Afstudeerverslag

Onderhoud aan de classes van Micros

Auteur	:	*D. Prins*
ID-Code	:	*98002261*
Opleiding	:	*Informatica & Informatiekunde*
Opleidingsvariant	:	*Management & Beheer van Informatie Technologie*
Afstudeerperiode	:	*2003-2.2*
Kenmerk	:	*DOA2003-2.2.52*
Examinatoren	:	*Mw. S.N. Minoun* *Dhr. E.M. van Doorn*
Bedrijf	:	*Micros B.V.*
Plaats	:	*Schiedam*
Bedrijfsmentor	:	*Dhr. J.P. Dees*
Datum	:	*25 maart 2004*

Colofon

Dit document is een uitgave van:
Micros B.V.
Fokkerstraat 574
Postbus 250
3100 AG Schiedam

Telefoon : (010) 437 86 11
Telefax : (010) 437 86 77

Wijzigingen voorbehouden.
Vermelde merknamen zijn beschermd door de desbetreffende eigenaar.

Referaat

Prins, D. , afstudeerder.
Informatica & Informatiekunde; Management & Beheer van Informatie Technologie (MBIT), Haagse Hogeschool.
Eindverslag Afstuderen, Schiedam, maart 2004.

In dit verslag zijn de werkzaamheden beschreven die zijn uitgevoerd tijdens het afstudeerproject van dhr. D. Prins bij Micros B.V. te Schiedam.

Descriptoren

- Visual Dataflex
- Onderhoud
- Classes
- Object Oriëntatie
- IAD
- Versiebeheer

Voorwoord

Dit verslag heb ik geschreven in het kader van mijn afstuderen aan de Haagse Hogeschool, studierichting Management en Beheer van Informatie Technologie (MBIT).

Het afstudeertraject was een zeer leerzame ervaring. Het gaf mij een goed beeld dat de theorie, behandeld tijdens mijn opleiding, niet altijd overeenkomt met de praktijk, zoals ervaren tijdens mijn afstudeertraject. Theorieonderwijs van MBIT is wel een goede basis voor de uitdagingen van de praktijk. Daarnaast heb ik veel geleerd over zaken die aan de orde komen bij onderhoud aan een bestaand systeem.

Ik wil hier tevens een aantal mensen bedanken die voor mij, tijdens mijn afstudeerperiode, een belangrijke rol hebben gespeeld. Allereerst wil ik mijn bedrijfsmentor dhr. J.P. Dees bedanken voor zijn begeleiding en steun die hij mij geboden heeft. Daarnaast wil ik mw. S.N. Minoun en dhr. E.M. van Doorn bedanken voor hun begeleiding vanuit de Haagse Hogeschool. Vervolgens wil ik mijn familie bedanken voor de steun die gegeven is tijdens mijn gehele studie, in het bijzonder mijn oom.

Ten slotte wil ik u, de lezer, bedanken voor het nemen van de tijd, mijn verslag te lezen.

Schiedam, 25 maart 2004

D. Prins

Inhoudsopgave

1. Inleiding

In dit verslag beschrijf ik mijn afstudeeropdracht. In het kort luidt de opdracht 'communicatie verbeteren tussen de Micros applicatie en andere software'. Door aanpassingen te maken aan sommige Micros *classes* is het mogelijk om het gebruik van andere software, beter te laten verlopen. De aanleiding hiervoor is, dat binnen de Micros-organisatie momenteel geen kennis meer aanwezig is, over de precieze werking van de Micros *classes,* doordat ze in het "verre verleden" gemaakt zijn. Door richtlijnen te maken voor onderhoud en toepasbaarheid is het mogelijk die kennis weer terug te krijgen.

Dit verslag is opgebouwd volgens de fasering, die gehanteerd wordt tijdens het afstudeertraject. Het afstudeerverslag beschrijft in globale zin het gehele project. Ter verduidelijking van problemen, oplossingen en onderzoeken zijn onder andere illustraties en notities opgenomen in dit afstudeerverslag.

Hoofdstuk 2, 'Organisatie' geeft een beschrijving van Micros B.V., de afstudeerplek ter afsluiting van mijn studie.

Het 3e hoofdstuk bevat de opdrachtomschrijving, om inzicht te verschaffen in de opdracht, die ik, ter afronding van mijn studie Management en Beheer van Informatie Technologie (MBIT) aan de Haagse Hogeschool, vervuld heb bij Micros B.V..

Hoofdstuk 4 is het globale plan van aanpak, dit om meer duidelijkheid te geven over de beheersing van het hele afstudeertraject.

In het 5e hoofdstuk geef ik een gedetailleerde beschrijving van de situatie bij aanvang van afstuderen. Verder beschrijf ik hier nieuw geleerde technieken en technologieën en geef ik aan hoe deze zijn onderzocht.

Het 6e hoofdstuk schetst uitleg over het verkrijgen van meer inzicht in de taal en de ontwikkelomgeving. Alsmede in het ontwikkeltraject, documentatie, codering, gebruikershandleiding en versiebeheer voor de ontwikkeling van nieuwe Micros *classes*. Tevens is hier uitleg te vinden over wat mogelijk als richtlijn voor een ontwikkeltraject voor Micros *classes* kan gelden.

Hoofdstuk 7 beschrijft het ontwikkeltraject voor Crystal Reports, van definitiestudie tot beschrijving van de invoering.

Hoofdstuk 8 beschrijft richtlijnen, die kunnen leiden tot het op correcte wijze onderhouden van Micros *classes*, met name door gebruik van versiebeheer.

Het laatste hoofdstuk is bedoeld als product- en proces-evaluatie.

2. Organisatie

In dit hoofdstuk beschrijf ik Micros B.V., de afstudeerplek ter afsluiting van mijn studie.

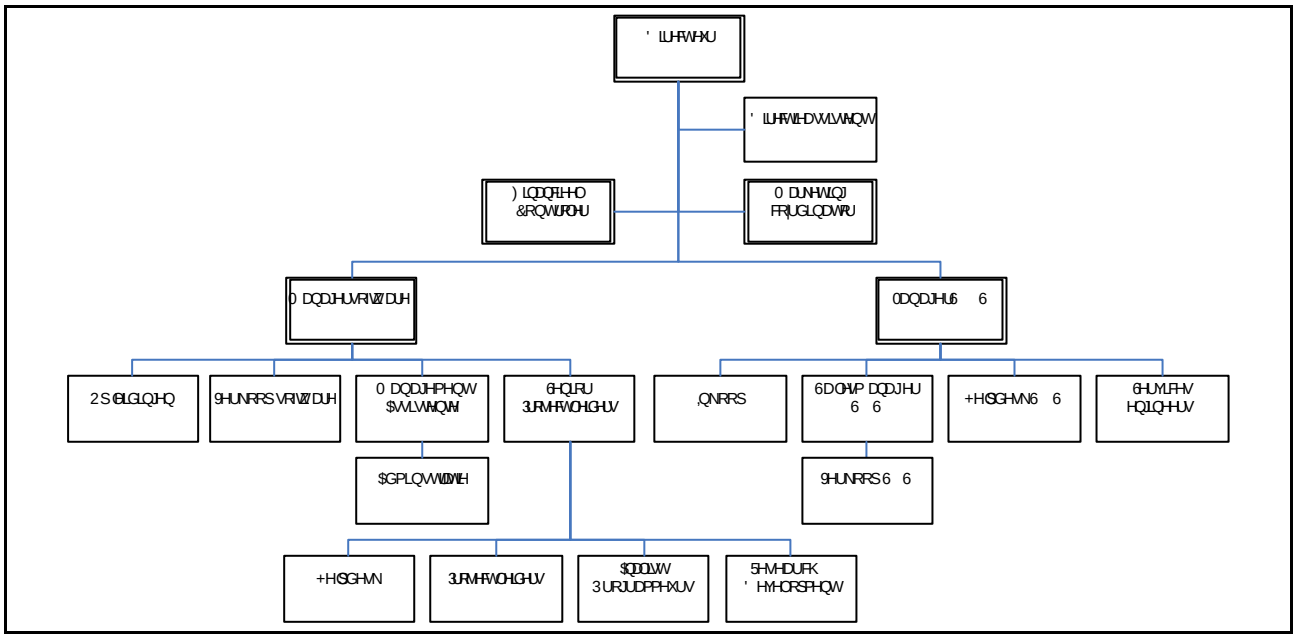
2.1 Algemeen

Micros B.V. is opgericht in 1981, wat het een relatief oud automatiseringsbedrijf maakt. Micros B.V. is met succes actief in de automatiseringsbranche en beschikt over een ruime deskundigheid in vele branches. Kenmerkend voor Micros is de gedegen en resultaatgerichte, integrale aanpak van automatiseringsoplossingen. Van vooronderzoek en advisering tot en met implementatie en ondersteuning. Dit biedt Micros B.V. de mogelijkheid om aan haar klanten een totaaloplossing op het gebied van automatisering aan te bieden. De programmatuur wordt ontwikkeld voor kleine handelsbedrijven en lokale overheden en kan standaardsoftware zijn, maar het is meestal maatwerk.

Doel van Micros B.V. is het ontwikkelen en verkopen van hoogwaardige branchespecifieke automatiseringsoplossingen en diensten.

Micros B.V. werkt met verschillende partners op software- en hardwaregebied. Op softwaregebied wordt voornamelijk samengewerkt met Microsoft en Data Access. Op het gebied van hardware is Hewlett-Packard hun grootste partner.

2.2 Interne organisatie

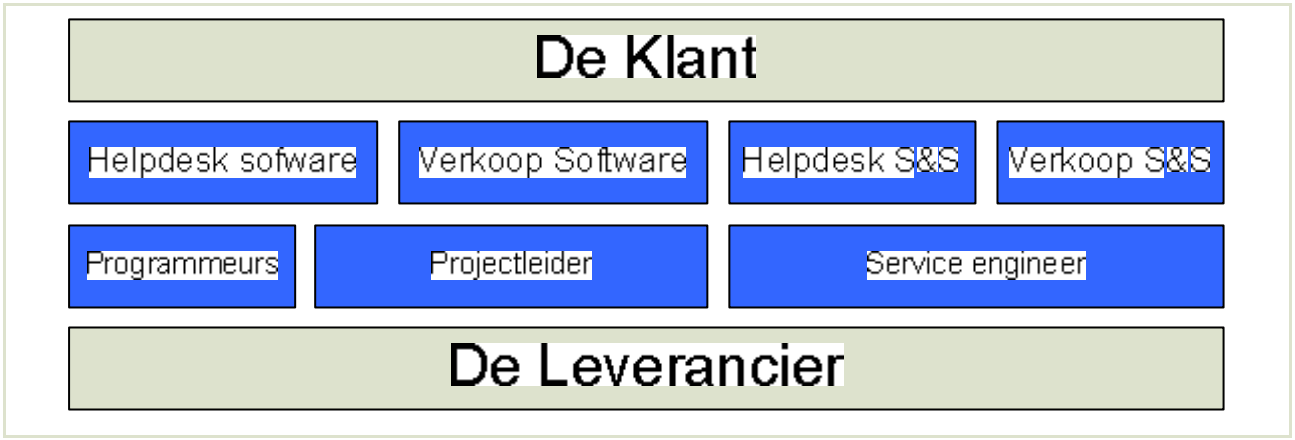


Figuur 2.1 "Organigram"

Het is in de organisatiestructuur te zien dat de organisatie in twee stukken is verdeeld. Namelijk de software- en hardwaretak (S&S staat voor Services & Support). Het is een logische structuur dat beide takken los staan van elkaar, omdat beide takken hun eigen diensten hebben die weliswaar samen aangeboden worden, maar niet verplicht behoeven samen te gaan. S&S beheert ook het interne netwerk van Micros B.V..

Bij Micros B.V. is de directie verantwoordelijk voor het selecteren en aannemen van personeel. De leidinggevende is verantwoordelijk voor het introduceren van de nieuwe medewerker in zijn werkzaamheden.

2.3 Externe organisatie



Figuur 2.2 "Externe organisatie"

Bij Micros B.V. wordt veel maatwerk geleverd voor haar klanten. Indien noodzakelijk zoekt een klant contact met de helpdesk of afdeling verkoop. Als de helpdeskmedewerker geen oplossing heeft, zal er contact opgenomen worden met de programmeur of in enkele gevallen met de projectleider. Dat Micros B.V. een klantvriendelijke organisatie is blijkt uit het feit dat ook de programmeurs de klant regelmatig te woord staan.

Voor extra hulp kan de leverancier ingeschakeld worden. Zo heeft bijvoorbeeld een medewerker van Data Access (software leverancier) binnen Micros B.V. meegewerkt aan een project.

2.4 Software Producten

Micros B.V is een automatiseringsbedrijf en heeft ook verschillende producten op softwaregebied waaronder:

- Cursus Administratie Systeem (KAS)
- Markt en Kermissen Administratie Systeem (MAS)
- Project Administratie Systeem (PAS)
- Relatie Administratie Systeem (RAS)
- Klachten Behandel Systeem (KBS)
- Handel Administratie Systeem (HANDEL)
- Verhuur Administratie Systeem

Voor de meeste klanten worden aanpassingen gemaakt in het reeds bestaande pakket. De aanpassingen gaan van het toevoegen van een invoerveld tot het maken van totaal nieuwe invoervensters.

2.5 Klanten

De uitgebreide klantenportefeuille Micros B.V. is te verdelen in 3 groepen: Lokale overheden, Opleidingsinstituten en Handelsbedrijven.

Onderdelen van de lokale overheden waaronder die van Amsterdam, Rotterdam, Den Haag en Schiedam maken gebruik van Markt en Kermissen Administratie Systeem, Haven Administratie Systeem en het Klachten Behandel Systeem .

Terwijl Handelsbedrijven gebruik maken van het pakket Handel en Opleidingsinstituten maken gebruik van het Cursus Administratie Systeem.

2.6 Strategie

2.6.1 Medewerkers

De medewerkers bepalen vaak het succes van een bedrijf. Daarom wil Micros B.V. zich goed inzetten voor haar medewerkers, dit gebeurt door middel van het aanbieden van een hoog en constant opleidingsniveau, continue kennisuitwisseling, specialisaties in verschillende pakketten en goede arbeidsvoorwaarden.

2.6.2 Klanten

De klanten van Micros B.V. behoren tot een bepaalde groep zoals hierboven te lezen is. Micros B.V. wil graag gefocust blijven op die groepen. Door het aanbieden van maatwerk en opleidingen probeert Micros B.V. de concurrentie de baas te blijven.

2.7 Mijn plek binnen Micros B.V.

Mijn algemene plek, als afstudeerder, binnen deze organisatie is op de afdeling software. Verder is mijn projectleider grotendeels verantwoordelijk voor R&D *(Research & Development).* Binnen de afdeling software behoor ik ook tot R&D.

3. Afstudeeropdracht

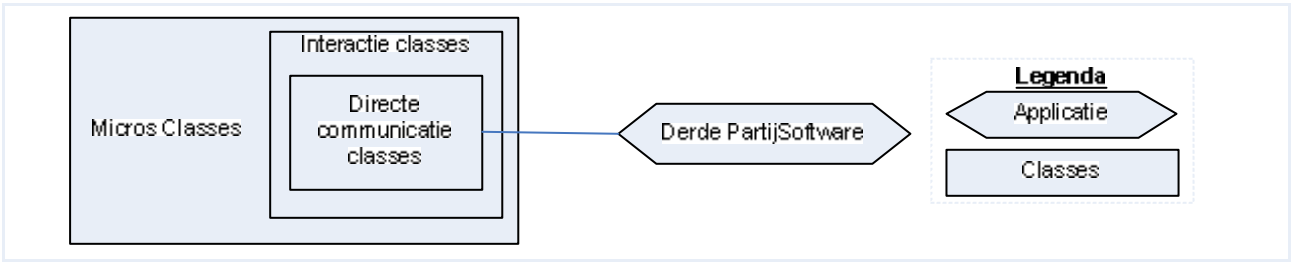
Dit hoofdstuk bevat de opdrachtsomschrijving, om inzicht te verschaffen in de opdracht die ik vervuld heb bij Micros B.V. ter afronding van mijn studie Management en Beheer van Informatie Technologie (MBIT) aan de Haagse Hogeschool.

Hieronder bevindt zich de opdracht gebaseerd op de opdrachtsomschrijving, welke een akkoord heeft gevonden tussen de afstudeerder en twee examinatoren op 6 januari 2004. De opdrachtomschrijving is de basis van het afstudeertraject. In de volgende paragrafen worden tevens enkele punten aangehaald, die van belang zijn voor het verkrijgen van meer inzicht in de probleemstelling.

3.1 Probleemstelling

Micros B.V. ontwikkelt software met behulp van Visual Dataflex 7 van Data Access. Mogelijk gaat Micros B.V. de overstap van Visual Dataflex 7 naar Visual Dataflex 9 maken. Binnen Visual Dataflex 7 wordt nog gebruik gemaakt van duizenden, door Micros intern ontwikkelde, *classes* (wordt nader uitgelegd in paragraaf 5.4.3) van Dataflex versie 4. Ongeveer 20% van deze *classes* is gebouwd voor interactie met software van een derde partij (Crystal Reports, MS-Word en Mail API). Een klein deel van deze 20% communiceert direct met software van een derde partij. (Zoals te zien in figuur 3.1.) Voor de communicatie met derde partij software worden deze *classes* soms niet gebruikt. Doordat software van een derde partij verder ontwikkeld wordt, moeten de *classes* die direct de software aanspreken iedere keer worden aangepast. Nieuwe versies van een derde partij software leveren daardoor problemen op. Deze problemen worden op dit moment opgelost door de *classes* te herschrijven zodanig dat de *classes* hun interactie met de bestaande software behouden.

De herschreven *class* verliest soms de mogelijkheid om te werken op de plaats van oude versies van dezelfde *class*. Tevens zijn er meerdere applicaties die gebruik maken van kopieën van dezelfde *classes*. Het is niet ongebruikelijk als de *classes* van beide applicaties niet aan elkaar gelijk zijn, al heeft de *class* een identieke naam en vergelijkbare functionaliteit. Dit zorgt er mede voor dat er duizenden *classes* zijn. Er is verder geen documentatie over de verschillen tussen de *classes,* de werking en het gebruik ervan.



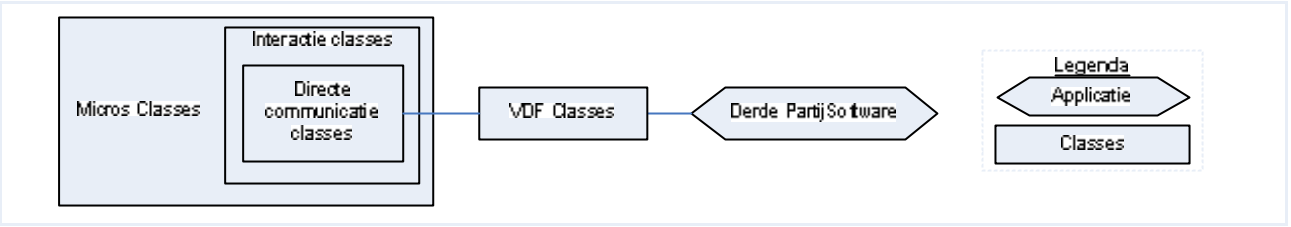
Figuur 3.1 "Opbouw classes"

Binnen Visual Dataflex 7 en 9 zijn nieuwe *classes* aanwezig die door Data Access (de makers van Visual Dataflex) ontwikkeld zijn en onderhouden worden. Sommige van deze *classes* zijn bedoeld voor interactie met andere software. Dit zijn *classes* die mogelijk dezelfde functionaliteit hebben als de *classes* die door Micros B.V. intern ontwikkeld zijn.

3.2 Doelstelling

Het doel van de afstudeeropdracht is tweeledig:

- Het verminderen en/of clusteren van de afhankelijkheid van derde partij software, door het vervangen van de relatie tussen de nuttige *classes* van Micros en derde partij software met een relatie tussen de *classes* van Micros en de nieuwe *classes* van Visual Dataflex 9.
- Het creëren van richtlijnen voor het gebruik van de nieuwe *classes* om ervoor te zorgen dat de nieuwe *classes* volgens een bepaalde standaard worden aangepast en toegepast.



Figuur 3.2 "Doelstelling 1"

3.3 Te hanteren methodieken en Technieken

Bij Micros B.V. is geen zichtbare ontwikkelmethode aanwezig voor het ontwikkelen van *classes*. De keuze is aan mij voor het bepalen van een ontwikkelmethode. Voor het realiseren van de afstudeeropdracht is gekozen voor de [IAD] (*Iterative Application Development*) ontwikkelmethode. SDM (*System Development Methodology*) werd afgeraden door de keurend docent, de heer van Doorn. [IAD] werd door betreffend docent en afstudeerder als goede ontwikkelmethode gezien voor de afstudeeropdracht.

De reden hiervoor is dat SDM een methode is die vooral wordt gebruikt bij grote, langlopende projecten. Iedere fase van systeemontwikkeling wordt met SDM slechts één keer doorlopen. Dit houdt in dat een fase volledig wordt afgerond alvorens aan de volgende fase te beginnen. Dit komt overeen met het watervalmodel. De nadelige gevolgen van deze methode zijn:

- Resultaten van implementatie zijn pas in een laat stadium zichtbaar. De opdrachtgever heeft alleen zicht op de vooruitgang van het project (de tussenproducten) en geen zicht op het uiteindelijke resultaat;
- Het complete systeem wordt steeds gedetailleerder uitgewerkt. Door de grote hoeveelheid ontwerpen en documentatie is de overzichtelijkheid en consistentie moeilijk te handhaven;
- SDM richt zich op het herontwerpen van handmatige procedures. Iets wat totaal niet aan de orde is voor dit afstudeerproject.

[IAD] werkt daarentegen op een iteratieve manier. Door evolutie van opeenvolgende versies, ook wel *pilots* genaamd, verloopt het ontwikkeltraject naar de uiteindelijke staat. [IAD] is een ontwikkelmethode die goed werkt in een object georiënteerde omgeving. Van [IAD] zal de techniek van het maken van *pilots* en het bijbehorende ontwikkeltraject gebruikt worden.

[UML] is een techniek die gebruikt is tijdens het afstudeerproject. [UML] is een duidelijke taal voor het in kaart brengen van verschillende relaties en verschaft een beeld van de werking van de verschillende *classes*. Behalve de bovengenoemde technieken is er ook gebruik gemaakt van flowcharts en PSD om de werking van verschillende procedures en functies te achterhalen.

3.4 Uitgangssituatie

Benodigde software

- Visual Dataflex 7 en 9;
- Microsoft Office Pakket.

Benodigde hardware

- Workstation voor rapportage;
- Testomgeving voor het testen van de *classes*.

Beschikbare rapporten

- Boeken over Visual Dataflex 7 en 9.

Aanwezige ideeën

- De huidige interface van de Micros *classes* in standhouden voor de interactie met de huidige software van Micros B.V. en binnen in de Micros *class* zoveel mogelijk gebruik te maken van de standaard *classes* van Visual Dataflex 7 en 9.

3.5 Te verrichten werkzaamheden

In het kader van de afstudeeropdracht zullen de volgende activiteiten verricht worden:

- Opstellen van een plan van aanpak;
- Bestuderen van de programmeermogelijkheden Visual Dataflex 7 en 9;
- Prototypes om de werking van de huidige situatie te achterhalen;
- Koppeling leggen tussen huidige *classes* en de nieuwe *classes* die met derde partij software communiceren;
- Definitiestudie/analyse voor verbeteringen en alternatieven;
- Plan van aanpak voor de gekozen verbeteringen;
- Definitiestudie van de gekozen verbeteringen;
- *Pilot*-ontwikkeling voor de gekozen verbeteringen;
- Invoeren van verbeteringen;
- Programmeren van de nieuwe Micros *classes*;
- Rapport schrijven met alle nieuw gemodelleerde Micros *classes*.

3.6 Op te leveren producten

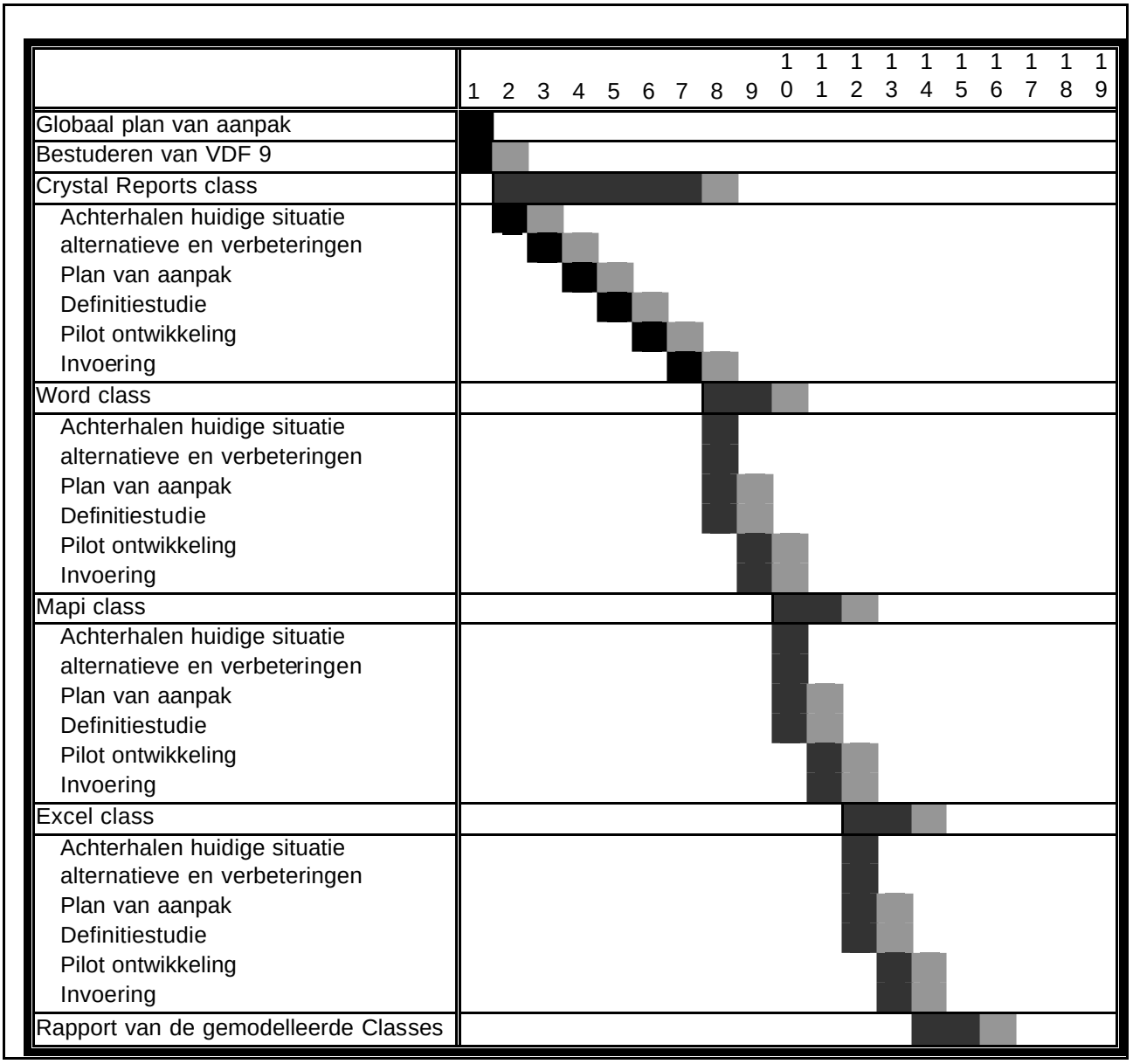
De volgende producten zullen opgeleverd worden:

- Globaal plan van aanpak;
- Plan van aanpak (per 3e partij software);
- Definitiestudie (per 3e partij software);
- Ontwerp van de nieuwe Micros *class(es)* (per 3e partij software);
- De nieuwe Micros *class(es)* (per 3e partij software);
- Rapport van alle nieuw gemodelleerde Micros *classes* voor onderhoud of verdere ontwikkelingsdoeleinden.

4. Globaal plan van aanpak

Het globale plan van aanpak, dat te zien is in de externe bijlage, is bedoeld ter beheersing over het gehele afstudeertraject. Ik heb voor één groot plan van aanpak gekozen en per onderdeel een gedetailleerder plan van aanpak.

Het plan van aanpak is gebaseerd op de opdrachtomschrijving die als basis dient voor het afstudeertraject. Welke derde partij software er gebruikt wordt door de Micros *classes* is achterhaald, door vragen te stellen binnen Micros B.V.. De planning is verdeeld over die vier derde partij software pakketten. Voor het bepalen van de volgorde in het maken van de verschillende derde partij software projecten heb ik "*Worst Things First*" (eerst het moeilijkste gedeelte, daarna de rest) toegepast. Voor het eerste pakket heb ik de meeste tijd ingecalculeerd, omdat ik niet echt goed kan inschatten hoe het Crystal Reports project zal verlopen.



Figuur 4.1 "Planning uit het globaal plan van aanpak"

5. Oriëntatie

Dit hoofdstuk geeft een gedetailleerde beschrijving van de situatie bij aanvang afstuderen. Verder beschrijf ik hier de nieuw geleerde technieken en technologieën en geef ik aan hoe deze zijn onderzocht.

5.1 Visual Dataflex

Bij Micros B.V. wordt op dit moment geprogrammeerd in Visual Dataflex 7. De Micros *classes* zijn grotendeels geprogrammeerd in Dataflex 2 en Visual Dataflex 4 en 7. Primair doel is dan ook het leren programmeren met Visual Dataflex. Aangezien mijn werkstation niet beschikte over een Visual Dataflex ontwikkelomgeving, was mijn eerste taak "*training guide - Visual Dataflex 8.2*" te bestuderen. Dit gaf mij de unieke mogelijkheid om verplicht de taal te onderzoeken, zonder gebruik te maken van de wel bekende *trail and error* methode.

Tijdens mijn leesstudie noteerde ik veel vragen omtrent de taal, welke ik zelf wilde beantwoorden, wanneer ik Visual Dataflex op mijn werkstation kan gebruiken. Op de meeste vragen heb ik wel een antwoord gevonden, hetzij in de *'training guide'* of tijdens het programmeren met de taal. De vragen waarop ik de antwoorden niet zelf kon vinden heb ik besproken met verschillende medewerkers van Micros B.V..

Visual Dataflex is een object georiënteerde taal. De focus van Visual Dataflex is het bouwen van administratieve database applicatie. De applicatie communiceert met de database door middel van datadictionaires.

Met het schrijven van een "*Hello world*" applicatie en een "Celsius naar Fahrenheit" *converter* ben ik begonnen, om zo het 'gevoel' te krijgen van de taal. Tijdens het leren werken met de taal werd het mij nog duidelijker dat Visual Dataflex enorm verschilt ten opzichte van Visual Basic, terwijl mij juist in eerste instantie verteld werd dat het verschil niet zo groot was. Het lezen van waardes tussen verschillende objecten is in Visual Dataflex omslachtiger dan in Visual Basic. Zo moet bij Visual Dataflex kenmerk en uitvoer van een functie, eerst in een variabel gezet worden, voordat het gelezen kan worden. Berekeningen met variabelen kunnen ook niet gemakkelijk plaats vinden. Alles moet in de `"move"` of `"calc"` functie berekend worden en tevens in een andere waarde gezet.

Er is geen *array* datatype aanwezig in Visual Dataflex. In plaats daarvan is er wel een *class* met de naam *array*. Hier kunnen objecten of *subclasses* van gemaakt worden. *Array's* in Visual Dataflex kunnen alleen maar gevuld worden met het datatype *string*. Het is wel mogelijk om van een *string* een `"Zerotype"` (Abstract datatype) te maken. Doormiddel van deze methode is het mogelijk om een *array* te maken die meerdere waardes bevat. Tevens zijn deze waardes niet verplicht van het type *string*.

5.1.1 Classes

Classes in Visual Dataflex werken zoals *classes* in de andere object georiënteerde talen. Een *class* bevat kenmerken (eigenschappen en *properties*) om waardes in op te slaan die voor de totale *class* beschikbaar zijn. Bewerkingen (functies en procedures) worden gebruikt om het gedrag van de *class* te definiëren.

Als er binnen een *class* een variabel gedeclareerd wordt, dan zijn de variabelen een globale variabel. Dit betekent dat dezelfde variabel nergens anders gedeclareerd mag worden maar wel overal gebruikt kan worden. Binnen object georiënteerd programmeren is het niet gebruikelijk om

gebruik te maken van globale variabelen. Hetzelfde geldt voor Visual Dataflex. Het meest verstandig is om binnen een *class* gebruik te maken van kenmerken.

Van een *class* kan een *subclass* gemaakt worden. Een *subclass* is een *class* die dezelfde eigenschappen heeft als de *(super)class* waar hij op gebaseerd is. Als binnen de *subclass* eenzelfde kenmerk en/of bewerking gedefinieerd wordt als in de *superclass* dan wordt het kenmerk of de bewerking in de *superclass* genegeerd.

Wat goed werkt in Visual Dataflex is overerving (*Inheritance*). Het is heel makkelijk *subclasses* te maken van bestaande *classes* van Visual Dataflex. Dit geldt ook voor *classes* die door Micros B.V. zelf ontwikkeld en gedefinieerd zijn.

5.1.2 Objecten

Objecten in Visual Dataflex werken niet exact hetzelfde als in andere talen. Natuurlijk is een object een instantie van een *class* maar in Visual Dataflex is een object meer. Je kunt namelijk in een object bewerkingen en kenmerken definiëren net als bij een *subclass*. Daardoor ben je ook in staat bewerkingen en kenmerken te definiëren van de *class* waar het object is gebaseerd. Het object zal dezelfde bewerkingen en kenmerken van de *class* negeren (net zoals bij de *subclass*) en zal gebruik maken van de bewerkingen en kenmerken in het object.

5.2 IAD

Voor mij is [IAD] een nieuwe ontwikkelmethode, die ik probeer te gebruiken tijdens het afstuderen. Om dit te bereiken heb ik "IAD - Het evolutionair ontwikkelen van informatiesystemen" van R.J.H. Tolido gelezen.

Het boek heb ik bestudeerd op relevante hoofdstukken en heb aantekeningen gemaakt (zoals in het figuur 5.1) over bijvoorbeeld de definitiestudie.

Definitiestudie
1.01 Plan van Aanpak (blz. 148)
Afspraken vastleggen over inhoud omvang van product.
1.02 Definieer ontwikkelscenario (blz.152)
Doelstelling van project, Huidige situatie, Impact op bedrijf
1.03 Voor bereiden Pilotplan-workshop (blz. 157)
Opzetten van pilotplan
1.04 Evalueer Pilot (blz. 160)
Evalueren vorige pilots
1.05 Definieer systeemeisen (blz. 160)
Probleem stelling, Wat moet het doen, prioriteiten stellen die weergave is van opdrachtgever, (Systeem eisen komen van de opdrachtgever)
1.06 Bepaal systeemconcept (blz. 166)
Hoe gaat die het doen een globale oplossing
1.07 Technische structuur (blz. 176)
Op welke manier en voor welke delen van het systeem dit gebruikt wordt.
1.08 Organisatorische inrichting (blz. 180)
Dit moet er in de organisatie gebeuren. Voor en nadelen over invoeren van de pilot bij de organis atie
1.09 Pilotplan (blz. 183)
We kunnen nu het besten zo werken. Prioriteiten lijst van pilots. Systeem eisen,
Aanzet systeem concept, Technische architectuur.

Figuur 5.1 "Aantekening IAD"

Aanvankelijk kon ik geen goed beeld krijgen van [IAD] als ontwikkelmethode voor onderhoud aan *classes*. Maar wat mij vooral aantrok in [IAD] was het bouwen in delen en "Workshops". Het bouwen in delen gebruik ik om één groot probleem (verbetering van de Mcros *classes* voor derde

partij software) onder te verdelen in kleinere problemen (per derde partij software pakket). [IAD] vertelt over *pilots,* wat een “coherente subset van het uiteindelijk beoogde informatiesysteem, die als zelfstandige, bruikbare eenheid kan worden ingevoerd binnen de organisatie” is. Het is dus mogelijk een deelproject verder te verdelen in deelprojecten, genaamd *pilots*. Zo zou een afgeronde *pilot* moeten werken binnen Micros B.V., zelfs zonder dat alle *pilots* afgerond zijn.

Voordat ik [IAD] van Tolido had gelezen dacht ik dat “Workshops” voor de ontwikkelaar een gestructureerde methode van leren werken met een specifiek onderdeel was. Tijdens het lezen van [IAD] kwam ik erachter dat “Workshops” meer bedoeld is, om de wensen van de gebruikers te achterhalen.

Het gebruik van de [IAD] techniek “Workshops” is een handige methode om wensen van gebruikers/opdrachtgever te achterhalen. Deze techniek is hier overbodig omdat mijn doelstelling ‘de werking binnen het huidige systeem’ al bekend is.

5.3 Crystal Reports

Crystal Reports is een programma van voorheen Crystal Decisions (sinds kort Business Objects). Het heeft als doel informatie uit bijvoorbeeld een database overzichtelijk te tonen. Door middel van een aantal gebruiksvriendelijke klikacties is heel snel een rapport te maken. Dit is tevens één van de derde partij applicaties waarmee de Micros applicaties communiceren. Voordat ik de werking van de huidige situatie kon verbeteren, moest ik eerst inzicht hebben in hoe deze applicatie werkte.

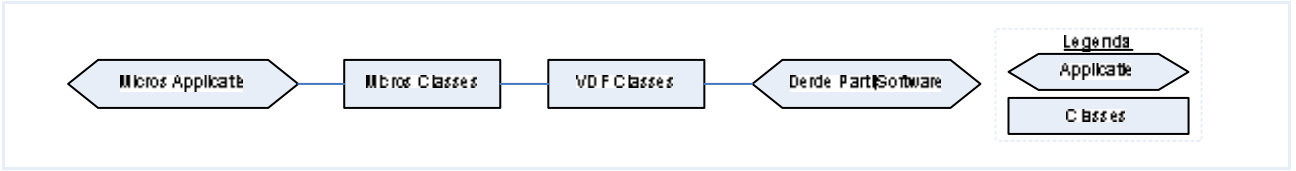
Voor mijn gebruik van Crystal Reports heb ik een Access database met 2 tabellen die een 1 tot N relatie hebben, geschreven. Om de verdere werking van Crystal Reports te begrijpen, heb ik gebruik gemaakt van sorteer- en selectieformules. Een selectieformule bevat de eis waar het overzicht aan moet voldoen. Deze testjes gaven mij een goed inzicht in een aantal nuttige mogelijkheden van Crystal Reports.

5.4 Analyse probleemgebied

Volgens Micros B.V. gaat de communicatie met derde partij software zoals in figuur 5.2. Het probleem is dat, als de derde partij software veranderingen aanbrengt in hun software, dan moeten ook de Micros *classes* worden aangepast.

Figuur 5.2 “Huidige situatie volgens Micros B.V.”

Om ervoor te zorgen dat dit niet gebeurt, stelde Micros B.V. ten doel de communicatie met derde partij software zoveel mogelijk te laten verlopen over standaard Visual Dataflex *classes* (zoals te zien in figuur 5.3).



Figuur 5.3 “Ideale oplossing volgens Micros B.V.”

In de volgende paragrafen geef ik een beeld van de nu ontstane situatie, waarbij het zoeken naar oplossingen moeilijker bleek dan aanvankelijk gedacht.

5.4.1 Documentatie

De meeste documentatie binnen Micros B.V. is documentatie, die geschreven is op applicatie niveau. De documentatie is bedoeld voor de klant: 'Wat wil de klant?' en 'Hoe moet de klant de applicatie gebruiken?'. Er is wel documentatie op Micros *classes* niveau, maar dat zijn gebruikershandleidingen voor programmeurs. Tevens is de meest recente gebruikershandleiding drie jaar geleden voor het laatst aangepast.

De broncode bevat slechts een gering commentaar. In de code, die ik gecontroleerd heb, komen twee namen regelmatig voor. Via interviews had ik vragen willen stellen aan de makers van de verschillende bewerkingen en *classes,* maar die werknemers zijn niet meer in dienst van Micros B.V.. Er is eigenlijk niemand die kan vertellen hoe het werkt(e).

5.4.2 Visual Dataflex bij Micros B.V.

Visual Dataflex heeft een eigen *editor* genaamd IDE (*Integrated Development Environment*), die gebruikt kan worden voor het slepen en visueel aanpassen van objecten. De Micros applicaties kunnen niet goed worden geopend met IDE. De reden hiervoor is dat Micros applicaties en Micros *classes* in een oudere versie zijn geprogrammeerd en niet correct zijn gestructureerd volgens de IDE. De compiler heeft echter geen enkele moeite met de Micros applicaties. Om niet verplicht te werken met de IDE heeft Micros een eigen programma geschreven dat op basis van de compiler de gehele of een gedeelte van een Micros applicatie compileert. Dit is dus ideaal voor het ontwikkelen en testen van nieuwe functionaliteiten. De naam van deze applicatie is "Launch". Het compileren met de Launch duurt tussen de 3 á 5 minuten.

Eén van de redenen dat het compileren zo lang duurt, is dat een groot aantal externe bestanden wordt ingesloten (met de *use* functie in Visual Dataflex).

Naam	Extensie
View bestand	VW
ReportView bestand	RV
Custom Package/Class bestand	PKG
Selection List bestand	SL
Dialog bestand	DG

Figuur 5.4 "Micros Visual Dataflex extensies"

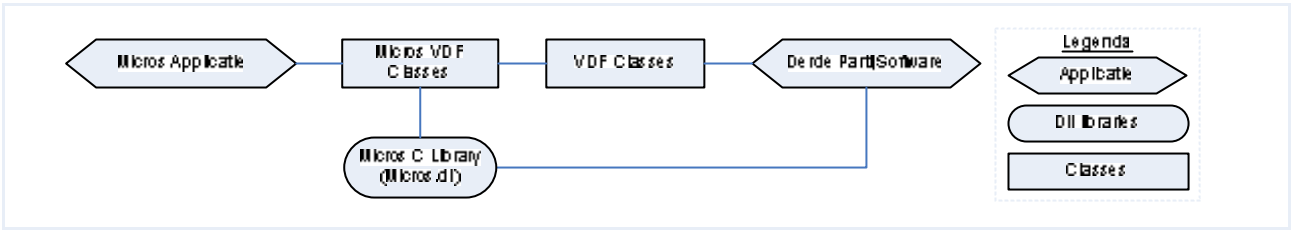
Alle bestandstypen, zoals in figuur 5.4, worden door de Micros Applicatie gebruikt. De PKG bestanden (dit zijn er veel), ook wel *packages* genoemd, bevatten de *classes,* die door Data Access geschreven zijn. Micros B.V. maakt ook gebruik van de *packages,* om hun eigen *classes* en objecten in op te slaan. De meeste *package* bestanden van Micros B.V. behoren tot de groepering 'Micros *classes'*.

5.4.3 Micros classes

Micros B.V. heeft verschillende *classes* zelf ontwikkeld om snel meer functionaliteit te bieden binnen hun applicaties. De Micros *classes* zijn in twee verschillende programmeertalen geprogrammeerd.

Een gedeelte van deze *classes* (soms ook een functie zonder deel uit te maken van een *class*) zijn geprogrammeerd in Borland C. Bij de extra functionaliteiten, die hierdoor worden aangeboden, moet gedacht worden aan *arrays*, lange *strings* (“LongText” of “LT”), geheugen allocatie, communicatie met een DDE macro (hier later meer over) en andere functies. De extra functionaliteit die gecreëerd is, was op dat moment niet mogelijk met Visual Dataflex en in sommige gevallen nog steeds niet mogelijk.

Het overgrote deel van de Micros *classes* is geprogrammeerd in Visual Dataflex. De meeste van deze *classes* zijn *subclasses* van standaard Visual Dataflex *classes* of van andere Micros *classes*. De Visual Dataflex Micros *classes* vormen ook de interface met de Micros C *library*.



Figuur 5.5 “Micros classes gesplitst”

Als je deze informatie samenvoegt met de informatie van figuur 5.3, ontstaat de structuur van figuur 5.5. Te zien is dat de Micros *classes* gesplitst zijn en dat er nu communicatie plaatsvindt tussen derde partij software en de C Borland Micros *Library*. De C Borland Micros *library* bestaat uit meer dan 800 functies.

Aangezien de hiervoor genoemde *classes* niet alleen maar uit *classes* bestaan, maar ook objecten, is hieronder een overzicht te zien van het aantal *classes* en objecten voor de Micros Applicatie KAS (Cursus Administratie Systeem)

	Classes	Objecten
Micros VDF Class	280	302
VDF Classes	363	217
Micros Applicatie (KAS)	148	7325

Figuur 5.6 “Verdeling classes en objecten”

5.4.4 Crystal Reports

De onderzoekfase om de huidige werking in kaart te brengen is één van de eerste punten, die aangegeven wordt in het plan van aanpak. Tijdens het bestuderen van Visual Dataflex 7, samen met Crystal Reports, ontdekte ik dat er verschillende *classes* zijn, die gebruik maken van functies van het *library* bestand Crpe32.dll. Crpe32.dll is de Crystal Reports *print engine*. De Crystal Reports *print engine* wordt gebruikt voor communicatie met Crystal Reports. Visual Dataflex spreekt dus niet direct met Crystal Reports, maar met een API van Crystal Reports de *print engine*.

Het gebruik van deze Crystal Reports *print engine* vindt plaats op verschillende niveaus binnen een Micros Applicatie. Hieruit valt af te leiden dat er geen zichtbare structuur aanwezig is wie of wat de Crystal Reports *print engine* functies aanroept binnen in de Micros *classes*.

Door een tekort aan documentatie en structuur was het bijna onmogelijk om vast te stellen wat de exacte werking is van de verschillende bewerkingen binnen de *classes*, die bedoeld zijn voor directe en indirecte communicatie met Crystal Reports (zie voor resultaten paragraaf 5.4.6). Tevens wordt direct gebruik gemaakt van een, op dat moment, onbekende functie uit de Borland C

Micros *classes*. Dit leidde er toe dat de onderzoekfase van richting veranderde en wel van het onderzoeken van de huidige situatie, naar het onderzoeken van het extern aansturen van Crystal Reports met Visual Dataflex 7. Vooral omdat ik geen inzicht had hoe de huidige situatie op dat moment werkte. In overleg met mijn bedrijfsmentor is toen gekozen voor deze wending in de onderzoeksfase.

5.4.5 Werking van Crystal Reports Class

Door gebruik te maken van de standaard Visual Dataflex *class* voor communicatie met Crystal Reports, plus de daarbij behorende documentatie, die door Data Access gemaakt is, heb ik onderzocht, hoe je kunt communiceren met Crystal Reports vanuit Visual Dataflex 7.

Door mijn eerste ervaringen met Crystal Reports wist ik wat het pakket kon doen. Door gebruik te maken van de standaard Visual Dataflex *class* probeerde ik hetzelfde te bereiken voor communicatie met Crystal Reports.

Dit is dan ook de basis van het onderzoek. Het onderzoek naar communicatie met Crystal Reports heb ik zo gestructureerd mogelijk aangepakt, met name door gebruik te maken van prototypes. Volgens [IAD] is een prototype een beperkte versie van een *pilot*. In het algemeen is het doel van de experimenten: het verminderen van risico's. Dat is in dit stadium ook mijn insteek. Experimenteren met het gebruik van deze *class* om later, met minder risico, te kunnen bewijzen of dit een haalbaar traject is.

Figuur 5.7 bevat een lijst met wat ik wilde bereiken, op het gebied van communicatie via de standaard Visual Dataflex *class* voor communicatie met Crystal Reports.

Wat wil ik	Gelukt
Verbinding opzetten	Ja
Rapport uitvoeren	Ja
Verbinding verbreken	Ja
Bestandsnaam veranderen	Ja
Sqlquery zien	Nee
Selectie criteria zien	Ja
Selectie criteria aanpassen	Ja
Selectie criteria verwijderen	Ja
Sorteer volgorde zien	Ja
Sorteer volgorde aanpassen	Ja
Sorteer volgorde verwijderen	Ja
Groepen zien	Ja/Nee (Kreeg alleen nummers geen namen)
tabellen zien	Ja
kolomen zien	Nee

Figuur 5.7 "Crystal Reports communicatie onderzoek"

5.4.6 Werking van MsCrystalReports

Toen het werken met de standaard Crystal Reports *class* in Visual Dataflex mij duidelijk werd, ben ik mijn onderzoek naar de huidige werking van de Micros Crystal Reports *class* wederom gestart, ditmaal door gebruik te maken van prototypes. In het bijzonder heb ik de `"MsCrystalReports"` *class* onderzocht. Dit is een *(sub)class,* die gebaseerd is op de standaard Visual Dataflex *class* voor communicatie met Crystal Reports. Deze *class* is ook de basis van de communicatie tussen de Micros *classes* en Crystal Reports.

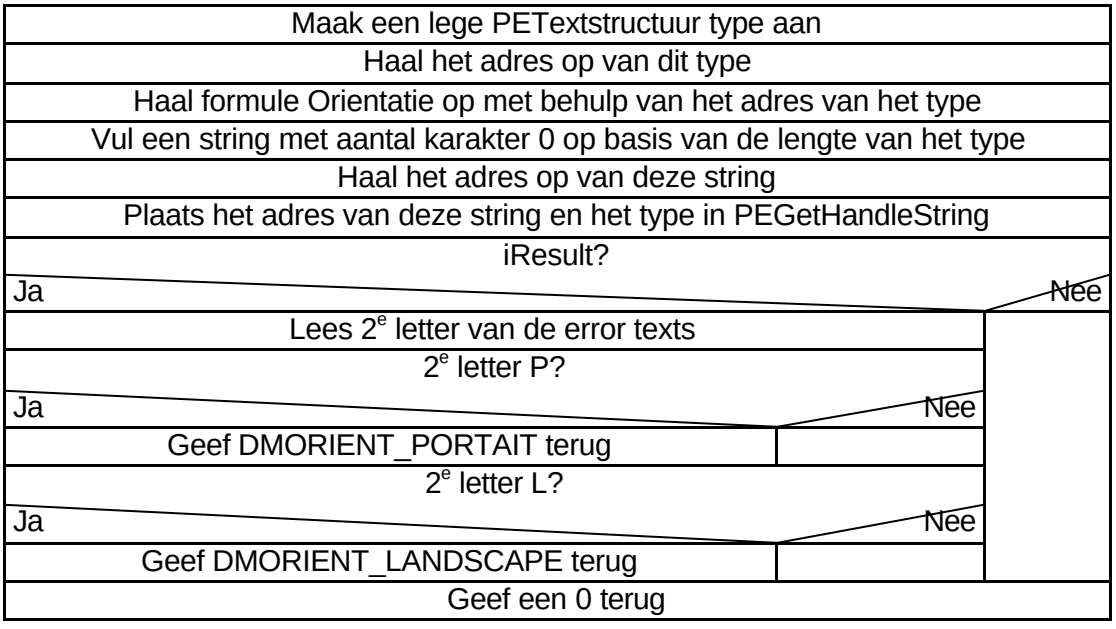
```
FillType PETextStructure With 0 To sPETextStructure
Get PrintJob To iPrintJob
GetAddress Of sPETextStructure To pPETextStructure
If (PEGetFormula(iPrintJob, "Orientatie", pPETextStructure,  pPETextStructure+4)) begin
end

GetBuff From sPETextStructure At PETextStructure.textHandle To pTextHandle
GetBuff From sPETextStructure At PETextStructure.textLength To iTextLength
Move (Repeat(Character(0),iTextLength+1)) To sErrorText
GetAddress Of sErrorText To pErrorText
Move (PEGetHandleString(pTextHandle,pErrorText,iTextLength)) To iResult

If iResult begin
        Move (Cstring(sErrorText)) To sErrorText
        uppercase sErrorText
        showln "Orientatie(2): [" sErrorText "]"
        if sErrorText eq 'P' function_return DMORIENT_PORTRAIT
        if sErrorText eq 'L' function_return DMORIENT_LANDSCAPE
end
Function_Return 0
```

Figuur 5.8 "MsOrientatie uit MsCrystal.pkg"

De broncode, zoals in figuur 5.8, bestuderen en begrijpen is nu aan de orde. Voor iedere bewerking, ieder kenmerk en object dat gebruikt wordt binnen een bewerking van de Micros Crystal Reports *class* heb ik ook gekeken waar de declaratie zich bevindt. Het gebruik van alle kenmerken en bewerkingen die gedeclareerd worden in de Micros Crystal Reports *class* is ook onderzocht. Om alles duidelijk in kaart te brengen, heb ik de gevonden werking van de bewerkingen in een PSD gezet.



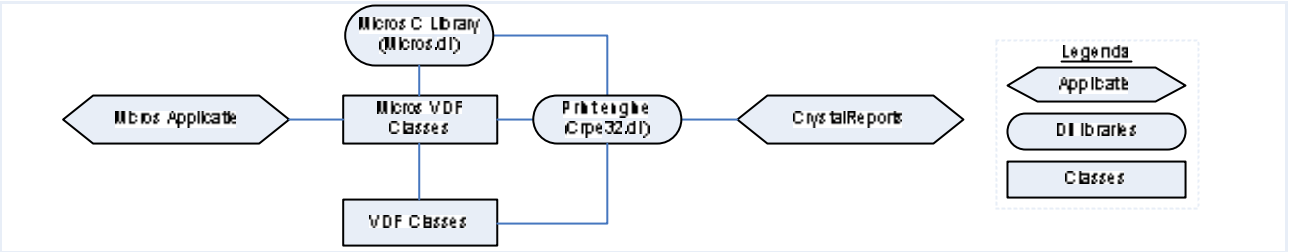
Figuur 5.9 "MsOrientatie PSD uit het MsCrystal werkdocument"

Behalve de PSD's heb ik ook een lijst per bewerking, met de bewerkingen en kenmerken, die aangeroepen worden binnen de bewerking. Dit om te achterhalen welke bewerkingen aangetast zullen worden, als grote aanpassingen plaatsvinden aan de bewerking in de Micros Crystal Reports *class*.

Naam	Type	Class	Bestand	Maker	Wat doet het
msRapportAfdrukken	Procedure	MsCrystal	MsCrystal.pkg	MS	
Reporttitle	Functie	Crystalreport	crystalreport.pkg	VDF	Haal titel op van het rapport

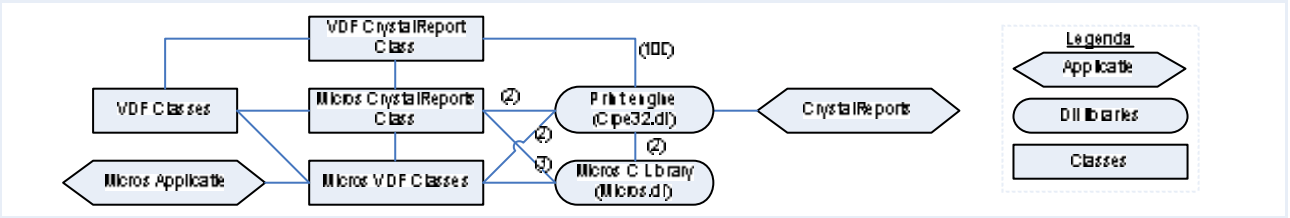
Figuur 5.10 "msQuickprint externe aanroepen uit het MsCrystal werkdocument"

Figuur 5.5 schetst een globaal schema van communicatie met derde partij software, waarbij de Micros *classes* gesplitst zijn. Een overeenkomstig schema met Crystal Reports, gesplitst in Crystal Reports en Crystal Reports *print engine*, zien we in figuur 5.11.



Figuur 5.11 "Communicatie via de print engine"

Figuur 5.11 verder uitdiepend, door de Micros Crystal Reports *class* te splitsen van de andere Micros *classes* en door de standaard Crystal Reports *class* te splitsen van de Visual Dataflex *classes*, geeft een communicatiebeeld zoals in figuur 5.12



Figuur 5.12"Huidige situatie communicatie met Crystal Reports"

Wat opvalt is de hoeveelheid communicatielijnen (aangegeven met een getal) naar de *print engine*. Voor de ideale toekomstige situatie wil je communicatie met de *print engine* slechts laten verlopen over de standaard Visual Dataflex *class* voor Crystal Reports. De gegevens die ik tot dusver gevonden heb, zijn gebundeld in een werkdocument MsCrystal.doc.

5.4.7 RDC

Een externe aansturing van een Crystal Reports rapport is onder andere mogelijk via de Crystal Reports *print engine*, zoals hiervoor te lezen is. Crystal Decisions, de toenmalige makers van Crystal Reports, ondersteunt het gebruik van de *print engine* niet volledig meer. Dit komt omdat Crystal Reports haar externe communicatie meer via het .Net *framework* wil laten lopen (tegenwoordig wordt Crystal Reports meegeleverd met .Net).

Crystal Decisions heeft RDC (Report Designer Component) ontworpen, om te werken in de .Net omgeving. Data Access heeft een programma gemaakt voor gelimiteerd aansturen van COM objecten vanuit Visual Dataflex. Een verdieping met behulp van prototypes in het gebruik van RDC met Visual Dataflex blijft voor mij ondoorzichtig.

Data Access biedt geen ondersteuning op het gebied van RDC. Data Access doet dit wel voor de Crystal Reports *print engine*. Verder zou de nieuwe Micros *class* geen gebruik meer maken van standaard Visual Dataflex *classes* voor communicatie, afwijkend dus van mijn afstudeeropdracht. Dit is de reden dat Micros B.V. en ik ervoor gekozen hebben ons te beperken tot de standaard Crystal Reports *class* voor communicatie met Crystal Reports.

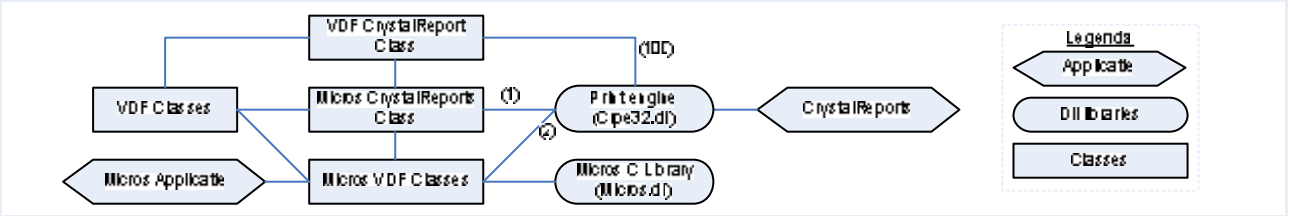
5.4.8 Weghalen van de C componenten

Om een ideale situatie (figuur 5.2) te creëren, moet ervoor gezorgd worden dat de Micros C *library* en de Crystal Reports *print engine* niet meer met elkaar communiceren. Voordat ik een oplossing voor dat probleem kan aandragen, moet ik eerst weten waarom communicatie via deze methode noodzakelijk is. Het antwoord schuilt in het type communicaties. Er wordt namelijk gebruik gemaakt van het geheugen om lange *strings* naar het Crystal Reports rapport te sturen. In oudere versies van Visual Dataflex was het niet mogelijk om *strings* van meer dan 255 karakters te maken. Micros B.V. heeft echter een eigen type ontwikkeld, om *strings* groter dan 255 karakters te gebruiken.

Nu we het ontstaan begrijpen, kan er gekeken worden naar een mogelijke oplossing. Er zijn drie vragen die beantwoord moeten worden, om het gebruik van de Micros C *library* uit de Micros Crystal Reports *class* te halen:

1. Kan de huidige Visual Dataflex een *string* van meer dan 255 karakters aan?
2. Kan de plek gevonden worden, waar de selectieformules in het geheugen worden gezet?
3. Kunnen de selectieformules meegegeven worden aan de standaard Crystal Reports *class*?

Voor het zoeken naar de antwoorden heb ik gebruik gemaakt van prototypes. Het antwoord op deze drie vragen luidt 'Ja'. Figuur 5.13 brengt dat in beeld.



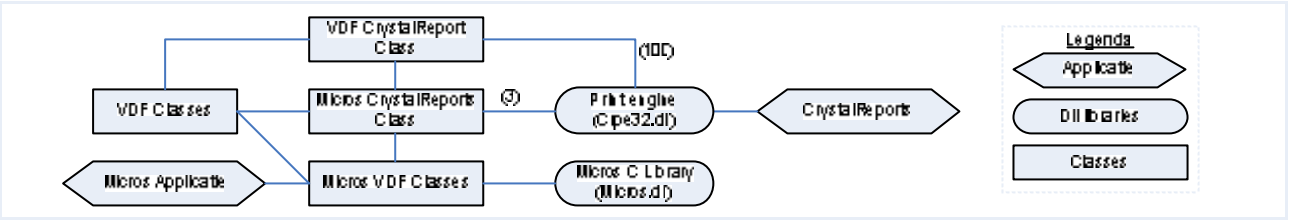
Figuur 5.13 "Loskoppeling Micros C library"

5.4.9 Overige communicatie

Zoals figuur 5.13 al aangeeft, is er nog steeds communicatie met de *print engine*, vanuit de Visual Dataflex Micros *classes* en de Micros Crystal Reports *class*. De directe communicatie met de *print engine* is nodig omdat de standaard Visual Dataflex Crystal Reports *class* niet de totale functionaliteit van de *print engine* benut. Drie mogelijke opties om te vervolgen doen zich voor:

- Toevoegen van *print engine* aanroepen aan de standaard Visual Dataflex *class* voor communicatie met Crystal Reports;
- Verplaatsten van het aanroepen van de *print engine* naar de Micros Crystal Reports *class*;
- Alles laten zoals het is.

De eerste optie is geen goede: er moet te allen tijde een scheiding blijven tussen de *classes* van Data Access en Micros B.V.. De tweede optie heeft mijn voorkeur: door het verplaatsen van het aanroepen van de *print engine* kun je alle communicatie clusteren in één Micros *class*. De volgende toekomstige situatie, zoals in figuur 5.14, kan dan mogelijkerwijs gecreëerd worden.



Figuur 5.14 "Toekomstige Situatie"

5.4.10 Microsoft Word

Om brieven te genereren, maken de Micros applicaties gebruik van Microsoft Word. Het opzetten van een verbinding tussen Word en de Micros applicatie gebeurd met *Dynamic Data Exchange* (DDE). *Dynamic Data Exchange* is bedoeld voor het uitwisselen van informatie tussen verschillende programma's. Het gebruikt hiervoor het geheugen als gemeenschappelijk medium voor het uitwisselen van die informatie.

Micros B.V. maakt gebruik van de Micros C *library* voor het opzetten en verbreken van de DDE verbinding. Tussen het opzetten en verbreken van die verbinding, worden ook nog twee DDE commando's aangeroepen. Het eerste commando is het openen van een sjabloon met daarin Macro, het tweede is het uitvoeren van die Macro.

De Macro communiceert met de Micros C *library*. De Macro van het Word sjabloon vraagt iets en de Micros C *library* geeft antwoord. Wanneer het antwoord wordt verstuurd, wordt de volgende vraag door het Word Sjabloon macro teruggegeven.

Het probleem met communicatie tussen Word en Visual Dataflex is, dat deze communicatie in twee richtingen verloopt. Terwijl de Word *class* van Visual Dataflex slechts in één richting werkt, namelijk informatiestroom naar Word (ontvangen is niet mogelijk). Dit zal de reden zijn geweest dat Micros B.V. gebruikt maakt van de Micros C *library* voor de communicatie met Word.

De manier waarop Micros werkt maakt het onmogelijk om via de standaard Visual Dataflex *class* met Word te communiceren.

5.4.11 Excel

Een andere derde partij software waar Micros B.V. gebruik van maakt is Microsoft Excel. Micros B.V. maakt gebruik van Crystal Reports om een Excel sheet te exporteren. Binnen Visual Dataflex is een *class* aanwezig om te werken met Excel. Door testen met deze *class* was het mogelijk om Excel op te starten en bestanden te openen. Het was echter niet mogelijk om waardes te lezen in een cel van Excel en om informatie naar een cel te schrijven, terwijl de functies wel aanwezig zijn binnen de standaard functies van Visual Dataflex.

5.4.12 MAPI

MAPI staat voor *Messaging Application Program Interface* en is een Microsoft Windows programma interface dat het mogelijk maakt om binnen een applicatie e-mailen te verzenden. MAPI bestaat uit standaard functies die in een *library* staan. (vergelijk de Crystal Reports *print engine*).

De applicaties van Micros B.V. maken ook gebruik van de MAPI voor het versturen van meerdere e-mail berichten. De MAPI wordt niet direct aangesproken maar door de Micros C *library*. Om te achterhalen hoe de MAPI werkt in Visual Dataflex heb ik verschillende kleine prototypes gemaakt.

In eerste instantie wilde de MAPI niet werken. Reden: mijn werkstation was niet goed geconfigureerd om met de MAPI te werken.

Om het gebruik van de MAPI in de Micros applicatie te onderzoeken, heb ik eerst de toepassing, om gebruik te maken van de Micros Applicatie met betrekking tot e-mailverkeer, bestudeerd. Vervolgens heb ik gezocht in relevante delen van de Micros applicatie en Micros *classes* voor het gebruik van de MAPI.

Zo ontdekte ik dat het mogelijk is om het e-mailverkeer te laten verlopen via de standaard *classes* van Visual Dataflex, maar dat de Visual Dataflex *class* niet voldoende is, om de huidige methode te vervangen. Het is bijvoorbeeld niet mogelijk om te werken met een ander profiel voor het versturen van e-mailen. Wat wel mogelijk is met de MAPI en wat ook gebeurt via de Micros C *library*.

5.5 Richtlijnen

De voor de toekomst bedoelde richtlijnen zijn in eerste instantie bedoeld om een gestructureerde wijze van programmeren en een soort van huisstijl te creëren, zodat broncodes op elkaar gingen lijken. Later kwam ter sprake dat de richtlijnen een beschrijving zijn van een standaard Windows interface.

Samen met mijn bedrijfsmentor ben ik tot de conclusie gekomen dat de richtlijnen meer bedoeld zijn voor onderhoud. Onderhoud kan alleen maar gestructureerd plaatsvinden, als er documentatie is over het onderdeel waar onderhoud op gepleegd wordt en er een geschiedenis is van reeds gepleegd onderhoud. Zoals hiervoor reeds beschreven, is er geen documentatie en ook geen exacte methode voor het ontwerpen van *classes*. Er is wel een methode voor het schrijven van een gebruikershandleiding, alle zijn op identieke wijze gestructureerd.

Dit houdt in dat er documentatie moet komen over reeds ontworpen onderdelen. In ieder geval moet er een gebruikershandleiding zijn voor iedere *class,* zodat die correct kan worden toegepast. Alles wat ik ontwikkel zal een ontwerp en een gebruikershandleiding hebben.

Onderhoud van verschillende producten gaat ook hand in hand met versiebeheer. Er moeten dus richtlijnen komen zodanig dat er versiebeheer plaats kan vinden.

6. Dropdown menu

Om nog meer inzicht te krijgen in de taal en de IDE, heb ik een eigen *class* ontwikkeld, die door IDE geaccepteerd en gebruikt kan worden in een Visual Dataflex 7 omgeving. Dit is niet het enige doel van deze *class*. De *class* geeft ook inzicht in een ontwikkeltraject, documentatie, codering, gebruikershandleiding en versiebeheer voor de ontwikkeling van een Micros *class*.
Dit traject kan gebruikt worden als richtlijn voor een ontwikkeltraject voor Micros *classes*.

6.1 Ontwerp

Voor het ontwikkelen van een Micros *class* maak ik gebruik van de meest directe methode. Micros B.V. en haar medewerkers behoeven geen complex ontwerp.

6.1.1 Doel

Een belangrijk deel van het ontwerp is de argumentatie: 'Waarom iets nieuws ontwikkelen?'. Minder van belang is dan welke methode er gebruikt wordt. Meestal wil je iets nieuws ontwikkelen, vanwege een probleem met een bestaande situatie. Zo ook hier. Mijn handelwijze is het vaststellen van het probleem (lees: huidige situatie) en komen tot voorstellen ter oplossing van dat probleem (lees: doelstelling), zoals hierna beschreven is.

Binnen VDF is er comboform Class. Comboform's hebben het uiterlijk van een dropdown selectie. De waarde die je kunt selecteren wordt ook gezien als de waarde van de selectie optie. Als je een gebruiker de optie geeft om iets te selecteren moet er voor de gebruiker zinnig informatie staan waar die duidelijk uit kan selecteren.

Figuur 6.1 "Probleem"

Aansluitend heb ik in het ontwerp beschreven, wat nodig is om een mogelijke oplossing te realiseren: een lijst van benodigdheden (lees: globale structuur). Figuur 6.2 schetst de benodigdheden voor ontwikkeling van *dropdown class*.

De functies, procedures en objecten nodig om dit te realiseren:

- *Toevoegen van opties aan de selectie;*
- *Lezen van huidige waarde van de selectie;*
- *Lezen van een specifieke waarde;*
- *Verwijderen van alle waardes;*
- *Array van een datatype met 2 Strings. Echte waarde en de te laten zien waarde.*
    - *Toevoegen van waardes aan Array;*
    - *Lezen van specifieke waarde;*
    - *Overzicht van alle waardes in het Array;*
    - *Leeg maken van het Array.*
- *Toevoegen aan IDE interface voor gebruiksvriendelijkheid.*

Figuur 6.2 "Benodigdheden"

6.1.2 Classes

Als de benodigdheden bekend zijn, kan het ontwerp van de *class* gemaakt worden. De beste aanpak lijkt mij om scheidingen te maken in types, kenmerken en bewerkingen, die binnen de *class* gebruikt worden.

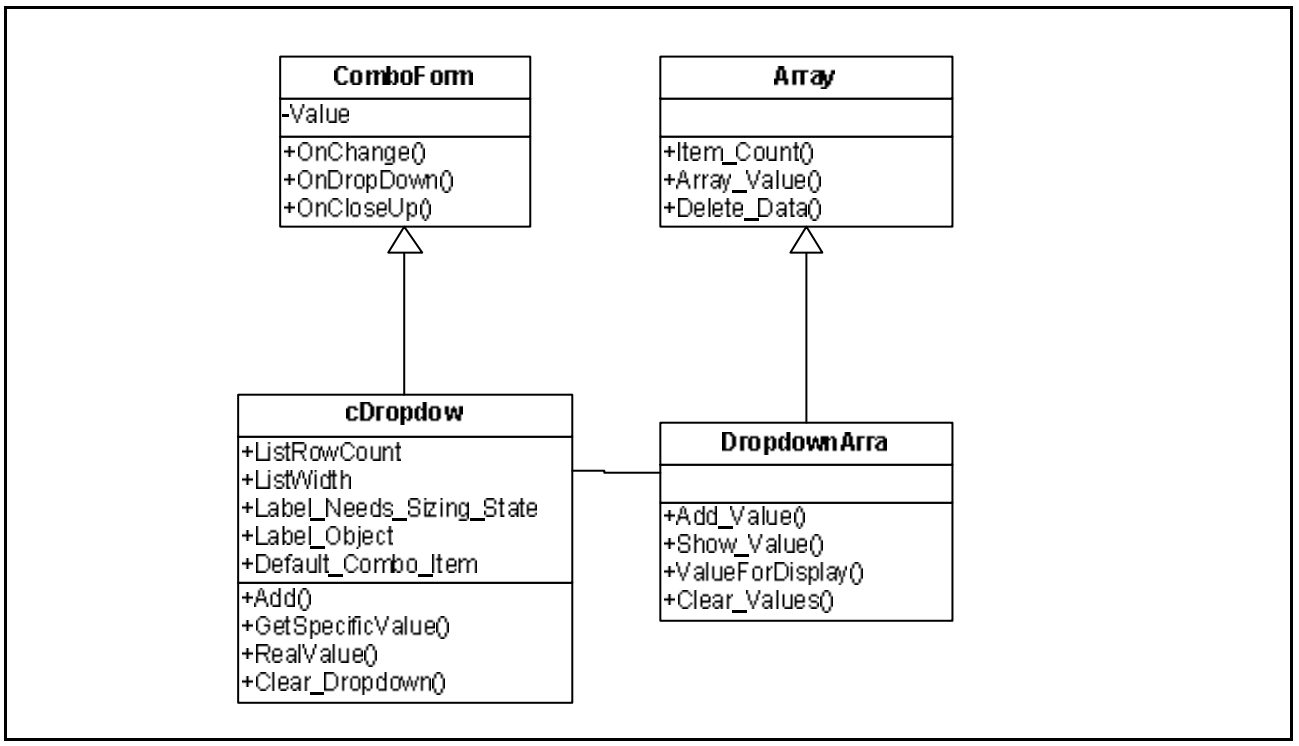
Bij types, die gedefinieerd worden binnen de nieuwe *class,* is het verstandig om te vertellen 'waarom' dit type nodig is. Hetzelfde geldt eigenlijk voor kenmerken en bewerkingen.

De beschrijving van bewerkingen is wat complexer. De meeste bewerkingen bestaan niet uit meer dan één regel. Een beschrijving is dan niet altijd duidelijk genoeg, extra uitleg is dan wenselijk. Voor het ontwerp dat ik gemaakt heb voor de *dropdown class* heb ik gebruik gemaakt van PSD's. PSD is niet de enige techniek die gebruikt wordt voor extra uitleg; andere mogelijkheden zijn bijvoorbeeld sequentie diagrammen, flowcharts en objectmodellen.

Het belangrijkste, naar mijn mening, is dat derden de beschrijving ook begrijpen en kunnen hanteren, soms met extra uitleg maar meestal zonder uitleg. Door de ideeën van de ontwerper te kennen, zal dit later helpen bij onderhoud op het desbetreffende onderdeel.

6.1.3 Class Diagram

Doordat we werken in een objectgeoriënteerde omgeving is het verstandig om de *class* te tonen in relatie met andere *classes,* zoals hieronder. Als er genoeg *class* diagrammen gemaakt worden zal het mogelijk zijn om een totaal beeld te creëren van alle Micros *classes* in relatie met de standaard Visual Dataflex *classes*.



Figuur 6.3 "Class diagram"

6.2 Codering

Op dit moment worden de wijzigingen en versienummers niet opgenomen in de broncode van een desbetreffende *class*. Op basis van de kop die Data Access aan hun broncode geeft kan Micros B.V. hetzelfde doen voor haar *classes* (zoals te zien in figuur 6.4). Dit als mogelijke *header* standaard voor Micros B.V..

```
// ************************************************************************ //
// **                                                                    ** //
// ** File       : dropdown.pkg                                         ** //
// **                                                                    ** //
// ** Author     : Dennie Prins                                         ** //
// **            : Micros B.V.                                          ** //
// **            : 18 december 2003                                     ** //
// **                                                                    ** //
// ** Purpose    : This package contains the creation of a dropdownmenu.** //
// **                                                                    ** //
// ** Version    : 1.0.0.1                                              ** //
// **                                                                    ** //
// ** Design     : dropdown.doc                                         ** //
// **                                                                    ** //
// ** Status     : Tested, No bugs found.                               ** //
// **                                                                    ** //
// ** 18-12-2003 : Header created                                       ** //
// **                                                                    ** //
// ** 18-12-2003 : Clear Dropdown added to clear dropdown of all values ** //
```

Figuur 6.4 "Broncode kop"

Commentaar in de broncode is tevens een belangrijk punt. Broncode is toch één van de eerste stukken documentatie, waarin gelezen wordt tijdens het zoeken naar antwoorden. Commentaar schept dus duidelijkheid in de broncode. Dit gebeurt al binnen de Micros *classes* maar naar mijn mening nog niet genoeg.

```
//For the current value displayed in the comboform get the corresponding value.
function RealValue returns String
   string sValue
   get Value to sValue //Get comboform value
   get ValueForDisplay of InternalDropdownArray sValue to sValue //Get corresponding value
   function_return sValue
```

Figuur 6.5 "bewerking RealValue uit dropdown.pkg"

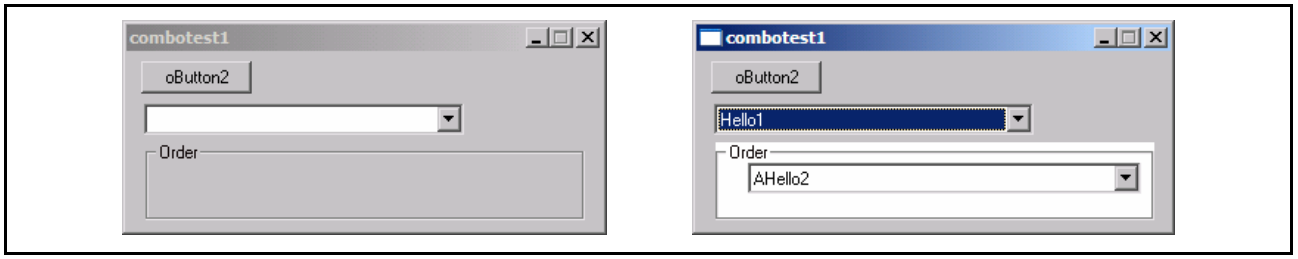
6.3 IDE interface

Als laatste punt in figuur 6.2 staat "Toevoegen aan IDE interface voor gebruiksvriendelijkheid". Omdat op dit moment sommige Micros applicaties niet werken binnen de IDE, heb ik bekeken of dit wel mogelijk zou zijn voor deze *class* en andere, die nog ontwikkeld zullen worden.

Micros B.V. maakt al gebruik van integratie in IDE, maar dat functioneert niet. Er werd mij geadviseerd geen pakketten te openen in IDE, data gaat daarmee verloren, omdat de IDE gegevens opslaat, als je naar een view wilt gaan.

Wat ik probeerde te bereiken is het *dropdown* menu toe te voegen aan de IDE en te bekijken of het haalbaar is dit te doen voor de door mij te creëren nieuwe *classes*. Het toevoegen van het *dropdown* menu moest via een aantal toevoegingen aan een globaal bestand en het toevoegen van een aantal kleine bestanden. Met wat moeite is dit uiteindelijk toch gelukt.

Aangezien binnen in de Micros *classes*, *classes* zijn die meerdere objecten bevatten, is het logisch te kijken of je een *class* kan definiëren, met daarin objecten van een zelfgedefinieerde *class*. Dit blijkt mogelijk, er kleeft alleen één nadeel aan, van belang bij het ontwerpen van schermen. Het ontwerpscherm (figuur 6.6 links) bevat niet alle informatie, die getoond wordt in het resultaatscherm (figuur 6.6 rechts).



Figuur 6.6 "Ontwerp en resultaat scherm"

6.4 Gebruikershandleiding

Ter verduidelijking omtrent het gebruik, is een gebruikershandleiding geschreven voor de programmeurs. Het is in twee stukken verdeeld, gebruik en definitie. 'Definitie' geeft een beschrijving van de *class*, het aanroepen van de *class* en wat die aanroep doet. 'Gebruik' verschaft, met name de programmeur, nuttige informatie over de *class.*

7. Crystal Reports

Dit hoofdstuk beschrijft het ontwikkeltraject voor de nieuwe Micros *class* voor de communicatie met Crystal Reports, van definitiestudie tot beschrijving van de invoering. Tevens de kleine aanpassingen, die aan het huidige systeem getroffen worden, om de nieuwe *class* volledig te laten functioneren.

7.1 Definitie studie

In de definitiestudie wordt het doel van het systeemdeel (de nieuwe *classes*) geanalyseerd, evenals de beperkingen. Door middel van een aantal stappen, welke voorgeschreven zijn binnen [IAD] over de definitiestudie, heb ik deze opgesteld voor het Crystal Reports project.

Hieronder de stappen, die genomen zijn, om vanuit de huidige situatie, de toekomstige te realiseren en welke veranderingen dit tot gevolg kan hebben binnen de organisatie. De stappen zullen een sequentieel beeld geven over hoe er tot het resultaat is gekomen, maar vooral ook de problemen en bijzonderheden die ik ben tegengekomen.

7.1.1 Plan van aanpak

De definitiestudie nam een aanvang met het schrijven van een plan van aanpak, omtrent het ontwikkeltraject voor het Crystal Reports. Het plan van aanpak aan het begin van de definitiestudie is een voorschrift van de ontwikkelmethode [IAD]. Definitiestudie en plan van aanpak zijn gebaseerd op de keuze die Micros B.V. gemaakt heeft om gebruik te maken van *print engine* en niet van RDC (Zie paragraaf 5.4.7). De afbakening is een reflectie van de gemaakte keuze.

Het verbeteren, verminderen en clusteren van de operatie die direct communiceren met Crystal Reports. Hoofdzakelijk gaat het hierbij om de volgende punten:

- Het verbeteren van de code van de Micros Crystal Reports Class;
- Het clusteren van communicatie met Crystal Reports zoveel mogelijk binnen één Class;
- Het creëren van richtlijnen voor het gebruik van de nieuwe Classes om ervoor te zorgen dat de nieuwe Classes volgens een bepaalde standaard worden aangepast en toegepast;
- Toevoegen van extra functionaliteit voor later gebruik.

Figuur 7.1 "Afbakening Crystal Reports opdracht"

De planning die ik voor het Crystal Reports traject heb gemaakt, is gebaseerd op een doorlooptijd van vijf werkweken. De onderzoekfase voor Crystal Reports, niet tot de vijf werkweken behorend, staat beschreven in het hoofdstuk 'Oriëntatie' van dit eindverslag.

7.1.2 Definieer ontwikkelscenario

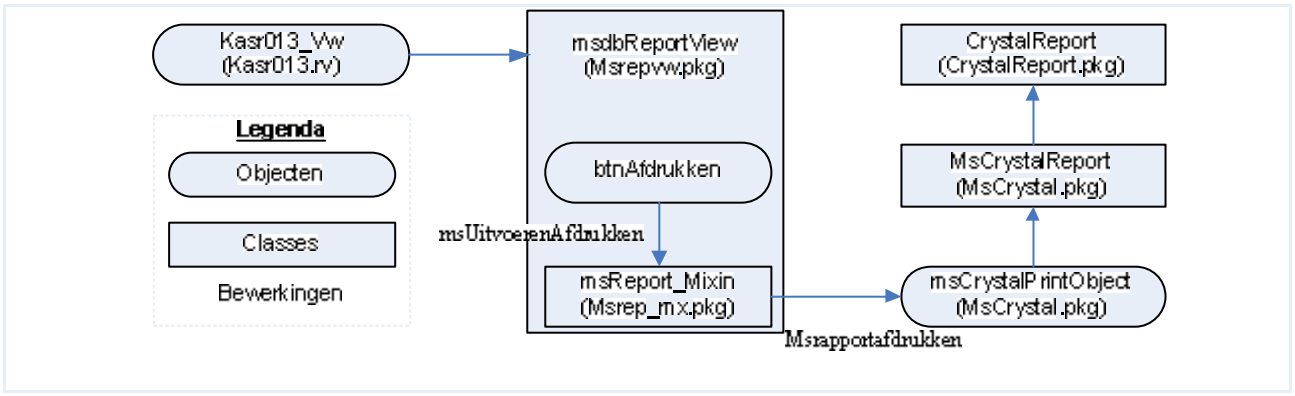
In deze paragraaf van de definitiestudie behandel ik de doelstelling, huidige situatie en verwachte impact. De doelstelling van het Crystal Reports project is de wensen (zoals hieronder te zien is) die Micros B.V. heeft voor het Crystal Reports project te realiseren.

Het verminderen van de afhankelijkheid van Crystal Reports, door het vervangen van de relatie tussen de nuttige Classes van Micros en Crystal Reports met een relatie tussen de Classes van Micros en een nieuwe Class die communiceert met de standaard VFD Class voor communicatie met Crystal Reports. Het creëren van richtlijnen voor het gebruik van de nieuwe Classes om ervoor te zorgen dat de nieuwe Classes volgens een bepaalde standaard worden aangepast en toegepast.

Figuur 7.2 "Doelstelling uit de definitie studie"

In de definitiestudie (paragraaf 3.2.2 "Algemene Communicatie" in de definitiestudie in de externe bijlage) heb ik een communicatieschema gemaakt dat lijkt op figuur 5.12 van dit document. Het geeft een beeld van de relaties van de verschillende termen die gebruikt worden in dit project. Deze termen zijn: *Print engine* (Crpe32.dll), Micros C *Library* (Micros.dll), Micros Crystal Reports *class* en Visual Dataflex Crystal Reports *class*.

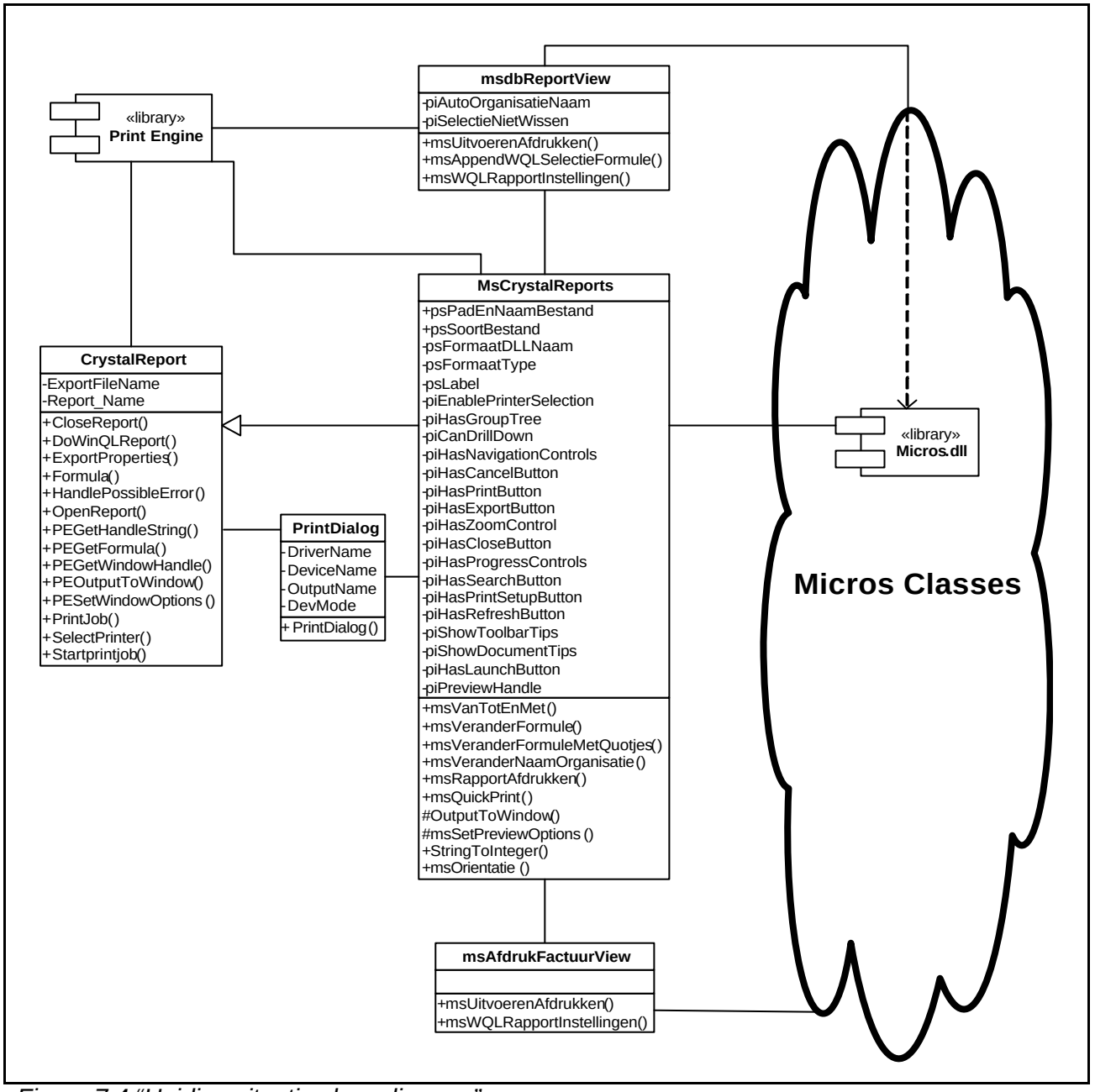
Omdat er gewerkt wordt over verschillende bestanden, is het soms niet altijd duidelijk wat het doel van elk bestand is. Om tijdens het ontwerp een duidelijk beeld te krijgen van de structuur van bestanden in relatie tot de *class*, het object en de bewerkingen, heb ik figuur 7.3 toegevoegd. Daar is te zien hoe gecommuniceerd wordt, van reportview naar Crystal Reports *class*. Er is naar mijn mening geen techniek die hetzelfde weer geeft. Een bewezen, door MBIT gedoceerde, techniek hanteer ik echter niet.



Figuur 7.3 "Structuur"

De impact die het Crystal Reports project zal hebben op Micros B.V. is, naar mijn mening, minimaal. Bij veranderen van versie zal alles in theorie nog gewoon moeten functioneren, wat echter onopgemerkt zal blijven. De medewerkers van Micros B.V. zullen nog steeds dezelfde aanroepen en dezelfde *classes* gebruiken.

In figuur 7.4 is te zien wat het *class* diagram van de definitiestudie voor de huidige situatie is. Het *class* diagram bevat een wolk, wat ongebruikelijk is binnen een *class* diagram en [UML]. De wolk, als het onbekende, is geen ongebruikelijke methode in de wereld van de ICT. Binnen het kader van dit *class* diagram stelt de wolk, alle andere Micros *classes* voor. Het aangeven van de *libraries* was noodzakelijk om het beeld te geven dat er niet alleen *classes* een rolspelen, maar ook globale functies.



Figuur 7.4 "Huidige situatie class diagram"

7.1.3 Systeemeisen

"Bereid pilotplan-workshop voor", de volgende stap volgens [IAD], komt door het ontbreken van een workshop niet aan de orde. De reden, dat er geen workshops gehouden zijn zoals hiervoor al geschreven in paragraaf 5.2, omdat de eisen (hieronder duidelijk gemaakt) die aan het systeem zijn gesteld, al reeds bekend zijn. "Evalueer pilot", het volgende hoofdstuk volgens [IAD] komt ook te vervallen, omdat nog geen *pilot* voor dit project heeft plaatsgevonden.

De volgende fase, in [IAD], is "Definieer systeemeisen". In de beschrijving van "Definieer systeemeisen" staat dat het een weergave van de behoeften van de organisatie moet zijn. Ik heb hiervoor, samen met mijn bedrijfsmentor, de opties doorgenomen, omtrent eisen die Micros B.V. stelt aan de Micros *class,* die gebruikt wordt voor Crystal Reports. Aan de hand van het werkdocument over de werking van de Micros Crystal Reports *class* en de verschillende prototypes (paragraaf 5.4.5 tot en met 5.4.8) heb ik verbeteringsalternatieven aangeboden.

Micros B.V. wil de huidige structuur in standhouden, maar wel meer inzicht in de werking van de huidige structuur krijgen. Mijn bedrijfsmentor wenst daarbij de mogelijkheid tot het kiezen van de

sorteerrichting van het Crystal Reports rapport. Dit omdat er veel vraag naar is, vanuit de klantenkring van Micros B.V..

Aangezien Micros B.V. de huidige structuur in stand wil houden, heb ik alle kenmerken en bewerkingen, die in de huidige Crystal Reports *class* gedefinieerd staan, besproken. Bij iedere kenmerk heb ik het type vastgesteld, hoe het kenmerk in de nieuwe *class* aanwezig moet zijn. Bij de bewerkingen heb ik de invoer- en uitvoerparameters besproken (zoals in figuur 7.5 te zien is).

Functie StringToInteger

Invoerparameters:

Naam	Type	Beschrijving
sVal	String	Een willekeurige string

Uitvoerparameters:

Type	Beschrijving
Integer	Optelling van de ASCII waardes van de string.

Figuur 7.5 "Eisen voor bewerking StringToInteger"

7.1.4 Systeemconcept

[IAD] beschrijft, onder andere in "Bepaal systeemconcept", dat er gemodelleerd moet worden, uit het gezichtspunt van de gebruiker. Tevens dat het systeemconcept een beschrijvende weergave is op globaal niveau van de oplossing.

Voor de globale oplossing heb ik een model van de *classes* opgenomen, die als middelpunt de nieuwe Crystal Reports *class* heeft. Een totaalbeeld geven van alle *classes* bij Micros B.V. kost teveel tijd. Het plaatje zou tevens heel erg druk zijn wat weer niet ten goede komt van de verduidelijking van Crystal Reports *class*.

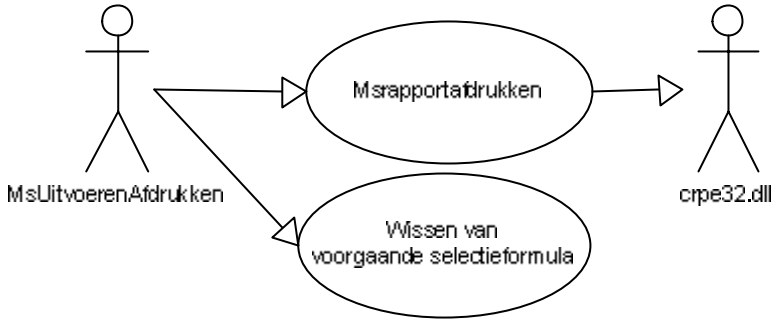
De kenmerken (eigenschappen of proporties), die al bekend gemaakt zijn bij de systeemeisen, zijn aangevuld met commentaar. Verder heb ik vier nieuwe kenmerken, die nodig zijn voor het correct functioneren van de nieuwe *class*, toegevoegd. Kennis, opgedaan bij het werken met prototypes, leverde de nieuwe kenmerken.

De oude bewerkingen zijn wederom opgenomen om de *class* te laten functioneren als de oude *class*. De nieuwe bewerkingen zijn bedoeld om meer functionaliteit aan te bieden aan de programmeur. Verder heb ik mijn ideeën, bij de nieuwe bewerkingen beschreven en bij de oude bewerkingen de ideeën die de programmeur mogelijk destijds had. De ideeën van de bewerkingen van de *class* heb ik gedeeltelijk uit het vooronderzoek gehaald (door middel van de prototypes paragraaf 5.4.5 tot en met 5.4.8) die ik gemaakt heb om de werking van de oude *class* te achterhalen. Tevens heb ik de bewerkingen geclusterd op functionaliteit, om later iets gemakkelijker te kunnen terugvinden.

Terugkomend op het moduleren vanuit het gezichtpunt van de gebruikers. Aangezien de *class* niet direct zal worden aangesproken door een gebruiker van de applicatie, maar door de applicatie zelf (wat al een bestaand systeem is), heb ik ervoor gekozen om, in plaats vanuit het gezichtspunt van de gebruiker, te kiezen voor bewerkingen die communiceren met de nieuwe *class*.

Daarbij maak ik gebruik van *use-cases*. Omdat we kijken vanuit bewerkingen, die communiceren met de nieuwe *class* , kies ik voor een *class* en bewerkingen als *actor* en niet een persoon.

Het hoofddoel van de class is het afdrukken van Crystal Reports rapport door gebruik te maken van Crystal Reports Print Engine (crpe32.dll). De Procedure MsUitvoerenAfdrukken is de huidige plek waar die gebeurt. Dit zal niet veranderen ten opzichte van de huidige situatie maar. Voor het uitvoeren van Msrapportafdrukken en het lezen van nieuwe selectie formule moet wel de oude selectie formule gewist worden.



Figuur 7.6 "Use-case"

Doordat ik aangeef waar en waarom een dergelijke aanroep plaatsvindt binnen de applicatie, is het makkelijker te achterhalen waar, binnen de huidige applicatie, aanpassingen of toevoegingen gemaakt moeten worden.

7.1.5 Organisatorische inrichting

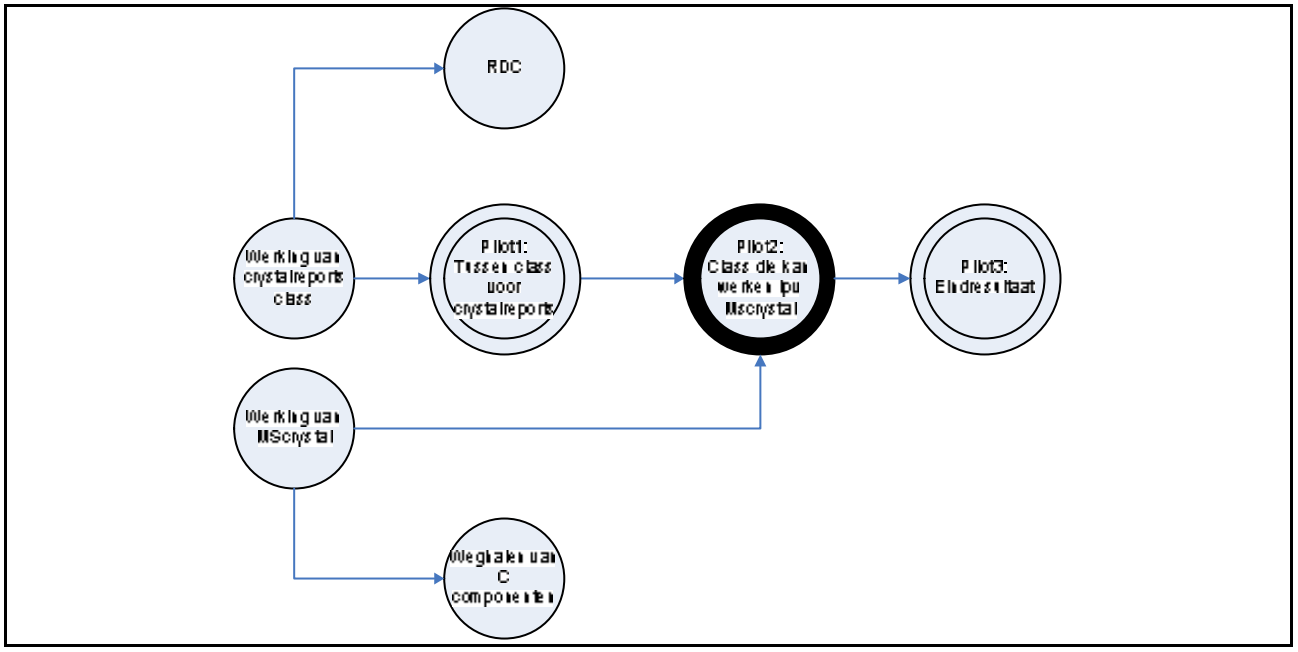
Het volgende stap in [IAD] zou zijn "Technische structuur", maar deze inhoud is verspreid over de volgende hoofdstukken van de definitiestudie, omdat zo een betere lijn in de definitiestudie komt.

Het doel van dit hoofdstuk is uitzoeken wat de impact is op het bedrijf,in mijn geval Micros B.V. Aangezien ik aan de *backend* werk is er geen echte impact van deze ontwikkelingen bij Micros B.V.. Zoals als ik al had geschreven in voorgaande paragrafen. Doordat het geen significante impact zal hebben bij Micros B.V. heb ik de impact op de bestaande bestanden van Micros *classes* beschreven. De Micros *classes* bestaan uit veel verschillende bestanden. Twee bestanden moeten tevens worden aangepast wil de nieuwe, verbeterde *class* werken in de huidige situatie. Ik geef hier een globale beschrijving van welke bestanden aangepast worden en waarom. Dit alles dus aansluitend bij het systeemconcept.

Tevens is een figuur opgenomen, die lijkt op figuur 5.14 van dit document, om de toekomstige situatie weer te geven. Zo ook een beschrijving waarom de toekomstige situatie niet exact lijkt op de ideale situatie.

7.1.6 Pilotplan

In het hoofdstuk *pilotplan* beschrijf ik de prototypes en de te maken *pilots*. De vier prototypes die gemaakt zijn staan al beschreven in dit document. (paragraaf 5.4.5 tot en met 5.4.8). Verder heb ik de conclusies die ik getrokken hebt uit de prototypes nader toegelicht.



Figuur 7.7 "Pilot plan"

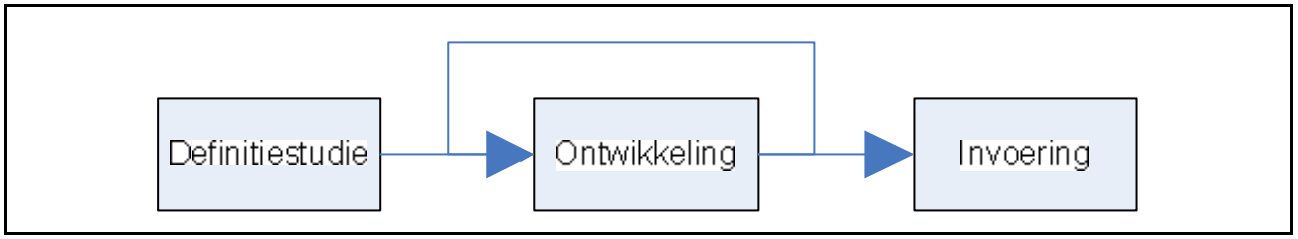
De planning was om drie opéénvolgende *pilots* te maken (zoals hierboven te zien is) Een *pilot* is volgens [IAD] een coherente subset van het uiteindelijke resultaat die als zelfstandige, bruikbare eenheid kan worden ingevoerd bij Micros B.V..

De eerste *pilot* heeft als doel de kennis van de Crystal Reports *class* te benutten en een bruikbare communicatie *subclass* te maken van de Crystal Reports *class* van Visual Dataflex. Die kennis is deels verkregen vanuit het prototype (paragraaf 5.4.5) voor het aansturen van Crystal Reports. Een prototype is een experiment en mag niet direct gebuikt worden voor de *pilot*. Toch heb ik de kennis van het prototype gebuikt bij het ontwerp van de nieuwe *class*.

De tweede *pilot* is bedoeld om de werking van de oude Micros Crystal Reports *class* toe te voegen aan de eerste *pilot*. De Micro Crystal Reports *class* is ook een *subclass* van de Crystal Reports *class* van Visual Dataflex. Dit zal het samenvoegen ten goede komen. Ook wilde ik ervoor zorgen dat met deze *pilot*, het gebruik van de Micros C *library,* bij de communicatie met Crystal Reports, tot het verleden behoort. Hiervoor gebruik ik de selectieformule-bewerkingen van de eerste *pilot*.

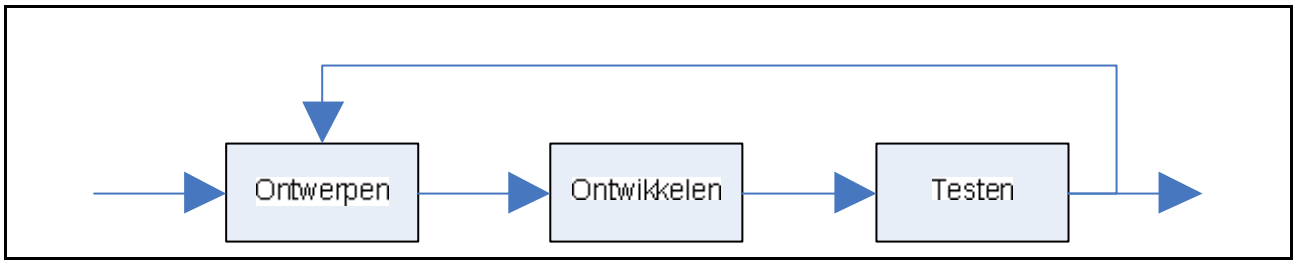
Als laatste *pilot* wil ik alle communicatie met Crystal Reports *print engine*, die niet plaatvindt via de Crystal Reports *class*, in het nieuwe bestand clusteren. Tevens wil ik met deze *class* de mogelijkheid geven tot het sorteren van de informatie in een Crystal Reports rapport binnen de Micros applicaties.

In dit hoofdstuk beschrijf ik tevens dat de *pilots* incrementeel, via de methode van [IAD], ontwikkeld zijn. De reden hiervoor is dat al aan het begin het doel (definitie) bekend is. Dit betekent dan er geen iteratieslagen noodzakelijk zijn voor de definitiestudie. De impact op het bedrijf is gering, vroegtijdige invoering is dus geen noodzaak, maar toepasbaarheid geldt voor latere versies van de Micros *classes.* Verder zal het *pilot* ontwikkelingstraject meerdere malen worden uitgevoerd om tot de laatste *pilot* te komen.



Figuur 7.8 "incrementeel ontwikkelen"

Het ontwikkelproces wordt per *pilot* totaal herhaald. Het ontwikkelproces zelf, kan ook gedeeltelijk herhaald worden, bijvoorbeeld als een ontwerpfout blijkt, dan wordt het gerectificeerd. Meestal komen deze fouten aan het licht tijdens het ontwikkelen en testen.



Figuur 7.9 "ontwikkelproces"

7.1.7 Pilot ontwikkelplan

Als afsluiter van de definitiestudie heb ik het *pilot* ontwikkelplan opgenomen. Het *pilot* ontwikkelplan vertelt mij wat er ontwikkeld moet worden. De bewerkingen verdeeld over de *pilots* en het doel van de *pilot* herhaald.

7.2 Pilot ontwikkeling

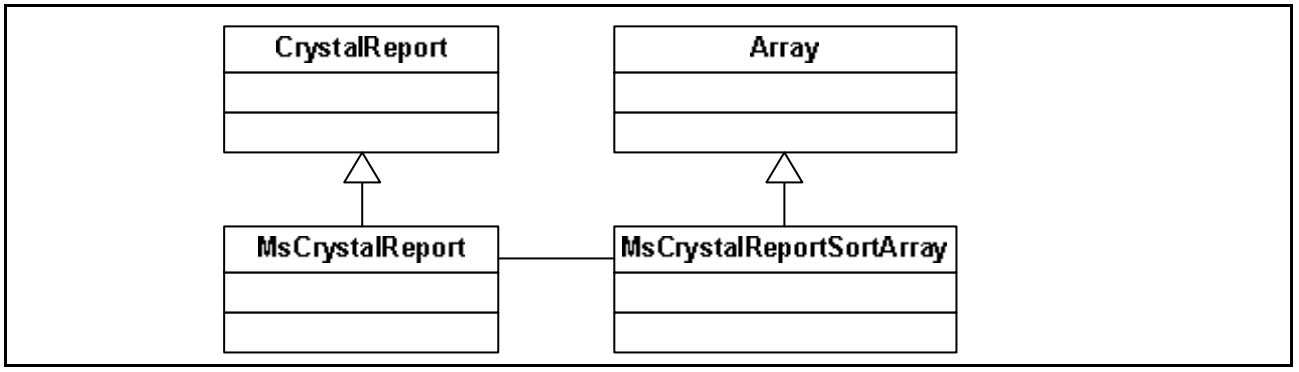
Het project is verdeeld in drie *pilots*. In de volgende paragrafen geef ik uitleg over het hoe en waarom de *pilots* te ontwikkelen. Het doel van deze fase is om het systeemconcept, zoals beschreven is in de definitiestudie, volledig uit te werken. De laatste *pilot* zal de realisatie zijn van het Crystal Reports project. Omdat er geen workshops voor de *pilots* gepland stonden in het plan van aanpak, wat ik voor iedere *pilot* heb gemaakt, ben ik direct naar het ontwerp van bouweenheden gegaan.

7.2.1 Pilot 1

De eerste *pilot* is bedoeld voor het gebruiken van de standaard *class* van Visual Dataflex voor de communicatie met Crystal Reports.

Class diagram

Om een globaal beeld te creëren van de *pilot* en de omgeving waarin hij zich bevindt, heb ik ervoor gekozen een gedetailleerde *class* diagram te maken. Het diagram krijgt steeds meer bewerkingen en kenmerken naar mate het ontwerp en de bouw van de nieuwe *class* vordert.

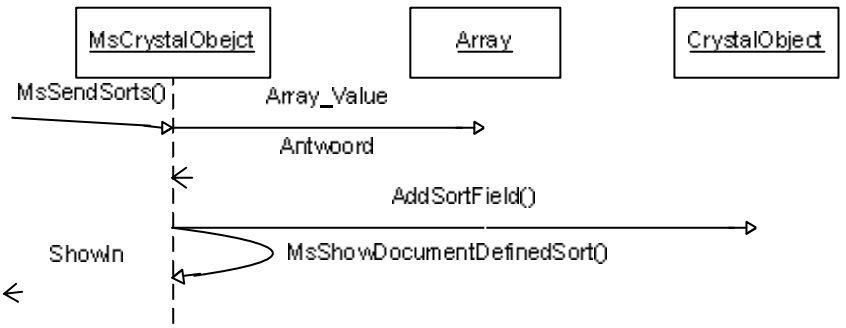


Figuur 7.10 "Class diagram pilot 1"

Er zit een technisch verschil in het *class* diagram van het systeemconcept in de definitiestudie en het *class* diagram in het ontwerp van de eerste *pilot*. (zie figuur 7.10 als representatie van het *class* diagram in het ontwerp van de eerste *pilot*) Het verschil ligt in het feit dat bij het systeemconcept "`MsCrystalReportSortArray`" een kenmerk is en in het ontwerp document "`MsCrystalReportSortArray`" een *class* is. Hiervoor, in paragraaf 5.1, heb ik uitgelegd dat het datatype *array* niet aanwezig is. De *class* is slechts een methode van opslag van data, dat is de reden dat ik de *array* niet direct als *class* heb getekend. De *class* is niet bedoeld voor gebruik door andere *classes*, al is dat wel mogelijk.

Nieuwe Classes

Het ontwerp van de bewerkingen van de *subclass* "`MsCrystalReport`", die communiceert met de Crystal Reports *class* van Visual Dataflex, zijn gegroepeerd zoals in de definitiestudie beschreven staat. Alle bewerkingen bevatten een beschrijving, *sequence diagram* en de invoervariabelen van de bewerking, indien die aanwezig zijn.



bewerkingen, overwoog ik om flowcharts en/of PSD's te gebruiken. Ik heb ervoor gekozen geen flowchart en/of PSD te benutten, omdat de meeste bewerkingen daar niet complex genoeg voor zijn.

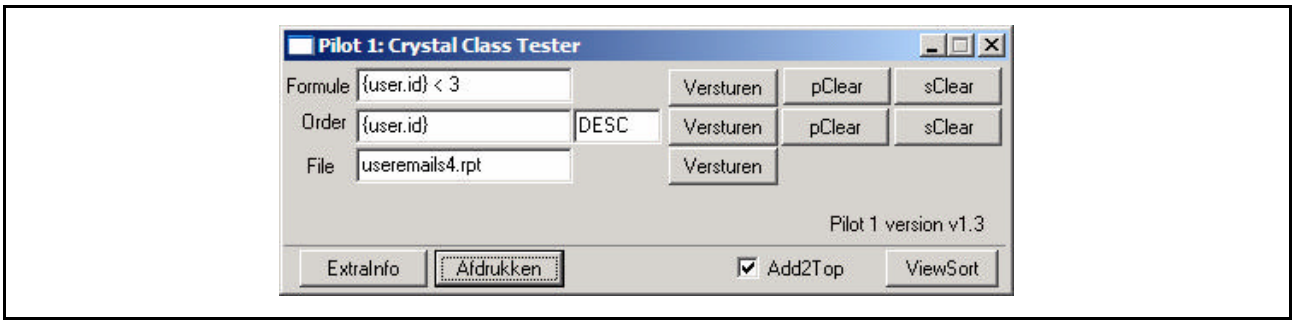
Bestaande classes
Zoals in het klasse model (figuur 7.10) te zien is, zijn er meerdere *classes*. Vandaar een ander hoofdstuk met bestaande *classes*. Deze *classes* behoeven niet aangepast te worden.

Omdat ik reeds bij het ontwerp aangeven heb welke externe bewerkingen de nieuwe bewerking aanroept, was het niet zo moeilijk te achterhalen welke externe bewerkingen gebruikt worden. De reden van deze inbegrepen informatie is een goed beeld te hebben van wat er gebruikt wordt van de bestaande *classes*. Als er in de mogelijke toekomst aanpassingen komen aan de Visual Dataflex *class*, die met Crystal Reports communiceren, dan is dat een potentieel probleem. Omdat in het ontwerp staat welke externe bewerkingen er gebruikt worden, is het effect te achterhalen op de nieuwe *class*. Door het nalopen van het ontwerp is ook te achterhalen, welke bewerkingen aangetast zijn door het potentiële probleem. Door het testen op die specifieke bewerkingen, kan gekeken worden of die ene bewerking nog naar behoren functioneert.

Ontwikkelen
Het ontwikkelen van de nieuwe *class* ging zonder veel problemen. De nieuwe *class* die ik gemaakt heb werkt in de IDE (*Integrated Development Environment*). De *class* maken in een geschikte werkomgeving is een groot pluspunt. Mijn handelwijze is beginnen met het uitvoeren van een Crystal Reports testrapport en vervolgens selectieformules toe te passen op het testrapport.

Voor sorteren heb ik een iets andere volgorde gehanteerd. Ik ben begonnen met het maken van het sorteer *array*. Daarna heb ik de sorteerbewerkingen geschreven voor de nieuwe Crystal Reports *class*. Wat mij bevreemd is dat ik de definitie voor de `"Zerotype"` gedaan had in de *class* voor het sorteer *array* en dat de `"Zerotype"` ook werkt in de *class* (de nieuwe Crystal Reports *class*) die het sorteer *array* aanroept. Dat het niet werkt zou logischer zijn. Dit betekent waarschijnlijk dat een `"Zerotype"` een globale waarde is.

Testen
Het testen van de *pilot* heb ik gedaan met de "*Pilot* 1 tester", te zien in figuur 7.12.



Figuur 7.12 "Pilot 1 tester"

Voor het testen heb ik gebruik gemaakt van testsets zoals te zien is in figuur 7.13. Ik maak onderscheid tussen verwachte en waargenomen verschillen. Als bij testen van latere versies, dezelfde testsets gebruikt worden, dan is te achterhalen of de *class* nog *backwards compatible* is.

Logisch	Verwacht	Waargenomen
Afdrukken	Crystal met resultaten kleiner dan 3	Crystal met resultaten kleiner dan 3
Alf+f4	Crystal sluit af	Crystal sluit af
Formule = {user.id} > 1	Outputwindow die verteld dat er een toevoeging is aan de selectieformules	Outputwindow die verteld dat er een toevoeging is aan de selectieformules
Afdrukken	Crystal met resultaten waar userid 2 is.	Crystal met resultaten waar userid 2 is.
Alf+f4	Crystal sluit af	Crystal sluit af
pClear	Melding dat selectieformules van het document niet worden opgenomen in resultaat	Melding dat selectieformules van het document niet worden opgenomen in resultaat
Afdrukken	Crystal met resultaten groter dan 1	Crystal met resultaten groter dan 1
Alf+f4	Crystal sluit af	Crystal sluit af
pClear	Melding dat selectieformules van gewist zijn.	Melding dat selectieformules van gewist zijn.
Afdrukken	Crystal start met alle resultaten	Crystal start met alle resultaten

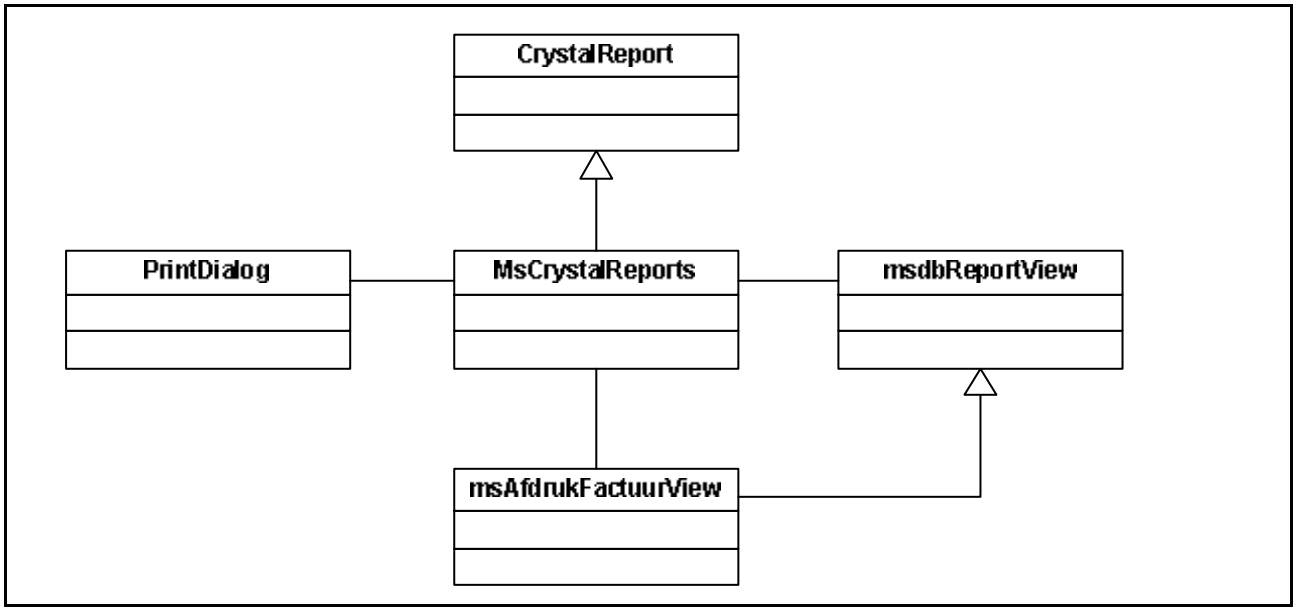
Figuur 7.13 "Testset"

7.2.2 Pilot 2

De tweede *pilot* is bedoeld om de functionaliteit van de eerste *pilot* (alle nieuwe bewerkingen en kenmerken) en de huidige Micros Crystal Reports *class* samen te voegen tot één *subclass* van de standaard Visual Dataflex *Crystal Reports class*.

Class diagram

Om wederom een globaal beeld te creëren van de *pilot* en de omgeving waarin hij zich bevindt, heb ik, net als in het ontwerp van de vorige *pilot*, gekozen een *class* diagram te maken die in grote lijnen lijkt op figuur 7.14.



Figuur 7.14 "Class diagram pilot 2"

In het bovenstaande *class* diagram en het *class* diagram in de ontwerpdocumentatie van deze *pilot* is geen "`MsCrystalReportSortArray`" in het diagram inbegrepen. De reden hiervoor is, dat op dit moment de opslag van sorteergegevens geen interessant gegeven is. Vooral niet voor de integratie van de huidige *class* voor communicatie met Crystal Reports die niet eens gebruik maakt van de sorteerfuncties.

De workshop die volgens [IAD] uitgevoerd had moeten worden, is niet uitgevoerd. De huidige situatie volgens de nieuwe methode laten lopen had de voorkeur. Een workshop met verschillende programmeurs heeft mijn inziens geen zin. De huidige programmeurs maken geen direct gebruik van deze bewerkingen, maar werken meestal twee á drie lagen hierboven.

Toevoegen en verbeteren oude bewerkingen en kenmerken
In het ontwerpdocument van de tweede *pilot* is een hoofdstuk toegewijd aan de bewerkingen en kenmerken van de huidige situatie, die geïntegreerd worden in de nieuwe situatie. Bij de bewerkingen staat ook beschreven, welke aanpassingen er gemaakt moeten worden, om van de oude bewerking een bewerking te maken die past in de nieuwe situatie. Het formuleren van de bewerkingen heb ik op identieke wijze gedaan als voor de eerste *pilot*.

Wijzigingen externe componenten
Zoals te lezen is in paragraaf 5.4.8 is onderzoek geweest (door middel van een prototype) naar het laten vervallen van Micros C *library*. Voor de realisatie moeten er in totaal twee veranderingen plaats vinden in één externe *class*.

Eén van deze veranderingen is wederom bedoeld voor de selectieformules. De selectieformules worden door één specifieke bewerking in het geheugen gezet. De bewerking is gevonden tijdens het onderzoek. Andere momenten waren ook mogelijk, maar in verband met de verschillende soorten maatwerk, is een bewerking die diep in de structuur aanwezig is, de beste optie voor het toevoegen van de selectieformules.

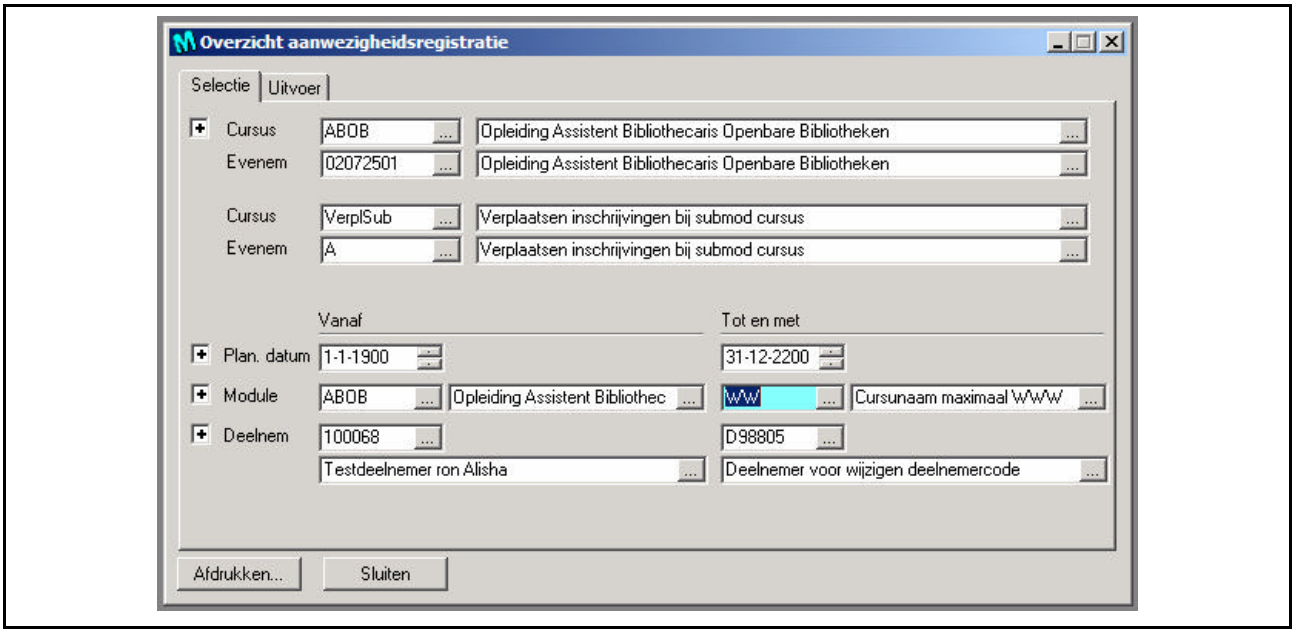
De laatste aanpassing, die ik heb verricht aan de bestaande *class* is, ervoor te zorgen dat ieder rapport een uniek object heeft om mee te werken. In de huidige situatie maakt het niet uit of het object uniek is, omdat het geheugen iedere keer wordt overschreven, als er een nieuw rapport wordt aangeroepen. In de nieuwe situatie worden de oude selectieformules niet overal automatisch overschreven.

De plek waar het nieuwe standaard Micros Crystal Reports object wordt aangemaakt is in de *class*, die de basis is voor ieder reportview `"MsdbReportView"`. Het bestand `"msrep_mx.pkg"` is de plek waar de wijzigingen gemaakt moeten worden. Het is wel toeval dat voor de tweede *pilot* alle aanpassingen aan externe componenten, in dezelfde *class* en hetzelfde bestand plaatsvindt.

Ontwikkelen / testen
Het ontwikkelen van de tweede *pilot* verliep stroever dan de ontwikkeling van de eerste *pilot*. De nieuw ontwikkelde *class*, werkt niet in de IDE (*Integrated Development Environment*). Evenmin als in de huidige situatie. Dit heeft niets te maken met de gemaakte toevoegingen aan de *class*. Dit was een groot minpunt want ik kon de *class* niet geschikt maken voor de werkomgeving van de IDE.

Om niet nog meer tijd te verliezen, heb ik gekozen voor het verder ontwikkelen van een eigen versie van bestaande Micros applicaties. De eigen versies van KAS en RAS, die ik gebruikt hebt zijn exacte kopieën van de standaard applicaties zonder maatwerk. Het voordeel dat ik een totaal eigen versie heb is, dat als ik veranderingen/fouten maak, die alleen voor mij zichtbaar zijn en er kan getest worden in een werkelijke situatie.

Met behulp van de *editor* Multiedit heb ik de oude bewerkingen en kenmerken van de Micros Crystal Reports *class* aangevuld aan de Crystal Reports *class*, die gemaakt is tijdens de eerste *pilot*.



Figuur 7.15 "Pilot 2 tester"

Zoals hiervoor al beschreven staat heb ik de ontwikkeling van de tweede *pilot* gedaan in bestaande Micros applicaties. Met reportviews zoals hierboven heb ik testsets gemaakt waarmee ik getest heb of de oude bewerkingen nog steeds correct functioneren. Bij het testen is vooral op het feit gelet of de selectie voor de selectieformules correct in het Crystal Reports worden uitgevoerd.

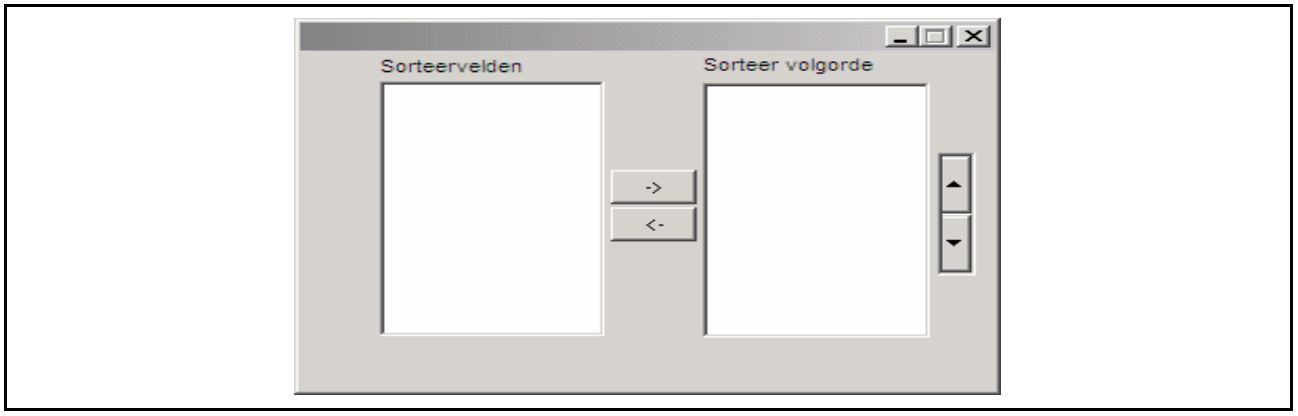
7.2.3 Pilot 3

De derde *pilot* is bedoeld als afsluiter van het Crystal Reports project. De *pilot* heeft twee taken:

- Het clusteren van het aanroepen van de Crystal Reports *print engine* door de Micros *classes* in Micros *class*;
- Een methode om een sorteerkeuze te maken in Micros Applicaties voor Crystal Reports.

Voor het uitvoeren van de derde *pilot* heeft een soort workshop plaatsgevonden tussen mijn bedrijfsmentor en mij. De workshop handelde over hoe we de gebruiker konden laten kiezen voor de sorteervolgorde en hoe de programmeur kan kiezen welke velden er gekozen mogen worden door de gebruiker.

Mijn bedrijfsmentor wilde graag dat men in het Crystal Reports rapport kan klikken om een selectie te maken waarop gesorteerd kan worden. Het was technisch onmogelijk dit te realiseren met de huidige Crystal Reports *print engine*. Ik kwam zelf met twee voorstellen.
Het eerste voorstel:
Een *dropdown* menu, waarbij op het geselecteerde veld gesorteerd wordt. Als er een nieuwe selectie is in een *dropdown* menu, dan wordt het geselecteerde veld het eerste veld waarop gesorteerd wordt en het oude veld, het tweede veld waarop gesorteerd wordt.
Het tweede voorstel:
Een optiescherm, met twee lijsten. (zoals figuur 7.16) Sorteervelden die gekozen kunnen worden en een lijst met velden die gekozen zijn voor de sorteerrichting.



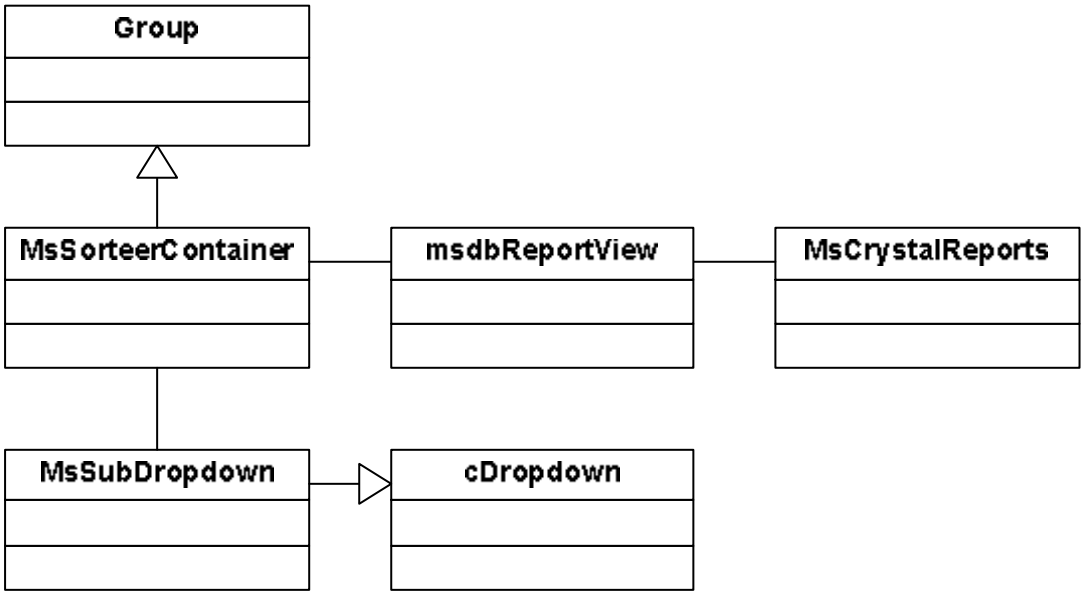
Figuur 7.16 "Sorteer optie"

Gekozen wordt de eerste optie. De belangrijkste reden daarvoor is dat deze optie het gemakkelijkst te begrijpen is voor alle gebruikers.

Om overal een unieke sorteervolgorde optioneel te kunnen aanbieden, heb ik voorgesteld om een *class* te maken die al de communicatie regelt tussen een *dropdown* menu en de nieuwe Micros Crystal Reports *class*. In een object van deze nieuwe *class*, die aangeroepen kan worden in alle reportviews gebaseerd op `"MsdbReportview"`, kan de sorteervolgorde die gewenst is met een paar regels code gecreëerd worden.

Class diagram

Om wederom een globaal beeld te creëren van de derde *pilot* en de omgeving waarin deze zich bevindt, heb ik net als in het ontwerp van de vorige *pilot*, voor het maken van een *class* diagram gekozen (zie figuur 7.17).



Nieuwe Classes

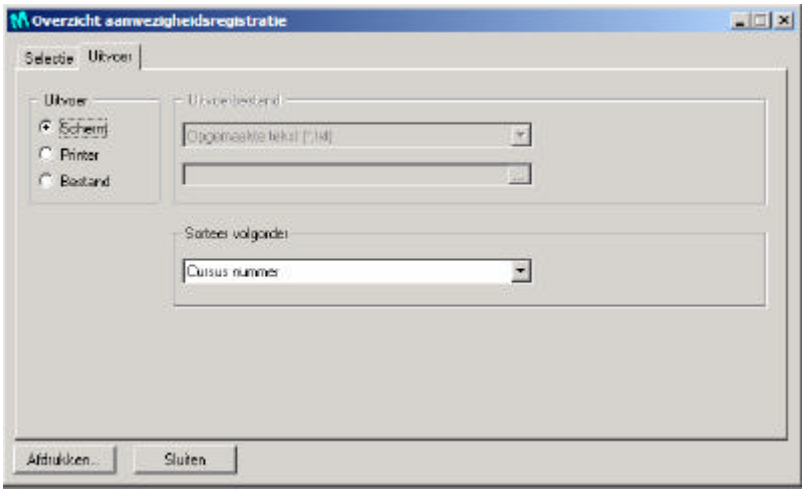
De twee nieuw gecreëerde *classes*, heb ik in één hoofdstuk, dat van het ontwerp van de derde *pilot*, behandeld. Omdat je in een object, dat gedefinieerd is binnen een *class*, geen gebruik mag maken van de `"onchange"` bewerking, heb ik een extra *class* toegevoegd, zodanig dat dit wel kan. De `"MsSorteerContainer"` maakt gebruik van het doorgeven van de extra *class* op de selectiekeuze aan de nieuwe Crystal Reports *class*. Door het toevoegen van een paar regels code zoals in figuur 7.18 te zien is, kan het *dropdown* menu gevuld worden.

```
Object MsCrystalOrder is a MsSorteerContainer
  send MsToevoegenWaarde "{CURSUS.CURSUS#}" "Cursus nummer"
  send MsToevoegenWaarde "{CURSUS.NAAM}" "Cursus naam"
  send MsWissenVanSorteerInDocument
End_Object
```

Figuur 7.18 "Aanroep MsSorteerContainer"

Ontwikkelen / testen

Het ontwikkelen van de derde *pilot* ging haast vlekkeloos. Dat de `"onchange"` bewerking in een eigen *class* moet ontdekte ik tijdens de ontwikkeling. De derde *pilot* is ook getest in bestaande versies van de Micros Applicaties. Hieronder is het resultaat te zien van de code van figuur 7.18 in een reportview van KAS.



Figuur 7.19 "Pilot 3 resultaat"

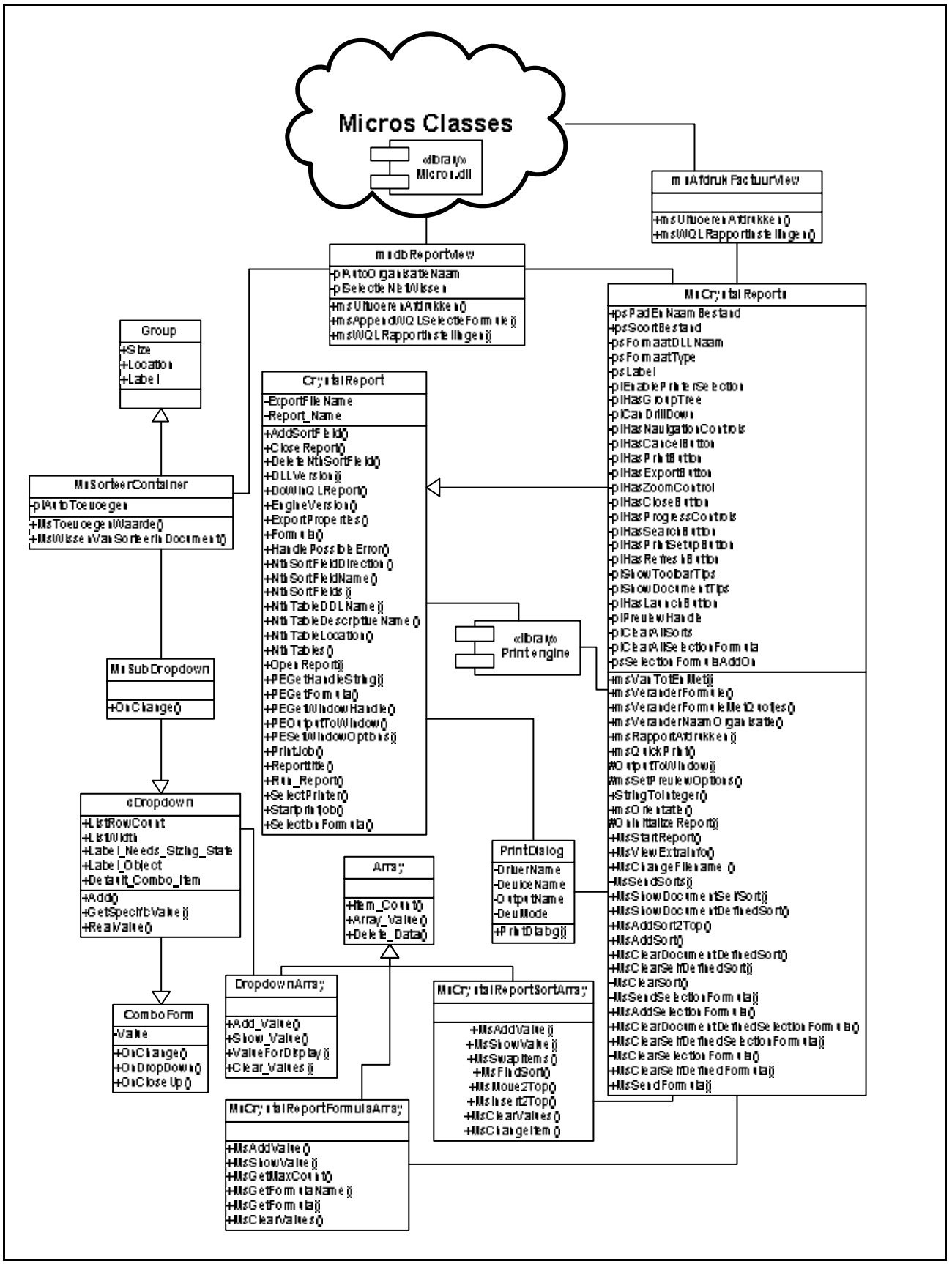
Doordat bij een selectieoptie, een extra object aangemaakt moet worden, betekent dat als het extra object niet aangemaakt wordt, de huidige structuur geen invloed ondervindt. Dit heb ik getest op meerdere reportviews door gebruik te maken van de code die lijkt op figuur 7.18 en het weglaten van de code. Dit was één van de belangrijkste eisen van het project: 'alles blijft werken in de oude structuur!'.

7.3 Invoering

Het invoeren van het resultaat van het hele *pilot* traject verliep met behulp van verschillende documenten, "Totaal ontwerp" en "Gebruikers handleidingen".

7.3.1 Totaal ontwerp

Om het ontwerp niet verspreid te hebben over meerdere documenten heb ik alle *pilot* ontwerpen tot één groot ontwerp samengevoegd. Het resulteerde in één groot klasse model zoals hieronder.



Figuur 7.20 "Toekomstige class diagram"

De verbinding tussen de "MsCrystalReports" *class* en de *print engine* is wat de ideale situatie scheidt van de toekomstige situatie.

Als één van de eerste hoofdstukken in het totale ontwerpdocument, heb ik het hoofdstuk 'Voorwaarde voor onderhoud' opgenomen. Dit hoofdstuk vertelt hoe je het best kan omgaan met het maken van aanpassingen aan de *class*. Dit hoofdstuk is toegevoegd om meer richtlijnen te geven aan hoe er bij Micros B.V. aanpassingen aan bestaande *classes* gemaakt worden.

Alle ontwerpen van de bewerkingen en de werking van de kenmerken van de nieuwe Micros Crystal Reports *class* heb ik in het totale ontwerpdocument opgenomen onder het hoofdstuk "MsCrystalReports". De bewerkingen zijn onderverdeeld in verschillende groepen, die al aan het licht kwamen tijdens het maken van de eerste en tweede *pilot*. Vooraf gaand aan de bewerkingen is een overzicht van alle bewerkingen, die aangeeft of een bewerking bedoeld is voor interne aanroep of externe aanroep. Wat weer nuttige informatie is voor het onderhoud.

In het hoofdstuk 'Extra *classes*' staan de nieuwe *classes* en zijn bewerkingen en kenmerken beschreven. Onder deze *classes* valt, "MsCrystalReportSortArray", "MsSubDropdown" en "MsSorteerContainer".

In het hoofdstuk 'Externe *classes*' staan de *classes* waarvan de bovenstaande *classes* gebruik maken. Onder deze *classes* vallen, "MsdbReportView", "PrintDialog", "CrystalReports", "Array", "msAfdrukFactuurView" en "cDropDown".

Het hoofdstuk 'Invoering' is geschreven voor het invoeren van de nieuwe functionaliteit in het huidige systeem. Er is maar één bestand dat aangepast moet worden om de nieuwe situatie werkend te krijgen en dat is "msrep_mx.pkg". De aanpassingen staan al bij de het ontwerp van de tweede *pilot* beschreven, maar ter verduidelijking staan ze hier nogmaals, want we werken vanuit één document.

Dit document dient ter verduidelijking, dat doet het zeker voor mij, want ik gebruik dit document meer dan de losse ontwerpdocumenten. 'Invoering' is duidelijk en gestructureerd.

7.3.2 Gebruikershandleidingen

Voor de duidelijkheid, omtrent de werkingen, is voor programmeurs een gebruikershandleiding geschreven. Voor de gebruikers handleiding heb ik dezelfde methode gebruikt als beschreven in paragraaf 6.4. Bij de 'definitie', die er ongeveer uitziet als Figuur 7.18 voor "MsSorteerContainer", staat hoe je de *class* aanroept en wat die doet. Bij 'gebruik' staat alles wat de *class* doet, wat de programmeur moet weten voor extern gebruik van de *class* "MsSorteerContainer".

8. Versiebeheer

Versiebeheer zorgt ervoor dat aanpassen en verbeteren van de *class* gestructureerd plaatsvindt. Als versiebeheer correct wordt toegepast is het mogelijk dat oude versies, van bijvoorbeeld deze *class*, vervangen kunnen worden door een nieuwere versie. Na de vervanging moet de programmatuur, waarin de *class* zich bevindt, nog steeds correct werken.

Om versiebeheer voor de Micros *classes* bij Micros B.V. toe te passen heb ik een document met richtlijnen geschreven, die vooral 'Versiebeheer' behandelt. Als basis heb ik de theorie van versiebeheer, die ons gedoceerd is tijdens SW-07, gebruikt om versiebeheer voor de Micros *classes* te creëren bij Micros B.V.. De theorie vertelt over het gebruik van projectleider, configuratiebeheerder, documentbeheerder, applicatie- en documentatietester en programmeurs. Bij Micros B.V. zijn in de huidige situatie, alleen maar projectleiders en programmeurs aanwezig. De projectleiders zijn tevens applicatietesters.

Micros B.V. wil een zo laag mogelijke kostenpost voor het versiebeheer. De programmeurs en projectleiders willen een zo min mogelijke verandering in hun werkwijze. Dit resulteerde in mijn voorstel: programmeurs zijn nog steeds verantwoordelijk voor het aanpassen en testen van de Micros *classes*, maar aanpassingen worden niet direct door hen uitgevoerd de aanpassingen moeten eerst gedocumenteerd en goedgekeurd worden door de projectleider.

Het filesysteem geeft de mogelijkheid dat de programmeurs gebruik kunnen maken van de standaard Micros *classes* bij het compileren van de applicaties. Gebruik van duidelijke documentatie en broncode geeft ook inzicht in de werking van de Micros *class*. Als de huidige Micros *classes* en bijbehorende documentatie op *read only* staat zal het niet mogelijk zijn, dat de programmeur aanpassingen maakt aan de Micros *class*. Als tevens gezorgd wordt dat oude versies, voor de programmeurs, niet beschikbaar zijn, dan zal altijd gewerkt worden met de meest *up-to-date* Micros *classes*.

Voor het maken van aanpassingen neemt de programmeur contact op met zijn projectleider. De programmeur doet een wijzigingsvoorstel om de *class* aan te passen. De projectleiders zijn echter verantwoordelijk voor de Micros *classes* en bijbehorende documentatie. Die neemt in overweging of de voorgestelde wijziging verstandig is of niet. Aangezien Micros B.V. veel maatwerk levert, moet rekening gehouden worden met het feit dat de veranderingen geen negatief effect hebben op het gemaakte maatwerk. Micros B.V. zou, in verband met dat maatwerk, het best over kunnen stappen op het gebruik van *subclasses* voor aanpassingen aan de Micros *classes*.

9. Evaluatie

De evaluatie is verdeeld over de opgeleverde producten enerzijds en het proces die tot die producten heeft geleid anderzijds.

9.1 Product

Tijdens het afstudeertraject zijn er ten behoeve van Micros B.V. diverse producten ontwikkeld. Deze producten zijn vooraf vastgesteld in de definitieve opdrachtsomschrijving.

9.1.1 Globaal plan van aanpak

Het document 'Globaal plan van aanpak' functioneert als basis voor het afstudeertraject. Het plan van aanpak staat als kern voor de probleemstelling en de doelstelling. Mijn bedrijfsmentor heeft, na een kleine wijziging op zijn aangeven, de probleemstelling goedgekeurd.

9.1.2 Onderzoek huidige situatie met betrekking tot Crystal Reports

Dit is een tweeledig product. Het eerste product is een beschrijving van mijn bevindingen in het onderzoek naar de werking van de huidige situatie. Dit document is door mij veel gebruikt tijdens het gehele Crystal Reports traject. Op basis van dit document kon ik de latere definitiestudie maken. Het tweede product, gebaseerd op het eerste product, draagt ideeën aan over verbeteringen aan de Crystal Reports *class*. Per verbetering heb ik de voor- en nadelen beschreven. De keuze voor het instandhouden van de huidige structuur en verbeteringen aan te brengen is basis voor de definitiestudie.

9.1.3 Definitiestudie voor Crystal Reports

De definitiestudie voor Crystal Reports is een product, van groot belang voor het Crystal Reports project. Behalve de planning, die daarin wordt aangeven, als ook de probleemstelling, geeft de definitiestudie mij ook een beeld van het uiteindelijk beoogde resultaat (doelstelling). Ik heb de kennis van alle onderdelen gebruikt bij het maken van de ontwerpen. De definitiestudie heb ik als zeer nuttig ervaren en heet naar mijn mening een geslaagd product.

9.1.4 Het Crystal Reports ontwerp rapport

Op basis van de definitiestudie en de kennis van de prototypes zijn drie ontwerpen ontwikkeld. De drie ontwerpen zijn gemaakt volgens de voorgeschreven [IAD] methode. Uiteindelijk heb ik een totaalontwerp gerealiseerd. Dit ontwerp zal als basis dienen voor latere veranderingen aan de *class(es).* Het ontwerp geeft een duidelijke beschrijving van alle bewerkingen en kenmerken die in de *class(es)* aanwezig zijn. Het document is bedoeld om per bewerking inzicht te krijgen in welke acties plaatsvinden of zouden moeten prototype plaatsvinden. Hier in is het ontwerp, naar mijn mening, in geslaagd.

In dit document wordt ook duidelijk gemaakt hoe Micros B.V. de nieuwe *classes* moet invoeren in een bestaande situatie. Aangezien er veel verschillende versies zijn van de Micros applicaties, zal iedere aanpassing met de nodige voorzorg gemaakt en getest moeten worden.

9.1.5 De realisatie van het Crystal Reports ontwerp

De Crystal Reports *class* is gebaseerd op het ontwerp en functioneert goed. Enigszins jammer is dat ik het systeem niet vollédig via de nieuwe en duidelijkere bewerkingen kan laten verlopen, dat zou namelijk betekenen dat veel maatwerk niet meer zal werken omdat die gebruik maken van de oude aanroepen. Ook jammer vind ik dat het aanroepen van de *print engine* aanbleef. De afhankelijkheid met Crystal Reports *class* en de *print engine* is geclusterd in één bestand.

De aanpassingen aan de Micros *classes* zullen, voor 99% zeker, geen effect hebben op het maatwerk, zeker als gewerkt wordt volgens de beschrijving van het hoofdstuk 'Invoering van het ontwerp'. Hier ben ik zeer tevreden over. Micros B.V. zal de nieuwe Crystal Reports mogelijkheden gaan toepassen in de nieuwe generaties van hun applicaties.

9.1.6 Word, Excel & MAPI verbeteringen

Naast Crystal Reports, zijn er nog andere derde partij software pakketten, waarvan de communicatie moest worden verbeterd, te weten Word, Excel en MAPI. In het hoofdstuk 'Oriëntatie' is mijn onderzoek naar gebruik van deze derde partij software pakketten te lezen. De standaard *classes* van Visual Dataflex voor aansturing van deze software pakketten bevatten echter niet voldoende functionaliteit om gebruikt te worden door Micros *classes*.

Na vooronderzoek heb ik een negatief advies gegeven over toepassing van 'verbeteringen' voor communicatie met deze drie software pakketten. Wat betekent dat Micros B.V. op de huidige manier verder moet gaan, omdat de standaard Visual Dataflex *classes* geen goed alternatief bieden. Deze conclusie is geen echt product, maar het resultaat van het onderzoek kan wel als product gezien worden. De resultaten die deze producten hebben opgeleverd zijn niet wat Micros B.V. zich had voorgesteld, maar omdat het mij en Micros B.V. inzicht geeft in de mogelijkheden van Visual Dataflex en Micros *classes* kunnen de producten echter toch gekarakteriseerd worden als 'geslaagd'. Vooral doordat beide partijen begrijpen waarom een negatief advies geldt.

9.1.7 De richtlijnen

De richtlijnen voor het gebruik van de nieuwe *classes* zijn aanwezig. De noodzaak voor versiebeheer is aanwezig binnen Micros B.V.. Het is aan Micros B.V. om de kennis en visies die ik heb overgedragen in gebruiken te nemen. Het is een degelijk product wat vooral bedoeld is voor gebruik binnen de huidige situatie bij Micros B.V. en zonder te veel bureaucratie.

9.2 Proces

In deze paragraaf zal de procesgang over het gehele afstudeertraject besproken worden. Wat vooral naar voren zal komen is de planning ten opzichte van het werkelijke verloop en wat ik, aan de hand van de evaluatie, een volgende keer aan mijn doelstellingen zou veranderen.

9.2.1 Planning

De planning is altijd één van de moeilijkste onderdelen van een project. Het is heel moeilijk in te schatten hoe lang je over een onderdeel zal doen als je niet bekend bent met dat onderdeel. Het tijdpad inschatten berust op ervaring. Het maken van een plan van aanpak geschiedt op basis van het standaard plan van aanpak, gemaakt door mij en de projectgroepen over de jaren. Wat in de toekomst zal veranderen is dat door meer en meer ervaring mijn tijdsinschatting zal verbeteren. Al zat ik er aanvankelijk goed naast, in de loop van het afstudeertraject kon ik betere inschattingen maken over ontwikkelingen in Visual Dataflex.

9.2.2 Onderzoeken

Het onderzoeken van de verschillende onderdelen heeft veel tijd gekost. Zeker voor wat betreft de programmeertaal. Het uitzoeken van nieuwe mogelijkheden voor bijvoorbeeld Crystal Reports was echter niet erg problematisch, omdat Data Access goede documentatie had gemaakt. Mijn onderzoek naar de werking van de Excel verliep weer wat stroever, omdat de desbetreffende documentatie niet klopte met de werkelijkheid.

Ook het onderzoeken van de werking van het huidige systeem verliep problematisch. Dit onderzoek, samen met dat van de programmeertaal, heb ik onderschat. Niemand kon mij echt helpen met het zoeken naar kennisinzicht van de Micros Visual Dataflex *classes* voor communicatie met derde partijen. Wel kon ik met technische vragen over de programmeertaal bij mijn collega's terecht. Uiteindelijk is het mij gelukt door de code te begrijpen en verwijzingen te zoeken zoals te lezen is in 'Oriëntatie'. (Wel wil ik hier stellen dat het de gemiddelde werknemer, na het volgen van een cursus (!), een jaar kost om de taal goed te begrijpen). Behalve de standaard applicaties moest ik ook rekening houden met maatwerk, dat op de *class* aanwezig kan zijn.

Terugkijkend heb ik enorm veel baat gehad van het langdurige vooronderzoek. Ik begrijp nu de interne werking van de door mij onderzochte onderdelen. Met het begrijpen van de interne werking ontstaat ook meer inzicht in de taal. Toch zal ik een volgend project met Visual Dataflex niet direct toejuichen.

9.2.3 De Crystal Reports definitiestudie

De definitiestudie is gebaseerd, aan de hand van de keuze die Micros B.V. gemaakt heeft, op de informatie, die ik verzameld heb uit vooronderzoek. De lijst met voor- en nadelen is duidelijk voor beide partijen. Als ontwikkelmethode heb ik gekozen voor [IAD]. [IAD] was voor mij een nieuwe en tevens moeilijk, voor mijn situatie te gebruiken, ontwikkelmethode. [IAD] is wel weer makkelijker te gebruiken dan SDM, zeker voor deze situatie. Achteraf denk ik dat meer tijd investeren in het zoeken naar een methode voor onderhoud aan ongedocumenteerde *classes* mij beter uit had gekomen.

De voorgeschreven hoofdstukken van [IAD] heb ik geprobeerd te gebruiken voor de definitiestudie van de Crystal Reports *class*, dit in samenwerking met [UML]. Het gebruik van workshops zou voor de werking van [IAD] verduidelijkend zijn geweest. Voor de wensen die Micros B.V. had zou een vroegtijdige workshop alleen maar interessant zijn geweest. Toch zou ik een workshop daar niet voor willen toepassen. Workshops werken, naar mijn mening, alleen voor grote ingrepen. Het uitzoeken van welke [UML] onderdelen ik, per hoofdstuk van [IAD], kon gebruiken, was lastig. De definitiestudie als proces is goed verlopen zonder vertraging.

9.2.4 Het Crystal Reports Ontwerp rapport

"Hoe zal ik het allemaal noteren" was bij het proces van ontwerpen voor mij behoorlijk lastig. Ik wilde het ontwerp zo maken, dat het overeenkomt met de methodes, die ik via MBIT aangereikt heb gekregen en dat het ontwerp nuttig is voor Micros B.V., zowel nu, als in de toekomst. Echter besloot ik om *class* diagram te gebruiken voor een totaaloverzicht, gebruikmakend van sequentie diagrammen en beschrijvende tabellen, om de functie van de bewerking te verduidelijken. Nog steeds ben ik van mening dat deze 'rechttoe, rechtaan methode' de meest duidelijk is voor ontwerp van bewerkingen voor Micros *classes*.

9.2.5 Ontwikkelen en testen van de Crystal Reports classes

Bij ontwikkelen heb ik gebruik gemaakt van ontwerpdocumentatie. Testen per onderdeel, om te achterhalen of dat onderdeel correct functioneert en een totaaltest tot slot. Een volgende keer zou ik dit niet anders doen. Het proces is goed verlopen, wat ik ontworpen heb, is gecreëerd.

9.2.6 Richtlijnen

Om een exacte invulling van richtlijnen bij Micros B.V. te vinden is lastig. Het proces is moeizaam verlopen omdat de exacte invulling bij Micros B.V. ook niet bekend is. Het invoeren van versiebeheer bij Micros is een langdurig proces dat sinds kort gerealiseerd is. Mijn aanvulling daarop is bedoeld voor de Micros *classes*.

Literatuurlijst

[IAD]
Tolido, R.J.H, IAD Het evolutionair ontwikkelen van informatiesystemen, tweede druk, Academic Service, 1997

[UML]
Warmer, J en Kleppe, A, Praktisch UML, tweede druk, Addison Wesley, 2001

Mulder R.I en Huijzers J, Software-configuratiemanagement, eerste druk, Academic Service, Schoonhoven1999

Data Access Corporation, Training Guide – VDF 8.2, 2000

Internet

UML referenties
http://www.uml.org/

Ideeën over versiebeheer
http://www.ictforyourbusiness.nl/main.asp?chapterid=1494

Technische referenties
http://whatis.techtarget.com/

Informatie over Crystal Reports
http://www.businessobjects.com/

Visual Dataflex
http://www.dataaccess.nl

Afkortingen en Terminologie

Uitleg van de verschillende termen die gebruikt worden binnen dit verslag

.Net	Microsoft .Net is een raamwerk om software technologie te creëren om mensen, informatie, systemen en apparaten met elkaar te verbinden. Zo ontstaat een compleet netwerk, waarbinnen alle verbonden apparaten kunnen communiceren en data uitwisselen.
Abstract Datatype	Een abstract datatype is een zelf gedefinieerd type. Gebaseerd op de standaard datatypes en/of een ander abstract datatype.
Bewerking	Ander woord voor procedure of functie die in *class* gedefinieerd is.
Class	Een *class* is een definitie van bewerkingen en kenmerken. Een *class* bevat geen waarden, alleen variabelen. *Classes* zijn de basis voor object georiënteerd programmeren.
COM object	COM (Component Object Model) is een manier om stukken functionaliteit in te bedden in een container en die te verspreiden COM is dus een manier om objecten los van een applicatie te houden. Van COM objecten is in principe alleen de interface zichtbaar.
Crystal Reports	Crystal Reports is een programma dat het doel heeft om informatie uit een database netjes en leesbaar te tonen.
Crystal Reports Class	Een standaard Visual Dataflex *class* voor het aansturen van Crystal Reports.
DDE	*Dynamic Data Exchange* is bedoeld voor het uitwisselen van informatie tussen verschillende programma's.
DDE macro	Een macro is een opgeslagen sequentie van bepaalde handelingen. DDE Macro communiceert via DDE om informatie op te halen voor zijn handelingen.
flowcharts	Is een grafische representatie van logische gevolgen binnen het programma of een bewerking.
IAD	Een iteratieve ontwikkelmethode voor het ontwikkelen van software
IDE interface	De IDE geeft de mogelijkheid voor het grafische bewerken van objecten. Deze interface kan aangepast worden waardoor het mogelijk is zelf gedefinieerde *classes* binnen de IDE te gebruiken
Inheritance	Overerving in object oriëntatie is dat een *subclass* dezelfde eigenschappen heeft als *(super)class* waar die op gebaseerd is.
Kenmerk	Ander woord voor eigenschap en proporties die in *class* gedefinieerd is.
Library	Is een collectie van gezamenlijk bruikbare classes en functies.
MAPI	MAPI staat voor *Messaging Application Program Interface* en is een Microsoft Windows programma interface dat het mogelijk maakt om binnen een applicatie berichten te verzenden.
MAPI *class*	Een standaard Visual Dataflex *class* voor gebruik van de MAPI
MBIT	Management en Beheer van Informatie Technologie is een studierichting aan de Haagse Hogeschool
MsCrystalReports	De *subclass* van Micros B.V. die de basis is voor het aansturen van Crystal Reports vanuit de Micros Applicaties
Object	Een Object is een gedefinieerde *class*. Het object heeft dezelfde kenmerken en bewerkingen als de *class*. Tevens zijn er weer bewerkingen en kenmerken te definiëren in het object
Object Oriëntatie	Is een methode voor het ontwikkelen van software
Objecten	Een Object is een gedefinieerde *class*. Het object heeft dezelfde kenmerken en bewerkingen als de *class*. Tevens zijn er weer bewerkingen en kenmerken te definiëren in het object

PKG bestanden	*Package* bestanden zijn binnen Visual Dataflex de locatie waar veel gebruikte classes en objecten zijn gedefinieerd.
Print engine	De Crystal Reports *print engine* wordt gebruikt voor het extern aansturen van Crystal Reports. Ook wel Crpe32.dll genoemd
PSD	Pseudo code geschreven representatie van het programma of bewerking
RDC	Report Designer Component is bedoeld voor het aansturen van Crystal Reports in een .Net omgeving.
SDM	Een lineaire ontwikkelmethode voor het ontwikkel van software
Sqlquery	Een SQL *query* is een geformuleerde regel om informatie uit de database te halen.
Subclass	Een *subclass* is een *class* die gebaseerd is op een andere *(super)class*.
Superclass	*Superclass* is een *class* waarop een *(sub)class* gebaseerd is.
Testset	Een lijst van handelingen en verwachte resultaten, die gebruikt wordt voor het testen van een applicatie of een deel daarvan.
UML	Unified Modeling Language is een techniek voor het modeleren van *classes*
VDF 9	Versie 9 van Visual Dataflex. Voor meer informatie zie Visual Dataflex
Versiebeheer	Een stel van procedures om ervoor te zorgen dat software, de componenten en de documentatie altijd up-to-date is.
Visual Dataflex	Visual Dataflex is een object georiënteerde programmeer taal die de basis is voor alle applicaties van Micros B.V.
Zerotype	Zie Abstract Datatype.