeeld door het falen van externe systemen).

|  |  |  |
| --- | --- | --- |
| **Stap** | **Actor** | **Omschrijving** |
| 1 | Gebruiker | Gebruiker navigeert naar het onderdeel “Operation.” |
| 2 | Systeem | Het systeem toont het hoofdscherm voor het onderdeel “operation”. Op dit hoofdscherm is het eerste tab “Processing problems” zichtbaar |
| 3 | Gebruiker | De gebruiker navigeert naar het bericht wat hij opnieuw wilt laten verwerken, en selecteert dit. |
| 4 | Systeem | Het systeem toont een uitgebreide weergave van het te verwerken bericht |
| 5 | Gebruiker | De gebruiker geef aan dat dit bericht opnieuw verwerkt moet worden |
| 6 | Systeem | Het systeem geeft visueel aan dat het bericht opnieuw verwerkt zal worden. |

#### Bekijken eigenschappen bericht

Hieronder volgt de use-case voor het bekijken van de eigenschappen van een bericht.

|  |  |  |
| --- | --- | --- |
| **Stap** | **Actor** | **Omschrijving** |
| 1 | Gebruiker | Gebruiker navigeert naar het onderdeel “Operation.” |
| 2 | Systeem | Het systeem toont het hoofdscherm voor het onderdeel “operation”. Op dit hoofdscherm is een tab “Messages” zichtbaar |
| 3 | Gebruiker | De gebruiker klikt op het tabblad “Messages” |
| 4 | Systeem | Het systeem toont een weergave van alle berichten in het systeem |
| 5 | Gebruiker | De gebruiker navigeert naar het betreffende bericht, en klikt hierop. |
| 6 | Systeem | Het systeem toont een uitgebreide weergave van het bericht. |
| 8 | Gebruiker | De gebruiker klikt op de knop “view details” om het volledige bericht te bekijken. |
| 9 | Systeem | Het systeem toont het bericht aan de gebruiker. |

#### Bericht openen

Hieronder volgt de use-case voor het openen van een bericht.

|  |  |  |
| --- | --- | --- |
| **Stap** | **Actor** | **Omschrijving** |
| 1 | Gebruiker | Gebruiker navigeert naar het onderdeel “Operation.” |
| 2 | Systeem | Het systeem toont het hoofdscherm voor het onderdeel “operation”. Op dit hoofdscherm is een tab “Messages” zichtbaar |
| 3 | Gebruiker | De gebruiker klikt op het tabblad “Messages” |
| 4 | Systeem | Het systeem toont een weergave van alle berichten in het systeem |
| 5 | Gebruiker | De gebruiker navigeert naar het betreffende bericht, en klikt hierop. |
| 6 | Systeem | Het systeem toont een uitgebreide weergave van het bericht. |
| 8 | Gebruiker | De gebruiker klikt op de knop “view details” om het volledige bericht te bekijken. |
| 9 | Systeem | Het systeem toont het bericht aan de gebruiker. |
| 10 | Gebruiker | De gebruiker klikt op de knop “Open message” |
| 11 | Systeem | Afhankelijk van het type bericht, zal deze worden opgeslagen als e-mail of tekstbestand. Na het opslaan zal het systeem het bericht met de standaard geïnstalleerde applicatie openen. |

#### Job runnen

Hieronder volgt de use-case voor het handmatig runnen van een job.

|  |  |  |
| --- | --- | --- |
| **Stap** | **Actor** | **Omschrijving** |
| 1 | Gebruiker | Gebruiker navigeert naar het onderdeel “Operation.” |
| 2 | Systeem | Het systeem toont het hoofdscherm voor het onderdeel “operation”. Op dit hoofdscherm is een tab “Messages” zichtbaar |
| 3 | Gebruiker | De gebruiker klikt op het tabblad “Run Jobs” |
| 4 | Systeem | Het systeem toont een weergave van alle jobs in het systeem |
| 5 | Gebruiker | De gebruiker navigeert naar de betreffende job, en klikt hierop. |
| 6 | Systeem | Het systeem markeert de job als geselecteerd |
| 8 | Gebruiker | De gebruiker klikt op de knop “run selected job” |
| 9 | Systeem | Het systeem voert de job uit |

#### Direct uitvoeren van een trigger

Hieronder volgt een use-case voor het direct starten van een trigger.

|  |  |  |
| --- | --- | --- |
| **Stap** | **Actor** | **Omschrijving** |
| 1 | Gebruiker | Gebruiker navigeert naar het onderdeel “Operation.” |
| 2 | Systeem | Het systeem toont het hoofdscherm voor het onderdeel “operation”. Op dit hoofdscherm is een tab “Run Jobs” zichtbaar |
| 3 | Gebruiker | De gebruiker klikt op het tabblad “Fire triggers” |
| 4 | Systeem | Het systeem toont een weergave van alle beschikbare triggers. |
| 5 | Gebruiker | De gebruiker navigeert naar de betreffende trigger, en klikt hierop. |
| 8 | Gebruiker | De gebruiker klikt op de knop “Fire selected trigger” |
| 9 | Systeem | Het systeem start de trigger. |

## Management-Console Tools

De categorie tools zal gebruikt worden om tools in te plaatsen. Tools zijn niet een primair onderdeel van de Management console, maar kunnen gebruikt worden voor test doeleinden en handmatige correcties. Tools worden weergegeven in de body region.

### Send E-mail

#### Visueel



#### Functioneel

**Basispad**

|  |  |  |
| --- | --- | --- |
| **Stap** | **Actor** | **Omschrijving** |
| 1 | Gebruiker | Navigeert via het menu tools naar het onderdeel “Send E-mail” |
| 2 | Systeem | Het systeem toont een weergave om e-mails te versturen |
| 3 | Gebruiker | Vul minimaal in het onderdeel “To” een e-mail adres in waar naar verstuurd mag worden. Voer eventueel gegevens in voor de onderdelen CC, BCC, From, Subject en bericht. Voeg eventueel een bijlage toe. |
| 4 | Systeem | Het systeem toont de ingevoerde gegevens. |
| 5 | Gebruiker | Klik op “Send” |
| 6 | Systeem | Het systeem verzendt de e-mail en sluit het venster |

**Alternatief pad**

|  |  |  |
| --- | --- | --- |
| **Stap** | **Actor** | **Omschrijving** |
| 6B | Systeem | Het systeem toont een foutmelding dat de bijlagen in het e-mailbericht groter zijn dan 25MB, en verwijdert de bijlagen uit de e-mail. |
| 7 | Gebruiker | De gebruiker kan eventueel nieuwe/andere bijlage toevoegen en/of het e-mailbericht wijzigen. En gaat verder met stap 5 van deze use-case. |

### Send SMS

#### Visueel



#### Functioneel

**Basispad**

|  |  |  |
| --- | --- | --- |
| **Stap** | **Actor** | **Omschrijving** |
| 1 | Gebruiker | Navigeert via het menu tools naar het onderdeel “Send SMS” |
| 2 | Systeem | Het systeem toont een weergave om SMSen te versturen |
| 3 | Gebruiker | Vul het onderdeel “Recipient”, “From” en “berichttekst” in. |
| 4 | Systeem | Het systeem toont de ingevoerde gegevens. |
| 5 | Gebruiker | Klik op “Send” |
| 6 | Systeem | Het systeem verzend de SMS en sluit het venster |

**Alternatief pad**

|  |  |  |
| --- | --- | --- |
| **Stap** | **Actor** | **Omschrijving** |
| 6B | Systeem | Het systeem toont een foutmelding dat de berichttekst te groot is. |
| 7 | Gebruiker | De gebruiker kan het bericht bewerken, en verder gaan bij stap 5 van deze use-case. |

### Send Fax

#### Visueel



#### Functioneel

**Basispad**

|  |  |  |
| --- | --- | --- |
| **Stap** | **Actor** | **Omschrijving** |
| 1 | Gebruiker | Navigeert via het menu tools naar het onderdeel “Send Fax” |
| 2 | Systeem | Het systeem toont een weergave om faxen te versturen |
| 3 | Gebruiker | Vul het onderdeel “Recipient”, en “berichttekst” in. Voeg eventueel enkele bijlages toe. |
| 4 | Systeem | Het systeem toont de ingevoerde gegevens. |
| 5 | Gebruiker | Klik op “Send” |
| 6 | Systeem | Het systeem verzendt de Fax en sluit het venster |

**Alternatief pad**

|  |  |  |
| --- | --- | --- |
| **Stap** | **Actor** | **Omschrijving** |
| 6B | Systeem | Het systeem toont een foutmelding dat de bijlagen in het e-mailbericht groter zijn dan 25MB, en verwijdert de bijlagen uit de e-mail. |
| 7 | Gebruiker | De gebruiker kan het bericht bewerken, en verder gaan bij stap 5 van deze use-case. |

# Bijlage D – Testscripts

## Testscripts

Zorg voor de testscript dat de library “Tjip.Processmanager.Performancetest” is ingesteld als library met Jobs. Als mailserver wordt er gebruik gemaakt van MercuryMail. Voor alle tests geld dat MercuryMail op de achtergrond actief is.

### Verbinding maken, Opnieuw verbinden

|  |  |  |
| --- | --- | --- |
| Stap | Actie | Verwacht resultaat |
| 1 | Start de Processmanager client | Er draait een processmanager client welke aangeeft dat deze probeert verbinding te maken met de server. |
| 2 | Start Processmanager | Er wordt een instantie van Processmanager gestart  Nadat Processmanager is opgestart, zal de client in de statusbalk weergeven dat deze verbonden is. |
| 3 | Sluit Processmanager | De client toont een scherm dat er verbinding geprobeerd wordt te maken met de Processmanager |
| 4 | Start Processmanager | Nadat Processmanager is opgestart, zal de client in de statusbalk weergeven dat deze verbonden is. |

### Uitvoeren Job

|  |  |  |
| --- | --- | --- |
| Stap | Actie | Verwacht resultaat |
| 1 | Start Processmanager Start een Processmanager Client (A)  Start een Processmanager Client (B) | Er draaien twee clients en een instantie van Tjip Processmanager |
| 2 | Ga in client A naar “Operations 🡪 Run Jobs”. | Een lijst met uit te voeren jobs wordt getoond. |
| 3 | Selecteer Job: “TimeSpandingJob3” | “TimeSpandingJob3” wordt gemarkeerd als geselecteerd |
| 4 | Klik op de knop “Run Job” | In client B moet op het dashboard verschijnen bij het onderdeel “Executing Jobs” dat “TimeSpandingJob3” aan het draaien is |

### Verzenden e-mail (minimale gegevens)

|  |  |  |
| --- | --- | --- |
| Stap | Actie | Verwacht resultaat |
| 1 | Start Processmanager Start een Processmanager Client | Er draait een instantie van Tjip Processmanager en een Processmanager client. |
| 2 | Ga in het menu naar het onderdeel “Tools” en klik op “Send e-mail” | Er wordt een invoerscherm getoond voor een e-mailbericht. |
| 3 | Vul in het onderdeel “To” in: test@localhost.net |  |
| 4 | Voer als berichttest “Test” in. |  |
| 5 | Ga naar “Operations 🡪 Messages”. | Er wordt een lijst met berichten weergegeven |
| 6 | Selecteer het eerste bericht | De uitgebreide details voor het bericht worden getoond. |
| 7 | Klik op de knop “Details” | De uitgebreide details van het bericht worden weergegeven. |
| 8 | Controleer of het from e-mailadres “ProcessManager@localhost” is |  |
| 9 | Controleer of het to e-mailadres “test@localhost.net” is |  |
| 10 | Klik op “Open Message” | Het systeem opent een e-mail client met het bericht erin. |
| 11 | Controleer of de berichttekst “Test” is. |  |

### Verzenden e-mail (alle gegevens)

|  |  |  |
| --- | --- | --- |
| Stap | Actie | Verwacht resultaat |
| 1 | Start Processmanager Start een Processmanager Client | Er draait een instantie van Tjip Processmanager en een Processmanager client. |
| 2 | Ga in het menu naar het onderdeel “Tools” en klik op “Send e-mail” | Er wordt een invoerscherm getoond voor een e-mailbericht. |
| 3 | Vul in het onderdeel “To” in: test@localhost.net |  |
| 4 | Vul in het onderdeel “CC” in: test2@localhost.net |  |
| 5 | Vul in het onderdeel “BCC” in: test3@localhost.net |  |
| 6 | Vul in het onderdeel “Subject” in: Testmessage |  |
| 7 | Vul in het onderdeel “From” in: tester@localhost.net |  |
| 8 | Voer als berichttest “Test” in. |  |
| 9 | Ga naar “Operations 🡪 Messages”. | Er wordt een lijst met berichten weergegeven |
| 10 | Selecteer het eerste bericht | De uitgebreide details voor het bericht worden getoond. |
| 11 | Klik op de knop “Details” | De uitgebreide details van het bericht worden weergegeven. |
| 12 | Controleer of het from e-mailadres “ProcessManager@localhost” is |  |
| 13 | Controleer of het to e-mailadres “test1@localhost.net” is |  |
| 14 | Controleer of het CC e-mailadres “test2@localhost.net” is |  |
| 15 | Controleer of het BCC e-mailadres “test3@localhost.net” is |  |
| 16 | Controleer of het subject: “Testmessage” is |  |
| 17 | Klik op “Open Message” | Het systeem opent een e-mail client met het bericht erin. |
| 18 | Controleer of de berichttekst “Test” is. |  |

### Verzenden e-mail (met bijlagen, regulier)

|  |  |  |
| --- | --- | --- |
| Stap | Actie | Verwacht resultaat |
| 1 | Start Processmanager Start een Processmanager Client | Er draait een instantie van Tjip Processmanager en een Processmanager client. |
| 2 | Ga in het menu naar het onderdeel “Tools” en klik op “Send e-mail” | Er wordt een invoerscherm getoond voor een e-mailbericht. |
| 3 | Vul in het onderdeel “To” in: test@localhost.net |  |
| 4 | Voer als berichttest “Test” in. |  |
| 5 | Voeg een bijlage van ongeveer 2MB toe aan de e-mail. |  |
| 6 | Klik op “Send” |  |
| 7 | Ga naar “Operations 🡪 Messages”. | Er wordt een lijst met berichten weergegeven |
| 8 | Selecteer het eerste bericht | De uitgebreide details voor het bericht worden getoond. |
| 9 | Klik op de knop “Details” | De uitgebreide details van het bericht worden weergegeven. |
| 10 | Controleer of het from e-mailadres “ProcessManager@localhost” is |  |
| 11 | Controleer of het to e-mailadres “test@localhost.net” is |  |
| 12 | Klik op “Open Message” | Het systeem opent een e-mail client met het bericht erin. |
| 13 | Controleer of het bestand als bijlage is toegevoegd aan de e-mail. |  |
| 14 | Controleer of de berichttekst “Test” is. |  |

### Verzenden e-mail (met bijlagen, te groot, 1 item)

In deze test zal worden gecontroleerd of het mogelijk is om een bijlage te versturen. Als test maken we gebruik van een

|  |  |  |
| --- | --- | --- |
| Stap | Actie | Verwacht resultaat |
| 1 | Start Processmanager Start een Processmanager Client | Er draait een instantie van Tjip Processmanager en een Processmanager client. |
| 2 | Ga in het menu naar het onderdeel “Tools” en klik op “Send e-mail” | Er wordt een invoerscherm getoond voor een e-mailbericht. |
| 3 | Vul in het onderdeel “To” in: test@localhost.net |  |
| 4 | Voer als berichttest “Test” in. |  |
| 5 | Voeg een bijlage van ongeveer 30MB toe aan de e-mail. |  |
| 6 | Klik op “Send” | Het systeem toont een melding dat de bijlagen te groot zijn. |
| 8 | Klik op “Ok” om de melding weg te klikken. | Het systeem leegt de lijst met bijlagen. |

### Verzenden e-mail (met bijlagen, te groot, meerdere items)

|  |  |  |
| --- | --- | --- |
| Stap | Actie | Verwacht resultaat |
| 1 | Start Processmanager Start een Processmanager Client | Er draait een instantie van Tjip Processmanager en een Processmanager client. |
| 2 | Ga in het menu naar het onderdeel “Tools” en klik op “Send e-mail” | Er wordt een invoerscherm getoond voor een e-mailbericht. |
| 3 | Vul in het onderdeel “To” in: test@localhost.net |  |
| 4 | Voer als berichttest “Test” in. |  |
| 5 | Voeg meerdere bijlagen toe, totdat de totale bestandsgrote groter is dan 25MB. |  |
| 6 | Klik op “Send” | Het systeem toont een melding dat de bijlagen te groot zijn. |
| 8 | Klik op “Ok” om de melding weg te klikken. | Het systeem leegt de lijst met bijlagen. |

### Verzenden fax (regulier)

Voor het verzenden van faxen, gaan we er vanuit dat “Tjip.Process\Sources\Tjip.Process.Service\bin\Debug\TestOut\Faxen” is ingesteld als dropfolder, en dat de dropfolder leeg is.

|  |  |  |
| --- | --- | --- |
| Stap | Actie | Verwacht resultaat |
| 1 | Start Processmanager Start een Processmanager Client | Er draait een instantie van Tjip Processmanager en een Processmanager client. |
| 2 | Start de scheduler door in Processmanager op “1” te drukken | Het systeem start de scheduler. |
| 3 | Ga in het menu naar het onderdeel “Tools” en klik op “Send fax” | Er wordt een invoerscherm getoond voor een fax. |
| 4 | Voer bij recipient “012345689” | Het systeem toont een ingevulde tekstbox |
| 5 | Voer als berichttekst “test123” in | Het systeem toont een ingevulde tekstbox |
| 6 | Klik op “Send” | Het systeem verzendt de fax |
| 7 | Navigeer naar de dropfolder (Tjip.Process\Sources\Tjip.Process.Service\bin\Debug\TestOut\Faxen) |  |
| 8 | Controleer of de fax aanwezig is, en open het bijhorende .txt bestand | Het systeem opent het txt bestand. |
| 9 | Controleer of in het bestand de tekst “test123” wordt weergegeven. |  |

### Verzenden fax (met bijlagen)

Voor het verzenden van faxen, gaan we er vanuit dat “Tjip.Process\Sources\Tjip.Process.Service\bin\Debug\TestOut\Faxen” is ingesteld als dropfolder, en dat de dropfolder leeg is.

Zorg voor deze tekst dat er een word-document beschikbaar is met als inhoud “faxTest123.”

|  |  |  |
| --- | --- | --- |
| Stap | Actie | Verwacht resultaat |
| 1 | Start Processmanager Start een Processmanager Client | Er draait een instantie van Tjip Processmanager en een Processmanager client. |
| 2 | Start de scheduler door in Processmanager op “1” te drukken | Het systeem start de scheduler. |
| 3 | Ga in het menu naar het onderdeel “Tools” en klik op “Send fax” | Er wordt een invoerscherm getoond voor een fax. |
| 4 | Voer bij recipient “012345689” | Het systeem toont een ingevulde tekstbox |
| 5 | Voer als berichttekst “test123” in | Het systeem toont een ingevulde tekstbox |
| 6 | Voeg als bijlage het worddocument toe | Het systeem voegt het document als bijlage toe aan de fax. |
| 7 | Klik op “Send” | Het systeem verzendt de fax |
| 8 | Navigeer naar de dropfolder (Tjip.Process\Sources\Tjip.Process.Service\bin\Debug\TestOut\Faxen) |  |
| 9 | Controleer of de fax aanwezig is, en open het bijhorende .txt bestand | Het systeem opent het txt bestand. |
| 10 | Controleer of in het bestand de tekst “test123” wordt weergegeven. |  |
| 11 | Sluit het bestand, en open het .docx bestand wat ook in de map wordt weergegeven. |  |
| 12 | Controleer of de inhoud van het bestand “faxTest123.“ is. |  |

### Verzenden SMS

|  |  |  |
| --- | --- | --- |
| Stap | Actie | Verwacht resultaat |
| 1 | Start Processmanager Start een Processmanager Client | Er draait een instantie van Tjip Processmanager en een Processmanager client. |
| 2 | Start de scheduler door in Processmanager op “1” te drukken | Het systeem start de scheduler. |
| 3 | Ga in het menu naar het onderdeel “Tools” en klik op “Send SMS” | Er wordt een invoerscherm getoond voor een sms. |
| 4 | Voer in het scherm “Recipient” het nummer van het ontvangende toestel in. | Het systeem toont het ingevoerde telefoonnummer |
| 5 | Voer als berichttekst “Test 123” in | Het systeem toont de ingevoerde berichttekst. |
| 6 | Klik op de knop “Send” | Het systeem verzend het bericht naar de recipient en sluit het scherm. |
| 7 | Controleer op het ontvangende toestel of de berichttekst “test 123” is. |  |

### Weergeven Configuratie (Database)

Zorg dat in de configuratie de database als volgt is geconfigureerd:

<Database server=".\SQLEXPRESS" databaseName="ProcessManager" databaseFilepath=".\LocalTestDatabase.mdf" allowCreate="true">

<AutomaticCleanup enabled="true" maximumItemCount="1000" maximumAgeInDays="100" processingIntervalInMinutes="45">

<ExecutionLog maximumAgeInDays="3" />

<OutgoingEmail enabled="false"/>

</AutomaticCleanup>

</Database>

|  |  |  |
| --- | --- | --- |
| Stap | Actie | Verwacht resultaat |
| 1 | Start Processmanager Start een Processmanager Client | Er draait een instantie van Tjip Processmanager en een Processmanager client. |
| 2 | Ga naar het onderdeel “Settings” | Het systeem toont de hoofdpagina van de configuratie. |
| 3 | Selecteer het onderdeel “Database” in de outlook bar. | Het systeem toont de configuratie voor de database. |
| 4 | Controleer dat het onderdeel “Server” de waarde “.\SQLEXPRESS” heeft |  |
| 5 | Controleer dat het onderdeel “Databse name” de waarde “ProcessManager” heeft |  |
| 6 | Controleer dat het onderdeel “Allow Create” de waarde “True” heeft |  |

### Weergeven Configuratie (SMTP)

Zorg dat in de configuratie de SMTP als volgt geconfigureerd is:

<SendMail smtpServer="localhost" processingIntervalInSeconds="5" batchSize="250"/>

|  |  |  |
| --- | --- | --- |
| Stap | Actie | Verwacht resultaat |
| 1 | Start Processmanager Start een Processmanager Client | Er draait een instantie van Tjip Processmanager en een Processmanager client. |
| 2 | Ga naar het onderdeel “Settings” | Het systeem toont de hoofdpagina van de configuratie. |
| 3 | Selecteer het onderdeel “SMTP” in de outlook bar. | Het systeem toont de configuratie voor de SMTP. |
| 4 | Controleer dat het onderdeel “SMTP Server” de waarde “localhost” heeft |  |
| 5 | Controleer dat het onderdeel “SMTP Port” de waarde “25” heeft |  |
| 6 | Controleer dat het onderdeel “Batch size” de waarde “250” heeft |  |
| 7 | Controleer dat het onderdeel “Processing interval” de waarde “5” heeft |  |

### Weergeven Configuratie (SMS)

Zorg dat in de configuratie de SMS als volgt geconfigureerd is:

<SendSms provider="Mollie" batchSize="5" username="" password="58DA195C3BC29989FBCAF96922C8E11A" />

|  |  |  |
| --- | --- | --- |
| Stap | Actie | Verwacht resultaat |
| 1 | Start Processmanager Start een Processmanager Client | Er draait een instantie van Tjip Processmanager en een Processmanager client. |
| 2 | Ga naar het onderdeel “Settings” | Het systeem toont de hoofdpagina van de configuratie. |
| 3 | Selecteer het onderdeel “SMTP” in de outlook bar. | Het systeem toont de configuratie voor de SMTP. |
| 4 | Controleer dat het onderdeel “SMTP Server” de waarde “localhost” heeft |  |
| 5 | Controleer dat het onderdeel “SMTP Port” de waarde “25” heeft |  |
| 6 | Controleer dat het onderdeel “Batch size” de waarde “250” heeft |  |
| 7 | Controleer dat het onderdeel “Processing interval” de waarde “5” heeft |  |

### Weergeven Configuratie (Fax)

Zorg dat de configuratie voor het verzenden van faxen als volgt is ingesteld:

<SendFax provider="FaxMaker" batchSize="7" faxMakerDropFolder=".\TestOut\Faxen" faxMakerSenderEmailAddress="eca.send.fax@tjip.com"/>

|  |  |  |
| --- | --- | --- |
| Stap | Actie | Verwacht resultaat |
| 1 | Start Processmanager Start een Processmanager Client | Er draait een instantie van Tjip Processmanager en een Processmanager client. |
| 2 | Ga naar het onderdeel “Settings” | Het systeem toont de hoofdpagina van de configuratie. |
| 3 | Selecteer het onderdeel “Fax” in de outlook bar. | Het systeem toont de configuratie voor de FAX. |
| 4 | Controleer dat het onderdeel “Provider” de waarde “FaxMaker” heeft |  |
| 5 | Controleer dat het onderdeel “Folder” de waarde “.\TestOut\Faxen” heeft |  |
| 6 | Controleer dat het onderdeel “Email” de waarde “eca.send.fax@tjip.com” heeft |  |
| 7 | Controleer dat het onderdeel “batch-size” de waarde “7” heeft |  |
| 8 | Controleer dat het onderdeel “recipient override” de waarde “” heeft |  |

### Message Details (Incoming e-mail & Outgoing e-mail)

Zorg dat de e-mail jobs als volgd geconfigureerd staan:

CustomConfiguration.xml

<Add name="TestInPop3" description="Incoming POP3 connector" consumingJobName="MsgTestJob2" >

<Pop3 server="localhost" accountName="Pop3" />

</Add>

<Add name="TestInImap" description="Incoming IMAP connector" consumingJobName="MsgTestJob2" >

<Imap server="localhost" accountName="Imap" />

</Add>

StaticConfiguration.xml

<Add name="Pop3" username="test" password="tjipnl" />

<Add name="Imap" username="no-reply" password="tjipnl" />

|  |  |  |
| --- | --- | --- |
| Stap | Actie | Verwacht resultaat |
| 1 | Start Processmanager Start een Processmanager Client | Er draait een instantie van Tjip Processmanager en een Processmanager client. |
| 2 | Start de scheduler door in Processmanager op “1” te drukken | Het systeem start de scheduler. |
| 3 | Zend vanuit processmanager een e-mail naar [test@localhost.nl](mailto:test@localhost.nl) met als onderwerp “testbericht” en als berichttekst: “berichttekst” |  |
| 4 | Ga na het verzenden naar het onderdeel “Operation🡪Messages” |  |
| 5 | Klik op de knop “Refresh” | Het net verstuurde bericht verschijnt in de lijst. |
| 6 | Selecteer het meest nieuwe bericht wat een outgoingEmail type is. |  |
| 7 | Klik op het bericht, en vervolgens op de knop “details” |  |
| 8 | Controleer dat het e-mailadres “test@localhost” is. |  |
| 9 | Controleer dat het onderwerp “testbericht” is. |  |
| 10 | Controleer dat de berichttekst “berichttekst” is. |  |
| 11 | Klik op de knop “Back” |  |
| 12 | Selecteer het meest nieuwe bericht wat een incomingEmail type is. |  |
| 13 | Klik op het bericht, en vervolgens op de knop “details” |  |
| 14 | Controleer dat het e-mailadres “test@localhost” is. |  |
| 15 | Controleer dat het onderwerp “testbericht” is. |  |

# Bijlage E - Performancetest Prototype

Om het prototype te testen is er gekozen om een loadtest te ontwikkelen. Deze loadtest zal worden ingezet met meerdere doelen:

* Het testen van de performance van TJIP Processmanager
* Het testen van de werking van TJIP Processmanager. Doordat vrijwel alle connectors worden gebruikt, wordt een groot deel van de code geraakt met deze test.
* Gebruik bij endurance testen. Deze test genereerd load, wat gebeurt er met de applicatie als de SMTP-server stilvalt, of e-mail niet opgehaald kan worden?

Hieronder een schematisch waarin wordt toegelicht hoe de loadtest werkt:



## Bevindingen

|  |  |  |
| --- | --- | --- |
| **Omschrijving** | **Tester** | **Versie** |
| Programma bevriest bij het inlezen van bestanden | Jeroen Niesen | 6-2-2012 |
| GoSecureIncomming-connector verwerkt geen bestanden | Jeroen Niesen | 6-2-2012 |
| Deadlock excepties op diverse plaatsen in de applicatie, bij grote hoeveelheden berichtenverkeer | Jeroen Niesen | 6-2-2012 |
| Conversieprobleem bij MIMEtype-berichten | Jeroen Niesen | 6-2-2012 |
| Probleem bij het plaatsen van grote hoeveelheden bestanden in de “TestIn”-map | Jeroen Niesen | 9-2-2012 |

## Verbeteringen

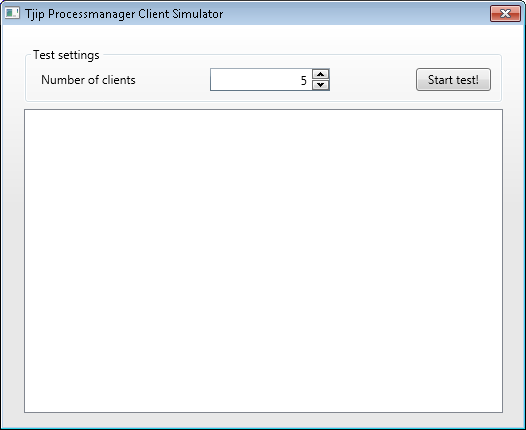
Aan hand van bovenstaande bevindingen, zijn onderstaande reparaties verricht.

* GoSecure connector aangepast. Hier werd geprobeerd het attachment-bericht te verwerken, in plaatst van het XML-bestand. Ik heb ervoor gezorgd dat nu het XML-bestand en het attachment-bestand verwerkt worden.
* Het lock-statement toegevoegd op diverse plaatsen in de code, en waar nodig het isolation-level aangepast, zodat er geen deadlocks meer optreden.
* Lengte gegeven aan de byte-array welke resultaat is van een UTF8-conversie voor mime-type berichten. Wanneer een dergelijke byte-array wordt omgezet naar een string, worden alle lege posities in de array omgezet tot white-space (spaties).
* Timer toegevoegd welke mappen uitleest met een bepaald interval. Wanneer er een bestand wordt geplaatst in een map, en dit zelfde bestand direct door een andere applicatie uitgelezen moet worden, ontstaat er een probleem. Het bestand wordt nog weggeschreven en is daardoor nog in gebruik. Hiervoor is een timer toegevoegd welke met een interval een map uitleest.

# Bijlage F – WCF Performancetest

Om de performance van de WCF-service van TJIP Processmanager in kaart te brengen is er een WCF-test uitgevoerd. De stresstest kan worden uitgevoerd met de op maat gemaakte Client Simulator. In deze applicatie is het mogelijk om een aantal clients in te stellen. Wanneer de test wordt gestart zullen er per client twee of drie requests worden uitgevoerd op de WCF Service.

Tijdens het uitvoeren van de test zal TJIP Processmanager bezig zijn met het verwerken van berichten.



## Bevindingen

De test is uitgevoerd met 270 clients, het Eurocross scenario. Tijdens deze test was het zichtbaar dat requests niet meer uitgevoerd konden worden na een bepaalde tijd. Ook waren er exceptions bij het gebruik van het start en/of stopcommando.

## Verbeteringen

Aan hand van bovenstaande bevindingen zijn de volgende verbeteringen uitgevoerd:

* Linq queries zijn vervangen voor pre-compliled linq queries.
* De database is getuned met de MS SQL Tuning advisor. Op advies van deze applicatie zijn er enkele indexen en statistics toegevoegd.

# Bijlage G – Testrapport

## Testonderwerpen

Testonderwerpen voor dit project waren:

* TJIP Processmanager
* TJIP Processmanager ManagementConsole

## Uitgevoerde testen

### Installatie test

Er is geen installatietest uitgevoerd

### Functionele testen

* Verzenden
  + Verzenden E-mail (minimale gegevens)
  + Verzenden E-mail (alle gegevens)
  + Verzenden E-mail (met bijlagen, regulier)
  + Verzenden E-mail (met bijlagen, te groot, 1 item)
  + Verzenden E-mail (met bijlagen, te groot, meerdere items)
  + Verzenden Fax (regulier)
  + Verzenden Fax (met bijlagen)
  + Verzenden SMS
* Configuratie
  + Weergeven Configuratie (database)
  + Weergeven Configuratie (SMTP)
  + Weergeven Configuratie (SMS)
  + Weergeven Configuratie (Fax)
* Berichten
  + Ingoing E-mail
  + Outgoing E-mail

### WCF Performance test

De WCF service binnen Processmanager is op performance getest.

### Performancetest Processmanager

Het verwerkingsmechanisme en connectoren van Processmanager zijn getest op performance.

### Unittests

|  |  |
| --- | --- |
| Totaal aantal unit-tests | 50 |
| Aantal geslaagde unit-tests | 50 |

### Niet getest

* Berichten
  + Outgoing SMS
  + Outgoing Fax
  + Incoming Message
  + Outgoing Message
* Executionlogs
* Configuratie
  + Webservice

## Bevindingen

|  |  |
| --- | --- |
| Totaal aantal bevindingen: | 6 |
| Aantal wijzigingen aan hand van bevindingen: | 7 |

## Advies

**TJIP Processmanager**Alle tests zijn succesvol doorlopen. TJIP Processmanager kan door voor de acceptatietest.

**TJIP Processmanager Managementconsole**   
Niet alle onderdelen van de managementconsole zijn getest. Het advies luid om de testset verder uit te breiden.