

Afstudeerverslag

**Opdrachtgever**

Lars Truijens

**Begeleider**

Norbert Radocz

**Expert**

Arnaud Poelman

**Auteurs**

Youri Raggers (20068333)

**Datum:** Zaterdag 23 Maart 2013

**Versie:** 1.0

Referaat

**Omschrijving**  Dit document beschrijft alle ontwikkelingen rond een afstudeerproject gericht op het vervangen van een rapportgenerator.

**Descriptoren** WPF, C#, MVVM, Delphi, Adviesrapport, Interview, Conversie, ICS, ChipSoft,

**Organisatie** ChipSoft B.V. Amsterdam TT Melissaweg 23-25

**Afstudeerperiode** 3 sept 2012 t/m 11 jan 2013

**Examinator (begeleider)** V. Broeren

**Examinator (expert)** R. Ruijsenaars

**Afstudeerbegeleider** N. Radocz

**Auteur** Youri Raggers

Voorwoord

Als afsluiting van de HBO opleiding informatica van de Haagse Hoge School is het gebruikelijk voor de student om een laatste keer op stage te gaan. Deze afstudeerstage duurt een periode van zeventien weken waarin de student zelfstandig een project uitvoert. Dit document is het verslag van mijn afstudeerstage bij het bedrijf ChipSoft B.V.

Voor de hulp die ik tijdens dit project heb gehad wil ik een aantal mensen bedanken.

Allereerst wil ik alle medewerkers bij ChipSoft die mij geholpen, ondersteund of voor een fijne werksfeer hebben gezorgd, bedanken. In het bijzonder wil ik Norbert Radocz, mijn bedrijfsbegeleider, bedanken voor de begeleiding en hulp. Ook wil ik Lars Truijens, opdrachtgever, bedanken voor de mogelijkheid om mijn afstudeerstage te volgen bij ChipSoft en de hulp tijdens de evaluaties. Als laatst wil ik Arnaud Poelman bedanken voor zijn expertise.

Van de HHS wil ik mijn examinatoren Vincent Broeren en Remco Ruijsenaars voor hun begeleiding en ondersteuning bij het schrijven van dit verslag bedanken.

Als laatst wil ik mijn familie bedanken voor de hulp bij het maken van dit document met betrekking tot mijn dyslexie.

Noordwijk, 24 maart 2013

Youri Raggers

Inhoud

[Referaat 2](#_Toc351897999)

[Voorwoord 3](#_Toc351898000)

[1 Inleiding 7](#_Toc351898001)

[2 Bedrijf 8](#_Toc351898002)

[2.1 Geschiedenis 8](#_Toc351898003)

[2.2 Missie 8](#_Toc351898004)

[2.3 Marktaandeel 8](#_Toc351898005)

[2.4 Structuur 9](#_Toc351898006)

[2.5 Sfeer 9](#_Toc351898007)

[3 Project 10](#_Toc351898008)

[3.1 Omgeving 10](#_Toc351898009)

[3.1.1 ICS datastructuur 10](#_Toc351898010)

[3.1.2 MVVM 11](#_Toc351898011)

[3.2 Probleemstelling 11](#_Toc351898012)

[3.3 Doelstelling 11](#_Toc351898013)

[3.4 Resultaat 11](#_Toc351898014)

[4 Definities 12](#_Toc351898015)

[4.1 Rapportdefinities 12](#_Toc351898016)

[4.2 Klasse voorbeeld 13](#_Toc351898017)

[4.3 Veldenkiezer 13](#_Toc351898018)

[5 Opstart van het project 14](#_Toc351898019)

[5.1 Maken Plan van Aanpak 14](#_Toc351898020)

[5.1.1 Omschrijven opdracht 14](#_Toc351898021)

[5.1.2 Projectmanagement keuzen 14](#_Toc351898022)

[5.1.3 Opstellen planning 15](#_Toc351898023)

[6 Sprint 1: Informatie verzamelen 16](#_Toc351898024)

[6.1 Volgen cursussen 16](#_Toc351898025)

[6.2 Interviews afnemen 16](#_Toc351898026)

[6.2.1 Uitkomst interviews 16](#_Toc351898027)

[6.3 Opstellen oriëntatie document 17](#_Toc351898028)

[6.4 Zoeken rapportgenerators 18](#_Toc351898029)

[6.5 Onderzoeken rapportgeneratoren 19](#_Toc351898030)

[6.6 Retrospective 19](#_Toc351898031)

[7 Sprint 2: Onderzoek 20](#_Toc351898032)

[7.1 Onderzoeken rapportgeneratoren 20](#_Toc351898033)

[7.2 Retrospective 21](#_Toc351898034)

[8 Sprint 3: Prototype 1 22](#_Toc351898035)

[8.1 Opstellen use-case 22](#_Toc351898036)

[8.2 Maken klasse-diagram 23](#_Toc351898037)

[8.2.1 Verwerken datastructuur 24](#_Toc351898038)

[8.2.2 Opzetten conversie 26](#_Toc351898039)

[8.2.3 Implementeren StimulSoft 28](#_Toc351898040)

[8.3 Retrospective 28](#_Toc351898041)

[9 Sprint 4: Prototype 2 29](#_Toc351898042)

[9.1 Implementeren DevExpress 29](#_Toc351898043)

[9.1.1 WinForm problemen 29](#_Toc351898044)

[9.1.2 DevExpress probleem 30](#_Toc351898045)

[9.2 Ombouwen converter 31](#_Toc351898046)

[9.3 Omzetten datastructuur 33](#_Toc351898047)

[9.4 Retrospective 35](#_Toc351898048)

[10 Sprint 5: Prototype 3 36](#_Toc351898049)

[10.1 Implementeren SharpShooterReports 36](#_Toc351898050)

[10.2 Updaten oude prototypes 37](#_Toc351898051)

[10.2.1 Updaten ITypedList (DevExpress) 37](#_Toc351898052)

[10.2.2 Uitzoeken oplossing Aero effect (DevExpress) 37](#_Toc351898053)

[10.2.3 opzetten dataprovider (StimulSoft) 38](#_Toc351898054)

[10.2.4 Onderzoeken problemen enumerator 38](#_Toc351898055)

[10.3 Updaten converters 39](#_Toc351898056)

[10.4 Retrospective 39](#_Toc351898057)

[11 Sprint 6: Prototype 4 40](#_Toc351898058)

[11.1 Zelf maken rapportgenerator 40](#_Toc351898059)

[11.1.1 Ontwerp GUI 40](#_Toc351898060)

[11.1.2 Maken klasse-diagram 41](#_Toc351898061)

[11.2 Onderzoek Microsoft Reportbuilder 3.0 42](#_Toc351898062)

[11.3 Performance testen 42](#_Toc351898063)

[11.3.1 Uitvoeren tests 42](#_Toc351898064)

[11.4 Retrospective 43](#_Toc351898065)

[12 Sprint 7: Adviesrapport 44](#_Toc351898066)

[12.1 Opstellen adviesrapport 44](#_Toc351898067)

[12.2 Converter 45](#_Toc351898068)

[12.2.1 Maken klasse-diagram 47](#_Toc351898069)

[12.3 Retrospective 48](#_Toc351898070)

[13 Evaluatie 49](#_Toc351898071)

[13.1 Sprint 1: Evaluatie 49](#_Toc351898072)

[13.1.1 Productevaluatie 49](#_Toc351898073)

[13.1.2 Procesevaluatie 49](#_Toc351898074)

[13.2 Sprint 2: Evaluatie 50](#_Toc351898075)

[13.2.1 Productevaluatie 50](#_Toc351898076)

[13.2.2 Procesevaluatie 50](#_Toc351898077)

[13.3 Sprint 3: Evaluatie 51](#_Toc351898078)

[13.3.1 Productevaluatie 51](#_Toc351898079)

[13.3.2 Procesevaluatie 52](#_Toc351898080)

[13.4 Sprint 4: Evaluatie 53](#_Toc351898081)

[13.4.1 Productevaluatie 53](#_Toc351898082)

[13.4.2 Procesevaluatie 53](#_Toc351898083)

[13.5 Sprint 5: Evaluatie 54](#_Toc351898084)

[13.5.1 Productevaluatie 54](#_Toc351898085)

[13.5.2 Procesevaluatie 55](#_Toc351898086)

[13.6 Sprint 6: Evaluatie 56](#_Toc351898087)

[13.6.1 Productevaluatie 56](#_Toc351898088)

[13.6.2 Procesevaluatie 56](#_Toc351898089)

[13.7 Sprint 7: Evaluatie 58](#_Toc351898090)

[13.7.1 Productevaluatie 58](#_Toc351898091)

[13.7.2 Procesevaluatie 58](#_Toc351898092)

[13.8 Competenties 59](#_Toc351898093)

[13.8.1 1.2 Voorbereiden en opstarten softwareontwikkeltraject 59](#_Toc351898094)

[13.8.2 1.4 Uitvoeren analyse door definitie van requirements 59](#_Toc351898095)

[13.8.3 3.2 Ontwerpen systeemdeel 59](#_Toc351898096)

[13.8.4 3.3 Bouwen applicatie 59](#_Toc351898097)

[13.8.5 2.3 Uitvoeren gegevensconversie 59](#_Toc351898098)

# Inleiding

Dit document is geschreven om inzicht te geven in het proces dat ik heb doorgelopen tijdens mijn afstudeerproject bij ChipSoft. In dit document wordt beschreven welke werkzaamheden ik heb uitgevoerd en welke keuzes ik daarbij heb genomen.

In de 17 weken dat ik gewerkt heb bij ChipSoft heb ik een onderzoek gedaan naar vervanging van hun rapportgenerator. Hierbij zijn een aantal prototypes gemaakt om tot een algehele conclusie te komen.

Dit document is opgedeeld in drie delen. Het eerste deel is ter achtergrondinformatie en gaat over het bedrijf waar ik heb gewerkt en wat mijn opdracht binnen dat bedrijf precies inhield.

Het tweede deel betreft de sprints die ik heb uitgevoerd tijdens mijn stage. Dit deel geeft de werkzaamheden en keuzes die ik heb gemaakt weer. Elke sprint is opgedeeld in de activiteiten die ik had gepland en retrospective.

Het laatste deel is de evaluatie van mijn proces. Er is een evaluatie over het geheel en een evaluatie per sprint.

# Bedrijf

## Geschiedenis

ChipSoft is ontstaan door een simpele IBM applicatie in de jaren 80. Deze applicatie was gemaakt door Hans Mulder met als doel het maken van facturen voor zijn vader, Gerrit Mulder een Chirurg. Deze software voorzag in een behoefte waardoor andere specialisten hier ook interesse in kregen.

Deze applicatie is gegroeid tot het punt dat 60% van de specialisten in de jaren 90 hier gebruik van maakte. Tijdens de jaren 90 is er meer vraag gekomen naar software gericht op verschillende afdelingen waar ChipSoft op heeft ingespeeld met modules voor logistiek, voorraad en patiënt gegevens. Dit had als resultaat dat in 1994 sprake was van een volledig ziekenhuisinformatiesysteem (ZIS). Dit product is verder geëvolueerd totdat het uiteindelijk in 2003 resulteerde in de introductie van het volledig Windows-gebaseerde systeem met de naam ChipSoft Elektronisch ZIS (CS-EZIS).

In de loop van de tijd is ChipSoft verder blijven uitbreiden en met de groei van het internet en portalen is er uiteindelijk in 2006 een Zorg Portal toegevoegd aan EZIS.

Door de groei van EZIS zijn er in totaal 300 mensen in dienst van ChipSoft verdeeld over drie locaties (Amsterdam hoofdkantoor, Drachten, Niel in België). Van deze 300 werknemers zijn er 90 ontwikkelaars die EZIS nog steeds aan het uitbreiden en verbeteren zijn.

## Missie

De missie van ChipSoft is: “Met flexibele software streeft ChipSoft ernaar de klant een zo optimaal mogelijke oplossing te bieden die bijdraagt aan efficiency en betere zorg.”

ChipSoft voert een proactief innovatiebeleid uit met als doel in 90% van de gevallen een functionaliteit al aan te kunnen bieden op het moment dat de vraag in de markt ontstaat.

Door dit proactieve productbeleid en door te kiezen voor een toekomst vaste architectuur, zal ChipSoft de komende jaren met de huidige productlijnen dezelfde sterke groei kunnen realiseren als in de afgelopen jaren. Als marktleider participeert ChipSoft in allerlei overheidsinitiatieven rondom bijvoorbeeld het Elektronisch Medicatie Dossier en de unieke zorgverlener identificatie.

Door de participerende en stimulerende rol in deze landelijke en regionale initiatieven wordt ChipSoft in toenemende mate gezien als een belangrijke partner in de transmurale ambities van zorginstellingen. Deze rol zal ChipSoft de komende jaren verder uitbouwen.

## Marktaandeel

Op het moment is ChipSoft marktleider op het gebied van ZIS binnen Nederland. Ongeveer 60% van alle ziekenhuizen maken hier gebruik van. Daarnaast is het door de modulaire opbouw van EZIS mogelijk om het te gebruiken bij Zelf Behandeling Centra en bij GGZ-Instellingen. Deze opbouw zorgt er ook voor dat EZIS in delen wordt afgenomen. Zo wordt het Operatiekamer-Managementsysteem bij 60 instellingen gebruikt en het facturensysteem bij 55 instellingen.

## Structuur

ChipSoft is een lijn organisatie. Met aan het hoofd een driedelige directie. Als voorzitter Hans Mulder, voor afdeling Research & Development (R&D) Robert Hardholt en Implementatie & Support (I&S) directeur Gerrit Mulder. Zij worden ondersteund door het secretariaat en de administratie.

Onder de directie zijn ook een drietal afdelingen gelijk aan de rollen van de directie.

Figuur 1 Organogram

Op de afdeling R&D heeft mijn stage plaats gevonden. De R&D afdeling is opgedeeld per module die aanwezig is in EZIS. Ik stond zelf apart van deze modules. Mijn opdrachtgever en begeleider werken allebei voor de WPF Core module (EZIS interface). Mede omdat de Rapportbuilder in de Core module verwerkt zit. De Expert die geholpen heeft bij dit project onderhoudt specifiek de Rapportbuilder in de WPF en Delphi omgeving.

## Sfeer

Bij ChipSoft heerst een prettig informele sfeer. Ook wordt binnen ChipSoft veel in teamverband gewerkt waarbij de collegialiteit zeer belangrijk is. Dit houdt in dat de mensen hier gewend zijn om zich aan te passen aan een nieuwe situatie of aan snelle veranderingen. Een bijkomend voordeel hiervan is dat een applicatie die wordt ontwikkeld, gelijk kan worden voorzien van de nieuwe technologieën.

# Project

## Omgeving

ChipSoft levert een uitgebreid software pakket aan zorginstellingen. Dit pakket is opgebouwd op basis van een multi-tier architectuur. Eén van de onderdelen in de multi-tier is een rapportgenerator, waarmee de eindgebruiker rapporten kan uitprinten die geheel naar eigen wensen zijn opgemaakt. Hiervoor wordt gebruik gemaakt van ReportBuilder Pro, dat naadloos aansluit op de multi-tier architectuur.

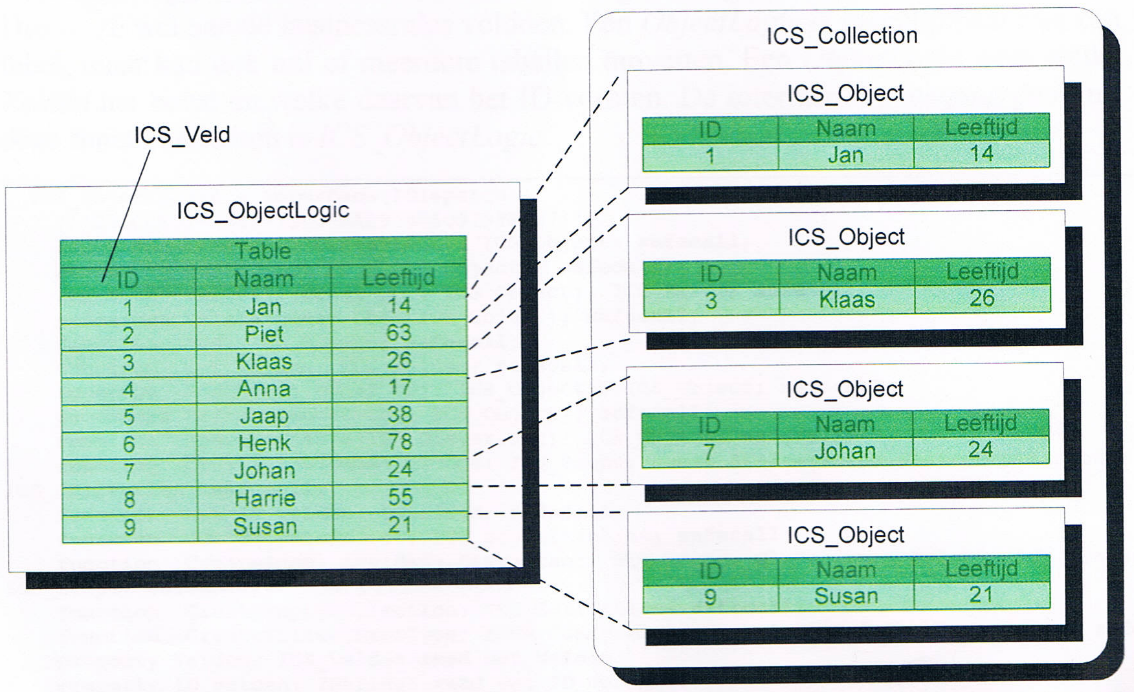
De rapportgenerator is een onderdeel van de basis onderdelen van het EZIS en is aan te spreken vanuit elke module om een rapport te genereren. Omdat EZIS gebruik maakt van een eigen datastructuur ‘Interface van ChipSoft’ (ICS Zie 3.1.1) is de code van het programma aangepast om daarmee om te kunnen gaan.

In EZIS 6.0 wordt gebruik gemaakt van de design pattern Model View ViewModel (MVVM Zie 3.1.2). Dit design pattern maakt het mogelijk om het interface te scheiden van de business laag. Deze architectuur keuze wordt gebruikt in samenwerking Windows Presentation Foundation (WPF)

### ICS datastructuur

Deze structuur is de logica-laag van EZIS en bestaat uit verschillende interfaces

(Zie Figuur 2). De samenwerking tussen deze interfaces zorgt ervoor dat de presentatie-laag aan zijn gegevens kan komen en dat de business rules nageleefd worden.



Figuur 2 ICS Datastructuur

Een ICS Logic kan gezien worden als een query die gebruikt wordt om data uit de database te halen. Het resultaat van die query is een ICS Collection gevuld met records. Elk record in een collectie is een ICS Object dat bestaat uit velden met waarden, deze velden kunnen ook complexe datatype zijn bijv. ICS\_Objecten of ICS\_Collections.

Met de ICS datastructuur is het mogelijk om collecties op twee manieren op te vragen.

1. De live collectie; deze collectie haalt een tiental records uit de database en laad deze in het geheugen. Als er meer records nodig zijn, worden de volgende tien opgehaald enzovoorts.
2. De memory collectie; met deze collectie worden alle records in een keer in het geheugen geladen.

### MVVM

EZIS maakt gebruik van een MVVM architectuur. Om hiervan gebruik te maken zijn er drietal klassen die gebruikt moeten worden.

* **View**

De view is de visuele interpretatie van data. Deze bestaat alleen uit XAML[[1]](#footnote-1) en maakt geen gebruik van andere codes.

* **ViewModel**

In de ViewModel vinden de bewerking van de Models plaats. De Models die hierin gebruikt worden kunnen worden weergegeven in de View.

* **Models**

Models vertegenwoordigen objecten die gebruikt worden in het programma.

De MVVM architectuur die bij ChipSoft wordt gebruikt, wijkt af van de standaard manier. Er wordt gebruik gemaakt van een Controller klass die een connectie maakt tussen de View en de ViewModel. Dit zorgt ervoor dat de Views en ViewModels binnen EZIS compleet gescheiden zijn van elkaar.

## Probleemstelling

Recent is ChipSoft overgestapt naar een C# omgeving met WPF. Omdat de huidige rapport- generatoren in win32 is geschreven met het gebruik van Borland VCL (Delphi) is het niet mogelijk om deze goed te verwerken in de WPF user interface.

## Doelstelling

Het doel van het project is om een vervanging te zoeken voor de rapportgeneratoren en deze te implementeren in de multi-tier architectuur. Om het project af te kunnen ronden is dit doel verdeeld in drie delen.

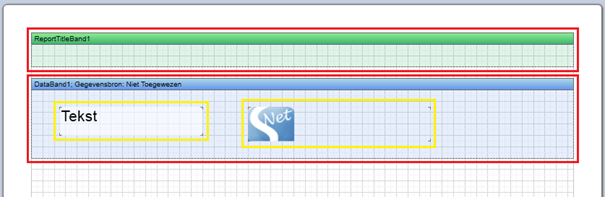
Ten eerste moet er onderzocht worden welke eisen er gesteld worden aan de rapport- generator. Als tweede wordt er een adviesrapport geschreven om nieuwe kandidaten voor de rapportgenerator te beoordelen. Dit wordt gedaan aan de hand van de eisen en een aantal prototypes. Als laatst wordt de generator die uit het advies naar voren komt geïmplementeerd in de multi-tier applicatie.

## Resultaat

Het gewenste eindresultaat van het project is een volledige implementatie van de nieuwe rapportgenerator in de Multi-tier architectuur van ChipSoft. Omdat er tijdens de onderzoeksfase uitgebreide prototypes gemaakt moeten worden om tot een volledig advies te komen, is het mogelijk dat er tijdens het project niet voldoende tijd is voor een volledige implementatie.

# Definities

## Rapportdefinities

In dit document wordt er veel gesproken over rapporten en verschillende elementen die daarin van toepassing zijn. In Figuur 3 worden verschillende elementen gemarkeerd en gevolgd door een legenda die deze uitlegt. Deze definities worden door het gehele document gebruikt.

Figuur 3 Rapport Page

**Page**

De page is de pagina die de layout van het rapport representeert. Hierop worden de elementen geplaatst die weergegeven worden.

**Band**

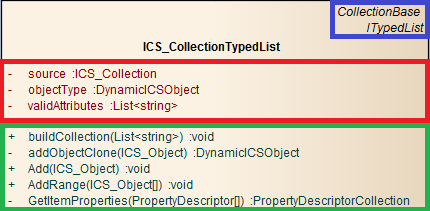
Dit zijn elementen die over de gehele breedte van een Page weer worden gegeven. Bands worden gebruikt om gegevens weer te geven op een bepaalde frequentie. Een Rapport Band wordt bijvoorbeeld maar een keer per rapport getoond en een Data Band wordt zo vaak herhaald als er records in een gegeven collectie aanwezig zijn.

**Object**

Objecten zijn elementen die in een rapport geplaatst kunnen worden met verschillende properties. Grootste onderscheid tussen objecten en bands is de mogelijkheid om coördinaten vast te stellen, hoogte en breedte op te geven.

## Klasse voorbeeld

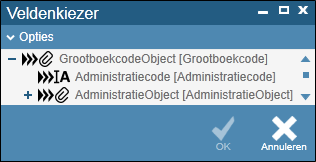
Tijdens dit project is er veel gebruik gemaakt van overerving. Omdat deze abstracte klassen en interfaces waarvan wordt overgeërfd, de diagrammen te groot maken voor dit document is er voor gekozen om deze niet te tonen. Om te laten zien wanneer er sprake is van overerving word er in de rechterbovenhoek van een klasse de namen getoond van de geërfde klassen. Als een klasse begint met een de letter ‘I’ duid dat aan dat het een Interface is (Zie Figuur 4).



Figuur 4 Klasse Header

## Veldenkiezer

De Veldenkiezer is een scherm binnen EZIS waarin een ICS Collectie weergegeven kan worden. De velden van een collectie worden in een TreeView weergegeven wat het mogelijk maakt om te navigeren door de structuur. Bij elk veld wordt getoond welk datatype het is en in het geval van een Object of Collectie kunnen deze geopend worden (Zie Figuur 5).



Figuur 5 Veldenkiezer met collectie artsen

# Opstart van het project

Het opstarten van het project nam de eerste twee weken van het project in beslag. De opstart was gesplitst in twee delen. Ten eerste kennismaking met de werkomgeving. Dit werd gedaan door middel van een viertal cursussen met betrekking op WPF, de architectuur van ChipSoft en de daarbij horende datastructuur. Het tweede deel was het opstellen van het Plan van Aanpak (Zie Bijlage 1).

## Maken Plan van Aanpak

Om het project een goede richting aan te geven, heb ik in het begin een plan van aanpak gemaakt. Dit document beschrijft de aanpak die ik tijdens dit project heb gevolgd.

### Omschrijven opdracht

Om de opdracht goed in kaart te brengen is er gebruik gemaakt van meerdere bronnen. Zo was er opdracht formulier met de globale omschrijving van de opdracht. Omdat hier niet precies werd uitgelegd welke problemen er waren is er een kort gesprek geweest met de expert die het onderhoud regelt van de rapportgenerator. Ook is er een gesprek geweest met mijn begeleider en opdrachtgever over de randvoorwaarden en de omgeving.

### Projectmanagement keuzen

ChipSoft gebruikt als software ontwikkelmethode de Agile methodiek Scrum

(Zie Bijlage 1 Hfd Aanpak). Tijdens dit project wordt er dan ook gebruik gemaakt van deze methode.

Scrum is een management methodiek bedoeld voor het ontwikkelen van een programma en sluit daardoor niet goed aan op dit project. Aan het einde van een Sprint moet volgens de regels van Scrum een deployable product afgeleverd worden. Omdat er bij dit project een onderzoek wordt verricht en niet alleen een product ontwikkeld wordt, gaat dit niet op.

Om toch enigszins aan deze eis van Scrum te voldoen wordt er aan het einde van elke onderzoekssprints-documenten met de gevonden bevindingen opgeleverd. Bij het opzetten van de productbacklog is globaal besproken wat er per onderzoek onderzocht moet worden en wat gebruikt wordt als referentie voor de “Definition of Done”. Deze definitie wordt tijdens het opzetten van de sprint verder uit gewerkt.

Omdat ik deze opdracht alleen uitvoer wordt de Stand-up Meeting niet uitgevoerd maar had ik elke week een bespreking met mijn begeleider. Bij deze bespreking werden vervolgens de ontwikkelingen van die week besproken en enige knelpunten aangekaart. Aan het eind van een Sprint wordt samen met de begeleider, opdrachtgever en expert het eindresultaat besproken en de volgende Sprint grofweg gepland. De Retrospective werd ook gebruikt om de sprint te evalueren om zo mijn opdrachtgever een beter zicht te geven op de voortgang en te betrekken in het beslissingsproces.

De laatste aanpassing is het omgaan met de eisen. Tijdens dit project wordt er gebruik gemaakt van twee lijsten met functionaliteiten.

* **Product Backlog** (Zie Bijlage 7)

In de product backlog zijn de eisen opgenomen gerelateerd aan dit project en de functionaliteiten die prototypes moeten hebben.

* **Orientatie Document** (Zie Bijlage 3)

Deze functionaliteit is gebonden aan de 3th party programma’s die onderzocht worden. Hierbij kunnen bepaalde functies ontbreken om deze rede is er gekozen om MoSCoW te gebruiken voor deze functionaliteiten.

### Opstellen planning

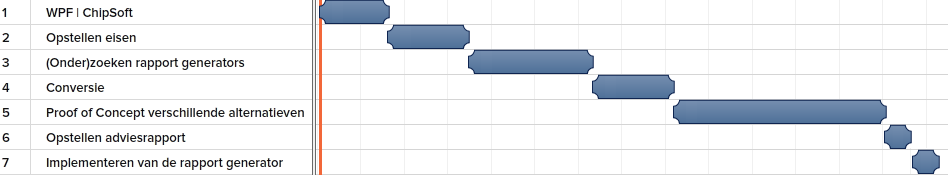
Omdat bij Scrum de sprints pas worden ingevuld aan het einde van de voorgaande sprint is een detail planning goed mogelijk. Daarom heb ik het project onderverdeeld in een viertal fases. Deze fase zijn niet gerelateerd aan Scrum en zijn alleen gebruikt om een grove planning aan te geven (Zie Figuur 6).

De eerste fase is de interview fase. Tijdens deze fase worden de eisen die gesteld worden aan de rapportbuilder in kaart gebracht. Ook wordt er tijdens deze fase gekeken naar de huidige situatie en hoe deze is geïmplementeerd.

Fase twee wordt het opzetten van het adviesrapport. Hierbij zal gezocht worden naar rapportgeneratoren die mogelijk als vervanger kunnen gelden en worden deze afgewogen tegenover de eisen die vastgesteld zijn tijdens de eerste fase. Deze fase eindigt met een initieel advies.

Na het advies rapport wordt er aan de hand van het initiële advies een aantal prototypes gemaakt. Er wordt vanuit gegaan dat er tijd is om drie prototypes te maken in de gegeven tijd. Deze prototypes zullen ontworpen worden met Unified Modeling Language (UML) en gebruik maken van de data uit EZIS. Na het maken van alle prototypes wordt het advies rapport uitgebreid met de vinding van deze fase.

De laatste fase van dit project zal de implementatie van de nieuwe rapportgenerator en de conversie van de oude rapporten zijn.

****

Figuur 2 Planning

# Sprint 1: Informatie verzamelen

Het doel van de eerste sprint is het verzamelen van de wensen die gesteld worden aan het product en problemen die er op het moment aanwezig zijn. Dit werd gedaan door middel van een aantal interviews. Daarnaast wordt er tijdens deze sprint gezocht naar mogelijke vervangers voor de rapportbuilder.

## Volgen cursussen

Omdat ik niet wist wat rapportbuilder pro is en wat deze voor mogelijkheden heeft heb ik besloten om dat uit te zoeken. Dit heb ik gedaan door het volgen van een tweetal cursussen. Deze cursussen worden door ChipSoft zelf aangeboden aan zowel klanten als medewerkers en gegeven door medewerkers van de I&S afdeling.

De eerste cursus is gericht op de overzichtsgenerator. Dit is een tool die gebruikt wordt om data te filteren, sorteren en groeperen voordat het wordt door gestuurd naar een output. Deze cursus heb ik gevolgd om meer inzicht te krijgenover hoe de data naar de rapportgenerator gestuurd kan worden.

De andere cursus is het gebruiken van de rapportbuilder. Hierbij kreeg ik inzicht op de huidige mogelijkheden en werking van het programma. Deze kennis zou ik tijdens het beoordelen van de vervangende generatoren kunnen gebruiken om te kunnen bepalen hoe een eventuele overgang er zal uitzien. De cursus rapportbuilder is uiteindelijk uitgesteld door ChipSoft zelf. Omdat ik hier toch naar toe wilde heb ik mij ingeschreven voor de daaropvolgende cursus. De eerste volgende cursus werd pas een maand gegeven dus viel buiten deze sprint.

## Interviews afnemen

Om een goede doelgroep te krijgen voor de interviews heb ik eerst gekeken naar de mensen die het meest met de rapportbuilder werken. Dit heb ik gedaan door eerst een interview te hebben met de persoon die de reportbuilder onderhoudt, Arnaud Poelman (de expert tijdens dit project). Tijdens dat gesprek adviseerde hij te praten met de afdeling I&S. Ook kwamen een aantal klachten tijdens dit gesprek naar voren (Zie 6.3).

Omdat de afdeling I&S is onderverdeeld in een aantal groepen (overeenkomend met de modules in EZIS) moest ik eerst bepalen wie ik daar zou interviewen. Aan de hand van de klachten die ik tijdens mijn interview met expert heb gekregen, heb ik besloten om de I&S groepen waarvan die klachten kwamen te spreken. Naast de I&S leek het mij ook verstandig om de docent van de cursus te spreken over de rapportbuilder omdat zij hier veel mee werkt.

Bij de interviews is er gevraagd welke mogelijkheden een rapportbuilder voor hen nodig heeft. En welke problemen zij tegen kwamen tijdens het gebruik van de huidige reportbuilder.

### Uitkomst interviews

Initieel was ik van plan om nog een ronde te doen langs de andere I&S groepen om te kijken of er nog andere wensen waren. Hier heb ik tijdens de sprint van afgezien na de gespreken met medicatie, factuur en de docent van de reportbuilder cursus. Uit de interviews kwamen al snel veel van dezelfde eisen (Bijv. het weergeven van tekst en het printen van rapporten) naar voren tevens waren alle klachten vastgelegd bij de expert. Ik heb toen in overleg met mijn begeleider besloten om het bij een viertal interviews te houden. (Zie Bijlage 2)

## Opstellen oriëntatie document

In het oriëntatie document heb ik de interviews verwerkt (Zie Bijlage 3). Dit document is verdeeld in twee delen:

**Huidige situatie**

De informatie hierin is vooral afkomstig uit de gespreken met de expert en mijn opdrachtgever. In dit hoofdstuk wordt een versimpelde architectuur weergegeven van de rapportbuilder en daaraan gerelateerde modules. Ik heb zelf een klasse-diagram gereverse engineerd om een beeld te krijgen van de huidige implementatie. Aan het einde worden alle problemen opgesomd (Zie tabel 1).

|  |
| --- |
| Air-Space probleem |
| veroorzaakt door Win32 element te gebruiken binnen een WPF omgeving. Work-Around voor aanwezig. (Zie Figuur 16) |
| Memory Leak |
| veroorzaakt door slechte communicatie tussen de C# omgeving en de oude Delphi omgeving. |
| Task Module |
| Als er tijdens het een print batch het memory probleem zich voor doet blijft de Task Module hangen. |
| Print probleem |
| Het is mogelijk om EZIS in een Client-Server omgeving te gebruiken. Hierbij kon het voorkomen dat er op de verkeerde printer werd geprint. |

Tabel 1 Klachten

**Eisen**

Alle eisen zijn afkomstig uit de interviews en gespreken met de opdrachtgever. De eisen hebben allemaal een beschrijving gekregen zodat duidelijk is wat de bedoeling is. Tevens heb ik gebruik gemaakt van MoSCoW [[2]](#footnote-2)om het belang van alle eisen aan te kunnen duiden.

|  |
| --- |
| WPF Viewer |
| De nieuwe rapportbuilder is bedoeld voor EZIS 6.0. Het nieuwe systeem is geschreven in WPF daarom is een WPF oplossing het beste te integreren. |
| Designer |
| Het moet mogelijk zijn voor iedereen om een rapport te maken. Er moet dus een grafisch element aanwezig zijn, waarin het mogelijk is om een rapport te maken door het plaatsen en slepen van rapport elementen. Deze designer hoeft niet verwerkt te zijn in EZIS. |
| Tekst |
| De data uit een de ICS Objecten of Collecties moet weergegeven kunnen worden als tekst. |
| Besturen van uit code |
| Op het moment kan er voor het printen een rapport worden geselecteerd en kan deze gelijk geprint worden. Het moet mogelijk zijn om een rapport te printen zonder dat de rapportbuilder gestart moet worden. |
| Printen |
| Ook al wordt er steeds minder geprint wordt dit voor sommige modules nog steeds gedaan bijv. factuur. Deze moeten nota’s uitprinten om deze naar verzekeringsmaatschappijen te sturen. |

Tabel 2 Primaire eisen

## Zoeken rapportgenerators

Voor het zoeken naar rapportgenerators is gebruik gemaakt van Google en Developer forums zoals Stackoverflow[[3]](#footnote-3) en Code Project[[4]](#footnote-4). Hierbij is gezocht naar report en C#. Er is niet specifiek gezocht naar een WPF generator omdat dit eventuele code library’s zou overslaan.

Tijdens het zoeken zijn er zestien reportbuilders gevonden. De meeste generatoren bestonden uit een engine voor het maken van een rapport en een Graphical User Interface (GUI) die gebruikt maakt van die engine. Dit maakt het eventueel mogelijk om zelf een GUI te maken voor een gekozen engine.

Van deze generators bleken er een viertal een vorm van Java te zijn die een wrapper naar .Net hadden. Omdat het een eis is dat het zou werken met een C# omgeving vielen deze al af. Dit stamt af van het huidige memory leak probleem dat veroorzaakt wordt door cross language communicatie (Zie 6.3).

Vervolgens is er gekeken naar de mogelijkheid voor functioneren onder WPF. De overige generatoren hadden allemaal wel een WPF Viewer waarmee gemaakte documenten bekeken kunnen worden. Alleen ontbrak de mogelijkheid om de reports op te maken in WPF of werd er gebruik gemaakt van een Visual Studio add-in waarmee reports worden gemaakt. Deze laatste categorie is tijdens deze sprint uitgesloten naar aanleiding van de interview waarin een afkeer tegen splitsing van de designer naar voren kwam (Zie bijlage 3).

## Onderzoeken rapportgeneratoren

Tijdens het onderzoek kwam ik er al snel achter dat de mogelijkheden van de generatoren veelal overeen kwamen. Het onderzoek is daarom niet gericht op de mogelijkheden maar op het gebruik, support opties, licentie en kosten.

Bij het testen van het gebruik is de manier hoe data geladen wordt enigszins genegeerd. Dit is pas van toepassing bij de prototype fase van het project.

Bij het gebruiken van de generatoren is er ook gebruik gemaakt van de daarbij geleverde handleidingen om zo een aantal test rapporten te maken. Hierbij is gekeken naar de eenvoud van de handelingen.

Bij de beoordeling van support is gekeken naar de mogelijkheden hoe support wordt aangeboden door de bedrijven. Er is gekeken naar mogelijke community forums en hoe die opgezet zijn, hoeveelheid gebruikers en de reactie tijd van administrators. De duidelijkheid van de handleidingen zijn hier ook in opgenomen. Bij drie van de tien generators is het nodig geweest om contact op te nemen met de support en is het mogelijk geweest op de reactietijd te vergelijken.

Het laatste gedeelte van het onderzoek was het uitzoeken van de kosten en de licenties die daarbij hoorde. Deze staan gepubliceerd op de websites van de applicaties producenten en kwamen veelal overeen. (Zie Bijlage 4)

## Retrospective

Tijdens de sprint retrospective waren naast mijn begeleider ook de opdrachtgever en expert aanwezig. De bespreking was op te delen in drie delen:

* **Oriëntatie document**

Ik heb na de bespreking nog even met expert gesproken over de huidige situatie maar tijdens de bespreking waren we vooral geïnteresseerd in de samengestelde eisen. Ik heb van een aantal eisen uitgelegd wat deze inhielden. Uiteindelijk moesten van sommige de omschrijvingen uitgewerkt worden, maar zijn er geen eisen afgeschreven of bij gekomen. Wel is er een discussie geweest over de MoSCoW waarde die ik gegeven had aan de eisen.

* **Initieel onderzoekrapport**

De eerste vraag die ik kreeg bij het besprek van het rapport was hoe ik de generatoren had geselecteerd. Ik ben er vanuit gegaan dat de GUI specifiek gemaakt zou moeten zijn in WPF om het te integreren. Omdat veel rapportgeneratoren die ik gevonden had wel een WPF Viewer hadden maar een WinForm designer had ik die afgeschreven. Omdat er door deze keuze veel opties zijn afgevallen is er besloten de eis WPF te splitsen in Viewer en Designer. De WPF Viewer is vervolgens toegevoegd aan de Must have eisen en de WPF Designer aan de Could have eisen. Tevens moesten support en gebruikersgemak een waarde krijgen zodat deze mee gewogen kunnen worden in het rapport.

Tevens moest er een overzicht gemaakt worden tegenover de nu vastgestelde eisen.

* **Volgende sprint**

Ik verwachte zelf nog een week bezig te zijn met het opstellen van het initiële advies. Na wat besprekingen over scenario’s uitwerken besloot ik toch nog een hele sprint te besteden aan het opstellen van het advies. Hierbij zou ik de usability verder uitwerken en een aantal extra generatoren.

# Sprint 2: Onderzoek

Deze sprint is gericht op het onderzoeksrapport zodanig aanvullen dat het mogelijk zou zijn om een initieel advies te kunnen maken. De tasks die tijdens deze sprint zijn opgenomen, zijn gemaakt in het verlengde van de vorige sprint gezien ik daar ook al bezig was met het opstellen van het rapport.

## Onderzoeken rapportgeneratoren

Omdat er tijdens de evaluatie van de vorige sprint besloten is om toch WinForm designers toe te laten heb ik mijn eerste lijst met zestien generatoren gepakt. Omdat er specifiek gericht werd op producten gemaakt in C# zijn zes generatoren ook van deze lijst gehaald.

Omdat ik van vijf van deze tien al een aantal onderdelen had onderzocht, besloot ik om dat ook te doen voor de nieuwe generatoren.

Nadat ik alle generatoren tot op hetzelfde niveau had gekregen ben ik het document gaan uitbreiden. Er is tijdens de evaluatie besloten dat het gebruik en de support[[5]](#footnote-5) ook een waarde moest krijgen, zodat deze gebruikt zouden kunnen worden bij het beslissing proces.

Om dit mogelijk te maken heb ik voor het gebruik een scenario bedacht en deze uitgevoerd. Dit scenario is een simpel Master-Detail report waarin gebruik wordt gemaakt van een groot deel van de Must en Should have eisen. Aan de hand van dit scenario was het mogelijk om de gebruiksvriendelijkheid en de eenvoud van de programma’s een beetje in kaart te brengen. Voor de support is er gekeken welke mogelijkheden er waren en hoe deze benut werden om tot een score te komen.

Ten slotte heb ik een tabel gemaakt waarin de eisen gezet werden tegenover de rapportgenerators (Zie Tabel 3 & Bijlage 4 Hfd Grids). Dit maakt het in een oogopslag duidelijk welke generatoren voldeden aan de gestelde eisen. In de conclusie zijn daar ook de waarde van de support, usability en licenties bij opgenomen. Om een totaal waarde toe te kennen aan een generator is er vervolgens per niveau aan eisen een waarde gegeven. Zo heeft Must have een hogere waarde dan de veel lagere Wanna have. Dit is ook gedaan bij de support en usability omdat deze anders niet in proportie zouden staan tegenover de eisen. De support die geleverd wordt is meer van relevatie voor ChipSoft dan voor de klant, daarom heeft deze een lagere waarde toegekend gekregen dan het gebruik, waar de klant wel direct voordeel van heeft.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | | |  | **Telerik** | **Stimulsoft** | **DevExpress** | **SyncFusion** | **MindFusion** | **C1Reports** | **Reportbuilder** | **Infragistics** | **SharpShooter** | **Open Engine** |
| **M** | | **\* 5** | **: 35** | **30** | **35** | **35** | **35** | **25** | **35** | **30** | **25** | **35** | **25** |
| **S** | | **\* 3** | **: 24** | **19.5** | **24** | **24** | **18** | **9** | **21** | **18** | **16.5** | **21** | **7.5** |
| **C** | | **\* 1** | **: 5** | **4** | **5** | **4** | **2** | **3** | **3** | **2** | **2** | **4** | **1** |
| **W** | | **\* 0.5** | **: 1.5** | **1** | **0** | **0** | **0.5** | **1** | **0.5** | **0.5** | **0.5** | **0.5** | **0.5** |
| **Total** | |  | **: 65.5** | **54.5** | **64** | **60** | **55.5** | **38** | **59.5** | **40.5** | **44** | **60.5** | **34** |
|  | | |  |  |  |  |  |  |  |  |  |  |  |
| **Support** | **\* 2** | | **: 10** | **7** | **8** | **8** | **7** | **4** | **6** | **6** | **7** | **6** | **X** |
| **Usability** | **\* 3** | | **: 15** | **12** | **13** | **10.5** | **7.5** | **6** | **10.5** | **9** | **10.5** | **10.5** | **X** |
|  | | |  |  |  |  |  |  |  |  |  |  |  |
| **Totaal** | | | **90.5** | **73.5** | **85** | **81.5** | **70** | **48** | **76** | **65.5** | **61.5** | **77** | **34** |

Tabel 3 Eisen Grid

## Retrospective

Dit gesprek was gericht op het maken van een initiële keuze. Hierbij is vooral gekeken naar de resultaten die weergegeven waren in de conclusie.

Tijdens dit gesprek kwamen de volgende onderwerpen aan de orde. Een daarvan was de Microsoft Report Builder (Zie Bijlage 4 Microsoft Reportbuilder). Omdat ChipSoft een Microsoft partner is wil de directeur graag dat deze generator gebruikt wordt. Ik moest mij dan ook verantwoorden waarom ik dit niet adviseer. Dit kwam er vooral op neer dat het programma Stand-Alone is en het niet mogelijk is om aan te sturen via EZIS tevens is de viewer die hierbij gebruikt wordt alleen te gebruiken in WinForm.

Daarna heeft de opdrachtgever de drie generatoren met de hoogste punten gekozen. Ik bracht toen het onderwerp voor zelf bouw of open source naar voren. Het onderzoeksrapport is alleen gericht op al bestaande producten daarom is deze mogelijkheid daar niet in op genomen (op een open source engine na). Dit gesprek ging ook in op de complexiteit van een rapportgenerator en de benodigde tijd om deze te maken. Om hiervoor een tussenweg te maken is er gekozen om voor de open source engine een GUI te maken en deze uit te breiden tot het product kon voldoen aan de meeste Must mogelijkheden (Zie 11.1).

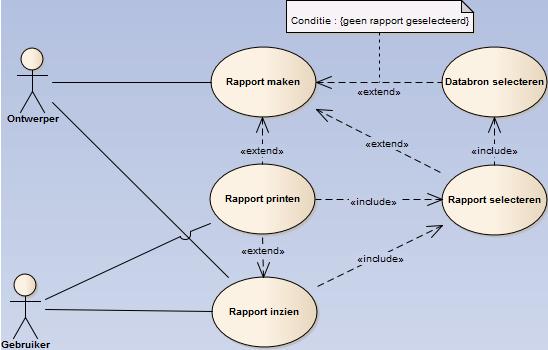
# Sprint 3: Prototype 1

Het programma dat gebruikt werd bij het eerste prototype is de rapportgenerator van StimulSoft. Deze kwam als beste uit het onderzoeksrapport naar voren en was naar mijn mening het startpunt.

Bij het maken van de sprint backlog is er besloten om een algemene use-case te maken waaraan alle prototypes moesten voldoen. Ook wordt er tijdens deze sprint een klasse-diagram gemaakt gericht op een evolutionair prototype. Als laatste is het uitwerken van het prototype.

## Opstellen use-case

Het use-case is opgesteld aan de hand van de eisen van het project. Omdat er bij de prototypes vooral gebruik gemaakt wordt van 3th party programma’s is het overbodig om de eisen die gericht zijn aan de rapportgenerator op te nemen in het diagram.



Figuur 3 Use Case

Ik heb het daarom gehouden bij de generale eigenschappen van de rapportgenerator.

* Het moet mogelijk zijn om een rapport op te stellen.
* Een rapport moet ingezien kunnen worden
* Rapporten kunnen worden uitgeprint.

Omdat het binnen EZIS mogelijk is om het maken en inzien van rapporten te scheiden voor toegangsniveaus is er sprake van meerdere actoren.

Ten slotte is er bij het inzien externe data nodig en bij het ontwerpen is dit optioneel; deze zijn daarom in Excludes en Includes opgenomen. (Zie Figuur 7 & Bijlage 5)

## Maken klasse-diagram

Het doel van het prototype was het functioneren met data uit EZIS, integreren in

EZIS en het converteren van oude rapporten. Omdat deze onderdelen allemaal onafhankelijk van elkaar zijn er verschillende diagrammen gemaakt voor elk onderdeel. Met uitzondering van de conversie diagram; omdat het inlezen van een rapport bij alle prototypes hetzelfde zal zijn wordt dit klasse-diagram gescheiden in een klasse-diagram gericht op het inlezen van een rapport en een apart diagram voor het omzetten van het ingelezen rapport.

Er zijn in totaal vier klasse-diagrammen voor een prototype.

* **Datastructuur**

De datastructuur diagram is bedoeld voor alle prototypes en maakt het mogelijk om de ICS Datastructuur om te zetten naar een DataSet[[6]](#footnote-6) (Zie 5.2.1).

* **Oud Rapport -> Object**

Dit diagram beschrijft hoe een oud rapport wordt ingelezen en opgezet in een iteratief object (Zie 5.2.2).

* **Object -> Rapport**

Omdat elke rapport generator een aparte manier heeft om zijn rapport op te bouwen is het nodig om voor elke generator een implementatie te maken om het ingelezen object om te zetten (Zie 5.2.2).

* **Designer & Viewer**

Deze twee klassen zijn bijna identiek en kunnen eventueel in een diagram weergegeven worden. Ik heb dit echter gescheiden omdat ik op voorhand al wist dat dit niet het geval zal zijn bij de andere prototypes. Om uniform te blijven heb ik ze hier ook al gescheiden (Zie 5.2.3).

### Verwerken datastructuur

Voordat ik een klasse-diagram voor de datastructuur kan maken moet ik meer informatie hebben over Stimulsoft en hoe het functioneert met de ICS datastructuur. In de handleidingen van StimulSoft is te vinden dat er ondersteuning aanwezig is voor een aantal List interfaces van Microsoft[[7]](#footnote-7). Gezien de ICS datastructuur gebruikmaakt van een List interface heb ik het vermoeden gehad dat deze ondersteund zou worden door Stimulsoft.

Het is mogelijk voor StimulSoft om de ICS Collectie uit te lezen. Dit gaat niet foutloos en heeft een tweetal ongewenste effecten.

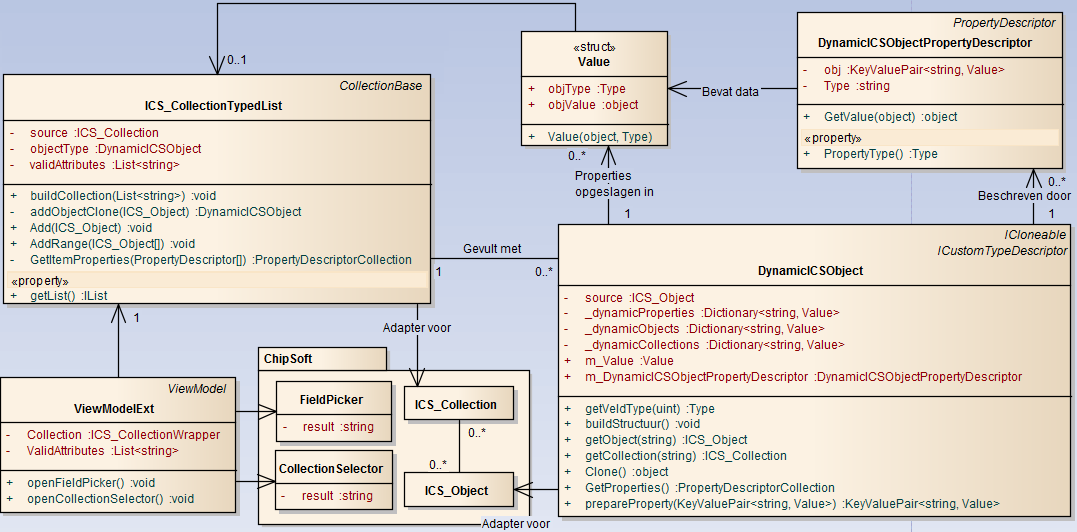
Een probleem doet zich voor als StimulSoft een datasource toegekend krijgt. StimulSoft leest vervolgens alle data uit. Omdat het run-time laden van data een van de belangrijke functies is van de ICS Collectie is dat niet gewenst. Tevens duurt dit proces langer en neemt meer geheugen in beslag naarmate de grootte van de collectie. Daarnaast is er de mogelijkheid aanwezig dat het programma crashed omdat het een op zichzelf terugkerend object tegenkomt.

Het andere probleem is de weergave van de data. In plaats dat de collectie wordt uitgelezen als een gelaagd object wordt elk object / collectie aangezien als een nieuwe databron. Deze databronnen worden vervolgens weergegeven als (ParentName\_Name), als een object meerdere lagen diep wordt dit snel onduidelijk (Zie Figuur 8 & 9).

|  |  |
| --- | --- |
| Out of the box | Gewenst |
| Figuur 4 Out of de Box weergaven van ICS Datastructuur | Figuur 5 Gewenste weergaven |

**Klasse-diagram**

Omdat het direct aansluiten van een ICS Collectie op StimulSoft niet naar wens verloopt, moet er naar een andere oplossing worden gezocht. Hierbij is gebruik gemaakt van een Adapter Pattern[[8]](#footnote-8) om een ICS Collectie om te zetten naar een werkende oplossing.

Omdat properties van een ICS Object kunnen variëren aan de hand van de ICS Logic waaruit zij afkomstig zijn is het niet mogelijk om een vaste structuur te maken voor deze objecten, mijn adapter moet daarom om kunnen gaan met een dynamische structuur. Het eerste idee was om een DataSet te genereren en vervolgens te vullen. Deze methode heeft alleen een aantal nadelen. Zo duurt het vullen van de DataSet langer naarmate er meer objecten in een collectie zitten. Tevens neemt een gevulde DataSet veel geheugen in waardoor dit geen goede oplossing is en vallen de voordelen van een live collectie weg.

Figuur 6 ICustomTypedDescriptor Implementatie

Gezien een dataset geen oplossing bleek te zijn, heb ik vervolgens gekeken naar de mogelijkheid om een eigen datastructuur te maken. Hierbij is gebruik gemaakt van de ICustomTypeDescriptor; deze interface maakt het mogelijk om zelf aan te geven hoe een property van een object wordt uitgelezen (Zie Figuur 10).

Omdat ik verwachte dat de adapter langzaam zou functioneren als de gehele structuur van een collectie wordt opgebouwd heb ik gebruik gemaakt van de Veldenkiezer. Met de Veldenkiezer wordt aangegeven welke objecten moeten worden uitgelezen. Aan de hand van die gegeven wordt de structuur van de adapter gemaakt. Gezien het maken van de structuur voor elk individueel object veel tijd in beslag zou nemen is er gebruik gemaakt van de IClone interface. Met deze interface is het mogelijk om de adapter structuur een keer op te bouwen en deze daarna te kopiëren voor elk object.

Na het implementeren van deze oplossing bleek het erg langzaam te zijn en werd de collectie nog niet weergegeven zoals gewenst (Zie Figuur 9).

**Problemen**

Omdat de snelheid van het inladen mij het meeste kopzorgen bezorgde, hebben mijn begeleider en ik geprobeerd een trace te uit te voeren om te achterhalen waar wij veel geheugen aan kwijt raakte. Dit is gedaan met behulp van de stack tracer en profiler aanwezig in Visual Studio. Uiteindelijk bleken de problemen te komen doordat StimulSoft de data die het binnen krijgt eerst omzet naar een dataset die hij zelf genereerd. Deze wordt dan ook gelijk gevuld met de data. Dit zorgde voor het geheugenverlies.

Omdat het laden van de data zo zwaar was kwam ik op het idee om een collectie van een object aan StimulSoft te geven zodat deze zijn eigen dataset zou genereren. Ik zou dan vervolgens die dataset ophalen en vullen indien nodig.

Dit was uiteindelijk ook niet mogelijk omdat de gegenereerde dataset zoveel verschilde van de geleverde collectie dat het onmogelijk was deze te vullen.

Ik bedacht nog een oplossing en dat was het maken van een eigen dataprovider om StimulSoft te laten denken dat het een connectie had met een database (Zie 10.2.3). Omdat dit een erg technische oplossing was en de sprint afgelopen was heb ik dit idee niet kunnen uitproberen.

### Opzetten conversie

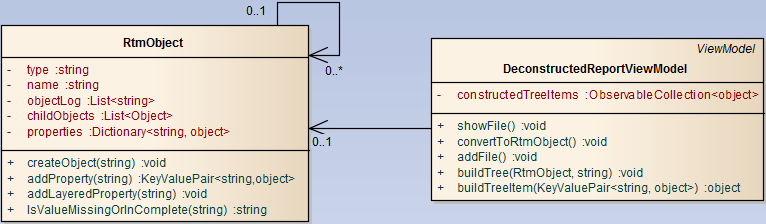
De converter bestaat uit twee delen. Het eerste gedeelte is bedoeld voor het uit lezen van de oude documenten. Het tweede deel moet specifiek gemaakt worden voor elk prototype.

**Oud Rapport -> Object**

De oude bestanden hebben een XML achtige opbouw (Zie Figuur 11) wat het mogelijk maak om deze iteratief uit te lezen (Zie Figuur 13). Omdat het tijdens dit stadia moeilijk te zien is of een document goed is in gelezen heb ik een speciale converter geschreven die dit object omvormt naar een TreeView (Zie Figuur 12). Deze TreeView maakte het vervolgens mogelijk om te kijken of het document goed was ingelezen.

|  |  |
| --- | --- |
| RTM | TreeView |
| Version = '10.04'  mmColumnWidth = 0  DataPipelineName = 'CS\_Artsen'  object ppTitleBand1: TppTitleBand  mmBottomOffset = 0  mmHeight = 14023  mmPrintPosition = 0  object ppLabel1: TppLabel  UserName = 'Label1'  AutoSize = False  end  Figuur 7 RTM | Figuur 8 TreeView |

De bedoeling is dat de TreeView na de conversie de status weergeeft van de geconverteerde bestanden en eventuele meldingen geeft van complicaties en tips. Omdat het inlezen universeel is voor alle prototypes is het de bedoeling dat de deze klasse wordt overgeërfd bij de andere prototypes



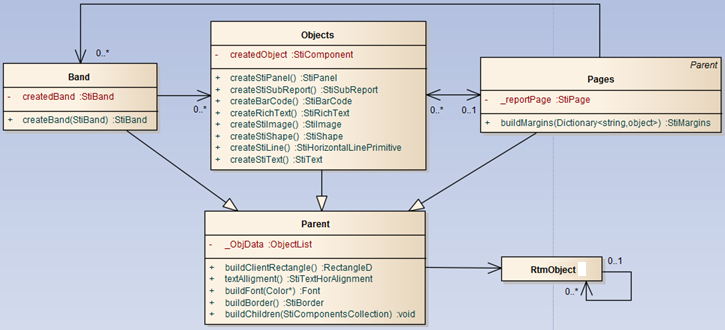
Figuur 9 Converter Core

**Object -> Rapport**

Voordat ik hieraan kon beginnen moest ik eerst uitzoeken hoe een rapport van StimulSoft is opgebouwd. Bij het openen van deze rapporten via applicatie Notepad blijkt dat dit een eigen structuur heeft wat te vergelijken is met XML bestand. Vervolgens geprobeerd zelf een kleine test rapport te schrijven in deze XML structuur om te kijken of dit zou werken. Het test rapport bevatte alleen twee tekst blokken. Dit eindigde met compileerfouten bij het openen van het rapport. Dit kwam door een fout die ik gemaakt had bij het schrijven van mijn document.

Tijdens onderzoek kwam naar boven dat het ook mogelijk was om een rapport te maken via code. Omdat deze manier minder foutgevoelig is dan het maken van het rapport aan de hand van tekst ben ik hier naar overgestapt.

Om een rapport te generen aan de hand van de code zijn er drie hoofdelementen: de Page, Band en Object. Omdat alle drie de elementen dezelfde properties hebben is gebruik gemaakt van overerving (Zie Figuur 14).

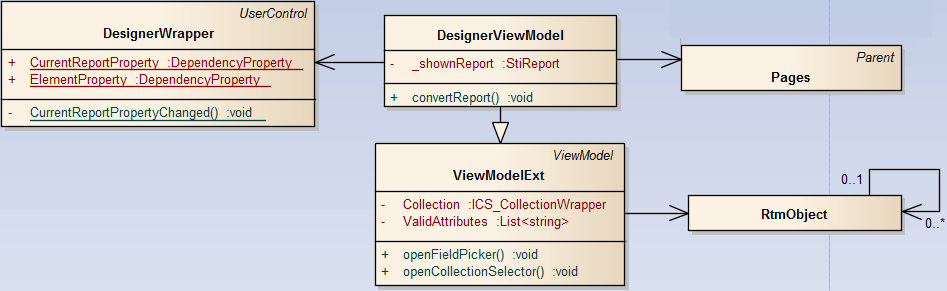


Figuur 10 Converter StimulSoft

### Implementeren StimulSoft

StimulSoft is op het gebied van WPF een beetje vreemd. Ze hebben het product specifiek gemaakt en gericht op WPF maar ze ondersteunen geen embedding. Dit houdt in dat de properties van hun Designer en Viewer geen dependency properties[[9]](#footnote-9) zijn. Dit maakt het dus mogelijk om via de Code Behind de controls aan te spreken maar onmogelijk om de properties aan te spreken via de XAML.

Omdat er bij ChipSoft gebruik gemaakt wordt van MVVM is het niet toegestaan om via de Code Behind te werken en moet er gebruik gemaakt worden van de XAML. Om dit probleem te omzeilen is er gebruik gemaakt van een Adapter Pattern. Deze klasse dient als middelman tussen de ViewModel en de StimulSoft Controls door alle nodige informatie van de ViewModel door te geven naar de controls (Zie Figuur 15).



Figuur 11 StimulSoft designer Implementatie

## Retrospective

Omdat er problemen waren met het verwerken van de datastructuur naar de StimulSoft control heb ik eerst de converter getoond. De opdrachtgever had initieel niet verwacht dat een conversie tool veel over zou kunnen zetten. De opdrachtgever is echter tevreden over de mate waarin dit toch mogelijk was.

Vervolgens kwam de datastructuur in combinatie met de StimulSoft controls. Ik heb tijdens het proberen met het aansluiten van de ICS datastructuur op de StimulSoft control een aantal problemen gehad waarvan opdrachtgever graag wilde weten waar deze vandaan kwamen. Ook al heb ik tijdens deze sprint deze problemen vaak gehad was het voor mij niet meer mogelijk om deze te reproduceren. Deze had ik niet gedocumenteerd en kon de situatie die het veroorzaakte niet her creëren.

Omdat het tijdens deze sprint aan het licht kwam dat het lastig is om de ICS datastructuur aan StimulSoft te koppelen is er aan een deel van het prototype voldaan. Daarom is besloten om in de volgende sprint niet stil te blijven staan bij StimulSoft maar te kijken hoe een andere generator presteert. Het doel is wel om later terug te keren bij StimulSoft om te kijken of de oplossingen van de andere generatoren daar ook werken.

# Sprint 4: Prototype 2

Voor dit prototype is gekozen om gebruik te maken van de DevExpress XtraReports. Deze heeft een gedeelde tweede/derde plek (Zie Tabel 3) en is tevens al in het bezit van ChipSoft.

Omdat er tijdens deze sprint net als in de vorige een prototype wordt gemaakt, wordt een deel van de sprint backlog overgenomen. Omdat de gemaakte Use-Case van algemeen gebruik is, is dit item vervallen en is er meer tijd weggelegd voor de converter.

## Implementeren DevExpress

Bij het implementeren van de DevExpress Designer zijn er een aantal problemen naar voren gekomen. Deze zijn op te splitsen in problemen gerelateerd on WinForm gebruik binnen WPF en aan keuzes gemaakt door DevExpress in het maken van hun applicatie.

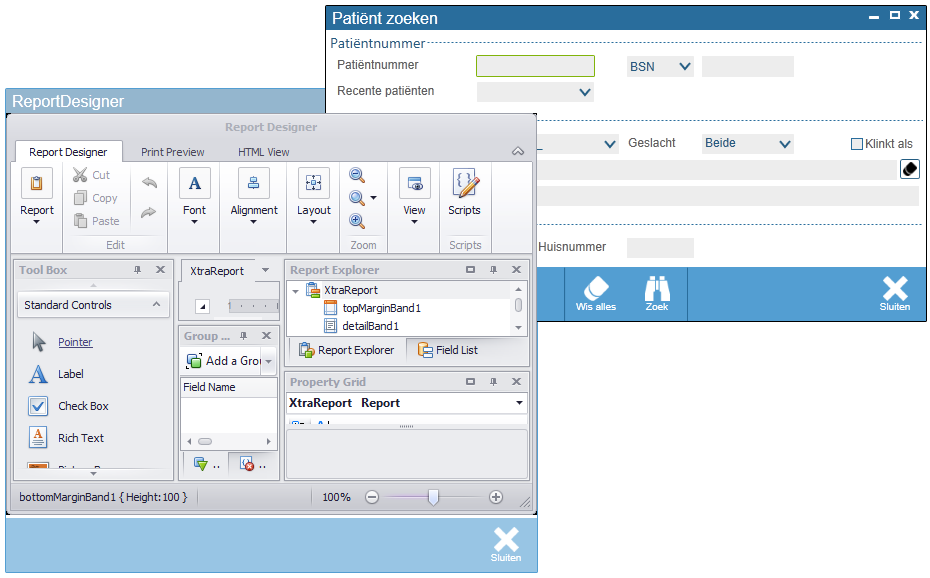
### WinForm problemen

Omdat de designer van deze rapport generator gemaakt is in WinForms moet er een oplossing gevonden worden om deze te hosten in WPF. Voor dit probleem heeft Windows een WindowsFormHost control in WPF. Hierin kan een WinForms control geplaatst worden en dus indirect toch wordt weergegeven in WPF. Ik ben al bekend met deze mogelijkheid en heb gelijk geprobeerd om dit te gebruiken.

Het is alleen niet mogelijk om deze manier te gebruiken. Dit komt omdat een control dat gehost wordt in een WindowsFormHost niet TopLevel mag zijn. Dit houdt in dat een Form geen Parent heeft. Gezien de designer een MDI-Parent[[10]](#footnote-10) is moet deze TopLevel zijn. Er is dus gekeken naar een andere oplossing.

Na onderzoek kwam ik erachter dat de WindowsFormHost de Microsoft implementatie is van de abstracte klasse HwndHost. Met de hulp van het MicroSoft Developer Network (MSDN)[[11]](#footnote-11) heb ik hier een oplossing voor gevonden zodat het integeren wel mogelijk is[[12]](#footnote-12). Dit heeft als voordeel dat ik mijn HwndHost specifiek kon maken voor DevExpress met alle Dependency Properties die nodig waren.

Naast het TopLevel probleem is er een ander nadeel aan WinForms, dat is het Air-Space probleem. Dit wordt veroorzaakt door de WinForm controls die binnen WPF worden gehost. Het probleem doet zich voor als een ander control over de WinForm control wordt geplaatst. Het WinForm control zal altijd boven op de andere controls liggen (Zie figuur 15).

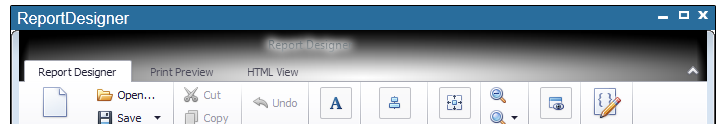


Figuur 12 Air Space probleem

Als Work Around voor dit probleem wordt er een screenshot genomen van de WinForm Vervolgens wordt het WinForm element onzichtbaar gemaakt en wordt het screenshot getoond.

### DevExpress probleem

Door het Air-Space probleem op te lossen bleek er nog een probleem te zijn. De DevExpress control maakt gebruik van een Ribbon[[13]](#footnote-13). Doordat WPF beter werkt met de grafische processen van een PC wordt het uiterlijk van de Ribbon doorzichtig (Zie Figuur 17). Dit heeft te maken met Windows Aero[[14]](#footnote-14). Omdat dit een probleem is dat bij DevExpress ligt heb ik hun support gevraagd naar een oplossing daarvoor. Hun antwoord was dat zij dit nog niet ondersteunden. Ik heb zelf verder gezocht of het mogelijk was om Aero voor een control uit te schakelen. Dit kan via code, alleen wordt Aero dan op de hele PC uitgezet. Bij afsluiten van de control wordt het vervolgens wel weer aangezet.[[15]](#footnote-15)

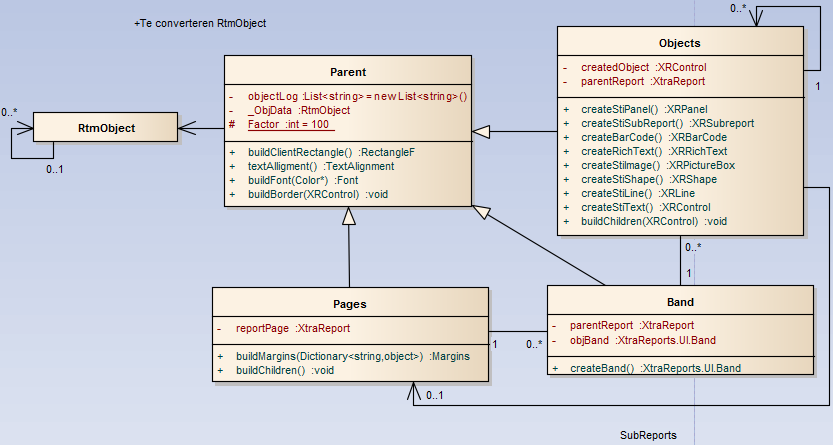


Figuur 13 Glass probleem

Ik vind dit zelf niet de beste oplossing dus ik heb een aantal alternatieven geadviseerd aan ChipSoft. Zo is het ook mogelijk om:

* De control met een oudere bar interface weer te geven, wat minder gebruiksvriendelijk is,
* De control buiten EZIS weer te geven, dan komt dit probleem niet voor,
* De Ribbon te verstoppen en alle controls er van over te nemen in een eigen WPF Ribbon.

## Ombouwen converter

Omdat ik bij StimulSoft heb gezien dat het mogelijk is om via code een rapport op te stellen heb ik eerst gekeken of deze optie ook aanwezig was in DevExpress (Zie 8.2.2). Dit bleek mogelijk te zijn. Ik heb vervolgens uitgezocht hoe de samenhang tussen Pages, Bands en Objects is om zodanig de StimulSoft converter om te kunnen zetten. 

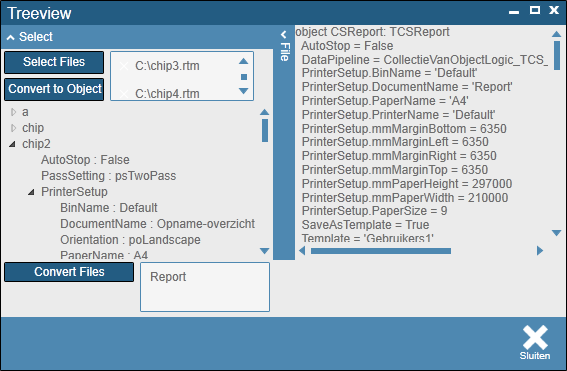
Figuur 14 DevExpress Converter

Zoals in Figuur 18 is te zien, is de structuur iets anders. Zo kunnen objecten niet direct op een pagina weergegeven worden en is het mogelijk om Bands in Bands te plaatsen.

Op dat moment was het mogelijk om een bestand per keer te converteren. Omdat er later ongeveer 2000 rapporten moeten worden omgezet, heb ik het mogelijk gemaakt meerdere tegelijk te converteren.

Daarbij heb ik wel rekening gehouden dat het mogelijk is om naar verschillende types te converteren. Er wordt daarom eerst naar een RtmObject geconverteerd (Zie Figuur 13) en daarna pas naar het juiste type.

Om inzicht te hebben tijdens het converteren heb ik een interface gemaakt (Zie Figuur 19). Met deze interface is het mogelijk om bestanden te selecteren, in te zien en te converteren naar RtmObjecten. Vervolgens wordt er door middel van overerving de Converter en Open functies overschreven om zo ondersteuning te bieden voor de converters van de verschillende prototypes.

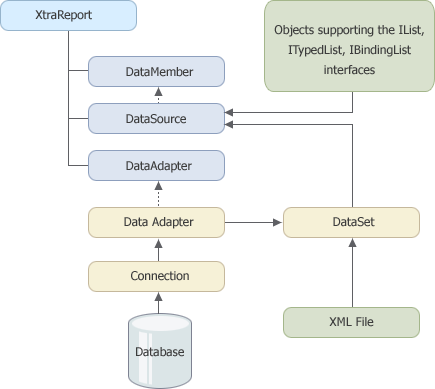


Figuur 15 Het gele control is relatief aan het rapport waarna geconverteerd wordt.

Na het converteren is het mogelijk om files weer te geven met de juiste rapportgenerator om eventuele aanpassingen te maken.

## Omzetten datastructuur

Voordat er een klasse-diagram is gemaakt voor deze sprint is er eerst gekeken of DevExpress om kan gaan met de ICS datastructuur of met mijn eigen implementatie.



Figuur 16 DevExpress Datasources

Ik heb daarbij de documentatie geraadpleegd van DevExpress. In deze handleiding staat een diagram dat weergeeft welke type data gebruikt konden worden door DevExpress (Zie Figuur 20). Gezien de ICS Collectie gebruikmaakt van de IList interface en mijn implementatie van de CollectionBase Interface (Zie Figuur 10 welke gebruikt wordt door alle List Interfaces) had ik het vermoeden dat beiden zouden kunnen functioneren.

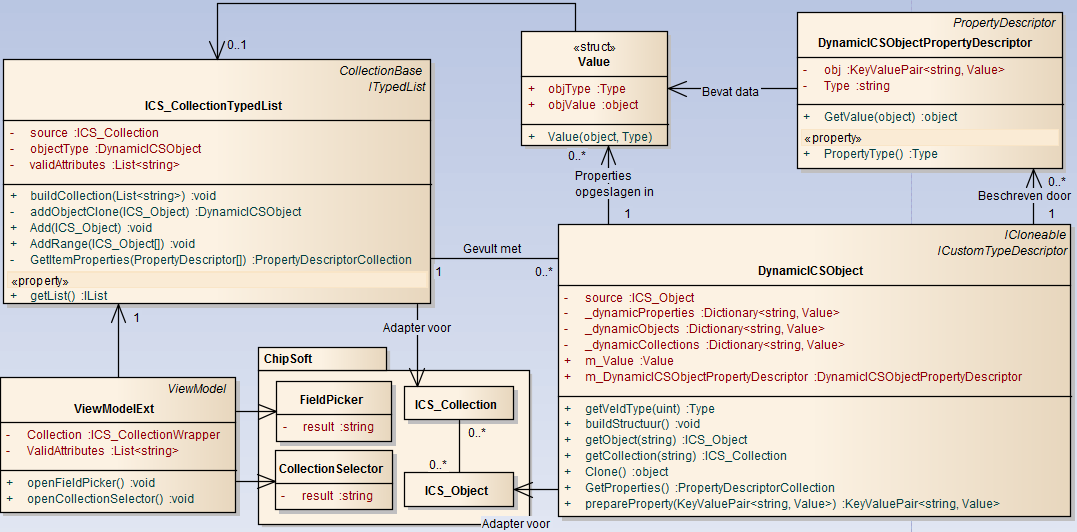
Bij het gebruik van een ICS Collectie bleek dat het wel mogelijk is om deze uit te lezen alleen weet het programma niet hoe het de complexe velden moest uitlezen. Tevens wordt alle data in een keer opgehaald wat juist een voordeel is aan een ICS Live Collectie.

Na het testen met de ICS Collectie heb ik de datastructuur die ik tijdens het eerste prototype heb gemaakt erbij gepakt. DevExpress kan deze wel in lezen, alleen gaf de verkeerde data weer. In plaats van de properties van de ICS Collectie te tonen, worden de properties van de DynamicICSObject (Zie Figuur 10) klasse getoond.

Omdat mijn huidige implementatie niet goed functioneerde, heb ik mij verder verdiept in de interfaces die ondersteund worden door DevExpress. Als eerste heb ik gekeken naar de ITypedList; de naam van deze interface deed mij vermoeden dat er een relatie was met de ICustomTypeDescriptor (Zie Figuur 10) die tijdens het eerste prototype is gebruikt.

De ITypedList wordt op MSDN omschreven als “Provides functionality to discover the schema for a bindable list, where the properties available for binding differ from the public properties of the object to bind to.” In het kort houdt dit in dat het gebruik van deze interface het mogelijk maakt om te specificeren welke properties getoond moeten worden. Omdat ik een probleem heb met het weergeven van de goede properties is dit de meest geschikte interface om mijn probleem op te lossen (Zie Figuur 21).

De IBindingList bleek een interface te zijn waarmee het mogelijk is om objecten binnen een list af te schermen door editing aan of uit te schakelen. En de IList is de standaard list interface die al eerdere beschreven is (Zie 8.2.1).



Figuur 17 ITypedList Implementatie

Naast het maken van de nieuwe structuur heb ik ook gekeken naar de mogelijkheid om data later in te laden. Mijn eerste idee was om data pas in te laden als de preview werd weergegeven. Ik zocht toen naar een Event dat afgaat als er naar de preview window wordt overgegaan. Tijdens dat zoeken kwam ik de DataRowChanged Event tegen. Dit event gaat af elke keer als een object uit een collectie wordt gelezen. Met dit event heb ik er voor gezorgd dat alleen data die nodig is wordt ingeladen.

Dit zorgt er wel voor wel dat het opbouwen van de preview langer duurde maar er wordt minder geheugen gebruikt en het systeem werkt daardoor vloeiender.

## Retrospective

Tijdens de retrospective is er vooral gesproken over de designer en de datastructuur. De opdrachtgever had al verwacht dat de convert ook bij dit product zou functioneren.

Er is gesproken over de WinForm problemen. Wat de problemen waren hoe het opgelost is en eventuele oplossingen die er waren (Zie 9.1.2).Op dat moment maakte het nog niet uit omdat dit nog verwerkt moest worden in het advies rapport. Na het advies rapport wordt dit weer besproken.

We hebben het langst gesproken over de datastructuur. Wij zijn over het algemeen wel tevreden over het laden van data, zo ook over de optie om het laden stop te zetten indien gewenst. Wat ook ter sprake kwam was de Veldenkiezer die ik gebruikte om collecties verder te openen. Dit zou mensen veel tijd kosten om steeds dat scherm te openen om collecties of objecten te openen. Daarnaast is dit lastig zijn als een rapport geprint moet worden omdat dan eerst de velden geselecteerd moeten worden.

Uiteindelijk is er besloten om voor de volgende sprint te kijken naar het doorlopen van een collectie zonder de Veldenkiezer maar wel met de opbouw van de Veldenkiezer, hoe de verschillende prototypes omgaan met de enumarator en een nieuw prototype maken van SharpShooterReports.

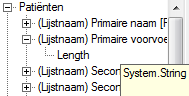
# Sprint 5: Prototype 3

De vijfde sprint zal zich richten op de laatst gekozen rapport generator SharpShooterReports. Daarnaast zal er gekeken worden naar het gebruik van de Enumarator bij de verschillende producten en wordt ITypedList veranderd zodat deze werkt zonder Field Picker.

## Implementeren SharpShooterReports

Omdat de designer van SharpShooter gemaakt is in WinForm heb ik eerst geprobeerd om deze te hosten met een WindowsFormHost. Net als bij DevExpress lukte dit niet en heb ik mijn eigen implementatie gebruikt om het werkend te krijgen (Zie 9.1).

Bij het testen van de geïntegreerde designer bleek dat de deze niet meer functioneerde. De Ribbon bleek volledig inactief op een aantal dropdown boxes na (Zie Figuur 23). Gezien er ook een versie is zonder ribbon heb ik ook deze geprobeerd te hosten. Ook deze deed het niet goed. Als er een control op het rapport wordt geplaatst, wordt het rapport niet gerefreshed en lijkt het alsof er niks gebeurd. Pas als je het scherm handmatig refreshed is de control zichtbaar.De enige oplossing die ik kon bedenken was SharpShooter buiten het EZIS weergeven.



Na de integratie van de designer heb ik geprobeerd direct een ICS collectie op te geven aan SharpShooter. En als enige Rapport generator leest deze de structuur goed uit. Op een paar uitzonderingen na. Zo worden properties van het type string ook als type gezien met als extra waarde de lengte en worden afbeeldingen niet goed uitgelezen (Zie Figuur 22).

Figuur 18 Objectachtige String

Om te kijken of een collectie goed word uitgelezen heb ik een test rapport gemaakt en probeerde ik deze weer te geven in een preview. SharpShooter host zijn preview niet binnen het programma en start zijn eigen previewer.

Ik heb toen het preview event onderschept en mijn eigen event geschreven. In dit zelf geschreven event zorgde ik ervoor dat de data in geladen wordt en wordt het rapport vervolgens naar de WPF Viewer door gestuurd.

Bij het proberen om een document door te sturen naar de WPF Viewer kwam er naar voren dat deze een path wilde naar een fysiek document voordat deze weergegeven kon worden. Wat niet mogelijk is als er met een business object wordt gewerkt.

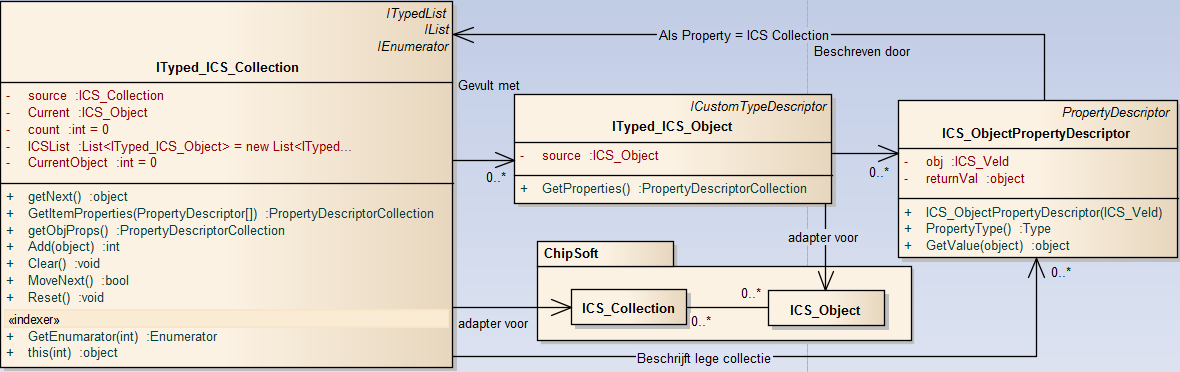
De enige oplossing was de WinForm Viewer te gebruiken wat ervoor zorgt dat er niet wordt voldaan aan een Must have eis.

|  |  |
| --- | --- |
| Embedded | Buiten |
| Figuur 19 Inactief | Figuur 20 Actief |

## Updaten oude prototypes

### Updaten ITypedList (DevExpress)

Het doel van het updaten van de ITypedList was het proberen om de Veldenkiezer overbodig te maken. Om dit te kunnen doen heb ik mij eerst verder verdiept in de GetProperties functie van de ITypedList. Bij het testen van mijn datastructuur kwam ik erachter dat deze functie pas afgevuurd wordt als er een complex veld in een TreeView wordt geopend. Dit zou betekenen dat de structuur nog niet bekend was voordat het object geopend is, hetgeen waar de Veldenkiezer juist voor gebruikt werd.



Figuur 21 Final Datastructuur

Omdat de Veldenkiezer bij de ITypedList datastructuur overbodig bleek te zijn heb ik opnieuw gekeken naar de datastructuur en deze opnieuw ontworpen (Zie Figuur 25).

Omdat het niet nodig is om de structuur van tevoren vast te leggen is het niet is de IClone interface overbodig geworden. Ook is het niet meer nodig om properties vast te leggen.

### Uitzoeken oplossing Aero effect (DevExpress)

De opdrachtgever heeft tijdens de retrospective gevraagd of het mogelijk was om een panel achter de DevExpress reporter te zetten om zo het doorschijnen van de WPF te stoppen. Ik kon niet zeggen of dit zou werken dus heb ik dit geprobeerd. Net als bij de WindowsFormHost kreeg ik hierbij TopLevel errors. Ik heb toen met de object browser de designer onderzocht en vond ergens een property ‘AllowFormGlass’ verstopt. Met het gebruik van deze property is het probleem gelijk opgelost.

### opzetten dataprovider (StimulSoft)

Het idee om een dataprovider te maken heb ik gehad aan het einde van de Stimulsoft prototype (Zie 8.2.1). Omdat ik hier toch oude prototypes ging updaten leek dit mij een goed idee. De bedoeling was om Rapport generatoren te laten denken dat zij een connectie hadden met een database. Dit zou ervoor zorgen dat data niet in een keer wordt ingelezen.

Dit zou eventueel bij alle projecten gebruikt kunnen worden en zelfs bij de Windows Reportbuilder 3.0; welke afgekeurd is omdat het geen connectie kan maken met de ICS datastructuur. Ik heb op het MSDN en op het internet gezocht hoe dit gemaakt moet worden en kwam uiteindelijk een tutorial van Microsoft tegen. Dit is een ingewikkelde tutorial met update, deletes transactions en alle andere opties van een database maar gezien ik alleen data moest lezen werd dit versimpeld. Ik heb als eerst geprobeerd met alleen het eerste niveau van een collectie om te kijken of het inderdaad mogelijk zou zijn.

Bij het testen met DevExpress werkt mijn dataprovider goed. Toen ik vervolgens een connectie wilde maken met StimulSoft kwam ik erachter dat zij hun eigen data adapters schrijven en mijn implementatie niet werkt. Ik heb vervolgens geprobeerd om deze interface te gebruiken om een connectie te maken. Dit lukte deels. ik kon een connectie maken met de ICS datastructuur en het eerste niveau uitlezen. Het lukte alleen niet om de data door te sturen.

### Onderzoeken problemen enumerator

Om te kijken hoe de verschillende rapportgenerators omgingen met de een Enumerator heb ik een lege implementatie van een IList gebruikt om te kijken welke methodes werden aangeroepen. Hieruit kwam naar voren dat DevExpress gebruik maakt van een For Loop, StimulSoft van een For Each Loop en SharpShooter beide ondersteund.

Ik heb vervolgens besloten om een document met mijn bevindingen van de verschillende rapport generatoren weer te geven. Hierbij heb ik gekeken naar de mogelijkheid om geïmplementeerd te worden in WPF, hoe het aansluit op de datastructuur en hoe makkelijk of moeilijk het is om de data aan te geven. Dit document moet later nog uitgebreid worden met de informatie over de performance van de verschillende rapport generatoren. Het leek mij in dit stadia niet verstandig, omdat ik de oude prototypes nog moest updaten of nakijken.

## Updaten converters

Net zoals bij de andere prototypes is er een converter geschreven voor SharpShooter. Ook bij deze is het wederom mogelijk om met code een rapport in elkaar te zetten en is er gebruik gemaakt van de vorige prototypes.

Bij het testen van de converters kwam er een probleem naar voren. In de huidige rapporten worden de objecten van een rapport gebonden aan de ‘Name’ van een property. Dit komt omdat de huidige generator de Name gebruikt om de TreeView op te bouwen. Omdat ik de opbouw van de Veldenkiezer gebruik ( DiplayName [Name]) komt dit niet overeen. Dit houdt in dat de databinds in het nieuwe rapport nooit overeen zouden kunnen komen.

Om dit probleem op te lossen heb ik een tweetal oplossingen bedacht.

1. De eerste is afzien van de Veldenkiezer opbouw en alleen het ‘Name’ veld weergeven. Is minder duidelijk voor de gebruik, maar aangezien dat nu al wordt gebruikt niet een groot probleem.
2. De tweede optie is om na een conversie een rapport als ‘Dirty’ op te slaan in de database. Zodra een ‘Dirty’ rapport wordt geopend, gaat de conversie verder. Er wordt dan gebruikgemaakt van de data die toegekend wordt aan een rapport om alle databindings opnieuw te definiëren.

## Retrospective

* **SharpShooter**

Niemand is tevreden met SharpShooter. Dat het niet mogelijk is om het in EZIS te hosten was een groot punt wat de opdrachtgever niet beviel. Tevens is iedereen van mening dat het objectachtige gedrag van string type velden erg verwarrend is.

De SharpShooter converter is kort ter sprake gekomen omdat er vanuit werd gegaan dat dit geen probleem zou zijn.

* **Datastructuur**

De ITypedList zorgt nu dat het gewenste effect bereikt wordt. De opdrachtgever was hier dan ook tevreden mee.

Het idee van de dataprovider vindt hij ook een goede keuze. Hij vond het jammer dat dit nu de oplossing was voor StimulSoft.

* **Vervolg**

Als laatst hebben we gesproken over het vervolg van de opdracht. Wat ik nog van plan was en hoeveel tijd er nog was tot het einde van het project.

Omdat de directeur graag wil dat de ReportBuilder 3.0 gebruikt wordt, wilde de opdrachtgever dat ik nog een keer naar de dataprovider zou kijken en dan specifiek gericht naar de ReportBuilder 3.0. Dit om aan te kunnen tonen dat er een goed onderbouwde keuze is om niet of wel de ReportBuilder 3.0 te gebruiken.

Ook is er besloten om het laatste prototype te maken. Dit wordt gedaan met de Open Source Engine.

# Sprint 6: Prototype 4

De zesde sprint is gericht op het uitzoeken of het zelf maken van een rapportgenerator een optie zou zijn voor ChipSoft. Het tweede doel is het uitzoeken of Microsoft Reportbuilder 3.0 gebruikt zou kunnen worden. Als laatst is er voor elk prototype een performance test uitgevoerd.

## Zelf maken rapportgenerator

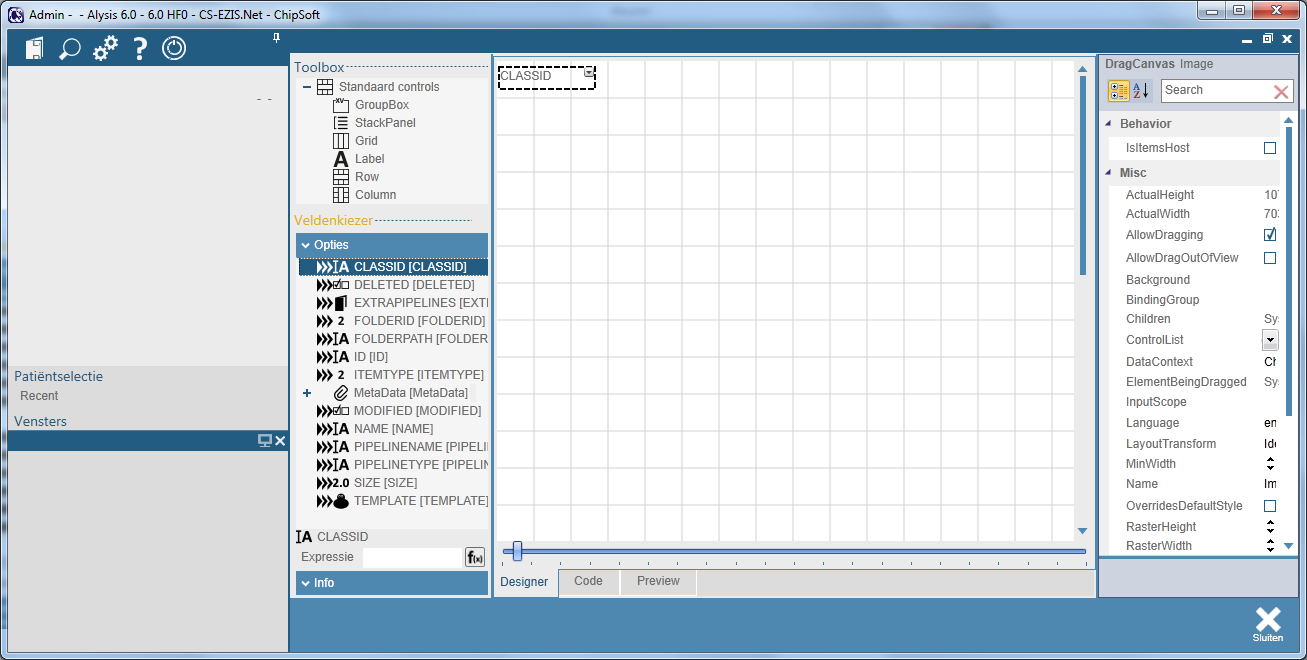
Het was tijdens het retrospective van de tweede sprint gepland om te kijken of het zelf maken van een rapportgenerator een mogelijk optie zou zijn (Zie 7.2). Omdat het van de grond af opbouwen van een rapport generator te veel tijd zou innemen is er gekozen om van de gevonden open source engine gebruik te maken. Het zelf maken was daarom meer gericht op het maken van de GUI dan van de rapportgenerator zelf.

### Ontwerp GUI

De layout van schermen in EZIS staat niet vast en kunnen aangepast worden. Dit wordt gedaan aan de hand van een Designer die in EZIS is verwerkt. Omdat er al een ontwerptool aanwezig is, heb ik eerst gekeken of het mogelijk was om deze aan te passen om te werken als rapport generator.

Het aanpassen van de EZIS Designer bleek geen goede oplossing. Deze designer maakt gebruik van een tabel systeem waarbij rijen en kolommen moeten worden toegevoegd om daar elementen in te plaatsten. Wat ik gezien heb bij het maken van de andere prototypes is dat rapportgeneratoren een absoluut systeem gebruiken waarbij de coördinaten van elementen worden vastgelegd.

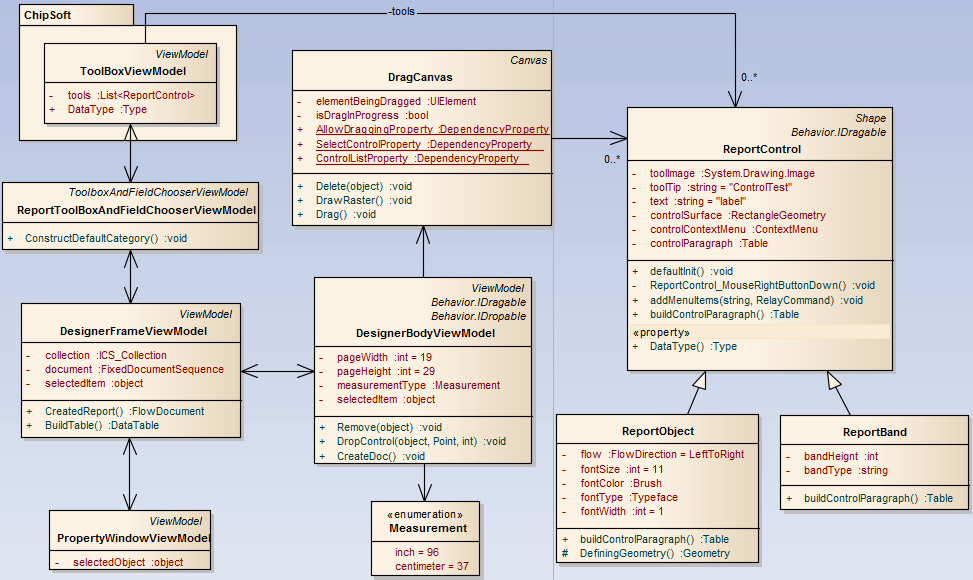
Ik vind het zelf onlogisch om een tabel systeem te gebruiken omdat hierbij meer handelingen nodig zouden zijn om een rapport te maken. Daarom heb ik alleen de layout gebruikt van de EZIS designer bij het maken van mijn eigen GUI (Zie Figuur 26).



Figuur 22 Eigen Rapport generator

### Maken klasse-diagram

Ik wilde de interface zo maken dat het mogelijk zou zijn om de verschillende onderdelen te verkleinen of uit te zetten. Om deze reden is er een frame waarin de layout wordt gespecifieerd en verwezen wordt naar de andere onderdelen (Zie Figuur 27).



Figuur 23 Ontwerp eigen generator

* **DesignerFrame**

Deze klasse bepaalt de layout van de designer en genereert het rapport aan de hand van het ontwerp.

* **DesignerBody**

In de body worden dimensies van een rapport vastgesteld. Ook wordt de visuele data van het ontwerp bijgehouden.

* **Canvas**

Het canvas maakt gebruik van de WPF canvas en voegt daar de mogelijkheid om de slepen aan toe. Dit wordt gebruikt om een rapport te ontwerpen door middel van de ReportControls

* **ReportControl**

Dit zijn de elementen die geplaatst kunnen worden op het rapport.

* **ToolBoxAndFieldChooser**

Deze klasse wordt gebruik door de EZIS designer om de tools weer te geven en de databron. Omdat ik de EZIS designer layout als basis gebruikte heb ik hiervan gebruikgemaakt. Door middel van overerving zijn de huidige controls overgeschreven met mijn ReportControls.

## Onderzoek Microsoft Reportbuilder 3.0

De Reportbuilder van Microsoft wordt op het moment al op twee gebieden binnen EZIS gebruikt. De directie zou het daarom prettig vinden als deze ook gebruikt kan worden voor de rapportgenerator. Om deze reden wil mijn opdrachtgever dat ik kan aantonen dat het geen goede oplossing is, maar als het toch gekozen wordt dat zij het wel kunnen implementeren.

Ik heb al eerder gekeken naar de mogelijkheid om gebruik te maken van een dataprovider om zo een product te laten denken dat het aan een database hangt (zie 9.2.3). Omdat een dataprovider binnen een programma niet geregisteerd is op de computer kan deze niet gevonden worden door externe programma’s.

Ik heb daarom gekeken hoe connectie gemaakt kan worden met databases. De oplossing hiervan is een OLE-DB adapter. Er moet een adapter geschreven worden en deze wordt vervolgens in het registeren van de computer opgeslagen, dit maakt het vervolgens mogelijk voor programma’s om het te gebruiken.

## Performance testen

Voor het uitvoeren van de performance test zijn er eerst een aantal scenario’s bedacht waaraan deze test moet voldoen. Bij deze scenario’s is gekeken naar de bekende klachten over de huidige rapport generator.

Omdat geheugengebruik op het moment een probleem is bij grote hoeveelheden data, is de grote van een collectie data belangrijk. De tests worden daarom allemaal vier keer uitgevoerd elke keer met een grotere collectie aan gegevens.

Gezien er meerdere Logic’s zijn binnen het EZIS systeem is er ook gekeken of de prototypes problemen hadden met verschillende collecties. Er is daarom gebruik gemaakt van twee aparte collecties, de lijst met Patiënten, 60.000+ records, en de lijst met Artsen, 10.000+ records.

Ook is er gekeken naar de snelheid waarmee het programma werkt op verschillende gebieden zoals het opstarten, genereren van een rapport en printen.

Om te zorgen dat alle tests goed uitgevoerd zouden worden bij alle prototypes is er een rapport layout gebruikt.

### Uitvoeren tests

Voor het uitvoeren van de geheugen test is gebruik gemaakt van het programma .Net Memory Profiler 4.0. Dit programma wordt binnen ChipSoft gebruikt om memory leaks op te zoeken en geheugen gebruik binnen EZIS in te zien (Zie Tabel 4).

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Geheugen (KBytes) | | |  | | | |
| **# Artsen** | | **Designer** | **Preview** | **Image** | **Print** | **Export** | |
| **0-98** | **Totaal** | **90.058 kb** | **25.476 kb** | **16.456 kb** | **3.907 kb** | **5.968 kb** | |
| **0-998** | **Totaal** | **90.058 kb** | **31.392 kb** | **25.864 kb** | **6.243 kb** | **7.457 kb** | |
| **0-9998** | **Totaal** | **90.058 kb** | **84.071 kb** | **96.010 kb** | **23.340 kb** | **25.911 kb** | |
| **# Patient** | |  |  |  |  |  | |
| **0-19998** | **Totaal** | **90.058 kb** | **61.912 kb** | **139.313 kb** | **25.736 kb** | **22.414 kb** | |

Tabel 4 DevExpress Geheugen Test

Voor de snelheid is er eerst gebruik gemaakt van de Profiler binnen Visual Studio. Deze Profiler maakt het mogelijk om te zien hoeveel tijd een bepaalde functie actief is geweest. Omdat veel van de relevante functies binnen de 3th party programma’s gebeurde, is dit geen goede oplossing. Er is toen gebruik gemaakt van Break Point [[16]](#footnote-16)op de start en eind events van de gewenste functies. Deze Break Point printte de tijd waarop zij gepasseerd waren in de console; hiermee is het tijdverschil berekend (Zie Tabel 5).

Tabel 5 DevExpress Snelheid Test

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Tijd (seconden) | |  | | |
| **# Artsen** | **Designer** | **Preview** | **Print** | **Export** |
| **0-98** | 1,04 | 3,67 | 14,12 | 13,21 |
| **0-998** | 1,04 | 17,15 | 18,71 | 12,64 |
| **0-9998** | 1,04 | 243,29 | 21,13 | 19,74 |
| **# Patient** |  |  |  |  |
| **0-19998** | 1,04 | 167,22 | 20,12 | 28,73 |

De resultaten van de performance test worden in het adviesrapport opgenomen. Gezien geheugen een probleem was bij de huidige rapportgenerator spelen deze resultaten een belangrijke rol in de eind beslising. (Zie Bijlage 6 voor alle test uitslagen)

## Retrospective

* **Eigen ontworpen rapport generator**

Het eerste onderwerp tijdens deze bespreking was de GUI. Omdat de EZIS Designer werkt aan de hand van een grid systeem heb ik mijn keuze moeten verantwoorden. De opdrachtgever is zelf van mening dat een grid systeem een strakker rapport zou kunnen maken. De expert bij mijn project is het met mij eens, er was dus een gespleten mening over dit onderwerp.

Het tweede dat ter sprake kwam was de rapport engine. Deze voldoet absoluut niet aan de eisen van het project. Er is daarom gediscussieerd over de mogelijkheid om zelf een rapport engine te bouwen. Hieruit kwam naar voren dat het te veel tijd en geld zou kosten voor ChipSoft of dit te doen tegenover het kopen van een product en deze zo nodig aan te passen.

* **Reportbuilder 3.0**

Aan de hand van het opgeleverde document zijn we het er allemaal mee eens dat dit geen goede oplossing is. De OLE-DB is het lastig maken voor een Client-Server omgeving en ook het opslaan en inzien van gemaakte rapporten wordt niet goed ondersteund.

* **Performance test**

De resultaten die uit de performance test kwamen, waren al te zien tijdens het gebruik van de prototypes. Mijn opdrachtgever is tevreden met de test. Er is mij gevraagd hoe ik de data heb vergaard en welke waardes ik daarbij heb gebruikt.

# Sprint 7: Adviesrapport

De laatste sprint is gebruikt om het uiteindelijke adviesrapport op te stellen. Hierbij is gebruik gemaakt van alle documenten die gemaakt zijn tijdens de loop van dit project. Omdat dit document niet de gehele sprint zou kunnen vullen, is er ook besloten om de converter verder uit te breiden.

## Opstellen adviesrapport

Het advies rapport is opgesteld aan de hand van de drie fases die tijdens de planning zijn beschreven. Bij elke fase wordt beschreven welke handelingen gedaan zijn en de resultaten daarvan.

De fase die het meest uitgelicht wordt in dit rapport is de prototype fase. Vooraf wordt beschreven waaraan een prototype moet voldoen en hoe er gecontroleerd is of er aan deze eisen wordt voldaan (Zie Bijlage 6 Prototypes).

Vervolgens wordt er per prototype beschreven hoe het omging met EZIS en welke problemen geconstateerd zijn. Ook wordt er een kleine toelichting geven over de “complexiteit” van het gebruik van de prototypes/3th party programma’s. Als laatst wordt er een score gegeven aan de prototype. Bij deze score wordt gekeken naar de performance test, integratie en het gebruik van EZIS data (Zie Tabel 4).

Na de prototypes wordt er nog uitgelegd waarom ik zelfbouw uitsloot en de Microsoft Reportbuilder niet adviseer. Als laatst volgt het advies. Voor het advies heb ik gebruik gemaakt van de tabel die gebruikt is tijdens het initiële rapport en deze aangevuld met de score van de prototypes.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | |  | **DevExpress** | **Stimulsoft** | **SharpShooter** |
| **Must** | **\* 5** | **: 35** | **35** | **35** | **35** |
| **Should** | **\* 3** | **: 24** | **24** | **24** | **21** |
| **Could** | **\* 1** | **: 5** | **4** | **5** | **4** |
| **Would** | **\* 0,5** | **: 1,5** | **0** | **0** | **0,5** |
|  |  |  |  |  |  |
| **Support** | **\* 2** | **: 10** | **8** | **8** | **6** |
| **Usability** | **\* 3** | **: 15** | **10,5** | **13** | **10,5** |
|  |  |  |  |  |  |
| **WPF** | **\* 3** | **: 15** | **9** | **15** | **3** |
| **Geheugen** | **\* 3** | **: 15** | **15** | **3** | **12** |
| **Snelheid** | **\* 3** | **: 15** | **15** | **6** | **6** |
| **Datastructuur** | **\* 3** | **: 15** | **12** | **6** | **15** |
|  |  |  |  |  |  |
| **Totaal** |  | **: 150,5** | **132,5** | **115** | **113** |

Tabel 6 Advies

Net zoals bij de Support en Usability is een waarde gegeven aan de prototype score om te zorgen dat deze kan opboxen tegen de MoSCoW.

Na het geven van mijn advies heb ik hier nog wat meer informatie over gegeven. Mijn ideeën over implementatie mogelijkheden, bijvoorbeeld het zelf ontwerpen van een GUI om een

EZIS feel te geven en het Air-Space te omzeilen.

Ook geef ik informatie over de Adapter die ik gemaakt heb om samenwerken met EZIS mogelijk te maken, hoe deze geïntegreerd kan worden en bekende problemen.

Als laatst vertel ik iets over de mogelijkheid van het converteren van oude rapporten. Hierbij wordt ook weer vermeld welk probleem wordt veroorzaakt als de “Veldenkiezer” opbouw wordt gebruikt en hoe deze eventueel is op te lossen.

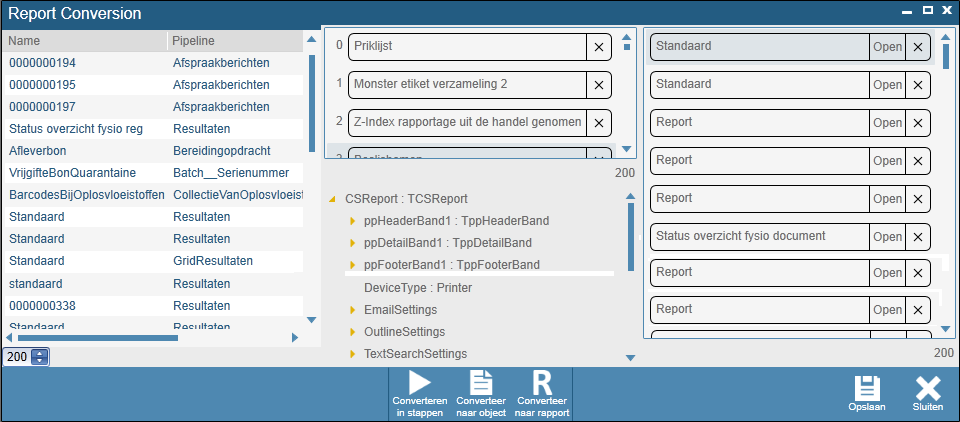
## Converter

Het is bij met de converter op dit moment alleen mogelijk om rapporten te converteren als deze geselecteerd werden als losse file van de computer. Omdat de rapporten in een database staan, wordt de nieuwe converter geschreven die wel rapporten uit de database gebruikt.

De layout van de GUI is ook aangepast voor de nieuwe converter. Het is mogelijk om de layout aan te passen door de gebruiker.

* **Driedelig**

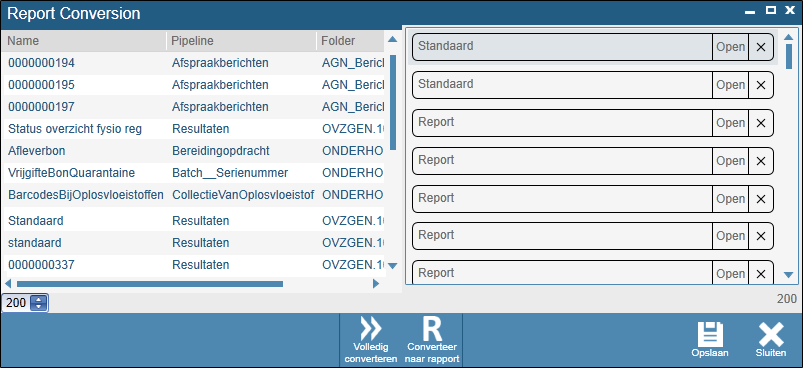
De driedelige layout geeft meer inzicht in het converteer proces. Bij deze methode wordt het rapport eerst naar een object omgezet en kan deze ingezien worden. Als er te veel fouten voor komen kan er gekozen worden om bepaalde objecten te verwijderen. Het is vervolgens mogelijk om de ingelezen rapporten om te zetten naar de nieuwe rapport structuur

(Zie Figuur 28).

Figuur 24 Driedelige layout

* **Tweedelig**

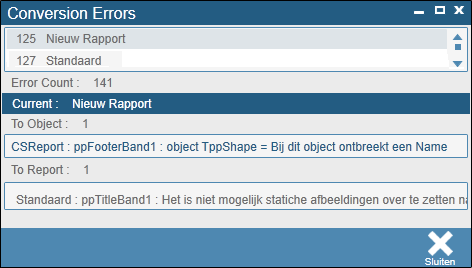
De tweedelige layout zet een oud rapport in een keer om in een nieuw rapport. Ook na het converteren naar een rapport kunnen deze nagekeken worden en eventueel verwijderd worden (Zie Figuur 29).



Figuur 25 Tweedelige layout

Na elke conversie (RTM naar RtmObject, RtmObject naar rapport of RTM naar rapport) wordt er een er een error dialog weergegeven (Zie Figuur 30). Hierin is de naam van het rapport te zien met daarbij de vastgestelde errors. De errors wordt gesplitst op problemen met inlezen van een rapport en het omzetten naar de nieuwe rapport formaat.

Deze geconstateerde problemen worden ook weergegeven in de layout als er een rapport met een error geselecteerd wordt.

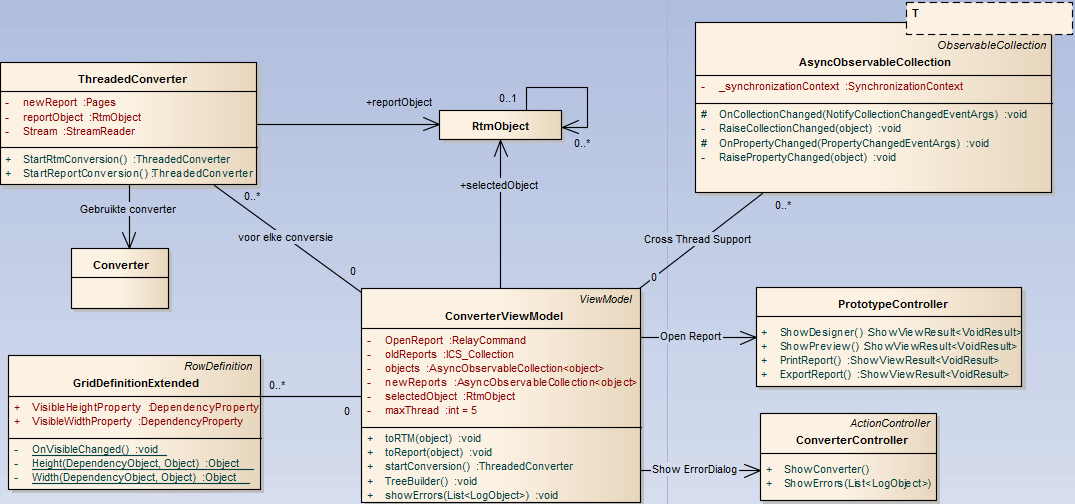


Figuur 26 Error Dialog

### Maken klasse-diagram

In een ziekenhuis kan een groot aantal rapport layouts aanwezig zijn. In de test omgeving wordt al gebruik gemaakt van 2.784 rapporten.

Hoe complexer een rapport des te langer het duurt om deze te converteren. Dit proces zou daarom veel tijd in beslag kunnen nemen. Als enige mogelijkheid die ik zag om dit proces sneller uit te kunnen voeren was om gebruik te maken van meerdere Threads (ik ging er van uit dat een PC in een ziekenhuis beschikt over meerdere CPU Core’s).

Om te kunnen converteren met meerdere Threads zijn er twee belangrijke klassen toegevoegd. Er is voor de view een klasse gemaakt deze is meer voor het uiterlijk gemaakt (Zie Figuur 31).

Figuur 27 Ezis Converter

* **ThreadedConverter**

Deze klasse wordt geïnitieerd als er een thread wordt opgestart.Hij voert vervolgens de gewenste conversie type uit en wordt gedisposed. Net als bij de andere converters is het rapport waarnaar geconverteerd wordt variable.

* **AsyncObservableCollection**

Een ObservableCollection is een list die gebruik wordt in WPF & MVVM. Als een object aan een normale List wordt toegevoegd wordt de View niet geupdate. Bij een ObservableCollection gebeurt dit wel. De AsyncObservableCollection is nodig omdat objecten aan deze lijst worden toegevoegd vanuit andere Threads dit is niet mogelijk bij een normale ObservaleCollection.

* **GridDefinitionExtended**

Het is niet mogelijk om een row of column in een grid onzichtbaar te maken. Deze

klasse voegt deze mogelijk daaraan toe.

## Retrospective

* **Adviesrapport**

Het was voordat ik het advies rapport had geschreven al duidelijk welk prototype het beste had gefunctioneerd. Er is daarom niet veel discussie geweest over mijn advies. Enige opmerking die naar voren kwam waren de namen van bepaalde hoofdstukken die aangepast moesten worden en dat er een Managementsamenvatting aan moest worden toegevoegd.

* **Converter**

De opdrachtgever is tevreden met de laatste versie van de converter. De error weergave geeft meer inzicht in het proces en de snelheid was een plus punt.

# Evaluatie

## Sprint 1: Evaluatie

### Productevaluatie

Voor deze sprint heb ik uiteindelijk twee producten opgeleverd het oriëntatie document en deel een van het onderzoeksrapport.

* **Oriëntatie document**

Over het eindresultaat ben ik zelf tevreden. Ik ben van mening dat het deel over de huidige situatie een goede representatie is van hoe de reportbuilder nu gebruikt wordt. Het document gaf mij ook een beeld waarom de reportbuilder vervangen moest worden. Het klasse-diagram had ik alleen anders in het document moeten plaatsten; daar word je niet veel wijzer van zoals het nu is, opsplitsen in kleinere diagrammen had een betere optie geweest.

* **Onderzoeksrapport**

Dit document was een opzet naar het uiteindelijke verslag en dat was ook te merken. Ik had geen goed doel genomen om de usability te kunnen meten van de individuele producten. Dit heb ik in de tweede sprint daarom ook aangepast. Waar ik wel tevreden over was, was de beschrijving van de support.

### Procesevaluatie

* **Interviews**

Dit was mijn eerste sprint en was nog redelijk nieuw binnen het bedrijf. Toen ik begon met het zoeken naar mensen om te interviewen ging dat ook nog niet heel erg soepel. Ik was eerst de I&S afdeling rond gelopen om daar wat vragen te stellen en werd daar beetje rondgestuurd. Dat was niet een super start en ben daarna naar de expert gegaan voor de klachten om daar gebruik van te maken. Daarna kon ik gerichter naar informatie vragen en liep dat een stuk gemakkelijker.

* **In kaart brengen huidige situatie**

Het in kaart brengen van de huidige situatie is eigenlijk gelijkmatig tot stand gekomen. De samenhang van modules rond de rapport generator kwamen bijvoorbeeld ter sprake bij mijn interview met I&S Medicatie. Ik ben hier later nog wel even voor naar de expert gegaan, omdat hij daar meer kennis over had.

Ik heb ook geprobeerd de huidige rapport generator te reverse enigineren. Dit had ik eigenlijk helemaal niet hoeven doen, gezien daar geen meerwaarde aan zat. Ik had gehoopt om wat meer kennis hierover te krijgen maar ik had beter naar de Routing module kunnen kijken. Gelukkig was het maar twee dagen ‘verloren tijd’.

* **Zoeken generatoren**

Zoals in het document al gezegd, was had ik er in eerste instantie zestien gevonden, waarvan ik er maar vier kon gebruiken. Ik vond dit zelf nogal weinig en kon op dat moment niet goed pijlen hoe mijn project zou verlopen met zo weinig mogelijkheden.

* **Schrijven document**

Het schrijven van het document stelde voor een deel niet veel voor. Zou was het stuk over licenties eigenlijk gewoon knippen en plakken. Toen ik had besloten om dieper in te gaan op de support en usability kwam er wat meer bij kijken.

Ik moest voor het eerst uitzoeken hoe een rapport generator überhaupt werkte. Omdat de cursus was uitgesteld, wist ik niet eens hoe de huidige generator werkte. Ik heb daarom met de handleidingen geprobeerd een paar voorbeelden te maken met StimulSoft. Dit zorgde er wel voor dat ik na een redelijke tijd een voorkeur had voor dat product.

## Sprint 2: Evaluatie

### Productevaluatie

Voor deze sprint heb ik uiteindelijk twee producten opgeleverd het oriëntatie document en een deel een van het onderzoeksrapport.

* **Onderzoeksrapport**

Tijdens deze sprint is dit document voor een deel herschreven. Ik ben nu tevreden over het document. Het gebruik van een scenario om de usability te kunnen bepalen is een goede manier geweest om de score te bepalen. Ook vind ik het overzicht dat gemaakt wordt door de conclusie duidelijk en geven de verschillende vormen daarvan een duidelijk beeld.

### Procesevaluatie

Tijdens deze sprint is niet veel nieuws gebeurd gezien het vooral een verlengde was van de eerste sprint. Het idee om een scenario te gebruiken voor de usability betekende wel dat ik een deel van het document moest herschrijven en alle oude programma’s opnieuw moest beoordelen.

Ook ben ik tijdens deze sprint in totaal een week ziek geweest. Dit is ook te zien in de Burn Down chart van de sprint (Zie Figuur 30). Het lijkt in de sprintbacklog dat ik een week langer aan deze sprint bezig ben geweest om te kunnen compenseren voor de afwezige dagen; dat is niet het geval geweest. Ik werd ziek een dag voor het evaluatie gesprek. Toen ik weer beter was, moest ik deze afspraak eerst weer inplannen voordat we de sprint konden afsluiten. Dit komt alleen voor omdat ik de enige ontwikkelaar aan dit project was. Had dit project meerdere ontwikkelaars gehad dan had de retrospective gewoon door gegaan zonder mij.

Figuur 28 Burn Down Sprint 2

## Sprint 3: Evaluatie

### Productevaluatie

* **Adapters**

Over de adapters om de Stimulsoft controls is niet veel te zeggen. Ze werken zoals ze moeten en zijn niet ingewikkeld. Het was mijn eerste keer dat ik gebruik maakte van de Wrapper Pattern dus het was wel interessant.

* **Converter**

De converter ben ik zeer tevreden over. Het kan alle test bestanden die ik uit EZIS heb gehaald converteren naar een Object en deze weergeven in de TreeView zonder enig probleem. Er zijn een tweetal situaties bekend waar de conversie niet helemaal goed mee om kan gaan en daar moet nog een oplossing voor gezocht worden.

De StimulSoft converter werkt uitstekend. Het is mogelijk om alles behalve Charts Images, Script, ChipSoft specifieke components te converteren.

1. Charts heb ik opzettelijk overgeslagen omdat dit dusdanig ingewikkeld en uitgebreid zou zijn en dat het veel tijd zou kosten om het uit te zoeken en daar had ik in deze sprint geen tijd meer voor. het
2. Images werken voor 50% zolang het verwijst naar de database en het een byte[] gebruikt is het geen probleem. Maar als het een fysieke image is, dan wordt deze door delphi als TJPEG in Hex opgeslagen. Het is mij nog niet gelukt om terug te draaien.
3. ChipSoft components zijn extra components die door ChipSoft zijn toegevoegd. Deze moeten eerst gemaakt worden voordat daarnaar geconverteerd kan worden.
4. Script is gecodeerd opgeslagen en kan niet worden uitgelezen.

Gebruikers moeten na het converteren van een bestand nog wel even nakijken of alles op de juiste positie terecht is gekomen. Sommige StimulSoft elementen hebben een andere punt van oorsprong dan de oude elementen. Ook moeten gebruikers weer de script toevoegen hoever dat nu nodig is.

* **Datastructuur**

De datastructuur was teleurstellend. Ik ben zelf van mening dat StimulSoft een erg goed product is, maar het kan niet goed om gaan met de ICS\_Datastructuur. Ik heb tijdens deze sprint drie implementaties geprobeerd om de data snel naar de controls te krijgen maar dat is niet geslaagd.

### Procesevaluatie

Ik vond dit zelf een sprint die op sommige plekken heel goed verliep zoals de converter en wrappers en op andere moment dramatisch. Ik vond StimulSoft een goed programma om mee te werken en zij hadden ook een van de beste supports. Dat ze dan zo slecht omgaan met data vond ik dan ook vervelend en hoopte dat de andere programma’s hier niet zo moeilijk over zouden doen.

* **Opstellen Use Case**

Ik had al enige tijd geen use cases gemaakt of gebruikt dus ik moest mijn boek er weer bij pakken. Dit was vooral met betrekking tot extends en includes. Ik had eerst nog gedacht om alle eisen op papier te zetten maar dat leek mij overdreven omdat ik deze toch niet zelf ging implementeren.

* **Opstellen Klassen diagram**

Omdat het al bekend was dat ik meerdere prototypes moest maken heb ik gelijk geprobeerd centrale producten te scheiden, zoals de datastuctuur en de convert.

Waar ik mij wel even zorgen over maakte, is dat ik veel iteratieve klassen en functies heb. Zo was de converter initieel een heel ingewikkelde klasse die ik uit elkaar getrokken heb.

* **Maken wrappers**

Voordat ik mijn stage bij ChipSoft begon, heb ik niet veel gewerkt met WPF en al helemaal niet met Bindings en Dependency Properties. Ik heb tijdens de cursus wel enig inzicht hierin gekregen, maar het gebruiken hiervan was een andere ervaring.

De keuze voor wrappers had ik al sneller gemaakt omdat ik deze pattern al eerder heb moeten gebruiken. Toen ik had uitgevonden hoe ik de Bindings moest maken was dit vrij eenvoudig.

* **Maken datastructuur**

De datastructuur heeft mij heel erg gefrustreerd. Eerst heb ik het geprobeerd met DataSet en kwam er al redelijk snel achter dat die manier teveel geheugen in zou nemen en veel te complex zou zijn.

Ik zat daarna een tijd vast omdat ICS\_Objecten een dynamiche structuur hebben en ik niet wist hoe ik een implementatie kon maken die dat aan kon. Toen ik eindelijk wat gevonden had, bleek dit nog vrij ingewikkeld te zijn ook. Omdat de GetProperties methodes oproepen binnen de programma’s was het lastig om te debuggen en was het oplossen van problemen soms gis werk.

Het laatste probleem waar ik tegenaan liep, was het feit dat de ICS datastructuur gemaakt is uit Interfaces en het dus niet mogelijk was om deze direct naar mijn structuur te casten.

* **Maken converter**

Het maken van de converter ging vrij vlot en liep goed. Het enige probleem waar ik bij vast heb gezeten waren afbeeldingen die opgeslagen waren in het tekst bestand. Deze ware geencrypt naar een TJPEG bestand (Delphi) en heb ik alles geprobeerd om terug te krijgen maar dat is mij uiteindelijk niet gelukt. Vond ik erg jammer.

Ik had de converter van Object naar Rapport eerst ontworpen als een klasse. Dit zorgde voor veel problemen en hierbij raakte ik vaak de weg kwijt. Ik heb daarna ook het klasse-diagram aan gepast. Toen kwam ik er ook achter dat het mogelijk was om met code een rapport te maken en was een stuk makkelijker.

## Sprint 4: Evaluatie

### Productevaluatie

* **Designer**

De implementatie was niet helemaal naar mijn zijn. Mijn eigen HwndHost functioneerde goed alleen kon ik het probleem met de Ribbon niet maken. Gezien de rest van dit product het goed deed vond ik dat jammer.

* **Datastructuur**

Ik was zeer tevreden over deze implementatie van de datastructuur. Ook al werd er nog gebruikt van de Veldenkiezer om verder in een object te kunnen navigeren. Ik was vooraf wat angstig na de slechte start met het programma van StimulSoft.

* **Converter**

De converter werkte zo goed als ik had gehoopt. Het enige wat niet goed functioneert, is de scaling. Alle controls zijn ongeveer 50% te groot en om daar een verandering in aan te brengen zorgt ervoor dat ze of veel te klein zijn of nog groter worden.

### Procesevaluatie

* **Designer**

Toen het niet mogelijk was om de control in een WindowsFormHost te plaatsen, was ik eerste instantie bang dat dit prototype ook zou falen gezien het een Must was dat het programma geïntegreerd zou zijn.

Omdat er niet veel informatie te vinden is over WPF en WindowsFormHost in combinatie met TopLevel was ik erg blij dat er toch een oplossing tussen zat. De HwndHost gebruikt wel user dll’s en window handle’s waar ik niet mee bekend was.

Daarom heb ik blindelings gebruik gemaakt van deze oplossing en weet dus niet helemaal wat er precies gebeurt.

* **Datastructuur**

Ik had verwacht dat mijn ICustomTypedDescriptor goed zou werken met DevExpress. Het was ook vrij vervelend toen dat niet werkte, omdat ik niet gelijk wist hoe ik dat probleem zou moeten oplossen. Gelukkig was de documentatie van het product goed en kon ik daarmee verder.

Enig nadeel aan het maken van een ITypedList is dat er weinig voorbeelden zijn op het internet en in combinatie met een dynamic object zijn er geen van. Dit maakte het moeilijk om problemen op te lossen omdat ik gewoon niet wist waar ze vandaan kwamen.

Ik had verwacht dat andere producten een Business Object ook zouden omzetten naar hun eigen structuur zoals bij StimulSoft. Dit leek mij logisch, omdat je dan maar op een manier naar data hoeft te vragen. Deze aan namen bleek gelukkig fout.

* **Converter**

Er was bij mij in het begin wat verwarring over de sub rapporten van DevExpress. Ik was er van overtuigd dat deze apart opgeslagen moesten worden. Ik wist dan ook niet goed hoe ik deze moest verwerken in de converter.

Ik heb uiteindelijk gewoon de StimulSoft methode aangehouden en kwam er achter dat mijn eerdere overtuiging niet klopte en dat sub rapporten gewoon in het zelfde document geplaatst werden.

## Sprint 5: Evaluatie

### Productevaluatie

* **Designer**

Dramatisch. Net als bij DevExpress was de control TopLevel en moest ik de HwndHost gebruiken. Dit werkte, alleen ging het hele programma kapot. De ene control reageerde nergens op en de andere wilde zichzelf niet refreshen. Ik moest het uit eindelijk buiten EZIS hosten; wat niet de bedoeling was.

Ik heb ook geprobeerd de preview weer te geven in de WPF viewer om zo minder schermen te open en kwam er achter dat deze een compleet andere manier van aan roepen heeft. Dit vond ik zonde.

* **Datastructuur**

Dit is het enige programma dat goed omgaat met de ICS datatructuur. Jammer genoeg niet helemaal zoals gewenst. Zo weet het niet wat er met TCSImages (afbeeldingen) gedaan moet worden.

Ook met de ITypedList gaf het precies weer wat er weergegeven moest worden. Het is alleen jammer dat gelijk alles opgehaald wordt.

* **Converter**

Werkt niet helemaal zoals het moet. Bands hebben bij deze rapporten een x,y,width,height wat vreemd is gezien ze altijd net zo breed zijn als de pagina. Ook is de hoogte van elementen niet relatief aan de andere bands. Dit zorgde ervoor dat alles op gekropt werd, maar is uiteindelijk wel verholpen.

* **Updates**

Over de updates van DevExpress en de ITypedList ben ik erg tevreden. De ribbon wordt eindelijk fatsoenlijk weergegeven en ook is het mogelijk om een collectie helemaal weer te geven met de ITypedList zonder ook maar iets uit te lezen.

StimulSoft was dan weer jammer. Ik had gehoopt dat ik daar een data-adapter voor zou kunnen maken om er nog een goed prototype van te kunnen maken. Helaas niet gelukt.

### Procesevaluatie

* **Designer**

Het implementeren van de designer ging makkelijk. Het TopLevel probleem had ik al bij de vorige sprint opgelost, dus wist ik wat er moest gebeuren. Dat alles niet werkte was wel frustrerend. Ik vond het ook vervelend dat er op het forum ook geen hulp te vinden was.

Ik heb er niet te lang bij stil gestaan en ben verder gegaan om ook de andere delen van het programma te testen. Ik was ook best verrast dat het als enige programma de ICS datastructuur goed uit kon lezen.

* **Datastructuur**

Ik heb tijdens dit project steeds meer inzicht gekregen in de ICustomTypedDiscriptor en de ITypedlist en kon de uitbreiding tijdens deze sprint vlot uitwerken. Ik snap nu ook hoe ik deze interfaces moet debuggen en begrijp de precieze werking.

Het maken van de dataprovider was ook weer iets nieuws. Gelukkig hoefde mijn provider geen update of delete statements te ondersteunen, dat het een stuk makkelijker maakte.

* **Converter**

Het probleem met de converter in combinatie met de ITypedList had ik al snel door. Ik had gelijk een idee om het op te lossen maar heb dit wel even met mij begeleider besproken om zijn mening te vernemen Ik heb daarna zelf de oplossing wel gemaakt uitgewerkt en alsnog de keuze geopperd bij de retrospective.

## Sprint 6: Evaluatie

### Productevaluatie

* **Zelfbouw**

Over de GUI die ik gemaakt ben ik tevreden. Slepen binnen WPF is redelijk ingewikkeld en dat is mij goed gelukt. Ook werkte het in- en uitzomen op het rapport goed. Waar ik niet tevreden over was, dat was de engine die wij gekozen hadden om te gebruiken. Deze bleek minder functies te ondersteunen dan gedacht en was ook slecht uit te breiden.

* **Reportbuilder 3.0 rapport**

Ik denk dat een ontwikkelaar na het lezen van dat document een OLE-DB adapter in elkaar zou kunnen zetten. Ik heb alle informatie die ik kon vinden goed verwerkt en duidelijk weergegeven.

* **Testen documentatie**

De beschrijving van de test leek mij redelijk duidelijk. Het sommeren van de data is naar mijn mening makkelijk uit de tabellen uit te lezen.

### Procesevaluatie

* **Zelfbouw**

Het maken van de GUI liep goed. Ik had wat problemen met het gebruik van drag en drop in MVVM, gelukkig had ChipSoft een voorbeeld in EZIS zitten en was er op het internet ook redelijk wat informatie en voorbeelden te vinden.

Het omzetten van de layout van een rapport naar de open source enige was echt een drama. Ik had de rapport engine kort onderzocht tijdens de inventarisatie fase en er daarna niet meer naar omgekeken. Toen ik dit product wilde gebruiken bij dit prototype liep ik daarbij tegen de lamp aan. ging het mis.

Het bleek alleen mogelijk om rapporten op te bouwen aan de hand van DataSet’s en tabellen, dat niet het doel was. Ook was het aanpassen van het product niet goed mogelijk omdat er gebruik werd gemaakt van het Microsoft FlowDocument object. Hiermee worden documenten gemaakt en is het niet mogelijk om coördinaten op te geven van elementen.

Ik had hier beter naar moeten kijken voor ik deze sprint begon.

* **Reportbuilder 3.0**

OLE-DB is al een oud begrip bij computers en maakt nog gebruik van COM interfaces. Dit houdt in dat er veel oude data te vinden is en het lastig is om te zoeken naar relevante gegeven. Tevens maken de meeste voorbeelden gebruik van C++ dat ik al een tijd niet heb gebruikt.

Gelukkig is MSDN een goed punt van informatie dus is het uit eindelijk toch goed afgelopen en vond ik het een pittig stukje techniek.

* **Testen**

Het testen van de prototypes was niet mijn favoriete activiteit tijdens dit project. Ten eerste moesten alle test apart uitgevoerd worden omdat ze anders elkaar zouden tegenwerken.

Ook zorgde elke test die ik deed ervoor dat ik mijn PC niet kon gebruiken omdat dit invloed op de test zou kunnen hebben.

Ook heb ik de tests meerdere keren moeten uitvoeren, omdat ik dacht dat ik inconsistenties zag. Zo kon het zijn dat een collectie met meer data opeens minder geheugen gebruikte dat niet mogelijk zou moeten zijn (Om dit op te lossen heb ik de serie test overnieuw gedaan).

Een ander probleem was dat ik nog nooit gewerkt had met de .Net Mermory Profiler. In eerste instantie heb ik alleen gekeken naar de Managed Mermory[[17]](#footnote-17) van EZIS. Omdat ik het gevoel had dat de waarde die ik hierbij keer op keer niet klopte, heb ik advies gevraagd en heb ik ook het Unmanaged Memory[[18]](#footnote-18) erbij gepakt.

Toen ik klaar was, had ik wel het gevoel dat ik een goed proces had afgewerkt en was ik er tevreden over.

## Sprint 7: Evaluatie

### Productevaluatie

* **Adviesrapport**

Dit rapport is het eind product van dit project. Ik heb het gevoel dat dit een compleet product is met alle informatie daarin verwerkt die vergaard is. Het is volgens mij ook duidelijk voor de lezer hoe ik tot mijn conclusie gekomen ben. Ik weet ook dat ChipSoft dit document kan gebruiken om haar definitieve keuzen te kunnen maken.

* **Converter**

De nieuwe converter werkt erg goed. Bij het testen bleek dat het mogelijk was om alle 2.784 rapporten binnen twee minuten om te zetten naar een nieuwe rapport formaat. Ik denk ook dat de GUI duidelijk is voor mensen om te gebruiken. De Error afhandeling had misschien iets beter gekund,voelt niet helemaal geweldig. Het staat los van de converter zelf en hebben ook geen invloed op elkaar. Als het mogelijk had geweest om een geselecteerde error te highlighten in de converter had ik het beter gevonden.

### Procesevaluatie

* **Adviesrapport**

Het maken van het advies rapport was eigenlijk een grote samenvatting van alle documenten die ik al eerder had opgeleverd. Ik had al mijn bevindingen van de prototypes gedocumenteerd evenals de processen tijdens het inventariseren. Het maken hiervan verliep vrij soepel.

* **Converter**

Bij het maken van de nieuwe converter is veel code gebruikt van de oude converters. Het enige echt nieuwe aan dit product is de GUI. De problemen die zich voordeden tijdens het maken hiervan waren vooral het multi-threading toepassen op het product en een paar Cross-Thread calls die gemaakt moesten worden om de GUI te updaten met de geconverteerde rapporten.

Door een foutje bij het gebruiken van de database zorgde er wel voor dat ik per ongeluk een viertal rapporten had verwijderd. Gelukkig was dit eenvoudig terug te draaien.

## Competenties

### 1.2 Voorbereiden en opstarten softwareontwikkeltraject

Deze competentie beschrijft het vastleggen van problemen, de reden voor het project en aan de hand daarvan een project op te zetten.

Aan deze competentie heb ik voldaan met mijn plan van aanpak en het oriëntatie document. In deze documenten worden de problemen uitgelegd en de gewenste resultaten en aanpak beschreven (Zie Bijlage 1 & 3).

### 1.4 Uitvoeren analyse door definitie van requirements

Bij deze competentie moeten de eisen van een project gehaald worden bij meerdere personen.

Deze competentie is toegepast in het tot stand komen van het oriëntatie document, doormiddel van interviews bij meerdere groepen van de afdeling I&S. Deze eisen zijn vervolgens gesorteerd volgens de MoSCoW regels. En is er van de globaal relevante eisen een Use Case opgesteld (Zie Bijlage 2 & 3).

### 3.2 Ontwerpen systeemdeel

Beschrijving van het systeem met UML en GUI uitwerking

Ik heb geprobeerd de prototypes op een evolutionaire manier te maken. Hierbij zijn de klasse-diagrammen zo veel mogelijk uit elkaar getrokken om ze te kunnen hergebruiken.

Zo staan de kern onderdelen zoals de datastructuur en de converter los van de individuele prototypes en kunnen door allemaal gebruikt worden. Tevens zijn delen van de prototypes hergebruikt bij de daarop volgende prototypes bijv. de HwndHost implementatie en voor een groot deel de specifieke converters.

Ook is er een GUI gemaakt voor de Converter en voor het laatste prototype.

### 3.3 Bouwen applicatie

Het uitwerken van de ontwerpen die gemaakt zijn tijdens 3.2.

Ik heb alle diagrammen die ik gemaakt heb tijdens het ontwerpen van de systeemdelen uitgewerkt om de prototypes te kunnen maken. Hierbij zijn ook ingewikkelde iteratieve functies geschreven en is er gebruikgemaakt van Mircosoft Interfaces.

### 2.3 Uitvoeren gegevensconversie

Bij deze competentie gaat het om converteren van gegevens.

Tijdens dit project heb ik niet de daadwerkelijke conversie uitgevoerd maar wel de tools gemaakt om dit mogelijk te maken. Het is mogelijk om alle rapporten te converteren naar het uiteindelijk gekozen prototype.

Na een conversie is het ook mogelijk om te zien of een bestand goed is geconverteerd en zijn er opmerkingen genoteerd waar eventueel op gelet moet worden.

1. Extensible Application Markup Language: Is een XML gebaseerde taal die gebruik wordt om een interface te werpen in een WPF omgeving [↑](#footnote-ref-1)
2. MoSCoW Niveaus: Must Have, Should Have, Could Have, Wanna Have [↑](#footnote-ref-2)
3. [www.StackOverflow.com](http://www.StackOverflow.com) is een forum gemaakt door ontwikkelaars voor ontwikkelaars. Het is mogelijk voor iedereen om vragen te stellen of te beantwoorden. Op het moment zijn er meer dan vier miljoen programmeer gerelateerde vragen te vinden. [↑](#footnote-ref-3)
4. [www.CodeProject.com](http://www.CodeProject.com) is een website waar programmeurs articles en demo’s kunnen plaatsen om zo andere ontwikkelaars inzicht te geven in bepaalde technieken of handige tips. Met een community van negen miljoen en met ongeveer dertig duizend leden online op elk moment van de dag. [↑](#footnote-ref-4)
5. De support is de hulp die geboden wordt door de provider van het 3th party programma. Dit is inclusief de eventuele community (Forum) die zij ondersteunen. [↑](#footnote-ref-5)
6. Een DataSet is een onderdeel van ADO.Net en is een Datastructuur die binnen de .Net omgeving gebruikt kan worden [↑](#footnote-ref-6)
7. De List Interfaces van Microsoft maken het mogelijk om voor een programmeur gewenste functionaliteiten toe te voegen aan de standaard List die in C# aanwezig is. ChipSoft maakt hier gebruik van om data in delen te laden (live collecties Zie 3.1.1). [↑](#footnote-ref-7)
8. “Het Adapter-patroon wordt gebruikt om de interface van de ene [klasse](http://nl.wikipedia.org/wiki/Klasse_%28informatica%29) om te zetten in de interface van een andere klasse. Adapter kan twee klassen laten samenwerken die anders niet zouden samenwerken door hun verschillende interfaces.” Quote Wikipedia [↑](#footnote-ref-8)
9. DependencyProperties zijn de enige properties van een usercontrol die via de XAML gebonden kunnen worden aan data. Komt alleen voor in WPF. [↑](#footnote-ref-9)
10. Een MDI-Parent is een type WinForm waarin het mogelijk is om meerdere windows weer te geven. [↑](#footnote-ref-10)
11. Op deze site staan de handleidingen voor Microsoft producten en interfaces. Ook is er een forum aanwezig voor gebruikers. [↑](#footnote-ref-11)
12. <http://shevaspace.blogspot.nl/2007/02/how-to-host-top-level-hwnds-in-wpf.html> [↑](#footnote-ref-12)
13. Ribbon is de naam de MicroSoft heeft gegeven aan de menu balk in zijn programma’s [↑](#footnote-ref-13)
14. Aero maakt het mogelijk om de optimacy van menu balken in Windows aan te passen. [↑](#footnote-ref-14)
15. Tijdens de laatste sprint vond ik een optie om het Glass Effect uit te zetten binnen DevExpress. De support van DevExpress schoot hierbij te kort. (Zie 10.2.2) [↑](#footnote-ref-15)
16. Een Break-Point is een indicator binnen Visual Studio die actief wordt als dat wordt gepasseerd tijdens het draaien van de software. Het is vervolgens mogelijk om te stoppen en de status van het programma te controleren of om een waarde te printen. [↑](#footnote-ref-16)
17. Geheugen waarvan gezien kan worden welk programma het gebruikt en zelf door welk object. [↑](#footnote-ref-17)
18. Geheugen waarvan niet zeker is waardoor het gebruikt wordt. Voorbeeld hier van is code geschreven in een andere taal dan .Net of afbeeldingen. [↑](#footnote-ref-18)