ADD.net

Afstudeerder:	Vincent Kok
Studentnummer:	1107993
Eerste examinator:	drs. Maarten Schnerr
Bedrijfsbegeleider:	drs. ing. Peter van Diermen
Organisatie	DEVENTit B.V.
Datum:	Mei 2004

SCRIPTIE IE 2004 kok

VOORWOORD

Nu de afstudeerperiode bijna achter de rug is, wil in dit voorwoord de mensen bedanken die mede verantwoordelijk zijn voor het tot stand gekomen resultaat. Allereerst wil ik mijn bedrijfsbegeleider, Peter van Diermen, bedanken. Mede dankzij hem en de vele discussies met hem is het resultaat dat er nu is tot stand gekomen. Verder de begeleiding vanuit school, Maarten Schnerr, voor het kritisch bedoordelen en het geven van feedback over de ingeleverde documenten. Daarnaast mijn collega's bij DEVENTit B.V. voor de aangename sfeer tijdens de gehele periode.

Vincent Kok, Bunschoten

SAMENVATTING

Bij DEVENTit B.V., een softwareontwikkelingbedrijf, is er besloten om in de toekomst alle sofware te ontwikkelen op het .NET Framework van Microsoft. Het .NET Framework maakt het mogelijk om met de dezelfde programmeertaal en hulpmiddelen, zowel Internet als Windows-applicatie te ontwikkelen.

Hierdoor is het mogelijk om een wens die al langer bestaat binnen het bedrijf, het op uniforme wijze ontwikkelen van zowel Internet als Windowsapplicaties, te vervullen. Om dit te realiseren moet er een toolkit worden ontwikkeld, die de bovengenoemde wens realiteit zou maken. Voor deze toolkit waren er een aantal eisen, het moest gebruik maken van een bestaande applicatie waarmee het mogelijk is om informatie over de databasestructuur van een applicatie in te voeren.Ook moest er een component worden ontwikkeld die het mogelijk maakt voor de applicatieprogrammeur om deze informatie bij het ontwikkelen van software te gebruiken.

Het enige verschil tussen Internet en Windowsapplicaties is de schermafhandeling. Om dit weg te nemen is er gekozen voor het Model-View-Controller (MVC)-pattern. Deze scheidt de data en businesslaag van de presentatielaag.

Ook stond er een component op het programma die het 'disconnected' probleem van het .NET Framework zou oplossen. Dit 'disconnected' wil zeggen dat een gebruiker moet wachten voordat alle data uit de database is geladen voordat hij deze data kan gebruiken. De component zou er voor moeten zorgen dat de gebruiker met een deel van de data kan werken terwijl de rest van de data op de achtergrond wordt binnengehaald. Het ontwikkelen van deze component is door tijdgebrek helaas niet gelukt.

Voor de applicatieprogrammeur is er ten slotte nog een handleiding gemaakt die hem op weg moet helpen met de ontwikkelde toolkit. Naast deze handleiding heeft de applicatieprogrammeur ook nog de beschikking over een aantal applicaties die ontwikkeld zijn om de toolkit te testen.

INHOUDSOPGAVE

FIGUREN

1. INLEIDING

Ter afsluiting van de opleiding Information Engineering, die gevolgd wordt aan de Hogeschool van Utrecht, vindt er een afstudeerproject plaats. Dit project is een 'testcase' om aan te tonen dat de student klaar is om zijn opgedane vaardigheden in de praktijk te brengen.

Het laatste product van dit project is dit document, de scriptie.In deze scriptie wordt het gehele afstudeertraject besproken. Er wordt gestart met een beschrijving van het bedrijf, de uitgangssituatie en de plaats van de afstudeerder daarin.

Vervolgens wordt de opdracht, aanleiding van de opdracht en aanpak besproken. Hierna volgt een bespreking van de analyse en ontwerp dat gedaan is voor de realisatie van de opdracht. In het hoofdstuk dat hierop volgt komen de resultaten aan bod.

Ten slotte worden er een aantal conclusies gemaakt en een aantal aanbeveling gedaan. Ook wordt er teruggekeken naar de afgelopen afstudeerperiode in het hoofdstuk 'evaluatie'.

2. DE ORGANISATIE

2.1 Bedrijf

DEVENTit B.V. is een softwarehuis dat zich richt op de ontwikkeling en het onderhoud van zowel software op maat als productsoftware voor financieel/ administratieve, multimedia, intranet en Internet toepassingen.

Op diverse gebieden heeft DEVENTit B.V. een vooraanstaande positie verworven. DEVENTit B.V is bijvoorbeeld met Atlantis, een platform voor multimediale archiefsystemen, marktleider voor gemeentelijke, regionale en landelijke archiefinstellingen.

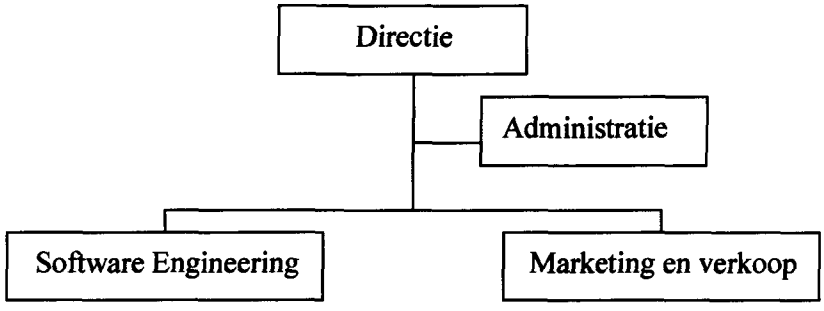
Daarnaast biedt het met Thematis®, een generieke oplossing voor het zoeken binnen meerdere op zichzelf staande informatiebronnen.

Voor Exact Software Centers en Exact Dealers is DEVENTit B.V. de partner voor de realisatie van maatwerk aan Exact Software. De integratie van Internet binnen reguliere toepassingen of volledig op het Internet gebaseerde oplossingen heeft een sterk groeiend aandeel in de activiteiten van het bedrijf.

De basis voor de softwareontwikkeling is objectoriëntatie waardoor een hoge mate van flexibiliteit, hergebruik en robuustheid van de programmatuur wordt bereikt.

2.2 Structuur

DEVENTit B.V. is een platte organisatie te noemen, het bestaat namelijk uit een directie met daaronder direct de afdelingen Software Engineering en Marketing en verkoop. Ten slotte is er nog de afdeling Administratie die een ondersteunende functie heeft voor de overige afdelingen. Het bedrijf bestaat uit 8 vaste mensen en een aantal parttimers. De organisatie is in het onderstaande organigram schematisch weergegeven.



Figuur 1: Organigram DEVENTit B.V.

2.3 Plaats van afstudeerder

Als afstudeerder ben ik de afgelopen periode werkzaam geweest binnen de afdeling Software-Engineering. Deze afdeling bestaat uit een aantal programmeurs die naast het programmeren van software aan het gehele traject van softwareontwikkeling bijdragen. Dit begint bij het eerste contact met de klant tot en met de aflevering van de software bij deze klant. Verder handelen zij ook de eventuele support die nog volgt wanneer de klant problemen mocht hebben met de software af.

In mijn opdracht was mijn directe opdrachtgever het bedrijf zelf, dus ik had verder niets te maken met externe partijen. De reden hiervoor is het feit dat mijn opdracht bestaat uit het maken van een toolkit die intern gebruikt gaat worden bij het ontwikkelen van software.

3. OPDRACHT

3.1 Aanleiding

Bij DEVENTit is er voor gekozen om nieuwe software te ontwikkelen met behulp van het Microsoft .NET Framework. Met dit framework is het mogelijk om met dezelfde programmeertaal en ontwikkelomgeving zowel Internet als traditionele Windowsapplicaties te ontwikkelen.

Om zowel Internet als Windows-applicaties op een uniforme manier te kunnen ontwikkelen, zal er een toolkit moeten worden ontwikkeld die overweg kan met beide typen applicaties. Ook zal deze toolkit gebruik moeten maken van de faciliteiten die worden aangeboden door het Microsoft .NET Framework.

3.2 Omschrijving

De opdracht bestaat uit het ontwikkelen van een toolkit, waarmee op een uniforme manier zowel Internet als Windows-applicaties ontwikkeld kunnen worden. Voor de toolkit zal er allereerst een ontwerp moeten worden gemaakt, dat verder bouwt op het al gedane onderzoek. Het ontwerp zal bestaan uit zowel een functioneel- als technisch ontwerp.

Binnen de opdracht bevindt zich nog een subopdracht. In de toolkit dient namelijk een databasecomponent te komen die om kan gaan met grote hoeveelheden data. De standaardcomponenten halen namelijk eerst alle data uit de database, voordat de gebruiker er mee kan werken. Bij grote hoeveelheden data resulteert dit echter in een onwerkbare situatie.

3.3 Doelstellingen

De te ontwikkelen toolkit zal voldoende functionaliteit moeten bieden om de volgende doelstellingen te kunnen bereiken.

- Minimaliseren van het verschil tussen het ontwikkelen van Internet en Windows-applicaties.
- Het zoveel mogelijk automatiseren van de schermafhandeling.
- Hergebruiken van generieke applicatielogica.
- Te integreren met de ontwikkelomgeving bij het maken van user-interfaces.
- Gebruik maken van de informatie die beschikbaar is vanuit de bestaande ADD invoermodule (deze zal in een komend hoofdstuk worden besproken).

3.4 Uitgangssituatie

Bij DEVENTit wordt al gebruik gemaakt van een toolkit voor het ontwikkelen van Windows-applicaties. Deze toolkit, de Algemene Data Dictionary (ADD), is echter ontwikkeld voor het ontwikkelen van C++ applicaties en niet geschikt voor het gebruik voor internetapplicaties. Wel kunnen methodes en technieken die gebruikt worden in deze toolkit gebruikt worden voor nieuwe toolkit. Bij deze toolkit hoort ook een Windows-applicatie, waarin het mogelijk is om het relationele model van een applicatie in te voeren. Deze applicatie zal ook gebruikt worden voor de nieuwe toolkit en zal daarom aangepast worden.

Voor het loskoppelen van de presentatielaag is door een collega al onderzoek gedaan. Dit loskoppelen is nodig om de toolkit geschikt te maken voor zowel Internet als Windows-applicaties, de schermafhandeling tussen deze typen applicaties is namelijk verschillend. Aan de hand van dit onderzoek is besloten om gebruik te maken van het Model-View-Controller (MVC) pattern. Dit pattern is speciaal ontwikkeld voor het loskoppelen van de presentatielaag van de andere lagen. Ook is er door deze collega een model gemaakt, met daarin een uitwerking van het MVC pattern in de context van de te ontwikkelen toolkit.

Ten slotte is het belangrijk dat de toolkit integreert met de huidige ontwikkelomgeving. Deze ontwikkelomgeving maakt het namelijk mogelijk om op eenvoudige wijze schermen te maken volgens het 'drag-and-drop' principe. Deze functionaliteit moet te gebruiken zijn in combinatie met de toolkit.

3.5 Aanpak

Bij aanvang van het afstuderen is er allereerst begonnen met de aanwezige documentatie over het project. Door een collega was reeds onderzoek gedaan naar het MVC-pattern, dit omdat dit onderdeel van het project de grootste risicofactor was. Verder is er gekeken naar de aanwezige literatuur over dit onderwerp. Naast het MVC-pattern zijn alle andere werkzaamheden ook geïnventariseerd, om zo enig idee te krijgen over de werkzaamheden die gedaan moesten worden tijdens de afstudeerperiode.

De werkzaamheden en de planning zijn vastgelegd in het plan van aanpak (zie bijlage B). Op de manier van werken zoals in het plan van aanpak is vermeld is echter in een vroeg stadium van het project gewijzigd. In het plan van aanpak stond vermeld dat de tweede fase zou bestaan uit het ontwerpen van de componenten, om deze vervolgens in de daarop volgende fase te realiseren. Er is echter voor gekozen om dit in een iteratief proces te doen, zodat er steeds een gedeelte ontwerp zou plaatsvinden, dit ontwerp te implementeren en te testen om vervolgens weer met een volgend onderdeel te beginnen. Om het project beheersbaar te houden is er voor de manier van werken gekozen.

Om een goede testsituatie te verkrijgen is er een gedeelte van een bestaande applicatie genomen, die intern gebruikt wordt voor relatiebeheer, te herschrijven met de ADD.NET toolkit.

Ten slotte stond er nog het ontwerp en ontwikkeling van een database-component op het programma. Hier is echter niet van gekomen, mede door het uitlopen van het ontwerp van het MVC gedeelte van het project.

3.6 Methoden en technieken

Tijdens de afstudeerperiode is gebruik gemaakt van verschillende methoden en technieken. Deze zijn in te delen in 4 gebieden. Ik zal hieronder voor elk van die gebieden de gebruikte methode en technieken kort beschrijven.

3.6.1 Projectmanagement

Om een project in goede banen te leiden is het belangrijk dat vooraf de werkzaamheden worden bepaald, zodat deze binnen het vooraf gestelde tijdsbestek uitgevoerd kunnen worden. Dit is bij aanvang van het project gedaan door een inventarisatie van de eisen. Aan de hand van deze inventarisatie is het plan van aanpak (zie bijlage B) geschreven. Dit plan heeft tijdens het project als leidraad gefunctioneerd. Hierbij valt echter één kanttekening te plaatsen. In het plan

van aanpak wordt namelijk uitgegaan van een fase ontwerp en een fase ontwikkeling. Deze zijn echter in elkaar gegroeid om een iteratief proces mogelijk te maken van ontwerp – ontwikkeling. Kort gezegd zijn de fases ontwerp en ontwikkeling samengevoegd tot één fase.

3.6.2 Ontwerp

Voor het ontwerpen van de software is gebruik gemaakt van de Unified Modelling Language (UML). Dit is gedaan omdat dit de huidige standaard is voor het ontwerpen en modelleren van objectgeoriënteerde software. Verder wordt deze methode ondersteund door de bij het bedrijf gebruikte CASE-tool, een softwarepakket voor het maken van UML-ontwerpen. Verder biedt een dergelijke tool ook vaak de mogelijkheid om de diagrammen om te zetten naar programmacode. Hier is echter geen gebruik van gemaakt, omdat de gebruikte programmeertaal nog niet door het pakket ondersteund werd.

3.6.3 Ontwikkeling

De software is ontwikkeld in de programmeertaal C#, uitgesproken als C Sharp, wat onderdeel uitmaakt van het Microsoft .NET Framework. De reden voor het gebruik van C# en het framework, komt voornamelijk door de keuze van het bedrijf. DEVENTIt B.V. heeft er namelijk voor gekozen om alle software in de toekomst hierop te baseren.

Dit is gedaan omdat het .NET Framework het verschil tussen Internet en Windowsapplicaties tot een minimum beperkt, doordat er met dezelfde taal en hulpmiddelen ontwikkeld kan worden.

Daarnaast is er gebruik gemaakt van design-patterns. Dit omdat design-patterns doordachte en in de praktijk bewezen oplossingen zijn voor softwareproblemen. Het gebruikte MVC-pattern sloot goed bij de eisen van de oplossing aan en was hierdoor de aangewezen kandidaat om gebruikt te worden in de oplossing.

Ten slotte moesten er ook nog enige aanpassingen gedaan worden aan de ADD invoermodule (zie hoofdstuk 4.2 voor meer informatie over deze module). Omdat deze in C++ is geschreven, is ook van deze programmeertaal gebruik gemaakt.

3.6.4 Testen

Om kwaliteit van software te kunnen garanderen moet deze goed worden getest. Een deel van de software een puur informatieve functie heeft, het stelt namelijk alleen meta-informatie beschikbaar, kan dit geautomatiseerd worden getest doormiddel van een Unit test.

Bij Unit testen wordt er uitgegaan van een component met een vooraf bepaald aantal functies (de zogenaamde interface). Deze functies zullen altijd een vooraf bepaalbaar resultaat teruggeven, wanneer deze goed zijn geïmplementeerd. Bij een Unit test zullen alle functies van een component worden aangesproken, om vervolgens het resultaat te testen op een vooraf gestelde waarde. Op deze manier kan een component snel worden getest en kunnen gemaakte fouten bij wijzigingen snel worden gedetecteerd.

Naast het informatieve gedeelte van de software is er nog een gedeelte dat voor de schermafhandeling zorgt. Dit is niet te testen doormiddel van een Unit test, daarom is er ervoor gekozen om dit te testen doormiddel van een testapplicatie. Hiervoor is een bestaande applicatie genomen, om deze vervolgens gedeeltelijk te herschrijven met behulp van de nieuwe toolkit.

3.6.5 Verslaglegging

Als afsluiting van het afstudeertraject is er een scriptie geschreven. Hiervoor is gebruik gemaakt van de aangeleerde technieken voor het opstellen van een goed rapport. Daarnaast is ook de richtlijn uit de afstudeerleidraad gebruikt. Verder is er gebruik gemaakt van opgedane vaardigheden bij het schrijven van eerdere rapporten.

3.7 Producten

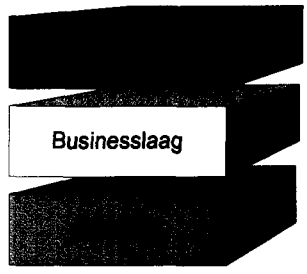
Uit de afgelopen periode zijn een aantal producten voortgekomen. In het plan van aanpak zijn deze terug te vinden als producten van een bepaalde fase. Alle producten, op de database-component na, zijn de afgelopen periode gerealiseerd. Hieronder een opsomming met een korte beschrijving.

- Plan van aanpak
  Het plan met daarin de gehele beschrijving van het traject. Ook is in dit document de planning opgenomen.
- Ontwerp
  Het ontwerp bestaat uit twee delen. Allereerst een functioneel ontwerp met daarin de functionele eisen. Deze eisen zijn vervolgens technisch uitgewerkt in een technisch ontwerp.
- ADD.NET toolkit
  Het uiteindelijke resultaat, bestaande uit een serie softwarecomponenten die voldoen aan de eisen gesteld in het functionele en technische ontwerp.
- Testprogrammatuur
  Om de toolkit aan een gedegen praktijktest te onderwerpen, zal dit bestaan uit een Internet en Windowsapplicatie die gebruikt maakt van alle beschikbare functionaliteit van de toolkit.
- Documentatie
  Om de toolkit voor programmeurs bruikbaar te maken zal deze goed gedocumenteerd moeten zijn. De documentatie zal in de vorm van een tutorial geschreven zijn.
- Afstudeerrapport
  Vervolgens wordt er een is er nog het afstudeerrapport geschreven met daarin alle bevindingen van het afstudeertraject.
- Presentatie
  Ter afsluiting van het afstudeertraject zal er een presentatie worden gegeven waarin het gehele afstudeertraject voor de afstudeercommissie en belangstellenden gepresenteerd wordt

4. PROBLEEMANALYSE

4.1 Applicatiestructuur

Een standaard applicatie is te verdelen in een aantal lagen. Deze zijn in het onderstaande diagram weergegeven.



Figuur 2: Applicatiestructuur

Om snelle ontwikkeling van een applicatie mogelijk te maken, zal zoveel mogelijk functionaliteit voor een applicatie worden hergebruikt. In een applicatie kost de schermafhandeling samen met de communicatie met de database de meeste ontwikkeltijd. Bovendien gaat het veelal om dezelfde bewerkingen. Om de applicatieprogrammeur zoveel mogelijk werk uit handen te nemen, zullen hiervoor standaardoplossingen komen.

Verder kan applicatielogica worden gestandaardiseerd door bijvoorbeeld validatie en standaardwaarden per datatype te implementeren. Hierdoor hoeft een applicatieprogrammeur alleen nog maar de uitzonderingen te implementeren.

Voor zowel het standaardiseren van de databaselaag als de schermafhandeling, zal de applicatie kennis moeten hebben van het relationele model van de database waarin de applicatie haar gegevens bewaard. Om deze gegevens in te voeren is er al een tool (de ADD) aanwezig, deze zal in de volgende paragraaf worden beschreven.

4.1.1 Internet vs. Windows-applicaties

Belangrijk is dat ADD.NET inzetbaar is voor zowel Internet als Windows-applicaties. Met de introductie van het Microsoft .NET Framework is het verschil tussen deze applicaties teruggebracht tot de presentatielaag van beide typen applicaties. Het grootste verschil in deze laag is het feit dat bij een Internetapplicatie de presentatie gebeurd op zowel de client als de server, terwijl dit bij een Windows-applicatie alleen plaatsvindt op client.

Verder is de interactie tussen gebruiker en applicatie bij een Internetapplicatie veel beperkter dan bij een Windows-applicatie. Dit komt doordat een deel van de functionaliteit afhankelijk is van de browser waarin de applicatie werkt.

Voor de business- en datalaag kan er gebruik gemaakt worden van dezelfde functionaliteit.

4.2 ADD Invoermodule

Voor de huidige ontwikkelomgeving is een Windows-applicatie beschikbaar, waarin het complete relationele model waarop de applicatie gebaseerd is kan worden ingevoerd. Hieronder vallen de tabellen met daarin één of meerdere velden. Tussen tabellen kunnen

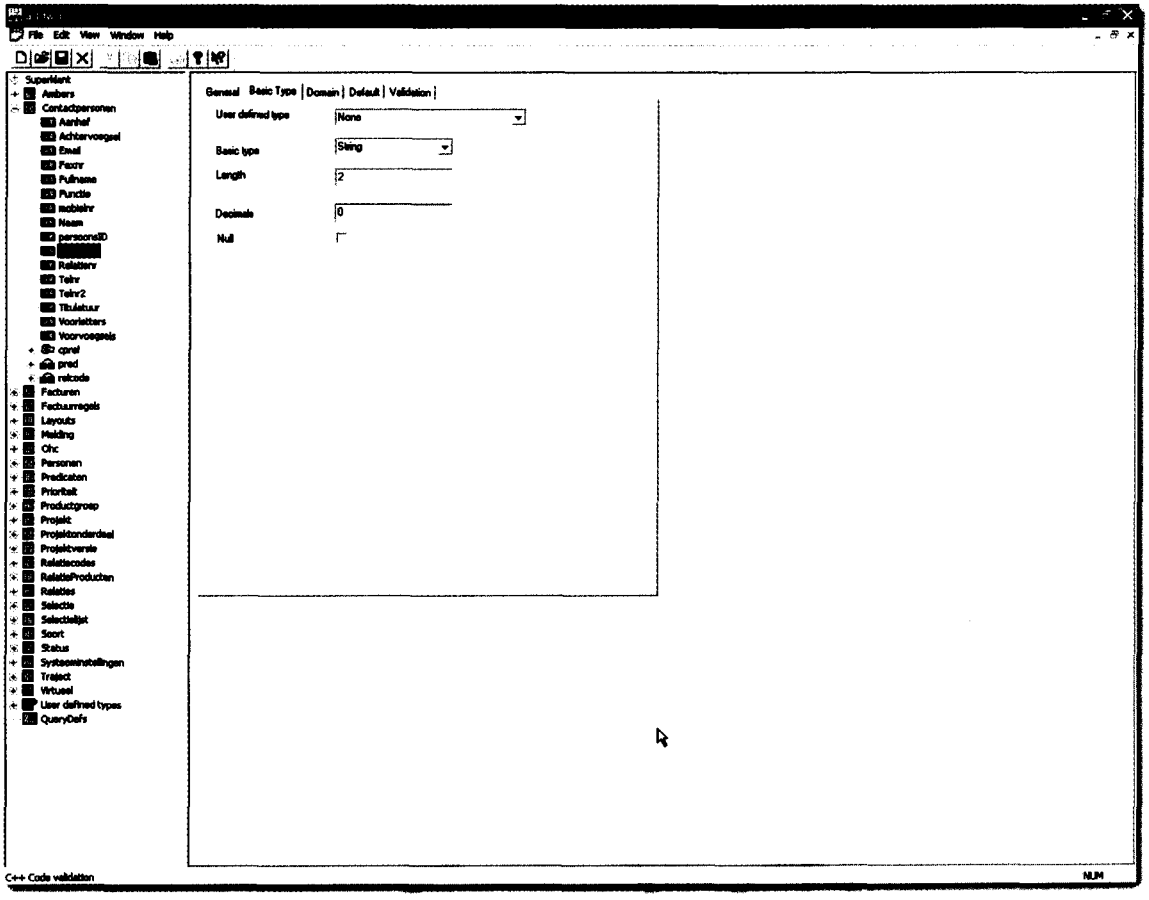
relaties worden gelegd. Per veld wordt er veel extra informatie opgeslagen. Hierbij valt te denken aan keys, foreign keys, datatypen, domeinen, standaardwaarden, helpteksten etc.

In het onderstaande figuur (Figuur 3) is de invoermodule te zien. In de boomstructuur is het relationele model van een applicatie te zien. In het rechter gedeelte van het scherm staat meer uitgebreide informatie over de geselecteerde knoop in de boom.

Deze informatie wordt door de huidige ADD gebruikt om schermen van data uit de database te voorzien, om ingevoerde data uit het scherm te lezen en deze op te slaan in de database, het valideren van ingevoerde waarden, het tonen van helpteksten, het tonen van foutmeldingen en het alvast invullen van een defaultwaarde.

Met behulp van de ingevoerde relationele data kan de ADD invoermodule ook een aantal bestanden genereren. Dit zijn SQL-scripts, C++ code en resourcebestanden.

Om deze invoermodule te kunnen gebruiken voor de nieuw te ontwikkelen toolkit, zal deze moeten worden aangepast en uitgebreid. Hier bij valt te denken aan de mogelijkheid voor het genereren van C# code.



Figuur 3: Interface ADD invoermodule

4.3 Design-patterns

Design-patterns zijn objectgeoriënteerde modeloplossingen, die helpen veel voorkomende problemen die zich voordoen tijdens het ontwerpen van objectgeoriënteerde software. In de ADD.NET Toolkit worden een aantal design-patterns gebruikt. Deze zijn als volgt.

- Composite-pattern
  Met behulp van dit pattern is het mogelijk om objecten hiërarchisch te kunnen opdelen. Een object kan zich op iedere niveau in de hiërarchie bevinden, terwijl ieder object er 'buitenaf' hetzelfde uitziet.
- Observer-pattern
  Met behulp van dit pattern kunnen objecten zich 'abonneren' op een object. Wanneer het object wijzigt zullen alle object die geabonneerd zijn op het object een signaal krijgen.
- Factory-pattern
  Dit pattern creëert objecten die gebaseerd zijn op een bepaald object. De factory baseert zijn keuze op basis van een meegegeven parameter.
- Model-View-Controller (MVC)-pattern
  Verdeelt een applicatie in drie lagen. Basisfunctionaliteit en data (Model), Presentatie (View) en het afhandelen van events (Controller).

4.4 Model-View-Controller (MVC)

Voor het scheiden van de presentatielaag van de rest van de applicatielogica is gekozen voor het MVC-pattern. Dit pattern zorgt ervoor dat een applicatie in drie lagen wordt verdeeld. De volgende lagen zijn te onderscheiden.

- Model, verantwoordelijk voor de data en kernfunctionaliteit.
- View, verantwoordelijk voor het tonen van de data.
- Controller, verantwoordelijk voor het afhandelen van de interactie tussen het programma en de gebruiker.

In de beschrijving van het pattern wordt uitgegaan van één model met daarop één of meerdere views. Een view wordt gezien als een bedieningselement op het scherm.Elke view heeft zijn eigen controller en toont de gegevens uit het model op een specifieke manier. Zo kan de ene view de gegevens in tabelvorm tonen terwijl de andere de data in een grafiek weergeeft.

5. ONTWERP

5.1 Componenten en packages

De ADD.NET oplossing bestaat uit een set van componenten die met elkaar communiceren via de interface van een component. Voor ADD.NET zijn de volgende componenten te identificeren.

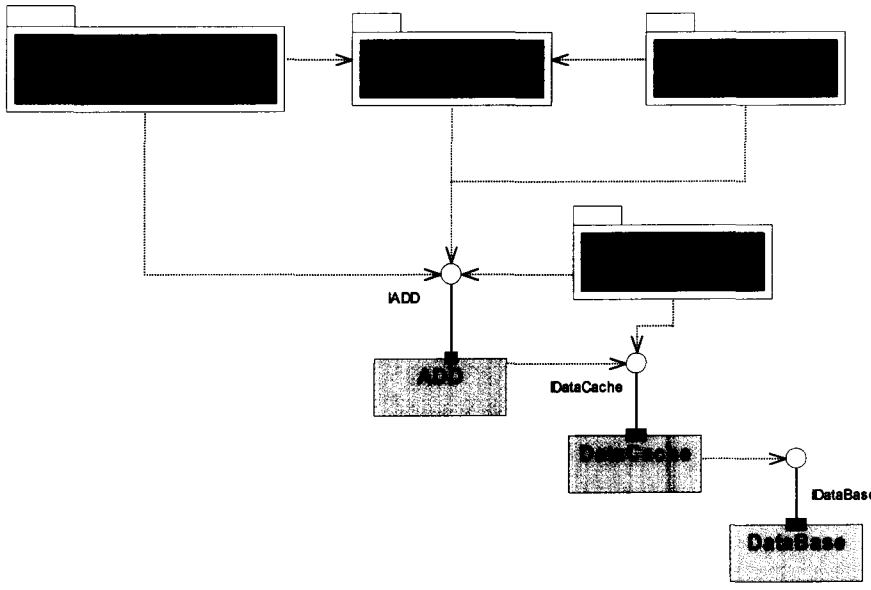
- ADD-component
  De ADD-component heeft een ondersteunende functie. Het zorgt er namelijk voor dat de informatie, die ingevoerd is in de ADD invoermodule, beschikbaar is voor de andere componenten. Naast het beschikbaar maken van deze informatie moet het met de ADD-component mogelijk zijn om volwaardige SQL-queries op te bouwen. Dus compleet met velden, tabellen, joins, where-clause en sorteringen. Verder levert deze component de basisschermen, waarvan de applicatiespecifieke schermen kunnen 'erven'.
- DataCache-component
  De DataCache-component moet het 'disconnected' probleem van ADO.NET oplossen.

  Deze component zal ervoor zorgen voor een emulatie van 'connected' databasetoegang. Dit houdt in dat er na uitvoeren van een SQL-query direct toegang is tot een deel van de gegevens terwijl het verdere resultaat nog wordt opgehaald.
- Database-component
  Het Database Management Systeem (DBMS) waar alle data van de applicatie is opgeslagen.

Naast componenten is een gedeelte van de oplossing ondergebracht in packages, die bestaan uit verzamelingen van klassen:

- MVC-package
  Dit package is voornamelijk verantwoordelijk voor de presentatie van de gegevens en de interactie tussen het programma en de gebruiker en vormt de brug tussen de presentatielaag en de rest van de applicatielogica.
- WindowsApplicatie-package
  De collectie specifieke schermen van een Windows-applicatie met daarin de bijbehorende logica.
- WebApplicatie-package
  De collectie specifieke schermen van een Webapplicatie en de bij behorende logica.
- Applicatielogica-package
  Alle niet schermgebonden logica die er nodig is voor een werkende applicatie. Deze logica is voor zowel Web als Windows-applicaties te gebruiken.

De onderlinge relatie tussen de verschillende packages en componenten is in het onderstaande diagram weergegeven.



Figuur 4: Structuur ADD.NET Toolkit

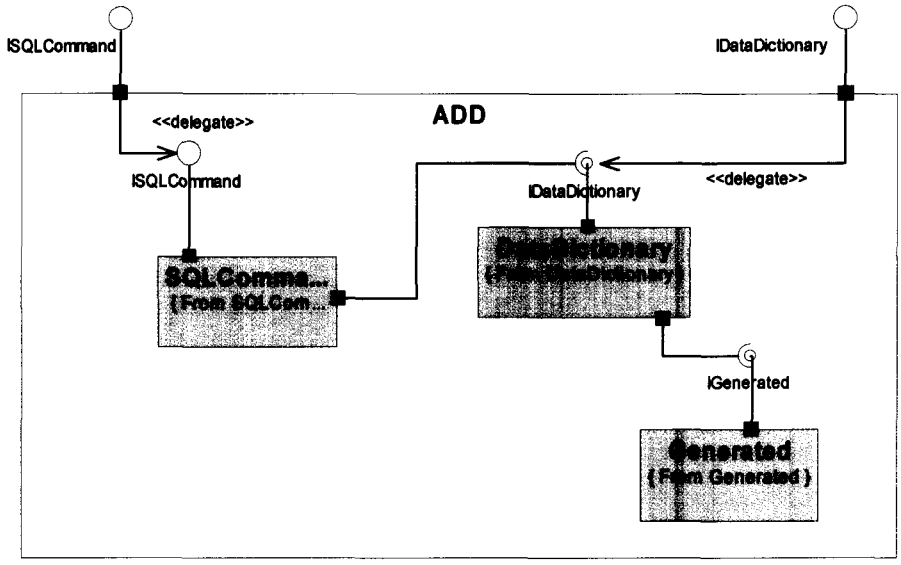
In het bovenstaande schema zijn de vooral de ADD-component, de DataCache-component en het MVC-package van belang. De overige onderdelen zijn in het schema geplaatst om te laten zien hoe deze samenwerken met de drie belangrijke onderdelen. Deze drie onderdelen vormen samen ook de eigenlijke ADD.NET oplossing.

5.2 ADD-Component

De ADD-Component bestaat uit drie verschillenden subcomponenten met elke hun eigen verantwoordelijkheden. De component heeft de volgende subcomponenten.

- DataDictionary
  Geeft toegang tot alle meta-informatie zoals die is ingevoerd in de ADD invoermodule.
- SQLCommand
  Met behulp van de gegevens uit de DataDictionary is het mogelijk om met deze component SQL-query's op te bouwen, en daarmee data uit de database te selecteren, in te voeren, te wijzigen of te verwijderen zonder dat de applicatieprogrammeur één regel SQL hoeft in te voeren.
- Generated
  De programmacode die gegenereerd wordt door de ADD invoermodule is via deze component te benaderen.

In het onderstaande UML-schema zijn de componenten en de onderlinge relaties weergegeven. In dit schema is ook goed te zien dat de 'Generated' component alleen wordt gebruikt door de 'DataDicionary' component en dat deze verder niet extern beschikbaar is.

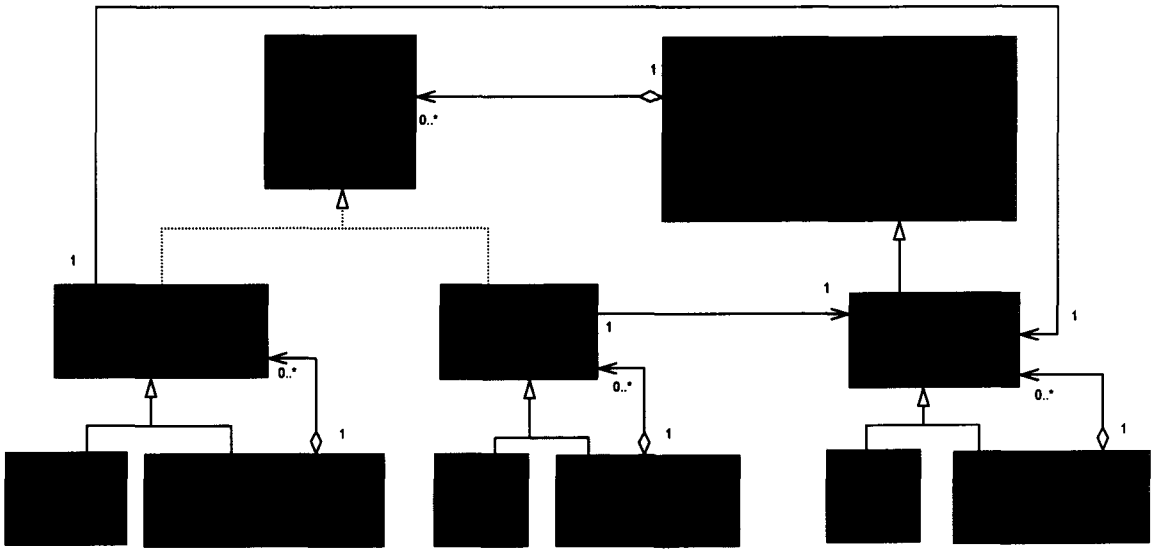


Figuur 5: Componenten-diagram ADD-component

5.3 MVC

5.3.1 Design-Pattern

MVC is een standaard design-pattern en ziet er, weergegeven in een class-diagram, als het volgt uit.



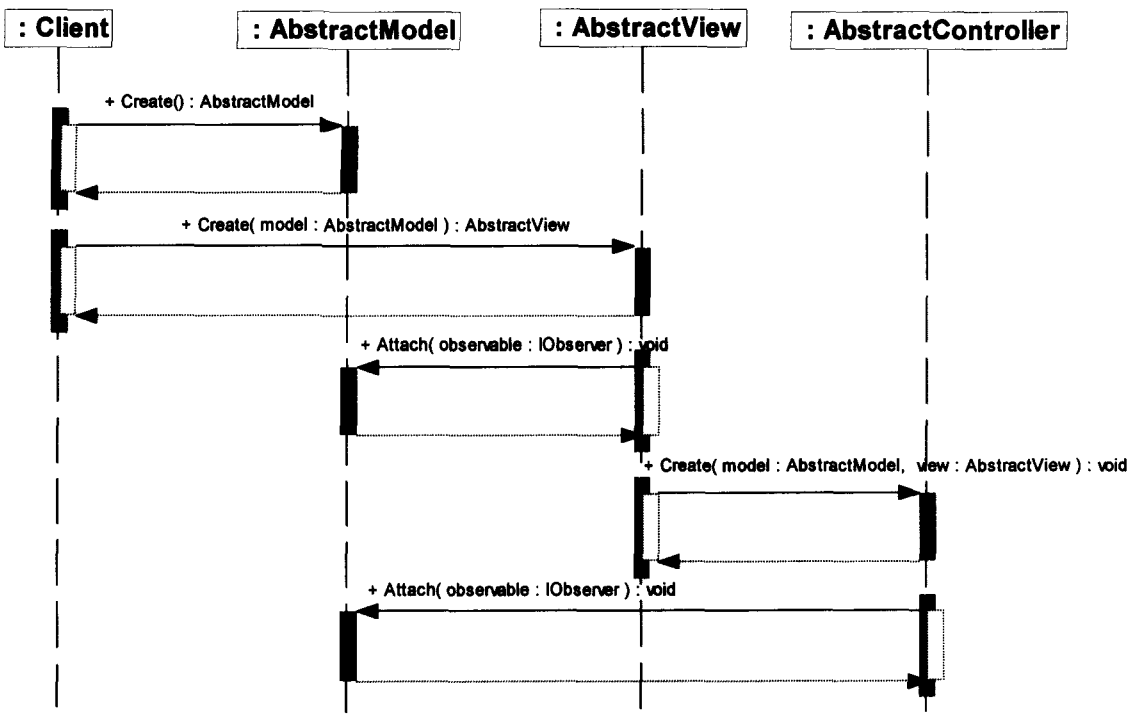
Figuur 6: Class-diagram MVC-pattern

In het schema zijn de drie onderdelen van het pattern goed te zien in de vorm van de classes 'AbstractModel', 'AbstractView' en 'AbstractController'. Deze classes zijn abstract en vormen

hierdoor een basis voor de werkelijke implementatie van deze drie elementen. Een abstracte class is een class die niet geïstantieerd kan worden. Hierdoor is het mogelijk om de structuur van een oplossing vast te leggen en de werkelijke implementatie vrij te laten aan de applicatieprogrammeur. Het model is het hoofdelement en de classes 'AbstractView' en 'AbstractController' hebben hier beiden een relatie mee.

De classes 'AbstractView' en 'AbstractController' implementeren beide de interface 'IObserver' waardoor deze classes in een 'Subject' geplaatst kunnen worden. Dit is een implementatie van het Observer-pattern. Hierdoor is het mogelijk om de classes die deze interface implementeren op de hoogte te brengen van een wijziging.

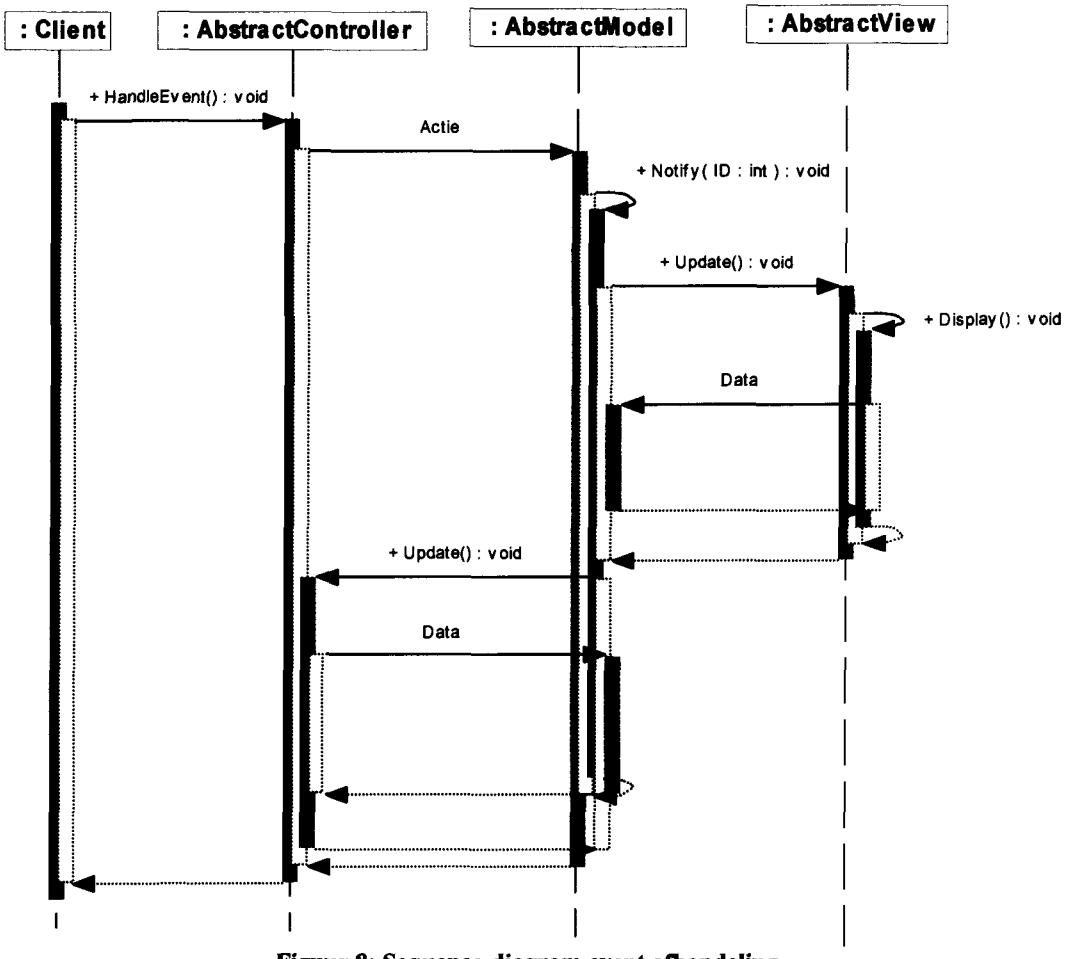
Om het instantieerproces van de classes goed in beeld te brengen, is dit in het onderstaande sequence-diagram weergegeven.



Figuur 7: Sequence-diagram MVC instantiatie

1. Een 'AbstractModel' gecreëerd.
2. Een 'AbstractView' wordt gecreëerd met als parameter een het zojuist gecreëerde 'AbstractModel'.
3. De 'AbstractView' koppelt zichzelf aan zijn 'AbstractModel'
4. De 'AbstractView' creëert een 'AbstractController' met als parameters zijn 'AbstractModel' en zichzelf.
5. De 'AbstractController' koppelt zichzelf aan zijn 'AbstractModel'.

Wanneer alle onderdelen geïnitialiseerd zijn, kan er worden begonnen met het afhandelen van 'events'. Een event is een gebeurtenis in de software, waarop een applicatieprogrammeur kan reageren. Een voorbeeld hiervan is een knop die ingedrukt wordt, of de waarde van een veld data veranderd. Het opvangen van 'events' wordt gedaan door de controller. Het hele proces voor het afhandelen van 'events' is te zien in het onderstaande diagram.



Figuur 8: Sequence-diagram event afhandeling.

1. Bij het binnenkomen van een event, wordt deze doorgegeven aan de juiste controller.
2. De controller roept bij het model een actie, bijvoorbeeld een validatie functie, aan.
3. Door het aanroepen van een dergelijke actie kan de data in het model gewijzigd zijn. Daarom roept het model de functie 'Notify' aan.
4. Door het aanroepen van deze functie worden de functies van de bijbehorende view en controller aangeroepen.
5. De view roept hierbij zijn 'eigen' functie 'Display' aan. Deze functie haalt de eventueel gewijzigde data op bij zijn model en toont deze op de view specifieke manier.
6. De controller haalt bij het aanroepen van de functie 'Update' de eventueel gewijzigde data op bij zijn model en kan op basis hiervan beslissen om bepaalde events te 'disablen'.

5.3.2 Implementatie

Nu de werking van MVC bekend is, kan er nu worden gekeken naar de daadwerkelijke implementatie van het pattern. Hierbij zijn vier onderdelen te onderscheiden te weten.

- Classes die toebehoren aan het MVC pattern.
  Dit zijn de classes die onderdeel uitmaken van het pattern. Deze zijn te zien in figuur 6.
- Classes die toebehoren aan de MVC toolkit.
  De classes die verder bouwen op het pattern. Dit zijn classes die zijn afgeleid zijn van de classes die onderdeel uitmaken van het pattern. Dit zijn de geel gekleurde classes in bijlage A.6 en A.7.
- Classes die door de applicatieprogrammeur zelf geschreven dienen te worden en daarmee dus onderdeel zijn van de specifieke applicatielogica.
  De classes die de MVC toolkit uitbreiden en een specifieke implementatie vormen voor een bepaalde applicatie. Dit zijn de groen gekleurde classes in bijlage A.6 en A.7
- Classes die onderdeel uitmaken van het .NET Framework.
  De classes die meegeleverd worden met het .NET Framework van Microsoft. Deze zijn dus geschreven door programmeurs van Microsoft en vormen de basis van het .NET Framework. In bijlage A.6 en A.7 zijn deze blauw gekleurd.

Omdat de implementatie verschilt tussen een Internet en een Windows-applicatie, zullen deze apart van elkaar worden besproken.

Windowsapplicatie

Een Windows-applicatie bestaat altijd uit een hoofdscherm (het zogeheten 'MainFrame') met daarin één of meerdere schermen. Op deze schermen bevinden zich weer 'Controls' zoals een 'Editbox', 'Combobox' of 'Button'. Al deze controls hebben hun eigen 'Events' en hun eigen manier voor het tonen van data. Daarom zal er voor elk type control dat gebruikt wordt een eigen View en Controller gemaakt moeten worden. Deze zullen het afhandelen van de specifieke events en het tonen van de data voor hun rekening nemen. Het volledige UML-diagram is opgenomen in bijlage A.6.

Internetapplicatie

Een Internetapplicatie verschilt ten opzichte van een traditionele Windows-applicatie. Zo verschillen de afhandeling van events en het tonen van data. Verder is geen 'MainFrame' aan te wijzen, maar bestaat een Internetapplicatie uit en serie pagina's die opgeroepen worden door middel van hyperlinks.

Een ander belangrijk punt, is dat een Internetapplicatie stateless is. Dit houdt in, dat een pagina iedere keer zijn gegevens kwijt is, wanneer deze opnieuw opgevraagd wordt. Om dit probleem op te lossen heeft een applicatieprogrammeur de beschikking over een Sessie-object. Met behulp van dit object heeft de applicatieprogrammeur de beschikking over de gegevens die hij nodig heeft. In het diagram is dit principe te zien in het feit dat de objecten geen directe relatie hebben met het 'Page' object, maar dat deze relatie via het 'HttpSession' object verloopt. Het volledige UML-diagram is opgenomen in bijlage A.7.

5.4 Relaties

Om data naar de database te kunnen schrijven dienen de tabellen in de juiste volgorde wegschreven te worden. Een factuurregel kan bijvoorbeeld alleen naar de database worden

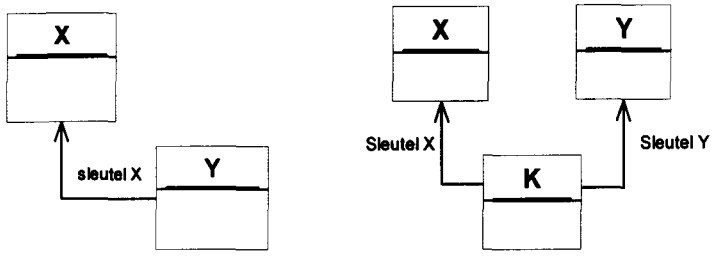
geschreven indien de factuur daar al bestaat. Het in de juiste volgorde zetten van de tabellen kan gedaan worden met behulp van de aanwezige informatie, over de relaties onderling, uit de DataDictionary.

5.4.1 1 op N relaties

1 op N relaties zijn de meest eenvoudige relaties die er in een database kan voorkomen. Dit is een relatie tussen de tabellen X en Y waarbij een record van X een relatie kan hebben met meerdere records in tabel Y terwijl een record in tabel Y maar een relatie heeft met een record in tabel X. Een goed voorbeeld hiervan is de relatie tussen facturen en factuurregels.

5.4.2 N op N relaties

Naast bovenstaande relatie bestaan er ook nog N op N relaties. Dit houdt in dat records in beide tabellen meerdere relaties met elkaar kunnen hebben. Zo kan een record in tabel X een relatie hebben met meerdere records uit tabel Y en een record uit tabel Y kan een relatie hebben met meerdere records uit tabel X. Om dit mogelijk te maken dient er gebruik gemaakt te worden van een koppeltabel, met daarin een sleutel uit beide tabellen. Een goed voorbeeld hiervan zijn rechten en gebruikersgroepen. Een recht kan namelijk bij meerdere groepen horen en een groep kan meerdere rechten hebben.



Figuur 9: 1 op N en N op N relaties

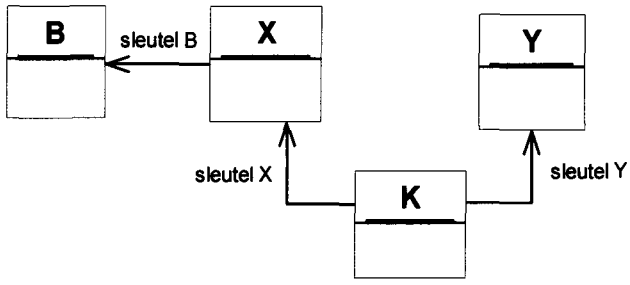
5.4.3 Detecteren relaties

Om relaties te herkennen, zal er een algoritme moeten komen die in staat is om op basis van twee tabellen de relaties daartussen te herkennen. Ook al zijn deze niet direct aan elkaar gekoppeld.

Als uitgangssituatie voor het detecteren van relaties zal steeds een koppel van twee tabellen genomen worden. Dit detecteren van relaties zal voor beide typen relaties dienen te werken. Verder moet het algoritme om kunnen gaan met combinaties van beide relaties.

Het detecteren van 1 op N relaties zijn vrij eenvoudig, dit omdat de relatie tussen beide tabellen direct is. Het detecteren van relaties tussen N op N relaties is mogelijk door het detecteren van alle relaties die naar zowel tabel X als tabel Y lopen. Wanneer daar voor beide tabellen tabel K uitkomt, is er sprake van een koppeltabel.

Wanneer de relaties dieper genest zijn, zie hiervoor de onderstaande figuur, zal het zo zijn dat de relatie niet direct wordt gevonden. Het algoritme zal in dat geval recursief moeten zoek totdat deze de relatie gevonden is.



Figuur 10: Combinatie van relaties

Zo zal bij het zoeken van het pad van tabel B naar Y eerst tabel X gevonden worden. Vervolgens dient er gekeken te worden hoe of er een relatie is tussen X en Y. Wanneer het algoritme volgens dit recursieve patroon werkt, zal het niet uitmaken hoe diep de relaties genest zijn, het zal namelijk altijd gevonden worden.

Quicksort

De eerste manier voor het controleren van afhankelijkheden is door de gehele lijst met tabellen te sorteren doormiddel van het quicksort algoritme. Dit algoritme neemt steeds twee tabellen en bekijkt vervolgens de afhankelijkheden onderling. Deze sortering resulteert in een lijst met tabellen waarin een tabel die afhankelijk is van een andere tabel altijd later in de lijst voortkomt dan de tabel waarvan hij afhankelijk is. Dus lijst bij een 1 op N relatie zal er dus uitzien als X, Y. Hierdoor is de eerste controle op afhankelijkheden dus de locatie van beide tabellen in de gesorteerde lijst.

5.4.4 Mutaties

Een relatie heeft altijd een aantal regels met betrekking tot mutaties. Zo kan het zijn dat een relatie wel of niet verplicht is. Al deze regels voor een relatie hebben betrekking op mutaties hiervan. Mutaties zijn te verdelen in drie delen deze zullen apart worden besproken. Deze regels worden ingevoerd in de ADD Invoermodule en zijn opvraagbaar doormiddel van de DataDictionary-component.

Toevoegingen

Bij het toevoegen van een record aan een gerelateerde tabel, zal het nodig zijn om de sleutel van het gerelateerde record ook toe te voegen. Dit echter alleen wanneer deze relatie 'Not Null' is, wat inhoud dat de relatie in een tabel verplicht is. Indien de relatie 'Null' is hoeft er niet verplicht een gerelateerde sleutel toegevoegd te worden. Dit is vooral belangrijk bij koppeltabellen. Omdat bij een verplichte relatie ook de sleutels van deze gerelateerde tabel moet worden toegevoegd.

Wijzigingen

De manier waarop er omgegaan wordt met wijzingen van een sleutel in een relatie is afhankelijk hoe deze is ingesteld in database. Dit kan zijn 'Cascading' wat inhoud dat de gewijzigde sleutel in de gerelateerde tabellen mee veranderd of 'Restricted' wat inhoud dat de wijziging niet mag indien er gegevens aan de sleutel zijn gekoppeld.

Verwijderingen

Voor verwijdering geldt eigenlijk hetzelfde als bij wijzigingen. Wanneer de relatie 'Cascading' is zullen bij verwijdering van een sleutel alle gerelateerde records ook verwijderd worden.

Wanneer de relatie 'Restricted' zal een verwijdering niet mogen wanneer er records aan de desbetreffende sleutel gekoppeld zijn. Dit geld echter alleen wanneer de relatie een 'Not Null' relatie is. Is een relatie 'Null' dan hoeft alleen de sleutel in de gerelateerde records verwijderd te worden in plaats van het gehele record.

6. REALISATIE

6.1 Constructie

Voor ieder schermelement dat als toepassing het toen en of wijzigen van data heeft, wordt een model, view en controller geconstrueerd. Het scherm zelf bevat een containervariant van zowel model, view als controller. In deze containers bevinden zich de models, views en controllers die zijn aangemaakt voor de schermelementen.

6.2 Events

Zoals eerder gezegd worden events afgehandeld door de controllers. Oorspronkelijk was het de bedoeling om elk event aan alle aanwezige controllers door te geven, om deze vervolgens zelf te laten bepalen of het event wel of niet relevant zou zijn voor de desbetreffende controller. Hier zit echter een groot nadeel aan. Iedere controller zal bij ieder event gewaarschuwd moeten worden wat erg veel onnodige overhead oplevert.

Om dit probleem op te lossen, is er gekozen voor een andere oplossing. Iedere controller 'abonneert' zich op de events die relevant voor hem zijn. Hierdoor is het niet nodig om elk event aan iedere controller door te geven. Verder hoeft een controller niet te bepalen of een event wel voor hem bedoeld is, omdat dit altijd het geval is.

6.2.1 Internet

Ook op het Internet wordt op de bovengenoemde manier met events omgegaan. Hierbij valt echter één kanttekening te plaatsen, dit door het 'stateless' karakter van een Internetapplicatie. Hierdoor is het bij het verversen van een scherm, bijvoorbeeld door het maken van een andere selectie, nodig om een controller opnieuw op de events te abonneren.

6.3 Data

Het bepalen van de gegevens die opgehaald dienen te worden vanuit de database, gebeurd op basis van de gecreëerde models. Afhankelijk van de aanwezige relaties (zie paragraaf 4.4) worden aan de op te halen gegevens extra criteria toegevoegd. Zo zal bij een relatie tussen relaties en contactpersonen alleen de contactpersonen van de geselecteerde relaties uit de database worden opgehaald en niet alle contactpersonen.

6.3.1 Wijziging selectie

Wanneer een selectie wijzigt zullen gerelateerde tabellen ook moeten wijzigen. Zo zal in het bovengenoemde voorbeeld bij het kiezen van een andere relatie de bijbehorende contactpersonen moeten wijzigen. Dit gebeurd door een 'signaal' van het model dat zijn selectie veranderd. Deze stuurt naar alle andere models een signaal dat hij veranderend is. Op basis hiervan kan elk model voor zichzelf bepalen of hij nieuwe gegevens dient op te halen of juist helemaal niets hoeft te doen.

7. RESULTATEN

In de afgelopen afstudeerperiode is veel gedaan en zijn er ook goede resultaten behaald. Deze resultaten zijn te verdelen in twee onderdelen. Er zijn namelijk aanpassingen gemaakt aan de bestaande ADD invoermodule en verder is er natuurlijk een complete toolkit ontwikkeld.

7.1 ADD Invoermodule

De bestaande invoermodule was alleen in staat om C++ code te genereren, dit omdat deze code alleen vanuit C++ applicaties aangesproken werd. In principe is het wel mogelijk om deze C++ code vanuit C# aan te spreken. Dit is binnen het bedrijf namelijk als tijdelijke oplossing gebruikt. Om deze reden is de codegenerator van de invoermodule uitgebreid, zodat deze ook in staat is om C# code te genereren.

7.2 ADD.NET Toolkit

Met de ADD.NET Toolkit is het nu mogelijk om op een generieke wijze applicaties te ontwikkelen die gebruik maken van één en dezelfde toolkit. Op dit moment is het creëren van schermen vrijwel het enige verschil tussen Internet en Windows-applicaties.

7.2.1 DataDictionary

Voor het uitlezen van de informatie die ingevoerd is met behulp van de invoermodule is er een component 'DataDictionary'. Deze component biedt op een objectgeoriënteerde wijze toegang tot deze informatie. Dit wil zeggen dat er voor de ingevoerde kolommen een collectie kolom-objecten beschikbaar is met daarin alle ingevoerde informatie over kolom. Zo is dit voor ieder type gedaan. Andere types zijn: tabellen, primary en foreign keys.

7.2.2 SQLCommand

Voor het ophalen en wegschrijven van gegevens uit een database is er de component 'SQLCommand' ontwikkeld. Deze component is in staat op basis van informatie uit de 'DataDictionary' een complete SQL query op te bouwen. Aan de hand van deze query wordt er voor iedere tabel de benodigde query's opgebouwd voor het invoegen, wijzigen en verwijderen van gegevens in de database. Hierdoor hoeft een applicatieprogrammeur zich niet meer bezig te houden met databasetoegang, maar kan hij zich volledig richten op de applicatie zelf.

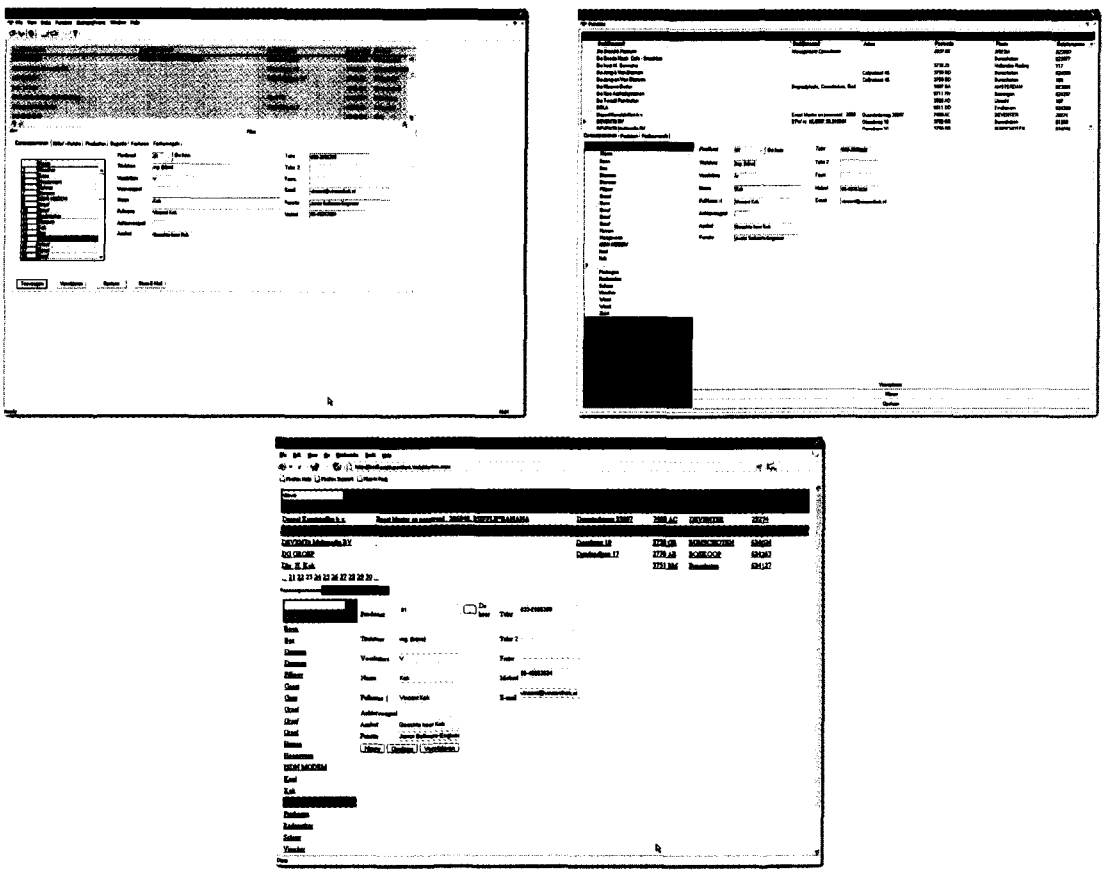
7.2.3 MVC

Voor het tonen van data op het scherm en het afhandelen van handelingen door de gebruiker is er een package met daarin een serie classes ontwikkeld die gebaseerd zijn op het MVC-pattern. Een volledig diagram is te zien in bijlage A. Om schermafhandeling voor zowel Internet als Windowsapplicaties mogelijk te maken, is de package ingedeeld in drie lagen. Zo is er een algemeen gedeelte die gebruikt kan worden voor zowel Internet als Windowsapplicaties, hier zijn alles 'Models' en de basis voor alle 'Views' en 'Controllers' vertegenwoordigd. Verder zijn er nog een serie specifieke 'Views' en 'Controllers' die geschikt zijn voor Internetapplicaties en een serie die geschikt zijn voor Windowsapplicaties.

7.3 Demoapplicaties

Zoals al eerder genoemd is er voor testdoeleinden een bestaande applicatie herschreven. Deze applicatie *Superklant* genaamd, wordt binnen het bedrijf gebruikt voor relatiebeheer. Met behulp van het programma worden naast de adresgegevens en contactpersonen van een klant ook bijgehouden welke producten een klant heeft, welke facturen er naar een klant zijn gestuurd of juist gestuurd moeten worden. Verder biedt het pakket nog veel meer functionaliteit, maar dit voert te ver om in dit document op in te gaan.

Doordat de toolkit ontwikkeling van zowel Windows als Internetapplicaties moet ondersteunen is er voor beide typen applicaties een demoversie geschreven. Met behulp van deze applicaties is het mogelijk geweest om de toolkit op functionaliteit te testen. Verder kunnen de applicaties in de toekomst als voorbeeld dienen. In de onderstaande afbeelding zijn respectievelijk de originele, de Windows en de Internetapplicatie afgebeeld.



Figuur 11: De bestaande, Windows en Web versie van Superklant

7.4 Handleiding voor de applicatieprogrammeur

Om de uiteindelijke gebruiker van de toolkit, de applicatieprogrammeur, te ondersteunen bij het ontwikkelen van een applicatie is er een handleiding geschreven. In de ze applicatie wordt op een puntsgewijze manier verteld hoe een applicatie met behulp van de toolkit kan worden geschreven. Deze handleiding is te vinden in bijlage C.

8. CONCLUSIES EN AANBEVELINGEN

8.1 Conclusies

Tijdens de afgelopen afstudeerperiode is er een toolkit ontwikkeld die in de nabije toekomst de applicatieprogrammeurs van DEVENTit B.V. moet ondersteunen bij het ontwikkelen van zowel Internet als Windowsapplicaties. Met behulp van de ontwikkelde toolkit en de bijbehorende ADD invoermodule is het nu mogelijk om op een uniforme manier applicaties te ontwikkelen voor zowel een Internet als Windows omgeving. Het is helaas niet gelukt om de gewenste DataCache-component te ontwerpen en te ontwikkelen. Dit kwam vooral doordat er meer tijd is besteed aan het ontwerp en ontwikkeling van het MVC gedeelte van het project. Met de uitbreiding van de ADD invoermodule is het nu mogelijk om code te genereren die specifiek is voor de nieuwe toolkit. Met het herschrijven van de relatiebeheerapplicatie is de toolkit direct aan een goede test onderworpen. Hierdoor zijn eerste kinderziektes al uit de toolkit verwijderd.

Kanttekening bij het resultaat dat behaald is dat de ontwikkeling van het resultaat alleen door mijzelf gedaan is. Hierdoor is het mogelijk dat er verkeerde of niet goed doordachte beslissingen zijn genomen die in een later stadium vervelende gevolgen kunnen hebben.

8.2 Aanbevelingen

De belangrijkste aanbeveling is het alsnog ontwerpen en ontwikkelen van de DataCache-component. Deze component moet het mogelijk maken om in de toekomst om te gaan met grote hoeveelheden data, zonder dat de eindgebruiker van een applicatie lang hoeft te wachten op de presentatie hiervan.

Ook is de ADD invoermodule incompleet. Zo zou het mogelijk moeten zijn om berekende kolommen aan een tabel toe te voegen. Deze kolommen bestaan niet werkelijk in de database maar bestaan uit een combinatie van waarden uit andere kolommen. Een voorbeeld hiervan is bijvoorbeeld het bedrag van een factuurregel. Dit bestaat vaak uit een vermenigvuldiging van het aantal en de prijs van een artikel. Daarnaast zijn de datatypes die gebruikt kunnen worden incompleet. Zo mist het type 'text' waarmee velden aangeduid worden die een grote hoeveelheid tekst bevatten.Verder dient het gegenereerde SQL-script, naast SQL Server ook geschikt te zijn voor Oracle databases.

Doordat het creëren van schermen het enige is wat nog verschilt tussen het ontwikkelen van Internet en Windowsapplicaties, verdiend een onderzoek naar XForms ten slotte nog een aanbeveling. XForms is een standaard van het World Wide Web Consortium die de indeling en inhoud van een scherm beschrijft doormiddel van XML. Onderzocht moet worden hoe deze standaard gebruikt kan worden voor zowel Internet als Windowsapplicaties om zo ook de schermbouw generiek te maken.

9. EVALUATIE

9.1 Project

Aan het eind van ieder project is het belangrijk dat er gekeken wordt naar het verloop van het project. Dit om zo van de eventueel gemaakte fouten te leren en de goede dingen in volgende projecten opnieuw toe te passen.

Het afstudeerproject dat ik de afgelopen vier maanden gedaan heb, is naar mijn mening goed verlopen. Het grootste probleem is toch wel de complexiteit van de opdracht geweest, waardoor de planning niet helemaal gehaald is. Door het uitlopen van het MVC gedeelte, is het niet meer gelukt om de DataCache-component te ontwikkelen. Dit is zeer jammer omdat dit zeer zeker een belangrijk onderdeel is van de toolkit.

Verder is het project goed verlopen, de overige doelstellingen zijn namelijk wel gehaald waardoor dit project toch wel als succesvol beschouwd mag worden.

9.2 Doel

Tijdens het afstuderen is het de bedoeling dat de afstudeerder laat zien dat hij of zij in een organisatie kan functioneren met de kennis die is opgedaan tijdens de studie. Dit is, naar mijn mening, zeker gelukt. Op een kleine misser, het niet kunnen ontwikkelen van de DataCache-component, na zijn de overige doelstellingen wel gehaald. Hierdoor heb ik bewezen dat ik staat ben om in een organisatie zelfstandig te functioneren. Het resultaat is echter wel mede tot stand gekomen door veel te overleggen met mijn bedrijfsbegeleider.

9.3 Persoonlijke ervaring

Over het algemeen heb ik de afgelopen afstudeerperiode als een leuke tijd ervaren. Er waren af en toe wel momenten dat het wat moeizamer ging. Dit was vooral tijdens het ontwerp, omdat dit af en toe vrij moeizaam ging door de complexiteit van de materie. Door goed overleg met de bedrijfsbegeleider is dit uiteindelijk zeker goed gekomen. Het feit dat ik bij dit bedrijf al een tijd parttime werk heb ik zeker als een voordeel ervaren. Dit omdat ik hierdoor als bekend was met de werkwijzen en gebruikte technieken binnen het bedrijf. Een goed voorbeeld hiervan is de ADD invoermodule. Het was namelijk een eis dat deze gebruikt kon blijven worden. Omdat ik met deze module al vaker had gewerkt, kon ik mij goed inleven in de situatie.

De sfeer binnen het bedrijf is zeer goed te noemen. Er heerst een informele sfeer en als student wordt je zeer zeker serieus genomen. Al met al heb ik een goede en een (nog steeds) leerzame tijd achter de rug.

10. LITERATUUR

Tijdens mijn afstuderen heb ik de volgende literatuur gebruikt, om over bepaalde zaken mijn kennis te verbreden. Dit bestaat uit zowel online bronnen als boeken.

10.1 Boeken

- Grady Booch, James Rumbauch en Ivar Jacobson: *The Unified Modeling Language User Guide*, Addison-Wesley, Upper Saddle River 1999
- Frank Buschmann, Regine Meunier, Hans Rohnert, Peter Sommerlad en Michael Stal: *Pattern-Oriented Software Architecture*, Wiley, Chichester 1996
- Erich Gamma, Richard Helm, Ralph Johnson en John Vlissides: *Design Patters Elements of Reusable Object-Oriented Software*, Addison-Wessley, Massachusetts 1995
- Ten Gevers en Tjerk Zijlstra: *Praktisch Projectmanagement*, Academic Service, Schoonhoven 2000
- Andrew Troelsen: *C# and the .NET Platform*, Apress, Berkeley 2001
- David Sceppa: *Microsoft ADO.NET*, Microsoft Press, Redmond 2002
- dr Olav Severijnen en drs. Rene Westbroek: *Netwerk basisboek Professionele Bedrijfscommunicatie*, Thieme, Zutphen 1999

10.2 Online

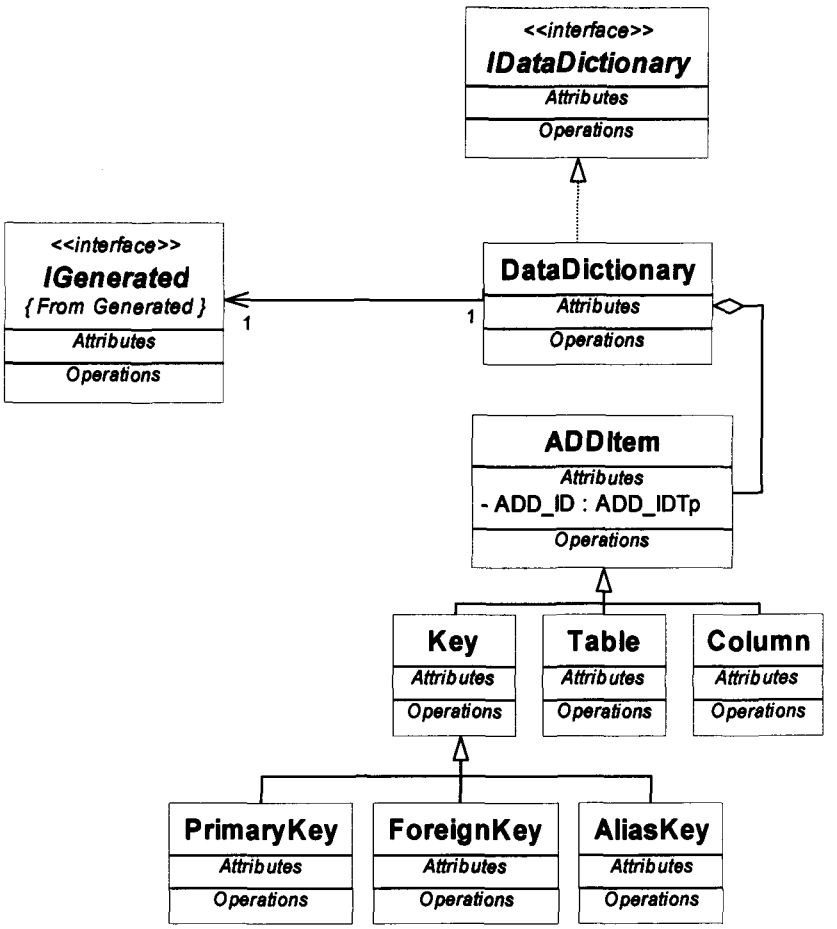
- .NET 247, http://www.dotnet247.com
- The Code Project, http://www.codeproject.com
- DevGuru Javascript Introduction, http://www.devguru.com/Technologies/ecmascript/quickref/javascript_intro.html
- Microsoft Patterns & Practices, http://www.microsoft.com/patterns
- MSDN Library Visual Studio 2003 Release
- World Wide Web Consortium, http://www.w3c.org

11. TERMINOLOGIE

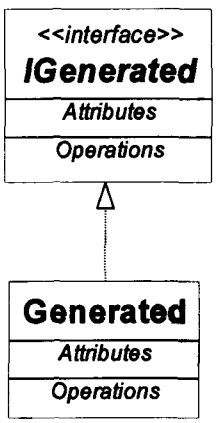
ADD	Algemene Data Dictionary. Programma voor het invoeren van een relationele model. T.b.v. het genereren van programmacode.
C#	Programmeertaal van Microsoft. Deze taal komt veel overeen met Java.
C++	Veel gebruikte objectgeoriënteerde programmeertaal.
Class	Verzameling van attributen en methoden in een objectgeoriënteerde programmeertaal.
Component	Op zichzelf staand softwareonderdeel
Control	Een element op het scherm. Bijvoorbeeld een knop.
Database	Software voor het gestructureerd opslaan van data.
DBMS	Database Management System. Zie verder database.
Design-pattern	objectgeorienteerde oplossing voor veelvoorkomende softwareproblemen.
Drag-and-Drop	Het slepen en plaatsen van elementen op het scherm d.m.v. de muis.
Interface	Structuur in een programmeertaal die een serie functies beschrijft.
Not Null	Verplicht.
Null	Niet verplicht.
Record	Item een database.
Package	Een verzameling functionaliteit in de vorm van classes.
SQL	Structured Query Language. Taal voor het opvragen van gegevens uit een database.
UML	Unified Modelling Language. Standaard voor het ontwerpen van objectgeoriënteerde software in de vorm van diagrammen.
Unit-testing	Het gestructureerd testen van componenten met behulp van een geschikte testsoftware.
User-interface	Het gedeelte van de software dat interactie heeft met de gebruiker.
W3C	World Wide Web Consortium. Organisatie die standaarden gericht op het World Wide Web beheerd.
XForms	Standaard die het uiterlijk en inhoud van een scherm beschrijft met hebulp van XML
XML	eXtended Markup Language. Taal voor het hiërarchisch structureren van data.

BIJLAGE A: UML SCHEMA'S

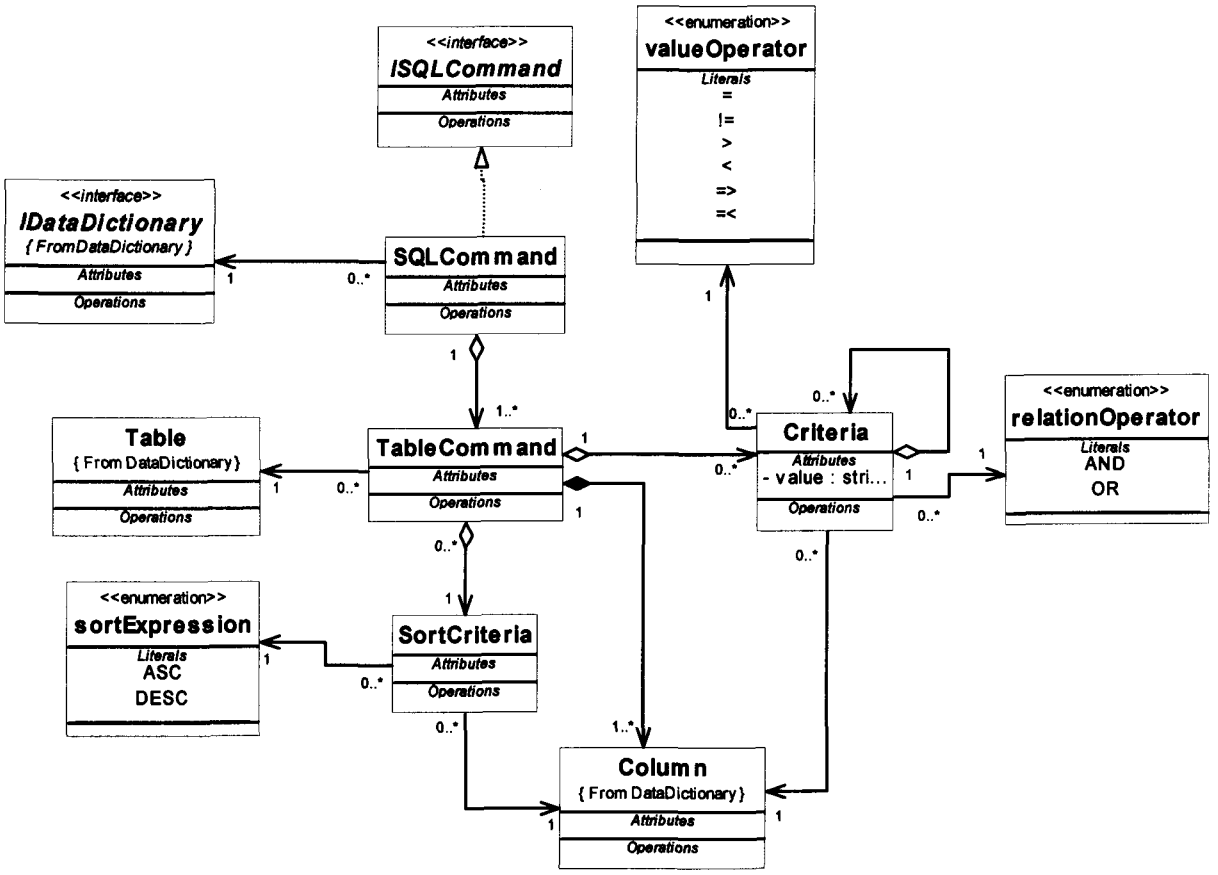
A.1 DataDictionary



A.2 Generated

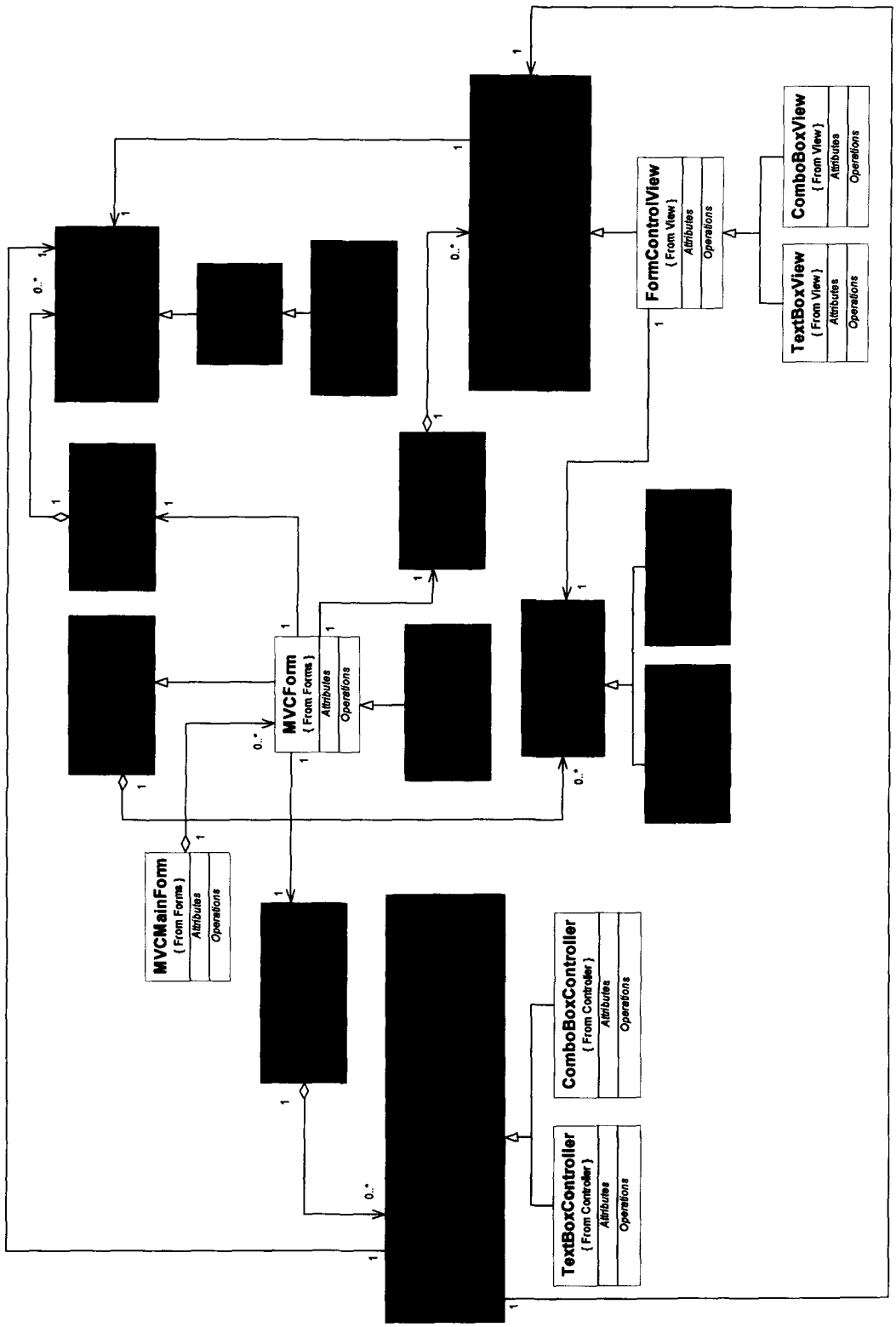


A.3 SQLCommand

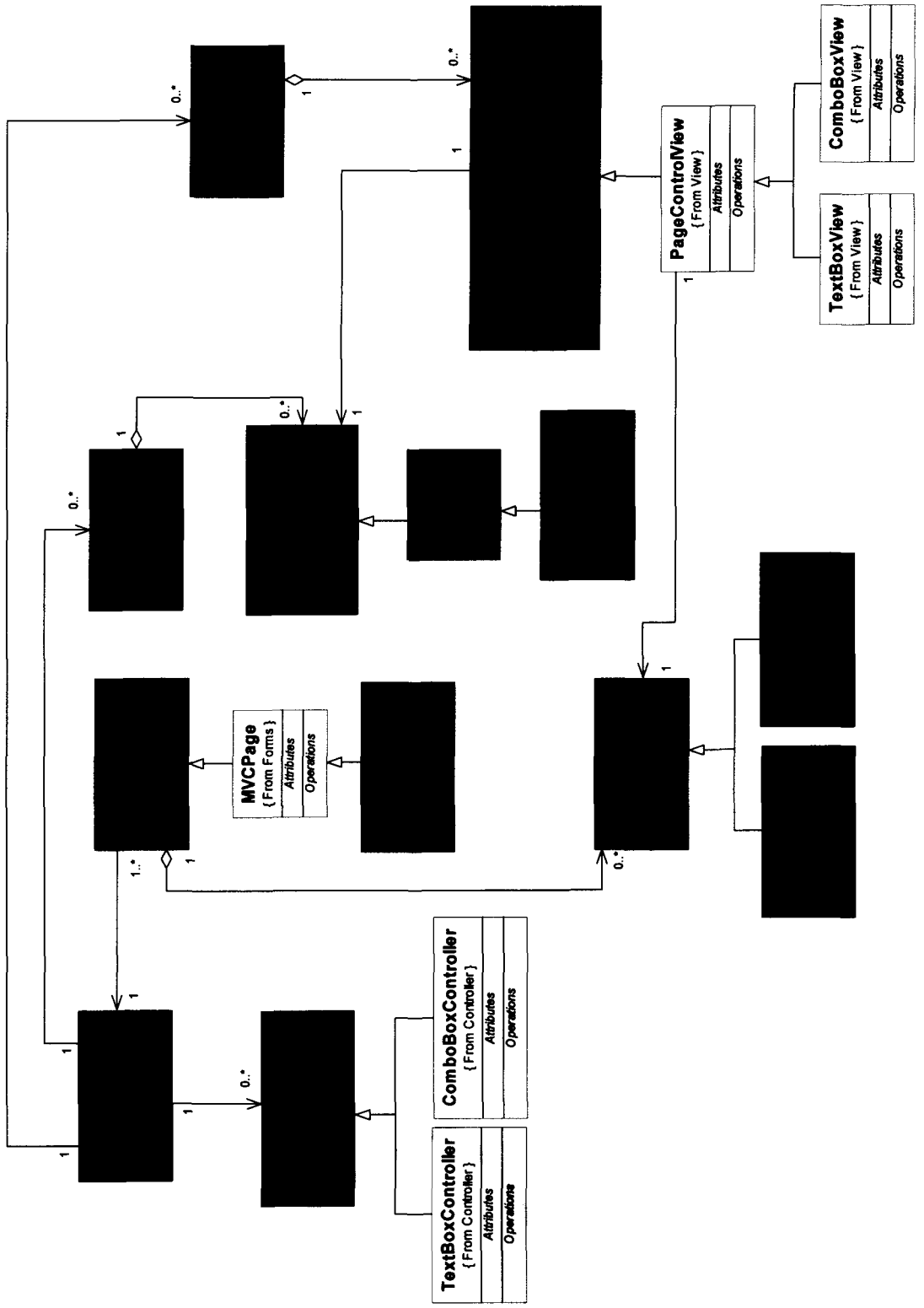


A.5 Legenda MVC

A.6 MVC Windows



A.7 MVC Internet



BIJLAGE B: PLAN VAN AANPAK

DEVENTit

Plan van aanpak

Bestemd voor: Hogeschool van Utrecht
Maarten Schnerr
Berkenweg 11
3818 LA Amersfoort
Nederland

Uitgebracht door: DEVENTit B.V.
Betrokkenen: Vincent Kok
Dieselweg 10, Bunschoten
Postbus 60
3750 GB Bunschoten

1. INLEIDING

Ter afsluiting van de opleiding Information Engineering, die gevolgd wordt aan de Hogeschool van Utrecht, vindt er een afstudeerproject plaats. Dit project is een 'testcase' om aan te tonen dat de student klaar is om zijn opgedane vaardigheden in de praktijk te brengen.

Voor dit project moet een plan opgesteld worden, waarin beschreven staat hoe het project wordt uitgevoerd. Dit plan staat in dit document beschreven.

Er wordt gestart met een korte beschrijving van het bedrijf, de aanleiding van het project en de huidige situatie.

Vervolgens zullen de opdracht, doelstellingen, het resultaat en de risico's beschreven.

Ten slotte zullen de fasering en het projectkader beschreven.

2. UITGANGSSITUATIE

2.1 Het bedrijf

DEVENTit B.V. is een softwarehuis dat zich richt op de ontwikkeling en het onderhoud van zowel software op maat als productsoftware voor financieel/ administratieve, multimedia, intranet en Internet toepassingen.

Op diverse gebieden heeft DEVENTit B.V. een vooraanstaande positie verworven. DEVENTit B.V is bijvoorbeeld met Atlantis, een platform voor multimediale archiefsystemen, marktleider voor gemeentelijke, regionale en landelijke archiefinstellingen.

Voor Exact Software Centers en Exact Dealers is DEVENTit B.V. de partner voor de realisatie van maatwerk aan Exact Software. De integratie van Internet binnen reguliere toepassingen of volledig op het Internet gebaseerde oplossingen heeft een sterk groeiend aandeel in de activiteiten van het bedrijf.

De basis voor de softwareontwikkeling is objectoriëntatie waardoor een hoge mate van flexibiliteit, hergebruik en robuustheid van de programmatuur wordt bereikt.

2.2 Aanleiding

Bij DEVENTit is er voor gekozen om nieuwe software te ontwikkelen met behulp van het Microsoft .NET Framework. Met dit framework is het mogelijk om met dezelfde programmeertaal en ontwikkelomgeving zowel traditionele Windows-applicaties als Internet-applicaties te ontwikkelen.

Om zowel traditionele als Windows-applicaties op een uniforme manier te kunnen ontwikkelen, zal er een toolkit moeten worden ontwikkeld die overweg kan met beide typen applicaties. Ook zal deze toolkit gebruik moeten maken van de faciliteiten die worden aangeboden door het Microsoft .NET Framework.

2.3 Huidige status

Bij DEVENTit wordt al gebruik gemaakt van een toolkit voor het ontwikkelen van Windows-applicaties. Deze toolkit, de Advisie Data Dictionary (ADD), is echter ontwikkeld voor het ontwikkelen van C++ applicaties en niet geschikt voor het gebruik voor internetapplicaties. Wel kunnen methodes en technieken die gebruikt worden in deze toolkit gebruikt worden voor nieuwe toolkit. Bij deze toolkit hoort ook een Windows-applicatie, waarin het mogelijk is om het relationele model van een applicatie in te voeren. Deze applicatie zal ook gebruikt worden voor de nieuwe toolkit en zal daarom dan aangepast moeten worden.

Voor het loskoppelen van de presentatielaag is door een collega al onderzoek gedaan. Dit loskoppelen is nodig om de toolkit geschikt te maken voor zowel Internet als Windows-applicaties, de schermafhandeling tussen deze typen applicaties is namelijk verschillend. Aan de hand van dit onderzoek is besloten om gebruik te maken van het Model-View-Controller (MVC) pattern. Dit pattern is speciaal ontwikkeld voor het loskoppelen van de presentatielaag van de andere lagen. Ook is er door deze collega een model gemaakt, met daarin een uitwerking van het MVC pattern in de context van de ontwikkelen toolkit.

Ten slotte is het belangrijk dat de toolkit integreert met de huidige ontwikkelomgeving. Deze ontwikkelomgeving maakt het namelijk mogelijk om op eenvoudige wijze schermen te maken volgens het 'drag-and-drop' principe. Deze functionaliteit moet te gebruiken zijn in combinatie met de toolkit.

2.4 Uitgangsdocumentatie

Het resultaat van het onderzoek dat gedaan is door een collega, is verwerkt in een document. In eerste instantie zal dit het belangrijkste uitgangspunt zijn. Daarnaast is er veel informatie over het MVC pattern zowel beschikbaar in aanwezige literatuur als op het Internet. Ook zullen veel ideeën uit de bestaande toolkit worden overgenomen. Over deze toolkit is echter geen documentatie aanwezig. Dit is verder geen probleem, de personen die deze toolkit ontwikkelt nog steeds bij het bedrijf werkzaam zijn.

3. PROJECTRESULTAAT

3.1 De opdracht

De opdracht bestaat uit het ontwikkelen van een toolkit, waarmee op een uniforme manier zowel Internet als Windows-applicaties ontwikkeld kunnen worden. Voor de toolkit zal er allereerst een ontwerp moeten worden gemaakt, dat verder bouwt op het al gedane onderzoek. Het ontwerp zal bestaan uit zowel een functioneel- als technisch ontwerp.

Binnen de opdracht bevindt zich nog een subopdracht. In de toolkit dient namelijk een databasecomponent te komen die om kan gaan met grote hoeveelheden data. De standaardcomponenten halen namelijk eerst alle data uit de database, voordat de gebruiker er mee kan werken. Bij grote hoeveelheden data resulteert dit echter in een onwerkbare situatie.

3.2 Doelstellingen

Het project heeft de volgende doelstellingen:

- Functioneel en technisch ontwerp van de toolkit.
- Implementatie van het ontwerp.
- Aangepaste ADD generator, zodat deze code kan genereren ten behoeve van de nieuwe toolkit.
- Toolkit dat gebruikt maakt van het MVC pattern
- Toolkit geschikt voor zowel Internet als Windows-applicaties.
- Databasecomponent die op de juiste manier omgaat met grote hoeveelheden data.
- Demo die het gebruik van de toolkit demonstreert, voor zowel internet- als Windows-applicaties.
- Documentatie voor de applicatieprogrammeur, zodat deze weet hoe hij de toolkit kan gebruiken.
- Afstudeerrapport, met daarin de bevindingen van het gehele traject.
- Presentatie voor de afstudeerzitting.

3.3 Resultaat

Het resultaat zal bestaan uit een serie componenten die gebruikt zullen worden bij het ontwikkelen van zowel Internet als Windows-applicaties. Deze serie componenten zullen voldoen aan zowel het functioneel- als technisch ontwerp. Verder zal de bestaande ADD generator aangepast zijn, zodat deze code en meta-informatie zal genereren waarvan de toolkit gebruik van zal maken.

Ook zal er een demoapplicatie worden geschreven die alle functionaliteit van de toolkit gebruikt. Het doel hiervan is de bruikbaarheid en de betrouwbaarheid van de toolkit te testen. Het document voor de applicatieprogrammeur, degene gebruik gaat maken van de toolkit, zal ook verwijzen naar deze demoapplicatie.

Verder zal er een afstudeerrapport worden geschreven met daarin een beschrijving van het gehele traject. Ten slotte zal er ook nog een presentatie worden gemaakt en gegeven met daarin een korten beschouwing van het traject en de resultaten.

3.4 Risicofactoren

Het project kent de volgende risicofactoren:

- Gebrek aan modelleerkennis. Tijdens de ontwerpfase zal er veel gemodelleerd moeten worden. Omdat het altijd lastig is om een op een bepaald niveau te modelleren, kan dit tijdens de ontwerpfase problemen opleveren. Dit probleem is echter goed op te vangen. Binnen het bedrijf is namelijk veel modelleerkennis aanwezig, zodat bij problemen altijd bij collega's om advies en hulp kan worden gevraagd.
- Complexiteit van de opdracht. Het project is best complex, daarom kan het moeilijk zijn om het overzicht te bewaren. Het is eigenlijk niet mogelijk om voor dit probleem een oplossing te vinden. Het is belangrijk dat er tijdens dit project gestructureerd gewerkt wordt, zodat hiermee het overzicht zoveel mogelijk bewaard blijft.
- Door langdurige ziekte kunnen de afgesproken doelstellingen niet gehaald worden. Dit probleem is niet te voorkomen, er is namelijk sprake van overmacht. Een mogelijke oplossing is het afzwakken van de doelstellingen, waardoor deze in de resterende tijd nog wel gehaald kunnen worden.
- Door te weinig overleg kan er verschil van gedachten ontstaan tussen de opdrachtgever en de uitvoerende. Hierdoor bestaat de mogelijkheid dat er niet aan de doelstellingen voldaan wordt. Om dit probleem op te vangen moet er gedurende het gehele project periodiek voortgangsoverleg plaatsvinden. Verder moet er bij enige twijfel bij de uitvoerende direct contact worden gezocht met de opdrachtgever om duidelijkheid te verkrijgen.

4. FASERING

4.1 Fase 1: Oriëntatie en planning

De eerste fase zal bestaan uit het vertrouwd raken met de opdracht. Ook zal de reeds aanwezige documentatie over deze materie worden bestudeerd. Wanneer er voldoende informatie bekend is over de opdracht, het verwachte resultaat en het tot stand komen van dit resultaat, zal er een Plan van Aanpak (dit document) worden geschreven met daarin onder andere de planning van het project.

- Producten: Plan van Aanpak
- Einddatum: 20 februari 2004

4.2 Fase 2: Ontwerp

Nu bekend is wat de precieze opdracht is en wat het resultaat moet zijn, kan er gestart worden met het ontwerp. Dit ontwerp bestaat uit twee delen, namelijk een functioneel en een technisch ontwerp. In het functioneel ontwerp zal worden beschreven wat de te ontwikkelen toolkit moet kunnen. In het technisch ontwerp zal juist worden beschreven hoe dit technisch gerealiseerd zal worden.

- Producten: Functioneel Ontwerp, Technisch Ontwerp
- Einddatum: 12 maart 2004

4.3 Fase 3: Ontwikkeling

Op basis van de specificatie die gemaakt is in de vorige fase zal er in deze fase een toolkit worden ontwikkeld. Naast de serie componenten die geschreven moeten worden, zal ook de huidige ADD generator moeten worden aangepast. Dit zodat deze applicatie de benodigde programmacode en meta-informatie voor de toolkit kan genereren op basis van de ingevoerde relationele data.

- Producent: ADD.NET toolkit, aangepaste ADD generator
- Einddatum: 16 april 2004

4.4 Fase 4: Databasecomponent

De toolkit is in de vorige fase afgerond en mist nu alleen nog een goed werkende database component. Deze component zal in deze fase worden ontwikkeld en wordt vervolgens geïntegreerd met de toolkit.

- Producten: Technisch ontwerp, Databasecomponent.
- Einddatum: 7 mei 2004

4.5 Fase 5: Testen en documenteren

In de vorige fase zijn de verschillende componenten natuurlijk al getest op hun functionaliteit. In deze fase zal het gebruik van de toolkit en de databasecomponent worden getest doormiddel van het maken van een testapplicatie die gebruik maakt van de volledige functionaliteit van de toolkit. Er zal zowel een Internet als Windows-applicatie worden gemaakt. Gelijktijdig zal worden gedocumenteerd hoe het schrijven van een applicatie met behulp van de toolkit in zijn

werk gaat. Dit zodat een applicatieprogrammeur snel gebruik kan maken van de toolkit. Mochten er tijdens deze fase nog onvolkomenheden voorkomen in de toolkit of de ADD generator, dan zullen deze worden opgelost.

- Producten: Demoapplicatie, Handleiding voor de applicatieprogrammeur
- Einddatum: 14 mei 2004

4.6 Fase 6: Voorbereiden afstudeerzitting

Als afsluiting van het afstuderen zal er een afstudeerzitting plaatsvinden. Voor de afstudeercommissie en andere belangstellenden zal er een presentatie gehouden worden. Verder zal het afstudeerrapport moeten worden verdedigd. In deze fase zal er daarom een presentatie worden gemaakt. Ook zal de verdediging zoveel mogelijk worden voorbereid door te anticiperen op vragen die gesteld kunnen worden.

- Producten: Presentatie
- Einddatum: Begin juni

4.7 Afstudeerrapport

Het schrijven van het afstudeerrapport is geen aparte fase. Gedurende de gehele afstudeerperiode zal er aan worden gewerkt. Natuurlijk zal er één à twee weken voor de inleverdatum extra aandacht aan worden besteedt, zodat het rapport aan alle eisen van school zal voldoen.

- Einddatum: Eind mei

5. PROJECTKADER

5.1 De uitvoerende

Dit project zal worden uitgevoerd door één 4e-jaars student Information Engineering aan de Hogeschool van Utrecht. Dit gebeurd in het kader van het afstuderen om de opleiding af te ronden.

De uitvoerende heeft al vaker in een projectverband gewerkt, zei het dan wel in een projectgroep van circa vier personen. De ervaring die de uitvoerende heeft opgedaan tijdens vorige projecten zal worden gebruikt om dit project succesvol te doen verlopen. Tevens zal dit project zijn kennis op het gebied van projectmatig werken, doen toenemen.

5.2 Voorwaarden aan de opdrachtgever

De eisen van de opdrachtgever zijn vertaald in een opdracht, doelstellingen en resultaat. Tijdens het project zal er dan ook alles aan worden gedaan om deze doelstellingen en het gewenste resultaat te realiseren. Tijdens het verloop van het project zal er continue contact zijn tussen de uitvoerende en de opdrachtgevers. Dit om eventuele misverstanden vroegtijdig te detecteren en op te lossen. Verder kan er aan bepaalde doelstellingen meer of minder prioriteit worden gegeven, zodat de belangrijkste punten wel gerealiseerd kunnen worden.

Een andere belangrijke voorwaarde is natuurlijk de tijd. Het gehele traject loopt van 9 febuari 2004 tot begin juni 2004

5.3 Voorwaarden aan derden

Tijdens dit project is er ook een derde partij. Deze partij bestaat uit de Hogeschool van Utrecht in de persoon van Maarten Schnerr die afstudeerbegeleider is. Deze persoon houdt namens de Hogeschool van Utrecht het verloop van het afstuderen in het oog.

5.4 Projectcommunicatie

Het contact tussen de uitvoerende en de opdrachtgever zal vooral mondeling verlopen. Dit omdat de kantoren van de verschillende betrokken zeer dicht bij elkaar gelegen zijn. Het contact met de derde partij, de Hogeschool van Utrecht, zal voornamelijk via e-mail verlopen. Verder zal deze derde partij één of meerdere keren de uitvoerende bezoeken op diens werkplek.

5.5 Faciliteiten en hulpmiddelen

De werkplek bestaat uit een krachtig werkstation. Deze machine is uitstekend geschikt om alle werkzaamheden uit te voeren. Voor de communicatie zijn Microsoft Office en Micrsoft Outlook geïnstalleerd. Voor het modelleren is de tool Embacadero Describe beschikbaar. Als ontwikkelomgeving is Microsoft.NET 2003 aanwezig. Voor technische documentatie is de MSDN Library aanwezig. Ten slotte beschikt het werkstation ook nog over een Internetaansluiting.

5.6 Procedures en richtlijnen

Omdat dit project door één persoon uitgevoerd wordt, zijn er geen afspraken tussen projectleden onderling. Het is echter wel van belang dat de afspraken tussen de uitvoerende, de opdrachtgever en de derde partij altijd nagekomen worden. Mocht een partij om een bepaalde reden zijn afspraak niet na kunnen komen moet deze dat vroegtijdig aan de andere partijen doorgeven.

Verder is het belangrijk dat de fasering met bijbehorende planning zoveel mogelijk wordt nageleefd en bewaakt. Dit zodat achteraf geen afspraken blijken te zijn die niet nagekomen zijn.

5.7 Betrokken personen

Het project wordt uitgevoerd door de volgende persoon:

Naam:	Vincent Kok
E-mail:	vkok@deventit.nl

De begeleiding vanuit DEVENTit verzorgt door:

Naam:	Drs. Ing. Peter van Diermen
E-mail:	pdiermen@deventit.nl

Deze beide personen zijn te bereiken op het volgende adres:

Bezoekadres:	Dieselweg 10 3752 LB Bunschoten-Spakenburg
Postadres:	Postbus 60 3750 GB Bunschoten-Spakenburg
Telefoon:	033-2992277

Vanuit de Hogeschool van Utrecht is de volgende begeleider aangewezen:

Naam:	Maarten Schnerr
Bezoekadres:	Berkenweg 11 3818 LA AMERSFOORT
Postadres:	Postbus 512 3800 AM Amersfoort
E-mail:	m.schnerr@fnt.hvu.nl
Telefoon:	033-4228904

BIJLAGE C: ADD.NET TOOLKIT TUTORIAL

ADD.NET Tutorial

Bestemd voor: DEVENTit B.V. Ontwikkelaars

Uitgebracht door: DEVENTit B.V.
Betrokkenen: Vincent Kok
Dieselweg 10, Bunschoten
Postbus 60
3750 GB Bunschoten

1. INLEIDING

Dit document is ervoor bedoeld om de applicatieprogrammeur een handvat te geven voor het werken met de ADD.NET toolkit. De toolkit bestaat uit de volgende onderdelen.

- ADD
- MVC
- Controls

Voordat er gebruik gemaakt kan worden van de toolkit moet deze uit SourceSafe worden gehaald. Het project heeft als naam: ADD_NET. En de 'working folder' moet worden ingesteld op P:\ADD_NET.

Na het compileren van een 'Release' versie is de toolkit klaar voor gebruik.

2. NIEUW PROJECT

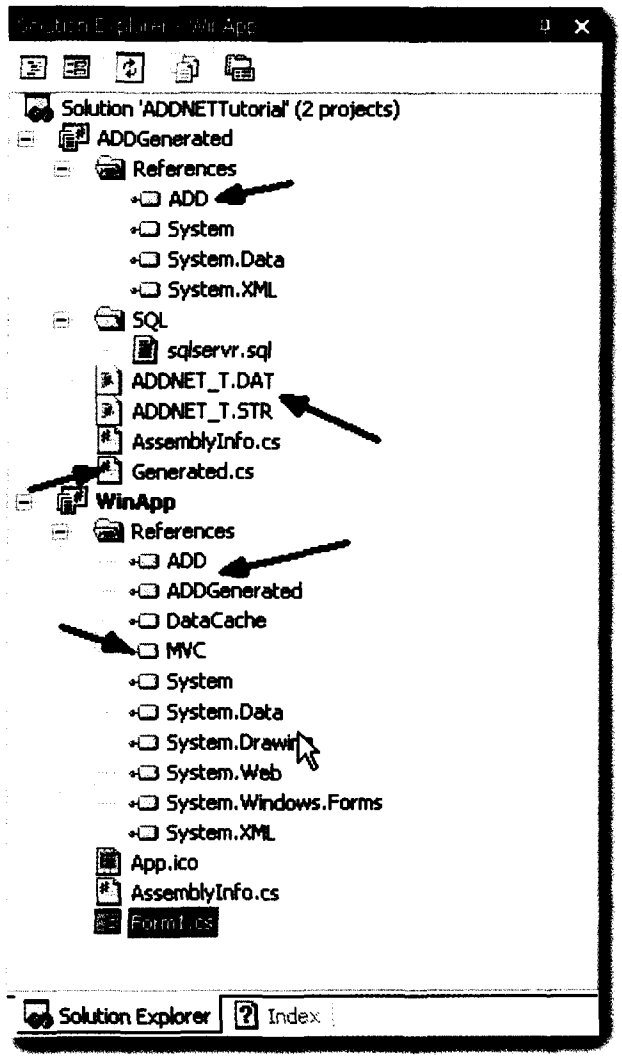
Na het aanmaken van een nieuw project, zal er een 'Reference' moet worden gemaakt met de toolkit. Dit kan door het wijzen naar de 'Release' versie van het bestand ADD.dll en de 'Release' versie van het bestand MVC.dll. Deze zijn respectievelijk te vinden in de map: 'P:\ADD_NET\ADD\bin\Release' en 'P:\ADD_NET\MVC\bin\Release'.

Omdat de ADD naast de DLL ook gebruik maakt van gegenereerde code, zal hiervoor ook een project voor moeten worden aangemaakt. Gebruik hiervoor een project van het type: 'Class Library' en noem deze: 'ADDGenerated'. Visual Studio zal automatisch een bestand met de naam Class1.cs aanmaken, deze kan direct verwijderd worden. Ook in dit project dient een verwijzing naar de 'ADD.DLL' te worden gemaakt.

De ADD invoermodule zal een bestand met de naam 'Generated.cs' genereren. Het genereren van de benodigde bestanden gebeurd met behulp van het aanroepen van de menuoptie: 'Generate – ADD.NET Source code and Run-Time data files'.

Om het project te kunnen compileren zal het bestand 'Generated.cs'worden toegevoegd aan het project: 'ADDGenerated'. In de ADD invoermodule zal moeten worden aangegeven, dat het bestand in de map van dit project moet worden gegenereerd. Voor het sql script is het verstandig om een subdirectory: 'SQL' te maken, zodat de SQL-scripts daarin kunnen worden geplaatst.

In de applicatie die gebouwd, in dit voorbeeld 'WinApp', wordt zal een 'Reference' naar het project 'ADDGenerated' gemaakt moeten worden. Dit kan door te kiezen voor 'Add Reference' en vervolgens op de tab 'Projects' het project 'ADDGenerated' te kiezen. Wanneer dit gedaan is, ziet het 'Project' eruit als in de onderstaande afbeelding.


Solution 'ADDNETTutorial' (2 projects)
ADDGenerated
References
ADD
System
System.Data
System.XML
SQL
sqlservr.sql
ADDNET_T.DAT
ADDNET_T.STR
AssemblyInfo.cs
Generated.cs
WinApp
References
ADD
ADDGenerated
DataCache
MVC
System
System.Data
System.Drawin
System.Web
System.Windows.Forms
System.XML
App.ico
AssemblyInfo.cs
Form1.cs
Solution Explorer
Index

3. ADD

De ADD component bestaat uit de volgende namespaces:

- DataDictionary
- SQLCommand
- Generated
- Tools

De namespaces kunnen worden gebruikt met behulp van het keyword 'using'. Hieronder een voorbeeld:

```
using ADD.DataDictionary;
using ADD.Generated;
using ADD.SQLCommand;
using ADD.Tools;
```

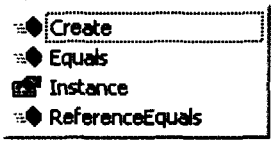
3.1 DataDictionary

De DataDictionary is een object van het type 'Singleton' en wordt aangemaakt door het aanroepen van de functie 'Create'. De functie 'Create' heeft als parameters de het pad naar de *.DAT en de *.STR bestanden en een instantie van de class 'ADDGenerated'. Vervolgens is de DataDictionary te benaderen via 'DataDictionary.Instance'. Hieronder een codevoorbeeld:

```
string ADDPATH = @"P:\ADDNETTutorial\ADDGenerated\";

DataDictionary.Create(new Generated(),ADDPATH);

DataDictionary.
```



Met behulp van de DataDictionary is het mogelijk om informatie die ingevoerd is in de ADD invoermodule op te vragen. Dit gebeurd op basis van ID's. Dit zijn een verzameling constanten die elke element in de DataDictionary uniek identificeren. De constanten zijn verzameld in de class 'ADD_IDS'. Hieronder een codevoorbeeld:

```
ADD_IDS.
    ADD_DATATYPES_EENINT
    ADD_DATATYPES_STRINGTP
    ADD_DATATYPES_TIMETP
    ADD_DATATYPES_UNSIGNEDTP
    ADD_ITEM
    ADD_ITEM_ITEMID
    ADD_ITEM_KEY_ITEMKEY
    ADD_ITEM_OMSCHRIJVING
    ADD_NEPTABEL
    ADD_NEPTABEL_NEPCOLUMN
```

3.2 SQLCommand

Met behulp van de class 'SQLCommand' kunnen queries worden opgebouwd en worden uitgevoerd. Een 'SQLCommand' kan worden opgebouwd met behulp van een array van ADD ID's van kolommen of tabellen. Eventueel kan er nog selectiecriteria en sortering worden meegegeven. Hieronder een voorbeeld:

```
string[] columns = {    ADD_IDS.ADD_ITEM_ITEMID,
                        ADD_IDS.ADD_ITEM_OMSCHRIJVING,
                        ADD_IDS.ADD_SUBITEM_SUBITEMID,
                        ADD_IDS.ADD_SUBITEM_OMSCHRIJVING
                    };

CriteriaCollection criteria = new CriteriaCollection();
criteria.AddCriteria(ADD_IDS.ADD_ITEM_ITEMID,"60",ValueOperator.Equals,RelationOperator.OR);
criteria.AddCriteria(ADD_IDS.ADD_ITEM_ITEMID,"70",ValueOperator.Equals,RelationOperator.OR);

SortCriteriaCollection sortCriteria = new SortCriteriaCollection();
sortCriteria.AddSortCriteria(ADD_IDS.ADD_ITEM_ITEMID,SortExpression.ASC);

SQLCommand command = new SQLCommand(dbConnection,columns,criteria,sortCriteria);
```

Dit resulteert in de volgende query's:

```
select
        ItemID, Omschrijving
from
        ITEM
where
        ItemID = 60   OR   ItemID = 70
order by
        ItemID ASC
```

```
select
        SubItemID, ItemID, Omschrijving
from
        SUBITEM
where
        SUBITEM.ItemID = 60  OR  SUBITEM.ItemID = 70
```

Naast deze query's wordt er voor iedere tabel ook een INSERT, UPDATE en DELETE statement opgebouwd, zodat de tabel 'vanzelf' kan worden weggeschreven naar de database. De query's worden per tabel opgebouwd, doordat een DataSet ook met meerdere tabellen, in plaats van een platgeslagen resultaat, werkt.

Met behulp van de functie 'Fill' kan er vervolgens een dataset worden gevuld. Het wegschrijven van data naar de database kan zowel met als zonder een transactie. Hiervoor bestaan de functies 'Update' en 'UpdateTransacted'. In het onderstaande codevoorbeeld worden de functies gebruikt.

```
DataSet ds = new DataSet();
//Vullen van de dataset.
command.Fill(ds);

// Code voor het bewerken van de data.

//Update zonder transactie.
command.Update(ds);

//Update met transactie
command.UpdateTransacted(ds);

//Opvragen volledige query
string sql = command.ToSQL();
```

Ten slotte ondersteund is het ook mogelijk om een volledige query over meerdere tabellen op te vragen met behulp van de functie 'ToSQL' (zie bovenstaande voorbeeld). Dit resulteert in de volgende query:

```
select
        ITEM.ItemID ,
        ITEM.Omschrijving ,
        SUBITEM.SubItemID ,
        SUBITEM.ItemID ,
        SUBITEM.Omschrijving
from
        ITEM ,
        SUBITEM
where
        ITEM.ItemID = SUBITEM.ItemID  AND
        (
                ITEM.ItemID = 60  OR
                ITEM.ItemID = 70
        )
order by
        ITEM.ItemID  ASC
```

4. MVC

De MVC.dll is opgedeeld in twee namespaces te weten:

- MVC.Windows
- MVC.Web

Deze worden respectievelijk voor Windows of Internetapplicaties gebruikt.

4.1 Windowsapplicatie

Een Windowsapplicatie bestaat bijna altijd uit twee delen. Een mainframe met één of meerder child-windows. Hiervoor zijn er in de MVC.Windows namespace twee classes beschikbaar. Te weten:

- MVCMainForm
- MVCForm

4.1.1 MVCMainForm

Deze class functioneert als MainForm voor de applicatie. Van deze class moet één Windows Form van worden afgeleidt die dient als MainForm. Dit ziet er als volgt uit:

```
public class MainForm : MVCMainForm
```

In deze Form worden alle childforms van een applicatie getoond. Verder bevat deze class een aantal virtuele functies die overridden dienen te worden voor het goed functioneren van de applicatie. Dit zijn de volgende functies:

- ADDPath
  Moet het pad naar de *.DAT en *.STR bestanden retourneren.
- GeneratedCode
  Deze functie retourneert een instantie van de gegenereerde code, oftwel een instantie van ADDGenerated.

- DBConnection
  Dient een instantie van OleDBConnection te retourneren. Om een toegang tot de database te verkrijgen.

In code ziet dit er als volgt uit:

```
protected override string ADDPath
{
    get { return  @"P:\ADD_NET\ADDGenerated\";  }
}

protected override IGenerated GeneratedCode
{
    get { return new Generated();}
}

protected override OleDbConnection DBConnection
{
    get
    {
        OleDbConnection c = new OleDbConnection("Provider=SQLOLEDB;
        c.Open();
        return c;
    }
}
```

4.1.2 MVCForm

Het MVCForm zorgt voor gehele afhandeling van het MVC-proces. Een Windows Form dient dus te worden afgeleidt van deze class. Dit ziet er als volgt uit:

```
public class ChildForm : MVCForm
```

Bij het laden van het scherm wordt voor iedere control op het scherm een MVC-trio aangemaakt. Het is mogelijk om in dit proces in te grijpen door het overridden van een van de volgende functies:

- CreateModel
- CreateView
- CreateController

Deze functies roepen normaliter de bijbehorende factory aan voor het creëren van het juiste component. Door het overridden van een van deze functies is het mogelijk om een eigen variant van een Model, View of Controller te gebruiken voor een element op het scherm.

Voor het zetten van relaties dient de functie OnSetRelations overridden te worden. Vervolgens kan door het aanroepen van de functie SetRelation van de member m_models een relatie worden aangegeven. In het onderstaande voorbeeld wordt een relatie gelegd tussen relaties en contactpersonen en tussen relaties en facturen.

```
protected override void OnSetRelations()
{
    m_models.SetRelation(ADD_IDS.ADD_RELATIES,ADD_IDS.ADD_CONTACTPERSONEN);
    m_models.SetRelation(ADD_IDS.ADD_RELATIES,ADD_IDS.ADD_FACTUREN);
}
```

Ten slotte is bestaat er nog de functie OnSelectionChanged. Deze wordt aangeroepen wanneer er een selectie veranderd in het model. Een voorbeeld van het gebruik van deze functie is aan of uitzetten van bepaalde menuitems bij het wijzigen van de selectie.

4.2 Internetapplicatie

Net als in de MVC.Windows namespace bevinden zich in MVC.Web namespace twee classes die als basis dienen voor nieuw te ontwikkelen applicaties. Te weten:

- HttpADDApplication
- MVCWebForm

4.2.1 HttpADDApplication

Voor een Internetapplicatie is er niet echt een hoofdscherm aan te wijzen. Om de benodigde initialisatie toch uit te voeren, is er de class HttpApplication. Deze class bevat de functie OnApplicationStart die wordt aangeroepen op het moment dat een Internetapplicatie voor de eerste maal wordt aangeroepen. Van deze class is een afgeleide gemaakt in de vorm van de class HttpADDApplication.

Net als de class MVCMainForm bevat deze class de virtuele functies GeneratedCode, ADDPath en DBConnection (zie paragraaf 4.1.1) die nodig zijn om de DataDictionary te initialiseren. Deze class dient gebruikt te worden door de class Global van een webapplicatie hiervan af te leiden. Hieronder een codevoorbeeld:

```
public class Global : HttpADDApplication
```

4.2.2 MVCWebForm

Voor het afhandelen van MVC voor een WebForm is er de class MVCWebForm. Deze biedt dezelfde functionaliteit als het MVCForm voor Windows-applicaties. Ook in deze class kunnen de functies CreateModel, CreateView, CreateController, OnSetRelations en OnSelectionChanged overridden worden. Een WebForm moet dus worden afgeleidt van een MVCWebForm, dit ziet er als volgt uit:

```
public class WebForm1 : MVC.Web.MVCWebForm
```

4.2.3 Validatie

In een Windows-applicatie is de validatie automatisch geregeld. In een Webapplicatie kan dit echter niet. Om toch validatie te hebben moet er een CustomValidator voor ieder control op het scherm geplaatst worden. De CustomValidator moet vervolgens worden toegekend aan een control. Vanaf dit punt is de toolkit 'slim' genoeg om verdere afhandeling te regelen.

5. CONTROLS

Voor zowel Internet als Windows zijn er een tweetal controls gemaakt die samenwerken met de DataDictionary. Deze zijn gebaseerd op bestaande controls, en zijn ervoor bedoeld om te voorkomen dat veel voorkomende zaken steeds opnieuw moeten worden gedaan.

De controls bestaan uit een grid en een control voor het tonen van data uit een andere tabel. Deze laatste bestaat uit een combinatie van een textbox voor de code, een knop voor het oproepen van een selectielijst en een label voor het tonen van de omschrijving.

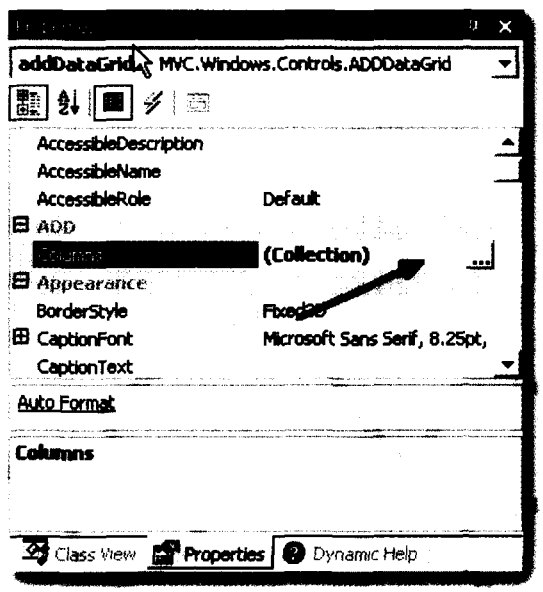
5.1 Gebruik

Voordat van de controls gebruik kan worden gemaakt, zullen deze geregistreerd moeten worden in Visual Studio. Dit kan door op de 'Toolbox' met rechtermuisknop te klikken en dan te kiezen voor 'Add/Remove Items'. Door vervolgens voor 'Browse' te kiezen en MVC.dll te selecteren wordt toolbox uitgebreid met een viertal controls. Dit zijn tweemaal een control met de naam 'ADDDatagrid' en tweemaal een control met de naam 'FKeyControl'. Afhankelijk van het feit dat er gewerkt wordt aan een WebForm of een Windows Form zullen er altijd maar twee te gebruiken zijn. Dit omdat er van beide controls zowel een Internet als een Windowsversie zijn.

5.2 Windowsapplicatie

5.2.1 ADDDataGrid

Een grid toont de data uit een tabel. Om aan te geven welke tabel, zal deze daarom het ADD ID van de tabel krijgen. Het bepalen van de getoonde kolommen gaat via de property 'Columns'.



Via deze property is het mogelijk om aan te geven welke kolommen er in de tabel komen te staan. Verder kunnen per kolom een aantal instellingen worden gedaan. Hierbij valt te denken aan breedte, header, read-only etc. Verder is er een optie 'Fill-up'. Wanneer voor een kolom deze optie op 'true' staat dan zal alle overgebleven ruimte in de grid aan deze kolom worden toegekend.

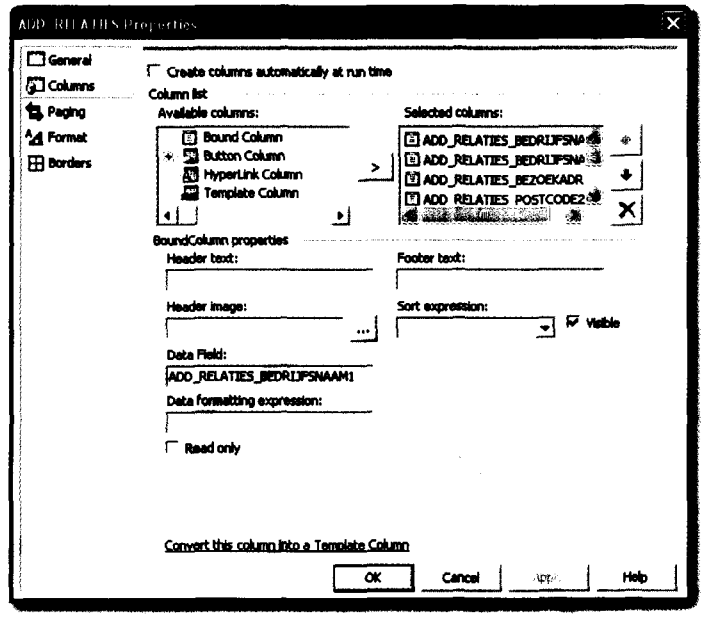
5.2.2 FKeyControl

Voor het tonen van data uit gerelateerde tabellen is er de FKeyControl. Deze toont de sleutel die wordt opgeslagen in de tabel met de omschrijving daarachter. Ook bevat de control een knop voor het oproepen van een selectielijstje. In de onderstaande afbeelding wordt de control gebruikt voor het tonen van een predikaat van een contactpersoon. Het enige wat hiervoor hoeft gedaan te worden, is aan het control het juiste ADD ID toe te kennen.

5.3 Internetapplicatie

5.3.1 ADDDatagrid

Net als de Windows-variant is het ID van de grid het ADD ID van de tabel waaruit gegevens getoond worden. Daarnaast dienen ook de kolommen aangegeven te worden die getoond moeten worden in de grid. Dit gaat eveneens met behulp van de columns property. Met behulp van deze property is het onderstaande scherm op te roepen. Door vervolgens één of meerdere 'Bound Columns' te voegen en aan de 'Data field property' vervolgens het juiste ADD ID toe te kennen zal de grid de juiste data tonen.



Daarnaast heeft deze control nog een aantal extra property's om zijn gedrag te kunnen beïnvloeden.

- GenerateLookUpField
  Indien deze op 'true' wordt gezet, zal er boven de grid een extra textbox worden gegeneerd waarmee het mogelijk is om te zoeken naar data in de grid.
- GenerateSelectColumns
  Indien deze op 'true' wordt gezet, zullen de waarden in de grid als hyperlink worden getoond waardoor het mogelijk is om een rij in de grid te selecteren.

5.3.2 FKeyControl

De FkeyControl voor het Internet werkt hetzelfde als degene voor Windows. Deze control heeft echter een aantal extra property's.

- ImageURL
  De url naar het image-bestand dat gebruikt wordt als knop.
- PopUpPage
  De url naar het *.aspx bestand die gebruikt wordt als selectielijst. Normaal gesproken zal hiervoor de reeds bestaande PopUp.aspx voor worden gebruikt maar het mogelijk dat de applicatieprogrammeur een eigen versie wil gebruiken.
- PopUpHeight
  De hoogte van de popup in pixels.
- PopUpWidth
  De breedte van de popup in pixels.